# Package: flexsdm (via r-universe)

October 25, 2026

**Type** Package

**Title** Flexible Species Distribution Modeling Workflow

**Version** 1.4.2

**Maintainer** Santiago J.E. Velazco <sjevelazco@gmail.com>

**URL** https://sjevelazco.github.io/flexsdm/,
https://github.com/sjevelazco/flexsdm

**BugReports** https://github.com/sjevelazco/flexsdm/issues

**Depends** R (>= 3.5.0)

**Imports** doParallel, dplyr, foreach, gbm, grDevices, kernlab, maxnet, mgcv, nnet, randomForest, spThin, terra (>= 1.5-21), methods, utils, ggplot2, patchwork, Rcpp

**Description** Provides a flexible and comprehensive workflow for species distribution modeling, from data preparation (occurrence filtering, calibration areas, pseudo-absence and background sampling, data partitioning, and collinearity reduction) to model fitting, tuning, evaluation, prediction, ensemble modeling, and post-processing. The package is described in Velazco et al. (2022) <doi:10.1111/2041-210X.13874>. Implemented methods include the Shape metric to measure model extrapolation of Velazco et al. (2023) <doi:10.1111/ecog.06992>, ensembles of small models of Breiner et al. (2015) <doi:10.1111/2041-210X.12403> and Breiner et al. (2018) <doi:10.1111/2041-210X.12957>, Gaussian process models of Golding and Purse (2016) <doi:10.1111/2041-210X.12523>, and occurrence-based constraints for post-processing predictions of Mendes et al. (2020) <doi:10.1016/j.ecolmodel.2020.109180>.

**License** GPL (>= 3)

**Encoding** UTF-8

**LazyData** true

**Suggests** covr, fs, knitr, rmarkdown, testthat (>= 3.0.0), tibble, tidyr

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**LinkingTo** Rcpp

**Config/roxygen2/version** 8.1.0

**Config/pak/sysreqs** libgdal-dev gdal-bin libgeos-dev libproj-dev libsqlite3-dev

**Repository** https://sjevelazco.r-universe.dev

**Date/Publication** 2026-10-14 19:57:49 UTC

**RemoteUrl** https://github.com/sjevelazco/flexsdm

**RemoteRef** HEAD

**RemoteSha** f6635814e3387feef27a725e004c0e16d8e2d23d

# Contents

abies . . . 3
backg . . . 4
calib_area . . . 5
correct_colinvar . . . 8
data_bpdp . . . 14
data_pdp . . . 17
env_outliers . . . 19
esm_gam . . . 23
esm_gau . . . 26
esm_gbm . . . 29
esm_glm . . . 32
esm_max . . . 35
esm_net . . . 39
esm_svm . . . 42
extra_eval . . . 45
extra_truncate . . . 49
fit_dom . . . 51
fit_ensemble . . . 55
fit_gam . . . 58
fit_gau . . . 62
fit_gbm . . . 65
fit_glm . . . 68
fit_max . . . 71
fit_net . . . 75
fit_raf . . . 78
fit_svm . . . 81
get_absences . . . 84
get_block . . . 86
hespero . . . 87
homogenize_na . . . 88
interp . . . 89
map_env_dist . . . 91

msdm_posteriori . . . 94
msdm_priori . . . 100
occ_reproj . . . 103
occfilt_env . . . 105
occfilt_geo . . . 108
occfilt_select . . . 113
p_bpdp . . . 116
p_extra . . . 120
p_pdp . . . 126
palms . . . 130
part_random . . . 131
part_sband . . . 133
part_sblock . . . 137
part_senv . . . 143
plot_res . . . 146
sample_background . . . 147
sample_pseudoabs . . . 153
sdm_directory . . . 159
sdm_eval . . . 161
sdm_extract . . . 165
sdm_predict . . . 166
sdm_summarize . . . 171
sdm_uncertainty . . . 173
sdm_varimp . . . 176
spp . . . 180
tune_gbm . . . 181
tune_max . . . 184
tune_net . . . 188
tune_raf . . . 191
tune_svm . . . 194

**Index** **199**

---

`abies` *A data set containing localities and environmental condition of an Abies (fir tree) species in California, USA*

---

## Description

A data set containing localities and environmental condition of an Abies (fir tree) species in California, USA

## Usage

```
abies
```

### Format

A tibble object with 5000 rows and 10 variables:

**ID** presences and absences records ID

**pr_ab** presence and absences denoted by 1 and 0 respectively

**x y** columns with coordinates in Albers Equal Area Conic coordinate system

**from column aet to landform** columns with values for environmental variables at each locality ...

### Examples

```
require(dplyr)
data("abies")
abies
```

---

## backg — *A data set containing environmental conditions of background points*

---

### Description

A data set containing environmental conditions of background points

### Usage

```
backg
```

### Format

A tibble object with 5000 rows and 10 variables:

**pr_ab** background point denoted by 0

**x y** columns with geographical coordinates

**from column aet to landform** columns with values of environmental variables at coordinate locations ...

### Examples

```
require(dplyr)
data("backg")
backg
```

`calib_area` *Delimit calibration area for constructing species distribution models*

## Description

This function offers different methods to define the calibration area. The output could be used with other flexsdm functions like sample_backgroud, sample_pseudoabs, and sdm_predict, among others

## Usage

```
calib_area(data, x, y, method, groups = NULL, crs = NULL)
```

## Arguments

`data` data.frame or tibble. Database with presences

`x` character. Column name with longitude data

`y` character. Column name with latitude data

`method` character. Method used for delimiting a calibration area. Could be necessary to concatenate (c()) different objects for this argument. The following methods are implemented:

- buffer: calibration area is defined by a buffer around presences. Usage method = c('buffer', width=40000). A value of the buffer width in m must be provided if CRS has a longitude/latitude, or in map units in other cases
- mcp: calibration area is defined by a minimum convex polygon. Usage method = 'mcp'.
- bmcp: calibration area is defined by buffered minimum convex polygon with buffer width. Usage method = c('bmcp', width=40000). A value of the buffer width in m must be provided if CRS has a longitude/latitude, or in map units in other cases
- mask: calibration area is defined by selected polygons in a spatial vector object intersected by presences. Usage method = c("mask", clusters, "DN"). The second concatenated element must be a SpatVector, the third element is a character with the column name from SpatVector used for filtering polygons.

`groups` character. Column name indicating differentiated subsets of points. This could be used with mcp and bmcp method. Default NULL

`crs` character. Coordinate reference system used for transforming occurrences and outputs. If set as NULL, the result of mask method will have the same crs as the SpatVector used. Define a crs is mandatory for buffer, mcp and bmcp method.

## Value

A SpatVector

## Examples

```
data("spp")
sp1 <- spp[spp$species == "sp1" & spp$pr_ab == 1, ]
clusters <- terra::vect(system.file("external/clusters.shp", package = "flexsdm"))

# Calibration area: buffer of 40 km around the records
ca_toy <- calib_area(
  data = sp1,
  x = "x",
  y = "y",
  method = c("buffer", width = 40000),
  crs = terra::crs(clusters)
)
ca_toy


require(terra)
require(dplyr)
data("spp")
clusters <- system.file("external/clusters.shp", package = "flexsdm")
clusters <- terra::vect(clusters)


single_spp <-
  spp %>%
  dplyr::filter(species == "sp1") %>%
  dplyr::filter(pr_ab == 1) %>%
  dplyr::select(-pr_ab)


plot(clusters)
points(single_spp[-1], col = "red")
crs(clusters, proj = TRUE) # coordinate reference system (CRS) used for this points database
# note that the unit of this CRS is in m, consequently the buffer width
# will be interpreted in m too

# buffer method
ca_1 <- calib_area(
  data = single_spp,
  x = "x",
  y = "y",
  method = c("buffer", width = 40000),
  crs = crs(clusters)
)
plot(ca_1)
points(single_spp[, 2:3], pch = 19, cex = 0.5)

# mcp method
ca_2 <- calib_area(
  data = single_spp,
  x = "x",
  y = "y",
```

```
  method = "mcp",
  crs = crs(clusters)
)
plot(ca_2)
points(single_spp[, 2:3], pch = 19, cex = 0.5)

# mcp method for different groups
single_spp <- single_spp %>% mutate(groups = ifelse(x > 150000, "a", "b"))

plot(single_spp[, 2:3], pch = 19, col = "blue")
points(single_spp[single_spp$groups == "a", 2:3], col = "red", pch = 19)
points(single_spp[, 2:3])

ca_2.1 <- calib_area(
  data = single_spp,
  x = "x",
  y = "y",
  method = c("mcp"),
  crs = crs(clusters),
  groups = "groups"
)
plot(ca_2.1)
points(single_spp[, 2:3], pch = 19, cex = 0.5)

# bmcp method
ca_3 <- calib_area(
  data = single_spp,
  x = "x",
  y = "y",
  method = c("bmcp", width = 30000),
  crs = crs(clusters)
)
plot(ca_3)
points(single_spp[, 2:3], pch = 19, cex = 0.5)

# bmcp method for different groups
ca_3.1 <- calib_area(
  data = single_spp,
  x = "x",
  y = "y",
  method = c("bmcp", width = 30000),
  crs = crs(clusters),
  groups = "groups"
)
plot(ca_3.1)
points(single_spp[, 2:3], pch = 19, cex = 0.5)

# mask method
plot(clusters)
names(clusters)

ca_3.1 <- calib_area(
  data = single_spp,
```

```
  x = "x",
  y = "y",
  method = c("mask", clusters, "clusters"),
)
plot(ca_3.1)
points(single_spp[, 2:3], pch = 19, cex = 0.5, col = "red")
```

---

`correct_colinvar` *Collinearity reduction of predictor variables*

---

## Description

Collinearity reduction of predictor variables

## Usage

```
correct_colinvar(
  env_layer,
  method,
  proj = NULL,
  save_proj = NULL,
  restric_to_region = NULL,
  restric_pca_proj = FALSE,
  maxcell = NULL,
  based_on_points = FALSE,
  data = NULL,
  x = NULL,
  y = NULL
)
```

## Arguments

`env_layer` SpatRaster An object of class SpatRaster containing the predictors. This function does not allow categorical variables

`method` character. Collinearity reduction method. It is necessary to provide a vector for this argument. The next methods are implemented:

- pearson: Highlights correlated variables according to Pearson correlation. A threshold of maximum correlation must be specified. Otherwise, a threshold of 0.7 is defined as default. Usage method = c('pearson', th='0.7').
- vif: Select variables by Variance Inflation Factor, a threshold can be specified by user. Otherwise, a threshold of 10 is defined as default.Usage method = c('vif', th = '10').
- pca: Perform a Principal Component Analysis and use the principal components as the new predictors. The selected components account for 95% of the whole variation in the system. Usage method = c('pca').

- fa: Perform a Factorial Analysis and select, from the original predictors, the number of factors is defined by Broken-Stick and variables with the highest correlation to the factors are selected. Usage method = c('fa').

`proj` character. Only used for pca method. Path to a folder that contains sub-folders for the different projection scenarios. Variables names must have the same names as in the raster used in env_layer argument. Usage proj = "C:/User/Desktop/Projections" (see in Details more about the use of this argument)

`save_proj` character. Directory to save PCA projection. If NULL (default), the projections are saved in a folder named 'Projection_PCA' within the temporary directory of the R session (see `tempdir`); use this argument to keep them in a folder of your choice. The path is returned in the 'proj' element of the output.

`restric_to_region` SpatVector. Area used to restrict cells of env_layer at moment to perform collinearity reduction. Default: NULL.

`restric_pca_proj` logical. Area used to restrict geographically PCA projection within SpatVector used in restric_to_region. Only use for PCA analysis. Default: FALSE.

`maxcell` numeric. Number of raster cells to be randomly sampled. Taking a sample could be useful to reduce memory usage for large rasters. If NULL, the function will use all raster cells. Default NULL. Usage maxcell = 50000.

`based_on_points` logical. If TRUE, collinearity reduction method will be based on species points data (i.e., presences, and absences, pseudo-absences or background points). If TRUE, data, x and y arguments must be provided. Default FALSE.

`data` tibble or data.frame. Database with species data used to model (i.e., presence + absence, or presence + pseudo-absence + background points) with x and y coordinates

`x` character. Column name with spatial x coordinates

`y` character. Column name with spatial y coordinates

## Details

In the case of having environmental variables for the current conditions and other time periods (future or present), it is recommended to perform the PCA analysis with the current environmental condition and project the PCA for the other time periods. To do so, it is necessary to use "proj" argument. Path to a folder (e.g., projections) that contains sub-folders for the different projection scenarios (e.g., years and emissions). Within each sub-folder must be stored single or multiband rasters with the environmental variables.

For example:

C:/Users/my_pc/projections/
|– MRIESM_2050_ssp126
| |– var1.tif
| |– var2.tif
| '– var3.tif
|– MRIESM_2080_ssp585
| |– var1.tif

| |– var2.tif
| '– var3.tif
'– UKESM_2050_ssp370
|– var1.tif
|– var2.tif
'– var3.tif

If pca method is run with time projections, correct_colinvar function will create the Projection_PCA folder (in the folder given in the save_proj argument, or in the temporary directory if save_proj is NULL; the exact path is in the proj element returned by the function) with the same system of sub-folders and multiband raster with the principal components (pcs.tif)

C:/Users/my_pc/Projection_PCA/
|– MRIESM_2050_ssp126
| '– pcs.tif # a multiband tif with principal components
|– MRIESM_2080_ssp585
| '– pcs.tif
'– UKESM_2050_ssp370
'– pcs.tif

Perform collinearity reduction based on points

Evaluating collinearity based on all environmental conditions of a calibration area or study area could yield different results than evaluating collinearity based on points used to construct the models. If you want to perform collinearity reduction based on species points data, it is strongly recommended to use all the point data used for modeling (i.e., presence + absence or presence + pseudo-absence/background points).

## Value

#' If 'pearson', returns a list with the following elements:

- cor_table: a matrix object with pairwise correlation values of the environmental variables
- cor_variables: a list object with the same length of the number of environmental values containing the pairwise relations that exceeded the correlation threshold for each one of the environmental variables

If 'vif' method, returns a list with the following elements:

- env_layer: a SpatRaster object with selected environmental variables
- removed_variables: a character vector with removed environmental variables
- vif_table: a data frame with VIF values for all environmental variables

If 'pca' method, returns a list with the following elements:

- env_layer: SpatRaster with scores of selected principal component (PC) that sum up 95% of the whole variation or original environmental variables
- coefficients: a matrix with the coefficient of principal component (PC) for predictors
- cumulative_variance: a tibble with the cumulative variance explained in selected principal component (PC)

If 'fa' method, returns a list with the following elements:

- env_layer: SpatRaster with scores of selected variables due to correlation to factors.
- number_factors: number of factors selected according to the Broken-Stick criteria,
- removed_variables: removed variables,
- uniqueness: uniqueness of each environmental variable according to the factorial analysis,
- loadings: environmental variables loadings in each of the chosen factors

## Examples

```
# Toy raster: coarser resolution to keep this example fast
somevar <- terra::rast(system.file("external/somevar.tif", package = "flexsdm"))
somevar <- terra::aggregate(somevar, fact = 6)

# Perform Pearson collinearity control
var <- correct_colinvar(env_layer = somevar, method = c("pearson", th = "0.7"))
var$cor_table
var$cor_variables

# Perform VIF collinearity control
var <- correct_colinvar(env_layer = somevar, method = c("vif", th = "8"))
var$vif_table


require(terra)
require(dplyr)

somevar <- system.file("external/somevar.tif", package = "flexsdm")
somevar <- terra::rast(somevar)

# Perform pearson collinearity control
var <- correct_colinvar(env_layer = somevar, method = c("pearson", th = "0.7"))
var$cor_table
var$cor_variables

# For all correct_colinvar methods it is possible to take a sample or raster to reduce memory
var <- correct_colinvar(env_layer = somevar, method = c("pearson", th = "0.7"), maxcell = 10000)
var$cor_table
var$cor_variables

# Perform vif collinearity control
var <- correct_colinvar(env_layer = somevar, method = c("vif", th = "8"))
var$env_layer
var$removed_variables
var$vif_table

# Perform pca collinearity control
var <- correct_colinvar(env_layer = somevar, method = c("pca"))
plot(var$env_layer)
var$env_layer
var$coefficients
var$cumulative_variance
```

```
# Perform pca collinearity control with different projections
## Below will be created a set of folders to simulate the structure of the
## directory where environmental variables are stored for different scenarios
dir_sc <- file.path(tempdir(), "projections")
dir.create(dir_sc)
dir_sc <- file.path(dir_sc, c("scenario_1", "scenario_2"))
sapply(dir_sc, dir.create)

somevar <-
  system.file("external/somevar.tif", package = "flexsdm")
somevar <- terra::rast(somevar)

terra::writeRaster(somevar, file.path(dir_sc[1], "somevar.tif"), overwrite = TRUE)
terra::writeRaster(somevar, file.path(dir_sc[2], "somevar.tif"), overwrite = TRUE)

## Perform pca with projections
dir_w_proj <- dirname(dir_sc[1])
dir_w_proj
var <- correct_colinvar(env_layer = somevar, method = "pca", proj = dir_w_proj)
var$env_layer
var$coefficients
var$cumulative_variance
var$proj


# Perform fa colinearity control
var <- correct_colinvar(env_layer = somevar, method = c("fa"))
var$env_layer
var$number_factors
var$removed_variables
var$uniqueness
var$loadings

## %#######################################################%##
#                                                           #
####            Other option to perform PCA             ####
####      considering cell restricted to a region       ####
#                                                           #
## %#######################################################%##
data("abies")

# Define a calibration area
abies2 <- abies %>%
  dplyr::select(x, y, pr_ab) %>%
  dplyr::filter(pr_ab == 1)

plot(somevar[[1]])
points(abies2[-3])
ca <- calib_area(abies2, x = "x", y = "y", method = c("mcp"), crs = crs(somevar))
plot(ca, add = TRUE)

# Full geographical range to perform PCA
```

```
pca_fr <- correct_colinvar(
  env_layer = somevar,
  method = c("pca"),
  maxcell = NULL,
  restric_to_region = NULL,
  restric_pca_proj = FALSE
)

# Perform PCA only with cell delimited by polygon used in restric_to_region
pca_rr <- correct_colinvar(
  env_layer = somevar,
  method = c("pca"),
  maxcell = NULL,
  restric_to_region = ca,
  restric_pca_proj = FALSE
)

# Perform and predicted PCA only with cell delimited by polygon used in restric_to_region
pca_rrp <- correct_colinvar(
  env_layer = somevar,
  method = c("pca"),
  maxcell = NULL,
  restric_to_region = ca,
  restric_pca_proj = TRUE
)

plot(pca_fr$env_layer) # PCA with all cells
plot(pca_rr$env_layer) # PCA with calibration area cell but predicted for entire region
plot(pca_rrp$env_layer) # PCA performed and predicted for cells within calibration area (ca)


## %#######################################################%##
#                                                           #
####       Use correct_colinvar with points data        ####
#                                                           #
## %#######################################################%##
data("abies")

# Presence-absence database
abies2 <- abies %>%
  dplyr::select(x, y, pr_ab)

# Perform collinearity control
# Pearson
correct_colinvar(
  env_layer = somevar,
  method = c("pearson", th = "0.6"),
  based_on_points = TRUE,
  data = abies2,
  x = "x",
  y = "y"
)
```

```
# VIF
correct_colinvar(
  env_layer = somevar,
  method = c("vif", th = "8"),
  based_on_points = TRUE,
  data = abies2,
  x = "x",
  y = "y"
)

# PCA
correct_colinvar(
  env_layer = somevar,
  method = c("pca"),
  based_on_points = TRUE,
  data = abies2,
  x = "x",
  y = "y"
)

# FA
correct_colinvar(
  env_layer = somevar,
  method = "fa",
  based_on_points = TRUE,
  data = abies2,
  x = "x",
  y = "y"
)
```

---

`data_bpdp` *Calculate data to construct partial dependence surface plots*

---

## Description

Calculate data to construct Partial dependence surface plot (i.e., bivariate dependence plot) for two predictor set

## Usage

```
data_bpdp(
  model,
  predictors,
  resolution = 50,
  training_data = NULL,
  training_boundaries = NULL,
  projection_data = NULL,
  clamping = FALSE
)
```

### Arguments

`model` A model object of class "gam", "gbm", "glm", "graf", "ksvm", "ksvm", "maxnet", "nnet", and "randomForest" This model can be found in the first element of the list returned by any function from the fit_, tune_, or esm_ function families

`predictors` character. Vector with two predictor name(s) to plot. If NULL all predictors will be plotted. Default NULL

`resolution` numeric. Number of equally spaced points at which to predict continuous predictors. Default 50

`training_data` data.frame. Database with response (0,1) and predictor values used to fit a model. Default NULL

`training_boundaries` character. Plot training conditions boundaries based on training data (i.e., presences, presences and absences, etc). If training_boundaries = "convexh", function will delimit training environmental region based on a convex-hull. If training_boundaries = "rectangle", function will delimit training environmental region based on four straight lines. If used any methods it is necessary provide data in training_data argument. If NULL all predictors will be used. Default NULL.

`projection_data` SpatRaster. Raster layer with environmental variables used for model projection. Default NULL

`clamping` logical. Perform clamping. Only for maxent models. Default FALSE

### Value

A list with two tibbles "pdpdata" and "resid".

- pspdata: has data to construct partial dependence surface plot, the first two column includes values of the selected environmental variables, the third column with predicted suitability.
- training_boundaries: has data to plot boundaries of training data.

### See Also

data_pdp, p_bpdp, p_pdp

### Examples

```
# Toy data: a small subset of 'abies' to keep this example fast
data("abies")
abies_toy <- abies[c(1:30, 701:730), ]
abies_toy <- part_random(
  data = abies_toy,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

glm_toy <- fit_glm(
  data = abies_toy,
```

```
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin"),
  partition = ".part",
  thr = "max_sens_spec"
)

bpdp_toy <- data_bpdp(
  model = glm_toy$model,
  predictors = c("aet", "cwd"),
  resolution = 10,
  training_boundaries = "rectangle",
  training_data = abies_toy
)
names(bpdp_toy)
bpdp_toy$pspdata


library(terra)
library(dplyr)

somevar <- system.file("external/somevar.tif", package = "flexsdm")
somevar <- terra::rast(somevar) # environmental data
names(somevar) <- c("aet", "cwd", "tmx", "tmn")
data(abies)

abies2 <- abies %>%
  select(x, y, pr_ab)

abies2 <- sdm_extract(abies2,
  x = "x",
  y = "y",
  env_layer = somevar
)
abies2 <- part_random(abies2,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 5)
)

m <- fit_svm(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmx", "tmn"),
  partition = ".part",
  thr = c("max_sens_spec")
)

df <- data_bpdp(
  model = m$model,
  predictors = c("aet", "cwd"),
  resolution = 50,
  projection_data = somevar,
  training_boundaries = "rectangle",
  training_data = abies2,
```

```
  clamping = TRUE
)

df
names(df)
df$pspdata
df$training_boundaries

# see p_bpdp to construct partial dependence plot with ggplot2
```

---

data_pdp *Calculate data to construct partial dependence plots*

---

## Description

Calculate data to construct partial dependence plots for a given predictor

## Usage

```
data_pdp(
  model,
  predictors,
  resolution = 50,
  resid = FALSE,
  training_data = NULL,
  projection_data = NULL,
  clamping = FALSE
)
```

## Arguments

| | |
|---|---|
| model | A model object of class "gam", "gbm", "glm", "graf", "ksvm", "ksvm", "maxnet", "nnet", and "randomForest" This model can be found in the first element of the list returned by any function from the fit_, tune_, or esm_ function families |
| predictors | character. Vector with a predictor name. |
| resolution | numeric. Number of equally spaced points at which to predict continuous predictors. Default 50 |
| resid | logical. Calculate residuals based on training data. Default FALSE |
| training_data | data.frame. Database with response (0,1) and predictor values used to fit a model. Default NULL |
| projection_data | SpatRaster. Raster layer with environmental variables used for model projection. When this argument is used, function will calculate partial dependence curves distinguishing conditions used in training and projection conditions (i.e., projection data present in projection area but not training). Default NULL |
| clamping | logical. Perform clamping. Only for maxent models. Default FALSE |

## Value

A list with two tibbles "pdpdata" and "resid".

- pdpdata: has data to construct partial dependence plots, the first column includes values of the selected environmental variable, the second column with predicted suitability, and the third column with range type, with two values Training and Projecting, referring to suitability calculated within and outside the range of training conditions. Third column is only returned if "projection_data" argument is used
- resid: has data to plot residuals. The first column includes values of the selected environmental variable and the second column with predicted suitability.

## See Also

data_bpdp, p_bpdp, p_pdp

## Examples

```
# Toy data: a small subset of 'abies' to keep this example fast
data("abies")
abies_toy <- abies[c(1:30, 701:730), ]
abies_toy <- part_random(
  data = abies_toy,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

glm_toy <- fit_glm(
  data = abies_toy,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin"),
  partition = ".part",
  thr = "max_sens_spec"
)

pdp_toy <- data_pdp(
  model = glm_toy$model,
  predictors = "aet",
  resolution = 10,
  training_data = abies_toy
)
pdp_toy


library(terra)
library(dplyr)

somevar <- system.file("external/somevar.tif", package = "flexsdm")
somevar <- terra::rast(somevar) # environmental data
names(somevar) <- c("aet", "cwd", "tmx", "tmn")
data(abies)
```

```
abies2 <- abies %>%
  select(x, y, pr_ab)

abies2 <- sdm_extract(abies2,
  x = "x",
  y = "y",
  env_layer = somevar
)
abies2 <- part_random(abies2,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 5)
)

svm_t1 <- fit_svm(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmx", "tmn"),
  partition = ".part",
  thr = c("max_sens_spec")
)

df <- data_pdp(
  model = svm_t1$model,
  predictors = c("aet"),
  resolution = 100,
  resid = TRUE,
  projection_data = somevar,
  training_data = abies2,
  clamping = FALSE
)

df
names(df)
df$pdpdata
df$resid

plot(df$pdpdata[1:2], type = "l")
points(df$resid[1:2], cex = 0.5)

# see p_pdp to construct partial dependence plot with ggplot2
```

---

`env_outliers` *Integration of outliers detection methods in environmental space*

---

## Description

This function performs different methods for detecting outliers in species distribution data based on the environmental conditions of occurrences. Some methods need presence and absence data (e.g.

Two-class Support Vector Machine and Random Forest) while other only use presences (e.g. Reverse Jackknife, Box-plot, and Random Forest outliers) . Outlier detection can be a useful procedure in occurrence data cleaning (Chapman 2005, Liu et al., 2018).

## Usage

```
env_outliers(data, x, y, pr_ab, id, env_layer)
```

## Arguments

| | |
|---|---|
| data | data.frame or tibble with presence (or presence-absence) records, and coordinates |
| x | character. Column name with longitude data. |
| y | character. Column name with latitude data. |
| pr_ab | character. Column name with presence and absence data (i.e. 1 and 0) |
| id | character. Column name with row id. Each row (record) must have its own unique code. |
| env_layer | SpatRaster. Raster with environmental variables |

## Details

This function will apply outliers detection methods to occurrence data. Box-plot and Reverse Jackknife method will test outliers for each variable individually, if an occurrence behaves as an outlier for at least one variable it will be highlighted as an outlier. If the user uses only presence data, Support Vector Machine and Random Forest Methods will not be performed. Support Vector Machine and Random Forest are performed with default hyper-parameter values. In the case of a species with < 7 occurrences, the function will not perform any methods (i.e. the additional columns will have 0 values); nonetheless, it will return a tibble with the additional columns with 0 and 1. For further information about these methods, see Chapman (2005), Liu et al. (2018), and Velazco et al. (2022).

## Value

A tibble object with the same database used in 'data' argument and with seven additional columns, where 1 and 0 denote that a presence was detected or not as outliers

- .out_bxpt: outliers detected with Box-plot method
- .out_jack: outliers detected with Reverse Jackknife method
- .out_svm: outliers detected with Support Vector Machine method
- .out_rf: outliers detected with Random Forest method
- .out_rfout: outliers detected with Random Forest Outliers method
- .out_sum: frequency of a presences records was detected as outliers based on the previews methods (values between 0 and 6).

## References


- Chapman, A. D. (2005). Principles and methods of data cleaning: Primary Species and Species- Occurrence Data. version 1.0. Report for the Global Biodiversity Information Facility, Copenhagen. p72. http://www.gbif.org/document/80528
- Liu, C., White, M., & Newell, G. (2018). Detecting outliers in species distribution data. Journal of Biogeography, 45(1), 164 - 176. https://doi.org/10.1111/jbi.13122
- Velazco, S.J.E.; Bedrij, N.A.; Keller, H.A.; Rojas, J.L.; Ribeiro, B.R.; De Marco, P. (2022) Quantifying the role of protected areas for safeguarding the uses of biodiversity. Biological Conservation, xx(xx) xx-xx. https://doi.org/10.1016/j.biocon.2022.109525


## Examples

```
# Toy data: small raster and records of one species to keep this example fast
data("spp")
somevar <- terra::rast(system.file("external/somevar.tif", package = "flexsdm"))
somevar <- terra::aggregate(somevar, fact = 6) # coarser resolution
sp3 <- spp[spp$species == "sp3", ]
sp3$idd <- seq_len(nrow(sp3)) # unique id for each record

outs <- env_outliers(
  data = sp3,
  pr_ab = "pr_ab",
  x = "x",
  y = "y",
  id = "idd",
  env_layer = somevar
)
outs


require(dplyr)
require(terra)
require(ggplot2)

# Environmental variables
somevar <- system.file("external/somevar.tif", package = "flexsdm")
somevar <- terra::rast(somevar)

# Species occurrences
data("spp")
spp
spp1 <- spp %>% dplyr::filter(species == "sp1")

somevar[[1]] %>% plot()
points(spp1 %>% filter(pr_ab == 1) %>% select(x, y), col = "blue", pch = 19)
points(spp1 %>% filter(pr_ab == 0) %>% select(x, y), col = "red", cex = 0.5)

spp1 <- spp1 %>% mutate(idd = 1:nrow(spp1))

# Detect outliers
outs_1 <- env_outliers(
```

```
  data = spp1,
  pr_ab = "pr_ab",
  x = "x",
  y = "y",
  id = "idd",
  env_layer = somevar
)

# How many outliers were detected by different methods?
out_pa <- outs_1 %>%
  dplyr::select(starts_with("."), -.out_sum) %>%
  apply(., 2, function(x) sum(x, na.rm = TRUE))
out_pa

# How many outliers were detected by the sum of different methods?
outs_1 %>%
  dplyr::group_by(.out_sum) %>%
  dplyr::count()

# Let explor where are locate records highlighted as outliers
outs_1 %>%
  dplyr::filter(pr_ab == 1, .out_sum > 0) %>%
  ggplot(aes(x, y)) +
  geom_point(aes(col = factor(.out_sum))) +
  facet_wrap(. ~ factor(.out_sum))

# Detect outliers only with presences
outs_2 <- env_outliers(
  data = spp1 %>% dplyr::filter(pr_ab == 1),
  pr_ab = "pr_ab",
  x = "x",
  y = "y",
  id = "idd",
  env_layer = somevar
)

# How many outliers were detected by different methods
out_p <- outs_2 %>%
  dplyr::select(starts_with("."), -.out_sum) %>%
  apply(., 2, function(x) sum(x, na.rm = TRUE))

# How many outliers were detected by the sum of different methods?
outs_2 %>%
  dplyr::group_by(.out_sum) %>%
  dplyr::count()

# Let explor where are locate records highlighted as outliers
outs_2 %>%
  dplyr::filter(pr_ab == 1, .out_sum > 0) %>%
  ggplot(aes(x, y)) +
  geom_point(aes(col = factor(.out_sum))) +
  facet_wrap(. ~ factor(.out_sum))
```

```
# Comparison of function outputs when using it with
# presences-absences or only presences data.

bind_rows(out_p, out_pa)
# Because the second case only were used presences, outliers methods
# based in Random Forest (.out_rf) and Support Vector Machines (.out_svm)
# were not performed.
```

---

esm_gam *Fit and validate Generalized Additive Models based on Ensembles of Small Models approach*

---

## Description

This function constructs Generalized Additive Models using the Ensembles of Small Models (ESM) approach (Breiner et al., 2015, 2018).

## Usage

```
esm_gam(data, response, predictors, partition, thr = NULL, k = 2)
```

## Arguments

`data` data.frame. Database with the response (0,1) and predictors values.

`response` character. Column name with species absence-presence data (0,1)

`predictors` character. Vector with the column names of quantitative predictor variables (i.e. continuous variables). This function does not allow categorical variables and can only construct models with continuous variables. Usage predictors = c("aet", "cwd", "tmin").

`partition` character. Column name with training and validation partition groups.

`thr` character. Threshold used to get binary suitability values (i.e. 0,1). It is useful for threshold-dependent performance metrics. It is possible to use more than one threshold type. It is necessary to provide a vector for this argument. The following threshold criteria are available:

- equal_sens_spec: Threshold at which the sensitivity and specificity are equal.
- max_sens_spec: Threshold at which the sum of the sensitivity and specificity is the highest (aka threshold that maximizes the TSS).
- max_jaccard: The threshold at which the Jaccard index is the highest.
- max_sorensen: The threshold at which the Sorensen index is highest.
- max_fpb: The threshold at which FPB (F-measure on presence-background data) is highest.

- sensitivity: Threshold based on a specified sensitivity value. Usage thr = c('sensitivity', sens='0.6') or thr = c('sensitivity'). 'sens' refers to sensitivity value. If a sensitivity value is not specified, the default value is 0.9.

If the user wants to include more than one threshold type, it is necessary to concatenate threshold types, e.g., thr=c('max_sens_spec', 'max_jaccard'), or thr=c('max_sens_spec', 'sensitivity', sens='0.8'), or thr=c('max_sens_spec', 'sensitivity'). Function will use all thresholds if no threshold is specified

k integer. The dimension of the basis used to represent the smooth term. Default 2. Because ESM was proposed to fit models with little data, we recommend using small values of this parameter.

## Details

This method consists of creating bivariate models with all pair-wise combinations of predictors and perform an ensemble based on the average of suitability weighted by Somers' D metric (D = 2 x (AUC -0.5)). ESM is recommended for modeling species with few occurrences. This function does not allow categorical variables because the use of these types of variables could be problematic when using with few occurrences. For further detail see Breiner et al. (2015, 2018).

This function fits GAM using mgvc package, with Binomial distribution family and thin plate regression spline as a smoothing basis (see ?mgvc::s).

## Value

A list object with:

- esm_model: A list with "gam" class object from mgcv package for each bivariate model. This object can be used for predicting an ensemble of small models with the `sdm_predict` function.
- predictors: A tibble with variables use for modeling.
- performance: Performance metrics (see `sdm_eval`). Threshold dependent metrics are calculated based on the threshold specified in the argument.
- performance_part: Performance metric for each replica and partition (see `sdm_eval`).

## References


- Breiner, F. T., Guisan, A., Bergamini, A., & Nobis, M. P. (2015). Overcoming limitations of modelling rare species by using ensembles of small models. Methods in Ecology and Evolution, 6(10), 1210-218. https://doi.org/10.1111/2041-210X.12403
- Breiner, F. T., Nobis, M. P., Bergamini, A., & Guisan, A. (2018). Optimizing ensembles of small models for predicting the distribution of species with few occurrences. Methods in Ecology and Evolution, 9(4), 802-808. https://doi.org/10.1111/2041-210X.12957


## See Also

`esm_gau`, `esm_gbm`, `esm_glm`, `esm_max`, `esm_net`, and `esm_svm`.

## Examples

```
# Toy data: a small subset of 'abies' to keep this example fast
data("abies")
abies_toy <- abies[c(1:30, 701:730), ]
abies_toy <- part_random(
  data = abies_toy,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

esm_toy <- esm_gam(
  data = abies_toy,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin"),
  partition = ".part",
  thr = NULL,
  k = 3
)
esm_toy$performance


data("abies")
require(dplyr)

# Using k-fold partition method
set.seed(10)
abies2 <- abies %>%
  na.omit() %>%
  group_by(pr_ab) %>%
  dplyr::slice_sample(n = 10) %>%
  group_by()

abies2 <- part_random(
  data = abies2,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)
abies2

# Without threshold specification and with kfold
esm_gam_t1 <- esm_gam(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin", "ppt_djf", "ppt_jja", "pH", "awc", "depth"),
  partition = ".part",
  thr = NULL
)

esm_gam_t1$esm_model # bivariate model
esm_gam_t1$predictors
esm_gam_t1$performance
esm_gam_t1$performance_part
```

```
# Test with rep_kfold partition
abies2 <- abies2 %>% select(-starts_with("."))

set.seed(10)
abies2 <- part_random(
  data = abies2,
  pr_ab = "pr_ab",
  method = c(method = "rep_kfold", folds = 3, replicates = 2)
)
abies2

esm_gam_t2 <- esm_gam(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin", "ppt_djf", "ppt_jja", "pH", "awc", "depth"),
  partition = ".part",
  thr = NULL
)
esm_gam_t2$esm_model # bivariate model
esm_gam_t2$predictors
esm_gam_t2$performance

# Test with other bootstrap
abies2 <- abies2 %>% select(-starts_with("."))

set.seed(10)
abies2 <- part_random(
  data = abies2,
  pr_ab = "pr_ab",
  method = c(method = "boot", replicates = 3, proportion = 0.7)
)
abies2

esm_gam_t3 <- esm_gam(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin", "ppt_djf", "ppt_jja", "pH", "awc", "depth"),
  partition = ".part",
  thr = NULL
)
esm_gam_t3$esm_model # bivariate model
esm_gam_t3$predictors
esm_gam_t3$performance
```

---

esm_gau *Fit and validate Gaussian Process models based on Ensembles of Small Models approach*

---

## Description

This function constructs Gaussian Process models using the Ensembles of Small Models (ESM) approach (Breiner et al., 2015, 2018).

## Usage

```
esm_gau(data, response, predictors, partition, thr = NULL, background = NULL)
```

## Arguments

`data` data.frame. Database with the response (0,1) and predictors values.

`response` character. Column name with species absence-presence data (0,1)

`predictors` character. Vector with the column names of quantitative predictor variables (i.e. continuous variables). This function does not allow categorical variables and can only construct models with continuous variables. Usage predictors = c("aet", "cwd", "tmin")

`partition` character. Column name with training and validation partition groups.

`thr` character. Threshold used to get binary suitability values (i.e. 0,1). It is useful for threshold-dependent performance metrics. It is possible to use more than one threshold type. It is necessary to provide a vector for this argument. The following threshold criteria are available:

- equal_sens_spec: Threshold at which the sensitivity and specificity are equal.
- max_sens_spec: Threshold at which the sum of the sensitivity and specificity is the highest (aka threshold that maximizes the TSS).
- max_jaccard: The threshold at which the Jaccard index is the highest.
- max_sorensen: The threshold at which the Sorensen index is highest.
- max_fpb: The threshold at which FPB (F-measure on presence-background data) is highest.
- sensitivity: Threshold based on a specified sensitivity value. Usage thr = c('sensitivity', sens='0.6') or thr = c('sensitivity'). 'sens' refers to sensitivity value. If a sensitivity value is not specified, the default value is 0.9.

If the user wants to include more than one threshold type, it is necessary to concatenate threshold types, e.g., thr=c('max_sens_spec', 'max_jaccard'), or thr=c('max_sens_spec', 'sensitivity', sens='0.8'), or thr=c('max_sens_spec', 'sensitivity'). Function will use all thresholds if no threshold is specified

`background` data.frame. Database with response column only with 0 and predictors variables. All column names must be consistent with data. Default NULL

## Details

This method consists of creating bivariate models with all the pair-wise combinations of predictors and performs an ensemble based on the average of suitability weighted by Somers' D metric (D = 2 x (AUC -0.5)). ESM is recommended for modeling species with few occurrences. This function does not allow categorical variables because the use of these types of variables could be problematic when using with few occurrences. For further detail see Breiner et al. (2015, 2018).

## Value

A list object with:

- esm_model: A list with "graf" class object for each bivariate model. This object can be used for predicting ensembles of small models with sdm_predict function.
- predictors: A tibble with variables use for modeling.
- performance: Performance metric (see sdm_eval). Those threshold dependent metric are calculated based on the threshold specified in the argument.
- performance_part: Performance metric for each replica and partition (see sdm_eval).

## References


- Breiner, F. T., Guisan, A., Bergamini, A., & Nobis, M. P. (2015). Overcoming limitations of modelling rare species by using ensembles of small models. Methods in Ecology and Evolution, 6(10), 1210-218. https://doi.org/10.1111/2041-210X.12403
- Breiner, F. T., Nobis, M. P., Bergamini, A., & Guisan, A. (2018). Optimizing ensembles of small models for predicting the distribution of species with few occurrences. Methods in Ecology and Evolution, 9(4), 802-808. https://doi.org/10.1111/2041-210X.12957


## See Also

esm_gam, esm_gbm, esm_glm, esm_max, esm_net, and esm_svm.

## Examples

```
# Toy data: a small subset of 'abies' to keep this example fast
data("abies")
abies_toy <- abies[c(1:30, 701:730), ]
abies_toy <- part_random(
  data = abies_toy,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

esm_toy <- esm_gau(
  data = abies_toy,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin"),
  partition = ".part",
  thr = NULL
)
esm_toy$performance


data("abies")
require(dplyr)

# Using k-fold partition method
set.seed(10)
abies2 <- abies %>%
```

```
  na.omit() %>%
  group_by(pr_ab) %>%
  dplyr::slice_sample(n = 10) %>%
  group_by()

abies2 <- part_random(
  data = abies2,
  pr_ab = "pr_ab",
  method = c(method = "rep_kfold", folds = 3, replicates = 5)
)
abies2

# Without threshold specification and with kfold
esm_gau_t1 <- esm_gau(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin", "ppt_djf", "ppt_jja", "pH", "awc", "depth"),
  partition = ".part",
  thr = NULL
)

esm_gau_t1$esm_model # bivariate model
esm_gau_t1$predictors
esm_gau_t1$performance
esm_gau_t1$performance_part
```

---

esm_gbm *Fit and validate Generalized Boosted Regression models based on Ensembles of Small Models approach*

---

## Description

This function constructs Generalized Boosted Regression using the Ensembles of Small Models (ESM) approach (Breiner et al., 2015, 2018).

## Usage

```
esm_gbm(
  data,
  response,
  predictors,
  partition,
  thr = NULL,
  n_trees = 100,
  n_minobsinnode = NULL,
  shrinkage = 0.1
)
```

### Arguments

`data` data.frame. Database with the response (0,1) and predictors values.

`response` character. Column name with species absence-presence data (0,1)

`predictors` character. Vector with the column names of quantitative predictor variables (i.e. continuous variables). This can only construct models with continuous variables and does not allow categorical variables. Usage predictors = c("aet", "cwd", "tmin")

`partition` character. Column name with training and validation partition groups.

`thr` character. Threshold used to get binary suitability values (i.e. 0,1). It is useful for threshold-dependent performance metrics. It is possible to use more than one threshold type. It is necessary to provide a vector for this argument. The following threshold criteria are available:

- equal_sens_spec: Threshold at which the sensitivity and specificity are equal.
- max_sens_spec: Threshold at which the sum of the sensitivity and specificity is the highest (aka threshold that maximizes the TSS).
- max_jaccard: The threshold at which the Jaccard index is the highest.
- max_sorensen: The threshold at which the Sorensen index is highest.
- max_fpb: The threshold at which FPB (F-measure on presence-background data) is highest.
- sensitivity: Threshold based on a specified sensitivity value. Usage thr = c('sensitivity', sens='0.6') or thr = c('sensitivity'). 'sens' refers to sensitivity value. If a sensitivity value is not specified, the default value is 0.9.

In the case of use more than one threshold type it is necessary concatenate threshold types, e.g., thr=c('max_sens_spec', 'max_jaccard'), or thr=c('max_sens_spec', 'sensitivity', sens='0.8'), or thr=c('max_sens_spec', 'sensitivity'). Function will use all thresholds if no threshold is specified.

`n_trees` Integer specifying the total number of trees to fit. This is equivalent to the number of iterations and the number of basis functions in the additive expansion. Default is 100.

`n_minobsinnode` Integer specifying the minimum number of observations in the terminal nodes of the trees. Note that this is the actual number of observations, not the total weight. If n_minobsinnode is NULL, this parameter will assume a value equal to nrow(data)*0.5/4. Default is NULL.

`shrinkage` Numeric. This parameter applied to each tree in the expansion. Also known as the learning rate or step-size reduction; 0.001 to 0.1 usually works, but a smaller learning rate typically requires more trees. Default is 0.1.

### Details

This method consists of creating bivariate models with all the pair-wise combinations of predictors and perform an ensemble based on the average of suitability weighted by Somers' D metric (D = 2 x (AUC -0.5)). ESM is recommended for modeling species with few occurrences. This function does not allow categorical variables because the use of these types of variables could be problematic when using with few occurrences. For further detail see Breiner et al. (2015, 2018).

## Value

A list object with:

- esm_model: A list with "gbm" class object from gbm package for each bivariate model. This object can be used for predicting ensembles of small models with sdm_predict function.
- predictors: A tibble with variables use for modeling.
- performance: Performance metrics (see sdm_eval). Threshold dependent metrics are calculated based on the threshold specified in thr argument.
- performance_part: Performance metric for each replica and partition (see sdm_eval).

## References


- Breiner, F. T., Guisan, A., Bergamini, A., & Nobis, M. P. (2015). Overcoming limitations of modelling rare species by using ensembles of small models. Methods in Ecology and Evolution, 6(10), 1210-218. https://doi.org/10.1111/2041-210X.12403
- Breiner, F. T., Nobis, M. P., Bergamini, A., & Guisan, A. (2018). Optimizing ensembles of small models for predicting the distribution of species with few occurrences. Methods in Ecology and Evolution, 9(4), 802-808. https://doi.org/10.1111/2041-210X.12957


## See Also

esm_gam, esm_gau, esm_glm, esm_max, esm_net, and esm_svm.

## Examples

```
# Toy data: a small subset of 'abies' to keep this example fast
data("abies")
abies_toy <- abies[c(1:30, 701:730), ]
abies_toy <- part_random(
  data = abies_toy,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

esm_toy <- esm_gbm(
  data = abies_toy,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin"),
  partition = ".part",
  thr = NULL,
  n_trees = 20,
  n_minobsinnode = 3,
  shrinkage = 0.1
)
esm_toy$performance


data("abies")
require(dplyr)
```

```
# Using k-fold partition method
set.seed(10)
abies2 <- abies %>%
  na.omit() %>%
  group_by(pr_ab) %>%
  dplyr::slice_sample(n = 10) %>%
  group_by()

abies2 <- part_random(
  data = abies2,
  pr_ab = "pr_ab",
  method = c(method = "rep_kfold", folds = 3, replicates = 5)
)
abies2

# Without threshold specification and with kfold
esm_gbm_t1 <- esm_gbm(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin", "ppt_djf", "ppt_jja", "pH", "awc", "depth"),
  partition = ".part",
  thr = NULL,
  n_trees = 100,
  n_minobsinnode = NULL,
  shrinkage = 0.1
)

esm_gbm_t1$esm_model # bivariate model
esm_gbm_t1$predictors
esm_gbm_t1$performance
esm_gbm_t1$performance_part
```

---

esm_glm *Fit and validate Generalized Linear Models based on Ensembles of Small Models approach*

---

## Description

This function constructs Generalized Linear Models using the Ensembles of Small Models (ESM) approach (Breiner et al., 2015, 2018).

## Usage

```
esm_glm(
  data,
  response,
  predictors,
  partition,
  thr = NULL,
```

```
  poly = 0,
  inter_order = 0
)
```

## Arguments

`data` data.frame. Database with the response (0,1) and predictors values.

`response` character. Column name with species absence-presence data (0,1).

`predictors` character. Vector with the column names of quantitative predictor variables (i.e. continuous variables). This can only construct models with continuous variables and does not allow categorical variables. Usage predictors = c("aet", "cwd", "tmin").

`partition` character. Column name with training and validation partition groups.

`thr` character. Threshold used to get binary suitability values (i.e. 0,1). It is useful for threshold-dependent performance metrics. It is possible to use more than one threshold type. It is necessary to provide a vector for this argument. The following threshold criteria are available:

- equal_sens_spec: Threshold at which the sensitivity and specificity are equal.
- max_sens_spec: Threshold at which the sum of the sensitivity and specificity is the highest (aka threshold that maximizes the TSS).
- max_jaccard: The threshold at which Jaccard is the highest.
- max_sorensen: The threshold at which Sorensen is highest.
- max_fpb: The threshold at which FPB (F-measure on presence-background data) is highest.
- sensitivity: Threshold based on a specified sensitivity value. Usage thr = c('sensitivity', sens='0.6') or thr = c('sensitivity'). 'sens' refers to sensitivity value. If a sensitivity value is not specified, the default value is 0.9.

If the user wants to include more than one threshold type, it is necessary to concatenate threshold types, e.g., thr=c('max_sens_spec', 'max_jaccard'), or thr=c('max_sens_spec', 'sensitivity', sens='0.8'), or thr=c('max_sens_spec', 'sensitivity'). Function will use all thresholds if no threshold is specified

`poly` integer >= 2. If used with values >= 2 model will use polynomials for those continuous variables (i.e. used in predictors argument). Default is 0. Because ESM are constructed with few occurrences it is recommended no to use polynomials to avoid overfitting.

`inter_order` integer >= 0. The interaction order between explanatory variables. Default is 0. Because ESM are constructed with few occurrences it is recommended not to use interaction terms.

## Details

This method consists of creating bivariate models with all the pair-wise combinations of predictors and perform an ensemble based on the average of suitability weighted by Somers' D metric (D = 2 x (AUC -0.5)). ESM is recommended for modeling species with few occurrences. This function does not allow categorical variables because the use of these types of variables could be problematic when using with few occurrences. For further detail see Breiner et al. (2015, 2018).

## Value

A list object with:

- esm_model: A list with "glm" class object from stats package for each bivariate model. This object can be used for predicting ensembles of small models with sdm_predict function.
- predictors: A tibble with variables use for modeling.
- performance: Performance metric (see sdm_eval). Those threshold dependent metric are calculated based on the threshold specified in thr argument.
- performance_part: Performance metric for each replica and partition (see sdm_eval).

## References


- Breiner, F. T., Guisan, A., Bergamini, A., & Nobis, M. P. (2015). Overcoming limitations of modelling rare species by using ensembles of small models. Methods in Ecology and Evolution, 6(10), 1210-218. https://doi.org/10.1111/2041-210X.12403
- Breiner, F. T., Nobis, M. P., Bergamini, A., & Guisan, A. (2018). Optimizing ensembles of small models for predicting the distribution of species with few occurrences. Methods in Ecology and Evolution, 9(4), 802-808. https://doi.org/10.1111/2041-210X.12957


## See Also

esm_gam, esm_gau, esm_gbm, esm_max, esm_net, and esm_svm.

## Examples

```
# Toy data: a small subset of 'abies' to keep this example fast
data("abies")
abies_toy <- abies[c(1:30, 701:730), ]
abies_toy <- part_random(
  data = abies_toy,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

esm_toy <- esm_glm(
  data = abies_toy,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin"),
  partition = ".part",
  thr = NULL,
  poly = 1,
  inter_order = 0
)
esm_toy$performance


data("abies")
require(dplyr)

# Using k-fold partition method
```

```
set.seed(10)
abies2 <- abies %>%
  na.omit() %>%
  group_by(pr_ab) %>%
  dplyr::slice_sample(n = 10) %>%
  group_by()

abies2 <- part_random(
  data = abies2,
  pr_ab = "pr_ab",
  method = c(method = "rep_kfold", folds = 3, replicates = 5)
)
abies2

# Without threshold specification and with kfold
esm_glm_t1 <- esm_glm(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin", "ppt_djf", "ppt_jja", "pH", "awc", "depth"),
  partition = ".part",
  thr = NULL,
  poly = 0,
  inter_order = 0
)

esm_glm_t1$esm_model # bivariate model
esm_glm_t1$predictors
esm_glm_t1$performance
esm_glm_t1$performancePart
```

---

esm_max *Fit and validate Maximum Entropy Models based on Ensemble of Small of Model approach*

---

## Description

This function constructs Maxent Models using the Ensemble of Small Model (ESM) approach (Breiner et al., 2015, 2018).

## Usage

```
esm_max(
  data,
  response,
  predictors,
  partition,
  thr = NULL,
  background = NULL,
```

```
  clamp = TRUE,
  classes = "default",
  pred_type = "cloglog",
  regmult = 2.5
)
```

## Arguments

`data` data.frame. Database with the response (0,1) and predictors values.

`response` character. Column name with species absence-presence data (0,1)

`predictors` character. Vector with the column names of quantitative predictor variables (i.e. continuous variables). This function can only construct models with continuous variables, and does not allow categorical variables Usage predictors = c("aet", "cwd", "tmin").

`partition` character. Column name with training and validation partition groups.

`thr` character. Threshold used to get binary suitability values (i.e. 0,1). It is useful for threshold-dependent performance metrics. It is possible to use more than one threshold type. It is necessary to provide a vector for this argument. The following threshold criteria are available:

- equal_sens_spec: Threshold at which the sensitivity and specificity are equal.
- max_sens_spec: Threshold at which the sum of the sensitivity and specificity is the highest (aka threshold that maximizes the TSS).
- max_jaccard: The threshold at which Jaccard is the highest.
- max_sorensen: The threshold at which Sorensen is highest.
- max_fpb: The threshold at which FPB (F-measure on presence-background data) is highest.
- sensitivity: Threshold based on a specified sensitivity value. Usage thr = c('sensitivity', sens='0.6') or thr = c('sensitivity'). 'sens' refers to sensitivity value. If no sensitivity value is specified, the default is 0.9

If the user wants to include more than one threshold type, it is necessary concatenate threshold types, e.g., thr=c('max_sens_spec', 'max_jaccard'), or thr=c('max_sens_spec', 'sensitivity', sens='0.8'), or thr=c('max_sens_spec', 'sensitivity'). Function will use all thresholds if no threshold is specified.

`background` data.frame. Database with response column only with 0 and predictors variables. All column names must be consistent with data. Default NULL

`clamp` logical. It is set with TRUE, predictors and features are restricted to the range seen during model training.

`classes` character. A single feature of any combinations of them. Features are symbolized by letters: l (linear), q (quadratic), h (hinge), p (product), and t (threshold). Usage classes = "lpq". Default "default" (see details).

`pred_type` character. Type of response required available "link", "exponential", "cloglog" and "logistic". Default "cloglog"

`regmult` numeric. A constant to adjust regularization. Because ESM are used for modeling species with few records default value is 2.5

## Details

This method consists of creating bivariate models with all the pair-wise combinations of predictors and perform an ensemble based on the average of suitability weighted by Somers' D metric (D = 2 x (AUC -0.5)). ESM is recommended for modeling species with few occurrences. This function does not allow categorical variables because the use of these types of variables could be problematic when using with few occurrences. For further detail see Breiner et al. (2015, 2018). This function use a default regularization multiplier equal to 2.5 (see Breiner et al., 2018)

When the argument "classes" is set as default MaxEnt will use different features combination depending of the number of presences (np) with the follow rule: if np < 10 classes = "l", if np between 10 and 15 classes = "lq", if np between 15 and 80 classes = "lqh", and if np >= 80 classes = "lqph"

When presence-absence (or presence-pseudo-absence) data are used in data argument in addition to background points, the function will fit models with presences and background points and validate with presences and absences. This procedure makes maxent comparable to other presences-absences models (e.g., random forest, support vector machine). If only presences and background points data are used, function will fit and validate model with presences and background data. If only presence-absences are used in data argument and without background, function will fit model with the specified data (not recommended).

## Value

A list object with:

- esm_model: A list with "maxnet" class object from maxnet package for each bivariate model. This object can be used for predicting ensembles of small models with `sdm_predict` function.
- predictors: A tibble with variables use for modeling.
- performance: Performance metrics (see `sdm_eval`). Those threshold dependent metric are calculated based on the threshold specified in the argument.
- performance_part: Performance metric for each replica and partition (see `sdm_eval`).

## References


- Breiner, F. T., Guisan, A., Bergamini, A., & Nobis, M. P. (2015). Overcoming limitations of modelling rare species by using ensembles of small models. Methods in Ecology and Evolution, 6(10), 1210-218. https://doi.org/10.1111/2041-210X.12403
- Breiner, F. T., Nobis, M. P., Bergamini, A., & Guisan, A. (2018). Optimizing ensembles of small models for predicting the distribution of species with few occurrences. Methods in Ecology and Evolution, 9(4), 802-808. https://doi.org/10.1111/2041-210X.12957


## See Also

`esm_gam`, `esm_gau`, `esm_gbm`, `esm_glm`, `esm_net`, and `esm_svm`.

## Examples

```
# Toy data: a small subset of 'abies' to keep this example fast
data("abies")
abies_toy <- abies[c(1:30, 701:730), ]
abies_toy <- part_random(
```

```
  data = abies_toy,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

# Use the same records as background points (toy example only)
backg_toy <- abies_toy
backg_toy$pr_ab <- 0

esm_toy <- esm_max(
  data = abies_toy,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin"),
  partition = ".part",
  thr = NULL,
  background = backg_toy,
  clamp = TRUE,
  classes = "lq",
  pred_type = "cloglog",
  regmult = 1
)
esm_toy$performance


data("abies")
data("backg")
require(dplyr)

# Using k-fold partition method
set.seed(10)
abies2 <- abies %>%
  na.omit() %>%
  group_by(pr_ab) %>%
  dplyr::slice_sample(n = 10) %>%
  group_by()

abies2 <- part_random(
  data = abies2,
  pr_ab = "pr_ab",
  method = c(method = "rep_kfold", folds = 3, replicates = 2)
)
abies2

set.seed(10)
backg2 <- backg %>%
  na.omit() %>%
  group_by(pr_ab) %>%
  dplyr::slice_sample(n = 100) %>%
  group_by()

backg2 <- part_random(
  data = backg2,
  pr_ab = "pr_ab",
```

```
  method = c(method = "rep_kfold", folds = 3, replicates = 2)
)
backg2

# Without threshold specification and with kfold
esm_max_t1 <- esm_max(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin", "ppt_djf", "ppt_jja", "pH", "awc", "depth"),
  partition = ".part",
  thr = NULL,
  background = backg2,
  clamp = TRUE,
  classes = "default",
  pred_type = "cloglog",
  regmult = 1
)

esm_max_t1$esm_model # bivariate model
esm_max_t1$predictors
esm_max_t1$performance
esm_max_t1$performance_part
```

---

`esm_net` *Fit and validate Neural Networks based on Ensembles of Small of Models approach*

---

## Description

This function constructs Neural Networks using the Ensembles of Small Models (ESM) approach (Breiner et al., 2015, 2018).

## Usage

```
esm_net(data, response, predictors, partition, thr = NULL, size = 2, decay = 0)
```

## Arguments

| | |
|---|---|
| `data` | data.frame. Database with the response (0,1) and predictors values. |
| `response` | character. Column name with species absence-presence data (0,1) |
| `predictors` | character. Vector with the column names of quantitative predictor variables (i.e. continuous variables). This function only can construct models with continuous variables and does not allow categorical variables. Usage predictors = c("aet", "cwd", "tmin"). |
| `partition` | character. Column name with training and validation partition groups. |

`thr` character. Threshold used to get binary suitability values (i.e. 0,1). It is useful for threshold-dependent performance metrics. It is possible to use more than one threshold type. It is necessary to provide a vector for this argument. The following threshold criteria are available:

- equal_sens_spec: Threshold at which the sensitivity and specificity are equal.
- max_sens_spec: Threshold at which the sum of the sensitivity and specificity is the highest (aka threshold that maximizes the TSS).
- max_jaccard: The threshold at which the Jaccard index is the highest.
- max_sorensen: The threshold at which the Sorensen index is highest.
- max_fpb: The threshold at which FPB (F-measure on presence-background data) is highest.
- sensitivity: Threshold based on a specified sensitivity value. Usage thr = c('sensitivity', sens='0.6') or thr = c('sensitivity'). 'sens' refers to sensitivity value. If a sensitivity values is not specified, the default used is 0.9

If the user wants to include more than one threshold type, it is necessary concatenate threshold types, e.g., thr=c('max_sens_spec', 'max_jaccard'), or thr=c('max_sens_spec', 'sensitivity', sens='0.8'), or thr=c('max_sens_spec', 'sensitivity'). Function will use all thresholds if no threshold is specified

`size` numeric. Number of units in the hidden layer. Can be zero if there are skip-layer units. Default 2.

`decay` numeric. Parameter for weight decay. Default 0.

## Details

This method consists of creating bivariate models with all the pair-wise combinations of predictors and perform an ensemble based on the average of suitability weighted by Somers' D metric (D = 2 x (AUC -0.5)). ESM is recommended for modeling species with few occurrences. This function does not allow categorical variables because the use of these types of variables could be problematic when using with few occurrences. Further detail see Breiner et al. (2015, 2018).

## Value

A list object with:

- esm_model: A list with "nnet" class object from nnet package for each bivariate model. This object can be used for predicting ensemble of small model with `sdm_predict` function.
- predictors: A tibble with variables use for modeling.
- performance: Performance metric (see `sdm_eval`). Those threshold dependent metric are calculated based on the threshold specified in thr argument.
- performance_part: Performance metric for each replica and partition (see `sdm_eval`).

## References

- Breiner, F. T., Guisan, A., Bergamini, A., & Nobis, M. P. (2015). Overcoming limitations of modelling rare species by using ensembles of small models. Methods in Ecology and Evolution, 6(10), 1210-218. https://doi.org/10.1111/2041-210X.12403

- Breiner, F. T., Nobis, M. P., Bergamini, A., & Guisan, A. (2018). Optimizing ensembles of small models for predicting the distribution of species with few occurrences. Methods in Ecology and Evolution, 9(4), 802-808. https://doi.org/10.1111/2041-210X.12957


## See Also

esm_gam, esm_gau, esm_gbm, esm_glm, esm_max, and esm_svm.

## Examples

```
# Toy data: a small subset of 'abies' to keep this example fast
data("abies")
abies_toy <- abies[c(1:30, 701:730), ]
abies_toy <- part_random(
  data = abies_toy,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

esm_toy <- esm_net(
  data = abies_toy,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin"),
  partition = ".part",
  thr = NULL,
  size = 2,
  decay = 1
)
esm_toy$performance


data("abies")
require(dplyr)

# Using k-fold partition method
set.seed(10)
abies2 <- abies %>%
  na.omit() %>%
  group_by(pr_ab) %>%
  dplyr::slice_sample(n = 10) %>%
  group_by()

abies2 <- part_random(
  data = abies2,
  pr_ab = "pr_ab",
  method = c(method = "rep_kfold", folds = 3, replicates = 5)
)
abies2

# Without threshold specification and with kfold
esm_net_t1 <- esm_net(
  data = abies2,
```

```
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin", "ppt_djf", "ppt_jja", "pH", "awc", "depth"),
  partition = ".part",
  thr = NULL
)

esm_net_t1$esm_model # bivariate model
esm_net_t1$predictors
esm_net_t1$performance
esm_net_t1$performance_part
```

---

esm_svm *Fit and validate Support Vector Machine models based on Ensembles of Small of Models approach*

---

## Description

This function constructs Support Vector Machine models using the Ensembles of Small Models (ESM) approach (Breiner et al., 2015, 2018).

## Usage

```
esm_svm(
  data,
  response,
  predictors,
  partition,
  thr = NULL,
  sigma = "automatic",
  C = 1
)
```

## Arguments

| | |
|---|---|
| data | data.frame. Database with the response (0,1) and predictors values. |
| response | character. Column name with species absence-presence data (0,1). |
| predictors | character. Vector with the column names of quantitative predictor variables (i.e. continuous variables). This function only can construct models with continuous variables and does not allow categorical variables. Usage predictors = c("aet", "cwd", "tmin"). |
| partition | character. Column name with training and validation partition groups. |
| thr | character. Threshold used to get binary suitability values (i.e. 0,1). It is useful for threshold-dependent performance metrics. It is possible to use more than one threshold type. It is necessary to provide a vector for this argument. The following threshold criteria are available: |

- equal_sens_spec: Threshold at which the sensitivity and specificity are equal.
- max_sens_spec: Threshold at which the sum of the sensitivity and specificity is the highest (aka threshold that maximizes the TSS).
- max_jaccard: The threshold at which the Jaccard index is the highest.
- max_sorensen: The threshold at which the Sorensen index is highest.
- max_fpb: The threshold at which FPB (F-measure on presence-background data) is highest.
- sensitivity: Threshold based on a specified sensitivity value. Usage thr = c('sensitivity', sens='0.6') or thr = c('sensitivity'). 'sens' refers to sensitivity value. If a sensitivity value is not specified, the default is 0.9

If the user wants to include more than one threshold type, it is necessary concatenate threshold types, e.g., thr=c('max_sens_spec', 'max_jaccard'), or thr=c('max_sens_spec', 'sensitivity', sens='0.8'), or thr=c('max_sens_spec', 'sensitivity'). Function will use all thresholds if no threshold is specified

`sigma` numeric. Inverse kernel width for the Radial Basis kernel function "rbfdot". Default "automatic".

`C` numeric. Cost of constraints violation, the 'C' constant of the regularization term in the Lagrange formulation. Default 1

## Details

This method consists of creating bivariate models with all the pair-wise combinations of predictors and perform an ensemble based on the average of suitability weighted by Somers' D metric (D = 2 x (AUC -0.5)). ESM is recommended for modeling species with few occurrences. This function does not allow categorical variables because the use of these types of variables could be problematic when using with few occurrences. Further detail see Breiner et al. (2015, 2018). This function constructs 'C-svc' classification type and uses Radial Basis kernel "Gaussian" function (rbfdot). See details in ksvm

## Value

A list object with:

- esm_model: A list with "ksvm" class object from ksvm package for each bivariate model. This object can be used for predicting ensemble of small model with `sdm_predict` function.
- predictors: A tibble with variables use for modeling.
- performance: Performance metric (see `sdm_eval`). Those threshold dependent metric are calculated based on the threshold specified in thr argument.
- performance_part: Performance metric for each replica and partition (see `sdm_eval`).

## References


- Breiner, F. T., Guisan, A., Bergamini, A., & Nobis, M. P. (2015). Overcoming limitations of modelling rare species by using ensembles of small models. Methods in Ecology and Evolution, 6(10), 1210-218. https://doi.org/10.1111/2041-210X.12403

- Breiner, F. T., Nobis, M. P., Bergamini, A., & Guisan, A. (2018). Optimizing ensembles of small models for predicting the distribution of species with few occurrences. Methods in Ecology and Evolution, 9(4), 802-808. https://doi.org/10.1111/2041-210X.12957


## See Also

esm_gam, esm_gau, esm_gbm, esm_glm, esm_max,, and esm_net.

## Examples

```
# Toy data: a small subset of 'abies' to keep this example fast
data("abies")
abies_toy <- abies[c(1:30, 701:730), ]
abies_toy <- part_random(
  data = abies_toy,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

esm_toy <- esm_svm(
  data = abies_toy,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin"),
  partition = ".part",
  thr = NULL,
  sigma = 0.1,
  C = 2
)
esm_toy$performance


data("abies")
require(dplyr)

# Using k-fold partition method
set.seed(10)
abies2 <- abies %>%
  na.omit() %>%
  group_by(pr_ab) %>%
  dplyr::slice_sample(n = 10) %>%
  group_by()

abies2 <- part_random(
  data = abies2,
  pr_ab = "pr_ab",
  method = c(method = "rep_kfold", folds = 3, replicates = 5)
)
abies2

# Without threshold specification and with kfold
esm_svm_t1 <- esm_svm(
  data = abies2,
```

```
  response = "pr_ab",
  predictors = c(
    "aet", "cwd", "tmin", "ppt_djf", "ppt_jja",
    "pH", "awc", "depth"
  ),
  partition = ".part",
  thr = NULL
)

esm_svm_t1$esm_model # bivariate model
esm_svm_t1$predictors
esm_svm_t1$performance
esm_svm_t1$performance_part
```

---

`extra_eval` *Measure model extrapolation based on Shape extrapolation metric*

---

## Description

Measure extrapolation comparing environmental data used for modeling calibration and area for model projection. This function use the Shape metric proposed by Velazco et al., 2023 (doi:10.1111/ecog.06992)

## Usage

```
extra_eval(
  training_data,
  pr_ab,
  projection_data,
  metric = "mahalanobis",
  univar_comb = FALSE,
  aggreg_factor = 1
)
```

## Arguments

| | |
|---|---|
| `training_data` | data.frame or tibble with environmental conditions of presence and absence (or background points or pseudo-absences) used for constructing models |
| `pr_ab` | character. Column name with presence and absence (or background points or pseudo-absences) data (i.e., 1 and 0) |
| `projection_data` | SpatRaster, data.frame or tibble with environmental condition used for projecting a model (e.g., a larger, encompassing region, a spatially separate region, or a different time period). If data.frame or tibble is used function will return a tibble object. Otherwise, as SpatRaster object. |
| `metric` | character. Metric used to measure degree of extrapolation. Default = mahalanobis. |

- mahalanobis: Degree of extrapolation is calculated based on Mahalanobis distance.
- euclidean: Degree of extrapolation is calculated based on Euclidean distance.

`univar_comb` logical. If true, the function will add a layer or column to distinguish between univariate (i.e., projection data outside the range of training conditions) and combinatorial extrapolation (i.e., projection data within the range of training conditions) using values 1 and 2, respectively. Default FALSE

`aggreg_factor` positive integer. Aggregation factor expressed as number of cells in each direction to reduce raster resolution. Use value higher than 1 would be useful when measuring extrapolation using a raster with a high number of cells. The resolution of output will be the same as raster object used in 'projection_data' argument. Default 1, i.e., by default, no changes will be made to the resolution of the environmental variables.

## Details

This function measure model extrapolation base on the Shape metric (Velazco et al., 2023; doi:10.1111/ecog.06992). Shape is a model-agnostic approach that calculates the extrapolation degree for a given projection data point by its multivariate distance to the nearest training data point. Such distances are relativized by a factor that reflects the dispersion of the training data in environmental space. Distinct from other approaches (e.g., MESS-Multivariate Environmental Similarity Surfaces, EO-Environmental Overlap, MOP-Mobility-Oriented Parity, EXDET-Extrapolation Detection, or AOA-Area of Applicability), Shape incorporates an adjustable threshold to control the binary discrimination between acceptable and unacceptable extrapolation degrees (see `extra_truncate`).

See this vignette at flexsdm website for further details about Shape metric, model truncation, and tools to explore model extrapolation.

## Value

A SpatRaster or tibble object with extrapolation values measured by Shape metric. Also it is possible estimate univariate and combinatorial extrapolation metric (see 'univar_comb' argument).

## References


- Velazco, S.J.E., Brooke, M.R., De Marco Jr., P., Regan, H.M. and Franklin, J. 2023. How far can I extrapolate my species distribution model? Exploring Shape, a novel method. Ecography: e06992. doi:10.1111/ecog.06992


## See Also

`extra_truncate`, `p_extra`, `p_pdp`, `p_bpdp`

## Examples

```
# Toy data: small raster and records of one species to keep this example fast
data("spp")
somevar <- terra::rast(system.file("external/somevar.tif", package = "flexsdm"))
somevar <- terra::aggregate(somevar, fact = 6) # coarser resolution
```

```
sp3 <- spp[spp$species == "sp3", ]
sp3 <- sdm_extract(data = sp3, x = "x", y = "y", env_layer = somevar)

extr <- extra_eval(
  training_data = sp3,
  pr_ab = "pr_ab",
  projection_data = somevar,
  metric = "mahalanobis"
)
extr


require(dplyr)
require(terra)

data(spp)
f <- system.file("external/somevar.tif", package = "flexsdm")
somevar <- terra::rast(f)
names(somevar) <- c("aet", "cwd", "tmx", "tmn")


spp$species %>% unique()
sp <- spp %>%
  dplyr::filter(species == "sp3", pr_ab == 1) %>%
  dplyr::select(x, y, pr_ab)

# Calibration area based on some criterion such as dispersal ability
ca <- calib_area(sp,
  x = "x", y = "y",
  method = c("bmcp", width = 50000),
  crs = crs(somevar)
)

plot(somevar[[1]])
points(sp)
plot(ca, add = TRUE)


# Sampling pseudo-absences
set.seed(10)
psa <- sample_pseudoabs(
  data = sp,
  x = "x",
  y = "y",
  n = nrow(sp) * 2,
  method = "random",
  rlayer = somevar,
  calibarea = ca
)

# Merge presences and absences databases to get a complete calibration data
sp_pa <- dplyr::bind_rows(sp, psa)
sp_pa
```

```
# Get environmental condition of calibration area
sp_pa_2 <- sdm_extract(
  data = sp_pa,
  x = "x",
  y = "y",
  env_layer = somevar
)
sp_pa_2

# Measure degree of extrapolation based on Mahalanobis and
# for a projection area based on a SpatRaster object
extr <-
  extra_eval(
    training_data = sp_pa_2,
    projection_data = somevar,
    pr_ab = "pr_ab",
    aggreg_factor = 1,
    metric = "mahalanobis"
  )
plot(extr, main = "Extrapolation pattern")


# Let's fit, predict and truncate a model with extra_truncate
sp_pa_2 <- part_random(
  data = sp_pa_2,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 5)
)

a_model <- fit_glm(
  data = sp_pa_2,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmx", "tmn"),
  partition = ".part",
  thr = c("max_sorensen")
)

predsuit <- sdm_predict(
  models = a_model,
  pred = somevar,
  thr = "max_sorensen"
)
predsuit # list with a raster with two layer
plot(predsuit[[1]])

# Truncate a model based on a given value of extrapolation
# using 'extra_truncate' function
oldpar <- par(mfrow = c(1, 2))
plot(extr, main = "Extrapolation")
plot(predsuit[[1]][[1]], main = "Suitability")
par(oldpar) # restore graphical parameters
```

```
predsuit_2 <- extra_truncate(
  suit = predsuit[[1]],
  extra = extr,
  threshold = c(50, 100, 200)
)
predsuit_2 # a list of continuous and binary models with
# different truncated at different extrapolation thresholds

plot(predsuit_2$`50`)
plot(predsuit_2$`100`)
plot(predsuit_2$`200`)


## %#######################################################%##
####        Measure degree of extrapolation for         ####
####        projection area based on data.frame         ####
## %#######################################################%##

extr_df <-
  extra_eval(
    training_data = sp_pa_2,
    projection_data = as.data.frame(somevar, xy = TRUE),
    pr_ab = "pr_ab",
    aggreg_factor = 1,
    metric = "mahalanobis"
  )
extr_df
# see 'p_extra()' to explore extrapolation or suitability pattern in the
# environmental and/or geographical space

## %#######################################################%##
####             Explore Shape metric with              ####
####     univariate and combinatorial extrapolation     ####
## %#######################################################%##
extr <-
  extra_eval(
    training_data = sp_pa_2,
    projection_data = somevar,
    pr_ab = "pr_ab",
    aggreg_factor = 1,
    metric = "mahalanobis",
    univar_comb = TRUE
  )

extr
plot(extr) # In the second layer, values equal to 1 and 2
# depict univariate and combinatorial extrapolation, respectively
```

---

`extra_truncate` *Truncate suitability predictions based on an extrapolation value*

---

## Description

Exclusion of suitability predictions in environmental conditions assumed to be extrapolative.

## Usage

```
extra_truncate(suit, extra, threshold = 50, trunc_value = 0)
```

## Arguments

| | |
|---|---|
| suit | SpatRaster with suitability values |
| extra | SpatRaster with extrapolation values preferable measured with extra_eval function |
| threshold | numeric. Vector with one or more extrapolation values used for truncate suitability Default 50% |
| trunc_value | numeric. Numeric value to be used to those cells assumed to be extrapolative |

## Details

Exclusion of suitability predictions in environmental conditions assumed to be extrapolative. In this function it is possible to use any metric measuring degree of extrapolation (e.g., MESS-Multivariate Environmental Similarity Surfaces, EO-Environmental Overlap, MOP-Mobility-Oriented Parity, EXDET-Extrapolation Detection, or AOA-Area of Applicability). However, we recommend to use Shape approach (see extra_eval, and Velazco et al., 2023; doi:10.1111/ecog.06992).

This function truncates suitability predictions assigning a given value, generally 0 or NA. Usage trunc_value = NA. Default 0.

To those cells assumed to be extrapolative, i.e., higher than a given threshold of a given extrapolation metric.

See this vignette at flexsdm website for further details about Shape metric, model truncation, and tools to explore model extrapolation.

## Value

A SpatRaster object with truncated suitability values

## See Also

extra_eval, p_extra, p_pdp, p_bpdp

## Examples

```
# Toy data: small raster and records of one species to keep this example fast
data("spp")
somevar <- terra::rast(system.file("external/somevar.tif", package = "flexsdm"))
somevar <- terra::aggregate(somevar, fact = 6) # coarser resolution
sp3 <- spp[spp$species == "sp3", ]
sp3 <- sdm_extract(data = sp3, x = "x", y = "y", env_layer = somevar)

# Extrapolation (Shape metric) in the projection area
```

```
extr <- extra_eval(
  training_data = sp3,
  pr_ab = "pr_ab",
  projection_data = somevar,
  metric = "mahalanobis"
)

# Any suitability layer can be truncated; here, a toy layer rescaled to 0-1
suit <- somevar[[1]]
suit <- suit / terra::global(suit, "max", na.rm = TRUE)[[1]]
names(suit) <- "suit"

# Suitability is set to 0 where extrapolation is higher than the threshold
suit_trunc <- extra_truncate(
  suit = suit,
  extra = extr,
  threshold = 50,
  trunc_value = 0
)
suit_trunc # one raster for each threshold value

# see more examples in extra_eval function
```

---

`fit_dom` *Fit and validate Domain models*

---

## Description

Fit and validate Domain models

## Usage

```
fit_dom(
  data,
  response,
  predictors,
  predictors_f = NULL,
  partition = NULL,
  thr = NULL
)
```

## Arguments

| | |
|---|---|
| `data` | data.frame. Database with response (0,1) and predictors values. |
| `response` | character. Column name with species absence-presence data (0,1). |
| `predictors` | character. Vector with the column names of quantitative predictor variables (i.e. continuous variables). Usage predictors = c("aet", "cwd", "tmin") |
| `predictors_f` | character. Vector with the column names of qualitative predictor variables (i.e. ordinal or nominal variables; factors). Usage predictors_f = c("landform") |

`partition` character. Column name with training and validation partition groups. If partition = NULL, the model will be validated with the same data used for fitting.

`thr` character. Threshold used to get binary suitability values (i.e. 0,1). This is useful for threshold-dependent performance metrics. It is possible to use more than one threshold type. It is necessary to provide a vector for this argument. The following threshold criteria are available:

- lpt: The highest threshold at which there is no omission.
- equal_sens_spec: Threshold at which the sensitivity and specificity are equal.
- max_sens_spec: Threshold at which the sum of the sensitivity and specificity is the highest (aka threshold that maximizes the TSS).
- max_jaccard: The threshold at which the Jaccard index is the highest.
- max_sorensen: The threshold at which the Sorensen index is highest.
- max_fpb: The threshold at which FPB (F-measure on presence-background data) is highest.
- sensitivity: Threshold based on a specified sensitivity value. Usage thr = c('sensitivity', sens='0.6') or thr = c('sensitivity'). 'sens' refers to sensitivity value. If a sensitivity value is not specified, the default used is 0.9.

If more than one threshold type is used they must be concatenated, e.g., thr=c('lpt', 'max_sens_spec', 'max_jaccard'), or thr=c('lpt', 'max_sens_spec', 'sensitivity', sens='0.8'), or thr=c('lpt', 'max_sens_spec', 'sensitivity'). Function will use all threshold types if none is specified.

## Details

This function fits and validates Domain models. The Domain model is a simple model that uses the Gower distance to calculate environmental similarity between the presence data and test data (Carpenter et al., 1993). Gower range of values area based on presences data. Gower distance are transformed to max(0, 1 - Gower). This involves subtracting the distance from 1 and then ensuring the result is not negative (clamping it at zero). Gower distance is calculated with `map_env_dist` function

This function fit and validate Domain models. The Domain model is a simple model that uses the Gower distance to calculate the similarity between the presences training and presence-absences test data.

## Value

A list object with:

- model: A tibble with presences. This object can be used for predicting.
- predictors: A tibble with quantitative (c column names) and qualitative (f column names) variables use for modeling.
- performance: Performance metric (see `sdm_eval`). Threshold dependent metrics are calculated based on the threshold specified in the argument.
- performance_part: Performance metric for each replica and partition (see `sdm_eval`).
- data_ens: Predicted suitability for each test partition. This database is used in `fit_ensemble`

## References


- Carpenter, G., Gillison, A.N., Winter, J., 1993. DOMAIN: a flexible modelling procedure for mapping potential distributions of plants and animals. Biodiversity & Conservation 2, 667–680


## See Also

fit_gam, fit_gau, fit_gbm, fit_glm, fit_max, fit_net, fit_raf, and fit_svm.

## Examples

```
# Toy data: a small subset of 'abies' to keep this example fast
data("abies")
abies_toy <- abies[c(1:30, 701:730), ]
abies_toy <- part_random(
  data = abies_toy,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

dom_toy <- fit_dom(
  data = abies_toy,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin"),
  partition = ".part",
  thr = "max_sens_spec"
)
dom_toy$performance


require(dplyr)
require(terra)

data("spp")
somevar <- system.file("external/somevar.tif", package = "flexsdm")
somevar <- terra::rast(somevar)

# Extract data
some_sp <- spp %>%
  filter(species == "sp2")

some_sp <-
  sdm_extract(
    data = some_sp,
    x = "x",
    y = "y",
    env_layer = somevar
  )

# Partition
some_sp <- part_random(
  data = some_sp,
```

```
  pr_ab = "pr_ab",
  method = c(method = "rep_kfold", folds = 3, replicates = 5)
)


## %#######################################################%##
#                                                           #
####                Fit a Domain model                  ####
#                                                           #
## %#######################################################%##
# Fit some models
mdom <- fit_dom(
  data = some_sp,
  response = "pr_ab",
  predictors = c("CFP_1", "CFP_2", "CFP_3", "CFP_4"),
  predictors_f = NULL,
  partition = ".part",
  thr = c("max_sens_spec")
)

mdom

# Predict model
ind_p <- sdm_predict(
  models = mdom,
  pred = somevar,
  thr = "max_sens_spec",
  con_thr = TRUE,
  predict_area = NULL
)
plot(ind_p$dom)

## %#######################################################%##
#                                                           #
####             Explore Domain suitabiltiy             ####
####             in the environmental space             ####
#                                                           #
## %#######################################################%##

p_extra(
  training_data = some_sp %>% dplyr::filter(pr_ab == 1), # select only presences
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  extra_suit_data = ind_p$dom$dom,
  projection_data = somevar,
  geo_space = FALSE,
  prop_points = 0.3,
  alpha_p = 0.8,
  color_p = "black",
  color_gradient = c("#000033", "#1400FF", "#C729D6", "#FF9C63", "#FFFF60")
)
```

```
p_extra(
  training_data = some_sp %>% dplyr::filter(pr_ab == 1), # select only presences
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  predictors = c("CFP_1", "CFP_2"), # Just the first two predictors
  extra_suit_data = ind_p$dom$dom > 0.96, # a binary map
  projection_data = somevar,
  geo_space = TRUE,
  prop_points = 0.4,
  alpha_p = 0.8,
  color_p = "black",
  color_gradient = c("#1400FF", "#C729D6")
)
```

---

`fit_ensemble` *Ensemble model fitting and validation*

---

## Description

Ensemble model fitting and validation

## Usage

```
fit_ensemble(
  models,
  ens_method = c("mean", "meanw", "meansup", "meanthr", "median"),
  thr = NULL,
  thr_model = NULL,
  metric = NULL
)
```

## Arguments

`models` list. A list of models fitted with fit_ or tune_ function family. Models used for ensemble must have the same presences-absences records, partition methods, and threshold types.

`ens_method` character. Method used to create ensemble of different models. A vector must be provided for this argument. For meansup, meanw or pcasup method, it is necessary to provide an evaluation metric and threshold in 'metric' and 'thr_model' arguments respectively. By default all of the following ensemble methods will be performed:

- mean: Simple average of the different models.
- meanw: Weighted average of models based on their performance. An evaluation metric and threshold type must be provided.
- meansup: Average of the best models (those with the evaluation metric above the average). An evaluation metric must be provided.

- meanthr: Averaging performed only with those cells with suitability values above the selected threshold.
- median: Median of the different models.

Usage ensemble = "meanthr". If several ensemble methods are to be implemented it is necessary to concatenate them, e.g., ensemble = c("meanw", "meanthr", "median")

`thr` character. Threshold used to get binary suitability values (i.e. 0,1). It is useful for threshold-dependent performance metrics. It is possible to use more than one threshold criterion. A vector must be provided for this argument. The following threshold criteria are available:

- lpt: The highest threshold at which there is no omission.
- equal_sens_spec: Threshold at which the sensitivity and specificity are equal.
- max_sens_spec: Threshold at which the sum of the sensitivity and specificity is the highest (aka threshold that maximizes the TSS).
- max_jaccard: The threshold at which Jaccard is the highest.
- max_sorensen: The threshold at which Sorensen is highest.
- max_fpb: The threshold at which FPB (F-measure on presence-background data) is highest.
- sensitivity: Threshold based on a specified sensitivity value. Usage thr = c('sensitivity', sens='0.6') or thr = c('sensitivity'). 'sens' refers to sensitivity value. If a sensitivity values is not specified, default is 0.9.

In the case of using more than one threshold type it is necessary concatenate threshold types, e.g., thr=c('lpt', 'max_sens_spec', 'max_jaccard'), or thr=c('lpt', 'max_sens_spec', 'sensitivity', sens='0.8'), or thr=c('lpt', 'max_sens_spec', 'sensitivity'). Function will use all thresholds if no threshold is specified.

`thr_model` character. This threshold is needed for conduct meanw, meandsup, and meanthr ensemble methods. It is mandatory to use only one threshold, and this must be the same threshold used to fit all the models used in the "models" argument. Usage thr_model = 'equal_sens_spec'

`metric` character. Performance metric used for selecting the best combination of hyperparameter values. One of the following metrics can be used: SORENSEN, JACCARD, FPB, TSS, KAPPA, AUC, IMAE, and BOYCE. Default TSS. Usage metric = BOYCE

## Value

A list object with:

- models: A list of models used for performing ensemble.
- thr_metric: Threshold and metric specified in the function.
- predictors: A tibble of quantitative (column names with c) and qualitative (column names with f) variables used in each models.
- performance: A tibble with performance metrics (see `sdm_eval`).
- performance_part: Performance metric for each replica and partition (see `sdm_eval`). Those metrics that are threshold-dependent are calculated based on the threshold specified in the argument.

## Examples

```
# Toy data: a small subset of 'abies' to keep this example fast
data("abies")
abies_toy <- abies[c(1:30, 701:730), ]
abies_toy <- part_random(
  data = abies_toy,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

glm_toy <- fit_glm(
  data = abies_toy,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin"),
  partition = ".part",
  thr = "max_sens_spec"
)
svm_toy <- fit_svm(
  data = abies_toy,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin"),
  partition = ".part",
  thr = "max_sens_spec"
)

ens_toy <- fit_ensemble(
  models = list(glm_toy, svm_toy),
  ens_method = "mean",
  thr = NULL,
  thr_model = "max_sens_spec",
  metric = "TSS"
)
ens_toy$performance


require(dplyr)
require(terra)

# Environmental variables
somevar <-
  system.file("external/somevar.tif", package = "flexsdm")
somevar <- terra::rast(somevar)

# Species occurrences
data("spp")
set.seed(1)
some_sp <- spp %>%
  dplyr::filter(species == "sp2") %>%
  sdm_extract(
    data = .,
    x = "x",
    y = "y",
```

```
    env_layer = somevar,
    variables = names(somevar),
    filter_na = TRUE
  ) %>%
  part_random(
    data = .,
    pr_ab = "pr_ab",
    method = c(method = "kfold", folds = 3)
  )


# gam
mglm <- fit_glm(
  data = some_sp,
  response = "pr_ab",
  predictors = c("CFP_1", "CFP_2", "CFP_3", "CFP_4"),
  partition = ".part",
  poly = 2
)
mraf <- fit_raf(
  data = some_sp,
  response = "pr_ab",
  predictors = c("CFP_1", "CFP_2", "CFP_3", "CFP_4"),
  partition = ".part",
)
mgbm <- fit_gbm(
  data = some_sp,
  response = "pr_ab",
  predictors = c("CFP_1", "CFP_2", "CFP_3", "CFP_4"),
  partition = ".part"
)

# Fit and validate ensemble model
mensemble <- fit_ensemble(
  models = list(mglm, mraf, mgbm),
  ens_method = "meansup",
  thr = NULL,
  thr_model = "max_sens_spec",
  metric = "TSS"
)

mensemble
```

---

`fit_gam` *Fit and validate Generalized Additive Models*

---

## Description

Fit and validate Generalized Additive Models

## Usage

```
fit_gam(
  data,
  response,
  predictors,
  predictors_f = NULL,
  select_pred = FALSE,
  partition = NULL,
  thr = NULL,
  fit_formula = NULL,
  k = -1
)
```

## Arguments

`data` data.frame. Database with response (0,1) and predictors values.

`response` character. Column name with species absence-presence data (0,1).

`predictors` character. Vector with the column names of quantitative predictor variables (i.e. continuous variables). Usage predictors = c("aet", "cwd", "tmin")

`predictors_f` character. Vector with the column names of qualitative predictor variables (i.e. ordinal or nominal variables; factors). Usage predictors_f = c("landform")

`select_pred` logical. Perform predictor selection. Default FALSE.

`partition` character. Column name with training and validation partition groups. If partition = NULL, the model will be validated with the same data used for fitting.

`thr` character. Threshold used to get binary suitability values (i.e. 0,1). This is useful for threshold-dependent performance metrics. It is possible to use more than one threshold type. It is necessary to provide a vector for this argument. The following threshold criteria are available:

- lpt: The highest threshold at which there is no omission.
- equal_sens_spec: Threshold at which the sensitivity and specificity are equal.
- max_sens_spec: Threshold at which the sum of the sensitivity and specificity is the highest (aka threshold that maximizes the TSS).
- max_jaccard: The threshold at which the Jaccard index is the highest.
- max_sorensen: The threshold at which the Sorensen index is highest.
- max_fpb: The threshold at which FPB (F-measure on presence-background data) is highest.
- sensitivity: Threshold based on a specified sensitivity value. Usage thr = c('sensitivity', sens='0.6') or thr = c('sensitivity'). 'sens' refers to sensitivity value. If a sensitivity value is not specified, the default used is 0.9.

If more than one threshold type is used they must be concatenated, e.g., thr=c('lpt', 'max_sens_spec', 'max_jaccard'), or thr=c('lpt', 'max_sens_spec', 'sensitivity', sens='0.8'), or thr=c('lpt', 'max_sens_spec', 'sensitivity'). Function will use all threshold types if none is specified.

| | |
|---|---|
| fit_formula | formula. A formula object with response and predictor variables (e.g. formula(pr_ab ~ aet + ppt_jja + pH + awc + depth + landform)). Note that the variables used here must be consistent with those used in response, predictors, and predictors_f arguments |
| k | integer. The dimension of the basis used to represent the smooth term. Default -1 (i.e., k=10). See the help in ?mgcv::s. |

## Details

This function fits GAM using mgvc package, with Binomial distribution family and thin plate regression spline as a smoothing basis (see ?mgvc::s).

## Value

A list object with:

- model: A "gam" class object from mgcv package. This object can be used for predicting.
- predictors: A tibble with quantitative (c column names) and qualitative (f column names) variables use for modeling.
- performance: Performance metric (see sdm_eval). Threshold dependent metrics are calculated based on the threshold specified in the argument.
- performance_part: Performance metric for each replica and partition (see sdm_eval).
- data_ens: Predicted suitability for each test partition. This database is used in fit_ensemble

## See Also

fit_gau, fit_gbm, fit_glm, fit_max, fit_net, fit_raf, and fit_svm.

## Examples

```
# Toy data: a small subset of 'abies' to keep this example fast
data("abies")
abies_toy <- abies[c(1:30, 701:730), ]
abies_toy <- part_random(
  data = abies_toy,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

gam_toy <- fit_gam(
  data = abies_toy,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin"),
  partition = ".part",
  thr = "max_sens_spec",
  k = 3
)
gam_toy$performance
```

```
require(dplyr)
data("abies")

# Using k-fold partition method
abies2 <- part_random(
  data = abies,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)
abies2

gam_t1 <- fit_gam(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "ppt_jja", "pH", "awc", "depth"),
  predictors_f = c("landform"),
  select_pred = FALSE,
  partition = ".part",
  thr = "max_sens_spec"
)
gam_t1$model
gam_t1$predictors
gam_t1$performance
gam_t1$performance_part

# Specifying the formula explicitly
require(mgcv)
gam_t2 <- fit_gam(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "ppt_jja", "pH", "awc", "depth"),
  predictors_f = c("landform"),
  select_pred = FALSE,
  partition = ".part",
  thr = "max_sens_spec",
  fit_formula = stats::formula(pr_ab ~ s(aet) +
    s(ppt_jja) +
    s(pH) + landform)
)

gam_t2$model
gam_t2$predictors
gam_t2$performance %>% dplyr::select(ends_with("_mean"))

# Using repeated k-fold partition method
abies2 <- part_random(
  data = abies,
  pr_ab = "pr_ab",
  method = c(method = "rep_kfold", folds = 3, replicates = 2)
)
abies2

gam_t3 <- fit_gam(
```

```
  data = abies2,
  response = "pr_ab",
  predictors = c("ppt_jja", "pH", "awc"),
  predictors_f = c("landform"),
  select_pred = FALSE,
  partition = ".part",
  thr = "max_sens_spec"
)
gam_t3
```

---

fit_gau *Fit and validate Gaussian Process models*

---

## Description

Fit and validate Gaussian Process models

## Usage

```
fit_gau(
  data,
  response,
  predictors,
  predictors_f = NULL,
  background = NULL,
  partition = NULL,
  thr = NULL
)
```

## Arguments

| | |
|---|---|
| data | data.frame. Database with response (0,1) and predictors values. |
| response | character. Column name with species absence-presence data (0,1). |
| predictors | character. Vector with the column names of quantitative predictor variables (i.e. continuous variables). Usage predictors = c("aet", "cwd", "tmin") |
| predictors_f | character. Vector with the column names of qualitative predictor variables (i.e. ordinal or nominal variables type). Usage predictors_f = c("landform") |
| background | data.frame. Database with response column only with 0 and predictors variables. All column names must be consistent with data |
| partition | character. Column name with training and validation partition groups. If partition = NULL, the model will be validated with the same data used for fitting. |
| thr | character. Threshold used to get binary suitability values (i.e. 0,1), useful for threshold-dependent performance metrics. It is possible to use more than one threshold type. A vector must be provided for this argument. The following threshold criteria are available: |

- lpt: The highest threshold at which there is no omission.
- equal_sens_spec: Threshold at which the sensitivity and specificity are equal.
- max_sens_spec: Threshold at which the sum of the sensitivity and specificity is the highest (aka threshold that maximizes the TSS).
- max_jaccard: The threshold at which the Jaccard index is the highest.
- max_sorensen: The threshold at which the Sorensen index is the highest.
- max_fpb: The threshold at which FPB (F-measure on presence-background data) is highest.
- sensitivity: Threshold based on a specified sensitivity value. Usage thr = c('sensitivity', sens='0.6') or thr = c('sensitivity'). 'sens' refers to sensitivity value. If a sensitivity value is not specified, the default used is 0.9.

If more than one threshold type is used they must be concatenated, e.g., thr=c('lpt', 'max_sens_spec', 'max_jaccard'), or thr=c('lpt', 'max_sens_spec', 'sensitivity', sens='0.8'), or thr=c('lpt', 'max_sens_spec', 'sensitivity'). Function will use all threshold criteria if none is specified.

## Value

A list object with:

- model: A "graf" class object. This object can be used for predicting.
- predictors: A tibble with quantitative (c column names) and qualitative (f column names) variables use for modeling.
- performance: Performance metrics (see sdm_eval). Threshold dependent metrics are calculated based on the threshold criteria specified in the argument.
- performance_part: Performance metric for each replica and partition (see sdm_eval).
- data_ens: Predicted suitability for each test partition. This database is used in fit_ensemble

## See Also

fit_gam, fit_glm, fit_gbm, fit_max, fit_net, fit_raf, and fit_svm.

## Examples

```
# Toy data: a small subset of 'abies' to keep this example fast
data("abies")
abies_toy <- abies[c(1:30, 701:730), ]
abies_toy <- part_random(
  data = abies_toy,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

gau_toy <- fit_gau(
  data = abies_toy,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin"),
```

```
  partition = ".part",
  thr = "max_sens_spec"
)
gau_toy$performance


data("abies")
set.seed(1)
abies <- abies[sample(nrow(abies), 400), ] # subsample to speed up this example

# Using k-fold partition method
abies2 <- part_random(
  data = abies,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)
abies2

bg <- abies2
bg$pr_ab <- 0


gaup_t1 <- fit_gau(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "ppt_jja", "pH", "awc", "depth"),
  predictors_f = c("landform"),
  partition = ".part",
  background = bg,
  thr = c("max_sens_spec", "equal_sens_spec", "max_sorensen")
)

gaup_t1$model
gaup_t1$predictors
gaup_t1$performance
gaup_t1$performance_part
gaup_t1$data_ens

# Using bootstrap partition method only with presence-absence
abies2 <- part_random(
  data = abies,
  pr_ab = "pr_ab",
  method = c(method = "boot", replicates = 2, proportion = 0.7)
)
abies2

gaup_t2 <- fit_gau(
  data = abies2,
  response = "pr_ab",
  predictors = c("ppt_jja", "pH", "awc"),
  predictors_f = c("landform"),
  partition = ".part",
  thr = c(type = c("lpt", "max_sens_spec", "sensitivity"), sens = "0.8")
```

```
)
gaup_t2
```

---

`fit_gbm` *Fit and validate Generalized Boosted Regression models*

---

## Description

Fit and validate Generalized Boosted Regression models

## Usage

```
fit_gbm(
  data,
  response,
  predictors,
  predictors_f = NULL,
  fit_formula = NULL,
  partition = NULL,
  thr = NULL,
  n_trees = 100,
  n_minobsinnode = as.integer(nrow(data) * 0.9/4),
  shrinkage = 0.1
)
```

## Arguments

| | |
|---|---|
| `data` | data.frame. Database with response (0,1) and predictors values. |
| `response` | character. Column name with species absence-presence data (0,1). |
| `predictors` | character. Vector with the column names of quantitative predictor variables (i.e. continuous variables). Usage predictors = c("aet", "cwd", "tmin") |
| `predictors_f` | character. Vector with the column names of qualitative predictor variables (i.e. ordinal or nominal variables type). Usage predictors_f = c("landform") |
| `fit_formula` | formula. A formula object with response and predictor variables (e.g. formula(pr_ab ~ aet + ppt_jja + pH + awc + depth + landform)). Note that the variables used here must be consistent with those used in response, predictors, and predictors_f arguments. Default is NULL. |
| `partition` | character. Column name with training and validation partition groups. If partition = NULL, the model will be validated with the same data used for fitting. |
| `thr` | character. Threshold used to get binary suitability values (i.e. 0,1) needed for threshold-dependent performance metrics. It is possible to use more than one threshold type. It is necessary to provide a vector for this argument. The following threshold criteria are available:<br>• lpt: The highest threshold at which there is no omission. |

- equal_sens_spec: Threshold at which the sensitivity and specificity are equal.
- max_sens_spec: Threshold at which the sum of the sensitivity and specificity is the highest (aka threshold that maximizes the TSS).
- max_jaccard: The threshold at which the Jaccard index is the highest.
- max_sorensen: The threshold at which the Sorensen index is highest.
- max_fpb: The threshold at which FPB (F-measure on presence-background data) is highest.
- sensitivity: Threshold based on a specified sensitivity value. Usage thr = c('sensitivity', sens='0.6') or thr = c('sensitivity'). 'sens' refers to sensitivity value. If a sensitivity value is not specified, the default used is 0.9

If more than one threshold type is used they must be concatenated, e.g., thr=c('lpt', 'max_sens_spec', 'max_jaccard'), or thr=c('lpt', 'max_sens_spec', 'sensitivity', sens='0.8'), or thr=c('lpt', 'max_sens_spec', 'sensitivity'). Function will use all thresholds if no threshold is specified.

`n_trees` Integer specifying the total number of trees to fit. This is equivalent to the number of iterations and the number of basis functions in the additive expansion. Default is 100.

`n_minobsinnode` Integer specifying the minimum number of observations in the terminal nodes of the trees. Note that this is the actual number of observations, not the total weight. The default value used is nrow(data)*0.5/4

`shrinkage` Numeric. This parameter applied to each tree in the expansion. Also known as the learning rate or step-size reduction; 0.001 to 0.1 usually works, but a smaller learning rate typically requires more trees. Default is 0.1.

## Value

A list object with:

- model: A "gbm" class object from gbm package. This object can be used for predicting.
- predictors: A tibble with quantitative (c column names) and qualitative (f column names) variables use for modeling.
- performance: Performance metric (see `sdm_eval`). Threshold dependent metrics are calculated based on the threshold specified in thr argument.
- performance_part: Performance metric for each replica and partition (see `sdm_eval`).
- data_ens: Predicted suitability for each test partition based on the best model. This database is used in `fit_ensemble`

## See Also

`fit_gam`, `fit_gau`, `fit_glm`, `fit_max`, `fit_net`, `fit_raf`, and `fit_svm`.

## Examples

```
# Toy data: a small subset of 'abies' to keep this example fast
data("abies")
```

```
abies_toy <- abies[c(1:30, 701:730), ]
abies_toy <- part_random(
  data = abies_toy,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

gbm_toy <- fit_gbm(
  data = abies_toy,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin"),
  partition = ".part",
  thr = "max_sens_spec",
  n_trees = 20,
  n_minobsinnode = 3
)
gbm_toy$performance


data("abies")

# Using k-fold partition method
abies2 <- part_random(
  data = abies,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 10)
)
abies2

gbm_t1 <- fit_gbm(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "ppt_jja", "pH", "awc", "depth"),
  predictors_f = c("landform"),
  partition = ".part",
  thr = c("max_sens_spec", "equal_sens_spec", "max_sorensen")
)
gbm_t1$model
gbm_t1$predictors
gbm_t1$performance
gbm_t1$performance_part
gbm_t1$data_ens

# Using bootstrap partition method
abies2 <- part_random(
  data = abies,
  pr_ab = "pr_ab",
  method = c(method = "boot", replicates = 10, proportion = 0.7)
)
abies2

gbm_t2 <- fit_gbm(
  data = abies2,
```

```
  response = "pr_ab",
  predictors = c("ppt_jja", "pH", "awc"),
  predictors_f = c("landform"),
  partition = ".part",
  thr = "max_sens_spec"
)
gbm_t2
```

---

fit_glm *Fit and validate Generalized Linear Models*

---

## Description

Fit and validate Generalized Linear Models

## Usage

```
fit_glm(
  data,
  response,
  predictors,
  predictors_f = NULL,
  select_pred = FALSE,
  partition = NULL,
  thr = NULL,
  fit_formula = NULL,
  poly = 2,
  inter_order = 0
)
```

## Arguments

| | |
|---|---|
| data | data.frame. Database with response (0,1) and predictors values. |
| response | character. Column name with species absence-presence data (0,1). |
| predictors | character. Vector with the column names of quantitative predictor variables (i.e. continuous variables). Usage predictors = c("aet", "cwd", "tmin") |
| predictors_f | character. Vector with the column names of qualitative predictor variables (i.e. ordinal or nominal variables type). Usage predictors_f = c("landform") |
| select_pred | logical. Perform predictor selection. If TRUE predictors will be selected based on backward step wise approach. Default FALSE. |
| partition | character. Column name with training and validation partition groups. If partition = NULL, the model will be validated with the same data used for fitting. |
| thr | character. Threshold used to get binary suitability values (i.e. 0,1), needed for threshold-dependent performance metrics. More than one threshold type can be used. It is necessary to provide a vector for this argument. The following threshold criteria are available: |

- lpt: The highest threshold at which there is no omission.
- equal_sens_spec: Threshold at which the sensitivity and specificity are equal.
- max_sens_spec: Threshold at which the sum of the sensitivity and specificity is the highest (aka threshold that maximizes the TSS).
- max_jaccard: The threshold at which the Jaccard index is the highest.
- max_sorensen: The threshold at which the Sorensen index is highest.
- max_fpb: The threshold at which FPB (F-measure on presence-background data) is highest.
- sensitivity: Threshold based on a specified sensitivity value. Usage thr = c('sensitivity', sens='0.6') or thr = c('sensitivity'). 'sens' refers to sensitivity value. If it is not specified a sensitivity values, function will use by default 0.9

If more than one threshold type is used they must be concatenated, e.g., thr=c('lpt', 'max_sens_spec', 'max_jaccard'), or thr=c('lpt', 'max_sens_spec', 'sensitivity', sens='0.8'), or thr=c('lpt', 'max_sens_spec', 'sensitivity'). Function will use all thresholds if no threshold is specified.

`fit_formula` formula. A formula object with response and predictor variables (e.g. formula(pr_ab ~ aet + ppt_jja + pH + awc + depth + landform)). Note that the variables used here must be consistent with those used in response, predictors, and predictors_f arguments

`poly` integer >= 2. If used with values >= 2 model will use polynomials for those continuous variables (i.e. used in predictors argument). Default is 0.

`inter_order` integer >= 0. The interaction order between explanatory variables. Default is 0.

## Value

A list object with:

- model: A "glm" class object from stats package. This object can be used for predicting.
- predictors: A tibble with quantitative (c column names) and qualitative (f column names) variables use for modeling.
- performance: Performance metrics (see `sdm_eval`). Threshold dependent metric are calculated based on the threshold specified in thr argument.
- performance_part: Performance metric for each replica and partition (see `sdm_eval`).
- data_ens: Predicted suitability for each test partition. This database is used in `fit_ensemble`

## See Also

`fit_gam`, `fit_gau`, `fit_gbm`, `fit_max`, `fit_net`, `fit_raf`, and `fit_svm`.

## Examples

```
# Toy data: a small subset of 'abies' to keep this example fast
data("abies")
abies_toy <- abies[c(1:30, 701:730), ]
```

```
abies_toy <- part_random(
  data = abies_toy,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

glm_toy <- fit_glm(
  data = abies_toy,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin"),
  partition = ".part",
  thr = "max_sens_spec",
  poly = 1
)
glm_toy$performance


data("abies")
abies

# Using k-fold partition method
abies2 <- part_random(
  data = abies,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 5)
)
abies2

glm_t1 <- fit_glm(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "ppt_jja", "pH", "awc", "depth"),
  predictors_f = c("landform"),
  select_pred = FALSE,
  partition = ".part",
  thr = c("max_sens_spec", "equal_sens_spec", "max_sorensen"),
  poly = 0,
  inter_order = 0
)
glm_t1$model
glm_t1$predictors
glm_t1$performance
glm_t1$performance_part
glm_t1$data_ens

# Using second order polynomial terms and first-order interaction terms
glm_t2 <- fit_glm(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "ppt_jja", "pH", "awc", "depth"),
  predictors_f = c("landform"),
  select_pred = FALSE,
  partition = ".part",
```

```
  thr = c("max_sens_spec", "equal_sens_spec", "max_sorensen"),
  poly = 2,
  inter_order = 1
)

# Using repeated k-fold partition method
abies2 <- part_random(
  data = abies,
  pr_ab = "pr_ab",
  method = c(method = "rep_kfold", folds = 3, replicates = 5)
)
abies2

# Using third order polynomial terms and second-order interaction terms
glm_t3 <- fit_glm(
  data = abies2,
  response = "pr_ab",
  predictors = c("ppt_jja", "pH", "awc"),
  predictors_f = c("landform"),
  select_pred = FALSE,
  partition = ".part",
  thr = c("max_sens_spec", "equal_sens_spec", "max_sorensen"),
  poly = 3,
  inter_order = 2
)
```

---

`fit_max` *Fit and validate Maximum Entropy models*

---

## Description

Fit and validate Maximum Entropy models

## Usage

```
fit_max(
  data,
  response,
  predictors,
  predictors_f = NULL,
  fit_formula = NULL,
  partition = NULL,
  background = NULL,
  thr = NULL,
  clamp = TRUE,
  classes = "default",
  pred_type = "cloglog",
  regmult = 1
)
```

### Arguments

`data` data.frame. Database with response (0,1) and predictors values.

`response` character. Column name with species absence-presence data (0,1).

`predictors` character. Vector with the column names of quantitative predictor variables (i.e. continuous variables). Usage predictors = c("aet", "cwd", "tmin")

`predictors_f` character. Vector with the column names of qualitative predictor variables (i.e. ordinal or nominal variables type). Usage predictors_f = c("landform")

`fit_formula` formula. A formula object with response and predictor variables. See maxnet.formula function from maxnet package. Note that the variables used here must be consistent with those used in response, predictors, and predictors_f arguments. Default NULL.

`partition` character. Column name with training and validation partition groups. If partition = NULL, the model will be validated with the same data used for fitting.

`background` data.frame. Database including only those rows with 0 values in the response column and the predictors variables. All column names must be consistent with data. Default NULL

`thr` character. Threshold used to get binary suitability values (i.e. 0,1), needed for threshold-dependent performance metrics. More than one threshold type can be used. It is necessary to provide a vector for this argument. The following threshold criteria are available:

- lpt: The highest threshold at which there is no omission.
- equal_sens_spec: Threshold at which the sensitivity and specificity are equal.
- max_sens_spec: Threshold at which the sum of the sensitivity and specificity is the highest (aka threshold that maximizes the TSS).
- max_jaccard: The threshold at which the Jaccard index is the highest.
- max_sorensen: The threshold at which the Sorensen index is highest.
- max_fpb: The threshold at which FPB (F-measure on presence-background data) is highest.
- sensitivity: Threshold based on a specified sensitivity value. Usage thr = c('sensitivity', sens='0.6') or thr = c('sensitivity'). 'sens' refers to sensitivity value. If a sensitivity values is not specified the default used is 0.9.

If more than one threshold type is used they must be concatenated, e.g., thr=c('lpt', 'max_sens_spec', 'max_jaccard'), or thr=c('lpt', 'max_sens_spec', 'sensitivity', sens='0.8'), or thr=c('lpt', 'max_sens_spec', 'sensitivity'). Function will use all thresholds if no threshold is specified.

`clamp` logical. If TRUE, predictors and features are restricted to the range seen during model training.

`classes` character. A single feature of any combinations of them. Features are symbolized by letters: l (linear), q (quadratic), h (hinge), p (product), and t (threshold). Usage classes = "lpq". Default "default" (see details).

`pred_type` character. Type of response required available "link", "exponential", "cloglog" and "logistic". Default "cloglog"

`regmult` numeric. A constant to adjust regularization. Default 1.

## Details

When the argument "classes" is set as default MaxEnt will use different features combination depending of the number of presences (np) with the follow rule: if np < 10 classes = "l", if np between 10 and 15 classes = "lq", if np between 15 and 80 classes = "lqh", and if np >= 80 classes = "lqph"

When presence-absence (or presence-pseudo-absence) data are used in data argument in addition to background points, the function will fit models with presences and background points and validate with presences and absences. This procedure makes maxent comparable to other presences-absences models (e.g., random forest, support vector machine). If only presences and background points data are used, function will fit and validate model with presences and background data. If only presence-absences are used in data argument and without background, function will fit model with the specified data (not recommended).

## Value

A list object with:

- model: A "maxnet" class object from maxnet package. This object can be used for predicting.
- predictors: A tibble with quantitative (c column names) and qualitative (f column names) variables use for modeling.
- performance: Performance metrics (see sdm_eval). Threshold dependent metrics are calculated based on the threshold specified in thr argument.
- performance_part: Performance metric for each replica and partition (see sdm_eval).
- data_ens: Predicted suitability for each test partition based on the best model. This database is used in fit_ensemble

## See Also

fit_gam, fit_gau, fit_gbm, fit_glm, fit_net, fit_raf, and fit_svm.

## Examples

```
# Toy data: a small subset of 'abies' to keep this example fast
data("abies")
abies_toy <- abies[c(1:30, 701:730), ]
abies_toy <- part_random(
  data = abies_toy,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

# Use the same records as background points (toy example only)
backg_toy <- abies_toy
backg_toy$pr_ab <- 0

max_toy <- fit_max(
  data = abies_toy,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin"),
  partition = ".part",
```

```
  background = backg_toy,
  thr = "max_sens_spec",
  clamp = TRUE,
  classes = "lq",
  pred_type = "cloglog",
  regmult = 1
)
max_toy$performance


data("abies")
data("backg")
set.seed(1)
backg <- backg[sample(nrow(backg), 1000), ] # subsample to speed up this example
abies # environmental conditions of presence-absence data
backg # environmental conditions of background points

# Using k-fold partition method
# Note that the partition method, number of folds or replications must
# be the same for presence-absence and background points datasets
abies2 <- part_random(
  data = abies,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)
abies2

backg2 <- part_random(
  data = backg,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)
backg2

max_t1 <- fit_max(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "ppt_jja", "pH", "awc", "depth"),
  predictors_f = c("landform"),
  partition = ".part",
  background = backg2,
  thr = c("max_sens_spec", "equal_sens_spec", "max_sorensen"),
  clamp = TRUE,
  classes = "default",
  pred_type = "cloglog",
  regmult = 1
)
length(max_t1)

max_t1$model
max_t1$predictors
max_t1$performance
max_t1$performance_part
```

```
max_t1$data_ens
```

---

`fit_net` *Fit and validate Neural Networks models*

---

## Description

Fit and validate Neural Networks models

## Usage

```
fit_net(
  data,
  response,
  predictors,
  predictors_f = NULL,
  fit_formula = NULL,
  partition = NULL,
  thr = NULL,
  size = 2,
  decay = 0.1
)
```

## Arguments

| | |
|---|---|
| `data` | data.frame. Database with response (0,1) and predictors values. |
| `response` | character. Column name with species absence-presence data (0,1). |
| `predictors` | character. Vector with the column names of quantitative predictor variables (i.e. continuous variables). Usage predictors = c("aet", "cwd", "tmin") |
| `predictors_f` | character. Vector with the column names of qualitative predictor variables (i.e. ordinal or nominal variables type). Usage predictors_f = c("landform") |
| `fit_formula` | formula. A formula object with response and predictor variables (e.g. formula(pr_ab ~ aet + ppt_jja + pH + awc + depth + landform)). Note that the variables used here must be consistent with those used in response, predictors, and predictors_f arguments. Defaul NULL. |
| `partition` | character. Column name with training and validation partition groups. If partition = NULL, the model will be validated with the same data used for fitting. |
| `thr` | character. Threshold used to get binary suitability values (i.e. 0,1)., needed for threshold-dependent performance metrics. More than one threshold type can be specified. It is necessary to provide a vector for this argument. The following threshold criteria are available:<br>• lpt: The highest threshold at which there is no omission.<br>• equal_sens_spec: Threshold at which the sensitivity and specificity are equal. |

- max_sens_spec: Threshold at which the sum of the sensitivity and specificity is the highest (aka threshold that maximizes the TSS).
- max_jaccard: The threshold at which the Jaccard index is the highest.
- max_sorensen: The threshold at which the Sorensen index is highest.
- max_fpb: The threshold at which FPB (F-measure on presence-background data) is highest.
- sensitivity: Threshold based on a specified sensitivity value. Usage thr = c('sensitivity', sens='0.6') or thr = c('sensitivity'). 'sens' refers to sensitivity value. If a sensitivity value is not specified, the default is 0.9

If more than one threshold type is used they must be concatenated, e.g., thr=c('lpt', 'max_sens_spec', 'max_jaccard'), or thr=c('lpt', 'max_sens_spec', 'sensitivity', sens='0.8'), or thr=c('lpt', 'max_sens_spec', 'sensitivity'). Function will use all thresholds if no threshold is specified.

`size` numeric. Number of units in the hidden layer. Can be zero if there are skip-layer units. Default 2.

`decay` numeric. Parameter for weight decay. Default 0.1.

## Value

A list object with:

- model: A "nnet.formula" "nnet" class object from nnet package. This object can be used for predicting.
- predictors: A tibble with quantitative (c column names) and qualitative (f column names) variables use for modeling.
- performance: Performance metrics (see `sdm_eval`). Threshold dependent metric are calculated based on the threshold specified in the argument.
- performance_part: Performance metric for each replica and partition (see `sdm_eval`).
- data_ens: Predicted suitability for each test partition. This database is used in `fit_ensemble`

## See Also

`fit_gam`, `fit_gau`, `fit_gbm`, `fit_glm`, `fit_max`, `fit_raf`, and `fit_svm`.

## Examples

```
# Toy data: a small subset of 'abies' to keep this example fast
data("abies")
abies_toy <- abies[c(1:30, 701:730), ]
abies_toy <- part_random(
  data = abies_toy,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

net_toy <- fit_net(
  data = abies_toy,
  response = "pr_ab",
```

```
  predictors = c("aet", "cwd", "tmin"),
  partition = ".part",
  thr = "max_sens_spec",
  size = 2,
  decay = 1
)
net_toy$performance


data("abies")

# Using k-fold partition method
abies2 <- part_random(
  data = abies,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 10)
)
abies2

nnet_t1 <- fit_net(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "ppt_jja", "pH", "awc", "depth"),
  predictors_f = c("landform"),
  partition = ".part",
  thr = c("max_sens_spec", "equal_sens_spec", "max_sorensen"),
  fit_formula = NULL
)

nnet_t1$model
nnet_t1$predictors
nnet_t1$performance
nnet_t1$performance_part
nnet_t1$data_ens

# Using bootstrap partition method and only with presence-absence
# and get performance for several method
abies2 <- part_random(
  data = abies,
  pr_ab = "pr_ab",
  method = c(method = "boot", replicates = 10, proportion = 0.7)
)
abies2

nnet_t2 <- fit_net(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "ppt_jja", "pH", "awc", "depth"),
  predictors_f = c("landform"),
  partition = ".part",
  thr = c("max_sens_spec", "equal_sens_spec", "max_sorensen"),
  fit_formula = NULL
)
```

nnet_t2

---

fit_raf *Fit and validate Random Forests models*

---

## Description

Fit and validate Random Forests models

## Usage

```
fit_raf(
  data,
  response,
  predictors,
  predictors_f = NULL,
  fit_formula = NULL,
  partition = NULL,
  thr = NULL,
  mtry = sqrt(length(c(predictors, predictors_f))),
  ntree = 500
)
```

## Arguments

| | |
|---|---|
| data | data.frame. Database with response (0,1) and predictors values. |
| response | character. Column name with species absence-presence data (0,1). |
| predictors | character. Vector with the column names of quantitative predictor variables (i.e. continuous variables). Usage predictors = c("aet", "cwd", "tmin") |
| predictors_f | character. Vector with the column names of qualitative predictor variables (i.e. ordinal or nominal variables type). Usage predictors_f = c("landform") |
| fit_formula | formula. A formula object with response and predictor variables (e.g. formula(pr_ab ~ aet + ppt_jja + pH + awc + depth + landform)). Note that the variables used here must be consistent with those used in response, predictors, and predictors_f arguments. Default NULL |
| partition | character. Column name with training and validation partition groups. If partition = NULL, the model will be validated with the same data used for fitting. |
| thr | character. Threshold used to get binary suitability values (i.e. 0,1), needed for threshold-dependent performance metrics. More than one threshold type can be used. It is necessary to provide a vector for this argument. The following threshold criteria are available:<br>• lpt: The highest threshold at which there is no omission.<br>• equal_sens_spec: Threshold at which the sensitivity and specificity are equal. |

- max_sens_spec: Threshold at which the sum of the sensitivity and specificity is the highest (aka threshold that maximizes the TSS).
- max_jaccard: The threshold at which Jaccard is the highest.
- max_sorensen: The threshold at which Sorensen is highest.
- max_fpb: The threshold at which FPB (F-measure on presence-background data) is highest.
- sensitivity: Threshold based on a specified sensitivity value. Usage thr = c('sensitivity', sens='0.6') or thr = c('sensitivity'). 'sens' refers to sensitivity value. If it is not specified a sensitivity values, function will use by default 0.9

If more than one threshold type is used they must be concatenated, e.g., thr=c('lpt', 'max_sens_spec', 'max_jaccard'), or thr=c('lpt', 'max_sens_spec', 'sensitivity', sens='0.8'), or thr=c('lpt', 'max_sens_spec', 'sensitivity'). Function will use all thresholds if no threshold is specified.

`mtry` numeric. Number of variables randomly sampled as candidates at each split. Default sqrt(length(c(predictors, predictors_f)))

`ntree` numeric. Number of trees to grow. Default 500

## Value

A list object with:

- model: A "randomForest" class object from randomForest package. This object can be used for predicting.
- predictors: A tibble with quantitative (c column names) and qualitative (f column names) variables use for modeling.
- performance: Performance metrics (see `sdm_eval`). Threshold dependent metrics are calculated based on the threshold specified in the argument.
- performance_part: Performance metric for each replica and partition (see `sdm_eval`).
- data_ens: Predicted suitability for each test partition. This database is used in `fit_ensemble`

## See Also

`fit_gam`, `fit_gau`, `fit_gbm`, `fit_glm`, `fit_max`, `fit_net`, and `fit_svm`.

## Examples

```
# Toy data: a small subset of 'abies' to keep this example fast
data("abies")
abies_toy <- abies[c(1:30, 701:730), ]
abies_toy <- part_random(
  data = abies_toy,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

raf_toy <- fit_raf(
  data = abies_toy,
```

```
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin"),
  partition = ".part",
  thr = "max_sens_spec",
  mtry = 2,
  ntree = 50
)
raf_toy$performance


data("abies")

# Using k-fold partition method
abies2 <- part_random(
  data = abies,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 10)
)
abies2

rf_t1 <- fit_raf(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "ppt_jja", "pH", "awc", "depth"),
  predictors_f = c("landform"),
  partition = ".part",
  thr = c("max_sens_spec", "equal_sens_spec", "max_sorensen"),
  fit_formula = NULL
)

rf_t1$model
rf_t1$predictors
rf_t1$performance
rf_t1$performance_part
rf_t1$data_ens

# Using bootstrap partition method and only with presence-absence
# and get performance for several method
abies2 <- part_random(
  data = abies,
  pr_ab = "pr_ab",
  method = c(method = "boot", replicates = 10, proportion = 0.7)
)
abies2

rf_t2 <- fit_raf(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "ppt_jja", "pH", "awc", "depth"),
  predictors_f = c("landform"),
  partition = ".part",
  thr = c("max_sens_spec", "equal_sens_spec", "max_sorensen"),
  fit_formula = NULL,
```

```
  mtry = 2,
  ntree = 500
)
rf_t2
```

---

fit_svm *Fit and validate Support Vector Machine models*

---

## Description

Fit and validate Support Vector Machine models

## Usage

```
fit_svm(
  data,
  response,
  predictors,
  predictors_f = NULL,
  fit_formula = NULL,
  partition = NULL,
  thr = NULL,
  sigma = "automatic",
  C = 1
)
```

## Arguments

| | |
|---|---|
| data | data.frame. Database with response (0,1) and predictors values. |
| response | character. Column name with species absence-presence data (0,1). |
| predictors | character. Vector with the column names of quantitative predictor variables (i.e. continuous variables). Usage predictors = c("aet", "cwd", "tmin") |
| predictors_f | character. Vector with the column names of qualitative predictor variables (i.e. ordinal or nominal variables type). Usage predictors_f = c("landform") |
| fit_formula | formula. A formula object with response and predictor variables (e.g. formula(pr_ab ~ aet + ppt_jja + pH + awc + depth + landform)). Note that the variables used here must be consistent with those used in response, predictors, and predictors_f arguments |
| partition | character. Column name with training and validation partition groups. If partition = NULL, the model will be validated with the same data used for fitting. |
| thr | character. Threshold used to get binary suitability values (i.e. 0,1) needed for threshold-dependent performance metrics. More than one threshold type can be used. It is necessary to provide a vector for this argument. The following threshold criteria are available: |

- lpt: The highest threshold at which there is no omission.
- equal_sens_spec: Threshold at which the sensitivity and specificity are equal.
- max_sens_spec: Threshold at which the sum of the sensitivity and specificity is the highest (aka threshold that maximizes the TSS).
- max_jaccard: The threshold at which the Jaccard index is the highest.
- max_sorensen: The threshold at which the Sorensen index is highest.
- max_fpb: The threshold at which FPB (F-measure on presence-background data) is highest.
- sensitivity: Threshold based on a specified sensitivity value. Usage thr = c('sensitivity', sens='0.6') or thr = c('sensitivity'). 'sens' refers to sensitivity value. If a sensitivity value is not specified, the default used is 0.9

If more than one threshold type is used they must be concatenated, e.g., thr=c('lpt', 'max_sens_spec', 'max_jaccard'), or thr=c('lpt', 'max_sens_spec', 'sensitivity', sens='0.8'), or thr=c('lpt', 'max_sens_spec', 'sensitivity'). Function will use all thresholds if no threshold is specified.

`sigma` numeric. Inverse kernel width for the Radial Basis kernel function "rbfdot". Default "automatic".

`C` numeric. Cost of constraints violation, the 'C'-constant of the regularization term in the Lagrange formulation. Default 1

## Details

This function constructs 'C-svc' classification type and uses Radial Basis kernel "Gaussian" function (rbfdot). See details details in ksvm.

## Value

A list object with:

- model: A "ksvm" class object from kernlab package. This object can be used for predicting.
- predictors: A tibble with quantitative (c column names) and qualitative (f column names) variables use for modeling.
- performance: Performance metric (see `sdm_eval`). Threshold dependent metrics are calculated based on the threshold specified in the argument.
- performance_part: Performance metric for each replica and partition (see `sdm_eval`).
- data_ens: Predicted suitability for each test partition based on the best model. This database is used in `fit_ensemble`

## See Also

`fit_gam`, `fit_gau`, `fit_gbm`, `fit_glm`, `fit_max`, `fit_net`, and `fit_raf`.

## Examples

```
# Toy data: a small subset of 'abies' to keep this example fast
data("abies")
abies_toy <- abies[c(1:30, 701:730), ]
abies_toy <- part_random(
  data = abies_toy,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

svm_toy <- fit_svm(
  data = abies_toy,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin"),
  partition = ".part",
  thr = "max_sens_spec"
)
svm_toy$performance


data("abies")

# Using k-fold partition method
abies2 <- part_random(
  data = abies,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 5)
)
abies2

svm_t1 <- fit_svm(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "ppt_jja", "pH", "awc", "depth"),
  predictors_f = c("landform"),
  partition = ".part",
  thr = c("max_sens_spec", "equal_sens_spec", "max_sorensen"),
  fit_formula = NULL
)

names(svm_t1)
svm_t1$model
svm_t1$predictors
svm_t1$performance
svm_t1$performance_part
svm_t1$data_ens

# Using bootstrap partition method and only with presence-absence
# and get performance for several method
abies2 <- part_random(
  data = abies,
  pr_ab = "pr_ab",
```

```
  method = c(method = "boot", replicates = 10, proportion = 0.7)
)
abies2

svm_t2 <- fit_svm(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "ppt_jja", "pH", "awc", "depth"),
  predictors_f = c("landform"),
  partition = ".part",
  thr = c("max_sens_spec", "equal_sens_spec", "max_sorensen"),
  fit_formula = NULL
)
svm_t2
```

---

`get_absences` *Build a presence-absence database from multi-species occurrence data*

---

## Description

Converts a database containing occurrence records for multiple species into a presence-absence database, either for every species present in the data or for a specified subset of target species. For each target species, its own records are coded as presences and the records of all other species are recoded as absences for that species.

## Usage

```
get_absences(
  data,
  x = "x",
  y = "y",
  species = "species",
  target_species = NULL,
  pr_ab_name = "pr_ab"
)
```

## Arguments

| | |
|---|---|
| `data` | data.frame or tibble. A table with occurrence records for one or more species, including columns for species identity and x/y coordinates. |
| `x` | character. Name of the column in `data` with the x (longitude) coordinate. Default `"x"`. |
| `y` | character. Name of the column in `data` with the y (latitude) coordinate. Default `"y"`. |
| `species` | character. Name of the column in `data` with the species name/identifier. Default `"species"`. |

| | |
|---|---|
| target_species | character vector. One or more species names to build presence-absence data for. If NULL (default), a presence-absence database is built for every unique species found in data[[species]]. |
| pr_ab_name | character. Name to give the presence-absence column in the output, coded 1 for presences and 0 for absences. Default "pr_ab". |

## Details

Only the species, x, and y columns are retained in the output; any other columns present in data (e.g., environmental covariates or metadata) are dropped. If you need those columns downstream, re-join them after calling this function.

Because absences for a given target species are drawn from every other species' occurrence records, duplicate x/y coordinates across species in the input will produce duplicate absence coordinates in the output.

## Value

A tibble with columns species, x, y (using the names supplied in the species, x, and y arguments) and the presence-absence column named according to pr_ab_name. For each species in target_species (or each unique species in data, if target_species = NULL), the output contains one presence-absence block: records belonging to that species are coded 1, and records of all other species are recoded as belonging to that species with a value of 0. Blocks for all target species are stacked row-wise, so the returned tibble has nrow(data) * length(sp) rows, where sp is the set of target species.

## Examples

```
data <- data.frame(
  species = c("sp1", "sp1", "sp2", "sp2", "sp3"),
  x = c(-74.1, -74.2, -73.9, -73.8, -74.0),
  y = c(4.6, 4.65, 4.7, 4.72, 4.68)
)

# Presence-absence database for every species in the data
get_absences(data,
  x = "x", y = "y", species = "species",
  target_species = NULL, pr_ab_name = "pr_ab"
)

# Presence-absence database for a single target species
get_absences(data,
  x = "x", y = "y", species = "species",
  target_species = "sp1", pr_ab_name = "pr_ab"
)

# Presence-absence database for a subset of species, with a custom
# presence-absence column name
get_absences(data,
  x = "x", y = "y", species = "species",
  target_species = c("sp1", "sp2"), pr_ab_name = "occ"
)
```

get_block *Transform a spatial partition layer to the same spatial properties as environmental variables*

## Description

Transform a spatial partition layer to the same spatial properties as environmental variables

## Usage

```
get_block(env_layer, best_grid)
```

## Arguments

| | |
|---|---|
| env_layer | SpatRaster object with some environmental variables used in the block_partition or band_partition function. Function always will select the first layer |
| best_grid | SpatRaster object returned by block_partition or band_partition |

## Details

Transform a layer originating from the function block_partition or band_partition to the same spatial properties as the environmental variables

## Value

A SpatRaster layer with the same resolution and extent as the environmental variables

## Examples

```
# Toy data: small raster and records of one species to keep this example fast
data("spp")
somevar <- terra::rast(system.file("external/somevar.tif", package = "flexsdm"))
somevar <- terra::aggregate(somevar, fact = 6) # coarser resolution
sp3 <- spp[spp$species == "sp3", ]

sp_part <- part_sblock(
  env_layer = somevar,
  data = sp3,
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  min_res_mult = 10,
  max_res_mult = 40,
  num_grids = 3,
  n_part = 2,
  min_occ = 2,
  prop = 0.5
)
```

```
# Block layer with the same properties as the environmental variables
block_layer <- get_block(env_layer = somevar, best_grid = sp_part$grid)
block_layer


require(dplyr)
require(terra)
data(spp)
f <- system.file("external/somevar.tif", package = "flexsdm")
somevar <- terra::rast(f)

# Example for a single species
single_spp <- spp %>% dplyr::filter(species == "sp3")

part <- part_sblock(
  env_layer = somevar,
  data = single_spp,
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  min_res_mult = 100,
  max_res_mult = 500,
  num_grids = 10,
  min_occ = 5,
  n_part = 2
)

grid_env <- get_block(env_layer = somevar, best_grid = part$grid)
grid_env
part$grid

plot(part$grid)
plot(grid_env)
```

---

`hespero` *A data set containing localities of Hesperocyparis stephensonii species in California, USA*

---

## Description

A data set containing localities of Hesperocyparis stephensonii species in California, USA

## Usage

```
hespero
```

### Format

A tibble object with 14 rows and 4 variables:

**ID** presences records ID

**x y** columns with coordinates in Albers Equal Area Conic coordinate system

**pr_ab** presence denoted by 1 ...

### Examples

```
require(dplyr)
data("hespero")
hespero
```

---

## homogenize_na *Homogenize cells with NAs across all layers*

---

### Description

Homogenize cells with NAs across all layers

### Usage

```
homogenize_na(x)
```

### Arguments

| | |
|---|---|
| x | A SpatRaster. |

### Details

Homogenize cells with NAs across all layers in a SpatRaster so that in the resulting SpatRaster all layers have the same cells with NAa

### Value

a SpatRaster

### Examples

```
somevar <- terra::rast(system.file("external/somevar.tif", package = "flexsdm"))
somevar <- terra::aggregate(somevar, fact = 6) # coarser resolution to keep this example fast

somevar2 <- homogenize_na(somevar)

# In somevar2 all layers have the same cells with NAs
terra::global(somevar2, "notNA")
```

```
require(terra)

somevar <- system.file("external/somevar.tif", package = "flexsdm")
somevar <- terra::rast(somevar)

somevar2 <- homogenize_na(somevar)
oldpar <- par(mfrow = c(2, 1))
plot(somevar$CFP_4)
plot(somevar2$CFP_4)
par(oldpar) # restore graphical parameters

# In somevar2 all layers have the same cells with NAs
```

---

`interp` *Raster interpolation between two time periods*

---

## Description

This function interpolates values for each year between two specified years with simple interpolation using two raster objects containing e.g. habitat suitability values predicted using a species distribution model.

## Usage

```
interp(r1, r2, y1, y2, rastername = NULL, dir_save = NULL)
```

## Arguments

| | |
|---|---|
| `r1` | SpatRaster. Raster object for the initial year |
| `r2` | SpatRaster. Raster object for the final year |
| `y1` | numeric. Initial year |
| `y2` | numeric. Final year |
| `rastername` | character. Word used as prefix in raster file name. Default NULL |
| `dir_save` | character. Directory path and name of the folder in which the raster files will be saved. If NULL, function will return a SpatRaster object, else, it will save raster in a given directory. Default NULL |

## Details

This function interpolates suitability values assuming that annual changes in suitability are linear. This function could be useful for linking SDM output based on averaged climate data and climate change scenarios to other models that require suitability values disaggregated in time periods, such as population dynamics (Keith et al., 2008; Conlisk et al., 2013; Syphard et al., 2013).

## Value

If dir_save is NULL, the function returns a SpatRaster with suitability interpolation for each year. If dir_save is used, function outputs are saved in the directory specified in dir_save.

## References


- Keith, D.A., Akçakaya, H.R., Thuiller, W., Midgley, G.F., Pearson, R.G., Phillips, S.J., Regan, H.M., Araujo, M.B. & Rebelo, T.G. (2008) Predicting extinction risks under climate change: coupling stochastic population models with dynamic bioclimatic habitat models. Biology Letters, 4, 560-563.
- Conlisk, E., Syphard, A.D., Franklin, J., Flint, L., Flint, A. & Regan, H.M. (2013) Management implications of uncertainty in assessing impacts of multiple landscape-scale threats to species persistence using a linked modeling approach. Global Change Biology 3, 858-869.
- Syphard, A.D., Regan, H.M., Franklin, J. & Swab, R. (2013) Does functional type vulnerability to multiple threats depend on spatial context in Mediterranean-climate regions? Diversity and Distributions, 19, 1263-1274.


## Examples

```
f <- system.file("external/suit_time_step.tif", package = "flexsdm")
abma <- terra::rast(f)

int <- interp(
  r1 = abma[[1]],
  r2 = abma[[2]],
  y1 = 2010,
  y2 = 2020,
  rastername = "Abies",
  dir_save = NULL
)
int


require(terra)
require(dplyr)

f <- system.file("external/suit_time_step.tif", package = "flexsdm")
abma <- terra::rast(f)
plot(abma)

int <- interp(
  r1 = abma[[1]],
  r2 = abma[[2]],
  y1 = 2010,
  y2 = 2020,
  rastername = "Abies",
  dir_save = NULL
)

int
```

`map_env_dist` *Calculate environmental distance between presences and projection data*

## Description

Calculate environmental distance between presences and projection data

## Usage

```
map_env_dist(training_data, projection_data, metric = "domain", n_cores = 1)
```

## Arguments

`training_data` data.frame or tibble with environmental conditions of presence used for constructing models

`projection_data` SpatRaster, data.frame or tibble with environmental condition used for projecting a model (e.g., a larger, encompassing region, a spatially separate region, or a different time period). If data.frame or tibble is used function will return a tibble object. Otherwise, as SpatRaster object.

`metric` character. Metric used for measuring distance. Default 'domain'.

- domain: Gower distance between presences and projection data. The result is a similarity value that ranges from 0 to 1.
- euclidean: Euclidean distance. This metric Z-score standardizes the projection data based on the training data and then standardizes the resulting distance by the maximum pairwise distance within the training data. The result is a similarity value that ranges from 0 to 1.
- mahalanobis: Mahalanobis distance. This metric Z-score standardizes the projection data based on the training data and then standardizes the resulting distance by the maximum pairwise distance within the training data. The result is a similarity value that ranges from 0 to 1.

`n_cores` numeric. Number of cores to use for parallel processing when metric is "domain". Default 1 (no parallelization).

## Value

A SpatRaster or tibble object with the nearest environmental distance between presences and projection data. So far the Domain algorithm (based on the Gower distance; Carpenter et al., 1993), Euclidean distance, and Mahalanobis distance have been implemented. For Euclidean and Mahalanobis distances, the result is a similarity value ranging from 0 to 1, where values close to 1 represent high similarity (low distance) and values close to 0 represent low similarity (high distance).

## References


- Carpenter, G., Gillison, A.N., Winter, J., 1993. DOMAIN: a flexible modelling procedure for mapping potential distributions of plants and animals. Biodiversity & Conservation 2, 667–680


## Examples

```
# Toy data: small raster and records of one species to keep this example fast
data("spp")
somevar <- terra::rast(system.file("external/somevar.tif", package = "flexsdm"))
somevar <- terra::aggregate(somevar, fact = 6) # coarser resolution
sp3 <- spp[spp$species == "sp3", ]
somevar <- somevar[[1:2]] # two variables
names(somevar) <- c("aet", "cwd")
sp3 <- sdm_extract(data = sp3[sp3$pr_ab == 1, ], x = "x", y = "y", env_layer = somevar)

# Environmental distance between presences and the projection data
envdist <- map_env_dist(
  training_data = sp3,
  projection_data = somevar,
  metric = "euclidean"
)
envdist


require(dplyr)
require(terra)
require(ggplot2)
data(spp)
f <- system.file("external/somevar.tif", package = "flexsdm")
somevar <- terra::rast(f)

# Let's use only two variables to turn more evident the pater in the environmental space
somevar <- somevar[[1:2]]
names(somevar) <- c("aet", "cwd")


spp$species %>% unique()
sp <- spp %>%
  dplyr::filter(species == "sp3", pr_ab == 1) %>%
  dplyr::select(x, y, pr_ab)

# Get environmental condition of presences
sp_pa_2 <- sdm_extract(
  data = sp,
  x = "x",
  y = "y",
  env_layer = somevar
)
sp_pa_2

# Measure environmental distance between presences and projection data
```

```
clrs <- c("#000033", "#1400FF", "#C729D6", "#FF9C63", "#FFFF60")


# Domain
envdist <-
  map_env_dist(
    training_data = sp_pa_2,
    projection_data = somevar,
    metric = "domain"
  )
plot(envdist, main = "Domain")
p_extra(
  training_data = sp_pa_2,
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  extra_suit_data = envdist,
  projection_data = somevar,
  geo_space = FALSE,
  prop_points = 0.8,
  alpha_p = 0.9,
  color_p = "red",
  color_gradient = clrs
)


# Euclidean
envdist <-
  map_env_dist(
    training_data = sp_pa_2,
    projection_data = somevar,
    metric = "euclidean"
  )
p_extra(
  training_data = sp_pa_2,
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  extra_suit_data = envdist,
  projection_data = somevar,
  geo_space = TRUE,
  prop_points = 0.8,
  alpha_p = 0.9,
  color_p = "red",
  color_gradient = clrs
) +
  labs(title = "Euclidean")


# Mahalanobis
envdist <-
  map_env_dist(
    training_data = sp_pa_2,
```

```
    projection_data = somevar,
    metric = "mahalanobis"
  )
p_extra(
  training_data = sp_pa_2,
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  extra_suit_data = envdist,
  projection_data = somevar,
  geo_space = TRUE,
  prop_points = 0.8,
  alpha_p = 0.9,
  color_p = "red",
  color_gradient = clrs
) +
  labs(title = "Mahalanobis")
```

---

`msdm_posteriori` *Methods to correct overprediction of species distribution models based on occurrences and suitability patterns.*

---

## Description

These methods reduce overprediction of species distribution models based on a posteriori methods (see Mendes et al 2020), i.e., the combination of the patterns of species occurrences and predicted suitability

## Usage

```
msdm_posteriori(
  records,
  x,
  y,
  pr_ab,
  cont_suit,
  method = c("obr", "pres", "lq", "mcp", "bmcp"),
  pres_as_patch = FALSE,
  thr = "equal_sens_spec",
  con_thr = FALSE,
  buffer = NULL,
  crs = NULL
)
```

## Arguments

`records` tibble or data.frame. A database with spatial coordinates of species presences and absences (or pseudo-absence) used to create species distribution models.

`x` character. Column name with spatial x coordinates.

`y` character. Column name with spatial y coordinates.

`pr_ab` character. Column name with presence and absence data (i.e. 1 and 0)

`cont_suit` SpatRaster. Raster with continuous suitability predictions "species_specific" type calculates the minimum pairwise-distances between all occurrences and then selects the maximum distance, i.e., the value of the buffer will be the maximum distance from the minimum distance. This procedure depends on the spatial pattern of each species' occurrences; thus, for each species, a value of buffer width will be calculated (usage buffer="species_specific").

`method` character. A character string indicating which constraint method will be used (see in details).

`pres_as_patch` character. For the 'lq' method, assume that cells with presences but below the threshold are patches. Default FALSE.

`thr` character or numeric. Threshold used to get binary suitability values (i.e. 0,1), needed for threshold-dependent performance metrics. Only one threshold type can be specified. It is necessary to provide a vector for this argument. The following threshold criteria are available:

- lpt: The highest threshold at which there is no omission.
- equal_sens_spec: Threshold at which the sensitivity and specificity are equal.
- max_sens_spec: Threshold at which the sum of the sensitivity and specificity is the highest (aka threshold that maximizes the TSS).
- max_jaccard: The threshold at which the Jaccard index is the highest.
- max_sorensen: The threshold at which the Sorensen index is highest.
- max_fpb: The threshold at which FPB is highest.
- sensitivity: Threshold based on a specified sensitivity value. Usage thr = c('sensitivity', sens='0.6') or thr = c('sensitivity'). 'sens' refers to sensitivity value. If it is not specified a sensitivity values, function will use by default 0.9
- Also, it is possible to specify the threshold value using a numeric value (thr = 0.623)
  Default "equal_sens_spec".

`con_thr` logical. If TRUE, returns continuous suitability values for cells above the threshold, with other cells as 0. If False, returns a binary map (1 for above threshold, 0 for below). Default FALSE.

`buffer` numeric. Buffer width use in 'bmcp' approach. The buffer width will be interpreted in m if Coordinate reference system used in "crs" argument has a longitude/latitude, or map units in other cases. Usage buffer=50000. Default NULL

`crs` character. Coordinate reference system used for calculating buffer in "bmcp" method.

## Details

These function help reduce overprediction of species distribution models based on the combination of the patterns of species occurrences and predicted suitability. It is recommended to use these

approaches only for current distribution not for models projected for different time periods (past or future).

Five methods are implemented:

Abbreviation list

- SDM: species distribution model
- l: suitability patches that intercept species occurrences
- k: suitability patches that do not intercept species occurrences
- T: threshold distances used to select suitability patches

These methods reduce overprediction of species distribution models already fitted based on the occurrences and suitability patterns of species (see 'thr' arguments)

Method 'obr' (Occurrences Based Restriction). This method assumes that suitable patches intercepting species occurrences (l) are more likely to be part of species distributions than suitable patches that do not intercept any occurrence (k). Distance from all k patches to the closest l patch is calculated, then k patches are removed that exceed a species-specific distance threshold from SDMs models. This threshold (T) is calculated as the maximum distance in a vector of minimum pairwise distances between occurrences. Whenever a suitable pixel is within a k patch that is more than distance T from the closest l patch, the suitability of the pixel is reduced to zero. It is assumed that this simple threshold is a surrogate for the species-specific dispersal ability. If T is low, either the species has been sampled throughout its distribution, or the species is geographically restricted, justifying a narrow inclusion of k patches (Mendes et al., 2020).

Method 'lq' (Lower Quantile). This method is similar to the 'obr' method, except by the procedure to define a distance threshold to withdrawn k patches, which is the lower quartile distance between k patches to the closest l patch (i.e., threshold distance is based on patches with and without presences and not in presences points as in 'obr' method) . Whenever a suitable pixel is within a k patch, i.e., not within this lower quartile, the suitability of the pixel is reduced to zero. This means that 75% of k patches were withdrawn from the model (Mendes et al., 2020).

Method 'pres' (only occurrences based restriction). This is a more restrictive variant of the 'obr' method. It only retains those pixels in suitability patches intercepting occurrences (k) (Mendes et al., 2020).

Method 'mcp' (Minimum Convex Polygon). Compiled and adapted from Kremen et al. (2008), this method excludes from SDM predictions suitable pixels that do not intercept a minimum convex polygon, with interior angles smaller than 180, enclosing all occurrences of a species.

Method 'bmcp' (Buffered Minimum Convex Polygon). Compiled and adapted from Kremen et al. (2008), it is similar to the 'mcp' method except for the inclusion of a buffer zone surrounding minimum convex polygons. For this method a buffer width value must be provided in "buffer" argument and CRS in "crs" argument.

For further methodological and performance information of these methods see Mendes et al. (2020).

If using one these constraining methods, cite Mendes et al (2020).

## Value

This function return a SpatRaster with continuous and binary prediction.

## References


- Mendes, P.; Velazco S.J.E.; Andrade, A.F.A.; De Marco, P. (2020) Dealing with overprediction in species distribution models: how adding distance constraints can improve model accuracy, Ecological Modelling, in press. https://doi.org/10.1016/j.ecolmodel.2020.109180
- Kremen, C., Cameron, A., Moilanen, A., Phillips, S. J., Thomas, C. D., Beentje, H., . Zjhra, M. L. (2008). Aligning Conservation Priorities Across Taxa in Madagascar with High-Resolution Planning Tools. Science, 320(5873), 222-226. doi:10.1126/science.1155193


## See Also

msdm_priori

## Examples

```
# Toy data: small raster and records of one species to keep this example fast
data("spp")
somevar <- terra::rast(system.file("external/somevar.tif", package = "flexsdm"))
somevar <- terra::aggregate(somevar, fact = 6) # coarser resolution
sp3 <- spp[spp$species == "sp3", ]
sp3 <- sdm_extract(data = sp3, x = "x", y = "y", env_layer = somevar)

# Toy continuous suitability layer (rescaled to 0-1)
cont_suit <- somevar[[1]] / terra::global(somevar[[1]], "max", na.rm = TRUE)[[1]]
names(cont_suit) <- "suit"

# Constrain the suitability with the minimum convex polygon of the presences
m_mcp <- msdm_posteriori(
  records = sp3,
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  method = "mcp",
  cont_suit = cont_suit,
  thr = "equal_sens_spec",
  buffer = NULL
)
m_mcp


require(dplyr)
require(terra)

data("spp")
somevar <- system.file("external/somevar.tif", package = "flexsdm")
somevar <- terra::rast(somevar)


# Preparing data for modeling a species
set.seed(10)
occ <- spp %>%
  dplyr::filter(species == "sp2") %>% # filter a species
```

```
  sdm_extract(
    data = ., x = "x", y = "y",
    env_layer = somevar, filter_na = TRUE
  ) %>% # extrac variables values
  part_random(.,
    pr_ab = "pr_ab",
    method = c(method = "kfold", folds = 10)
  ) # add columns with partition

# Fit a model
m_glm <- fit_glm(
  data = occ,
  response = "pr_ab",
  predictors = names(somevar),
  partition = ".part",
  thr = "equal_sens_spec",
)


# Lets predict this model
m_pred <- sdm_predict(models = m_glm, pred = somevar, thr = NULL, con_thr = FALSE)
plot(m_pred[[1]])
m_pred[[1]] %>% plot()

# Lets extract the raster from this list
m_pred <- m_pred[[1]]

### bmcp method
m_bmcp <- msdm_posteriori(
  records = occ,
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  method = "bmcp",
  cont_suit = m_pred,
  thr = "equal_sens_spec",
  buffer = 30000,
  crs = crs(m_pred)
)

plot(m_bmcp)


### mcp method
m_mcp <- msdm_posteriori(
  records = occ,
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  method = "mcp",
  cont_suit = m_pred,
  thr = "equal_sens_spec",
  buffer = NULL
```

```
)

plot(m_mcp)


### pres method
m_pres <- msdm_posteriori(
  records = occ,
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  method = "pres",
  cont_suit = m_pred,
  thr = "equal_sens_spec",
  buffer = NULL
)

plot(m_pres)


### lq method
m_lq <- msdm_posteriori(
  records = occ,
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  method = "lq",
  cont_suit = m_pred,
  thr = "equal_sens_spec",
  buffer = NULL
)

plot(m_lq)


### obr method
m_obr <- msdm_posteriori(
  records = occ,
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  method = "obr",
  cont_suit = m_pred,
  thr = "equal_sens_spec",
  buffer = NULL
)

plot(m_obr)
```

msdm_priori *Create spatial predictor variables to reduce overprediction of species distribution models*

## Description

This function creates geographical predictor variables that, together with environmental variables, can be used to construct constrained species distribution models.

## Usage

```
msdm_priori(data, x, y, method = c("xy", "min", "cml", "ker"), env_layer)
```

## Arguments

| | |
|---|---|
| data | tibble or data.frame. A database with geographical coordinates of species presences. |
| x | character. Column name with spatial x coordinates. |
| y | character. Column name with spatial y coordinates. |
| method | character. A character string indicating which MSDM method will be used. The following methods are available: 'xy', 'min', 'cml', and 'ker'. Usage method = 'cml' |
| env_layer | A raster layer used to construct species distribution models. This object will be used to create constraining variables with the same resolution, extent, and pattern of empty cells as the environmental variables. It is advisable to use a raster of an environmental layer that will be used to create the species distribution models to avoid mismatch (e.g. resolution, extent, cells with NA) between environmental and constraining variables. |

## Details

This function creates geographical predictor variables that, together with environmental variables, can be used to construct constrained species distribution models. It is recommended to use these approaches to create models that will only be projected for current conditions and not for different time periods (past or future).

Four methods are implemented:

xy (Latlong method). This method assumes that spatial structure can partially explain species distribution (Bahn & McGill, 2007). Therefore, two raster layers will be created, containing the latitude and longitude of pixels, respectively. These raster layers should be included as covariates with the environmental layers to construct species distribution models. This method does not interact with species occurrence and is generic for a given study region; for this reason, it is possible to use this method for all species set that share the same study region.

min (Nearest neighbor distance method). Compiled and adapted from Allouche et al. (2008), this method calculates for each cell the Euclidean geographic distance to the nearest presence point.

cml (Cumulative distance method). Compiled and adapted from Allouche et al. (2008), this method assumes that pixels closer to presences are likely included in species distributions. Therefore, a raster layer will be created containing the sum of Euclidean geographic distances from each pixel to all occurrences of a species. Obtained values are normalized to vary from zero to one. This raster layer should be included with the environmental layers to construct species distribution models.

ker (Kernel method). Compiled and adapted from Allouche et al. (2008), this method, like cml, assumes that pixels located in areas with a higher density of occurrences are likely included in the actual species distribution. Thus, a raster layer will be created containing the Gaussian values based on the density of occurrences of a species. The standard deviation of the Gaussian distribution was the maximum value in a vector of minimum distances between pairs of occurrences of a species. Gaussian values are normalized to vary from zero to one. This raster layer should be included with the environmental layers to construct species distribution models.

See Mendes et al. (2020) for further methodological and performance details.

If used one these constraining method cite Mendes et al 2020.

## Value

This function returns a SpatRaster object. Such raster/s have to be used together with environmental variables to construct species distribution models. The 'xy' approach creates a single pair of raster layers that can be used for all species that share the same study region. Otherwise, 'cml', 'min', and 'ker' create a species-specific raster layer.

## References


- Mendes, P.; Velazco S.J.E.; Andrade, A.F.A.; De Marco, P. (2020) Dealing with overprediction in species distribution models: how adding distance constraints can improve model accuracy, Ecological Modelling, in press. https://doi.org/10.1016/j.ecolmodel.2020.109180
- Allouche, O.; Steinitz, O.; Rotem, D.; Rosenfeld, A.; Kadmon, R. (2008). Incorporating distance constraints into species distribution models. Journal of Applied Ecology, 45(2), 599-609. doi:10.1111/j.1365-2664.2007.01445.x
- Bahn, V.; McGill, B. J. (2007). Can niche-based distribution models outperform spatial interpolation? Global Ecology and Biogeography, 16(6), 733-742. doi:10.1111/j.1466-8238.2007.00331.x


## See Also

msdm_posteriori

## Examples

```
# Toy data: small raster and records of one species to keep this example fast
data("spp")
somevar <- terra::rast(system.file("external/somevar.tif", package = "flexsdm"))
somevar <- terra::aggregate(somevar, fact = 6) # coarser resolution
sp3 <- spp[spp$species == "sp3", ]
occ <- sp3[sp3$pr_ab == 1, ]

# Spatial constraint based on the coordinates of the records
m_xy <- msdm_priori(
  data = occ,
```

```
  x = "x",
  y = "y",
  method = "xy",
  env_layer = somevar[[1]]
)
m_xy


require(dplyr)
require(terra)

data("spp")
somevar <- system.file("external/somevar.tif", package = "flexsdm")
somevar <- terra::rast(somevar)

# Select the presences of a species
occ <- spp %>%
  dplyr::filter(species == "sp3", pr_ab == 1)

# Select a raster layer to be used as a basic raster
a_variable <- somevar[[1]]
plot(a_variable)
points(occ %>% dplyr::select(x, y))

### xy method
m_xy <- msdm_priori(
  data = occ,
  x = "x",
  y = "y",
  method = "xy",
  env_layer = a_variable
)

plot(m_xy)

### min method
m_min <- msdm_priori(
  data = occ,
  x = "x",
  y = "y",
  method = "min",
  env_layer = a_variable
)

plot(m_min)
points(occ %>% dplyr::select(x, y), col = "red", pch = 16)

### cml method
m_cml <- msdm_priori(
  data = occ,
  x = "x",
  y = "y",
  method = "cml",
```

```
  env_layer = a_variable
)

plot(m_cml)
points(occ %>% dplyr::select(x, y), col = "red", pch = 16)

### ker method
m_ker <- msdm_priori(
  data = occ,
  x = "x",
  y = "y",
  method = "ker",
  env_layer = a_variable
)

plot(m_ker)
points(occ %>% dplyr::select(x, y), col = "red", pch = 16)
```

---

occ_reproj *Reproject location coordinates to a new coordinate reference system*

---

## Description

Reprojects the x/y (longitude/latitude) coordinates of a set of species occurrence records from one coordinate reference system (CRS) to another, and returns the results either as new columns appended to the input data or as replacements for the original coordinate columns.

## Usage

```
occ_reproj(
  data,
  x = "x",
  y = "y",
  from = "EPSG:4326",
  to = NULL,
  replace_cols = FALSE
)
```

## Arguments

| | |
|---|---|
| data | data.frame or tibble. A table containing occurrence records with at least two columns for the x (longitude) and y (latitude) coordinates. |
| x | character. Name of the column in data containing the x (longitude) coordinate. Default "x". |
| y | character. Name of the column in data containing the y (latitude) coordinate. Default "y". |

| | |
|---|---|
| from | character. CRS of the input coordinates, provided as an EPSG code (e.g., "EPSG:4326") or any other CRS string accepted by terra::vect(). Default "EPSG:4326" (WGS84 unprojected). |
| to | character. CRS to project the coordinates to, provided as an EPSG code (e.g., "EPSG:32617") or any other CRS string accepted by terra::project(). This argument is mandatory; the function will stop with an error if to = NULL. |
| replace_cols | logical. If TRUE, the original x and y columns in data are overwritten with the reprojected coordinates. If FALSE (default), the reprojected coordinates are appended as new columns x_new and y_new, leaving the original columns unchanged. |

## Details

This function is a convenience wrapper around terra::vect() and terra::project() for reprojecting tabular occurrence coordinates without needing to first convert data into a spatial object. It is useful, for example, when occurrence records are stored in geographic coordinates (e.g., "EPSG:4326") but need to be converted to a projected CRS (e.g., a UTM zone) for distance-based calculations or to match the CRS of environmental raster layers used elsewhere in a species distribution modeling workflow.

## Value

A tibble (or data.frame, if replace_cols = TRUE) with the reprojected coordinates. If replace_cols = FALSE, two new columns, x_new and y_new, are added to data. If replace_cols = TRUE, the original x and y columns are replaced in place and the object class of data is preserved.

## See Also

project, vect

## Examples

```
data <- data.frame(
  species = c("sp1", "sp2", "sp3"),
  x = c(-74.1, -73.9, -74.0),
  y = c(4.6, 4.7, 4.65)
)

# Append reprojected coordinates as new columns
occ_reproj(data, x = "x", y = "y", from = "EPSG:4326", to = "EPSG:32618")

# Replace original coordinate columns with reprojected values
occ_reproj(
  data,
  x = "x", y = "y",
  from = "EPSG:4326", to = "EPSG:32618",
  replace_cols = TRUE
)
```

occfilt_env *Perform environmental filtering on species occurrences*

## Description

This function perform filtering on species occurrences based on their environmental conditions.

## Usage

```
occfilt_env(data, x, y, id, env_layer, nbins)
```

## Arguments

| | |
|---|---|
| data | data.frame. Data.frame or tibble object with presences (or presence-absence) records, and coordinates |
| x | character. Column name with spatial x coordinates |
| y | character. Column name with spatial y coordinates |
| id | character. Column names with rows id. It is important that each row has its own unique code. |
| env_layer | SpatRaster. Raster variables that will be used to fit the model. Factor variables will be removed. |
| nbins | integer. A number of classes used to split each environmental condition. It is possible to use single or several values. If several values are provided, the function will return a list with the results. Usage nbins = 5 or nbins = c(5, 10, 15) |

## Details

This function uses an approach adapted from the approach proposed by Varela et al. (2014). It consists of filtering occurrences in environmental space. First, a regular multidimensional grid is created in environmental space. This multidimensional grid is determined by the environmental variables (always use continuous variables) the grid cell size is defined by the number of bins, used for dividing variable range into interval classes (Varela et al. 2014; Castellanos et al., 2019). The number of bins is set in the "nbins" argument. Then, a single occurrence is randomly selected within each cell of the multidimensional grid. Consider that there is a trade-off between the number of bins and the number of filtered records because as the number of bins decreases, the cell size of the grids increases, and the number of filtered records decreases (Castellanos et al., 2019). occfilt_env works for any number of dimensions (variables) and with the original variables without performing a PCA beforehand.

The greater the number of predictor variables (i.e., the number of dimensions of the multidimensional environmental grid) and the greater the number of bins, the greater the time processing and the computer memory used. Therefore, it is recommended to use a small number of bins between 2-5 if more than ten variables are used.

Environmental filters are sensitive to the number of bins. A procedure for selecting the number of bins was used by Velazco et al. (2020) and it is implemented in `occfilt_select`.

## Value

If one value is used to filter occurrence funtion will return a tibble object with filtered data. If several values are used to filter occurrences, the function will return a list of tibbles with filtered data.

## References


- Castellanos, A. A., Huntley, J. W., Voelker, G., & Lawing, A. M. (2019). Environmental filtering improves ecological niche models across multiple scales. Methods in Ecology and Evolution, 10(4), 481-492. https://doi.org/10.1111/2041-210X.13142
- Varela, S., Anderson, R. P., Garcia-Valdes, R., & Fernandez-Gonzalez, F. (2014). Environmental filters reduce the effects of sampling bias and improve predictions of ecological niche models. Ecography, 37, 1084-1091. https://doi.org/10.1111/j.1600-0587.2013.00441.x
- Velazco, S. J. E., Svenning, J-C., Ribeiro, B. R., & Laureto, L. M. O. (2020). On opportunities and threats to conserve the phylogenetic diversity of Neotropical palms. Diversity and Distributions, 27, 512–523. https://doi.org/10.1111/ddi.13215


## See Also

occfilt_geo, occfilt_select

## Examples

```
# Toy data: small raster and records of one species to keep this example fast
data("spp")
somevar <- terra::rast(system.file("external/somevar.tif", package = "flexsdm"))
somevar <- terra::aggregate(somevar, fact = 6) # coarser resolution
sp3 <- spp[spp$species == "sp3", ]
sp3 <- sp3[sp3$pr_ab == 1, ]
sp3$idd <- seq_len(nrow(sp3)) # unique id of each record

# Filter records using 5 bins in the environmental space
filtered <- occfilt_env(
  data = sp3,
  x = "x",
  y = "y",
  id = "idd",
  env_layer = somevar,
  nbins = 5
)
nrow(sp3)
nrow(filtered)


require(terra)
require(dplyr)
require(ggplot2)

# Environmental variables
somevar <- system.file("external/somevar.tif", package = "flexsdm")
```

```
somevar <- terra::rast(somevar)

plot(somevar)

# Species occurrences
data("spp")
spp
spp1 <- spp %>% dplyr::filter(species == "sp1", pr_ab == 1)

somevar[[1]] %>% plot()
points(spp1 %>% select(x, y))

spp1$idd <- 1:nrow(spp1)


# split environmental variables into 5 bins
filtered_1 <- occfilt_env(
  data = spp1,
  x = "x",
  y = "y",
  id = "idd",
  env_layer = somevar,
  nbins = 5
)

# split into 8 bins
filtered_2 <- occfilt_env(
  data = spp1,
  x = "x",
  y = "y",
  id = "idd",
  env_layer = somevar,
  nbins = 8
)

# split into 12 bins
filtered_3 <- occfilt_env(
  data = spp1,
  x = "x",
  y = "y",
  id = "idd",
  env_layer = somevar,
  nbins = 12
)


## %#######################################################%##
####         ' # Test different number of bins          ####
## %#######################################################%##

filtered_dif_bins <- occfilt_env(
  data = spp1,
  x = "x",
```

```
  y = "y",
  id = "idd",
  env_layer = somevar,
  nbins = c(4, 6, 8, 10, 12, 14)
)

class(filtered_dif_bins)
names(filtered_dif_bins) # each elements of this list has the names of the bins

filtered_dif_bins %>%
  dplyr::bind_rows(.id = "bins") %>%
  dplyr::mutate(bins = as.numeric(bins)) %>%
  ggplot(aes(x = x, y = y)) +
  geom_point() +
  facet_wrap(~bins)
# note that the higher the nbins parameter the more
# classes must be processed (4 variables, 30 bins = 923521 classes)

# While the greater the greater the number of bins, the greater records retained


# It is possible select the best of filtered
# datasets using the occfilt_selec function

occ_selected <- occfilt_select(
  occ_list = filtered_dif_bins,
  x = "x",
  y = "y",
  env_layer = somevar,
  filter_prop = TRUE
)

occ_selected
```

---

occfilt_geo *Perform geographical filtering on species occurrences*

---

## Description

This function perform geographical filtering of species occurrences based on different approach to define the minimum nearest-neighbor distance between points.

## Usage

```
occfilt_geo(
  data,
  x,
  y,
```

```
  env_layer,
  method,
  prj = "+proj=longlat +datum=WGS84",
  reps = 20
)
```

## Arguments

| | |
|---|---|
| data | data.frame. Data.frame or tibble object with presences (or presence-absence) records, and coordinates |
| x | character. Column name with longitude data |
| y | character. Column name with latitude data |
| env_layer | SpatRaster. Raster variables that will be used to fit the model. Factor variables will be removed. |
| method | character. Method to perform geographical thinning. Pairs of points are filtered based on a geographical distance criteria. For the three method, it is possible to use several values. If several values are provided, the function will return a list with the results. The following methods are available:<br>• moran: records are filtered based on the smallest distance to the Moran's I value provided. If no Moran's I values is provided it will use 0.1. Usage method: method = c('moran') or method = c('moran', val = c(0.1, 0.15, 0.2, 0.25, 0.3, 0.35)).<br>• cellsize: records are filtered based on the resolution of the environmental variables which can be aggregated to coarser resolution defined by the factor. Usage method: method = c('cellsize', factor = '2') or method = c('cellsize', factor = c(1, 4, 8)).<br>• defined: records are filtered based on a distance value (d) provided in km. Usage method: method = c('defined', d = c(20, 40, 60, 80)). |
| prj | character. Projection string (PROJ4) for occurrences. Not necessary if the projection used is WGS84 ("+proj=longlat +datum=WGS84"). Default "+proj=longlat +datum=WGS84" |
| reps | integer. Number of times to repeat the thinning process. Default 20 |

## Details

In this function three alternatives are implemented to determine the distance threshold between pair of points: '

- "moran" determines the minimum nearest-neighbor distance that approximate to the spatial autocorrelation in occurrence data, following a Moran's I. To do so, a Principal Component Analysis with the environmental variables is performed and then the first Principal Component is used to calculate the semivariograms. Because of this, this method only allow the use of continuous variables. Sometimes, this method can (too) greatly reduce the number of presences.
- "cellsize" filters occurrences based on the predictors' resolution. This method will calculate the distance between the first two cells of the environmental variable and use this distance as minimum nearest-neighbor distance to filter occurrences. The resolution of the raster is

aggregated based on the values used in "factor". Thus, the distance used for filtering can be adjusted to represent a larger grid size.

- "determined" this method uses any minimum nearest-neighbor distance specified in km.

The "thin" function from spThin package is used to filter data (Aiello-Lammens et al., 2015) with the following argument settings reps = 20, write.files = FALSE, locs.thinned.list.return = TRUE, and write.log.file = FALSE.

In the case that more than one value was used in some of the methods, it is possible to use use occfilt_select function to automatically select a filtered database based on number of records and spatial autocorrelation.

## Value

If one value is used to filter occurrence function will return a tibble object with filtered data. If several values are used to filter occurrences, the function will return a list of tibbles with filtered data.

## References


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


## See Also

occfilt_env, occfilt_select

## Examples

```
# Toy data: small raster and records of one species to keep this example fast
data("spp")
somevar <- terra::rast(system.file("external/somevar.tif", package = "flexsdm"))
somevar <- terra::aggregate(somevar, fact = 6) # coarser resolution
sp3 <- spp[spp$species == "sp3", ]
sp3 <- sp3[sp3$pr_ab == 1, ]

# Geographic filtering based on a coarser resolution (cell size method)
filtered_occ <- occfilt_geo(
  data = sp3,
  x = "x",
  y = "y",
  env_layer = somevar,
  method = c("cellsize", factor = "3"),
  prj = terra::crs(somevar)
)
nrow(sp3)
nrow(filtered_occ)


require(terra)
require(dplyr)
```

```
require(ggplot2)

# Environmental variables
somevar <- system.file("external/somevar.tif", package = "flexsdm")
somevar <- terra::rast(somevar)

plot(somevar)

# Species occurrences
data("spp")
spp
spp1 <- spp %>% dplyr::filter(species == "sp1", pr_ab == 1)

somevar[[1]] %>% plot()
points(spp1 %>% select(x, y))

## %#######################################################%##
####                  Cellsize method                  ####
## %#######################################################%##

# Using cellsize method
filtered_occ <- occfilt_geo(
  data = spp1,
  x = "x",
  y = "y",
  env_layer = somevar,
  method = c("cellsize", factor = "3"),
  prj = crs(somevar)
)

somevar[[1]] %>% plot(col = gray.colors(10))
points(spp1 %>% select(x, y)) # raw data
points(filtered_occ %>% select(x, y), pch = 19, col = "yellow") # filtered data


# Using cellsize method with several values
filtered_occ <- occfilt_geo(
  data = spp1,
  x = "x",
  y = "y",
  env_layer = somevar,
  method = c("cellsize", factor = c(1, 8, 16)),
  prj = crs(somevar)
)

filtered_occ # Note that several values are provided for any filtering method
# fuction will return a list of tibbles with the results.
# So user must select the desired filtered dataset

# Let's explore the results
bind_rows(filtered_occ, .id = "cellSize") %>%
  dplyr::mutate(cellSize = as.numeric(cellSize)) %>%
  ggplot(aes(x, y)) +
```

```
  geom_point() +
  facet_wrap(~cellSize)


## %#######################################################%##
####                  Defined method                  ####
## %#######################################################%##
# Using defined method
filtered_occ <- occfilt_geo(
  data = spp1,
  x = "x",
  y = "y",
  env_layer = somevar,
  method = c("defined", d = "30"),
  prj = crs(somevar)
)

somevar[[1]] %>% plot(col = gray.colors(10))
points(spp1 %>% select(x, y)) # raw data
points(filtered_occ %>% select(x, y), pch = 19, col = "yellow") # filtered data

# Using defined method with several values
filtered_occ <- occfilt_geo(
  data = spp1,
  x = "x",
  y = "y",
  env_layer = somevar,
  method = c("defined", factor = c(5, 15, 35)),
  prj = crs(somevar)
)

bind_rows(filtered_occ, .id = "cellSize") %>%
  dplyr::mutate(cellSize = as.numeric(cellSize)) %>%
  ggplot(aes(x, y)) +
  geom_point() +
  facet_wrap(~cellSize)


## %#######################################################%##
####                 Moran's I method                 ####
## %#######################################################%##

# Using Moran's I method
filtered_occ <- occfilt_geo(
  data = spp1,
  x = "x",
  y = "y",
  env_layer = somevar,
  method = c("moran"),
  prj = crs(somevar)
)

somevar[[1]] %>% plot(col = gray.colors(10))
```

```
points(spp1 %>% select(x, y)) # raw data
points(filtered_occ %>% select(x, y), pch = 19, col = "yellow") # filtered data

# Using Moran's I method with several values
filtered_occ <- occfilt_geo(
  data = spp1,
  x = "x",
  y = "y",
  env_layer = somevar,
  method = c("moran", c(0.05, 0.2, 0.5)),
  prj = crs(somevar)
)

bind_rows(filtered_occ, .id = "moran") %>%
  dplyr::mutate(moran = as.numeric(moran)) %>%
  ggplot(aes(x, y)) +
  geom_point() +
  facet_wrap(~moran)

# It is possible select the best of filtered
# datasets using the occfilt_selec function

occ_selected <- occfilt_select(
  occ_list = filtered_occ,
  x = "x",
  y = "y",
  env_layer = somevar,
  filter_prop = TRUE
)

occ_selected
```

---

occfilt_select *Select filtered occurrences*

---

## Description

Select filtered occurrences based on number of records and spatial autocorrelation (see details)

## Usage

```
occfilt_select(occ_list, x, y, env_layer, filter_prop = FALSE)
```

## Arguments

| | |
|---|---|
| occ_list | list. A list with filtered specie occurrences testing several values (see occfilt_env and occfilt_geo) |
| x | character. Column name with longitude data |

| | |
|---|---|
| y | character. Column name with latitude data |
| env_layer | SpatRaster. Raster variables that will be used to fit the model. Factor variables will be removed. |
| filter_prop | logical. If TRUE, the function will return a list with the filtered occurrences and a tibble with the spatial autocorrelation and number of occurrence values |

## Details

The function implement the approach used in Velazco et al. (2020) which consists in calculating for each filtered dataset:

- 1- the number of occurrence.
- 2- the spatial autocorrelation based on Morans'I for each variable
- 3- the mean spatial autocorrelation among variables

Then function will select those dataset with average spatial autocorrelation lower than the mean of all dataset, and from this subset will select the one with the highest number occurrences.

If use occfilt_select cite Velazco et al. (2020) as reference.

## Value

If filter_prop = FALSE, a tibble with selected filtered occurrences. If filter_prop = TRUE, a list with following objects:

- A tibble with selected filtered occurrences
- A tibble with filter properties with columns:
  - filt_value: values used for filtering, the value with an asterisk will denote the one selected
  - n_records: number of occurrence
  - mean_autocorr: mean spatial autocorrelation.
  - the remaining columns have the spatial autocorrelation values for each variable.

## References


- Velazco, S. J. E., Svenning, J-C., Ribeiro, B. R., & Laureto, L. M. O. (2020). On opportunities and threats to conserve the phylogenetic diversity of Neotropical palms. Diversity and Distributions, 27, 512–523. https://doi.org/10.1111/ddi.13215


## See Also

occfilt_env, occfilt_geo

## Examples

```
data("spp")
somevar <- terra::rast(system.file("external/somevar.tif", package = "flexsdm"))
sp1 <- spp[spp$species == "sp1" & spp$pr_ab == 1, ]

# Filter records with several coarser resolutions (cell size method)
```

```
filtered_occ <- occfilt_geo(
  data = sp1,
  x = "x",
  y = "y",
  env_layer = somevar,
  method = c("cellsize", factor = c(4, 8, 12)),
  prj = terra::crs(somevar)
)

# Select the filtered data based on number of records and spatial autocorrelation
occ_selected <- occfilt_select(
  occ_list = filtered_occ,
  x = "x",
  y = "y",
  env_layer = somevar,
  filter_prop = FALSE
)
nrow(occ_selected)


require(terra)
require(dplyr)

# Environmental variables
somevar <- system.file("external/somevar.tif", package = "flexsdm")
somevar <- terra::rast(somevar)

plot(somevar)

# Species occurrences
data("spp")
spp
spp1 <- spp %>% dplyr::filter(species == "sp1", pr_ab == 1)

## %#######################################################%##
####                  Cellsize method                   ####
## %#######################################################%##
# Using cellsize method
filtered_occ <- occfilt_geo(
  data = spp1,
  x = "x",
  y = "y",
  env_layer = somevar,
  method = c("cellsize", factor = c(1, 4, 8, 12, 16, 20)),
  prj = crs(somevar)
)

filtered_occ

# Select filtered occurrences based on
# number of records and spatial autocorrelation
occ_selected <- occfilt_select(
  occ_list = filtered_occ,
```

```
  x = "x",
  y = "y",
  env_layer = somevar,
  filter_prop = FALSE
)
occ_selected

occ_selected <- occfilt_select(
  occ_list = filtered_occ,
  x = "x",
  y = "y",
  env_layer = somevar,
  filter_prop = TRUE
)
occ_selected$occ

occ_selected$filter_prop
```

---

p_bpdp *Bivariate partial dependence plot*

---

## Description

Create bivariate partial dependence plot(s) to explore the bivariate marginal effect of predictors on suitability

## Usage

```
p_bpdp(
  model,
  predictors = NULL,
  resolution = 50,
  training_data = NULL,
  training_boundaries = NULL,
  projection_data = NULL,
  clamping = FALSE,
 color_gradient = c("#000004", "#1B0A40", "#4A0C69", "#781B6C", "#A42C5F", "#CD4345",
    "#EC6824", "#FA990B", "#F7CF3D", "#FCFFA4"),
  color_training_boundaries = "white",
  theme = ggplot2::theme_classic()
)
```

## Arguments

| | |
|---|---|
| model | A model object of class "gam", "gbm", "glm", "graf", "ksvm", "ksvm", "maxnet", "nnet", and "randomForest" This model can be found in the first element of the list returned by any function from the fit_, tune_, or esm_ function families |

`predictors` character. Vector of predictor names to calculate partial dependence plots. If NULL all predictors will be used. Default NULL

`resolution` numeric. Number of equally spaced points at which to predict suitability values for continuous predictors. Default 50

`training_data` data.frame. Database with response (0,1) and predictor values used to fit a model. Default NULL

`training_boundaries` character. Plot training conditions boundaries based on training data (i.e., presences, presences and absences, etc). If training_boundaries = "convexh", function will delimit training environmental region based on a convex-hull. If training_boundaries = "rectangle", function will delimit training environmental region based on four straight lines. If used any methods it is necessary provide data in training_data argument. If NULL all predictors will be used. Default NULL.

`projection_data` SpatRaster. Raster layer with environmental variables used for model projection. Default NULL

`clamping` logical. Perform clamping. Only for maxent models. Default FALSE

`color_gradient` character. A vector with range of colors to plot. Default c("#FDE725", "#B3DC2B", "#6DCC57", "#36B677", "#1F9D87", "#25818E", "#30678D", "#3D4988", "#462777", "#440154")

`color_training_boundaries` character. A vector with one color used to color points of residuals, Default "white"

`theme` ggplot2 theme. Default ggplot2::theme_classic()

## Details

This function creates partial dependent surface plots to explore the bivariate marginal effect of predictors on suitability. If projection_data is used, function will extract the minimum and maximum values found in a region or time period to which a model will be projected. Partial dependence surface plot could be used to interpret a model or to explore how a model my extrapolate outside the environmental conditions used to train the model (convex hull polygon).

## Value

A ggplot object, or a patchwork of ggplot objects when more than two predictors are used, showing bivariate partial dependence surface plot(s).

## See Also

`data_pdp`, `data_bpdp`, `p_pdp`, `extra_eval`, `extra_truncate`

## Examples

```
# Toy data: a small subset of 'abies' to keep this example fast
data("abies")
abies_toy <- abies[c(1:30, 701:730), ]
```

```
abies_toy <- part_random(
  data = abies_toy,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

glm_toy <- fit_glm(
  data = abies_toy,
  response = "pr_ab",
  predictors = c("aet", "cwd"),
  partition = ".part",
  thr = "max_sens_spec",
  poly = 2
)

p_bpdp(
  model = glm_toy$model,
  training_data = abies_toy,
  resolution = 20
)


library(terra)
library(dplyr)

somevar <- system.file("external/somevar.tif", package = "flexsdm")
somevar <- terra::rast(somevar) # environmental data
names(somevar) <- c("aet", "cwd", "tmx", "tmn")
data(abies)

# set seed
abies2 <- abies %>%
  dplyr::select(x, y, pr_ab) %>%
  dplyr::group_by(pr_ab) %>%
  dplyr::slice_sample(prop = 0.5)

abies2 <- sdm_extract(abies2,
  x = "x",
  y = "y",
  env_layer = somevar
)
abies2 <- part_random(abies2,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 5)
)

svm_t1 <- fit_svm(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmx", "tmn"),
  partition = ".part",
  thr = c("max_sens_spec")
)
```

```
# Partial depence surface plot
p_bpdp(model = svm_t1$model, training_data = abies2)
p_bpdp(model = svm_t1$model, training_data = abies2, predictors = c("aet", "cwd"))
p_bpdp(model = svm_t1$model, training_data = abies2, resolution = 10)
p_bpdp(model = svm_t1$model, training_data = abies2, resolution = 70)
# With training condition boundaires
p_bpdp(
  model = svm_t1$model, training_data = abies2,
  training_boundaries = "convexh"
)
p_bpdp(
  model = svm_t1$model, training_data = abies2,
  training_boundaries = "rectangle", color_training_boundaries = "yellow"
)
p_bpdp(
  model = svm_t1$model, training_data = abies2, training_boundaries = "convexh",
  color_training_boundaries = "orange",
  color_gradient = c("#00007F", "#007FFF", "#7FFF7F", "#FF7F00", "#7F0000")
)
# With projection data
p_bpdp(
  model = svm_t1$model, training_data = abies2, training_boundaries = "rectangle",
  projection_data = somevar, # a SpatRaster used to predict or project the model
  color_training_boundaries = "white",
  color_gradient = c("#00007F", "#007FFF", "#7FFF7F", "#FF7F00", "#7F0000")
)

# Bivariate partial dependence plot for training and projection condition
plot(somevar[[1]], main = "Projection area")
p_bpdp(
  model = svm_t1$model, training_data = abies2,
  projection_data = somevar, # a SpatRaster used to predict or project the model
  training_boundaries = "convexh"
)


# Bivariate partial dependece plot with categorical variables
somevar <- system.file("external/somevar.tif", package = "flexsdm")
somevar <- terra::rast(somevar) # environmental data
names(somevar) <- c("aet", "cwd", "tmx", "tmn")
cat <- system.file("external/clusters.shp", package = "flexsdm")
cat <- terra::vect(cat)
cat$clusters <- paste0("c", cat$clusters)
cat <- terra::rasterize(cat, somevar, field = "clusters")
somevar <- c(somevar, cat)
plot(somevar)

# set seed
abies2 <- abies %>%
  dplyr::select(x, y, pr_ab) %>%
  dplyr::group_by(pr_ab) %>%
  dplyr::slice_sample(prop = 0.5)
```

```
abies2 <- sdm_extract(
  data = abies2,
  x = "x",
  y = "y",
  env_layer = somevar
)
abies2 <- part_random(abies2,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 5)
)

svm_t1 <- fit_svm(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmx", "tmn"),
  predictors_f = "clusters",
  partition = ".part",
  thr = c("max_sens_spec")
)

p_bpdp(model = svm_t1$model, training_data = abies2, training_boundaries = "convexh")
```

---

p_extra *Graphical exploration of extrapolation or suitability pattern in the environmental and geographical space*

---

## Description

Graphical exploration of extrapolation or suitability pattern in the environmental and geographical space

## Usage

```
p_extra(
  training_data,
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  extra_suit_data,
  projection_data,
  predictors = NULL,
  geo_space = TRUE,
  geo_position = "right",
  prop_points = 0.2,
  maxcells = 1e+05,
  alpha_p = 0.5,
  color_p = "black",
```

```
  alpha_gradient = 0.5,
 color_gradient = c("#FDE725", "#B3DC2B", "#6DCC57", "#36B677", "#1F9D87", "#25818E",
    "#30678D", "#3D4988", "#462777", "#440154"),
  theme = ggplot2::theme_classic()
)
```

## Arguments

| | |
|---|---|
| training_data | data.frame. Database with response (0,1) and predictor values used to fit a model. |
| x | character. Column name with spatial x coordinates |
| y | character. Column name with spatial y coordinates |
| pr_ab | character. Column name with species absence-presence, pseudo-absence-presence, or background-presence data (0,1). |
| extra_suit_data | SpatRaster. Raster layer with extrapolation or suitability values. extra_suit_data must have the same resolution and extent than projection_data |
| projection_data | SpatRaster. Raster layer with environmental variables used for model projection. projection_data must have the same resolution and extent than extra_suit_data |
| predictors | character. Vector of predictor name(s) to calculate partial dependence plots. If NULL all predictors will be used. Default NULL. |
| geo_space | logical. If TRUE will be produced a map. Default TRUE |
| geo_position | character. Map position regarding plot of environmental space, right, left, bottom, or upper. Default "right" |
| prop_points | numeric. Proportion of cells from extra_suit_data and projection_data to select for plotting. default. 0.5. |
| maxcells | integer. Maximum number of cells used to plot in the geographical space. Default 100000 |
| alpha_p | numeric. a value between 0 to 1 to control transparency of presence-absence points. Lower values corresponding to more transparent colors. Default 0.5 |
| color_p | character. A vector with a color used to color presence-absence points. Default "black" |
| alpha_gradient | numeric. a value between 0 to 1 to control transparency of projection data Lower values corresponding to more transparent colors. Default 0.5 |
| color_gradient | character. A vector with colors used to color projection data. Default c( "#FDE725", "#B3DC2B", "#6DCC57", "#36B677", "#1F9D87", "#25818E", "#30678D", "#3D4988", "#462777", "#440154") |
| theme | ggplot2 theme. Default ggplot2::theme_classic() |

## Value

a plot

## Examples

```
# Toy data: small raster and records of one species to keep this example fast
data("spp")
somevar <- terra::rast(system.file("external/somevar.tif", package = "flexsdm"))
somevar <- terra::aggregate(somevar, fact = 6) # coarser resolution
sp3 <- spp[spp$species == "sp3", ]
sp3 <- sdm_extract(data = sp3, x = "x", y = "y", env_layer = somevar)
extr <- extra_eval(
  training_data = sp3,
  pr_ab = "pr_ab",
  projection_data = somevar,
  metric = "mahalanobis"
)

p_extra(
  training_data = sp3,
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  extra_suit_data = extr,
  projection_data = somevar,
  geo_space = TRUE,
  prop_points = 0.5
)



require(dplyr)
require(terra)
require(ggplot2)

data(spp)
f <- system.file("external/somevar.tif", package = "flexsdm")
somevar <- terra::rast(f)
names(somevar) <- c("aet", "cwd", "tmx", "tmn")

spp$species %>% unique()
sp <- spp %>%
  dplyr::filter(species == "sp2", pr_ab == 1) %>%
  dplyr::select(x, y, pr_ab)

# Calibration area based on some criterion such as dispersal ability
ca <- calib_area(sp,
  x = "x", y = "y",
  method = c("buffer", width = 50000), crs = crs(somevar)
)

plot(somevar[[1]])
points(sp)
plot(ca, add = TRUE)
```

```
# Sampling pseudo-absences
set.seed(10)
psa <- sample_pseudoabs(
  data = sp,
  x = "x",
  y = "y",
  n = nrow(sp) * 2,
  method = "random",
  rlayer = somevar,
  calibarea = ca
)

# Merge presences and abasences databases to get a complete calibration data
sp_pa <- dplyr::bind_rows(sp, psa)
sp_pa

# Get environmental condition of calibration area
sp_pa_2 <- sdm_extract(data = sp_pa, x = "x", y = "y", env_layer = somevar)
sp_pa_2

# Measure extrapolation based on calibration data (presence and pseudo-absences)
# using SHAPE metric
extr <-
  extra_eval(
    training_data = sp_pa_2,
    pr_ab = "pr_ab",
    projection_data = somevar,
    metric = "mahalanobis",
    univar_comb = FALSE,
    aggreg_factor = 1
  )
plot(extr)

## %#######################################################%##
####            Explore extrapolation in the            ####
####        environmental and geographical space        ####
## %#######################################################%##

p_extra(
  training_data = sp_pa_2,
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  extra_suit_data = extr,
  projection_data = somevar,
  geo_space = TRUE,
  prop_points = 0.05
)

p_extra(
  training_data = sp_pa_2,
  x = "x",
  y = "y",
```

```
  pr_ab = "pr_ab",
  extra_suit_data = extr,
  projection_data = somevar,
  predictors = c("tmn", "cwd"),
  geo_space = TRUE,
  prop_points = 0.05
)

p_extra(
  training_data = sp_pa_2,
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  extra_suit_data = extr,
  projection_data = somevar,
  predictors = c("cwd", "tmx", "aet"),
  geo_space = TRUE,
  geo_position = "left",
  prop_points = 0.05,
  color_p = "white",
  alpha_p = 0.5,
  alpha_gradient = 0.2,
  color_gradient = c("#404096", "#529DB7", "#7DB874", "#E39C37", "#D92120"),
  theme = ggplot2::theme_dark()
)

p_extra(
  training_data = sp_pa_2,
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  extra_suit_data = extr,
  projection_data = somevar,
  geo_space = TRUE,
  prop_points = 0.05,
  color_p = "white",
  alpha_p = 0.5,
  alpha_gradient = 0.2,
  color_gradient = c("#404096", "#529DB7", "#7DB874", "#E39C37", "#D92120"),
  theme = ggplot2::theme_dark()
)

# Explore extrapolation only in the environmental space
p_extra(
  training_data = sp_pa_2,
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  extra_suit_data = extr,
  projection_data = somevar,
  geo_space = FALSE,
  prop_points = 0.05,
  color_p = "black",
```

```
  color_gradient = c("#085CF8", "#65AF1E", "#F3CC1D", "#FC6A9B", "#D70500"),
  theme = ggplot2::theme_minimal()
)


## %#######################################################%##
####                 Explore univariate                 ####
####          and combinatorial extrapolation           ####
## %#######################################################%##
extr <-
  extra_eval(
    training_data = sp_pa_2,
    pr_ab = "pr_ab",
    projection_data = somevar,
    metric = "mahalanobis",
    univar_comb = TRUE,
    aggreg_factor = 1
  )

plot(extr)


p_extra(
  training_data = sp_pa_2,
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  extra_suit_data = extr$uni_comb, # use  uni_comb layer
  projection_data = somevar,
  geo_space = TRUE,
  prop_points = 0.05,
  color_gradient = c("#B3DC2B", "#25818E")
)

## %#######################################################%##
####           With p_extra also is possible            ####
####       to explore the patterns of suitability       ####
## %#######################################################%##

sp_pa_2 <- part_random(
  data = sp_pa_2,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 5)
)

rf_m1 <- fit_raf(
  data = sp_pa_2,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmx", "tmn"),
  partition = ".part",
  thr = c("max_sorensen")
)
```

```
suit <- sdm_predict(models = rf_m1, pred = somevar)
plot(suit$raf)
suit <- suit$raf

# Pasterns of suitability in geographical and environmental space
p_extra(
  training_data = sp_pa_2,
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  extra_suit_data = suit,
  projection_data = somevar,
  geo_space = TRUE,
  prop_points = 0.05,
)

# Pasterns of suitability plotting as points only presences
p_extra(
  training_data = sp_pa_2 %>%
    dplyr::filter(pr_ab == 1),
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  extra_suit_data = suit,
  projection_data = somevar,
  geo_space = TRUE,
  prop_points = 0.05,
)

# Pasterns of suitability in the environmental space only
# and plotting as points only presences
p_extra(
  training_data = sp_pa_2 %>%
    dplyr::filter(pr_ab == 1),
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  extra_suit_data = suit,
  projection_data = somevar,
  geo_space = FALSE,
  prop_points = 0.05,
)
```

---

p_pdp *Partial Dependent Plot*

---

## Description

Create partial dependence plot(s) to explore the marginal effect of predictors on suitability

## Usage

```
p_pdp(
  model,
  predictors = NULL,
  resolution = 100,
  resid = FALSE,
  training_data = NULL,
  projection_data = NULL,
  clamping = FALSE,
  rug = FALSE,
  colorl = c("#462777", "#6DCC57"),
  colorp = "black",
  alpha = 0.2,
  theme = ggplot2::theme_classic()
)
```

## Arguments

| | |
|---|---|
| `model` | A model object of class "gam", "gbm", "glm", "graf", "ksvm", "ksvm", "maxnet", "nnet", and "randomForest" This model can be found in the first element of the list returned by any function from the fit_, tune_, or esm_ function families |
| `predictors` | character. Vector of predictor name(s) to calculate partial dependence plots. If NULL all predictors will be used. Default NULL |
| `resolution` | numeric. Number of equally spaced points at which to predict suitability values for continuous predictors. Default 50 |
| `resid` | logical. Calculate residuals based on training data. Default FALSE |
| `training_data` | data.frame. Database with response (0,1) and predictor values used to fit a model. Default NULL |
| `projection_data` | SpatRaster. Raster layer with environmental variables used for model projection. When this argument is used, function will calculate partial dependence curves distinguishing conditions used in training and projection conditions (i.e., projection data present in projection area but not training). Default NULL |
| `clamping` | logical. Perform clamping. Only for maxent models. Default FALSE |
| `rug` | logical. Display training data as a rug plot on the x-axis. Note: this could be time-consuming for large databases. Default FALSE |
| `colorl` | character. A vector with one or two colors used to color lines. If projection_data argument is used it is necessary to provide two colors. Default c("#462777", "#6DCC57") |
| `colorp` | character. A vector with one color used to color points of residuals, Default "black" |
| `alpha` | numeric. a value between 0 to 1 to control transparency of residual points. Lower values corresponding to more transparent colors. Default 0.2 |
| `theme` | ggplot2 theme. Default ggplot2::theme_classic() |

## Details

This function creates partial dependent plots to explore the marginal effect of predictors on suitability. If projection_data is used, function will extract the minimum and maximum values found in a region or time period to which a model will be projected. If the range of projection data is greater than that of the training data it will be plotted with a different color. Partial dependence curves could be used to interpret a model or to explore how a model may extrapolate outside the environmental conditions used to train the model.

## Value

A ggplot object, or a patchwork of ggplot objects when more than one predictor is used, showing partial dependence plot(s).

## See Also

data_pdp, data_bpdp, p_bpdp, extra_eval, extra_truncate

## Examples

```
# Toy data: a small subset of 'abies' to keep this example fast
data("abies")
abies_toy <- abies[c(1:30, 701:730), ]
abies_toy <- part_random(
  data = abies_toy,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

glm_toy <- fit_glm(
  data = abies_toy,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin"),
  partition = ".part",
  thr = "max_sens_spec"
)

# Partial dependence plot (one predictor, low resolution to keep it fast)
p_pdp(
  model = glm_toy$model,
  training_data = abies_toy,
  predictors = "aet",
  resolution = 20
)


library(terra)
library(dplyr)

somevar <- system.file("external/somevar.tif", package = "flexsdm")
somevar <- terra::rast(somevar) # environmental data
names(somevar) <- c("aet", "cwd", "tmx", "tmn")
```

```
data(abies)

# set seed
abies2 <- abies %>%
  dplyr::select(x, y, pr_ab) %>%
  dplyr::group_by(pr_ab) %>%
  dplyr::slice_sample(prop = 0.5)

abies2 <- sdm_extract(abies2,
  x = "x",
  y = "y",
  env_layer = somevar
)
abies2 <- part_random(abies2,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 5)
)

svm_t1 <- fit_svm(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmx", "tmn"),
  partition = ".part",
  thr = c("max_sens_spec")
)

# Partial depence plot
p_pdp(model = svm_t1$model, training_data = abies2)
p_pdp(model = svm_t1$model, training_data = abies2, predictors = c("aet", "cwd"))
p_pdp(model = svm_t1$model, training_data = abies2, resolution = 5)
p_pdp(model = svm_t1$model, training_data = abies2, resolution = 50)
p_pdp(model = svm_t1$model, training_data = abies2, resid = TRUE)
p_pdp(
  model = svm_t1$model, training_data = abies2, resid = TRUE,
  colorl = "black", colorp = "red", alpha = 0.1
)
p_pdp(
  model = svm_t1$model, training_data = abies2, resid = TRUE,
  colorl = "black", colorp = "red", alpha = 0.1, rug = TRUE
)

# Partial depence plot for training and projection condition found in a projection area
plot(somevar[[1]], main = "Projection area")
p_pdp(model = svm_t1$model, training_data = abies2, projection_data = somevar)
p_pdp(
  model = svm_t1$model, training_data = abies2, projection_data = somevar,
  colorl = c("#CC00FF", "#CCFF00")
)
p_pdp(
  model = svm_t1$model, training_data = abies2, projection_data = somevar,
  colorl = c("#CC00FF", "#CCFF00"), resid = TRUE, colorp = "gray"
)
p_pdp(
```

```
  model = svm_t1$model, training_data = abies2, projection_data = somevar,
  colorl = c("#CC00FF", "#CCFF00"), resid = TRUE, colorp = "gray", rug = TRUE,
  theme = ggplot2::theme_dark()
)
```

---

palms *A data set containing presences of palms species from Southern Brazil*

---

## Description

This data set contains presences of 11 palm species from Southern Brazil sourced by Calambás-Trochez et al. (2021).

## Usage

```
palms
```

## Format

A tibble with 327 rows and 3 variables:

**species** species names

**x** longitude of species occurrences

**y** latitude of species occurrences ...

## References

Calambás-Trochez, L.F., Velazco, S.J.E., Hoffmann, P.M., Brum, F.T., Carlucci, M.B., 2021. Climate and land-use changes coupled with low coverage of protected areas threaten palm species in South Brazilian grasslands. Perspectives in Ecology and Conservation 9. https://doi.org/10.1016/j.pecon.2021.03.010

## Examples

```
require(dplyr)
data("palms")
```

part_random *Conventional data partitioning methods*

## Description

This function provides different conventional (randomized, non-spatial) partitioning methods based on cross validation folds (kfold, rep_kfold, and loocv), as well as bootstrap (boot)

## Usage

```
part_random(data, pr_ab, method = NULL)
```

## Arguments

`data` data.frame. Database with presences, presence-absence, or pseudo-absence, records for a given species

`pr_ab` character. Column name of "data" with presences, presence-absence, or pseudo-absence. Presences must be represented by 1 and absences by 0

`method` character. Vector with data partitioning method to be used. Usage part=c(method= 'kfold', folds='5'). Methods include:

- kfold: Random partitioning into k-folds for cross-validation. 'folds' refers to the number of folds for data partitioning, it assumes value >=1. Usage method = c(method = "kfold", folds = 10).
- rep_kfold: Random partitioning into repeated k-folds for cross-validation. Usage method = c(method = "rep_kfold", folds = 10, replicates=10). 'folds' refers to the number of folds for data partitioning, it assumes value >=1. 'replicate' refers to the number of replicates, it assumes a value >=1.
- loocv: Leave-one-out cross-validation (a.k.a. Jackknife). It is a special case of k-fold cross validation where the number of partitions is equal to the number of records. Usage method = c(method = "loocv").
- boot: Random bootstrap partitioning. Usage method=c(method='boot', replicates='2', proportion='0.7'). 'replicate' refers to the number of replicates, it assumes a value >=1. 'proportion' refers to the proportion of occurrences used for model fitting, it assumes a value >0 and <=1. In this example proportion='0.7' mean that 70% of data will be used for model training, while 30% will be used for model testing.

## Value

A tibble object with information used in the 'data' argument and additional columns named .part containing the partition groups. The rep_kfold and boot method will return as many ".part" columns as replicated defined. For the rest of the methods, a single .part column is returned. For kfold, rep_kfold, and loocv partition methods, groups are defined by integers. In contrast, for boot method, the partition groups are defined by the characters 'train' and 'test'.

## References


- Fielding, A. H., & Bell, J. F. (1997). A review of methods for the assessment of prediction errors in conservation presence/absence models. Environmental Conservation, 24(1), 38-49. https://doi.org/10.1017/S0376892997000088


## See Also

part_sblock, part_senv, sample_pseudoabs, sample_background

## Examples

```
data("abies")
abies_toy <- abies[c(1:30, 701:730), ] # small subset to keep this example fast

# k-fold partition
abies_part <- part_random(
  data = abies_toy,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)
table(abies_part$.part)


require(dplyr)
data("abies")
abies$partition <- NULL
abies <- tibble(abies)

# K-fold method
abies2 <- part_random(
  data = abies,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 10)
)
abies2

# Repeated K-fold method
abies2 <- part_random(
  data = abies,
  pr_ab = "pr_ab",
  method = c(method = "rep_kfold", folds = 10, replicates = 10)
)
abies2

# Leave-one-out cross-validation (loocv) method
abies2 <- part_random(
  data = abies,
  pr_ab = "pr_ab",
  method = c(method = "loocv")
)
abies2
```

```
# Bootstrap method
abies2 <- part_random(
  data = abies,
  pr_ab = "pr_ab",
  method = c(method = "boot", replicates = 50, proportion = 0.7)
)
abies2
abies2$.part1 %>% table() # Note that for this method .partX columns have train and test words.
```

---

part_sband *Spatial band cross-validation*

---

## Description

This function explores different numbers of spatial bands and returns the most suitable value for a given presence or presence-absence database. The selection of the best number of bands is performed automatically considering spatial autocorrelation, environmental similarity, and the number of presence and absence records in each partition.

## Usage

```
part_sband(
  env_layer,
  data,
  x,
  y,
  pr_ab,
  type = "lon",
  n_part = 2,
  min_bands = 2,
  max_bands = 20,
  min_occ = 10,
  prop = 0.5
)
```

## Arguments

| | |
|---|---|
| env_layer | SpatRaster. Raster with environmental variable. Used to evaluate spatial autocorrelation and environmental similarity between training and testing partitions. Because this function calculate dissimilarity based on Euclidean distances, it can only be used with continuous environmental variables |
| data | data.frame. Data.frame or tibble object with presences (or presence-absence, or presence-pseudo-absence) records, and coordinates |
| x | character. Column name with spatial x coordinates |
| y | character. Column name with spatial y coordinates |

| | |
|---|---|
| pr_ab | character. Column with presences, presence-absence, or -pseudo-absence. Presences must be represented by 1 and absences by 0 |
| type | character. Specify bands across different degrees of longitude 'lon' or latitude 'lat'. Default is 'lon'. |
| n_part | integer. Number of partition. Default 2, values other than 2 has not yet been implemented. |
| min_bands | integer. Minimum number of spatial bands to be tested, default 2. |
| max_bands | integer. Maximum number of spatial bands to be tested, default 20. |
| min_occ | numeric. Minimum number of presences or absences in a partition fold. The min_occ value should be base on the number of predictors in order to avoid over-fitting or error when fitting models for a given fold. Default 10. |
| prop | numeric. Proportion of points used for testing autocorrelation between groups (values > 0 and <=1). The smaller this number is, the faster the function will work. Default 0.5 |

## Details

The part_sbands function allows testing different numbers of partitions using a range of latitudinal or longitudinal bands. This function explores a range of numbers of bands for a given number of partitions and automatically selects the best number of bands for a given presence, presence-absences, or presence-pseudo-absences dataset. Selection of number of bands is based on an optimization procedure that explores partitions in three dimensions determined by spatial autocorrelation (measured by Moran's I), environmental similarity (Euclidean distance), and difference in the amount of data among partition groups (Standard Deviation - SD; Velazco et al., 2019). This procedure is iterative; it will first select those partitions with autocorrelation values less than the lowest quartile of Morans I, then those with environmental similarity values greater than the third quartile of the Euclidean distances, then those with a difference in the amount of data less than the lowest quartile of SD. This selection is repeated until only one partition is retained (Velazco et al., 2019). The main benefits of this partition selection are that it i) is not subjective, ii) balances the environmental similarity and special autocorrelation between partitions groups, and iii) controls the selection of partitions with very little data that may be problematic for model fitting ("min_occ" argument).

Partitions that are geographically structured tend to evaluate model transferability more directly than conventional ones (e.g., those performed by `part_random`) (Roberts et al., 2017; Santini et al., 2021), being relevant for models that are to be used for projections in other regions outside the training area or for other time periods. Band partitions can be an option for those species where no best partition is found with part_sblock or for species that are distributed linearly (e.g., species that inhabit coastlines).

This function can interact with `get_block`, `sample_background`, and `sample_pseudoabs` for sampling background points or pseudo-absences within spatial partition groups

## Value

A list with:

- part: A tibble object with information used in 'data' arguments and a additional column .part with partition group.

- best_part_info: A tibble with information about the best partition. It contains the number of the best partition (n_grid), number of bands (n_bands), standard deviation of presences (sd_p), standard deviation of absences (sd_a), Moran's I spatial autocorrelation (spa_auto), and environmental similarity based on Euclidean distance (env_sim).
- grid: A SpatRaster object with bands

## References


- Roberts, D. R., Bahn, V., Ciuti, S., Boyce, M. S., Elith, J., Guillera-Arroita, G., Hauenstein, S., Lahoz-Monfort, J. J., Schroder, B., Thuiller, W., Warton, D. I., Wintle, B. A., Hartig, F., & Dormann, C. F. (2017). Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. Ecography, 40, 913-929. https://doi.org/10.1111/ecog.02881
- Santini, L., Benitez-Lopez, A., Maiorano, L., Cengic, M., & Huijbregts, M. A. J. (2021). Assessing the reliability of species distribution projections in climate change research. Diversity and Distributions, ddi.13252. https://doi.org/10.1111/ddi.13252
- Velazco, S. J. E., Villalobos, F., Galvao, F., & De Marco Junior, P. (2019). A dark scenario for Cerrado plant species: Effects of future climate, land use and protected areas ineffectiveness. Diversity and Distributions, 25(4), 660-673. https://doi.org/10.1111/ddi.12886


## See Also

part_random, part_sblock, part_senv, and get_block

## Examples

```
# Toy data: small raster and records of one species to keep this example fast
data("spp")
somevar <- terra::rast(system.file("external/somevar.tif", package = "flexsdm"))
somevar <- terra::aggregate(somevar, fact = 6) # coarser resolution
sp3 <- spp[spp$species == "sp3", ]

sp_part <- part_sband(
  env_layer = somevar,
  data = sp3,
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  type = "lon",
  min_bands = 2,
  max_bands = 4,
  n_part = 2,
  min_occ = 2,
  prop = 0.5
)
sp_part$best_part_info


require(terra)
require(dplyr)
```

```
# Load datasets
data(spp)
f <- system.file("external/somevar.tif", package = "flexsdm")
somevar <- terra::rast(f)

# Example of two longitudinal partitions with presences and absences
single_spp <- spp %>% dplyr::filter(species == "sp1")
part_1 <- part_sband(
  env_layer = somevar,
  data = single_spp,
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  type = "lon",
  min_bands = 2,
  max_bands = 20,
  n_part = 2,
  min_occ = 10,
  prop = 0.5
)

part_1$part # database with partition fold (.part)
part_1$part %>%
  group_by(pr_ab, .part) %>%
  count() # number of presences and absences in each fold
part_1$best_part_info # information of the best partition
part_1$grid # raster with folds

# Explore grid object and presences and absences points
plot(part_1$grid, col = gray.colors(20))
points(part_1$part[c("x", "y")],
  col = rainbow(8)[part_1$part$.part],
  cex = 0.9,
  pch = c(1, 19)[part_1$part$pr_ab + 1]
)


# Example of four latitudinal partition and only presences
single_spp <- spp %>% dplyr::filter(species == "sp1", pr_ab == 1)
part_2 <- part_sband(
  env_layer = somevar,
  data = single_spp,
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  type = "lat",
  min_bands = 8,
  max_bands = 40,
  n_part = 8,
  min_occ = 10,
  prop = 0.5
)
```

```
part_2$part
part_2$best_part_info
part_2$grid

# Explore Grid object and presences points
plot(part_2$grid, col = gray.colors(20))
points(part_2$part[c("x", "y")],
  col = rainbow(8)[part_2$part$.part],
  cex = 0.5,
  pch = 19
)
```

---

part_sblock *Spatial block cross-validation*

---

## Description

This function explores spatial blocks with different cell sizes and returns the most suitable size for a given presence or presence-absence database. The selection of the best grid size is performed automatically considering spatial autocorrelation, environmental similarity, and the number of presence and absence records in each partition.

## Usage

```
part_sblock(
  env_layer,
  data,
  x,
  y,
  pr_ab,
  n_part = 3,
  min_res_mult = 3,
  max_res_mult = 200,
  num_grids = 30,
  min_occ = 10,
  prop = 0.5
)
```

## Arguments

| | |
|---|---|
| env_layer | SpatRaster. Raster with environmental variable. Used to evaluate spatial autocorrelation and environmental similarity between training and testing partitions. Because this function calculate dissimilarity based on Euclidean distances, it can only be used with continuous environmental variables |
| data | data.frame. Data.frame or tibble object with presence (or presence-absence, or presences-pseudo-absence) records, and coordinates |

| | |
|---|---|
| x | character. Column name with spatial x coordinates |
| y | character. Column name with spatial y coordinates |
| pr_ab | character. Column with presence, presence-absence, or pseudo-absence records. Presences must be represented by 1 and absences by 0 |
| n_part | integer. Number of partition. Default 2. |
| min_res_mult | integer. Minimum value used for multiplying raster resolution and define the finest resolution to be tested, default 3. |
| max_res_mult | integer. Maximum value used for multiplying raster resolution and define the coarsest resolution to be tested, default 200. |
| num_grids | integer. Number of grid to be tested between min_res_mult X (raster resolution) and max_res_mult X (raster resolution), default 30 |
| min_occ | numeric. Minimum number of presences or absences in a partition fold. The min_occ value should be base on the amount of predictors in order to avoid over-fitting or error when fitting models for a given fold. Default 10. |
| prop | numeric. Proportion of point used for testing autocorrelation between groups (values > 0 and <=1). The smaller this proportion is, the faster the function will work. Default 0.5 |

## Details

The part_sblock allows test with different numbers of partitions using square blocks (like a checkerboard). This function explores a range of block sizes and automatically selects the best size for a given given presence, presence-absences, or presence-pseudo-absences dataset. Number of partition selection is based on an optimization procedure that explores partition size in three dimensions determined by spatial autocorrelation (measured by Moran's I), environmental similarity (Euclidean distance), and difference in the amount of data among partition groups (Standard Deviation - SD; Velazco et al., 2019). This procedure will iteratively select partitions, first those partitions with autocorrelation values less than the lowest quartile of Morans I, then those with environmental similarity values greater than the third quartile of the Euclidean distances than those with a difference in the amount of data less than the lowest quartile of SD. This selection is repeated until only one partition is retained (Velazco et al., 2019). The main benefit of this partition selection are that it i) is not subjective, ii) balances the environmental similarity and special autocorrelation between partitions, and iii) controls the selection of partitions with too few data that may be problematic for model fitting ("min_occ" argument).

Geographically structured partitions tend to evaluate model transferability more directly than conventional ones (e.g., those performed by `part_random`) (Roberts et al., 2017; Santini et al., 2021), and are relevant for models that are to be used for projections in other regions outside the training area or for other time periods.

This function can interact with `get_block`, `sample_background`, and `sample_pseudoabs` for sampling background points or pseudo-absences within spatial partition groups

## Value

A list with:

- part: A tibble object with information used in 'data' arguments and a additional column .part with partition group.

- best_part_info: A tibble with information about the best partition. It contains the number of the best partition (n_grid), cell size (cell_size), standard deviation of presences (sd_p), standard deviation of absences (sd_a), Moran's I spatial autocorrelation (spa_auto), and environmental similarity based on Euclidean distance (env_sim).
- grid: A SpatRaster object with blocks

## References


- Roberts, D. R., Bahn, V., Ciuti, S., Boyce, M. S., Elith, J., Guillera-Arroita, G., Hauenstein, S., Lahoz-Monfort, J. J., Schroder, B., Thuiller, W., Warton, D. I., Wintle, B. A., Hartig, F., & Dormann, C. F. (2017). Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. Ecography, 40, 913-929. https://doi.org/10.1111/ecog.02881
- Santini, L., Benitez-Lopez, A., Maiorano, L., Cengic, M., & Huijbregts, M. A. J. (2021). Assessing the reliability of species distribution projections in climate change research. Diversity and Distributions, ddi.13252. https://doi.org/10.1111/ddi.13252
- Velazco, S. J. E., Villalobos, F., Galvao, F., & De Marco Junior, P. (2019). A dark scenario for Cerrado plant species: Effects of future climate, land use and protected areas ineffectiveness. Diversity and Distributions, 25(4), 660-673. https://doi.org/10.1111/ddi.12886


## See Also

part_random, part_sband, part_senv, get_block, and plot_res.

## Examples

```
# Toy data: small raster and records of one species to keep this example fast
data("spp")
somevar <- terra::rast(system.file("external/somevar.tif", package = "flexsdm"))
somevar <- terra::aggregate(somevar, fact = 6) # coarser resolution
sp3 <- spp[spp$species == "sp3", ]

sp_part <- part_sblock(
  env_layer = somevar,
  data = sp3,
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  min_res_mult = 10,
  max_res_mult = 40,
  num_grids = 3,
  n_part = 2,
  min_occ = 2,
  prop = 0.5
)
sp_part$best_part_info


require(terra)
require(dplyr)
```

```
# Load datasets
data(spp)
f <- system.file("external/somevar.tif", package = "flexsdm")
somevar <- terra::rast(f)

# Example for one single species
single_spp <- spp %>% dplyr::filter(species == "sp3")
part <- part_sblock(
  env_layer = somevar,
  data = single_spp,
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  min_res_mult = 10,
  max_res_mult = 500,
  num_grids = 30,
  n_part = 2,
  min_occ = 5,
  prop = 0.5
)
part

part$part # database with partition fold (.part)
part$part %>%
  group_by(pr_ab, .part) %>%
  count() # number of presences and absences in each fold
part$best_part_info # information of the best partition
part$grid # raster with folds

# Explore the Grid object

plot(part$grid)
points(part$part[c("x", "y")],
  col = c("blue", "red")[part$part$.part],
  cex = 0.5,
  pch = 19
)

terra::res(part$grid)
terra::res(somevar)

# Note that this is a layer with block partition, but it has a
# different resolution than the original environmental variables.
# If you wish have a layer with the same properties
# (i.e. resolution, extent, NAs) as your original environmental
# variables you can use the \code{\link{get_block}} function.

grid_env <- get_block(env_layer = somevar, best_grid = part$grid)

plot(grid_env) # this is a block layer with the same layer
# properties as environmental variables.
points(part$part[c("x", "y")],
  col = c("blue", "red")[part$part$.part],
```

```
  cex = 0.5,
  pch = 19
)
# This layer is very useful if you need to sample
# pseudo_absence or background point
# See examples in \code{\link{backgroudp}} and \code{\link{pseudoabs}}


# Example of a higher number of partitions
part <- part_sblock(
  env_layer = somevar,
  data = single_spp,
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  min_res_mult = 10,
  max_res_mult = 500,
  num_grids = 30,
  n_part = 4,
  min_occ = 2,
  prop = 0.5
)

# Explore the Grid object
plot(part$grid, col = gray.colors(4))
points(part$part[c("x", "y")],
  col = rainbow(n = 4)[part$part$.part],
  cex = 0.5,
  pch = 19
)


# Using these functions with several species
spp2 <- split(spp, spp$species)
class(spp2)
length(spp2)
names(spp2)

part_list <- lapply(spp2, function(x) {
  result <- part_sblock(
    env_layer = somevar,
    data = x,
    x = "x",
    y = "y",
    pr_ab = "pr_ab",
    min_res_mult = 10,
    max_res_mult = 500,
    num_grids = 30,
    n_part = 2,
    min_occ = 5,
    prop = 0.5
  )
  result
```

```
})

part_list$sp3 # For this dataset a suitable partition was not found

# Create a single database for all species
occ_part <- lapply(part_list, function(x) {
  if (!length(x) > 0) {
    x[[1]]
  }
}) %>%
  dplyr::bind_rows(.id = "species")
occ_part

# Get the best grid info for all species
grid_info <- dplyr::bind_rows(lapply(
  part_list,
  function(x) x[[2]]
), .id = "species")

# Get the best grid layer for all species
grid_layer <- lapply(part_list, function(x) x$grid)
grid_layer2 <-
  lapply(grid_layer, function(x) {
    get_block(env_layer = somevar[[1]], best_grid = x)
  })
grid_layer2 <- terra::rast(grid_layer2)
grid_layer2
plot(grid_layer2)


# Block partition for presences-only database
single_spp <- spp %>%
  dplyr::filter(species == "sp1", pr_ab == 1)
single_spp
single_spp$pr_ab %>% unique() # only presences

part <- part_sblock(
  env_layer = somevar,
  data = single_spp,
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  min_res_mult = 10,
  max_res_mult = 500,
  num_grids = 30,
  n_part = 4,
  min_occ = 10,
  prop = 0.5
)

part$part %>% dim()
part$best_part_info
part$grid
```

```
plot(part$grid)
points(
  part$part[c("x", "y")],
  col = c("blue", "red", "green", "black")[part$part$.part],
  cex = 0.5,
  #' pch = 19
)
```

---

part_senv *Environmental and spatial cross-validation*

---

## Description

This function explores different numbers of environmental partitions (clusters) based on the K-means clustering algorithm and returns the number of partitions best suited for a given presence, presence-absences, or presence-pseudo-absences database. Selection of the best number of partitions is performed automatically considering spatial autocorrelation, environmental similarity, and the number of presence and/or absence records in each partition.

## Usage

```
part_senv(
  env_layer,
  data,
  x,
  y,
  pr_ab,
  min_n_groups = 2,
  max_n_groups = 10,
  min_occ = 10,
  prop = 0.5,
  include_coords = TRUE
)
```

## Arguments

| | |
|---|---|
| env_layer | SpatRaster. Raster with environmental variable. This will be used to evaluate spatial autocorrelation and environmental similarity between training and testing partitions. Because this function calculate dissimilarity based on Euclidean distances, it can only be used with continuous variables |
| data | data.frame. Data.frame or tibble object with presence (or presence-absence, or presences-pseudo-absence) records, and coordinates |
| x | character. Column name with spatial x coordinates |
| y | character. Column name with spatial y coordinates |

| | |
|---|---|
| pr_ab | character. Column with presences, presence-absence, or pseudo-absence. Presences must be represented by 1 and absences by 0 |
| min_n_groups | integer. Minimum number of groups to be tested. Default 2. |
| max_n_groups | integer. Maximum number of groups to be tested. Default 10. |
| min_occ | numeric. Minimum number of presences or absences in a partition fold. The min_occ value should be base on the amount of predictors in order to avoid over-fitting or error when fitting models for a given fold. Default 10. |
| prop | numeric. Proportion of point used for testing autocorrelation between groups (values > 0 and <=1). The smaller this number is, the faster the function will work. Default 0.5 |
| include_coords | logical. Whether to include coordinates in the clustering. If False, only environmental variables will be used for clustering. Default TRUE |

## Details

The part_sblock allows test with different numbers of partitions defined in the envirnomental clusters delimited the K-mean cluster algorithm. This function explores a range of environmental clusters and automatically selects the best number of clusters for a given presence, presence-absences, or presence-pseudo-absences dataset. Such selection of the number of clusters is based on an optimization procedure that explores partition size in three dimensions determined by spatial autocorrelation (measured by Moran's I), environmental similarity (Euclidean distance), and difference in the amount of data among clusters (Standard Deviation - SD; Velazco et al., 2019). This procedure will cyclically select those partitions with autocorrelation values less than the lowest quartile of Moran's I, then those with environmental similarity values greater than the third quartile of the Euclidean distances than those with a difference in the amount of data less than the lowest quartile of SD. This selection is repeated until only one partition is retained (Velazco et al., 2019). The main benefit of this partition selection are i) this is not subjective, ii) balances the environmental similarity and special autocorrelation between partitions, and iii) controls the partition selection with few data that may be problematic for model fitting ("min_occ" argument)..

Partitions environmentally or geographically structured tend to evaluate model transferability more directly than conventional ones (e.g., those performed by `part_random`) (Roberts et al., 2017; Santini et al., 2021), being relevant for models that want to be used for projections in other regions outside the training area or for other periods.

This function can interact with `get_block`, `sample_background`, and `sample_pseudoabs` for sampling background points or pseudo-absences within spatial partition groups

## Value

A list with:

- part: A tibble object with information used in 'data' arguments and a additional column .part with partition group.
- best_part_info: A tibble with information about the best partition. It contains the number of partition (n_groups), standard deviation of presences (sd_p), standard deviation of absences (sd_a), Moran's I spatial autocorrelation (spa_auto) and environmental similarity based on Euclidean distance (env_sim)

## References


- Roberts, D. R., Bahn, V., Ciuti, S., Boyce, M. S., Elith, J., Guillera-Arroita, G., Hauenstein, S., Lahoz-Monfort, J. J., Schroder, B., Thuiller, W., Warton, D. I., Wintle, B. A., Hartig, F., & Dormann, C. F. (2017). Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. Ecography, 40, 913-929. https://doi.org/10.1111/ecog.02881
- Santini, L., Benitez-Lopez, A., Maiorano, L., Cengic, M., & Huijbregts, M. A. J. (2021). Assessing the reliability of species distribution projections in climate change research. Diversity and Distributions, ddi.13252. https://doi.org/10.1111/ddi.13252
- Velazco, S. J. E., Villalobos, F., Galvao, F., & De Marco Junior, P. (2019). A dark scenario for Cerrado plant species: Effects of future climate, land use and protected areas ineffectiveness. Diversity and Distributions, 25(4), 660-673. https://doi.org/10.1111/ddi.12886


## See Also

part_random, part_sblock, and part_sband

## Examples

```
# Toy data: small raster and records of one species to keep this example fast
data("spp")
somevar <- terra::rast(system.file("external/somevar.tif", package = "flexsdm"))
somevar <- terra::aggregate(somevar, fact = 6) # coarser resolution
sp3 <- spp[spp$species == "sp3", ]
sp1 <- spp[spp$species == "sp1", ][1:300, ]

part_toy <- part_senv(
  env_layer = somevar,
  data = sp1,
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  min_n_groups = 2,
  max_n_groups = 3,
  min_occ = 5,
  prop = 0.2
)
part_toy$best_part_info


require(terra)
require(ggplot2)
require(dplyr)

f <- system.file("external/somevar.tif", package = "flexsdm")
somevar <- terra::rast(f)

# Select a species
spp1 <- spp %>% dplyr::filter(species == "sp1")

part1 <- part_senv(
  env_layer = somevar,
```

```
  data = spp1,
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  min_n_groups = 2,
  max_n_groups = 10,
  min_occ = 10,
  prop = 0.2,
  include_coords = TRUE
)

part1

ggplot(part1$part, aes(x, y, col = factor(.part))) +
  geom_point(aes(shape = factor(pr_ab)))

ggplot(part1$part, aes(x, y, col = factor(.part))) +
  geom_point(aes(shape = factor(pr_ab))) +
  facet_wrap(. ~ .part)

ggplot(part1$part, aes(x, y, col = factor(.part))) +
  geom_point(aes(shape = factor(pr_ab))) +
  facet_wrap(. ~ pr_ab)
```

---

`plot_res` *Plot different resolutions to be used in part_sblock*

---

## Description

This function is useful to display the maximum and minimum resolution that you want to test with the block_partition function. Note that if the resolution to be tested is very fine, the plot display may take a long time.

## Usage

```
plot_res(r, res_mult)
```

## Arguments

| | |
|---|---|
| `r` | SpatRaster. A raster layer, preferably a layer of environmental variables to be used |
| `res_mult` | numeric. Maximum or minimum resolution to be tested. |

## Value

A plot with the original raster overlapped by a grid with the resolution used

## Examples

```
f <- system.file("external/somevar.tif", package = "flexsdm")
r <- terra::rast(f)
r <- terra::aggregate(r$CFP_1, fact = 6) # coarser resolution to keep this example fast
plot_res(r, res_mult = 10)


f <- system.file("external/somevar.tif", package = "flexsdm")
r <- terra::rast(f)
r <- r$CFP_1
plot_res(r, res_mult = 100)
plot_res(r, res_mult = 200)
```

---

# sample_background *Sample background points*

---

## Description

Sampling background points with the options of using different geographical restrictions and sampling methods.

## Usage

```
sample_background(
  data,
  x,
  y,
  n,
  method = "random",
  rlayer,
  maskval = NULL,
  calibarea = NULL,
  rbias = NULL,
  sp_name = NULL
)
```

## Arguments

| | |
|---|---|
| data | data.frame or tibble. Database with presences records, and coordinates |
| x | character. Column name with spatial x coordinates |
| y | character. Column name with spatial y coordinates |
| n | integer. Number of background point to be sampled |
| method | character. Background allocation method. The methods implemented are:<br>• random: Random allocation of background points. Usage method = 'random' |

- thickening: Thickening background points is based on the Vollering et al. (2019) method. For this method, a buffer width must be defined that will be used around presences points. A buffer can be defined using the argument as method = c("thickening", width = 20000). Buffer width must be in m if raster (used in rlayer) has a longitude/latitude CRS, or map units in other cases. If a buffer width is not provided the function will use a width value equal to the mean of the pair-wise presence distances. If a width value is not provided, the argument must be used as method = 'thickening'.
- biased: This method, similar to "thickening", sample background biased with the same bias of presences. However, the background points are sampled used a presences probability throughout the entire study area, and not restricting such bias within buffers as in the "thickening" approach. For using this method, it is necessary to provide a layer with presences bias in "rbias" argument (Phillips et al., 2009).

Usage method='thickening' or method = c("thickening", width = 20000). Default 'random'

`rlayer` SpatRaster used for sampling background points. It is best to use a layer with the same resolution and extent that environmental variables that will be used for modeling. If using maskval argument, this raster layer must contain the values to constrain sampling

`maskval` integer, character, or factor. Values of the raster layer used for constraining sampling of background points

`calibarea` SpatVect that delimits the calibration area used for a given species (see calib_area function).

`rbias` SpatRaster used for choosing background points using the bias method. A raster with bias data must be provided. It is recommended that rbias match resolution and extent of rlayer.

`sp_name` character. Species name for which the output will be used. If this argument is used, the first output column will have the species name. Default NULL.

## Value

A tibble object with x y coordinates of sampled background points

## References


- Phillips, S. J., Dudík, M., Elith, J., Graham, C. H., Lehmann, A., Leathwick, J., & Ferrier, S. (2009). Sample selection bias and presence-only distribution models: Implications for background and pseudo-absence data. Ecological Applications, 19(1), 181-197.
- Vollering, J., Halvorsen, R., Auestad, I., & Rydgren, K. (2019). Bunching up the background betters bias in species distribution models. Ecography, 42(10), 1717-1727. https://doi.org/10.1111/ecog.04503


## See Also

`sample_pseudoabs` and `calib_area`.

## Examples

```
# Toy data: small raster and records of one species to keep this example fast
data("spp")
somevar <- terra::rast(system.file("external/somevar.tif", package = "flexsdm"))
somevar <- terra::aggregate(somevar, fact = 6) # coarser resolution
sp3 <- spp[spp$species == "sp3", ]
sp3 <- sp3[sp3$pr_ab == 1, ] # presences only

bg <- sample_background(
  data = sp3,
  x = "x",
  y = "y",
  n = 100,
  method = "random",
  rlayer = somevar[[1]]
)
bg


require(terra)
require(dplyr)
data(spp)
somevar <- system.file("external/somevar.tif", package = "flexsdm")
somevar <- terra::rast(somevar)

# Example for a single species
spp_pa <- spp %>% dplyr::filter(species == "sp3")

# Spatially structured partition
part <- part_sblock(
  env_layer = somevar,
  data = spp_pa,
  x = "x",
  y = "y",
  pr_ab = "pr_ab",
  min_res_mult = 100,
  max_res_mult = 500,
  num_grids = 30,
  min_occ = 5,
  n_part = 2
)

grid_env <- get_block(env_layer = somevar, best_grid = part$grid)
plot(grid_env)


## %#######################################################%##
#                                                           #
####             Random background method               ####
#                                                           #
## %#######################################################%##
```

```
# Sample background points throughout study area with random sampling method
spp_p <- spp_pa %>% dplyr::filter(pr_ab == 1)
bg <-
  sample_background(
    data = spp_p,
    x = "x",
    y = "y",
    n = 1000,
    method = "random",
    rlayer = grid_env,
    sp_name = "sp3"
  )

bg
plot(grid_env)
points(bg[-1])

# Sample random background points constrained to a region with a give set of values
plot(grid_env)
sample_background(
  data = spp_p,
  x = "x",
  y = "y",
  n = 1000,
  method = "random",
  rlayer = grid_env,
  maskval = 1
) %>% points()

plot(grid_env)
sample_background(
  data = spp_p,
  x = "x",
  y = "y",
  n = 1000,
  method = "random",
  rlayer = grid_env,
  maskval = 2
) %>% points()

plot(grid_env)
sample_background(
  data = spp_p,
  x = "x",
  y = "y",
  n = 1000,
  method = "random",
  rlayer = grid_env,
  maskval = c(1, 2)
) %>% points()

# Sample random background within a calibration area and constrained to a region
ca_ps1 <- calib_area(
```

```
  data = spp_pa,
  x = "x",
  y = "y",
  method = c("buffer", width = 50000),
  crs = crs(somevar)
)
plot(grid_env)
plot(ca_ps1, add = TRUE)
points(spp_pa[-1], col = "blue", cex = 0.7, pch = 19)
sample_background(
  data = spp_p,
  x = "x",
  y = "y",
  n = 1000,
  method = "random",
  rlayer = grid_env,
  maskval = 1,
  calibarea = ca_ps1
) %>%
  points(col = "red")


## %#######################################################%##
#                                                          #
####            Thickening background method            ####
#                                                          #
## %#######################################################%##

# Thickening background without constraining them
spp_p # presences database of a species
grid_env # The raster layer used for sampling background
bg <- sample_background(
  data = spp_p,
  x = "x",
  y = "y",
  n = 5000,
  method = "thickening",
  rlayer = grid_env,
)

plot(grid_env)
bg %>%
  points(col = "red")


# Thickening background
spp_p # presences database of a species
grid_env # The raster layer used for sampling background
bg <- sample_background(
  data = spp_p,
  x = "x",
  y = "y",
  n = 5000,
```

```
  method = c("thickening", width = 150000),
  rlayer = grid_env
)

plot(grid_env)
bg %>%
  points(col = "red")

# Sample thickening background within a calibration area and constrained to a region
bg <- sample_background(
  data = spp_p,
  x = "x",
  y = "y",
  n = 3000,
  method = "thickening",
  rlayer = grid_env,
  maskval = 2,
  calibarea = ca_ps1
)

plot(grid_env)
plot(ca_ps1, add = TRUE)
bg %>%
  points(col = "red", cex = 0.3)
points(spp_p[c("x", "y")], pch = 19)

## %#######################################################%##
#                                                          #
####             Biased background method               ####
#                                                          #
## %#######################################################%##
require(dplyr)
require(terra)
data(spp)

# Select the presences of a species
spp_p <- spp %>% dplyr::filter(species == "sp1", pr_ab == 1)

# Raster layer with density of points to obtain a biased sampling background
occ_density <- system.file("external/occ_density.tif", package = "flexsdm")
occ_density <- terra::rast(occ_density)
plot(occ_density)
points(spp_p %>% dplyr::select(x, y), cex = 0.5)

# A layer with region used to contrain background sampling area
regions <- system.file("external/regions.tif", package = "flexsdm")
regions <- terra::rast(regions)
plot(regions)
points(spp_p %>% dplyr::select(x, y), cex = 0.5)


# Biased background points
spp_p # presences database of a species
```

```
bg <- sample_background(
  data = spp_p,
  x = "x",
  y = "y",
  n = 3000,
  method = "biased",
  rlayer = regions,
  rbias = occ_density
)

plot(occ_density)
bg %>%
  points(col = "red", cex = 0.1)
spp_p %>%
  dplyr::select(x, y) %>%
  points(., col = "black", pch = 19, cex = 0.5)


# Biased background points constrained to a region
# It will be selected region 6
plot(regions)
plot(regions %in% c(1, 6))

bg <- sample_background(
  data = spp_p,
  x = "x",
  y = "y",
  n = 500,
  method = "biased",
  rlayer = regions,
  rbias = occ_density,
  maskval = c(1, 2)
)

plot(occ_density)
bg %>%
  points(col = "red", cex = 0.5)
spp_p %>%
  dplyr::select(x, y) %>%
  points(., col = "black", pch = 19, cex = 0.5)
```

---

`sample_pseudoabs` *Sample pseudo-absences*

---

## Description

This function provide several methods for sampling pseudo-absences, for instance totally random sampling method, with the options of using different environmental and or geographical constraints.

### Usage

```
sample_pseudoabs(
  data,
  x,
  y,
  n,
  method,
  rlayer,
  maskval = NULL,
  calibarea = NULL,
  sp_name = NULL
)
```

### Arguments

`data` data.frame or tibble. Database with presences (or presence-absence, or presences-pseudo-absence) records, and coordinates

`x` character. Column name with spatial x coordinates

`y` character. Column name with spatial y coordinates

`n` integer. Number of pseudo-absences to be sampled

`method` character. Pseudo-absence allocation method. It is necessary to provide a vector for this argument. The methods implemented are:

- random: Random allocation of pseudo-absences throughout the area used for model fitting. Usage method='random'.
- kmeans: Pseudo-absences are sampled randomly within environmental clusters defined by K-means cluster analysis (the number of clusters will equal than number of pseudo-absences defined in 'n'). For this method, it is necessary to provide a raster object with environmental variables. Usage method = c('kmeans', env = somevar). throughout the area used for model fitting. Usage method='random'.
- env_const: Pseudo-absences are environmentally constrained to regions with lower suitability values predicted by a Bioclim model. Areas with suitability values < 0.1 are selected as candidate regions to sample pseudo-absences. For this method, it is necessary to provide a raster object with environmental variables Usage method=c(method='env_const', env = somevar).
- env_const_kmeans: A combination of 'env_const' and 'kmeans' methods.
- geo_const: Pseudo-absences are allocated far from occurrences based on a geographical buffer. A value of the buffer width in m must be provided if raster (used in rlayer) has a longitude/latitude CRS, or in map units in other cases. Usage method=c('geo_const', width='50000').
- geo_const_kmeans: A combination of 'geo_const' and 'kmeans' methods.
- geoenv_const: Pseudo-absences are constrained environmentally (based on Bioclim model) and distributed geographically far from occurrences based on a geographical buffer. For this method, a raster with environmental variables stored as SpatRaster object should be provided. A value of the

buffer width in m must be provided if raster (used in rlayer) has a longitude/latitude CRS, or in map units in other cases. Usage method=c('geoenv_const', width='50000', env = somevar).

- geoenv_const_kmeans: Pseudo-absences are constrained using a three-level procedure; it is similar to the geoenv_const with an additional step which distributes the pseudo-absences in the environmental space using K-means cluster analysis. For this method, it is necessary to provide a raster object with environmental variables and a value of the buffer width in m if raster (used in rlayer) has a longitude/latitude CRS, or map units in other cases. Usage method=c('geoenv_const_kmeans', width='50000', env = somevar).

`rlayer` SpatRaster. A raster layer used for sampling pseudo-absence A layer with the same resolution and extent that environmental variables that will be used for modeling is recommended. In the case use maskval argument, this raster layer must contain the values used to constrain sampling

`maskval` integer, character, or factor. Values of the raster layer used for constraining the pseudo-absence sampling

`calibarea` SpatVector A SpatVector which delimit the calibration area used for a given species (see `calib_area` function).

`sp_name` character. Species name for which the output will be used. If this argument is used, the first output column will have the species name. Default NULL.

## Value

A tibble object with x y coordinates of sampled pseudo-absence points

## See Also

`sample_background` and `calib_area`.

## Examples

```
# Toy data: small raster and records of one species to keep this example fast
data("spp")
somevar <- terra::rast(system.file("external/somevar.tif", package = "flexsdm"))
somevar <- terra::aggregate(somevar, fact = 6) # coarser resolution
sp3 <- spp[spp$species == "sp3", ]
sp3 <- sp3[sp3$pr_ab == 1, ] # presences only

ps <- sample_pseudoabs(
  data = sp3,
  x = "x",
  y = "y",
  n = 50,
  method = "random",
  rlayer = somevar[[1]]
)
ps
```

```
require(terra)
require(dplyr)
data("spp")

somevar <- system.file("external/somevar.tif", package = "flexsdm")
somevar <- terra::rast(somevar)

regions <- system.file("external/regions.tif", package = "flexsdm")
regions <- terra::rast(regions)

plot(regions)


single_spp <-
  spp %>%
  dplyr::filter(species == "sp3") %>%
  dplyr::filter(pr_ab == 1) %>%
  dplyr::select(-pr_ab)


# Pseudo-absences randomly sampled throughout study area
ps1 <-
  sample_pseudoabs(
    data = single_spp,
    x = "x",
    y = "y",
    n = nrow(single_spp) * 10,
    method = "random",
    rlayer = regions,
    maskval = NULL,
    sp_name = "sp3"
  )
plot(regions, col = gray.colors(9))
points(single_spp[-1], col = "blue", cex = 0.7, pch = 19) # presences
points(ps1[-1], col = "red", cex = 0.7, pch = 19) # absences


# Pseudo-absences randomly sampled within a regions where a species occurs
## Regions where this species occurrs
samp_here <- terra::extract(regions, single_spp[2:3])[, 2] %>%
  unique() %>%
  na.exclude()

ps1 <-
  sample_pseudoabs(
    data = single_spp,
    x = "x",
    y = "y",
    n = nrow(single_spp) * 10,
    method = "random",
    rlayer = regions,
    maskval = samp_here
  )
```

```
plot(regions, col = gray.colors(9))
points(single_spp[-1], col = "blue", cex = 0.7, pch = 19)
points(ps1, col = "red", cex = 0.7, pch = 19)

# Pseudo-absences sampled with K-means approach
set.seed(123)
ps1 <-
  sample_pseudoabs(
    data = single_spp,
    x = "x",
    y = "y",
    n = nrow(single_spp) * 10,
    method = c(method = "kmeans", env = somevar),
    rlayer = regions
  )

plot(regions, col = gray.colors(9))
points(single_spp[-1], col = "blue", cex = 0.7, pch = 19)
points(ps1, col = "red", cex = 0.7, pch = 19)

# Pseudo-absences sampled with geographical constraint
ps1 <-
  sample_pseudoabs(
    data = single_spp,
    x = "x",
    y = "y",
    n = nrow(single_spp) * 10,
    method = c("geo_const", width = "30000"),
    rlayer = regions,
    maskval = samp_here
  )
plot(regions, col = gray.colors(9))
points(single_spp[-1], col = "blue", cex = 0.7, pch = 19)
points(ps1, col = "red", cex = 0.7, pch = 19)

# Pseudo-absences sampled with environmental constraint
ps1 <-
  sample_pseudoabs(
    data = single_spp,
    x = "x",
    y = "y",
    n = nrow(single_spp) * 10,
    method = c("env_const", env = somevar),
    rlayer = regions,
    maskval = samp_here
  )
plot(regions, col = gray.colors(9))
points(single_spp[-1], col = "blue", cex = 0.7, pch = 19)
points(ps1, col = "red", cex = 0.7, pch = 19)

# Pseudo-absences sampled with environmental and geographical constraint
ps1 <-
```

```
  sample_pseudoabs(
    data = single_spp,
    x = "x",
    y = "y",
    n = nrow(single_spp) * 10,
    method = c("geoenv_const", width = "50000", env = somevar),
    rlayer = regions,
    maskval = samp_here
  )
plot(regions, col = gray.colors(9))
points(single_spp[-1], col = "blue", cex = 0.7, pch = 19)
points(ps1, col = "red", cex = 0.7, pch = 19)

# Pseudo-absences sampled with environmental and geographical constraint and with k-mean clustering
ps1 <-
  sample_pseudoabs(
    data = single_spp,
    x = "x",
    y = "y",
    n = nrow(single_spp) * 10,
    method = c("geoenv_const_kmeans", width = "50000", env = somevar),
    rlayer = regions,
    maskval = samp_here
  )
plot(regions, col = gray.colors(9))
points(single_spp[-1], col = "blue", cex = 0.7, pch = 19)
points(ps1, col = "red", cex = 0.7, pch = 19)

# Sampling pseudo-absence using a calibration area
ca_ps1 <- calib_area(
  data = single_spp,
  x = "x",
  y = "y",
  method = c("buffer", width = 50000),
  crs = crs(somevar)
)
plot(regions, col = gray.colors(9))
plot(ca_ps1, add = TRUE)
points(single_spp[-1], col = "blue", cex = 0.7, pch = 19)

ps1 <-
  sample_pseudoabs(
    data = single_spp,
    x = "x",
    y = "y",
    n = nrow(single_spp) * 50,
    method = "random",
    rlayer = regions,
    maskval = NULL,
    calibarea = ca_ps1
  )
plot(regions, col = gray.colors(9))
plot(ca_ps1, add = TRUE)
```

```
points(ps1, col = "red", cex = 0.7, pch = 19)
points(single_spp[-1], col = "blue", cex = 0.7, pch = 19)


ps1 <-
  sample_pseudoabs(
    data = single_spp,
    x = "x",
    y = "y",
    n = nrow(single_spp) * 50,
    method = "random",
    rlayer = regions,
    maskval = samp_here,
    calibarea = ca_ps1
  )
plot(regions, col = gray.colors(9))
plot(ca_ps1, add = TRUE)
points(ps1, col = "red", cex = 0.7, pch = 19)
points(single_spp[-1], col = "blue", cex = 0.7, pch = 19)
```

---

`sdm_directory` *Create directories for saving the outputs of the flexsdm*

---

## Description

This function assists in creating a directory system with different sub-folders to assist in the organisation of the modelling process outputs.

## Usage

```
sdm_directory(
  main_dir,
  projections = NULL,
  calibration_area = TRUE,
  algorithm = NULL,
  ensemble = NULL,
  threshold = FALSE,
  return_vector = TRUE
)
```

## Arguments

| | |
|---|---|
| `main_dir` | character. Directory path containing main folder for saving model inputs and outputs. There is no default: the path must be provided by the user (e.g., in examples use a sub-folder of `tempdir()`). The folder is created if it does not exist. |
| `projections` | vector. Vector of folder names for future scenarios/different regions/time periods to save model projections output. |

`calibration_area`
: logical. If TRUE, function creates a folder in 1_Inputs for storing calibration area. Default TRUE

`algorithm`
: vector. Vector of model names that will be used. Usage algorithm = c(gam, tune_max, tune_net, esm_glm). If "all" is used the function creates folders with all algorithms available in flexsdm . i.e. 'gam', 'gau', 'gbm', 'glm', 'max', 'net', 'raf', and 'svm'. Default NULL

`ensemble`
: vector. Vector of methods used to ensemble different models. Usage ensemble = c("mean", "meanthr"). Default NULL

`threshold`
: logical. If TRUE sub-folders "/1_con", "/2_bin" will be created within each algorithm and/or ensemble folder. Used for storing continuous and binarized models separately. Default FALSE

`return_vector`
: logical. If TRUE function returns a vector with the path to all folders. Default TRUE

## Details

The sdm_directory function assists in saving workflow outputs by creating folders (directories) based on user specifications, such as choice of algorithms, ensemble methods, and model projections to new geographic regions or periods. The function first creates two folders within the user-specified project folder, one for model inputs (1_Inputs) and one for model outputs (2_Outputs). Within 1_Inputs, there are three sub-folders for users to store model inputs: 1_Occurrences, 2_Predictors, and 3_Calibration_area. If a user chooses to include projections in their modeling framework, the 2_Projections subfolder is created within the 2_Predictors folder to store the environmental data for the projection scenarios provided to the "projections" argument. Additionally, sdm_directory offers users enhanced flexibility in saving their modeling outputs, giving them offers users the option to save results from any modeling and ensemble technique presented in flexsdm

## Value

A character vector with the paths to created folders

## Examples

```
# Folders are created in a temporary directory; in a real analysis use
# the path of your own project folder
my_project <- file.path(tempdir(), "my_project_name")
# my_project <- file.path(getwd(), "flexsdm_results") # create folders in the working directory
dirs_1 <- sdm_directory(
  main_dir = my_project,
  projections = NULL,
  calibration_area = TRUE,
  algorithm = c("gam", "tune_max"),
  ensemble = c("mean", "meanthr"),
  threshold = FALSE,
  return_vector = TRUE
)
dirs_1
unlink(my_project, recursive = TRUE) # remove the folders created in this example
```

```
# With folders for model projections and for thresholded outputs
dirs_2 <- sdm_directory(
  main_dir = my_project,
  projections = c("cnrm_rpc8.5_2050", "cnrm_rpc4.5_2050"),
  calibration_area = TRUE,
  algorithm = "all",
  ensemble = c("mean", "meanthr"),
  threshold = TRUE
)
head(dirs_2)
unlink(my_project, recursive = TRUE)
```

---

sdm_eval *Calculate different model performance metrics*

---

## Description

This function calculates threshold dependent and independent model performance metrics.

## Usage

```
sdm_eval(p, a, bg = NULL, thr = NULL)
```

## Arguments

| | |
|---|---|
| p | numeric. Predicted suitability for presences |
| a | numeric. Predicted suitability for absences |
| bg | numeric. Predicted suitability for background points, used for BOYCE metric. If not provided (NULL), the Boyce index will be calculated using absence data instead. **Note:** Using absence data for the Boyce index is not standard practice and may result in inflated performance values. It is highly recommended to provide background points for a correct calculation. The Boyce index is calculated using the `boyce` function, which is an adaptation of the method implemented in the `ecospat` package. |
| thr | character. Threshold criterion used to get binary suitability values (i.e. 0,1). Used for threshold-dependent performance metrics. It is possible to use more than one threshold type. A vector must be provided for this argument. The following threshold criteria are available:<br>• lpt: The highest threshold at which there is no omission.<br>• equal_sens_spec: Threshold at which the Sensitivity and Specificity are equal.<br>• max_sens_spec: Threshold at which the sum of the Sensitivity and Specificity is the highest (aka threshold that maximizes the TSS).<br>• max_jaccard: The threshold at which the Jaccard index is the highest.<br>• max_sorensen: The threshold at which the Sorensen index is the highest. |

- max_fpb: The threshold at which FPB (F-measure on presence-background data) is the highest.
- sensitivity: Threshold based on a specified Sensitivity value. Usage thr = c('sensitivity', sens='0.6') or thr = c('sensitivity'). 'sens' refers to Sensitivity value. If a sensitivity value is not specified, the default value is 0.9

If more than one threshold type is used, concatenate threshold types, e.g., thr=c('lpt', 'max_sens_spec', 'max_jaccard'), or thr=c('lpt', 'max_sens_spec', 'sensitivity', sens='0.8'), or thr=c('lpt', 'max_sens_spec', 'sensitivity'). Function will use all thresholds if no threshold type is specified

## Details

This function is used for evaluating different models approaches base on the combination of presence-absences or presence-pseudo-absences and background point data and suitability predicted by any model or flexsdm modeling function families (fit_, esm_, and tune_.)

**Performance Metrics Formulas:**

It calculates the next performance metric:

| Performance metric | Threshold dependent | Values ranges |
|---|---|---|
| TPR (True Positive Rate, also called Sensitivity) | yes | 0 - 1 |
| TNR (True Negative Rate, also called Specificity) | yes | 0 - 1 |
| W_TPR_TNR (Weighted TPR-TNR; Jadhav 2020) | yes | 0 - 1 |
| SORENSEN | yes | 0 - 1 |
| JACCARD | yes | 0 - 1 |
| FPB (F-measure on presence-background) | yes | 0 - 2 |
| OR (Omission Rate) | yes | 0 - 1 |
| TSS (True Skill Statistic) | yes | -1 - 1 |
| KAPPA | yes | -1 - 1 |
| MCC (Matthews Correlation Coefficient; Matthews 1975) | yes | -1 - 1 (1 is best) |
| AUC (Area Under Curve) | no | 0 - 1 |
| BOYCE (continuous Boyce index)* | no | -1 - 1 |
| IMAE (Inverse Mean Absolute Error)** | no | 0 - 1 |
| CRPS (Continuous Ranked Probability Score based on Brier Score, Brier 1950)** | no | 0 - 1 |

* The continuous Boyce index is calculated based on presences and background points. If background points are not provided, it will be calculated using presences and absences (or pseudo-absences), which is not standard and may lead to misleading results. The code for calculating this metric is an adaptation of the `enmSdmX` package.

** IMAE and CRPS are calculated as 1-(Mean Absolute Error) and 1-(CRPS), respectively, in order to be consistent with the other metrics where the higher the value of a given performance metric, the greater the model's.

To define the formulas, the following components of the confusion matrix are used:

- `tp`: True Positives (presences correctly predicted as presences)
- `tn`: True Negatives (absences correctly predicted as absences)
- `fp`: False Positives (absences incorrectly predicted as presences)

- `fn`: False Negatives (presences incorrectly predicted as absences)
- `np`: Number of presences (`length(p)`)
- `na`: Number of absences (`length(a)`)

The formulas are:

- **TPR (Sensitivity)**:
$$TPR = \frac{tp}{tp + fn}$$
- **TNR (Specificity)**:
$$TNR = \frac{tn}{tn + fp}$$
- **W_TPR_TNR** (Li et al. 2020):
$$W_TPR_TNR = w \cdot TPR + (1 - w) \cdot TNR$$
, where
$$w = \frac{na}{na + np}$$
- **SORENSEN**:
$$Sorensen = \frac{2 \cdot tp}{fn + 2 \cdot tp + fp}$$
- **JACCARD**:
$$Jaccard = \frac{tp}{fn + tp + fp}$$
- **FPB**:
$$FPB = 2 \cdot Jaccard$$
- **OR**:
$$OR = 1 - TPR$$
- **TSS**:
$$TSS = TPR + TNR - 1$$
- **KAPPA**:
$$KAPPA = \frac{Pr(a) - Pr(e)}{1 - Pr(e)}$$
, where
$$Pr(a) = \frac{tp + tn}{tp + tn + fp + fn}$$
and
$$Pr(e) = \frac{(tp + fp)(tp + fn) + (fn + tn)(fp + tn)}{(tp + tn + fp + fn)^2}$$
- **MCC** (Matthews 1975):
$$MCC = \frac{(tp \cdot tn) - (fp \cdot fn)}{\sqrt{(tp + fp)(tp + fn)(tn + fp)(tn + fn)}}$$
- **AUC**: Calculated as the Wilcoxon-Mann-Whitney U statistic, which is equivalent to the area under the ROC curve.

- **BOYCE**: (Hirzel et al. 2006): The continuous Boyce index, which measures how model predictions differ from a random distribution of observed presences across the prediction gradient.
- **CRPS** (Brier 1950): For binary outcomes, this is calculated as

$$1 - \frac{\sum(predicted - observed)^2}{N}$$

, which is 1 minus the Brier Score.
- **IMAE**:

$$IMAE = 1 - \frac{\sum|predicted - observed|}{N}$$

, where N is the total number of records.

## Value

a tibble with next columns

- threshold: threshold names
- thr_value: threshold values
- n_presences: number of presences
- n_absences: number of absences
- from TPR to CRPS: performance metrics

## References


- Brier GW. (1950) Verification of forecasts expressed in terms of probability. Monthly Weather Review 78(1): 1–3. https://doi.org/10.1175/1520-0493(1950)078<0001:VOFEIT>2.0.CO;2
- Jadhav, AS. (2020). A novel weighted TPR-TNR measure to assess performance of the classifiers. Expert Systems with Applications 152, 113391. https://doi.org/10.1016/j.eswa.2020.113391
- Matthews BW. (1975) Comparison of the predicted and observed secondary structure of T4 phage lysozyme. Biochim Biophys Acta (BBA) Protein Struct. 405(2):442–51. https://doi.org/10.1016/0005-2795(75)90109-9
- Hirzel, A. H., Le Lay, G., Helfer, V., Randin, C., & Guisan, A. (2006). Evaluating the ability of habitat suitability models to predict species presences. Ecological Modelling, 199(2), 142-152. https://doi.org/10.1016/j.ecolmodel.2006.05.017


## Examples

```
require(dplyr)

set.seed(0)
p <- rnorm(50, mean = 0.7, sd = 0.3) %>% abs()
p[p > 1] <- 1
p[p < 0] <- 0

set.seed(0)
a <- rnorm(50, mean = 0.3, sd = 0.2) %>% abs()
a[a > 1] <- 1
```

```
a[a < 0] <- 0

set.seed(0)
bg_vals <- rnorm(1000, mean = 0.4, sd = 0.4) %>% abs()
bg_vals[bg_vals > 1] <- 1
bg_vals[bg_vals < 0] <- 0

# Function use without threshold specification
e <- sdm_eval(p, a)
e

# Function use with threshold specification
sdm_eval(p, a, thr = "max_sorensen")
sdm_eval(p, a, thr = c("lpt", "max_sens_spec", "max_jaccard"))
sdm_eval(p, a, thr = c("lpt", "max_sens_spec", "sensitivity"))
sdm_eval(p, a, thr = c("lpt", "max_sens_spec", "sensitivity", sens = "0.95"))

# Use of bg argument (it will only be used for calculating BOYCE index)
sdm_eval(p, a, thr = "max_sens_spec")
sdm_eval(p, a, thr = c("max_sens_spec"), bg = bg_vals)

# If background will be used to calculate all other metrics
# background values can be used in "a" argument
sdm_eval(p, bg_vals, thr = "max_sens_spec")
```

---

`sdm_extract` *Extract environmental data values from a spatial raster based on x and y coordinates*

---

## Description

Extract environmental data values from a spatial raster based on x and y coordinates

## Usage

```
sdm_extract(data, x, y, env_layer, variables = NULL, filter_na = TRUE)
```

## Arguments

| | |
|---|---|
| `data` | data.frame. Database with species presence, presence-absence, or pseudo-absence records with x and y coordinates |
| `x` | character. Column name with spatial x coordinates |
| `y` | character. Column name with spatial y coordinates |
| `env_layer` | SpatRaster. Raster or raster stack with environmental variables. |
| `variables` | character. Vector with the variable names of predictor (environmental) variables Usage variables. = c("aet", "cwd", "tmin"). If no variable is specified, function will return data for all layers. Default NULL |
| `filter_na` | logical. If filter_na = TRUE (default), the rows with NA values for any of the environmental variables are removed from the returned tibble. |

## Value

A tibble that returns the original data base with additional columns for the extracted environmental variables at each xy location from the SpatRaster object used in 'env_layer'

## Examples

```
require(terra)

# Load datasets
data(spp)
f <- system.file("external/somevar.tif", package = "flexsdm")
somevar <- terra::rast(f)

# Extract environmental data from somevar for all locations in spp
ex_spp <-
  sdm_extract(
    data = spp,
    x = "x",
    y = "y",
    env_layer = somevar,
    variables = NULL,
    filter_na = FALSE
  )

# Extract environmental for two variables and remove rows with NAs
ex_spp2 <-
  sdm_extract(
    data = spp,
    x = "x",
    y = "y",
    env_layer = somevar,
    variables = c("CFP_3", "CFP_4"),
    filter_na = TRUE
  )

ex_spp
ex_spp2
```

---

`sdm_predict` *Spatial predictions from individual and ensemble models*

---

## Description

This function allows the geographical prediction of one or more models constructed with the fit_ or tune_ function set, models fitted with esm_ function set (i.e., ensemble of small models approach), or models constructed with fit_ensemble function. It can return continuous or continuous and binary predictions for one or more thresholds

### Usage

```
sdm_predict(
  models,
  pred,
  nchunk = 1,
  thr = NULL,
  con_thr = FALSE,
  predict_area = NULL,
  clamp = TRUE,
  pred_type = "cloglog"
)
```

### Arguments

`models` list of one or more models fitted with fit_ or tune_ functions. In case use models fitted with fit_ensemble or esm_ family function only one model could be used. Usage models = mglm or models = list(mglm, mraf, mgbm)

`pred` SpatRaster. Raster layer with predictor variables. Names of layers must exactly match those used in model fitting.

`nchunk` integer. Number of chunks to split data used to predict models (i.e., SpatRaster used in pred argument). Predicting models in chunks helps reduce memory requirements in cases where models are predicted for large scales and high resolution. Default = 1

`thr` character. Threshold used to get binary suitability values (i.e., 0,1). It is possible to use more than one threshold type. It is mandatory to use the same threshold/s used to fit the models. The following threshold types are available:

- lpt: The highest threshold at which there is no omission.
- equal_sens_spec: Threshold at which the sensitivity and specificity are equal.
- max_sens_spec: Threshold at which the sum of the sensitivity and specificity is the highest (aka threshold that maximizes the TSS).
- max_jaccard: The threshold at which the Jaccard index is the highest.
- max_sorensen: The threshold at which the Sorensen index is highest.
- max_fpb: The threshold at which FPB is highest.
- sensitivity: Threshold based on a specified sensitivity value used to fit the models.
- all: All the threshold used in the model outputs used in 'models' argument will be used.

Usage thr = c('lpt', 'max_sens_spec', 'max_jaccard'), thr=c('lpt', 'max_sens_spec', 'sensitivity'), or thr='all'. If no threshold is specified (i.e., thr = NULL) function will return continuous prediction only. Default NULL

`con_thr` logical. If true predictions with suitability values above threshold/s will be returned. Default = FALSE

`predict_area` SpatVector, SpatialPolygon, or SpatialPolygonDataFrame. Spatial polygon used for restring prediction into only a given region. Default = NULL

| | |
|---|---|
| clamp | logical. It is set with TRUE, predictors and features are restricted to the range seen during model training. Only valid for Maxent model (see tune_mx and fit_mx). Default TRUE. |
| pred_type | character. Type of response required available "link", "exponential", "cloglog" and "logistic". Only valid for Maxent model (see tune_mx and fit_mx). Default "cloglog". |

## Value

A list of SpatRaster with continuous and/or binary predictions

## See Also

sdm_uncertainty

## Examples

```
# Toy data: small raster and records of one species to keep this example fast
data("spp")
somevar <- terra::rast(system.file("external/somevar.tif", package = "flexsdm"))
somevar <- terra::aggregate(somevar, fact = 6) # coarser resolution
sp3 <- spp[spp$species == "sp3", ]
sp3 <- sdm_extract(data = sp3, x = "x", y = "y", env_layer = somevar)
sp3 <- part_random(
  data = sp3,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

glm_toy <- fit_glm(
  data = sp3,
  response = "pr_ab",
  predictors = c("CFP_1", "CFP_2", "CFP_3", "CFP_4"),
  partition = ".part",
  thr = "max_sens_spec"
)

pred <- sdm_predict(
  models = glm_toy,
  pred = somevar,
  thr = "max_sens_spec",
  con_thr = FALSE,
  predict_area = NULL
)
pred


require(dplyr)
require(terra)

data("spp")
somevar <- system.file("external/somevar.tif", package = "flexsdm")
```

```
somevar <- terra::rast(somevar)

# Extract data
some_sp <- spp %>%
  filter(species == "sp3")

some_sp <-
  sdm_extract(
    data = some_sp,
    x = "x",
    y = "y",
    env_layer = somevar
  )

# Partition
some_sp <- part_random(
  data = some_sp,
  pr_ab = "pr_ab",
  method = c(method = "rep_kfold", folds = 3, replicates = 5)
)


## %#######################################################%##
#                                                          #
####          Create different type of models           ####
#                                                          #
## %#######################################################%##
# Fit some models
mglm <- fit_glm(
  data = some_sp,
  response = "pr_ab",
  predictors = c("CFP_1", "CFP_2", "CFP_3", "CFP_4"),
  partition = ".part",
  poly = 2
)
mraf <- fit_raf(
  data = some_sp,
  response = "pr_ab",
  predictors = c("CFP_1", "CFP_2", "CFP_3", "CFP_4"),
  partition = ".part",
)
mgbm <- fit_gbm(
  data = some_sp,
  response = "pr_ab",
  predictors = c("CFP_1", "CFP_2", "CFP_3", "CFP_4"),
  partition = ".part"
)

# Fit an ensemble model
mensemble <- fit_ensemble(
  models = list(mglm, mraf, mgbm),
  ens_method = "meansup",
  thr = NULL,
```

```
  thr_model = "max_sens_spec",
  metric = "TSS"
)

# Fit a model with the Ensembles of Small Models approach
# Without threshold specification and with kfold
msmall <- esm_gam(
  data = some_sp,
  response = "pr_ab",
  predictors = c("CFP_1", "CFP_2", "CFP_3", "CFP_4"),
  partition = ".part",
  thr = NULL
)


## %#######################################################%##
#                                                           #
####      Predict different kind of models               ####
#                                                           #
## %#######################################################%##

# sdm_predict can be used for predict one or more models fitted with fit_ or tune_ functions

# a single model
ind_p <- sdm_predict(
  models = mglm,
  pred = somevar,
  thr = "max_fpb",
  con_thr = FALSE,
  predict_area = NULL
)

# a list of models
list_p <- sdm_predict(
  models = list(mglm, mraf, mgbm),
  pred = somevar,
  thr = "max_fpb",
  con_thr = FALSE,
  predict_area = NULL
)

# Predict an ensemble model
# (only is possilbe use one fit_ensemble)
ensemble_p <- sdm_predict(
  models = mensemble,
  pred = somevar,
  thr = "max_fpb",
  con_thr = FALSE,
  predict_area = NULL
)

# Predict an ensemble of small models
# (only is possible to use one ensemble of small models)
```

```
small_p <- sdm_predict(
  models = msmall,
  pred = somevar,
  thr = "max_fpb",
  con_thr = FALSE,
  predict_area = NULL
)

## %#######################################################%##
#                                                          #
####              Predict model using chunks            ####
#                                                          #
## %#######################################################%##
# Predicting models in chunks helps reduce memory requirements in
# cases where models are predicted for large scales and high resolution

ind_p <- sdm_predict(
  models = mglm,
  pred = somevar,
  thr = "max_fpb",
  con_thr = FALSE,
  predict_area = NULL,
  nchunk = 4
)
```

---

`sdm_summarize` *Merge model performance tables*

---

## Description

Merge model performance tables

## Usage

```
sdm_summarize(models)
```

## Arguments

| | |
|---|---|
| `models` | list of one or more models fitted with fit_ or tune_ functions, or a fit_ensemble output, a esm_ family function output. A list a single or several models fitted with some of fit_ or tune_ functions or object returned by the `fit_ensemble` function. Usage models = list(mod1, mod2, mod3) |

## Value

Combined model performance table for all input models. Models fit with tune will include model performance for the best hyperparameters.

## Examples

```
# Toy data: a small subset of 'abies' to keep this example fast
data("abies")
abies_toy <- abies[c(1:30, 701:730), ]
abies_toy <- part_random(
  data = abies_toy,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

glm_toy <- fit_glm(
  data = abies_toy,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin"),
  partition = ".part",
  thr = "max_sens_spec"
)
gam_toy <- fit_gam(
  data = abies_toy,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin"),
  partition = ".part",
  thr = "max_sens_spec",
  k = 3
)

# Merge the performance tables of several models
sdm_summarize(models = list(glm_toy, gam_toy))


data(abies)
abies

# In this example we will partition the data using the k-fold method

abies2 <- part_random(
  data = abies,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

# Build a generalized additive model using fit_gam

gam_t1 <- fit_gam(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "ppt_jja", "pH", "awc", "depth"),
  predictors_f = c("landform"),
  partition = ".part",
  thr = c("max_sens_spec", "equal_sens_spec", "max_sorensen")
)
```

```
gam_t1$performance

# Build a generalized linear model using fit_glm

glm_t1 <- fit_glm(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "ppt_jja", "pH", "awc", "depth"),
  predictors_f = c("landform"),
  partition = ".part",
  thr = c("max_sens_spec", "equal_sens_spec", "max_sorensen"),
  poly = 0,
  inter_order = 0
)

glm_t1$performance

# Build a tuned random forest model using tune_raf

tune_grid <-
  expand.grid(
    mtry = c(2, 4),
    ntree = c(100, 300)
  )

rf_t1 <-
  tune_raf(
    data = abies2,
    response = "pr_ab",
    predictors = c(
      "aet", "cwd", "tmin", "ppt_djf",
      "ppt_jja", "pH", "awc", "depth"
    ),
    predictors_f = c("landform"),
    partition = ".part",
    grid = tune_grid,
    thr = c("max_sens_spec", "equal_sens_spec", "max_sorensen"),
    metric = "TSS",
  )

rf_t1$performance

# Merge sdm performance tables

merge_df <- sdm_summarize(models = list(gam_t1, glm_t1, rf_t1))

merge_df
```

---

`sdm_uncertainty` *Calculate species distribution model uncertainty*

---

## Description

This function calculates the uncertainty of a species distribution model by performing a bootstrap procedure. It refits the model multiple times on resampled data and then calculates the standard deviation of the predictions across all iterations.

## Usage

```
sdm_uncertainty(
  models,
  training_data,
  background = NULL,
  response,
  projection_data,
  iteration = 50,
  n_cores = 5,
  clamp = TRUE,
  pred_type = "cloglog"
)
```

## Arguments

| | |
|---|---|
| `models` | A flexsdm model object from ‘fit_*‘ and ‘tune_*‘ functions. |
| `training_data` | A data.frame or tibble with presence-absence/background data and predictors. |
| `background` | A data.frame or tibble with background data, used for Maxent and Gausian Process models. Default NULL. |
| `response` | character. Column name of the response variable. |
| `projection_data` | A SpatRaster object with the environmental layers for projection. |
| `iteration` | numeric. The number of bootstrap iterations. A large number of iterations will increase computation time, especially for large areas or high-resolution data. Default 50. |
| `n_cores` | numeric. The number of cores to use for parallel processing. Default 5. |
| `clamp` | logical. If TRUE, predictors and features are restricted to the range seen during model training. Only for Maxent models. Default TRUE. |
| `pred_type` | character. The type of response required. Available options are: "link", "exponential", "cloglog", and "logistic". Only for Maxent models. Default "cloglog". |

## Value

A SpatRaster object with a single layer representing the model uncertainty, calculated as the standard deviation of the bootstrap predictions.

## See Also

`sdm_predict`

## Examples

```
# Toy data: small raster and records of one species to keep this example fast
data("spp")
somevar <- terra::rast(system.file("external/somevar.tif", package = "flexsdm"))
somevar <- terra::aggregate(somevar, fact = 6) # coarser resolution
sp3 <- spp[spp$species == "sp3", ]
sp3 <- sdm_extract(data = sp3, x = "x", y = "y", env_layer = somevar)
sp3 <- part_random(
  data = sp3,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

glm_toy <- fit_glm(
  data = sp3,
  response = "pr_ab",
  predictors = c("CFP_1", "CFP_2", "CFP_3", "CFP_4"),
  partition = ".part",
  thr = "max_sens_spec"
)

# Few bootstrap iterations to keep this example fast
unc <- sdm_uncertainty(
  models = glm_toy,
  training_data = sp3,
  response = "pr_ab",
  projection_data = somevar,
  iteration = 2,
  n_cores = 1
)
unc


require(terra)
require(dplyr)

data(spp)
somevar <- system.file("external/somevar.tif", package = "flexsdm")
somevar <- terra::rast(somevar)

sp_data <- spp %>% filter(species == "sp3")
sp_data <- part_random(
  data = sp_data,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

sp_data <- sdm_extract(
  data = sp_data,
  x = "x",
  y = "y",
  env_layer = somevar
```

```
)

m <- fit_svm(
  data = sp_data,
  response = "pr_ab",
  predictors = c("CFP_1", "CFP_2", "CFP_3", "CFP_4"),
  partition = ".part"
)

unc <- sdm_uncertainty(
  models = m,
  training_data = sp_data,
  response = "pr_ab",
  projection_data = somevar,
  iteration = 10,
  n_cores = 2
)

plot(unc)
```

---

`sdm_varimp` *Calculate permutation-based variable importance scores for SDMs*

---

## Description

This function calculates variable importance scores for species distribution models (SDMs) based on permutation-based approach.

## Usage

```
sdm_varimp(
  models,
  data,
  response,
  predictors,
  n_sim = 50,
  n_cores = 1,
  thr = NULL,
  clamp = TRUE,
  pred_type = "cloglog"
)
```

## Arguments

| | |
|---|---|
| `models` | list of one or more models fitted with fit_ or tune_ functions. In case use models fitted with fit_ensemble or esm_ family function only one model could be used. Usage models = mglm or models = list(mglm, mraf, mgbm) |
| `data` | data.frame. Database with response (0,1) and predictors values. |

`response` character. Column name with species absence-presence data (0,1).

`predictors` character. Vector with the column names of predictor variables. Usage predictors = c("aet", "cwd", "tmin")

`n_sim` integer. The number of Monte Carlo replications to perform. Default is 50. The results from each replication are averaged together (the standard deviation will also be returned).

`n_cores` numeric. Number of cores use for parallelization. Default 1

`thr` character. Threshold criterion used to get binary suitability values (i.e. 0,1). Used for threshold-dependent performance metrics. It is possible to use more than one threshold type. A vector must be provided for this argument. The following threshold criteria are available:

- lpt: The highest threshold at which there is no omission.
- equal_sens_spec: Threshold at which the Sensitivity and Specificity are equal.
- max_sens_spec: Threshold at which the sum of the Sensitivity and Specificity is the highest (aka threshold that maximizes the TSS).
- max_jaccard: The threshold at which the Jaccard index is the highest.
- max_sorensen: The threshold at which the Sorensen index is the highest.
- max_fpb: The threshold at which FPB (F-measure on presence-background data) is the highest.
- sensitivity: Threshold based on a specified Sensitivity value. Usage thr = c('sensitivity', sens='0.6') or thr = c('sensitivity'). 'sens' refers to Sensitivity value. If a sensitivity value is not specified, the default value is 0.9

#' If more than one threshold type is used, concatenate threshold types, e.g., thr=c('lpt', 'max_sens_spec', 'max_jaccard'), or thr=c('lpt', 'max_sens_spec', 'sensitivity', sens='0.8'), or thr=c('lpt', 'max_sens_spec', 'sensitivity'). Function will use all thresholds if no threshold type is specified

`clamp` logical. If TRUE, predictors and features are restricted to the range seen during model training.

`pred_type` character. Type of response required available "link", "exponential", "cloglog" and "logistic". Default "cloglog"

## Details

This function calculates variable importance scores for species distribution models (SDMs) based on permutation-based approach. Thus, the function calculates the model performance using the original data and the permuted data. The difference between the two performances is the variable importance score.

## Value

a tibble with the columns:

- model: model name
- threshold: threshold names
- predictors: predictor names
- from TPR to IMAE: performance metrics

## Examples

```
# Toy data: a small subset of 'abies' to keep this example fast
data("abies")
abies_toy <- abies[c(1:30, 701:730), ]
abies_toy <- part_random(
  data = abies_toy,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

glm_toy <- fit_glm(
  data = abies_toy,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin"),
  partition = ".part",
  thr = "max_sens_spec"
)

# Permutation-based variable importance (few simulations to keep it fast)
sdm_varimp(
  models = glm_toy,
  data = abies_toy,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin"),
  n_sim = 2,
  n_cores = 1,
  thr = "max_sens_spec"
)


require(tidyr)
require(dplyr)
require(ggplot2)

data(abies)
abies

data(backg)
backg
set.seed(1)
backg <- backg[sample(nrow(backg), 1000), ] # subsample to speed up this example

# In this example we will partition the data using the k-fold method

abies2 <- part_random(
  data = abies,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

backg2 <- part_random(
  data = backg,
```

```
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

max_t1 <- fit_max(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "ppt_jja", "pH", "awc", "depth"),
  predictors_f = c("landform"),
  partition = ".part",
  background = backg2,
  thr = c("max_sens_spec", "equal_sens_spec", "max_sorensen"),
  clamp = TRUE,
  classes = "default",
  pred_type = "cloglog",
  regmult = 1
)

net_t1 <- fit_net(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "ppt_jja", "pH", "awc", "depth"),
  predictors_f = c("landform"),
  partition = ".part",
  thr = c("max_sens_spec", "equal_sens_spec", "max_sorensen")
)

svm_f1 <- fit_svm(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "ppt_jja", "pH", "awc", "depth"),
  predictors_f = c("landform"),
  partition = ".part",
  thr = c("max_sens_spec", "equal_sens_spec", "max_sorensen")
)

vip_t <- sdm_varimp(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "ppt_jja", "pH", "awc", "depth", "landform"),
  models = list(max_t1, net_t1, svm_f1),
  clamp = TRUE,
  pred_type = "cloglog",
  thr = c("max_sens_spec", "equal_sens_spec", "max_sorensen"),
  n_sim = 5,
  n_cores = 2
)

vip_t

# Plot the variable importance for AUC TSS and SORENSEN for the
# threshold that maximizes Sorensen metric and Maxent
vip_t %>%
```

```
pivot_longer(
  cols = TPR:IMAE,
  names_to = "metric",
  values_to = "value"
) %>%
dplyr::filter(threshold == "max_sorensen") %>%
dplyr::filter(metric %in% c("AUC", "TSS", "SORENSEN")) %>%
dplyr::filter(model == "max") %>%
ggplot(aes(x = reorder(predictors, value), y = value, fill = predictors)) +
geom_col(
  col = "black",
  show.legend = FALSE
) +
facet_wrap(~metric, scales = "free_x") +
labs(x = "Predictors", y = "Variable Importance") +
theme_classic() +
coord_flip()
```

---

spp *A data set containing presences and absences of three virtual species*

---

## Description

A data set containing presences and absences of three virtual species

## Usage

```
spp
```

## Format

A tibble with 1150 rows and 3 variables:

**species** virtual species names

**x** longitude of species occurrences

**y** latitude of species occurrences

**pr_ab** presences and absences denoted by 1 and 0 respectively ...

## Examples

```
require(dplyr)
data("spp")
spp
```

tune_gbm *Fit and validate Generalized Boosted Regression models with exploration of hyper-parameters that optimize performance*

## Description

This function fits and validates Generalized Boosted Regression models (GBM) while exploring different hyper-parameter combinations to find the one that optimizes model performance. It is a tune version of [fit_gbm()].

## Usage

```
tune_gbm(
  data,
  response,
  predictors,
  predictors_f = NULL,
  fit_formula = NULL,
  partition,
  grid = NULL,
  thr = NULL,
  metric = "TSS",
  n_cores = 1
)
```

## Arguments

| | |
|---|---|
| data | data.frame. Database with response (0,1) and predictors values. |
| response | character. Column name with species absence-presence data (0,1). |
| predictors | character. Vector with the column names of quantitative predictor variables (i.e. continuous variables). Usage predictors = c("aet", "cwd", "tmin") |
| predictors_f | character. Vector with the column names of qualitative predictor variables (i.e. ordinal or nominal variables type). Usage predictors_f = c("landform") |
| fit_formula | formula. A formula object with response and predictor variables (e.g. formula(pr_ab ~ aet + ppt_jja + pH + awc + depth + landform)). Note that the variables used here must be consistent with those used in response, predictors, and predictors_f arguments. Default is NULL. |
| partition | character. Column name with training and validation partition groups. |
| grid | data.frame. A data frame object with algorithm hyper-parameter values to be tested. It Is recommended to generate this data.frame with the grid() function. Hyper-parameters needed for tuning are 'n.trees', 'shrinkage', and 'n.minobsinnode'. |
| thr | character. Threshold used to get binary suitability values (i.e. 0,1) needed for threshold-dependent performance metrics. It is possible to use more than one threshold type. Provide a vector for this argument. The following threshold types are available: |

- lpt: The highest threshold at which there is no omission.
- equal_sens_spec: Threshold at which the sensitivity and specificity are equal.
- max_sens_spec: Threshold at which the sum of the sensitivity and specificity is the highest (aka threshold that maximizes the TSS).
- max_jaccard: The threshold at which the Jaccard index is the highest.
- max_sorensen: The threshold at which the Sorensen index is highest.
- max_fpb: The threshold at which FPB (F-measure on presence-background data) is highest.
- sensitivity: Threshold based on a specified sensitivity value. Usage thr = c('sensitivity', sens='0.6') or thr = c('sensitivity'). 'sens' refers to sensitivity value. If no sensitivity value is specified, the default used is 0.9

If more than one threshold type is used they must be concatenate, e.g., thr=c('lpt', 'max_sens_spec', 'max_jaccard'), or thr=c('lpt', 'max_sens_spec', 'sensitivity', sens='0.8'), or thr=c('lpt', 'max_sens_spec', 'sensitivity'). Function will use all threshold types if no threshold is specified.

`metric` character. Performance metric used for selecting the best combination of hyper-parameter values. The following metrics can be used: SORENSEN, JACCARD, FPB, TSS, KAPPA, AUC, and BOYCE. TSS is used as the default.

`n_cores` numeric. Number of cores use for parallelization. Default 1

## Value

A list object with:

- model: A "gbm" class object from gbm package. This object can be used for predicting.
- predictors: A tibble with quantitative (c column names) and qualitative (f column names) variables use for modeling.
- performance: Hyper-parameter values and performance metric (see `sdm_eval`) for the best hyper-parameter combination.
- performance_part: Performance metric for each replica and partition (see `sdm_eval`).
- hyper_performance: Performance metric (see `sdm_eval`) for each combination of the hyper-parameters.
- data_ens: Predicted suitability for each test partition based on the best model. This database is used in `fit_ensemble`

## See Also

[tune_max], [tune_net], [tune_raf], [tune_svm].

## Examples

```
# Toy data: a small subset of 'abies' to keep this example fast
data("abies")
abies_toy <- abies[c(1:30, 701:730), ]
abies_toy <- part_random(
```

```
  data = abies_toy,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

tune_toy <- tune_gbm(
  data = abies_toy,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin"),
  partition = ".part",
  grid = expand.grid(
    n.trees = c(20, 40),
    shrinkage = 0.1,
    n.minobsinnode = 3
  ),
  thr = "max_sens_spec",
  metric = "TSS",
  n_cores = 1
)
tune_toy$performance


data(abies)
abies

# Partition the data with the k-fold method

abies2 <- part_random(
  data = abies,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

# pr_ab is the name of the column with species presence and absences (i.e. the response variable)
# from aet to landform are the predictors variables (landform is a qualitative variable)

# Hyper-parameter values for tuning
tune_grid <-
  expand.grid(
    n.trees = c(20, 50),
    shrinkage = c(0.1, 0.5),
    n.minobsinnode = c(3, 7)
  )

gbm_t <-
  tune_gbm(
    data = abies2,
    response = "pr_ab",
    predictors = c(
      "aet", "cwd", "tmin", "ppt_djf", "ppt_jja",
      "ppt_jja", "pH", "awc", "depth"
    ),
    predictors_f = c("landform"),
```

```
    partition = ".part",
    grid = tune_grid,
    thr = "max_sens_spec",
    metric = "TSS",
    n_cores = 1
  )

# Outputs
gbm_t$model
gbm_t$predictors
gbm_t$performance
gbm_t$performance_part
gbm_t$hyper_performance
gbm_t$data_ens

# Graphical exploration of performance of each hyper-parameter setting
require(ggplot2)
pg <- position_dodge(width = 0.5)
ggplot(gbm_t$hyper_performance, aes(factor(n.minobsinnode),
  TSS_mean,
  col = factor(shrinkage)
)) +
  geom_errorbar(aes(ymin = TSS_mean - TSS_sd, ymax = TSS_mean + TSS_sd),
    width = 0.2, position = pg
  ) +
  geom_point(position = pg) +
  geom_line(
    data = gbm_t$tune_performance,
    aes(as.numeric(factor(n.minobsinnode)),
      TSS_mean,
      col = factor(shrinkage)
    ), position = pg
  ) +
  facet_wrap(. ~ n.trees) +
  theme(legend.position = "bottom")
```

---

tune_max *Fit and validate Maximum Entropy models with exploration of hyper-parameters that optimize performance*

---

## Description

Fit and validate Maximum Entropy models with exploration of hyper-parameters that optimize performance

## Usage

```
tune_max(
```

```
  data,
  response,
  predictors,
  predictors_f = NULL,
  background = NULL,
  partition,
  grid = NULL,
  thr = NULL,
  metric = "TSS",
  clamp = TRUE,
  pred_type = "cloglog",
  n_cores = 1
)
```

## Arguments

`data` data.frame. Database with response (0,1) and predictors values.

`response` character. Column name with species absence-presence data (0,1).

`predictors` character. Vector with the column names of quantitative predictor variables (i.e. continuous variables). Usage predictors = c("aet", "cwd", "tmin")

`predictors_f` character. Vector with the column names of qualitative predictor variables (i.e. ordinal or nominal variables type). Usage predictors_f = c("landform")

`background` data.frame. Database with response variable column only containing 0 values, and predictors variables. All column names must be consistent with data

`partition` character. Column name with training and validation partition groups.

`grid` data.frame. A data frame object with algorithm hyper-parameters values to be tested. It is recommended to generate this data.frame with the grid() function. Hyper-parameters needed for tuning are 'regmult' and 'classes' (any combination of following letters l -linear-, q -quadratic-, h -hinge-, p -product-, and t -threshold-).

`thr` character. Threshold used to get binary suitability values (i.e. 0,1)., needed for threshold-dependent performance metrics. More than one threshold type can be used. It is necessary to provide a vector for this argument. The following threshold types are available:

- lpt: The highest threshold at which there is no omission.
- equal_sens_spec: Threshold at which sensitivity and specificity are equal.
- max_sens_spec: Threshold at which the sum of the sensitivity and specificity is the highest (aka threshold that maximizes the TSS).
- max_jaccard: The threshold at which the Jaccard index is the highest.
- max_sorensen: The threshold at which the Sorensen index is highest.
- max_fpb: The threshold at which # FPB (F-measure on presence-background data) is highest.
- sensitivity: Threshold based on a specified sensitivity value. Usage thr = c('sensitivity', sens='0.6') or thr = c('sensitivity'). 'sens' refers to sensitivity value. If a sensitivity value is not specified, a default of 0.9 will be used.

If more than one threshold type is used, concatenate them, e.g., thr=c('lpt', 'max_sens_spec', 'max_jaccard'), or thr=c('lpt', 'max_sens_spec', 'sensitivity', sens='0.8'), or thr=c('lpt', 'max_sens_spec', 'sensitivity'). Function will use all thresholds if no threshold is specified.

`metric` character. Performance metric used for selecting the best combination of hyper -parameter values. One of the following metrics can be used: SORENSEN, JACCARD, FPB, TSS, KAPPA, AUC, and BOYCE. TSS is used as default.

`clamp` logical. If TRUE, predictors and features are restricted to the range seen during model training.

`pred_type` character. Type of response required available "link", "exponential", "cloglog" and "logistic". Default "cloglog"

`n_cores` numeric. Number of cores use for parallelization. Default 1

## Details

When presence-absence (or presence-pseudo-absence) data are used in data argument in addition to background points, the function will fit models with presences and background points and validate with presences and absences. This procedure makes maxent comparable to other presences-absences models (e.g., random forest, support vector machine). If only presences and background points data are used, function will fit and validate model with presences and background data. If only presence-absences are used in data argument and without background, function will fit model with the specified data (not recommended).

## Value

A list object with:

- model: A "maxnet" class object from maxnet package. This object can be used for predicting.
- predictors: A tibble with quantitative (c column names) and qualitative (f column names) variables use for modeling.
- performance: Hyper-parameters values and performance metrics (see `sdm_eval`) for the best hyper-parameters combination.
- performance_part: Performance metric for each replica and partition (see `sdm_eval`).
- hyper_performance: Performance metrics (see `sdm_eval`) for each combination of the hyper-parameters.
- data_ens: Predicted suitability for each test partition based on the best model. This database is used in `fit_ensemble`

## See Also

`tune_gbm`, `tune_net`, `tune_raf`, and `tune_svm`.

## Examples

```
# Toy data: a small subset of 'abies' to keep this example fast
data("abies")
abies_toy <- abies[c(1:30, 701:730), ]
```

```
abies_toy <- part_random(
  data = abies_toy,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

# Use the same records as background points (toy example only)
backg_toy <- abies_toy
backg_toy$pr_ab <- 0

tune_toy <- tune_max(
  data = abies_toy,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin"),
  background = backg_toy,
  partition = ".part",
  grid = expand.grid(regmult = c(0.5, 1), classes = c("l", "lq")),
  thr = "max_sens_spec",
  metric = "TSS",
  clamp = TRUE,
  pred_type = "cloglog",
  n_cores = 1
)
tune_toy$performance


data("abies")
data("backg")
abies # environmental conditions of presence-absence data
backg # environmental conditions of background points

# Using k-fold partition method
# Remember that the partition method, number of folds or replications must
# be the same for presence-absence and background points datasets
abies2 <- part_random(
  data = abies,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)
abies2

set.seed(1)
backg <- dplyr::sample_n(backg, size = 500, replace = FALSE)
backg2 <- part_random(
  data = backg,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)
backg


gridtest <-
  expand.grid(
```

```
    regmult = c(0.1, 1),
    classes = c("l", "lq")
  )

max_t1 <- tune_max(
  data = abies2,
  response = "pr_ab",
  predictors = c("aet", "pH", "awc", "depth"),
  predictors_f = c("landform"),
  partition = ".part",
  background = backg2,
  grid = gridtest,
  thr = "max_sens_spec",
  metric = "TSS",
  clamp = TRUE,
  pred_type = "cloglog",
  n_cores = 2 # activate two cores to speed up this process
)

length(max_t1)
max_t1$model
max_t1$predictors
max_t1$performance
max_t1$performance_part
max_t1$data_ens
```

---

tune_net *Fit and validate Neural Networks models with exploration of hyper-parameters*

---

## Description

Fit and validate Neural Networks models with exploration of hyper-parameters

## Usage

```
tune_net(
  data,
  response,
  predictors,
  predictors_f = NULL,
  fit_formula = NULL,
  partition,
  grid = NULL,
  thr = NULL,
  metric = "TSS",
  n_cores = 1
)
```

### Arguments

`data` data.frame. Database with response (0,1) and predictors values.

`response` character. Column name with species absence-presence data (0,1).

`predictors` character. Vector with the column names of quantitative predictor variables (i.e. continuous variables). Usage predictors = c("aet", "cwd", "tmin")

`predictors_f` character. Vector with the column names of qualitative predictor variables (i.e. ordinal or nominal variables type). Usage predictors_f = c("landform")

`fit_formula` formula. A formula object with response and predictor variables (e.g. formula(pr_ab ~ aet + ppt_jja + pH + awc + depth + landform)). Note that the variable names used here must be consistent with those used in response, predictors, and predictors_f arguments. Defaul NULL.

`partition` character. Column name with training and validation partition groups.

`grid` data.frame. A data frame object with algorithm hyper-parameters values to be tested. It is recommended to generate this data.frame with the grid() function.

`thr` character. Threshold used to get binary suitability values (i.e. 0,1), needed for threshold-dependent performance metrics. It is possible to use more than one threshold type. It is necessary to provide a vector for this argument. The following threshold types are available:

- lpt: The highest threshold at which there is no omission.
- equal_sens_spec: Threshold at which the sensitivity and specificity are equal.
- max_sens_spec: Threshold at which the sum of the sensitivity and specificity is the highest (aka threshold that maximizes the TSS).
- max_jaccard: The threshold at which the Jaccard index is the highest.
- max_sorensen: The threshold at which the Sorensen index is highest.
- max_fpb: The threshold at which FPB (F-measure on presence-background data) is highest.
- sensitivity: Threshold based on a specified sensitivity value. Usage thr = c('sensitivity', sens='0.6') or thr = c('sensitivity'). 'sens' refers to sensitivity value. If it is not specified a sensitivity values, function will use by default 0.9.

If using more than one threshold type concatenate them, e.g., thr=c('lpt', 'max_sens_spec', 'max_jaccard'), or thr=c('lpt', 'max_sens_spec', 'sensitivity', sens='0.8'), or thr=c('lpt', 'max_sens_spec', 'sensitivity'). Function will use all thresholds if no threshold is specified.

`metric` character. Performance metric used for selecting the best combination of hyper-parameter values. One of the following metrics can be used: SORENSEN, JACCARD, FPB, TSS, KAPPA, AUC, and BOYCE. TSS is used as default.

`n_cores` numeric. Number of cores use for parallelization. Default 1

### Value

A list object with:

- model: A "nnet" class object from nnet package. This object can be used for predicting.
- predictors: A tibble with quantitative (c column names) and qualitative (f column names) variables use for modeling.
- performance: Hyper-parameters values and performance metric (see sdm_eval) for the best hyper-parameters combination.
- performance_part: Performance metric for each replica and partition (see sdm_eval).
- hyper_performance: Performance metric (see sdm_eval) for each combination of the hyper-parameters.
- data_ens: Predicted suitability for each test partition based on the best model. This database is used in fit_ensemble

## See Also

tune_gbm, tune_max, tune_raf, and tune_svm.

## Examples

```
# Toy data: a small subset of 'abies' to keep this example fast
data("abies")
abies_toy <- abies[c(1:30, 701:730), ]
abies_toy <- part_random(
  data = abies_toy,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

tune_toy <- tune_net(
  data = abies_toy,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin"),
  partition = ".part",
  grid = expand.grid(size = c(2, 3), decay = c(0.1, 1)),
  thr = "max_sens_spec",
  metric = "TSS",
  n_cores = 1
)
tune_toy$performance


data(abies)
abies

# Partitioning the data with the k-fold method

abies2 <- part_random(
  data = abies,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)
```

```
# pr_ab columns is species presence and absences (i.e. the response variable)
# from aet to landform are the predictors variables (landform is a qualitative variable)

# Hyper-parameter values for tuning
tune_grid <-
  expand.grid(
    size = c(2, 4),
    decay = c(0.001, 1, 5)
  )

net_t <-
  tune_net(
    data = abies2,
    response = "pr_ab",
    predictors = c(
      "aet", "cwd", "tmin", "ppt_djf",
      "ppt_jja", "pH", "awc", "depth"
    ),
    predictors_f = c("landform"),
    partition = ".part",
    grid = tune_grid,
    thr = "max_sens_spec",
    metric = "TSS",
    n_cores = 1
  )

# Outputs
net_t$model
net_t$predictors
net_t$performance
net_t$performance_part
net_t$hyper_performance
net_t$data_ens
```

---

tune_raf *Fit and validate Random Forest models with exploration of hyper-parameters that optimize performance*

---

## Description

Fit and validate Random Forest models with exploration of hyper-parameters that optimize performance

## Usage

```
tune_raf(
  data,
  response,
```

```
  predictors,
  predictors_f = NULL,
  fit_formula = NULL,
  partition,
  grid = NULL,
  thr = NULL,
  metric = "TSS",
  n_cores = 1
)
```

## Arguments

`data` data.frame. Database with response (0,1) and predictors values.

`response` character. Column name with species absence-presence data (0,1).

`predictors` character. Vector with the column names of quantitative predictor variables (i.e. continuous variables). Usage predictors = c("aet", "cwd", "tmin")

`predictors_f` character. Vector with the column names of qualitative predictor variables (i.e. ordinal or nominal variables type). Usage predictors_f = c("landform")

`fit_formula` formula. A formula object with response and predictor variables (e.g. formula(pr_ab ~ aet + ppt_jja + pH + awc + depth + landform)). Note that the variables used here must be consistent with those used in response, predictors, and predictors_f arguments. Default NULL

`partition` character. Column name with training and validation partition groups.

`grid` data.frame. A data frame object with algorithm hyper-parameters values to be tested. It is recommended to generate this data.frame with the grid() function. Hyper-parameter needed for tuning is 'mtry' and 'ntree'. The maximum mtry cannot exceed the total number of predictors.

`thr` character. Threshold used to get binary suitability values (i.e. 0,1), needed for threshold-dependent performance metrics. It is possible to use more than one threshold type. It is necessary to provide a vector for this argument. The following threshold types are available:

- lpt: The highest threshold at which there is no omission.
- equal_sens_spec: Threshold at which the sensitivity and specificity are equal.
- max_sens_spec: Threshold at which the sum of the sensitivity and specificity is the highest (aka threshold that maximizes the TSS).
- max_jaccard: The threshold at which the Jaccard index is the highest.
- max_sorensen: The threshold at which the Sorensen index is highest.
- max_fpb: The threshold at which FPB is highest.
- sensitivity: Threshold based on a specified sensitivity value. Usage thr = c('sensitivity', sens='0.6') or thr = c('sensitivity'). 'sens' refers to sensitivity value. If it is not specified a sensitivity values, function will use by default 0.9

If using more than one threshold type concatenate them, e.g., thr=c('lpt', 'max_sens_spec', 'max_jaccard'), or thr=c('lpt', 'max_sens_spec', 'sensitivity', sens='0.8'), or

thr=c('lpt', 'max_sens_spec', 'sensitivity'). Function will use all thresholds if no threshold is specified.

`metric` character. Performance metric used for selecting the best combination of hyper -parameter values. One of the following metrics can be used: SORENSEN, JACCARD, FPB, TSS, KAPPA, AUC, and BOYCE. TSS is used as default.

`n_cores` numeric. Number of cores use for parallelization. Default 1

## Value

A list object with:

- model: A "randomForest" class object from randomForest package. This object can be used for predicting.
- predictors: A tibble with quantitative (c column names) and qualitative (f column names) variables use for modeling.
- performance: Hyper-parameters values and performance metric (see `sdm_eval`) for the best hyper-parameters combination.
- performance_part: Performance metric for each replica and partition (see `sdm_eval`).
- hyper_performance: Performance metric (see `sdm_eval`) for each combination of the hyper-parameters.
- data_ens: Predicted suitability for each test partition based on the best model. This database is used in `fit_ensemble`

## See Also

`tune_gbm`, `tune_max`, `tune_net`, and `tune_svm`.

## Examples

```
# Toy data: a small subset of 'abies' to keep this example fast
data("abies")
abies_toy <- abies[c(1:30, 701:730), ]
abies_toy <- part_random(
  data = abies_toy,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

tune_toy <- tune_raf(
  data = abies_toy,
  response = "pr_ab",
  predictors = c("aet", "cwd", "tmin"),
  partition = ".part",
  grid = expand.grid(mtry = c(1, 2), ntree = c(50, 100)),
  thr = "max_sens_spec",
  metric = "TSS",
  n_cores = 1
)
tune_toy$performance
```

```
data(abies)
abies

# Partition the data with the k-fold method

abies2 <- part_random(
  data = abies,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

tune_grid <- expand.grid(
  mtry = c(2, 4),
  ntree = c(100, 300)
)

tune_grid

rf_t <-
  tune_raf(
    data = abies2,
    response = "pr_ab",
    predictors = c(
      "aet", "cwd", "tmin", "ppt_djf",
      "ppt_jja", "pH", "awc", "depth"
    ),
    predictors_f = c("landform"),
    partition = ".part",
    grid = tune_grid,
    thr = "max_sens_spec",
    metric = "TSS",
    n_cores = 1
  )

# Outputs
rf_t$model
rf_t$predictors
rf_t$performance_part
rf_t$hyper_performance
rf_t$data_ens
```

---

`tune_svm` *Fit and validate Support Vector Machine models with exploration of hyper-parameters that optimize performance*

---

## Description

Fit and validate Support Vector Machine models with exploration of hyper-parameters that optimize performance

## Usage

```
tune_svm(
  data,
  response,
  predictors,
  predictors_f = NULL,
  fit_formula = NULL,
  partition,
  grid = NULL,
  thr = NULL,
  metric = "TSS",
  n_cores = 1
)
```

## Arguments

`data` data.frame. Database with response (0,1) and predictors values.

`response` character. Column name with species absence-presence data (0,1).

`predictors` character. Vector with the column names of quantitative predictor variables (i.e. continuous variables). Usage predictors = c("aet", "cwd", "tmin")

`predictors_f` character. Vector with the column names of qualitative predictor variables (i.e. ordinal or nominal variables type). Usage predictors_f = c("landform")

`fit_formula` formula. A formula object with response and predictor variables (e.g. formula(pr_ab ~ aet + ppt_jja + pH + awc + depth + landform)). Note that the variable names used here must be consistent with those used in response, predictors, and predictors_f arguments. Default NULL

`partition` character. Column name with training and validation partition groups.

`grid` data.frame. Provide a data frame object with algorithm hyper-parameters values to be tested. It Is recommended to generate this data.frame with grid() function. Hyper-parameters needed for tuning are 'size' and 'decay'.

`thr` character. Threshold used to get binary suitability values (i.e. 0,1). It is useful for threshold-dependent performance metrics. It is possible to use more than one threshold type. It is necessary to provide a vector for this argument. The next threshold area available:

- lpt: The highest threshold at which there is no omission.
- equal_sens_spec: Threshold at which the sensitivity and specificity are equal.
- max_sens_spec: Threshold at which the sum of the sensitivity and specificity is the highest (aka threshold that maximizes the TSS).
- max_jaccard: The threshold at which the Jaccard index is the highest.

- max_sorensen: The threshold at which the Sorensen index is highest.
- max_fpb: The threshold at which # FPB (F-measure on presence-background data) is highest.
- sensitivity: Threshold based on a specified sensitivity value. Usage thr = c('sensitivity', sens='0.6') or thr = c('sensitivity'). 'sens' refers to sensitivity value. If a sensitivity value is not specified, the default used is 0.9.

In the case of use more than one threshold type it is necessary concatenate threshold types, e.g., thr=c('lpt', 'max_sens_spec', 'max_jaccard'), or thr=c('lpt', 'max_sens_spec', 'sensitivity', sens='0.8'), or thr=c('lpt', 'max_sens_spec', 'sensitivity'). Function will use all thresholds if no threshold is specified

`metric` character. Performance metric used for selecting the best combination of hyper-parameter values. One of the following metrics can be used: SORENSEN, JACCARD, FPB, TSS, KAPPA, AUC, and BOYCE. TSS is used as default.

`n_cores` numeric. Number of cores use for parallelization. Default 1

## Value

A list object with:

- model: A "ksvm" class object from kernlab package. This object can be used for predicting.
- predictors: A tibble with quantitative (c column names) and qualitative (f column names) variables use for modeling.
- performance: Hyper-parameters values and performance metric (see `sdm_eval`) for the best hyper-parameters combination.
- performance_part: Performance metric for each replica and partition (see `sdm_eval`).
- hyper_performance: Performance metrics (see `sdm_eval`) for each combination of the hyper-parameters.
- data_ens: Predicted suitability for each test partition based on the best model. This database is used in `fit_ensemble`

## See Also

`tune_gbm`, `tune_max`, `tune_net`, and `tune_raf`.

## Examples

```
# Toy data: a small subset of 'abies' to keep this example fast
data("abies")
abies_toy <- abies[c(1:30, 701:730), ]
abies_toy <- part_random(
  data = abies_toy,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

tune_toy <- tune_svm(
  data = abies_toy,
  response = "pr_ab",
```

```
  predictors = c("aet", "cwd", "tmin"),
  partition = ".part",
  grid = expand.grid(C = c(2, 8), sigma = c(0.01, 0.1)),
  thr = "max_sens_spec",
  metric = "TSS",
  n_cores = 1
)
tune_toy$performance


data(abies)
abies

# Partition the data with the k-fold method

abies2 <- part_random(
  data = abies,
  pr_ab = "pr_ab",
  method = c(method = "kfold", folds = 3)
)

# pr_ab column is species presence and absences (i.e. the response variable)
# from aet to landform are the predictors variables (landform is a qualitative variable)

# Hyper-parameter values for tuning
tune_grid <-
  expand.grid(
    C = c(2, 8),
    sigma = c(0.01, 0.1, 0.3)
  )

svm_t <-
  tune_svm(
    data = abies2,
    response = "pr_ab",
    predictors = c(
      "aet", "cwd", "tmin", "ppt_djf",
      "ppt_jja", "pH", "awc", "depth"
    ),
    predictors_f = c("landform"),
    partition = ".part",
    grid = tune_grid,
    thr = "max_sens_spec",
    metric = "TSS",
    n_cores = 1
  )

# Outputs
svm_t$model
svm_t$predictors
svm_t$performance
svm_t$performance_part
svm_t$hyper_performance
```

```
svm_t$data_ens
```

# Index


∗ **datasets**
abies, 3
backg, 4
hespero, 87
palms, 130
spp, 180

abies, 3

backg, 4

calib_area, 5, *148*, *155*
correct_colinvar, 8

data_bpdp, 14, *18*, *117*, *128*
data_pdp, *15*, 17, *117*, *128*

env_outliers, 19
esm_gam, 23, *28*, *31*, *34*, *37*, *41*, *44*
esm_gau, *24*, 26, *31*, *34*, *37*, *41*, *44*
esm_gbm, *24*, *28*, 29, *34*, *37*, *41*, *44*
esm_glm, *24*, *28*, *31*, 32, *37*, *41*, *44*
esm_max, *24*, *28*, *31*, *34*, 35, *41*, *44*
esm_net, *24*, *28*, *31*, *34*, *37*, 39, *44*
esm_svm, *24*, *28*, *31*, *34*, *37*, *41*, 42
extra_eval, 45, *50*, *117*, *128*
extra_truncate, *46*, 49, *117*, *128*

fit_dom, 51
fit_ensemble, *52*, 55, *60*, *63*, *66*, *69*, *73*, *76*, *79*, *82*, *171*, *182*, *186*, *190*, *193*, *196*
fit_gam, *53*, 58, *63*, *66*, *69*, *73*, *76*, *79*, *82*
fit_gau, *53*, *60*, 62, *66*, *69*, *73*, *76*, *79*, *82*
fit_gbm, *53*, *60*, *63*, 65, *69*, *73*, *76*, *79*, *82*
fit_glm, *53*, *60*, *63*, *66*, 68, *73*, *76*, *79*, *82*
fit_max, *53*, *60*, *63*, *66*, *69*, 71, *76*, *79*, *82*
fit_net, *53*, *60*, *63*, *66*, *69*, *73*, 75, *79*, *82*
fit_raf, *53*, *60*, *63*, *66*, *69*, *73*, *76*, 78, *82*
fit_svm, *53*, *60*, *63*, *66*, *69*, *73*, *76*, *79*, 81

get_absences, 84
get_block, 86, *134*, *135*, *138*, *139*, *144*

hespero, 87
homogenize_na, 88

interp, 89

ksvm, *43*, *82*

map_env_dist, *52*, 91
msdm_posteriori, 94, *101*
msdm_priori, *97*, 100

occ_reproj, 103
occfilt_env, 105, *110*, *113*, *114*
occfilt_geo, *106*, 108, *113*, *114*
occfilt_select, *105*, *106*, *110*, 113

p_bpdp, *15*, *18*, *46*, *50*, 116, *128*
p_extra, *46*, *50*, 120
p_pdp, *15*, *18*, *46*, *50*, *117*, 126
palms, 130
part_random, 131, *134*, *135*, *138*, *139*, *144*, *145*
part_sband, 133, *139*, *145*
part_sblock, *132*, *135*, 137, *145*
part_senv, *132*, *135*, *139*, 143
plot_res, *139*, 146
project, *104*

sample_background, *132*, *134*, *138*, *144*, 147, *155*
sample_pseudoabs, *132*, *134*, *138*, *144*, *148*, 153
sdm_directory, 159
sdm_eval, *24*, *28*, *31*, *34*, *37*, *40*, *43*, *52*, *56*, *60*, *63*, *66*, *69*, *73*, *76*, *79*, *82*, 161, *182*, *186*, *190*, *193*, *196*
sdm_extract, 165
sdm_predict, *24*, *28*, *31*, *34*, *37*, *40*, *43*, 166, *174*

sdm_summarize, 171
sdm_uncertainty, *168*, 173
sdm_varimp, 176
spp, 180

tempdir, *9*
tune_gbm, 181, *186*, *190*, *193*, *196*
tune_max, 184, *190*, *193*, *196*
tune_net, *186*, 188, *193*, *196*
tune_raf, *186*, *190*, 191, *196*
tune_svm, *186*, *190*, *193*, 194

vect, *104*