Package: kuenm2 (via r-universe)

May 31, 2026

Type Package

Title Detailed Development of Ecological Niche Models

Version 0.1.3

Maintainer Weverton C. F. Trindade <wevertonf1993@gmail.com>

BugReports https://github.com/marlonecobos/kuenm2/issues

Date 2026-04-20

Description A new set of tools to help with the development of detailed ecological niche models using multiple algorithms. Pre-modeling analyses and explorations can be done to prepare data. Model calibration (model selection) can be done by creating and testing models with several parameter combinations. Handy options for producing final models with transfers are included. Other tools to assess extrapolation risks and variability in model transfers are also available. Methodological and theoretical basis for the methods implemented here can be found in: Peterson et al. (2011) <https://www.degruyter.com/princetonup/view/title/506966>, Radosavljevic and Anderson (2014) <doi:10.1111/jbi.12227>, Peterson et al. (2018) <doi:10.1111/nyas.13873>, Cobos et al. (2019) <doi:10.7717/peerj.6281>, Alkishe et al. (2020) <doi:10.1016/j.pecon.2020.03.002>, Machado-Stredel et al. (2021) <doi:10.21425/F5FBG48814>, Arias-Giraldo and Cobos (2024) <doi:10.17161/bi.v18i.21742>, Cobos et al. (2024) <doi:10.17161/bi.v18i.21742>.

Imports doSNOW, enmpa (>= 0.2.1), foreach (>= 1.5), fpROC (>= 0.1.0), glmnet (>= 4.1), grDevices, graphics, mgcv (>= 1.9), mop (>= 0.1.3), parallel, stats, terra (>= 1.6), utils

SystemRequirements GDAL (>= 2.2.3), GEOS (>= 3.4.0), PROJ (>= 4.9.3)

Depends R (>= 3.5)

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Roxygen list(markdown = TRUE)

Suggests knitr, rmarkdown, spelling

VignetteBuilder knitr

URL https://marlonecobos.github.io/kuenm2/

Language en-US

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

Repository https://marlonecobos.r-universe.dev

Date/Publication 2026-05-31 20:51:13 UTC

RemoteUrl https://github.com/marlonecobos/kuenm2

RemoteRef HEAD

RemoteSha 63fbf0cf7b02d30928fe31fc511323330ed7a250

Contents

---

kuenm2-package *kuenm2: Detailed Development of Ecological Niche Models*

Description

kuenm2 A new set of tools to help with the development of detailed ecological niche models using multiple algorithms, at the moment Maxnet and GLM. Pre-modeling analyses and explorations can be done to prepare data. Model calibration (model selection) can be done by creating and testing several candidate models, that are later selected based on a multicriteria approach. Handy options for producing final models with transfers are included. Other tools to assess extrapolation risks and variability in model transfers are also available.

Main functions by stage in the ENM process

Pre-modeling steps:

- Data preparation: `initial_cleaning()`, `advanced_cleaning()`, `prepare_data()`, `prepare_user_data()`
- Data exploration: `explore_calibration_hist()`, `explore_partition_env()`, `explore_partition_geo()`, `explore_partition_extrapolation()`, `plot_calibration_hist()`, `plot_explore_partition()`

Modeling process:

- Model calibration: `calibration()`, `select_models()`
- Model exploration: `fit_selected()`, `variable_importance()`, `plot_importance()`, `response_curve()`, `all_response_curves()`, `bivariate_response()`, `partition_response_curves()`
- Model projection: `predict_selected()`, `organize_for_projection()`, `organize_future_worldclim()`, `prepare_projection()`, `project_selected()`

Post-modeling analysis:

- Variability: `prediction_changes()`, `projection_changes()`, `projection_variability()`
- Uncertainty: `projection_mop()`, `single_mop()`

Author(s)

Maintainer: Weverton C. F. Trindade <wevertonf1993@gmail.com> (ORCID)

Authors:

- Luis F. Arias-Giraldo <lfarias.giraldo@gmail.com> (ORCID)
- Luis Osorio-Olvera <luismurao@gmail.com> (ORCID)
- A. Townsend Peterson <town@ku.edu> (ORCID)
- Marlon E. Cobos <manubio13@gmail.com> (ORCID)

See Also

Useful links:

- https://marlonecobos.github.io/kuenm2/
- Report bugs at https://github.com/marlonecobos/kuenm2/issues

---

`advanced_cleaning` *Advanced occurrence data cleaning*

Description

Advanced processes of data cleaning involving duplicate removal and movement of records.

Usage

```
advanced_cleaning(data, x, y, raster_layer, cell_duplicates = TRUE,
                  move_points_inside = FALSE, move_limit_distance = NULL,
                  verbose = TRUE)

remove_cell_duplicates(data, x, y,
                       raster_layer)

move_2closest_cell(data, x, y, raster_layer,
                   move_limit_distance, verbose = TRUE)
```

Arguments

`data`	data.frame with occurrence records. Rows with NA values will be omitted.
`x`	(character) name of the column in `data` containing longitude values.
`y`	(character) name of the column in `data` containing latitude values.
`raster_layer`	a raster layer (object of class `SpatRaster`).
`cell_duplicates`	(logical) whether to remove duplicate coordinates considering raster cells. Default = TRUE.
`move_points_inside`	(logical) whether to move records outside of raster cells with valid values to the closest cell with values. Default = FALSE.
`move_limit_distance`	maximum distance to move records outside cells with valid values. Default = NULL. Must be defined if `move_points_inside` = TRUE.
`verbose`	(logical) whether to print messages of progress. Default = TRUE.

Details

Data used in this functions should have gone through initial processes of cleaning and filtering.

Value

A data.frame with occurrence records resulting from advanced cleaning procedures. Other columns will be added to describe changes made in the original data.

See Also

initial_cleaning()

Examples

```
# Import occurrences
data(occ_data_noclean, package = "kuenm2")

# Import raster layers
var <- terra::rast(system.file("extdata", "Current_variables.tif",
                  package = "kuenm2"))

# Keep only one layer
var <- var$bio_1

# all basic cleaning steps
clean_init <- initial_cleaning(data = occ_data_noclean, species = "species",
                               x = "x", y = "y", remove_na = TRUE,
                               remove_empty = TRUE, remove_duplicates = TRUE,
                               by_decimal_precision = TRUE,
                               decimal_precision = 2)

# Advanced cleaning steps
# exclude duplicates based on raster cell (pixel)
celldup <- remove_cell_duplicates(data = clean_init, x = "x", y = "y",
                                  raster_layer = var)

# move records to valid pixels
moved <- move_2closest_cell(data = celldup, x = "x", y = "y",
                            raster_layer = var, move_limit_distance = 10)

# the steps at a time
clean_data <- advanced_cleaning(data = clean_init, x = "x", y = "y",
                                raster_layer = var, cell_duplicates = TRUE,
                                move_points_inside = TRUE,
                                move_limit_distance = 10)
```

---

bias *Example Bias File*

Description

A SpatRaster object representing a bias layer used for extracting background points with the prepare_data() function.

Format

A SpatRaster object.

Value

No return value. Used with function rast to bring raster variables to analysis.

Examples

```
bias <- terra::rast(system.file("extdata", "bias_file.tif",
                                package = "kuenm2"))

terra::plot(bias)
```

---

binarize_changes *Binarize changes based on the agreement among GCMs*

Description

Highlights areas where a specified number of Global Climate Models (GCMs) agree on a given outcome in the projected scenario. The function can identify areas classified as suitable, unsuitable, stable-suitable, stable-unsuitable, gained, or lost.

Usage

```
binarize_changes(changes_projections, outcome = "suitable", n_gcms)
```

Arguments

changes_projections	an object of class changes_projections, generated by projection_changes() or imported using import_projections(), containing the $Summary_changes element.
outcome	(character) The outcome to binarize. Available options are "suitable", "unsuitable", "stable-suitable", "stable-unsuitable", "gain" or "loss". Default is "suitable". See details.
n_gcms	(numeric) The minimum number of GCMs that must agree on the specified outcome for a cell to be included in that category.

Details

The interpretation of the outcomes depends on the temporal direction of the projection. **When projecting to future scenarios**:

- *suitable*: Areas that remain suitable (stable-suitable) or become suitable (gain) in the future.
- *unsuitable*: Areas that remain unsuitable (stable-unsuitable) or become unsuitable (loss) in the future.

- *gain*: Areas that are currently unsuitable become suitable in the future.
- *loss*: Areas that are currently suitable become unsuitable in the future.
- *stable-suitable or stable-unsuitable*: Areas that retain their current classification in the future, whether suitable or unsuitable.

When projecting to past scenarios:

- *suitable*: Areas that remain suitable (stable-suitable) or become unsuitable (loss) in the present.
- *unsuitable*: Areas that remain unsuitable (stable-unsuitable) or become suitable (gain) in the present
- *gain*: Areas that were unsuitable in the past are now suitable in the present.
- *loss*: Areas that were suitable in the past are now unsuitable in the present.
- *stable-suitable or stable-unsuitable*: Areas that retain their current classification in the future, whether suitable or unsuitable.

Value

A `SpatRaster` or a list of `SpatRaster` objects (one per scenario) with the binarized outcomes. For example, if `outcome = "suitable"` and `n_gcms = 3`, cells with a value of 1 indicate areas where three or more GCMs agree that the area is suitable for the species in that scenario.

Examples

```
# Step 1: Organize variables for current projection
## Import current variables (used to fit models)
var <- terra::rast(system.file("extdata", "Current_variables.tif",
                               package = "kuenm2"))

## Create a folder in a temporary directory to copy the variables
out_dir_current <- file.path(tempdir(), "Current_raw_bin")
dir.create(out_dir_current, recursive = TRUE)

## Save current variables in temporary directory
terra::writeRaster(var, file.path(out_dir_current, "Variables.tif"))


# Step 2: Organize future climate variables (example with WorldClim)
## Directory containing the downloaded future climate variables (example)
in_dir <- system.file("extdata", package = "kuenm2")

## Create a folder in a temporary directory to copy the future variables
out_dir_future <- file.path(tempdir(), "Future_raw_bin")

## Organize and rename the future climate data (structured by year and GCM)
### 'SoilType' will be appended as a static variable in each scenario
organize_future_worldclim(input_dir = in_dir, output_dir = out_dir_future,
                          name_format = "bio_", static_variables = var$SoilType)

# Step 3: Prepare data to run multiple projections
## An example with maxnet models
```

```
## Import example of fitted_models (output of fit_selected())
data(fitted_model_maxnet, package = "kuenm2")

## Prepare projection data using fitted models to check variables
pr <- prepare_projection(models = fitted_model_maxnet,
                         present_dir = out_dir_current,
                         future_dir = out_dir_future,
                         future_period = c("2081-2100"),
                         future_pscen = c("ssp585"),
                         future_gcm = c("ACCESS-CM2", "MIROC6"),
                         raster_pattern = ".tif*")

# Step 4: Run multiple model projections
## A folder to save projection results
out_dir <- file.path(tempdir(), "Projection_results/maxnet_bin")
dir.create(out_dir, recursive = TRUE)

## Project selected models to multiple scenarios
p <- project_selected(models = fitted_model_maxnet, projection_data = pr,
                      out_dir = out_dir)

# Step 5: Identify areas of change in projections
## Contraction, expansion and stability
changes <- projection_changes(model_projections = p, write_results = FALSE,
                              return_raster = TRUE)

# Step 6: Binarize changes
future_suitable <- binarize_changes(changes_projections = changes,
                                    outcome = "suitable",
                                    n_gcms = 1)
terra::plot(future_suitable)
```

`bivariate_response` *Bivariate response plot for fitted models*

Description

A plot of suitability prediction in a two-dimensional environmental space.

Usage

```
bivariate_response(models, variable1 , variable2, modelID = NULL, n = 500,
                   new_data = NULL, extrapolate = TRUE, add_bar = TRUE ,
                   add_limits = TRUE, color_palette = NULL,
                   xlab = NULL, ylab = NULL, ...)
```

Arguments

`models`	an object of class `fitted_models` returned by the `fit_selected()` function.

variable1	(character) name of the variable to be plotted in x axis.
variable2	(character) name of the variable to be plotted in y axis.
modelID	(character) name of the ModelID presents in the fitted object. Default = NULL.
n	(numeric) the number of breaks for plotting grid. Default = 500
new_data	a SpatRaster, data.frame, or matrix of variables representing an area of interest. Default = NULL.
extrapolate	(logical) whether to allow extrapolation to study the behavior of the response outside the calibration limits. Ignored if new_data is defined. Default = TRUE.
add_bar	(logical) whether to add bar legend. Default = TRUE.
add_limits	(logical) whether to add calibration limits if extrapolate = TRUE. Default = TRUE.
color_palette	(function) a color palette function to be used to assign colors in the plot. The default, NULL uses rev(hcl.colors(n, "terrain")).
xlab	(character) a label for the x axis. The default, NULL, uses the name defined in variable1.
ylab	(character) a label for the y axis. The default, NULL, uses the name defined in variable2.
...	additional arguments passed to image.

Value

A bivariate plot considering variable1 and variable2.

See Also

response_curve()

Examples

```
# Example with glmnet
# Import example of fitted_models (output of fit_selected())
data(fitted_model_maxnet, package = "kuenm2")

# Response curve (notice response affected by covariance)
bivariate_response(models = fitted_model_maxnet, modelID = "Model_219",
                   variable1 = "bio_1", variable2 = "bio_12")

# Example with glm
# Import example of fitted_models (output of fit_selected())
data(fitted_model_glm, package = "kuenm2")

# Response curve
bivariate_response(models = fitted_model_glm, modelID = "Model_85",
                   variable1 = "bio_1", variable2 = "bio_7")
```

`calib_results_glm` *Calibration Results (glm)*

Description

A `calibration_results` object resulted from `calibration()` using maxnet algorithm

Usage

```
data("calib_results_glm")
```

Format

A `calibration_results` with the following elements:

species Species names

calibration_data A `data.frame` with the variables extracted to presence and background points

formula_grid A `data.frame` with the ID, formulas, and regularization multipliers of each candidate model

part_data A `list` with the partition data, where each element corresponds to a replicate and contains the **indices of the test points** for that replicate

partition_method A `character` indicating the partition method

n_replicates A `numeric` value indicating the number of replicates or k-folds

train_proportion A `numeric` value indicating the proportion of occurrences used as train points when the partition method is 'subsample' or 'bootstrap'

data_xy A `data.frame` with the coordinates of the occurrence and background points

continuous_variables A `character` indicating the names of the continuous variables

categorical_variables A `character` indicating the names of the categorical variables

weights A `numeric` value specifying weights for the occurrence records. It's NULL, meaning it was not set weights.

pca A `prcomp` object storing PCA information. Is NULL, meaning PCA was not performed

algorithm A `character` indicating the algorithm (glm)

calibration_results A `list` containing the evaluation metrics for each candidate model

omission_rate A `numeric` value indicating the omission rate used to evaluate the models (10%)

addsamplestobackground A `logical` value indicating whether to add to the background any presence sample that is not already there.

selected_models A `data.frame` with the formulas and evaluation metrics for each selected model

summary A `list` with the number and the ID of the models removed and selected during selection procedure

`calib_results_maxnet` *Calibration Results (Maxnet)*

Description

A `calibration_results` object resulted from `calibration()` using maxnet algorithm

Usage

```
data("calib_results_maxnet")
```

Format

A `calibration_results` with the following elements:

species Species names

calibration_data A `data.frame` with the variables extracted to presence and background points

formula_grid A `data.frame` with the ID, formulas, and regularization multipliers of each candidate model

part_data A `list` with the partition data, where each element corresponds to a replicate and contains the **indices of the test points** for that replicate

partition_method A `character` indicating the partition method

n_replicates A `numeric` value indicating the number of replicates or k-folds

train_proportion A `numeric` value indicating the proportion of occurrences used as train points when the partition method is 'subsample' or 'bootstrap'

data_xy A `data.frame` with the coordinates of the occurrence and background points

continuous_variables A `character` indicating the names of the continuous variables

categorical_variables A `character` indicating the names of the categorical variables

weights A `numeric` value specifying weights for the occurrence records. It's NULL, meaning it was not set weights.

pca A `prcomp` object storing PCA information. Is NULL, meaning PCA was not performed

algorithm A `character` indicating the algorithm (maxnet)

calibration_results A `list` containing the evaluation metrics for each candidate model

omission_rate A `numeric` value indicating the omission rate used to evaluate the models (10%)

addsamplestobackground A `logical` value indicating whether to add to the background any presence sample that is not already there.

selected_models A `data.frame` with the formulas and evaluation metrics for each selected model

summary A `list` with the number and the ID of the models removed and selected during selection procedure

`calibration` *Fitting and evaluation of models, and selection of the best ones*

Description

This function fits and evaluates candidate models using the data and grid of formulas prepared with `prepare_data()`. It supports both algorithms `glm` and `maxnet`. The function then selects the best models based on unimodality (optional), partial ROC, omission rate, and AIC values.

Usage

```
calibration(data, error_considered, remove_concave = FALSE,
            proc_for_all = FALSE, omission_rate = NULL, delta_aic = 2,
            allow_tolerance = TRUE, tolerance = 0.01,
            addsamplestobackground = TRUE, use_weights = NULL,
            write_summary = FALSE, output_directory = NULL,
            skip_existing_models = FALSE, return_all_results = TRUE,
            parallel = FALSE, ncores = NULL, progress_bar = TRUE,
            verbose = TRUE)
```

Arguments

`data`	an object of class `prepared_data` returned by the `prepare_data()` function. It contains the calibration data, formulas grid, kfolds, and model type.
`error_considered`	(numeric) values from 0 to 100 representing the percentage of potential error due to any source of uncertainty in your data. This value is used to calculate omission rates and partial ROC. See details.
`remove_concave`	(logical) whether to remove candidate models presenting concave curves. Default is FALSE.
`proc_for_all`	(logical) whether to apply partial ROC tests to all candidate models or only to the selected models. Default is FALSE, meaning that tests are applied only to the selected models.
`omission_rate`	(numeric) values from 0 - 100, the maximum omission rate a candidate model can have to be considered as a potentially selected model. The default, NULL, uses the value in `error_considered`. If more that one value is used in `error_considered`, `omission_rate` must be defined.
`delta_aic`	(numeric) the value of delta AIC used as a threshold to select models. Default is 2.
`allow_tolerance`	(logical) whether to allow selection of models with minimum values of omission rates even if their omission rate surpasses the `omission_rate`. This is only applicable if all candidate models have omission rates higher than the `omission_rate`. Default is TRUE.

`tolerance` (numeric) The value added to the minimum omission rate if it exceeds the `omission_rate`. If `allow_tolerance = TRUE`, selected models will have an omission rate equal to or less than the minimum rate plus this tolerance. Default is 0.01.

`addsamplestobackground` (logical) whether to add to the background any presence sample that is not already there. Default is TRUE.

`use_weights` (logical) whether to apply the weights present in the data. The default, NULL, uses weights provided in `data`. If they are not present in `data`, NULL weights are 1 for presences and 100 for background. If turned to FALSE, it uses NULL weights even if present in `data`.

`write_summary` (logical) whether to save the evaluation results for each candidate model to disk. Default is FALSE.

`output_directory` (character) the file name, with or without a path, for saving the evaluation results for each candidate model. This is only applicable if `write_summary = TRUE`.

`skip_existing_models` (logical) whether to check for and skip candidate models that have already been fitted and saved in `output_directory`. This is only applicable if `write_summary = TRUE`. Default is FALSE.

`return_all_results` (logical) whether to return the evaluation results for each replicate. Default is TRUE, meaning evaluation results for each replicate will be returned.

`parallel` (logical) whether to fit the candidate models in parallel. Default is FALSE.

`ncores` (numeric) number of cores to use for parallel processing. Default is NULL and uses available cores - 1. This is only applicable if `parallel = TRUE`.

`progress_bar` (logical) whether to display a progress bar during processing. Default is TRUE.

`verbose` (logical) whether to display messages during processing. Default is TRUE.

Details

Partial ROC is calculated using the values defined in `error_considered` following Peterson et al. (2008).

Omission rates are calculated using separate testing data subsets. Users can specify multiple values of `error_considered` to calculate this metric (e.g., c(5, 10)), but only one can be used as the omission rate for model selection.

Model fitting and complexity (AICc) is assessed using models generated with the complete set of occurrences. AICc values are computed as proposed by Warren and Seifert (2011).

Value

An object of class 'calibration_results' containing the following elements:

- species: a character string with the name of the species.
- calibration data: a data.frame containing a column (`pr_bg`) that identifies occurrence points (1) and background points (0), along with the corresponding values of predictor variables for each point.

- formula_grid: data frame containing the calibration grid with possible formulas and parameters.
- kfolds: a list of vectors with row indices corresponding to each fold.
- data_xy: a data.frame with occurrence and background coordinates.
- continuous_variables: a character indicating the continuous variables.
- categorical_variables: a character, categorical variable names (if used).
- weights: a numeric vector specifying weights for data_xy (if used).
- pca: if a principal component analysis was performed with variables, a list of class "prcomp". See prcomp() for details.
- algorithm: the model type (glm or maxnet)
- calibration_results: a list containing a data frame with all evaluation metrics for all partitions (if `return_all_results = TRUE`) and a summary of the evaluation metrics for each candidate model.
- omission_rate: The omission rate used to select models.
- addsamplestobackground: a logical value indicating whether any presence sample not already in the background was added.
- selected_models: data frame with the ID and the summary of evaluation metrics for the selected models.
- summary: A list containing the delta AIC values for model selection, and the ID values of models that failed to fit, had concave curves, non-significant pROC values, omission rates above the threshold, delta AIC values above the threshold, and the selected models.

References

Ninomiya, Yoshiyuki, and Shuichi Kawano. "AIC for the Lasso in generalized linear models." (2016): 2537-2560.

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

Examples

```
# Import occurrences
data(occ_data, package = "kuenm2")

# Import raster layers
var <- terra::rast(system.file("extdata", "Current_variables.tif",
                               package = "kuenm2"))
# Use only continuous variables
var <- var[[c("bio_1", "bio_7", "bio_12", "bio_15")]]

# Prepare data for maxnet model
sp_swd <- prepare_data(algorithm = "maxnet", occ = occ_data,
                       x = "x", y = "y",
                       raster_variables = var,
                       species = occ_data[1, 1],
```

```
                        n_background = 100,
                        features = c("l", "lq"),
                        r_multiplier = 1,
                        partition_method = "kfolds")
# Model calibration (maxnet)
m <- calibration(data = sp_swd, error_considered = 10)
m

# Prepare data for glm model
sp_swd_glm <- prepare_data(algorithm = "glm", occ = occ_data,
                           x = "x", y = "y",
                           raster_variables = var,
                           species = occ_data[1, 1],
                           n_background = 100,
                           features = c("l", "lq"),
                           partition_method = "kfolds")
m_glm <- calibration(data = sp_swd_glm, error_considered = 10)
m_glm
```

`chelsa_current` *SpatRaster Representing present-day Conditions (CHELSA)*

Description

Raster layer containing bioclimatic variables representing present-day climatic conditions. The variables were resampled to a 10 arc-minute resolution and masked using the `m` region provided in the package. Data sourced from CHELSA: https://chelsa-climate.org/

Format

A `SpatRaster` object.

Value

No return value. Used with function `rast` to bring raster variables to analysis.

Examples

```
chelsa_current <- terra::rast(system.file("extdata",
                                          "Current_CHELSA.tif",
                                          package = "kuenm2"))
terra::plot(chelsa_current)
```

chelsa_lgm_ccsm4 *SpatRaster Representing LGM Conditions (GCM: CCSM4)*

Description

Raster layer containing bioclimatic variables representing Last Glacial Maximum (LGM) climatic conditions based on the CCSM4 General Circulation Model (GCM). The variables were resampled to 10arc-minutes and masked using the `m` provided in the package.

Format

A `SpatRaster` object.

Value

No return value. Used with function `rast` to bring raster variables to analysis.

Examples

```
chelsa_lgm_ccsm4 <- terra::rast(system.file("extdata",
                                            "CHELSA_LGM_CCSM4.tif",
                                             package = "kuenm2"))

terra::plot(chelsa_lgm_ccsm4)
```

chelsa_lgm_cnrm_cm5 *SpatRaster Representing LGM Conditions (GCM: CNRM-CM5)*

Description

Raster layer containing bioclimatic variables representing Last Glacial Maximum (LGM) climatic conditions based on the CNRM-CM5 General Circulation Model. The variables were resampled to 10arc-minutes and masked using the `m` provided in the package.

Format

A `SpatRaster` object.

Value

No return value. Used with function `rast` to bring raster variables to analysis.

Examples

```
chelsa_lgm_cnrm_cm5 <- terra::rast(system.file("extdata",
                                               "CHELSA_LGM_CNRM-CM5.tif",
                                               package = "kuenm2"))
terra::plot(chelsa_lgm_cnrm_cm5)
```

chelsa_lgm_fgoals_g2 *SpatRaster Representing LGM Conditions (GCM: FGOALS-g2)*

Description

Raster layer containing bioclimatic variables representing Last Glacial Maximum (LGM) climatic conditions based on the FGOALS-g2 General Circulation Model. The variables were resampled to 10arc-minutes and masked using the `m` provided in the package.

Format

A `SpatRaster` object.

Value

No return value. Used with function `rast` to bring raster variables to analysis.

Examples

```
chelsa_lgm_fgoals_g2 <- terra::rast(system.file("extdata",
                                                "CHELSA_LGM_FGOALS-g2.tif",
                                                package = "kuenm2"))
terra::plot(chelsa_lgm_fgoals_g2)
```

chelsa_lgm_ipsl *SpatRaster Representing LGM Conditions (GCM: IPSL-CM5A-LR)*

Description

Raster layer containing bioclimatic variables representing Last Glacial Maximum (LGM) climatic conditions based on the IPSL-CM5A-LR General Circulation Model. The variables were resampled to 10arc-minutes and masked using the `m` provided in the package.

Format

A `SpatRaster` object.

Value

No return value. Used with function `rast` to bring raster variables to analysis.

Examples

```
chelsa_lgm_ipsl <- terra::rast(system.file("extdata",
                                           "CHELSA_LGM_IPSL-CM5A-LR.tif",
                                           package = "kuenm2"))
terra::plot(chelsa_lgm_ipsl)
```

chelsa_lgm_miroc *SpatRaster Representing LGM Conditions (GCM: MIROC-ESM)*

Description

Raster layer containing bioclimatic variables representing Last Glacial Maximum (LGM) climatic conditions based on the MIROC-ESM General Circulation Model. The variables were resampled to 10arc-minutes and masked using the `m` provided in the package.

Format

A `SpatRaster` object.

Value

No return value. Used with function `rast` to bring raster variables to analysis.

Examples

```
chelsa_lgm_miroc <- terra::rast(system.file("extdata",
                                            "CHELSA_LGM_MIROC-ESM.tif",
                                            package = "kuenm2"))
terra::plot(chelsa_lgm_miroc)
```

chelsa_lgm_mpi *SpatRaster Representing LGM Conditions (GCM: MPI-ESM-P)*

Description

Raster layer containing bioclimatic variables representing Last Glacial Maximum (LGM) climatic conditions based on the MPI-ESM-P General Circulation Model. The variables were resampled to 10arc-minutes and masked using the `m` provided in the package.

Format

A `SpatRaster` object.

Value

No return value. Used with function `rast` to bring raster variables to analysis.

Examples

```
chelsa_lgm_mpi <- terra::rast(system.file("extdata",
                                          "CHELSA_LGM_MPI-ESM-P.tif",
                                          package = "kuenm2"))
terra::plot(chelsa_lgm_mpi)
```

chelsa_lgm_mri *SpatRaster Representing LGM Conditions (GCM: MRI-CGCM3)*

Description

Raster layer containing bioclimatic variables representing Last Glacial Maximum (LGM) climatic conditions based on the MRI-CGCM3 General Circulation Model. The variables were resampled to 10arc-minutes and masked using the `m` provided in the package.

Format

A `SpatRaster` object.

Value

No return value. Used with function `rast` to bring raster variables to analysis.

Examples

```
chelsa_lgm_mri <- terra::rast(system.file("extdata",
                                          "CHELSA_LGM_MRI-CGCM3.tif",
                                          package = "kuenm2"))
terra::plot(chelsa_lgm_mri)
```

colors_for_changes *Set Colors for Change Maps*

Description

This functions sets the color tables associated with the `SpatRaster` object resulting from `projection_changes()`. Color tables are used to associate specific colors with raster values when using `plot()`. This function defines custom colors for areas of gain, loss, and stability across scenarios.

Usage

```
colors_for_changes(
  changes_projections,
  gain_color = "#009E73",
  loss_color = "#D55E00",
  stable_suitable = "#0072B2",
  stable_unsuitable = "grey",
  max_alpha = 1,
  min_alpha = 0.25
)
```

Arguments

changes_projections	an object of class changes_projections, generated by projection_changes() or imported using import_projections(), containing the $Summary_changes element.
gain_color	(character) color used to define the palette for representing gains. Default is "#009E73" (teal green).
loss_color	(character) color used to define the palette for representing losses. Default is "#D55E00" (orange-red).
stable_suitable	(character) color used for representing areas that remain suitable across all scenarios. Default is "#0072B2" (oxford blue).
stable_unsuitable	(character) color used for representing areas that remain unsuitable across all scenarios. Default is "grey".
max_alpha	(numeric) opacity value (from 0 to 1) for areas where all GCMs agree on the change (gain, loss, or stability). Default is 1.
min_alpha	(numeric) opacity value (from 0 to 1) for areas where only one GCM predicts a given change. Default is 0.25

Value

An object of class changes_projections with the same structure and SpatRasters as the input changes_projections, but with color tables embedded in the SpatRasters. These colors are used automatically when visualizing the data with plot().

Examples

```
# Step 1: Organize variables for current projection
## Import current variables (used to fit models)
var <- terra::rast(system.file("extdata", "Current_variables.tif",
                               package = "kuenm2"))

## Create a folder in a temporary directory to copy the variables
out_dir_current <- file.path(tempdir(), "Current_raw_color_example")
dir.create(out_dir_current, recursive = TRUE)

## Save current variables in temporary directory
terra::writeRaster(var, file.path(out_dir_current, "Variables.tif"))


# Step 2: Organize future climate variables (example with WorldClim)
## Directory containing the downloaded future climate variables (example)
in_dir <- system.file("extdata", package = "kuenm2")

## Create a folder in a temporary directory to copy the future variables
out_dir_future <- file.path(tempdir(), "Future_raw_color_example")

## Organize and rename the future climate data (structured by year and GCM)
```

```
### 'SoilType' will be appended as a static variable in each scenario
organize_future_worldclim(input_dir = in_dir, output_dir = out_dir_future,
                          name_format = "bio_",
                          static_variables = var$SoilType)

# Step 3: Prepare data to run multiple projections
## An example with maxnet models
## Import example of fitted_models (output of fit_selected())
data(fitted_model_maxnet, package = "kuenm2")

## Prepare projection data using fitted models to check variables
pr <- prepare_projection(models = fitted_model_maxnet,
                         present_dir = out_dir_current,
                         future_dir = out_dir_future,
                         future_period = c("2081-2100"),
                         future_pscen = c("ssp126", "ssp585"),
                         future_gcm = c("ACCESS-CM2", "MIROC6"),
                         raster_pattern = ".tif*")

# Step 4: Run multiple model projections
## A folder to save projection results
out_dir <- file.path(tempdir(), "Projection_results/maxnet_color_example")
dir.create(out_dir, recursive = TRUE)

## Project selected models to multiple scenarios
p <- project_selected(models = fitted_model_maxnet, projection_data = pr,
                      out_dir = out_dir)

# Step 5: Identify areas of change in projections
## Contraction, expansion and stability
changes <- projection_changes(model_projections = p, by_gcm = TRUE,
                              by_change = TRUE, write_results = FALSE,
                              return_raster = TRUE)

#Step 6: Set Colors for Change Maps
changes_with_colors <- colors_for_changes(changes_projections = changes)
terra::plot(changes_with_colors$Summary_changes)
```

`detect_concave` *Detect concave curves in GLM and GLMNET models*

Description

Identifies the presence of concave response curves within the calibration range of GLM and GLMNET models.

Usage

```
detect_concave(model, calib_data, extrapolation_factor = 0.1,
               averages_from = "pr", var_limits = NULL, plot = FALSE,
               mfrow = NULL, legend = FALSE)
```

Arguments

model	an object of class glmnet_mx or glm.
calib_data	data.frame or matrix of data used for model calibration.
extrapolation_factor	(numeric) a multiplier used to calculate the extrapolation range. Larger values allow broader extrapolation beyond the observed data range. Default is 0.1.
averages_from	(character) specifies how the averages or modes of the variables are calculated. Available options are "pr" (to calculate averages from the presence localities) or "pr_bg" (to use the combined set of presence and background localities). Default is "pr". See details.
var_limits	(list) A named list specifying the lower and/or upper limits for some variables. The first value represents the lower limit, and the second value represents the upper limit. Default is NULL, meaning no specific limits are applied, and the range will be calculated using the extrapolation_factor. See details.
plot	(logical) whether to plot the response curve for the variables. Default is FALSE.
mfrow	(numeric) a vector of the form c(number of rows, number of columns) specifying the layout of plots. Default is c(1, 1), meaning one plot per window.
legend	(logical) whether to include a legend in the plot. The legend indicates whether the response curve is convex, concave outside the range limits, or concave within the range limits. Default is FALSE.

Details

Concave curves are identified by analyzing the beta coefficients of quadratic terms within the variable's range. The range for extrapolation is calculated as the difference between the variable's maximum and minimum values in the model, multiplied by the extrapolation factor. A concave curve is detected when the beta coefficient is positive, and the vertex (where the curve changes direction) lies between the lower and upper limits of the variable.

Users can specify the lower and upper limits for certain variables using `var_limits`. For example, if `var_limits = list("bio12" = c(0, NA), "bio15" = c(0, 100))`, the lower limit for `bio12` will be 0, and the upper limit will be calculated using the extrapolation factor. Similarly, the lower and upper limits for `bio15` will be 0 and 100, respectively.

For calculating the vertex position, a response curve for a given variable is generated with all other variables set to their mean values (or mode for categorical variables). These values are calculated either from the presence localities (if `averages_from = "pr"`) or from the combined set of presence and background localities (if `averages_from = "pr_bg"`).

Value

A list with the following elements for each variable:

- is_concave (logical): indicates whether the response curve for the variable is concave within the limit range. This occurs when the quadratic term's coefficient is positive and the vertex lies between x_min and x_max,
- vertex (numeric): the vertex of the parabola, representing the point where the curve changes direction.

- b2 (numeric): the coefficient of the quadratic term for the variable. Positive values indicate a concave curve.
- x_min and x_max (numeric): the range limits to identify concave curves, calculated as the observed data range multiplied by the extrapolation factor.
- real_x_min and real_x_max (numeric) the actual range of the data, excluding the extrapolation factor.

Examples

```
# Import example of a fitted_model (output of fit_selected()) that have
# concave curves
data("fitted_model_concave", package = "kuenm2")

#Response curves
ccurves <- detect_concave(model = fitted_model_concave$Models$Model_798$Full_model,
                          calib_data = fitted_model_concave$calibration_data,
                          extrapolation_factor = 0.2,
                          var_limits = list("bio_2" = c(0, NA),
                                            "sand" = c(0, NA),
                                            "clay" = c(0, NA)),
                          plot = TRUE, mfrow = c(2, 3), legend = TRUE)
```

`enmeval_block` *Spatial Blocks from ENMeval*

Description

A list resulting from `ENMeval::get.block()` to partition occurrence and background localities into bins for training and validation (or, evaluation and calibration). This object is used in the "Prepare Data for Model Calibration" vignette to demonstrate how to implement custom data partitions generated by `ENMeval` in `kuenm2`.

Usage

```
data("enmeval_block")
```

Format

A `list` with the following elements:

occs.grp A `numeric` vector indicating the spatial group to which each occurrence belongs

bg.grp A `numeric` vector indicating the spatial group to which each background point belongs

`explore_calibration_hist` *Explore variable distribution for occurrence and background points*

Description

This function prepares data to generate overlaid histograms to visualize the distribution of predictor variables for occurrence (presence) and background points.

Usage

```
explore_calibration_hist(data, include_m = FALSE, raster_variables = NULL,
                         magnify_occurrences = 2, breaks = 15)
```

Arguments

`data`	an object of class `prepared_data` returned by the `prepare_data`() function.
`include_m`	(logical) whether to include data for plotting the histogram of the entire area from which background points were sampled. Default is FALSE, meaning only background and presence information will be plotted.
`raster_variables`	(SpatRaster) predictor variables used to prepared the data with `prepared_data`. Only applicable if `include_m` is TRUE.
`magnify_occurrences`	(numeric) factor by which the frequency of occurrences is magnified for better visualization. Default is 2, meaning occurrence frequencies in the plot will be doubled.
`breaks`	(numeric) a single number giving the desired number of intervals in the histogram.

Value

A list of with information to plot informative histograms to explore data to be used in the modeling process. Histogram plots can be plotted with the function `plot_calibration_hist()`.

See Also

`plot_calibration_hist()`

Examples

```
# Import raster layers
var <- terra::rast(system.file("extdata", "Current_variables.tif",
                               package = "kuenm2"))

# Import occurrences
data(sp_swd, package = "kuenm2")
```

```
# Explore calibration data
calib_hist <- explore_calibration_hist(data = sp_swd,
                                       raster_variables = var,
                                       include_m = TRUE)

# To visualize results use the function plot_calibration_hist()
```

`explore_partition_env` *Explore the Distribution of Partitions in Environmental Space*

Description

Plots training and testing data (presences and backgrounds) in a two-dimensional environmental space. This space can be defined either by performing a PCA on all environmental variables or by specifying two environmental variables manually.

Usage

```
explore_partition_env(data, show_unused_data = FALSE,
                              raster_variables = NULL, mask = NULL,
                              variables = NULL, type_of_plot = "combined",
                              use_pca = TRUE, pcs = c("PC1", "PC2"),
                              partition_palette = "cols25",
                              custom_partition_palette = NULL,
                              include_test_background = TRUE,
                              pr_train_col = "#009E73",
                              pr_test_col = "#D55E00",
                              bg_train_col = "grey",
                              bg_test_col = "#56B4E9", pr_transparency = 0.75,
                              bg_transparency = 0.4, pch = 19, cex_plot = 1.2,
                              size_text_legend = 1, ...)
```

Arguments

`data`	an object of class `prepared_data` returned by the prepare_data() function
`show_unused_data`	(logical) whether to plot the distribution of environmental conditions that are not represented by the background points. If set to TRUE, the `raster_variables` must be provided. Only applicable when `type_of_plot = "combined"`.
`raster_variables`	a `SpatRaster` object representing the predictor variables used to calibrate the models. Preferably the same object used in `prepare_data`. Required only when `show_unused_data = TRUE`. Default is NULL.
`mask`	(SpatRaster, SpatVector, or SpatExtent) spatial object used to mask `raster_variables` to the area where the model will be calibrated. Preferably the same object used in `prepare_data` (if applicable). Only used when `show_unused_data = TRUE`. Default is NULL.

variables (character) names of the variables in data to define the two-dimensional environmental space. If use_pca = TRUE, these variables will be used to perform the PCA. If use_pca = FALSE, this must be a character vector with exactly two variable names (e.g., c("bio_1", "bio_12")). Default is NULL, meaning all variables in data will be used.

type_of_plot (character) the type of plot. Options are "combined" and "individual". See details. Default is "combined".

use_pca (logical) whether to use PCA variables to define the environmental space. If TRUE, a PCA will be performed on the variables, unless data already includes a PCA object from using prepare_data(do_pca = TRUE). Default is TRUE.

pcs (character) the two PCA axes to use to define the two-dimensional environmental space. Default is c("PC1", "PC2"), meaning the first two axes will be used. Only applicable if use_pca = TRUE.

partition_palette
(character) the color palette used to color the different partitions. See ?kuenm2_discrete_palettes to check available options. Default is "cols25".

custom_partition_palette
(character) a character vector defining custom colors for the different partitions. The number of values must match the number of partitions in data. Default is NULL, meaning the palette defined in partition_palette will be used.

include_test_background
(logical) whether to include background points that were not used for training when plotting individual partition plots. Default is TRUE.

pr_train_col (character) the color used for train records in the individual plots. Default is "009E73".

pr_test_col (character) the color used for test records in the individual plots. Default is "D55E00".

bg_train_col (character) the color used for train backgrounds in the individual plots. Default is "56B4E9".

bg_test_col (character) the color used for test backgrounds in the individual plots. Default is "gray". Only applicable if include_test_background = TRUE.

pr_transparency
(numeric) a value between 0 (fully transparent) and 1 (fully opaque) defining the transparency of the points representing presences. Default is 0.75.

bg_transparency
(numeric) a value between 0 (fully transparent) and 1 (fully opaque) defining the transparency of the points representing background points. Default is 0.4.

pch (numeric) a value between 1 and 25 to specify the point shape. See ?pch for details. Default is 19 (solid circle).

cex_plot (numeric) specify the size of the points in the plot. Default is 1.2.

size_text_legend
(numeric) specify the size of the text of the legend. Default is 1.

... additional arguments passed to plot.

Details

The function provides two types of plots:

- **combined**: two plots side by side, one showing the presences and another showing the background points. The colors of the points represent the partitions. This is the default option.
- **individual**: one plot per partition. In each plot, the colors of the points represent those used as train records, test records, train background, or test background (i.e., not used during training in the specified partition).

To obtain both types of plots, set: `type_of_plot = c("combined", "individual")`.

Value

Plots showing the training and testing data in a two-dimensional environmental space.

Examples

```
# Prepare data
# Import occurrences
data(occ_data, package = "kuenm2")

# Import raster layers
var <- terra::rast(system.file("extdata", "Current_variables.tif",
                               package = "kuenm2"))

# Prepare data for maxnet model
sp_swd <- prepare_data(algorithm = "maxnet", occ = occ_data,
                       x = "x", y = "y",
                       raster_variables = var,
                       species = occ_data[1, 1],
                       n_background = 100,
                       categorical_variables = "SoilType",
                       features = c("l", "lq"),
                       r_multiplier = 1,
                       partition_method = "kfolds")

# Explore the Distribution of Partitions in Environmental Space
explore_partition_env(data = sp_swd, show_unused_data = TRUE,
                      raster_variables = var,
                      type_of_plot = c("combined", "individual"))
```

---

explore_partition_extrapolation

Analysis of extrapolation risks in partitions using the MOP metric

---

Description

This function calculates environmental dissimilarities and identifies non-analogous conditions by comparing the training data against the test data for each partition, using the MOP (Mobility-Oriented Parity) metric.

Usage

```
explore_partition_extrapolation(data, include_train_background = TRUE,
                                      include_test_background = FALSE,
                                      variables = NULL,
                                      mop_type = "detailed",
                                      calculate_distance = TRUE,
                                      where_distance = "all",
                                      progress_bar = FALSE, ...)
```

Arguments

data	an object of class prepared_data returned by the prepare_data() function.
include_train_background	(logical) whether to include the background points used in training to define the environmental range of the training data. If set to FALSE, only the environmental conditions of the training presence records will be considered. Default is TRUE, meaning both presence and background points are used.
include_test_background	(logical) whether to compute MOP for both the test presence records and the background points not used during training. Default is FALSE, meaning MOP will be calculated only for the test presences.
variables	(character) names of the variables to be used in the MOP calculation. Default is NULL, meaning all variables in data will be used.
mop_type	(character) type of MOP analysis to be performed. Options available are "basic", "simple" and "detailed". Default is 'simples'. See projection_mop() for more details.
calculate_distance	(logical) whether to calculate distances (dissimilarities) between train and test data. Default is TRUE.
where_distance	(character) specifies which values in train data should be used to calculate distances. Options are: "in_range" (only conditions within the train range), "out_range" (only conditions outside the train range), and "all" (all conditions). Default is "all".
progress_bar	(logical) whether to display a progress bar during processing. Default is FALSE.
...	additional arguments passed to mop().

Value

A data.frame containing:

- MOP distances (if calculate_distance = TRUE);
- an indicator of whether environmental conditions at each test record fall within the training range;
- the number of variables outside the training range;
- the names of variables with values lower or higher than the training range;
- if the prepared_data object includes categorical variables, it will also contain columns indicating which values in the testing data were not present in the training data.

Examples

```
#Prepare data
# Import occurrences
data(occ_data, package = "kuenm2")

# Import raster layers
var <- terra::rast(system.file("extdata", "Current_variables.tif",
                               package = "kuenm2"))

# Prepare data for maxnet model
sp_swd <- prepare_data(algorithm = "maxnet", occ = occ_data,
                       x = "x", y = "y",
                       raster_variables = var,
                       species = occ_data[1, 1],
                       n_background = 100,
                       categorical_variables = "SoilType",
                       features = c("l", "lq"),
                       r_multiplier = 1,
                       partition_method = "kfolds")

# Analysis of extrapolation risks in partitions
res <- explore_partition_extrapolation(data = sp_swd)
```

---

explore_partition_geo *Explore the spatial distribution of partitions for occurrence and background points*

Description

Explore the spatial distribution of partitions for occurrence and background points

Usage

```
explore_partition_geo(data, raster_variables, mask = NULL,
                      show_partitions = TRUE, partition_palette = "cols25",
                      custom_partition_palette = NULL, pr_col = "#D55E00",
                      bg_col = "#0072B2", pr_bg_col = "#CC79A7",
                      calibration_area_col = "gray80", ...)
```

Arguments

data	an object of class prepared_data returned by the prepare_data() function.
raster_variables	(SpatRaster) predictor variables used for model calibration.
mask	(SpatRaster, SpatVector, or SpatExtent) spatial object used to mask raster_variables to the area where the model will be calibrated. Preferably the same object used in prepare_data (if applicable). Default is NULL.

show_partitions
: (logical) whether to return SpatRaster showing the spatial distribution of each partition for presence and background points. Default is TRUE.

partition_palette
: (character) the color palette used to color the different partitions. See ?kuenm2_discrete_palettes to check available options. Default is "cols25". Only applicable if show_partitions = TRUE.

custom_partition_palette
: (character) a character vector defining custom colors for the different partitions. The number of values must match the number of partitions in data. Default is NULL, meaning the palette defined in partition_palette will be used.

pr_col
: (character) the color used for cells with presence records. Default is "#D55E00".

bg_col
: (character) the color used for cells with background points. Default is "#0072B2".

pr_bg_col
: (character) the color used for cells with presences and background points. Default is "#CC79A7".

calibration_area_col
: (character) the color used for cells without presences or background points. Default is "gray80".

...
: additional arguments passed to terra::plot().

Value

A categorical SpatRaster with four factor values representing:

1 - Background cells

2 - Presence cells

3 - Cells with both presence and background

4 - Non-used cells

If show_partitions = TRUE, it also returns SpatRaster showing the spatial distribution of each partition for presence and background points.

Examples

```
# Import raster layers
var <- terra::rast(system.file("extdata", "Current_variables.tif",
                               package = "kuenm2"))

# Import prepared_data
data(sp_swd, package = "kuenm2")

# Explore partitions in the geographic space
pbg <- explore_partition_geo(data = sp_swd, raster_variables = var[[1]])
terra::plot(pbg)
```

`extract_occurrence_variables` *Extracts Environmental Variables for Occurrences*

Description

This function extracts values from environmental or predictor variables (`SpatRaster`) for georeferenced occurrence points. It also adds a column indicating that these are presence points(pr_bg = 1).

Usage

```
extract_occurrence_variables(occ, x, y, raster_variables)
```

Arguments

`occ`	A data.frame containing occurrence data. It must include columns with longitude (x) and latitude (y) coordinates.
`x`	(character) a string specifying the name of the column in occ that contains the longitude values.
`y`	(character) a string specifying the name of the column in occ that contains the latitude values.
`raster_variables`	(SpatRaster) predictor variables used to calibrate the models.

Value

A data.frame containing the original x and y coordinates of the occurrence points (x and y), the values of the variables extracted from `raster_variables`, and a new column `pr_bg` with a value of 1 for all occurrences.

Examples

```
# Import occurrences
data(occ_data, package = "kuenm2")

# Import raster layers
var <- terra::rast(system.file("extdata", "Current_variables.tif",
                               package = "kuenm2"))

# Extracts environmental variables for occurrences
occ_var <- extract_occurrence_variables(occ = occ_data, x = "x", y = "y",
                                        raster_variables = var)
```

extract_var_from_formulas *Extract predictor names from formulas*

Description

Extract predictor names from formulas

Usage

```
extract_var_from_formulas(formulas, ...)
```

Arguments

formulas	(character or formula) model formulas.
...	Arguments to pass to all.vars()

Value

A character vector or a list of the same length as formulas, containing the names of the predictors each formula.

Examples

```
# Import an example of calibration results
data(calib_results_maxnet, package = "kuenm2")

# Extract predictor names
vars <- extract_var_from_formulas(calib_results_maxnet$formula_grid$Formulas)
```

fit_selected *Fit models selected after calibration*

Description

This function fits models selected during model calibration().

Usage

```
fit_selected(calibration_results, replicate_method = "kfolds",
             n_replicates = 1, sample_proportion = 0.7, type = "cloglog",
             write_models = FALSE,
             file_name = NULL, parallel = FALSE, ncores = NULL,
             progress_bar = TRUE, verbose = TRUE, seed = 1)
```

Arguments

`calibration_results`
: an object of class `calibration_results` returned by the `calibration()` function.

`replicate_method`
: (character) method used for producing replicates. Available options are `"kfolds"`, `"subsample"`, and `"bootstrap"`. See **Details** for more information.

`n_replicates`
: (numeric) number of replicates or folds to generate. If `replicate_method` is `"subsample"` or `"bootstrap"`, this defines the number of replicates. If `"kfolds"`, it specifies the number of folds. Default is 4.

`sample_proportion`
: (numeric) proportion of occurrence and background points to be used to fit model replicates. Only applicable when `replicate_method` is `"subsample"` or `"bootstrap"`. Default is 0.7 (i.e., 70% of the data).

`type`
: (character) the format of prediction values for computing thresholds. For maxnet models, valid options are "raw", "cumulative", "logistic", and "cloglog". For glm models, valid options are "cloglog", "response" and "raw". Default is "cloglog".

`write_models`
: (logical) whether to save the final fitted models to disk. Default is FALSE.

`file_name`
: (character) the file name, with or without a path, for saving the final models. This is only applicable if `write_models = TRUE`.

`parallel`
: (logical) whether to fit the final models in parallel. Default is FALSE.

`ncores`
: (numeric) number of cores to use for parallel processing. Default is NULL and uses available cores - 1. This is only applicable if `parallel = TRUE`.

`progress_bar`
: (logical) whether to display a progress bar during processing. Default is TRUE.

`verbose`
: (logical) whether to display detailed messages during processing. Default is TRUE.

`seed`
: (numeric) integer value used to specify an initial seed to split the data. Default is 1.

Details

This function also computes model consensus (mean and median), the thresholds to binarize model predictions based on the omission rate set during model calibration to select models.

Value

An object of class 'fitted_models' containing the following elements:

`species`
: a character string with the name of the species.

`Models`
: a list of fitted models, including replicates (fitted with part of the data) and full models (fitted with all data).

`calibration_data`
: a data.frame containing a column (`pr_bg`) that identifies occurrence points (1) and background points (0), along with the corresponding values of predictor variables for each point.

selected_models
: a data frame with the ID and summary of evaluation metrics for the selected models.

weights
: a numeric vector specifying weights for the predictor variables (if used).

pca
: a list of class prcomp representing the result of principal component analysis (if performed).

addsamplestobackground
: a logical value indicating whether any presence sample not already in the background was added.

omission_rate
: the omission rate determined during the calibration step.

thresholds
: the thresholds to binarize each replicate and the consensus (mean and median), calculated based on the omission rate set in calibration().

Examples

```
# An example with maxnet models
data(calib_results_maxnet, package = "kuenm2")

# Fit models using calibration results
fm <- fit_selected(calibration_results = calib_results_maxnet,
                   n_replicates = 4)

# Output the fitted models
fm

# An example with GLMs
data(calib_results_glm, package = "kuenm2")

# Fit models using calibration results
fm_glm <- fit_selected(calibration_results = calib_results_glm,
                       replicate_method = "subsample",
                       n_replicates = 5)

# Output the fitted models
fm_glm
```

---

fitted_model_chelsa *Fitted model with CHELSA variables*

Description

A fitted_models object resulting from fit_selected() using calibration data based on CHELSA variables.

Usage

```
data("fitted_model_chelsa")
```

Format

A `fitted_models` with the following elements:

species Species names

Models A `list` with the fitted maxnet models (replicates and full models)

calibration_data A `data.frame` containing the variables extracted for presence and background points

continuous_variables A `character` indicating the names of the continuous variables

categorical_variables A `character` indicating the names of the categorical variables

selected_models A `data.frame` with formulas and evaluation metrics for each selected model

weights A `numeric` vector specifying weights for the occurrence records. `NULL` if no weights were set.

pca A `prcomp` object containing PCA results. `NULL` if PCA was not performed.

addsamplestobackground A `logical` value indicating whether to add any presence point not already included to the background.

omission_rate A `numeric` value indicating the omission rate used to evaluate models.

thresholds A `numeric` vector with thresholds used to binarize each replicate and the consensus (mean and median), calculated based on the omission rate defined in `calibration()`.

algorithm A `character` string indicating the algorithm used (maxnet).

partition_method A `character` string indicating the partitioning method used.

n_replicates A `numeric` value indicating the number of replicates or folds.

train_proportion A `numeric` value indicating the proportion of occurrences used for training when the partition method is 'subsample' or 'bootstrap'.

`fitted_model_concave` *Fitted model with concave curves*

Description

A maxnet `fitted_models` object resulting from `fit_selected()` with a model presenting concave curves.

Usage

```
data("fitted_model_concave")
```

Format

A `fitted_models` with the following elements:

species Species names

Models A `list` with the fitted maxnet models (replicates and full models)

calibration_data A `data.frame` containing the variables extracted for presence and background points

continuous_variables A `character` indicating the names of the continuous variables

categorical_variables A `character` indicating the names of the categorical variables

selected_models A `data.frame` with formulas and evaluation metrics for each selected model

weights A `numeric` vector specifying weights for the occurrence records. `NULL` if no weights were set.

pca A `prcomp` object containing PCA results. `NULL` if PCA was not performed.

addsamplestobackground A `logical` value indicating whether to add any presence point not already included to the background.

omission_rate A `numeric` value indicating the omission rate used to evaluate models.

thresholds A `numeric` vector with thresholds used to binarize each replicate and the consensus (mean and median), calculated based on the omission rate defined in `calibration()`.

algorithm A `character` string indicating the algorithm used (maxnet).

partition_method A `character` string indicating the partitioning method used.

n_replicates A `numeric` value indicating the number of replicates or folds.

`fitted_model_glm` *Fitted model with glm algorithm*

Description

A glm `fitted_models` object resulting from `fit_selected()` using calibration data with based on WorldClim variables.

Usage

```
data("fitted_model_glm")
```

Format

A `fitted_models` with the following elements:

species Species names

Models A `list` with the fitted maxnet models (replicates and full models)

calibration_data A `data.frame` containing the variables extracted for presence and background points

continuous_variables A `character` indicating the names of the continuous variables

categorical_variables A `character` indicating the names of the categorical variables

selected_models A `data.frame` with formulas and evaluation metrics for each selected model

weights A `numeric` vector specifying weights for the occurrence records. `NULL` if no weights were set.

pca A `prcomp` object containing PCA results. `NULL` if PCA was not performed.

addsamplestobackground A `logical` value indicating whether to add any presence point not already included to the background.

omission_rate A `numeric` value indicating the omission rate used to evaluate models.

thresholds A `numeric` vector with thresholds used to binarize each replicate and the consensus (mean and median), calculated based on the omission rate defined in `calibration()`.

algorithm A `character` string indicating the algorithm used (glm).

partition_method A `character` string indicating the partitioning method used.

n_replicates A `numeric` value indicating the number of replicates or folds.

train_proportion A `numeric` value indicating the proportion of occurrences used for training when the partition method is 'subsample' or 'bootstrap'.

`fitted_model_maxnet` *Fitted model with maxnet algorithm*

Description

A maxnet `fitted_models` object resulting from `fit_selected()` using calibration data with based on WorldClim variables.

Usage

```
data("fitted_model_maxnet")
```

Format

A `fitted_models` with the following elements:

species Species names

Models A `list` with the fitted maxnet models (replicates and full models)

calibration_data A `data.frame` containing the variables extracted for presence and background points

continuous_variables A `character` indicating the names of the continuous variables

categorical_variables A `character` indicating the names of the categorical variables

selected_models A `data.frame` with formulas and evaluation metrics for each selected model

weights A `numeric` vector specifying weights for the occurrence records. `NULL` if no weights were set.

pca A prcomp object containing PCA results. NULL if PCA was not performed.

addsamplestobackground A logical value indicating whether to add any presence point not already included to the background.

omission_rate A numeric value indicating the omission rate used to evaluate models.

thresholds A numeric vector with thresholds used to binarize each replicate and the consensus (mean and median), calculated based on the omission rate defined in calibration().

algorithm A character string indicating the algorithm used (maxnet).

partition_method A character string indicating the partitioning method used.

n_replicates A numeric value indicating the number of replicates or folds.

train_proportion A numeric value indicating the proportion of occurrences used for training when the partition method is 'subsample' or 'bootstrap'.

---

flexsdm_block *Spatial Blocks from flexsdm*

Description

A list resulting from flexsdm::part_sblock(), used to partition occurrence and background localities into bins for training and evaluation. This object is used in the "Prepare Data for Model Calibration" vignette to demonstrate how to implement custom data partitions generated by flexsdm in kuenm2

Usage

```
data("enmeval_block")
```

Format

A list with the following elements:

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.

future_2050_ssp126_access

SpatRaster Representing Future Conditions (2041-2060, SSP126, GCM: ACCESS-CM2)

Description

A raster layer containing bioclimatic variables representing future climatic conditions (2041-2060) based on the ACCESS-CM2 General Circulation Model under the SSP126 scenario. The variables were obtained at a 10 arc-minute resolution and masked using the `m` region provided in the package. Data sourced from WorldClim: https://worldclim.org/data/cmip6/cmip6climate.html

Format

A `SpatRaster` object.

Value

No return value. Used with function `rast` to bring raster variables to analysis.

Examples

```
future_2050_ssp126_access <- terra::rast(system.file("extdata",
                                    "wc2.1_10m_bioc_ACCESS-CM2_ssp126_2041-2060.tif",
                                     package = "kuenm2"))
terra::plot(future_2050_ssp126_access)
```

future_2050_ssp126_miroc

SpatRaster Representing Future Conditions (2041-2060, SSP126, GCM: MIROC6)

Description

A raster layer containing bioclimatic variables representing future climatic conditions (2041-2060) based on the MIROC6 General Circulation Model under the SSP126 scenario. The variables were obtained at a 10 arc-minute resolution and masked using the `m` region provided in the package. Data sourced from WorldClim: https://worldclim.org/data/cmip6/cmip6climate.html

Format

A `SpatRaster` object.

Value

No return value. Used with function `rast` to bring raster variables to analysis.

Examples

```
future_2050_ssp126_miroc <- terra::rast(system.file("extdata",
                                    "wc2.1_10m_bioc_MIROC6_ssp126_2041-2060.tif",
                                     package = "kuenm2"))
terra::plot(future_2050_ssp126_miroc)
```

---

future_2050_ssp585_access

SpatRaster Representing Future Conditions (2041-2060, SSP585, GCM: ACCESS-CM2)

Description

A raster layer containing bioclimatic variables representing future climatic conditions (2041-2060) based on the ACCESS-CM2 General Circulation Model under the SSP585 scenario. The variables were obtained at a 10 arc-minute resolution and masked using the `m` region provided in the package. Data sourced from WorldClim: https://worldclim.org/data/cmip6/cmip6climate.html

Format

A `SpatRaster` object.

Value

No return value. Used with function `rast` to bring raster variables to analysis.

Examples

```
future_2050_ssp585_access <- terra::rast(system.file("extdata",
                                    "wc2.1_10m_bioc_ACCESS-CM2_ssp585_2041-2060.tif",
                                     package = "kuenm2"))
terra::plot(future_2050_ssp585_access)
```

---

future_2050_ssp585_miroc

SpatRaster Representing Future Conditions (2041-2060, SSP585, GCM: MIROC6)

Description

A raster layer containing bioclimatic variables representing future climatic conditions (2041-2060) based on the MIROC6 General Circulation Model under the SSP585 scenario. The variables were obtained at a 10 arc-minute resolution and masked using the `m` region provided in the package. Data sourced from WorldClim: https://worldclim.org/data/cmip6/cmip6climate.html

Format

A SpatRaster object.

Value

No return value. Used with function rast to bring raster variables to analysis.

Examples

```
future_2050_ssp585_miroc <- terra::rast(system.file("extdata",
                                    "wc2.1_10m_bioc_MIROC6_ssp585_2041-2060.tif",
                                     package = "kuenm2"))
terra::plot(future_2050_ssp585_miroc)
```

---

future_2100_ssp126_access

SpatRaster Representing Future Conditions (2081-2100, SSP126, GCM: ACCESS-CM2)

Description

A raster layer containing bioclimatic variables representing future climatic conditions (2081-2100) based on the ACCESS-CM2 General Circulation Model under the SSP126 scenario. The variables were obtained at a 10 arc-minute resolution and masked using the m region provided in the package. Data sourced from WorldClim: https://worldclim.org/data/cmip6/cmip6climate.html

Format

A SpatRaster object.

Value

No return value. Used with function rast to bring raster variables to analysis.

Examples

```
future_2100_ssp126_access <- terra::rast(system.file("extdata",
                                    "wc2.1_10m_bioc_ACCESS-CM2_ssp126_2081-2100.tif",
                                     package = "kuenm2"))
terra::plot(future_2100_ssp126_access)
```

future_2100_ssp126_miroc

SpatRaster Representing Future Conditions (2081-2100, SSP126, GCM: MIROC6)

Description

A raster layer containing bioclimatic variables representing future climatic conditions (2081-2100) based on the MIROC6 General Circulation Model under the SSP126 scenario. The variables were obtained at a 10 arc-minute resolution and masked using the `m` region provided in the package. Data sourced from WorldClim: https://worldclim.org/data/cmip6/cmip6climate.html

Format

A `SpatRaster` object.

Value

No return value. Used with function `rast` to bring raster variables to analysis.

Examples

```
future_2100_ssp126_miroc <- terra::rast(system.file("extdata",
                                    "wc2.1_10m_bioc_MIROC6_ssp126_2081-2100.tif",
                                     package = "kuenm2"))
terra::plot(future_2100_ssp126_miroc)
```

future_2100_ssp585_access

SpatRaster Representing Future Conditions (2081-2100, SSP585, GCM: ACCESS-CM2)

Description

A raster layer containing bioclimatic variables representing future climatic conditions (2081-2100) based on the ACCESS-CM2 General Circulation Model under the SSP585 scenario. The variables were obtained at a 10 arc-minute resolution and masked using the `m` region provided in the package. Data sourced from WorldClim: https://worldclim.org/data/cmip6/cmip6climate.html

Format

A `SpatRaster` object.

Value

No return value. Used with function `rast` to bring raster variables to analysis.

Examples

```
future_2100_ssp585_access <- terra::rast(system.file("extdata",
                                    "wc2.1_10m_bioc_ACCESS-CM2_ssp585_2081-2100.tif",
                                     package = "kuenm2"))
terra::plot(future_2100_ssp585_access)
```

`future_2100_ssp585_miroc` *SpatRaster Representing Future Conditions (2081-2100, SSP585, GCM: MIROC6)*

Description

A raster layer containing bioclimatic variables representing future climatic conditions (2081-2100) based on the MIROC6 General Circulation Model under the SSP585 scenario. The variables were obtained at a 10 arc-minute resolution and masked using the `m` region provided in the package. Data sourced from WorldClim: https://worldclim.org/data/cmip6/cmip6climate.html

Format

A `SpatRaster` object.

Value

No return value. Used with function `rast` to bring raster variables to analysis.

Examples

```
future_2100_ssp585_miroc <- terra::rast(system.file("extdata",
                                    "wc2.1_10m_bioc_MIROC6_ssp585_2081-2100.tif",
                                     package = "kuenm2"))
terra::plot(future_2100_ssp585_miroc)
```

`glm_mx` *Maxent-like Generalized Linear Models (GLM)*

Description

This function fits a Generalized Linear Model (GLM) to binary presence-background data. It allows for the specification of custom weights, with a default in which presences have a weight of 1 and background 100.

Usage

```
glm_mx(formula, family = binomial(link = "cloglog"), data,
       weights = NULL, ...)
```

Arguments

formula	A formula specifying the model to be fitted, in the format used by glm.
family	A description of the error distribution and link function to be used in the model. Defaults to binomial(link = "cloglog"), which is commonly used for presence-background data.
data	A data.frame containing the variables in the model. Must include a column named pr_bg that indicates whether a record is a presence (1) or background (0), and at least another column with an independent variable (predictor).
weights	Optional. A numeric vector of weights for each observation. If not provided, default weights of 1 for presences and 100 for background are used.
...	Additional arguments to be passed to glm.

Details

For more details about glms using presence and background emulating what Maxent does, see Fithian and Hastie (2013) doi:10.1214/13-AOAS667.

Value

A fitted glm object. The model object includes the minimum and maximum values of the non-factor variables in the dataset, stored as model$varmin and model$varmax.

---

glmnet_mx *Maxent-like glmnet models*

Description

This function fits Maxent-like models using the glmnet package, designed for presence-background data.

Usage

```
glmnet_mx(p, data, f, regmult = 1.0, regfun = maxnet.default.regularization,
          addsamplestobackground = TRUE, weights = NULL, ...)
```

Arguments

p	A vector of binary presence-background labels, where 1 indicates presence and 0 indicates background.
data	A data.frame containing the predictor variables for the model. This must include the same number of rows as the length of p.
f	A formula specifying the model to be fitted, in the format used by model.matrix.
regmult	(numeric) Regularization multiplier, default is 1.0.
regfun	A function that calculates regularization penalties. Default is maxnet.default.regularization.

addsamplestobackground
: (logical) Whether to add presence points not in the background to the background data. Default is TRUE.

weights
: (numeric) A numeric vector of weights for each observation. Default is NULL, which sets weights to 1 for presence points and 100 for background points.

...
: Additional arguments to pass to glmnet.

Details

This function is modified from the package maxnet and fits a Maxent-like model using regularization to avoid over-fitting. Regularization weights are computed using a provided function (which can be changed) and can be multiplied by a regularization multiplier (regmult). The function also includes an option to calculate AIC.

Value

A fitted Maxent-like model object of class glmnet_mx, which includes model coefficients, AIC (if requested), and other elements such as feature mins and maxes, sample means, and entropy.

---

import_results *Import rasters resulting from projection functions*

Description

This function facilitates the import of results that have been generated and written to disk by the project_selected(), projection_changes(), variability_projections(), and projection_mop() functions. Users can select specific periods (past/future), emission scenarios, General Circulation Models (GCMs), and result types for import.

Usage

```
import_results(projection,
               consensus = c("median", "range", "mean", "stdev"),
               present = TRUE, past_period = NULL, past_gcm = NULL,
               future_period = NULL, future_pscen = NULL, future_gcm = NULL,
               change_types = c("summary", "by_gcm", "by_change"),
               mop_types = c("distances", "simple", "basic",
                             "towards_high_combined",
                             "towards_low_combined",
                             "towards_high_end",
                             "towards_low_end"))
```

Arguments

projection	an object of class model_projections, changes_projections, variability_projections, or mop_projections. This object is the direct output from one of the projection functions listed in the description.
consensus	(character) consensus measures to import. Available options are: 'median', 'range', 'mean' and 'stdev' (standard deviation). Default is c("median", "range", "mean", "stdev"), which imports all options. Only applicable if projection is a model_projections object.
present	(logical) whether to import present-day projections. Default is TRUE. Not applicable if projection is a changes_projections object.
past_period	(character) names of specific past periods (e.g., 'LGM' or 'MID') to import. Default is NULL, meaning all available past periods will be imported.
past_gcm	(character) names of specific General Circulation Models (GCMs) from the past to import. Default is NULL, meaning all available past GCMs will be imported.
future_period	(character) names of specific future periods (e.g., '2041-2060' or '2081-2100') to import. Default is NULL, meaning all available future periods will be imported.
future_pscen	(character) names of specific future emission scenarios (e.g., 'ssp126' or 'ssp585') to import. Default is NULL, meaning all available future scenarios will be imported.
future_gcm	(character) names of specific General Circulation Models (GCMs) from the future to import. Default is NULL, meaning all available future GCMs will be imported.
change_types	(character) names of the type of computed changes to import. Available options are: 'summary', 'by_gcm', 'by_change' and 'binarized'. Default is c("summary", "by_gcm", "by_change"), importing all types. Only applicable if projection is a changes_projections object.
mop_types	(character) type(s) of MOP to import. Available options are: 'basic', 'simple', 'towards_high_combined', 'towards_low_combined', towards_high_end', and 'towards_low_end'. Default is NULL, meaning all available MOPs will be imported. Only applicable if projection is a mop_projections object.

Value

A SpatRaster or a list of SpatRasters, structured according to the input projection class:

- If projection is model_projections: A stacked SpatRaster containing all selected projections.
- If projection is changes_projections: A list of SpatRasters, organized by the selected change_types (e.g., 'summary', 'by_gcm', and/or 'by_change').
- If projection is mop_projections: A list of SpatRasters, organized by the selected mop_types (e.g., 'simple' and 'basic').
- If projection is variability_projections: A list of SpatRasters, containing the computed variability.

See Also

prepare_projection(), projection_changes(), projection_variability(), projection_mop()

Examples

```
# Load packages
library(terra)
# Step 1: Organize variables for current projection
## Import current variables (used to fit models)
var <- terra::rast(system.file("extdata", "Current_variables.tif",
                               package = "kuenm2"))

## Create a folder in a temporary directory to copy the variables
out_dir_current <- file.path(tempdir(), "Current_raw2")
dir.create(out_dir_current, recursive = TRUE)

## Save current variables in temporary directory
terra::writeRaster(var, file.path(out_dir_current, "Variables.tif"))


# Step 2: Organize future climate variables (example with WorldClim)
## Directory containing the downloaded future climate variables (example)
in_dir <- system.file("extdata", package = "kuenm2")

## Create a folder in a temporary directory to copy the future variables
out_dir_future <- file.path(tempdir(), "Future_raw2")

## Organize and rename the future climate data (structured by year and GCM)
### 'SoilType' will be appended as a static variable in each scenario
organize_future_worldclim(input_dir = in_dir, output_dir = out_dir_future,
                          name_format = "bio_", static_variables = var$SoilType)

# Step 3: Prepare data to run multiple projections
## An example with maxnet models
## Import example of fitted_models (output of fit_selected())
data(fitted_model_maxnet, package = "kuenm2")

## Prepare projection data using fitted models to check variables
pr <- prepare_projection(models = fitted_model_maxnet,
                         present_dir = out_dir_current,
                         future_dir = out_dir_future,
                         future_period = "2041-2060",
                         future_pscen = c("ssp126", "ssp585"),
                         future_gcm = c("ACCESS-CM2", "MIROC6"),
                         raster_pattern = ".tif*")

# Step 4: Run multiple model projections
## A folder to save projection results
out_dir <- file.path(tempdir(), "Projection_results/maxnet")
dir.create(out_dir, recursive = TRUE)

## Project selected models to multiple scenarios
```

```
p <- project_selected(models = fitted_model_maxnet, projection_data = pr,
                      out_dir = out_dir)

# Use import_results to import results:
raster_p <- import_results(projection = p, consensus = "mean")
plot(raster_p)
```

`independent_evaluation` *Evaluate models with independent data*

Description

This function evaluates the selected models using independent data (i.e., data not used during model calibration). The function computes omission rate and pROC, and optionally assesses whether the environmental conditions in the independent data are analogous (i.e., within the range) to those in the calibration data.

Usage

```
independent_evaluation(fitted_models, new_data,
                               consensus = c("mean", "median"),
                               type = "cloglog", extrapolation_type = "E",
                               var_to_restrict = NULL, perform_mop = TRUE,
                               mop_type = "detailed",
                               calculate_distance = TRUE,
                               where_distance = "all",
                               return_predictions = TRUE,
                               return_binary = TRUE,
                               progress_bar = FALSE, ...)
```

Arguments

`fitted_models`	an object of class `fitted_models` returned by the `fit_selected()` function.
`new_data`	a `data.frame` containing environmental variables for independent test records. The column names must correspond exactly to the environmental variables used to fit the selected models, and each row to an individual test record.
`consensus`	(character) vector specifying the types of consensus to use. Available options are `"median"` and `"mean"`. Default is `c("median", "mean")`.
`type`	(character) the format of prediction values. For `maxnet` models, valid options are `"raw"`, `"cumulative"`, `"logistic"`, and `"cloglog"`. For `glm` models, valid options are `"response"` and `"cloglog"`. Default is `"cloglog"`.
`extrapolation_type`	(character) extrapolation type of model. Models can be transferred with three options: free extrapolation ('E'), extrapolation with clamping ('EC'), and no extrapolation ('NE'). Default = 'E'. See details.

`var_to_restrict`
: (character) vector specifying which variables to clamp or not extrapolate. Only applicable if extrapolation_type is "EC" or "NE". Default is `NULL`, meaning all variables will be clamped or not extrapolated.

`perform_mop`
: (logical) whether to execute a Mobility-Oriented Parity (MOP) analysis. This analysis assesses if the environmental conditions in the `new_data` are analogous (within ranges) to those in the calibration data. Defaults to `TRUE`.

`mop_type`
: (character) type of MOP analysis to be performed. Options available are "basic", "simple" and "detailed". Default is 'simples'. See `projection_mop()` for more details.

`calculate_distance`
: (logical) whether to calculate distances (dissimilarities) between new_data and calibration data. Default is TRUE.

`where_distance`
: (character) specifies which values in `new_data` should be used to calculate distances. Options are: "in_range" (only conditions within the calibration range), "out_range" (only conditions outside the calibration range), and "all" (all conditions). Default is "all".

`return_predictions`
: (logical) whether to return continuous predictions at the locations of independent records in `new_data`. Default is TRUE.

`return_binary`
: (logical) whether to return binary predictions at the locations of independent records in `new_data`. The predictions are binarized using the respective thresholds stores in `fitted_models`. Default is TRUE.

`progress_bar`
: (logical) whether to display a progress bar during mop processing. Default is FALSE.

`...`
: additional arguments passed to `mop()`.

Value

A list containing the following elements:

- **evaluation**: A `data.frame` with omission rate and pROC values for each selected model and for the consensus.
- **mop_results**: (Only if `perform_mop = TRUE`) An object of class `mop_results`, with metrics of environmental similarity between calibration and independent data.
- **predictions**: (Only if `return_predictions = TRUE`) A `list` of `data.frames` containing continuous and binary predictions at the independent record locations, along with MOP distances, an indicator of whether environmental conditions at each location fall within the calibration range, and the identity of the variables that have lower and higher values than the calibration range. If the `fitted_models` object includes categorical variables, the returned `data.frame` will also contain columns indicating which values in `new_data` were not present in the calibration data.

Examples

```
# Example with maxnet
# Import example of fitted_models (output of fit_selected())
```

```
data("fitted_model_maxnet", package = "kuenm2")

# Import independent records to evaluate the models
data("new_occ", package = "kuenm2")

# Import raster layers
var <- terra::rast(system.file("extdata", "Current_variables.tif",
                               package = "kuenm2"))

#Extract variables to occurrences
new_data <- extract_occurrence_variables(occ = new_occ, x = "x", y = "y",
                                         raster_variables = var)

#Add some fake data beyond the limits of calibration ranges
fake_data <- data.frame("pr_bg" = c(1, 1, 1),
                        "x" = c(NA, NA, NA),
                        "y" = c(NA, NA, NA),
                        "bio_1" = c(10, 15, 23),
                        "bio_7" = c(12, 16, 20),
                        "bio_12" = c(2300, 2000, 1000),
                        "bio_15" = c(30, 40, 50),
                        "SoilType" = c(1, 1, 1))
new_data <- rbind(new_data, fake_data)


# Evaluate models with independent data
res_ind <- independent_evaluation(fitted_models = fitted_model_maxnet,
                                  new_data = new_data)
```

`initial_cleaning` *Initial occurrence data cleaning steps*

Description

Simple occurrence data cleaning procedures.

Usage

```
initial_cleaning(data, species, x, y,
                 other_columns = NULL, keep_all_columns = TRUE,
                 sort_columns = TRUE, remove_na = TRUE, remove_empty = TRUE,
                 remove_duplicates = TRUE, by_decimal_precision = FALSE,
                 decimal_precision = 0, longitude_precision = NULL,
                 latitude_precision = NULL)

sort_columns(data, species, x, y, keep_all_columns = FALSE)

remove_missing(data, columns = NULL, remove_na = TRUE,
```

```
                remove_empty = TRUE, keep_all_columns = TRUE)

remove_duplicates(data, columns = NULL, keep_all_columns = TRUE)

remove_corrdinates_00(data, x, y)

filter_decimal_precision(data, x,
                         y, decimal_precision = 0,
                         longitude_precision = NULL,
                         latitude_precision = NULL)
```

Arguments

data	data.frame with occurrence records.
species	(character) name of the column in data containing species name.
x	(character) name of the column in data containing longitude values.
y	(character) name of the column in data containing latitude values.
other_columns	(character) vector of other column name(s) in data to be considered while performing cleaning steps, default = NULL.
keep_all_columns	(logical) whether to keep all columns in data. Default = TRUE.
sort_columns	(logical) whether to sort species, longitude, and latitude columns in data. Default = TRUE.
remove_na	(logical) whether to remove NA values in the columns considered. Default = TRUE.
remove_empty	(logical) whether to remove empty (missing) values in the columns considered. Default = TRUE.
remove_duplicates	(logical) whether to remove duplicates in the columns considered. Default = TRUE.
by_decimal_precision	(logical) whether to remove certain records with coordinate precision lower than that of the following three parameters. Default = FALSE
decimal_precision	(numeric) decimal precision threshold for coordinates. Default = 0. Ignored if the following two parameters are defined.
longitude_precision	(numeric) decimal precision threshold for longitude. Default = NULL.
latitude_precision	(numeric) decimal precision threshold for latitude. Default = NULL.
columns	(character) vector of additional column name(s) in data to be considered while removing missing or duplicate records, default = NULL.

Details

Function initial_cleaning helps to perform all simple steps of data cleaning.

Value

A data.frame with resulting occurrence records.

See Also

advanced_cleaning

Examples

```
# Import occurrences
data(occ_data_noclean, package = "kuenm2")

# remove missing data
mis <- remove_missing(data = occ_data_noclean, columns = NULL, remove_na = TRUE,
                      remove_empty = TRUE)

# remove exact duplicates
mis_dup <- remove_duplicates(data = mis, columns = NULL, keep_all_columns = TRUE)

# remove records with 0 for x and y coordinates
mis_dup_00 <- remove_corrdinates_00(data = mis_dup, x = "x", y = "y")

# remove coordinates with low decimal precision.
mis_dup_00_dec <- filter_decimal_precision(data = mis_dup_00, x = "x", y = "y",
                                           decimal_precision = 2)

# all basic cleaning steps
clean_init <- initial_cleaning(data = occ_data_noclean, species = "species",
                               x = "x", y = "y", remove_na = TRUE,
                               remove_empty = TRUE, remove_duplicates = TRUE,
                               by_decimal_precision = TRUE,
                               decimal_precision = 2)
```

---

kuenm2_discrete_palletes

Discrete palettes based on pals R package

Description

Color palettes designed for discrete, categorical data. Palettes retrieved from pals R package

Usage

```
data("kuenm2_discrete_palletes")
```

Format

A `list` with the following color palettes: "alphabet", "alphabet2", "cols25", "glasbey", "kelly", "polychrome", "stepped", "stepped2", "stepped3", "okabe", "tableau20", "tol", "tol.groundcover", "trubetskoy", and "watlington"

References

Wright K (2023). pals: Color Palettes, Colormaps, and Tools to Evaluate Them_. R package version 1.8, https://CRAN.R-project.org/package=pals.

`m` *SpatVector Representing Calibration Area for* Myrcia hatschbachii

---

Description

A spatial vector defining the calibration area used to extract background points for fitting models of *Myrcia hatschbachii*. The area was generated by creating a minimum convex polygon around presence records (`occ_data`), then applying a 300 km buffer.

Format

A `SpatVector` object.

Value

No return value. Used with function `vect` to bring raster variables to analysis.

Examples

```
m <- terra::vect(system.file("extdata",
                             "m.gpkg",
                              package = "kuenm2"))
terra::plot(m)
```

`new_occ` *Independent Species Occurrence*

Description

A `data.frame` containing the coordinates of 82 occurrences of *Myrcia hatschbachii* (a tree endemic to southern Brazil). The valid occurrences were sourced from NeotropicTree (Oliveira-Filho, 2017) and were used as independent data to test the models fitted with the `occ_data`.

Usage

```
data("new_occ")
```

Format

A `data.frame` with the following columns:

species The species name.

x Longitude.

y Latitude.

References

Oliveira_Filho, A.T. 2017. NeoTropTree, Flora arbórea da Região Neotropical: Um banco de dados envolvendo biogeografia, diversidade e conservação. Universidade Federal de Minas Gerais. (http://www.neotroptree,info).

---

occ_data *Species Occurrence*

Description

A `data.frame` containing the coordinates of 51 valid occurrences of *Myrcia hatschbachii* (a tree endemic to southern Brazil). The valid occurrences were sourced from Trindade & Marques (2024) and contains only the records retrieved from GBIF and SpeciesLink.

Usage

```
data("occ_data")
```

Format

A `data.frame` with the following columns:

species The species name.

x Longitude.

y Latitude.

References

Trindade, W.C.F., Marques, M.C.M., 2023. The Invisible Species: Big Data Unveil Coverage Gaps in the Atlantic Forest Hotspot. Diversity and Distributions 30, e13931. https://doi.org/10.1111/ddi.13931

---

occ_data_noclean *Species Occurrence with Erroneous Records*

Description

A `data.frame` containing the coordinates of 51 valid occurrences of *Myrcia hatschbachii* (a tree endemic to southern Brazil), along with a set of erroneous records used to demonstrate data cleaning procedures. The valid occurrences were sourced from Trindade & Marques (2024).

Usage

```
data("occ_data_noclean")
```

Format

A `data.frame` with the following columns:

species The species name.

x Longitude.

y Latitude.

References

Trindade, W.C.F., Marques, M.C.M., 2023. The Invisible Species: Big Data Unveil Coverage Gaps in the Atlantic Forest Hotspot. Diversity and Distributions 30, e13931. https://doi.org/10.1111/ddi.13931

`organize_for_projection` *Organize and structure variables for past and future projections*

Description

This function helps to organize climate variable files from past and future scenarios into folders categorized by time period ("Past" or "Future"), specific period (e.g., "LGM" or "2081–2100"), emission scenario (e.g., "ssp585"), and GCMs. This structure simplifies the preparation of climate data and ensures compatibility with the `prepare_projection()` function, making the variables properly organized for modeling projections. See **Details** for more information.

Usage

```
organize_for_projection(output_dir, models = NULL, variable_names = NULL,
                        categorical_variables = NULL, present_file = NULL,
                        past_files = NULL, past_period = NULL,
                        past_gcm = NULL, future_files = NULL,
                        future_period = NULL, future_pscen = NULL,
                        future_gcm = NULL, static_variables = NULL,
                        check_extent = TRUE, resample_to_present = TRUE,
                        mask = NULL, overwrite = FALSE)
```

Arguments

`output_dir`	(character) path to the folder where the organized data will be saved.
`models`	an object of class fitted_models returned by the `fit_selected()` function. Default is NULL.
`variable_names`	(character) names of the variables used to fit the model or do the PCA in the `prepare_data()` function. Only applicable if 'models' argument is not provided. Default is NULL.
`categorical_variables`	(character) names of the variables that are categorical. Default is NULL.

`present_file` (character) **full paths** to the variables from the present scenario. Default is NULL.

`past_files` (character) **full paths** to the variables from the past scenario(s). Default is NULL.

`past_period` (character) names of the subfolders within 'past_files', representing specific time periods (e.g., 'LGM' or 'MID'). Only applicable if 'past_files' is provided. Default is NULL.

`past_gcm` (character) names of the subfolders within 'past_files', representing specific General Circulation Models (GCMs). Only applicable if 'past_files' is provided. Default is NULL.

`future_files` (character) **full paths** to the variables from the future scenario(s). Default is NULL.

`future_period` (character) names of the subfolders within 'future_files', representing specific time periods (e.g., '2041-2060' or '2081-2100'). Only applicable if 'future_files' is provided. Default is NULL.

`future_pscen` (character) names of the subfolders within 'future_files', representing specific emission scenarios (e.g., 'ssp126' or 'ssp585'). Only applicable if 'future_files' is provided. Default is NULL.

`future_gcm` (character) names of the subfolders within 'future_files', representing specific General Circulation Models (GCMs). Only applicable if 'future_files' is provided. Default is NULL.

`static_variables` (SpatRaster) optional static variables (i.e., soil type) used in the model, which will remain unchanged in past or future scenarios. This variable will be included with each scenario. Default is NULL.

`check_extent` (logical) whether to ensure that the 'static_variables' have the same spatial extent as the bioclimatic variables. Applicable only if 'static_variables' is provided. Default is TRUE.

`resample_to_present` (logical) whether to resample past or future variables so they match the extent of the present variables. Only used when 'present_file' is provided. Default is TRUE.

`mask` (SpatRaster, SpatVector, or SpatExtent) spatial object used to mask the variables (optional). Default is NULL.

`overwrite` whether to overwrite existing files in the output directory. Default is FALSE.

Details

The listed input rasters must be stored as `.tif` files, with one file per scenario. Filenames should include identifiable patterns for time period, GCM, and (for future scenarios) the emission scenario (SSP).

For example:

- A file representing "Past" conditions for the "LGM" period using the "MIROC6" GCM should be named: `"Past_LGM_MIROC6.tif"`

- A file representing "Future" conditions for the period "2081–2100" under the emission scenario "ssp585" and the GCM "ACCESS-CM2" should be named: `"Future_2081-2100_ssp585_ACCESS-CM2.tif"`

All scenario files must contain the same variable names (e.g., `bio1`, `bio2`, etc.) and units as those used for model calibration with present-day data. Tip: When listing the files, use `list.files(path, full.names = TRUE)` to obtain the full file paths required by the function.

Value

A message indicating that the variables were successfully organized in the 'output_dir' directory.

See Also

prepare_projection organize_future_worldclim

Examples

```
# Set the input directory containing the climate variables.
# In this example, we use present and LGM variables from CHELSA
# located in the "inst/extdata" folder of the package.
present_lgm_dir <- system.file("extdata", package = "kuenm2")

# Define an output directory (here, using a temporary folder)
# Replace with your own working directory if needed.
out_dir <- file.path(tempdir(), "Projection_variables")

# List files for present-day conditions
present_list <- list.files(path = present_lgm_dir,
                       pattern = "Current_CHELSA", # Select only CHELSA present-day files
                           full.names = TRUE)

# List files for LGM conditions
lgm_list <- list.files(path = present_lgm_dir,
                       pattern = "LGM", # Select only LGM files
                       full.names = TRUE)

# Organize variables for projection
organize_for_projection(output_dir = out_dir,
                        variable_names = c("bio1", "bio7", "bio12", "bio15"),
                        present_file = present_list,
                        past_files = lgm_list,
                        past_period = "LGM",
                        past_gcm = c("CCSM4", "CNRM-CM5", "FGOALS-g2",
                                     "IPSL-CM5A-LR", "MIROC-ESM", "MPI-ESM-P",
                                     "MRI-CGCM3"),
                        resample_to_present = TRUE,
                        overwrite = TRUE)
```

`organize_future_worldclim` *Organize and structure future climate variables from WorldClim*

Description

This function imports future climate variables downloaded from WorldClim, renames the files, and organizes them into folders categorized by year, emission scenario (SSP) and General Circulation Model (GCM). It simplifies the preparation of climate data, making it compatible with the `prepare_projection()` function, ensuring that all required variables are properly structured for modeling projections.

Usage

```
organize_future_worldclim(input_dir, output_dir, name_format = "bio_",
                          variables = NULL, static_variables = NULL,
                          check_extent = TRUE, mask = NULL,
                          progress_bar = TRUE, overwrite = FALSE)
```

Arguments

`input_dir`	(character) path to the folder containing the future climate variables downloaded from WorldClim.
`output_dir`	(character) path to the folder where the organized data will be saved.
`name_format`	(character) the format for renaming variable. Options are "bio_", "Bio_", "bio_0", and "Bio_0". See details for more information. Default is "bio_".
`variables`	(character) the names of the variables to retain. Default is NULL, meaning all variables will be kept.
`static_variables`	(SpatRaster) optional static variables (i.e., soil type) used in the model, which will remain unchanged in future scenarios. This variable will be included with each future scenario. Default is NULL.
`check_extent`	(logical) whether to ensure that the `static_variables` have the same spatial extent as the bioclimatic variables. Applicable only if `static_variables` is provided. Default is TRUE.
`mask`	(SpatRaster, SpatVector, or SpatExtent) spatial object used to mask the variables (optional). Default is NULL.
`progress_bar`	(logical) whether to display a progress bar during processing. Default is TRUE.
`overwrite`	whether to overwrite existing files in the output directory. Default is FALSE.

Details

The raw variables downloaded from WorldClim are named as "Bio01", "Bio02", "Bio03", "Bio10", etc. The `name_format` parameter controls how these variables will be renamed:

- "bio_": the variables will be renamed to bio_1, bio_2, bio_3, bio_10, etc.
- "bio_0": the variables will be renamed to bio_01, bio_02, bio_03, bio_10, etc
- "Bio_": the variables will be renamed to Bio_1, Bio_2, Bio_3, Bio_10, etc.
- "Bio_0": the variables will be renamed to Bio_01, Bio_02, Bio_03, Bio_10, etc.

Value

A list of paths to the folders where the organized climate data has been saved.

See Also

prepare_projection()

Examples

```
# Import the current variables used to fit the model.
# In this case, SoilType will be treated as a static variable (constant
# across future scenarios).
var <- terra::rast(system.file("extdata", "Current_variables.tif",
                               package = "kuenm2"))

# Set the input directory containing the raw future climate variables.
# For this example, the data is located in the "inst/extdata" folder.
in_dir <- system.file("extdata", package = "kuenm2")

# Create a "Future_raw" folder in a temporary directory and copy the raw
# variables there.
out_dir <- file.path(tempdir(), "Future_raw")

# Organize and rename the future climate data, structuring it by year and GCM.
# The 'SoilType' variable will be appended as a static variable in each scenario.
# The files will be renamed following the "bio_" format
organize_future_worldclim(input_dir = in_dir, output_dir = out_dir,
                          name_format = "bio_",
                          static_variables = var$SoilType)

# Check files organized
dir(out_dir, recursive = TRUE)
```

---

partial_roc *Partial ROC calculation for multiple candidate models*

Description

Computes partial ROC tests for multiple candidate models.

Usage

```
partial_roc(formula_grid, data, omission_rate = 10,
            addsamplestobackground = TRUE, weights = NULL,
            algorithm = "maxnet", parallel = FALSE, ncores = NULL,
            progress_bar = TRUE)
```

Arguments

formula_grid	a data.frame with the grid of formulas defining the candidate models to test.
data	an object of class prepared_data returned by the prepare_data() function or an object of class calibration_results returned by the calibration() function. It contains the calibration data and k-folds.
omission_rate	(numeric) values from 0 to 100 representing the percentage of potential error due to any source of uncertainty. This value is used to calculate the omission rate. Default is 10. See details.
addsamplestobackground	(logical) whether to add to the background any presence sample that is not already there. Default is TRUE.
weights	(numeric) a numeric vector specifying weights for the occurrence records. Default is NULL.
algorithm	(character) type algorithm, either "glm" or "maxnet". Default is "maxnet".
parallel	(logical) whether to fit the candidate models in parallel. Default is FALSE.
ncores	(numeric) number of cores to use for parallel processing. Default is NULL and uses available cores - 1. This is only applicable if parallel = TRUE.
progress_bar	(logical) whether to display a progress bar during processing. Default is TRUE.

Details

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

Value

A data frame with summary statistics of the and AUC ratios and significance calculated from the replicates of each candidate model. Specifically, it includes the mean and standard deviation of these metrics for each model.

Examples

```
# Import prepared data to get model formulas
data(sp_swd, package = "kuenm2")

# Calculate proc for the first 5 candidate models
res_proc <- partial_roc(formula_grid = sp_swd$formula_grid[1:2,],
                        data = sp_swd, omission_rate = 10,
                        algorithm = "maxnet")
```

`partition_response_curves` *Response curves for selected models according to training/testing partitions*

Description

Variable responses in models selected after model calibration. Responses are based on training partitions and points are testing presence records.

Usage

```
partition_response_curves(calibration_results, modelID, n = 100,
                          averages_from = "pr_bg", col = "darkblue",
                          ylim = NULL, las = 1, parallel = FALSE,
                          ncores = NULL, ...)
```

Arguments

`calibration_results`	an object of class `calibration_results` returned by the `calibration()` function.
`modelID`	(character or numeric) number of the Model (its ID) to be considered for plotting.
`n`	(numeric) an integer guiding the number of breaks. Default = 100
`averages_from`	(character) specifies how the averages or modes of the variables are calculated. Available options are "pr" (to calculate averages from the presence localities) or "pr_bg" (to use the combined set of presence and background localities). Default is "pr_bg". See details.
`col`	(character) color for lines. Default = "darkblue".
`ylim`	(numeric) vector of length two indicating minimum and maximum limits for the y axis. The default, NULL, uses `c(0, 1)`.
`las`	(numeric) the stile of axis tick labels; options are: 0, 1, 2, 3. Default = 1.
`parallel`	(logical) whether to fit the models in parallel. Default is FALSE.
`ncores`	(numeric) number of cores to use for parallel processing. Default is NULL and uses available cores - 1. This is only applicable if `parallel = TRUE`.
`...`	additional arguments passed to `plot`.

Details

Response curves are generated using training portions of the data and points showed are the ones left out for testing. The partition labeled in plot panels indicates the portion left out for testing.

The response curves are generated with all other variables set to their mean values (or mode for categorical variables), calculated either from the presence localities (if averages_from = "pr") or from the combined set of presence and background localities (if averages_from = "pr_bg").

For categorical variables, a bar plot is generated with error bars showing variability across models (if multiple models are included).

Value

A plot with response curves for all `variables` used in the selected model corresponding to `modelID`. Each row in the plot shows response curves produced with training data that leaves out the partition labeled. The points represent the records left out for testing.

See Also

`response_curve()`

Examples

```
# Example with maxnet
# Import example of calibration results
data(calib_results_maxnet, package = "kuenm2")

# Options of models that can be tested
calib_results_maxnet$selected_models$ID

# Response curves
partition_response_curves(calibration_results = calib_results_maxnet,
                          modelID = 192)
```

---

perform_pca *Principal Component Analysis for raster layers*

Description

This function performs principal component analysis (PCA) with a set of raster variables.

Usage

```
perform_pca(raster_variables, exclude_from_pca = NULL, project = FALSE,
            projection_data = NULL, out_dir = NULL, overwrite = FALSE,
            progress_bar = FALSE, center = TRUE, scale = FALSE,
            variance_explained = 95, min_explained = 5)
```

Arguments

`raster_variables`
: (SpatRaster) set of predictor variables that the function will summarize into a set of orthogonal, uncorrelated components based on PCA.

`exclude_from_pca`
: (character) variable names within raster_variables that should not be included in the PCA transformation. Instead, these variables will be added directly to the final set of output variables without being modified. The default is NULL, meaning all variables will be used unless specified otherwise.

`project` (logical) whether the function should project new data from different scenarios (e.g. future variables) onto the PCA coordinates generated by the initial analysis. If TRUE, the argument projection_data needs to be defined. Default is FALSE.

`projection_data` an object of class `prepared_projection` returned by the `prepare_projection()` function. This file contains the paths to the raster files representing each scenario. Only applicable if `project = TRUE`. Default is NULL.

`out_dir` (character) a path to a root directory for saving the raster files of each projection. Default = NULL.

`overwrite` (logical) whether to overwrite SpatRaster if they already exists when projecting. Only applicable if `write_files` is set to TRUE. Default is FALSE.

`progress_bar` (logical) whether to display a progress bar during processing projections. Only applicable if `project = TRUE`. Default is FALSE

`center` (logical) whether the variables should be zero-centered. Default is TRUE.

`scale` (logical) whether the variables should be scaled to have unit variance before the analysis takes place. Default is FALSE.

`variance_explained` (numeric) the cumulative percentage of total variance that must be explained by the selected principal components. Default is 95.

`min_explained` (numeric) the minimum percentage of total variance that a principal component must explain to be retained. Default is 5.

Value

A list containing the following elements:

- env: A SpatRaster object that contains the orthogonal components derived from the PCA. PCs correspond to the variables used to perform the analysis.
- pca: an object of class prcomp, containing the details of the PCA analysis. See `prcomp`().
- variance_explained_cum_sum: The cumulative percentage of total variance explained by each of the selected principal components. This value indicates how much of the data's original variability is captured by the PCA transformation.
- projection_directory: the root directory where projection files were saved. Not NULL only if `project` was set to TRUE. This directory contains the projected raster files for each scenario.

Examples

```
# PCA with current variables
# Import raster layers
var <- terra::rast(system.file("extdata", "Current_variables.tif",
                               package = "kuenm2"))

# PCA
pca_var <- perform_pca(raster_variables = var, exclude_from_pca = "SoilType",
                       center = TRUE, scale = TRUE)

pca_var
```

```
# Project PCA for new scenarios (future)
# First, organize and prepare future variables
# Set the input directory containing the raw future climate variables
# For this example, the data is located in the "inst/extdata" folder.
in_dir <- system.file("extdata", package = "kuenm2")

# Create a "Future_raw" folder in a temporary directory and copy the variables.
out_dir_future <- file.path(tempdir(), "Future_raw1")

# Organize and rename the future climate data, structuring it by year and GCM.
# The 'SoilType' variable will be appended as a static variable in each scenario.
# The files will be renamed following the "bio_" format
organize_future_worldclim(input_dir = in_dir, output_dir = out_dir_future,
                          name_format = "bio_", static_variables = var$SoilType)

# Prepare projections
pr <- prepare_projection(variable_names = c("bio_1", "bio_7", "bio_12",
                                            "bio_15", "SoilType"),
                         future_dir = out_dir_future,
                         future_period = c("2041-2060", "2081-2100"),
                         future_pscen = c("ssp126", "ssp585"),
                         future_gcm = c("ACCESS-CM2", "MIROC6"),
                         raster_pattern = ".tif*")

# Create folder to save projection results
out_dir <- file.path(tempdir(), "PCA_projections")
dir.create(out_dir, recursive = TRUE)

# Perform and project PCA for new scenarios (future)
proj_pca <- perform_pca(raster_variables = var, exclude_from_pca = "SoilType",
                        project = TRUE, projection_data = pr,
                        out_dir = out_dir, center = TRUE, scale = TRUE)

proj_pca$projection_directory  # Directory with projected PCA-variables
```

`plot_calibration_hist` *Histograms to visualize data from explore_calibration objects*

Description

Plots histograms to visualize data from an `explore_calibration` object generated with the `explore_calibration_hist` function.

Usage

```
plot_calibration_hist(explore_calibration, color_m = "grey",
                      color_background = "#56B4E9",
                      color_presence = "#009E73", alpha = 0.4,
                      lines = FALSE, which_lines = c("cl", "mean"),
```

```
                    lty_range = 1, lty_cl = 2, lty_mean = 3,
                    lwd_range = 3, lwd_cl = 2, lwd_mean = 2,
                    xlab = NULL, ylab = NULL, mfrow = NULL)
```

Arguments

explore_calibration	an object of class explore_calibration generated by the explore_calibration_hist function.
color_m	(character) color used to fill the histogram bars for the entire area (M). Default is "grey".
color_background	(character) color used to fill the histogram bars for background data. Default is "#56B4E9".
color_presence	(character) color used to fill the histogram bars for presence data. Default is "#009E73".
alpha	(numeric) opacity factor to fill the bars, typically in the range 0-1. Default is 0.4.
lines	(logical) whether to add vertical lines to the plot representing the range, confidence interval, and mean of variables. Default = FALSE.
which_lines	(character) a vector indicating which lines to plot. Available options are "range", "cl" (confidence interval), and "mean". Default is c("range", "cl", "mean").
lty_range	(numeric) line type for plotting the ranges of variables. Default is 1, meaning a solid line.
lty_cl	(numeric) line type for plotting the confidence interval of variables. Default is 2, meaning a dashed line.
lty_mean	(numeric) line type for plotting the mean of variables. Default is 3, meaning a dotted line.
lwd_range	(numeric) line width for the line representing the range. Default is 3.
lwd_cl	(numeric) line width for the line representing the confidence interval. Default is 2.
lwd_mean	(numeric) line width for the line representing the mean. Default is 2.
xlab	(character) a vector of names for labeling the x-axis. It must have the same length as the number of variables. Default is NULL, meaning the labels will be extracted from the explore_calibration object.
ylab	(character) the label for the y-axis. Default is NULL, meaning the y-axis will be labeled as "Frequency".
mfrow	(numeric) a vector specifying the number of rows and columns in the plot layout, e.g., c(rows, columns). Default is NULL, meaning the grid will be arranged automatically based on the number of plots.

Value

No return value, called for side effects (plots histograms).

Examples

```
# Import raster layers
var <- terra::rast(system.file("extdata", "Current_variables.tif",
                               package = "kuenm2"))

# Import occurrences
data(sp_swd, package = "kuenm2")

# Explore calibration data
calib_hist <- explore_calibration_hist(data = sp_swd,
                                       raster_variables = var,
                                       include_m = TRUE)

# Plot histograms
plot_calibration_hist(explore_calibration = calib_hist)
```

`plot_explore_partition`

Plot extrapolation risks for partitions

Description

Visualize data from an `explore_partition` object generated with the `explore_partition_extrapolation` function.

Usage

```
plot_explore_partition(
  explore_partition,
  space = c("G", "E"),
  type_of_plot = c("distance", "simple"),
  variables = NULL,
  calibration_area = NULL,
  show_limits = TRUE,
  include_background = FALSE,
  distance_palette = NULL,
  break_type = "pretty",
  in_range_color = "#009E73",
  out_range_color = "#D55E00",
  calibration_area_col = "gray90",
  pr_alpha = 1,
  bg_alpha = 0.4,
  pch_in_range = 21,
  pch_out_range = 24,
  cex_plot = 1.4,
  size_text_legend = 1,
  legend.margin = 0.4,
  lwd_legend = 12,
```

```
  ncols = NULL,
  ...
)
```

Arguments

explore_partition	an object of class explore_partition returned by the explore_partition_extrapolation() function.
space	(character) vector specifying the space to plot. Available options are 'G' for geographical space and E' for environmental space. Default is c("G","E"), meaning both spaces are plotted.
type_of_plot	(character) vector specifying the type(s) of plot. Options are "simple", which shows whether the record in a partition is within the range of the other partitions, and "distance", which shows the Euclidean distance of the record to the set of conditions in the other partitions. Default is c("simple", "distance"), meaning both plots are produced.
variables	(character) A pair of variables used to define the axes of the environmental space. Default is NULL, meaning the first two continuous variables available in explore_partition are used to define the E space.
calibration_area	(SpatRaster, SpatVector, or SpatExtent) A spatial object representing the calibration area. Preferably, one of the raster layers used as variables to prepare_data. Required only when type_of_plot = "G". The default, NULL, uses a basic SpatVector of the world.
show_limits	(logical) whether to draw a box representing the lower and upper limits of the variables, considering the other partitions (i.e., in Partition 1, the box represents the limits considering Partitions 2, 3, and 4. Only applicable when "E" is included in type_of_plot. Default is TRUE.
include_background	(logical) whether to plot background points together with presence records. Only applicable if explore_partition was obtained using presence and background points (i.e., with include_test_background = TRUE in explore_partition_extrapolation). Default is FALSE.
distance_palette	(character) a vector of valid colors used to interpolate a palette for representing distance. Default is NULL, meaning a built-in palette is used (green for lower distances and red for higher distances). Only applicable if "distance" is included in type_of_plot.
break_type	(character) specifies the method used to define distance breaks. Options are "pretty" or "quantile". Default is "pretty", which uses the pretty() function to set the breaks. Only applicable if "distance" is included in type_of_plot.
in_range_color	(character) a color used to represent records that fall within the range of the other partitions. Default is "#009E73" (Seafoam Green).
out_range_color	(character) A color used to represent records that fall outside the range of the other partitions. Default is "#D55E00" (reddish-orange).

`calibration_area_col`
: (character) A color used to represent the calibration area. Default is "gray90".

`pr_alpha`
: (numeric) specifies the transparency of presence records. Default is 1, meaning fully opaque.

`bg_alpha`
: (numeric) specifies the transparency of background points. Default is 0.4. Only applicable if `include_background` is set to TRUE.

`pch_in_range`
: (numeric) specifies the symbol used for points that fall within the range of the other partitions. Default is 21 (filled circle). See `?pch` for other available options.

`pch_out_range`
: (numeric) specifies the symbol used for points that fall outside the range of the other partitions. Default is 24 (filled triangle). See `?pch` for other available options.

`cex_plot`
: (numeric) specifies the size of points in the plot. Default is 1.4

`size_text_legend`
: (numeric) specifies the size of the text in the legend. Default is 1.

`legend.margin`
: (numeric) specifies the height of the row in the layout that contains the legend. Default is 0.4, meaning the row will be 40% the height of the other rows in the layout.

`lwd_legend`
: (numeric) specifies the width of the legend bar representing distance. Default is 12. Applicable only if "distance" is included in `type_of_plot`.

`ncols`
: (numeric) specifies the number of columns in the plot layout. Default is NULL, meaning the number of columns is determined automatically based on the number of partitions.

`...`
: additional arguments passed to `plot()`.

Value

No return value, called for side effects (plots the partitions in G or E space).

Examples

```
# Load prepared_data with spatial blocks as the partitioning method (from ENMeval)
data(swd_spatial_block, package = "kuenm2")
# Analyze extrapolation risks across partitions
res <- explore_partition_extrapolation(data = swd_spatial_block,
                                       include_test_background = TRUE)
# Plot partition distribution in Geographic Space (Distance and Simple MOP)
plot_explore_partition(explore_partition = res, space = "G",
                       variables = c("bio_7", "bio_15"))

# Plot partition distribution in Environmental Space (Distance and Simple MOP)
plot_explore_partition(explore_partition = res, space = "E",
                       variables = c("bio_7", "bio_15"))
```

plot_importance *Summary plot for variable importance in models*

Description

See details in plot_importance

Usage

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

Arguments

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

Value

No return value, called for side effects (a barplot or boxplot depending on the number of models considered..

predict *Predict method for glmnet_mx (maxnet) models*

Description

Predict method for glmnet_mx (maxnet) models

Usage

```
predict.glmnet_mx(object, newdata, clamp = FALSE,
                  type = c("link", "exponential", "cloglog", "logistic",
                  "cumulative"))
```

Arguments

object	a glmnet_mx object.
newdata	data to predict on.
clamp	(logical) whether to clamp predictions. Default = FALSE.
type	(character) type of prediction to be performed. Options are: "link", "exponential", "cloglog", "logistic", and cumulative. Defaults to "link" if not defined.

Value

A glmnet_mx (maxnet) prediction.

---

predict_selected *Predict selected models for a single scenario*

Description

This function predicts selected models for a single set of new data using either `maxnet` or `glm` It provides options to save the output and compute consensus results (mean, median, etc.) across replicates and models.

Usage

```
predict_selected(models, new_variables, mask = NULL, write_files = FALSE,
                 write_replicates = FALSE, out_dir = NULL,
                 consensus_per_model = TRUE, consensus_general = TRUE,
                 consensus = c("median", "range", "mean", "stdev"),
                 extrapolation_type = "E", var_to_restrict = NULL,
                 type = "cloglog", overwrite = FALSE, progress_bar = TRUE)
```

Arguments

models	an object of class `fitted_models` returned by the `fit_selected()` function.
new_variables	a SpatRaster or data.frame of predictor variables. The names of these variables must match those used to calibrate the models or those used to run PCA if `do_pca = TRUE` in the `prepare_data()` function.
mask	(SpatRaster, SpatVector, or SpatExtent) spatial object used to mask the variables before predict. Default is NULL.
write_files	(logical) whether to save the predictions (SpatRasters or data.frame) to disk. Default is FALSE.
write_replicates	(logical) whether to save the predictions for each replicates to disk. Only applicable if `write_files` is TRUE. Default is FALSE.
out_dir	(character) directory path where predictions will be saved. Only relevant if `write_files = TRUE`.

consensus_per_model
: (logical) whether to compute consensus (mean, median, etc.) for each model across its replicates. Default is TRUE.

consensus_general
: (logical) whether to compute a general consensus across all models. Default is TRUE.

consensus
: (character) vector specifying the types of consensus to calculate across replicates and models. Available options are "median", "range", "mean", and "stdev" (standard deviation). Default is c("median", "range", "mean", "stdev").

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

var_to_restrict
: (character) vector specifying which variables to clamp or not to extrapolate for. Only applicable if extrapolation_type is "EC" or "NE". Default is NULL, clamping and no extrapolation will be done for all variables.

type
: (character) the format of prediction values. For maxnet models, valid options are "raw", "cumulative", "logistic", and "cloglog". For glm models, valid options are "cloglog", "response", "raw", "cumulative" and "link". Default is "cloglog.

overwrite
: (logical) whether to overwrite SpatRasters if they already exist. Only applicable if write_files = TRUE. Default is FALSE.

progress_bar
: (logical) whether to display a progress bar during processing. Default is TRUE.

Details

When predicting to areas where the variables are beyond the lower or upper limits of the calibration data, users can choose to free extrapolate the predictions (extrapolation_type = "E"), extrapolate with clamping (extrapolation_type = "EC"), or not extrapolate (extrapolation_type = "NE"). When clamping, the variables are set to minimum and maximum values established for the maximum and minimum values within calibration data. In the no extrapolation approach, any cell with at least one variable listed in var_to_restrict falling outside the calibration range is assigned a suitability value of 0.

Value

A list containing SpatRaster or data.frames predictions for each replicate, long with the consensus results for each model and the overall general consensus.

Examples

```
# Import variables to predict on
var <- terra::rast(system.file("extdata", "Current_variables.tif",
                               package = "kuenm2"))

# Example with maxnet
# Import example of fitted_models (output of fit_selected())
```

```
data("fitted_model_maxnet", package = "kuenm2")

# Predict to single scenario
p <- predict_selected(models = fitted_model_maxnet, new_variables = var)

# Example with GLMs
# Import example of fitted_models (output of fit_selected()) without replicates
data("fitted_model_glm", package = "kuenm2")

# Predict to single scenario
p_glm <- predict_selected(models = fitted_model_glm, new_variables = var)

# Plot predictions
terra::plot(c(p$General_consensus$median, p_glm$General_consensus),
            col = rev(terrain.colors(240)), main = c("MAXNET", "GLM"),
            zlim = c(0, 1))
```

`prediction_changes` *Compute changes of suitable areas in other scenarios (single scenario / GCM)*

Description

Compute changes of suitable areas in other scenarios (single scenario / GCM)

Usage

```
prediction_changes(current_predictions, new_predictions,
                   predicted_to = "future", fitted_models = NULL,
                   consensus = "mean", user_threshold = NULL,
                   force_resample = FALSE, gain_color = "#009E73",
                   loss_color = "#D55E00", stable_suitable = "#0072B2",
                   stable_unsuitable = "grey", write_results = FALSE,
                   output_dir = NULL, overwrite = FALSE,
                   write_bin_models = FALSE)
```

Arguments

`current_predictions`
: (SpatRaster) A `SpatRaster` object returned by `predict_selected()` with suitability predicted under current conditions.

`new_predictions`
: (SpatRaster) A `SpatRaster` object returned by `predict_selected()` with suitability predicted under new conditions (past or future). This SpatRaster must have the same resolution and extent as `current_predictions`.

`predicted_to`
: (character) a string specifying whether `new_predictions` represent "past" or "future" conditions. Default is "future".

`fitted_models`
: an object of class `fitted_models` returned by `fit_selected()`

`consensus` (character) the consensus metric stored in `fitted_models` used to binarize models. Available options are `"mean"`, `"median"`, `"range"`, and `"stdev"` (standard deviation). Default is `"mean"`.

`user_threshold` (numeric) an optional threshold for binarizing predictions. Default is `NULL`, meaning the function will apply the thresholds stored in `model_projections`, which were calculated earlier using the omission rate from `calibration()`.

`force_resample` (logical) whether to force rasters to have the same extent and resolution. Default is `TRUE`.

`gain_color` (character) color used to represent gains. Default is "#009E73" (teal green).

`loss_color` (character) color used to represent losses. Default is "#D55E00" (orange-red).

`stable_suitable` (character) color used for representing areas that remain suitable across scenarios. Default is "#0072B2" (oxford blue).

`stable_unsuitable` (character) color used for representing areas that remain unsuitable across scenarios. Default is "grey".

`write_results` (logical) whether to save the results to disk. Default is FALSE.

`output_dir` (character) directory path where results will be saved. Only relevant if `write_results = TRUE`.

`overwrite` (logical) whether to overwrite SpatRasters if they already exist. Only applicable if `write_results = TRUE`. Default is FALSE.

`write_bin_models` (logical) whether to write the binarized models for each scenario to the disk. Only applicable if `write_results = TRUE`. Default is FALSE.

Details

When projecting a niche model to different temporal scenarios (past or future), species' areas can be classified into three categories relative to the current baseline: **gain**, **loss** and **stability**. The interpretation of these categories depends on the temporal direction of the projection. **When projecting to future scenarios**:

- *Gain*: Areas that are currently unsuitable become suitable in the future.
- *Loss*: Areas that are currently suitable become unsuitable in the future.
- *Stability*: Areas that retain their current classification in the future, whether suitable or unsuitable.

When projecting to past scenarios:

- *Gain*: Areas that were unsuitable in the past are now suitable in the present.
- *Loss*: Areas that were suitable in the past are now unsuitable in the present.
- *Stability*: Areas that retain their past classification in the present, whether suitable or unsuitable.

Value

A `SpatRaster` showing the areas of **gain**, **loss** and **stability**.

Examples

```
# Import an example of fitted models (output of fit_selected())
data("fitted_model_maxnet", package = "kuenm2")


# Import current variables for prediction
present_var <- terra::rast(system.file("extdata", "Current_variables.tif",
                                       package = "kuenm2"))


# Import variables for a single future scenario for prediction
future_var <- terra::rast(system.file("extdata",
                                      "wc2.1_10m_bioc_ACCESS-CM2_ssp585_2081-2100.tif",
                                      package = "kuenm2"))

# Rename variables to match the variable names used in the fitted models
names(future_var) <- sub("bio0", "bio", names(future_var))
names(future_var) <- sub("bio", "bio_", names(future_var))

# Append the static soil variable to the future variables
future_var <- c(future_var, present_var$SoilType)

# Predict under present and future conditions
p_present <- predict_selected(models = fitted_model_maxnet,
                              new_variables = present_var)
p_future <- predict_selected(models = fitted_model_maxnet,
                             new_variables = future_var)


# Compute changes between scenarios
p_changes <- prediction_changes(current_predictions = p_present$General_consensus$mean,
                                new_predictions = p_future$General_consensus$mean,
                                fitted_models = fitted_model_maxnet,
                                predicted_to = "future")
# Plot result
terra::plot(p_changes)
```

---

prepare_data *Prepare data for model calibration*

Description

This function prepares data for model calibration, including optional PCA, background point generation, training/testing partitioning, and the creation of a grid of parameter combinations, including regularization multiplier values, feature classes, and sets of environmental variables.

Usage

```
prepare_data(algorithm, occ, x, y, raster_variables, species = NULL,
```

```
n_background = 1000, features = c("lq", "lqp"),
r_multiplier = c(0.1, 0.5, 1, 2, 3),
user_formulas = NULL,
partition_method = "kfolds",
n_partitions = 4, train_proportion = 0.7,
categorical_variables = NULL,
do_pca = FALSE, center = TRUE, scale = TRUE,
exclude_from_pca = NULL, variance_explained = 95,
min_explained = 5, min_number = 2, min_continuous = NULL,
bias_file = NULL, bias_effect = NULL, weights = NULL,
include_xy = TRUE, write_pca = FALSE, pca_directory = NULL,
write_file = FALSE, file_name = NULL, seed = 1)
```

Arguments

algorithm	(character) modeling algorithm, either "glm" or "maxnet".
occ	(data frame) a data.frame containing the coordinates (longitude and latitude) of the occurrence records.
x	(character) a string specifying the name of the column in occ that contains the longitude values.
y	(character) a string specifying the name of the column in occ that contains the latitude values.
raster_variables	(SpatRaster) predictor variables from which environmental values will be extracted using occ and a background will be sampled. Must correspond geographically with the area where model is calibrated.
species	(character) string specifying the species name (optional). Default is NULL.
n_background	(numeric) number of points to represent the background for the model. Default is 1000.
features	(character) a vector of feature classes. Default is c("q", "lq", "lp", "qp", "lqp").
r_multiplier	(numeric) a vector of regularization parameters for maxnet. Default is c(0.1, 1, 2, 3, 5).
user_formulas	(character) Optional character vector with custom formulas provided by the user. See Details. Default is NULL.
partition_method	(character) method used for data partitioning. Available options are "kfolds", "subsample", and "bootstrap". See **Details** for more information. Default = "kfolds".
n_partitions	(numeric) number of partitions to generate. If partition_method is "subsample" or "bootstrap", this defines the number of training testing replicates. If "kfolds", it specifies the number of folds. Must be > 1; default = 4.
train_proportion	(numeric) proportion of occurrence and background points to be used for model training in each partition. Only applicable when partition_method is "subsample" or "bootstrap". Default is 0.7 (i.e., 70% for training and 30% for testing).

`categorical_variables`
: (character) names of the variables that are categorical. Default is NULL.

`do_pca`
: (logical) whether to perform a principal component analysis (PCA) with the set of variables. Default is FALSE.

`center`
: (logical) whether the variables should be zero-centered. Default is TRUE.

`scale`
: (logical) whether the variables should be scaled to have unit variance before the analysis takes place. Default is FALSE.

`exclude_from_pca`
: (character) variable names within raster_variables that should not be included in the PCA transformation. Instead, these variables will be added directly to the final set of output variables without being modified. The default is NULL, meaning all variables will be used unless specified otherwise.

`variance_explained`
: (numeric) the cumulative percentage of total variance that must be explained by the selected principal components. Default is 95.

`min_explained`
: (numeric) the minimum percentage of total variance that a principal component must explain to be retained. Default is 5.

`min_number`
: (numeric) the minimum number of variables to be included in model formulas to be generated. Default = 2.

`min_continuous`
: (numeric) the minimum number of continuous variables required in a combination. Default is NULL.

`bias_file`
: (SpatRaster) a raster containing bias values (probability weights) that influence the selection of background points. It must have the same extent, resolution, and number of cells as the raster variables. Default is NULL.

`bias_effect`
: (character) a string specifying how the values in the `bias_file` should be interpreted. Options are "direct" or "inverse". If "direct", higher values in bias file increase the likelihood of selecting background points. If "inverse", higher values decrease the likelihood. Default = NULL. Must be defined if `bias_file` is provided.

`weights`
: (numeric) a numeric vector specifying weights for the occurrence records. The default, NULL, uses 1 for presence and 100 for background.

`include_xy`
: (logical) whether to include the coordinates (longitude and latitude) in the results from preparing data. Columns containing coordinates will be renamed as "x" and "y". Default is TRUE.

`write_pca`
: (logical) whether to save the PCA-derived raster layers (principal components) to disk. Default is FALSE.

`pca_directory`
: (character) the path or name of the folder where the PC raster layers will be saved. This is only applicable if `write_pca = TRUE`. Default is NULL.

`write_file`
: (logical) whether to write the resulting prepared_data list in a local directory. Default is FALSE.

`file_name`
: (character) name of file (no extension needed) to write resulting object in a local directory. Only needed if `write_file = TRUE`. Default is NULL.

`seed`
: (numeric) integer value to specify an initial seed to split the data and extract background. Default is 1.

Details

Training and testing are performed multiple times (i.e., the number set in `n_partitions`), and model selection is based on the average performance of models after running this routine. A description of the available data partitioning methods is below:

- **"kfolds"**: Splits the dataset into K subsets (folds) of approximately equal size, keeping proportion of 0 and 1 stable compared to the full set. In each training/test run, one fold is used as the test set, while the remaining folds are combined to form the training set.
- **"bootstrap"**: Creates the training dataset by sampling observations from the original dataset *with replacement* (i.e., the same observation can be selected multiple times). The test set consists of the observations that were not selected in that specific sampling.
- **"subsample"**: Similar to bootstrap, but the training set is created by sampling *without replacement* (i.e., each observation is selected at most once).

`user_formulas` must be a character vector of model formulas. Supported terms include linear effects, quadratic terms (e.g., `I(bio_7^2)`), products (e.g., `bio_1:bio_7`), hinge (e.g., `hinge(bio_1)`), threshold (e.g., `thresholds(bio_2)`), and categorical predictors (e.g., `categorical(SoilType)`). Example of a valid formula: `~ bio_1 + bio_7 + I(bio_7^2) + bio_1:bio_7 + hinge(bio_1) + thresholds(bio_2) + categorical(SoilType)`. All variables appearing in the formulas must exist in the raster supplied as `raster_variables`.

Value

An object of class `prepared_data` containing all elements necessary to perform further explorations of data and run a model calibration routine.

See Also

`calibration()`, `explore_calibration_hist()`, `explore_partition_env()`, `explore_partition_geo()`, `explore_partition_extrapolation()`, `plot_calibration_hist()`, `plot_explore_partition()`

Examples

```
# Import occurrences
data(occ_data, package = "kuenm2")

# Import raster layers
var <- terra::rast(system.file("extdata", "Current_variables.tif",
                               package = "kuenm2"))

# Import a bias file
bias <- terra::rast(system.file("extdata", "bias_file.tif",
                                package = "kuenm2"))

# Prepare data for maxnet model
sp_swd <- prepare_data(algorithm = "maxnet", occ = occ_data,
                       x = "x", y = "y",
                       raster_variables = var,
                       species = occ_data[1, 1],
                       categorical_variables = "SoilType",
```

```
                        n_background = 500, bias_file = bias,
                        bias_effect = "direct",
                        features = c("l", "q", "p", "lq", "lqp"),
                        r_multiplier = c(0.1, 1, 2, 3, 5))
print(sp_swd)

# Prepare data for glm model
sp_swd_glm <- prepare_data(algorithm = "glm", occ = occ_data,
                           x = "x", y = "y",
                           raster_variables = var,
                           species = occ_data[1, 1],
                           categorical_variables = "SoilType",
                           n_background = 500, bias_file = bias,
                           bias_effect = "direct",
                           features = c("l", "q", "p", "lq", "lqp"))
print(sp_swd_glm)
```

`prepare_projection` *Preparation of data for model projections*

Description

This function prepared data for model projections to multiple scenarios, storing the paths to the rasters representing each scenario.

Usage

```
prepare_projection(models = NULL, variable_names = NULL, present_dir = NULL,
                   past_dir = NULL, past_period = NULL, past_gcm = NULL,
                   future_dir = NULL, future_period = NULL,
                   future_pscen = NULL, future_gcm = NULL,
                   write_file = FALSE, filename = NULL,
                   raster_pattern = ".tif*")
```

Arguments

`models`	an object of class `fitted_models` returned by the `fit_selected()` function. Default is NULL.
`variable_names`	(character) names of the variables used to fit the model or do the PCA in the `prepare_data()` function. Only applicable if `models` argument is not provided. Default is NULL.
`present_dir`	(character) path to the folder containing variables that represent the current scenario for projection. Default is NULL.
`past_dir`	(character) path to the folder containing subfolders with variables representing past scenarios for projection. Default is NULL.
`past_period`	(character) names of the subfolders within `past_dir`, representing specific time periods (e.g., 'LGM' or 'MID').

past_gcm	(character) names of the subfolders within past_period folders, representing specific General Circulation Models (GCMs).
future_dir	(character) path to the folder containing subfolders with variables representing future scenarios for projection. Default is NULL.
future_period	(character) names of the subfolders within future_dir, representing specific time periods (e.g., '2041-2060' or '2081-2100'). Default is NULL.
future_pscen	(character) names of the subfolders within future_period, representing specific emission scenarios (e.g., 'ssp126' or 'ssp585'). Default is NULL.
future_gcm	(character) names of the subfolders within future_pscen folders, representing specific General Circulation Models (GCMs). Default is NULL.
write_file	(logical) whether to write the object containing the paths to the structured folders. This object is required for projecting models across multiple scenarios using the project_selected() function. Default is FALSE.
filename	(character) the path or name of the folder where the object will be saved. This is only applicable if write_file = TRUE. Default is NULL.
raster_pattern	(character) pattern used to identify the format of raster files within the folders. Default is ".tif*".

Value

An object of class prepared_projection containing the following elements:

- Present, Past, and Future: paths to the variables structured in subfolders.
- Raster_pattern: the pattern used to identify the format of raster files within the folders.
- PCA: if a principal component analysis (PCA) was performed on the set of variables with prepare_data(), a list with class "prcomp" will be returned. See ?stats::prcomp() for details.
- variables: names of the raw predictor variables used to project.

See Also

organize_future_worldclim()

Examples

```
# Import example of fitted_models (output of fit_selected())
data("fitted_model_maxnet", package = "kuenm2")

# Organize and structure future climate variables from WorldClim
# Import the current variables used to fit the model.
# In this case, SoilType will be treated as a static variable (constant
# across future scenarios).
var <- terra::rast(system.file("extdata", "Current_variables.tif",
                               package = "kuenm2"))

# Create a "Current_raw" folder in a temporary directory and copy the raw
# variables there.
```

```
out_dir_current <- file.path(tempdir(), "Current_raw")
dir.create(out_dir_current, recursive = TRUE)

# Save current variables in temporary directory
terra::writeRaster(var, file.path(out_dir_current, "Variables.tif"))

# Set the input directory containing the raw future climate variables.
# For this example, the data is located in the "inst/extdata" folder.
in_dir <- system.file("extdata", package = "kuenm2")

# Create a "Future_raw" folder in a temporary directory and copy the raw
# variables there.
out_dir_future <- file.path(tempdir(), "Future_raw")

# Organize and rename the future climate data, structuring it by year and GCM.
# The 'SoilType' variable will be appended as a static variable in each scenario.
# The files will be renamed following the "bio_" format
organize_future_worldclim(input_dir = in_dir,
                          output_dir = out_dir_future,
                          name_format = "bio_", variables = NULL,
                          static_variables = var$SoilType, mask = NULL,
                          overwrite = TRUE)

# Prepare projections using fitted models to check variables
pr <- prepare_projection(models = fitted_model_maxnet,
                         present_dir = out_dir_current,
                         past_dir = NULL,
                         past_period = NULL,
                         past_gcm = NULL,
                         future_dir = out_dir_future,
                         future_period = c("2041-2060", "2081-2100"),
                         future_pscen = c("ssp126", "ssp585"),
                         future_gcm = c("ACCESS-CM2", "MIROC6"),
                         write_file = FALSE,
                         filename = NULL,
                         raster_pattern = ".tif*")
pr

# Prepare projections using variables names
pr_b <- prepare_projection(models = NULL,
                           variable_names = c("bio_1", "bio_7", "bio_12"),
                           present_dir = out_dir_current,
                           past_dir = NULL,
                           past_period = NULL,
                           past_gcm = NULL,
                           future_dir = out_dir_future,
                           future_period = c("2041-2060", "2081-2100"),
                           future_pscen = c("ssp126", "ssp585"),
                           future_gcm = c("ACCESS-CM2", "MIROC6"),
                           write_file = FALSE,
                           filename = NULL,
                           raster_pattern = ".tif*")
pr_b
```

prepare_user_data *Prepare data for model calibration with user-prepared calibration data*

Description

This function prepares data for model calibration using user-prepared calibration data. It includes optional PCA, training/testing partitioning, and the creation of a grid parameter combinations, including distinct regularization multiplier values, various feature classes, and different sets of environmental variables.

Usage

```
prepare_user_data(algorithm, user_data, pr_bg, species = NULL, x = NULL,
                  y = NULL, features = c("lq", "lqp"),
                  r_multiplier = c(0.1, 0.5, 1, 2, 3),
                  user_formulas = NULL,
                  partition_method = "kfolds", n_partitions = 4,
                  train_proportion = 0.7, user_part = NULL,
                  categorical_variables = NULL,
                  do_pca = FALSE, center = TRUE, scale = TRUE,
                  exclude_from_pca = NULL, variance_explained = 95,
                  min_explained = 5, min_number = 2, min_continuous = NULL,
                  weights = NULL, include_xy = TRUE, write_pca = FALSE,
                  pca_directory = NULL, write_file = FALSE, file_name = NULL,
                  seed = 1)
```

Arguments

algorithm	(character) modeling algorithm, either "glm" or "maxnet".
user_data	(data frame) A data.frame with a column with presence (1) and background (0) records, together with variable values (one variable per column). See an example with `data("user_data", package = "kuenm2")`.
pr_bg	(character) the name of the column in `user_data` that contains the presence/background records.
species	(character) string specifying the species name (optional). Default is NULL.
x	(character) a string specifying the name of the column in `user_data` that contains the longitude values. Default is NULL. Must be defined if present in `user_data` otherwise it will be considered as another predictor variable.
y	(character) a string specifying the name of the column in `user_data` that contains the latitude values. Default is NULL. Must be defined if present in `user_data` otherwise it will be considered as another predictor variable.
features	(character) a vector of feature classes. Default is c("q", "lq", "lp", "qp", "lqp").
r_multiplier	(numeric) a vector of regularization parameters for maxnet. Default is c(0.1, 1, 2, 3, 5).

`user_formulas` (character) Optional character vector with custom formulas provided by the user. See Details. Default is NULL.

`partition_method`
(character) method used for data partitioning. Available options are `"kfolds"`, `"subsample"`, and `"bootstrap"`. See **Details** for more information. Default = "kfolds".

`n_partitions` (numeric) number of partitions to generate. If `partition_method` is `"subsample"` or `"bootstrap"`, this defines the number of training testing replicates. If `"kfolds"`, it specifies the number of folds. Must be > 1; default = 4.

`train_proportion`
(numeric) proportion of occurrence and background points to be used for model training in each partition. Only applicable when `partition_method` is `"subsample"` or `"bootstrap"`. Default is 0.7 (i.e., 70% for training and 30% for testing).

`user_part` a user provided list with partitions or folds for cross-validation to be used in model calibration. Each element of the list should contain a vector of indices indicating the test points, which will be used to split `user_data` into training and testing sets. Useful in experiments that require exactly the same partition sets.

`categorical_variables`
(character) names of the variables that are categorical. Default is NULL.

`do_pca` (logical) whether to perform a principal component analysis (PCA) with the set of variables. Default is FALSE.

`center` (logical) whether the variables should be zero-centered. Default is TRUE.

`scale` (logical) whether the variables should be scaled to have unit variance before the analysis takes place. Default is FALSE.

`exclude_from_pca`
(character) variable names within raster_variables that should not be included in the PCA transformation. Instead, these variables will be added directly to the final set of output variables without being modified. The default is NULL, meaning all variables will be used unless specified otherwise.

`variance_explained`
(numeric) the cumulative percentage of total variance that must be explained by the selected principal components. Default is 95.

`min_explained` (numeric) the minimum percentage of total variance that a principal component must explain to be retained. Default is 5.

`min_number` (numeric) the minimum number of variables to be included in the model formulas to be generated.

`min_continuous` (numeric) the minimum number of continuous variables required in a combination. Default is NULL.

`weights` (numeric) a numeric vector specifying weights for the occurrence records. Default is NULL.

`include_xy` (logical) whether to include the coordinates (longitude and latitude) in the results from preparing data. Default is TRUE.

`write_pca` (logical) whether to save the PCA-derived raster layers (principal components) to disk. Default is FALSE.

pca_directory	(character) the path or name of the folder where the PC raster layers will be saved. This is only applicable if write_pca = TRUE. Default is NULL.
write_file	(logical) whether to write the resulting prepared_data list in a local directory. Default is FALSE.
file_name	(character) the path or name of the folder where the resulting list will be saved. This is only applicable if write_file = TRUE. Default is NULL.
seed	(numeric) integer value to specify an initial seed to split the data. Default is 1.

Details

Training and testing are performed multiple times (i.e., the number set in n_partitions), and model selection is based on the average performance of models after running this routine. A description of the available data partitioning methods is below:

- **"kfolds"**: Splits the dataset into *K* subsets (folds) of approximately equal size, keeping proportion of 0 and 1 stable compared to the full set. In each training/test run, one fold is used as the test set, while the remaining folds are combined to form the training set.
- **"bootstrap"**: Creates the training dataset by sampling observations from the original dataset *with replacement* (i.e., the same observation can be selected multiple times). The test set consists of the observations that were not selected in that specific sampling.
- **"subsample"**: Similar to bootstrap, but the training set is created by sampling *without replacement* (i.e., each observation is selected at most once).

user_formulas must be a character vector of model formulas. Supported terms include linear effects, quadratic terms (e.g., I(bio_7^2)), products (e.g., bio_1:bio_7), hinge (e.g., hinge(bio_1)), threshold (e.g., thresholds(bio_2)), and categorical predictors (e.g., categorical(SoilType)). Example of a valid formula: ~ bio_1 + bio_7 + I(bio_7^2) + bio_1:bio_7 + hinge(bio_1) + thresholds(bio_2) + categorical(SoilType). All variables appearing in the formulas must exist in the data.frame supplied as user_data.

Value

An object of class prepared_data containing all elements necessary to perform further explorations of data and run a model calibration routine.

See Also

calibration(), explore_calibration_hist(), explore_partition_env(), explore_partition_geo(), explore_partition_extrapolation(), plot_calibration_hist(), plot_explore_partition()

Examples

```
# Import user-prepared data
data("user_data", package = "kuenm2")

# Prepare data for maxnet model
maxnet_swd_user <- prepare_user_data(algorithm = "maxnet",
                                     user_data = user_data, pr_bg = "pr_bg",
                                     species = "Myrcia hatschbachii",
```

```
                                    categorical_variables = "SoilType",
                                    features = c("l", "q", "p", "lq", "lqp"),
                                    r_multiplier = c(0.1, 1, 2, 3, 5))
maxnet_swd_user

# Prepare data for glm model
glm_swd_user <- prepare_user_data(algorithm = "glm",
                                  user_data = user_data, pr_bg = "pr_bg",
                                  species = "Myrcia hatschbachii",
                                  categorical_variables = "SoilType",
                                  features = c("l", "q", "p", "lq", "lqp"))
glm_swd_user
```

`print` *Print method for kuenm2 objects*

Description

Print method for kuenm2 objects

Usage

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

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

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

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

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

Arguments

`x`	an object of any of these classes: `prepared_data`, `calibration_results`, `fitted_models`, `projection_data`, or `model_projections`.
`...`	additional arguments affecting the summary produced. Ignored in these functions.

Value

A printed version of the object that summarizes the main elements contained.

`project_selected` *Project selected models to multiple sets of new data (scenarios)*

Description

This function performs predictions of selected models on multiple scenarios, as specified in a `projection_data` object created with the `prepare_projection()` function. In addition to generating predictions for each replicate, the function calculates consensus measures (e.g., mean, median) across replicates and models.

Usage

```
project_selected(models, projection_data, out_dir, mask = NULL,
                 consensus_per_model = TRUE, consensus_general = TRUE,
                 consensus = c("median", "range", "mean", "stdev"),
                 write_replicates = FALSE, extrapolation_type = "E",
                 var_to_restrict = NULL, type = NULL, overwrite = FALSE,
                 parallel = FALSE, ncores = NULL,
                 progress_bar = TRUE, verbose = TRUE)
```

Arguments

`models`	an object of class `fitted_models` returned by the `fit_selected()` function.
`projection_data`	an object of class `projection_data` returned by the `prepare_projection()` function. This file contains the paths to the rasters representing each scenario.
`out_dir`	(character) a path to a root directory for saving the raster file of each projection.
`mask`	(SpatRaster, SpatVector, or SpatExtent) spatial object used to mask the variables before predict. Default is NULL.
`consensus_per_model`	(logical) whether to calculate consensus across replicates when there are more than one replicate per model. Default is TRUE.
`consensus_general`	(logical) whether to calculate consensus across models when there are more than one selected model. Default is TRUE.
`consensus`	(character) consensus measures to calculate. Options available are 'median', 'range', 'mean' and 'stdev' (standard deviation). Default is c("median", "range", "mean", "stdev").
`write_replicates`	(logical) whether to write the projections for each replicate. Default is FALSE.
`extrapolation_type`	(character) extrapolation type of model. Models can be transferred with three options: free extrapolation ('E'), extrapolation with clamping ('EC'), and no extrapolation ('NE'). Default = 'E'. See details.

var_to_restrict
: (character) vector specifying which variables to clamp or not to extrapolate for. Only applicable if extrapolation_type is "EC" or "NE". Default is NULL, clamping and no extrapolation will be done for all variables.

type
: (character) the format of prediction values. For maxnet models, valid options are "raw", "cumulative", "logistic", and "cloglog". For glm models, valid options are "response" and "raw". If NULL (default), the function uses "cloglog" for maxnet models and "response" for glm models.

overwrite
: (logical) whether to overwrite SpatRaster if they already exists. Only applicable if write_files is set to TRUE. Default is FALSE.

parallel
: (logical) whether to fit the candidate models in parallel. Default is FALSE.

ncores
: (numeric) number of cores to use for parallel processing. Default is NULL and uses available cores - 1. This is only applicable if parallel = TRUE.

progress_bar
: (logical) whether to display a progress bar during processing. Default is TRUE.

verbose
: (logical) whether to display messages during processing. Default is TRUE.

Value

A model_projections object that provides the paths to the raster files with the projection results and the corresponding thresholds used to binarize the predictions.

See Also

organize_future_worldclim(), prepare_projection()

Examples

```
# Step 1: Organize variables for current projection
## Import current variables (used to fit models)
var <- terra::rast(system.file("extdata", "Current_variables.tif",
                               package = "kuenm2"))

## Create a folder in a temporary directory to copy the variables
out_dir_current <- file.path(tempdir(), "Current_raw_wc")
dir.create(out_dir_current, recursive = TRUE)

## Save current variables in temporary directory
terra::writeRaster(var, file.path(out_dir_current, "Variables.tif"))


# Step 2: Organize future climate variables (example with WorldClim)
## Directory containing the downloaded future climate variables (example)
in_dir <- system.file("extdata", package = "kuenm2")

## Create a folder in a temporary directory to copy the future variables
out_dir_future <- file.path(tempdir(), "Future_raw_wc")

## Organize and rename the future climate data (structured by year and GCM)
### 'SoilType' will be appended as a static variable in each scenario
```

```
organize_future_worldclim(input_dir = in_dir, output_dir = out_dir_future,
                          name_format = "bio_", static_variables = var$SoilType)

# Step 3: Prepare data to run multiple projections
## An example with maxnet models
## Import example of fitted_models (output of fit_selected())
data(fitted_model_maxnet, package = "kuenm2")

## Prepare projection data using fitted models to check variables
pr <- prepare_projection(models = fitted_model_maxnet,
                         present_dir = out_dir_current,
                         future_dir = out_dir_future,
                         future_period = c("2081-2100"),
                         future_pscen = c("ssp126", "ssp585"),
                         future_gcm = c("ACCESS-CM2", "MIROC6"),
                         raster_pattern = ".tif*")

# Step 4: Run multiple model projections
## A folder to save projection results
out_dir <- file.path(tempdir(), "Projection_results/maxnet_projections")
dir.create(out_dir, recursive = TRUE)

## Project selected models to multiple scenarios
p <- project_selected(models = fitted_model_maxnet, projection_data = pr,
                      out_dir = out_dir)
```

`projection_changes` *Compute changes of suitable areas between scenarios*

Description

This function performs map algebra operations to represent how and where suitable areas change compared to the scenario in which the model was trained. Changes are identified as loss (contraction), gain (expansion) and stability. If multiple climate models (GCM) are used, it calculates the level of agreement among them for each emission scenario.

Usage

```
projection_changes(model_projections, reference_id = 1, consensus = "median",
                   include_id = NULL, user_threshold = NULL, by_gcm = TRUE,
                   by_change = TRUE, general_summary = TRUE,
                   force_resample = TRUE, write_results = TRUE,
                   output_dir = NULL, overwrite = FALSE,
                   write_bin_models = FALSE, return_raster = FALSE)
```

Arguments

`model_projections`
: a `model_projections` object generated by the `project_selected()` function. This object contains the file paths to the raster projection results and the thresholds used for binarization of predictions.

`reference_id` (numeric) the reference ID for the projections corresponding to the current time in `model_projections`. Default is 1. See the details section for further information.

`consensus` (character) the consensus measure to use for calculating changes. Available options are 'mean', 'median', 'range', and 'stdev' (standard deviation). Default is 'median'.

`include_id` (numeric) a vector containing the reference IDs to include when computing changes. Default is `NULL`, meaning all projections will be included. See the details section for further information.

`user_threshold` (numeric) an optional threshold for binarizing the predictions. Default is `NULL`, meaning the function will apply the thresholds stored in model_projections, which were calculated earlier using the omission rate from `calibration()`.

`by_gcm` (logical) whether to compute changes across GCMs. Default is TRUE.

`by_change` (logical) whether to compute results separately for each change, identifying areas of gain, loss, and stability for each GCM. Default is TRUE.

`general_summary` (logical) whether to generate a general summary, mapping how many GCMs project gain, loss, and stability for each scenario. Default is TRUE.

`force_resample` (logical) whether to force the projection rasters to have the same extent and resolution as the raster corresponding to the `reference_id`, which represents the current projections. Default is TRUE.

`write_results` (logical) whether to write the raster files containing the computed changes to the disk. Default is TRUE.

`output_dir` (character) the directory path where the resulting raster files containing the computed changes will be saved. Only relevant if `write_results = TRUE`.

`overwrite` (logical) whether to overwrite SpatRaster if they already exist. Only applicable if `write_results` is set to TRUE. Default is FALSE.

`write_bin_models` (logical) whether to write the binarized models for each GCM to the disk. Default is FALSE.

`return_raster` (logical) whether to return a list containing all the SpatRasters with the computed changes. Default is FALSE, meaning the function will return a NULL object. Setting this argument to TRUE while using multiple GCMs at a large extent and fine resolution may overload the RAM.

Details

When projecting a niche model to different temporal scenarios (past or future), species' areas can be classified into three categories relative to the current baseline: **gain**, **loss** and **stability**. The interpretation of these categories depends on the temporal direction of the projection. **When projecting to future scenarios**:

- *Gain*: Areas that are currently unsuitable become suitable in the future.
- *Loss*: Areas that are currently suitable become unsuitable in the future.
- *Stability*: Areas that retain their current classification in the future, whether suitable or unsuitable.

When projecting to past scenarios:

- *Gain*: Areas that were unsuitable in the past are now suitable in the present.
- *Loss*: Areas that were suitable in the past are now unsuitable in the present.
- *Stability*: Areas that retain their past classification in the present, whether suitable or unsuitable.

The reference scenario (current conditions) can be accessed in the paths element of the model_projections object (model_projections$path). The ID will differ from 1 only if there is more than one projection for the current conditions.

Specific projections can be included or excluded from the analysis using the `include_id` argument. For example, setting 'include_id = c(3, 5, 7)' will compute changes only for scenarios 3, 5, and 7. Conversely, setting 'include_id = -c(3, 5, 7)' will exclude scenarios 3, 5, and 7 from the analysis.

Value

A `changes_projections` object. If return_raster = TRUE, the function returns a list containing the SpatRasters with the computed changes. The list includes the following elements:

- Binarized: binarized models for each GCM.
- Results_by_gcm: computed changes for each GCM.
- Results_by_change: a list where each SpatRaster represents a specific change.
- Summary_changes: A general summary that indicates how many GCMs project gain, loss, and stability for each scenario
- root_directory: the path to the directory where the results were saved if write_results was set to TRUE

If return_raster = FALSE, the function returns a NULL object.

See Also

`organize_future_worldclim()`, `prepare_projection()`, `project_selected()`

Examples

```
# Step 1: Organize variables for current projection
## Import current variables (used to fit models)
var <- terra::rast(system.file("extdata", "Current_variables.tif",
                               package = "kuenm2"))

## Create a folder in a temporary directory to copy the variables
out_dir_current <- file.path(tempdir(), "Current_raw3")
dir.create(out_dir_current, recursive = TRUE)

## Save current variables in temporary directory
terra::writeRaster(var, file.path(out_dir_current, "Variables.tif"))


# Step 2: Organize future climate variables (example with WorldClim)
## Directory containing the downloaded future climate variables (example)
```

```
in_dir <- system.file("extdata", package = "kuenm2")

## Create a folder in a temporary directory to copy the future variables
out_dir_future <- file.path(tempdir(), "Future_raw3")

## Organize and rename the future climate data (structured by year and GCM)
### 'SoilType' will be appended as a static variable in each scenario
organize_future_worldclim(input_dir = in_dir, output_dir = out_dir_future,
                          name_format = "bio_", static_variables = var$SoilType)

# Step 3: Prepare data to run multiple projections
## An example with maxnet models
## Import example of fitted_models (output of fit_selected())
data(fitted_model_maxnet, package = "kuenm2")

## Prepare projection data using fitted models to check variables
pr <- prepare_projection(models = fitted_model_maxnet,
                         present_dir = out_dir_current,
                         future_dir = out_dir_future,
                         future_period = c("2081-2100"),
                         future_pscen = c("ssp585"),
                         future_gcm = c("ACCESS-CM2", "MIROC6"),
                         raster_pattern = ".tif*")

# Step 4: Run multiple model projections
## A folder to save projection results
out_dir <- file.path(tempdir(), "Projection_results/maxnet1")
dir.create(out_dir, recursive = TRUE)

## Project selected models to multiple scenarios
p <- project_selected(models = fitted_model_maxnet, projection_data = pr,
                      out_dir = out_dir)

# Step 5: Identify areas of change in projections
## Contraction, expansion and stability
changes <- projection_changes(model_projections = p, write_results = FALSE,
                              return_raster = TRUE)

terra::plot(changes$Binarized)  # SpatRaster with the binarized predictions
terra::plot(changes$Results_by_gcm)  # SpatRaster with changes by GCM
changes$Results_by_change  # List of SpatRaster(s) by changes with GCM agreement
terra::plot(changes$Results_by_change$`Future_2081-2100_ssp585`) # an example of the previous
terra::plot(changes$Summary_changes)  # SpatRaster with a general summary
```

`projection_mop` *Analysis of extrapolation risks in projections using the MOP metric*

Description

Calculates the mobility-oriented parity metric and other sub-products to represent dissimilarities and non-analogous conditions when comparing a set of reference conditions (M) against model

projection conditions (G).

Usage

```
projection_mop(data, projection_data, out_dir,
               subset_variables = FALSE, mask = NULL, type = "basic",
               na_in_range = TRUE, calculate_distance = FALSE,
               where_distance = "in_range", distance = "euclidean",
               scale = FALSE, center = FALSE, fix_NA = TRUE, percentage = 1,
               comp_each = 2000, tol = NULL, rescale_distance = FALSE,
               parallel = FALSE, ncores = NULL, progress_bar = TRUE,
               overwrite = FALSE)
```

Arguments

data	an object of class fitted_models returned by the fit_selected() function or an object of class prepared_data returned by the prepare_data() function.
projection_data	an object of class projection_data returned by the prepare_projection()function. This file contains the paths to the rasters representing each scenario.
out_dir	(character) a path to a root directory for saving the raster file of each projection.
subset_variables	(logical) whether to include in the analysis only the variables present in the selected models. Only applicable if data is a fitted_models object. Default is FALSE
mask	(SpatRaster, SpatVector, or SpatExtent) spatial object used to mask the variables (optional). Default is NULL.
type	(character) type of MOP analysis to be performed. Options available are "basic", "simple" and "detailed". See Details for further information.
na_in_range	(logical) whether to assign NA to regions within the projected area (G) where environmental conditions fall within the range of the calibration data (M). If TRUE (default), these regions are assigned NA. If FALSE, they are assigned 0 in the simple and basic MOP outputs, and "within ranges" in the detailed MOP output.
calculate_distance	(logical) whether to calculate distances (dissimilarities) between m and g. The default, FALSE, runs rapidly and does not assess dissimilarity levels.
where_distance	(character) where to calculate distances, considering how conditions in g are positioned in comparison to the range of conditions in m. Options available are "in_range", "out_range" and "all". Default is "in_range".
distance	(character) which distances are calculated, euclidean or mahalanobis. Only applicable if calculate_distance = TRUE.
scale	(logical or numeric) whether to scale as in scale. Default is FALSE.
center	(logical or numeric) whether to center as in scale. Default is FALSE.

fix_NA	(logical) whether to fix layers so cells with NA values are the same in all layers. Setting to FALSE may save time if the rasters are big and have no NA matching problems. Default is TRUE.
percentage	(numeric) percentage of m closest conditions used to derive mean environmental distances to each combination of conditions in g.
comp_each	(numeric) number of combinations in g to be used for distance calculations at a time. Increasing this number requires more RAM
tol	(numeric) tolerance to detect linear dependencies when calculating Mahalanobis distances. The default, NULL, uses .Machine$double.eps.
rescale_distance	(logical) whether to re-scale distances 0-1. Re-scaling prevents comparisons of dissimilarity values obtained from runs with different values of percentage.
parallel	(logical) whether to fit the candidate models in parallel. Default is FALSE.
ncores	(numeric) number of cores to use for parallel processing. Default is NULL and uses available cores - 1. This is only applicable if parallel = TRUE.
progress_bar	(logical) whether to display a progress bar during processing. Default is TRUE.
overwrite	(logical) whether to overwrite SpatRaster if they already exists. Only applicable if write_files is set to TRUE. Default is FALSE.

Details

type options return results that differ in the detail of how non-analogous conditions are identified.

- **basic** - makes calculation as proposed by Owens et al. (2013) doi:10.1016/j.ecolmodel.2013.04.011.
- **simple** - calculates how many variables in the set of interest are non-analogous to those in the reference set.
- **detailed** - calculates five additional extrapolation metrics. See mop_detailed under Value below for full details.

where_distance options determine what values should be used to calculate dissimilarity

- **in_range** - only conditions inside m ranges
- **out_range** - only conditions outside m ranges
- **all** - all conditions

When the variables used to represent conditions have different units, scaling and centering are recommended. This step is only valid when Euclidean distances are used.

Value

An object of class mop_projections, with the root directory and the dataframe containing the file paths where the results were stored for each scenario. The paths contain the following files:

- **summary** - a data.frame with details of the data used in the analysis:
  - *variables* - names of variables considered.
  - *type* - type of MOP analysis performed.

  - *scale* - value according to the argument `scale`.
  - *center* - value according to the argument `center`.
  - *calculate_distance* - value according to the argument `calculate_distance`.
  - *distance* - option regarding distance used.
  - *percentage* - percentage of `m` used as reference for distance calculation.
  - *rescale_distance* - value according to the argument `rescale_distance`.
  - *fix_NA* - value according to the argument `fix_NA`.
  - *N_m* - total number of elements (cells with values or valid rows) in `m`.
  - *N_g* - total number of elements (cells with values or valid rows) in `g`.
  - *m_ranges* - the range (minimum and maximum values) of the variable in reference conditions (`m`)

- **mop_distances** - if `calculate_distance` = TRUE, a SpatRaster or vector with distance values for the set of interest (`g`). Higher values represent greater dissimilarity compared to the set of reference (`m`).
- **mop_basic** - a SpatRaster or vector, for the set of interest, representing conditions in which at least one of the variables is non-analogous to the set of reference. Values should be: 1 for non-analogous conditions, and NA for conditions inside the ranges of the reference set.
- **mop_simple** - a SpatRaster or vector, for the set of interest, representing how many variables in the set of interest are non-analogous to those in the reference set. NA is used for conditions inside the ranges of the reference set.
- **mop_detailed** - a list containing:
  - *interpretation_combined* - a data.frame to help identify combinations of variables in *towards_low_combined* and *towards_high_combined* that are non-analogous to `m`.
  - *towards_low_end* - a SpatRaster or matrix for all variables representing where non-analogous conditions were found towards low values of each variable.
  - *towards_high_end* - a SpatRaster or matrix for all variables representing where non-analogous conditions were found towards high values of each variable.
  - *towards_low_combined* - a SpatRaster or vector with values representing the identity of the variables found to have non-analogous conditions towards low values. If vector, interpretation requires the use of the data.frame *interpretation_combined*.
  - *towards_high_combined* - a SpatRaster or vector with values representing the identity of the variables found to have non-analogous conditions towards high values. If vector, interpretation requires the use of the data.frame *interpretation_combined*.

See Also

`organize_future_worldclim()`, `prepare_projection()`

Examples

```
# Step 1: Organize variables for current projection
## Import current variables (used to fit models)
var <- terra::rast(system.file("extdata", "Current_variables.tif",
                               package = "kuenm2"))

## Create a folder in a temporary directory to copy the variables
```

```
out_dir_current <- file.path(tempdir(), "Current_raw4")
dir.create(out_dir_current, recursive = TRUE)

## Save current variables in temporary directory
terra::writeRaster(var, file.path(out_dir_current, "Variables.tif"))


# Step 2: Organize future climate variables (example with WorldClim)
## Directory containing the downloaded future climate variables (example)
in_dir <- system.file("extdata", package = "kuenm2")

## Create a folder in a temporary directory to copy the future variables
out_dir_future <- file.path(tempdir(), "Future_raw4")

## Organize and rename the future climate data (structured by year and GCM)
### 'SoilType' will be appended as a static variable in each scenario
organize_future_worldclim(input_dir = in_dir, output_dir = out_dir_future,
                          name_format = "bio_", static_variables = var$SoilType)

# Step 3: Prepare data to run multiple projections
## An example with maxnet models
## Import example of fitted_models (output of fit_selected())
data(fitted_model_maxnet, package = "kuenm2")

## Prepare projection data using fitted models to check variables
pr <- prepare_projection(models = fitted_model_maxnet,
                         present_dir = out_dir_current,
                         future_dir = out_dir_future,
                         future_period = c("2041-2060", "2081-2100"),
                         future_pscen = c("ssp126", "ssp585"),
                         future_gcm = c("ACCESS-CM2", "MIROC6"),
                         raster_pattern = ".tif*")

# Step 4: Perform MOP for all projection scenarios
## Create a folder to save MOP results
out_dir <- file.path(tempdir(), "MOPresults")
dir.create(out_dir, recursive = TRUE)

## Run MOP
kmop <- projection_mop(data = fitted_model_maxnet, projection_data = pr,
                       out_dir = out_dir, type = "detailed")
```

`projection_variability` *Explores variance coming from distinct sources in model predictions*

Description

Calculates variance in model predictions, distinguishing between the different sources of variation. Potential sources include replicates, model parameterizations, and general circulation models (GCMs).

Usage

```
projection_variability(model_projections, from_replicates = TRUE,
                       from_parameters = TRUE, from_gcms = TRUE,
                       consensus = "median", write_files = FALSE,
                       output_dir = NULL, return_rasters = TRUE,
                       progress_bar = FALSE, verbose = TRUE,
                       overwrite = FALSE)
```

Arguments

model_projections	a model_projections object generated by the project_selected() function. This object contains the file paths to the raster projection results and the thresholds used for binarizing the predictions.
from_replicates	(logical) whether to compute the variance originating from replicates.
from_parameters	(logical) whether to compute the variance originating from model parameterizations.
from_gcms	(logical) whether to compute the variance originating from general circulation models (GCMs)
consensus	(character) (character) the consensus measure to use for calculating changes. Available options are 'mean', 'median', 'range', and 'stdev' (standard deviation). Default is 'median'.
write_files	(logical) whether to write the raster files containing the computed variance to the disk. Default is FALSE.
output_dir	(character) the directory path where the resulting raster files containing the computed changes will be saved. Only relevant if write_results = TRUE.
return_rasters	(logical) whether to return a list containing all the SpatRasters with the computed changes. Default is TRUE. Setting this argument to FALSE returns a NULL object.
progress_bar	(logical) whether to display a progress bar during processing. Default is TRUE.
verbose	(logical) whether to display messages during processing. Default is TRUE.
overwrite	whether to overwrite SpatRaster if they already exists. Only applicable if write_files is set to TRUE. Default is FALSE.

Value

An object of class variability_projections. If return_rasters = TRUE, the function returns a list containing the SpatRasters with the computed variances, categorized by replicate, model, and GCMs. If write_files = TRUE, it also returns the directory path where the computed rasters were saved to disk, and the object can then be used to import these files later with the import_results() function. If both return_rasters = FALSE and write_files = FALSE, the function returns NULL

See Also

organize_future_worldclim(), prepare_projection(), project_selected(), import_results()

Examples

```
# Step 1: Organize variables for current projection
## Import current variables (used to fit models)
var <- terra::rast(system.file("extdata", "Current_variables.tif",
                               package = "kuenm2"))

## Create a folder in a temporary directory to copy the variables
out_dir_current <- file.path(tempdir(), "Current_raw5")
dir.create(out_dir_current, recursive = TRUE)

## Save current variables in temporary directory
terra::writeRaster(var, file.path(out_dir_current, "Variables.tif"))


# Step 2: Organize future climate variables (example with WorldClim)
## Directory containing the downloaded future climate variables (example)
in_dir <- system.file("extdata", package = "kuenm2")

## Create a folder in a temporary directory to copy the future variables
out_dir_future <- file.path(tempdir(), "Future_raw5")

## Organize and rename the future climate data (structured by year and GCM)
### 'SoilType' will be appended as a static variable in each scenario
organize_future_worldclim(input_dir = in_dir, output_dir = out_dir_future,
                          name_format = "bio_", static_variables = var$SoilType)

# Step 3: Prepare data to run multiple projections
## An example with maxnet models
## Import example of fitted_models (output of fit_selected())
data(fitted_model_maxnet, package = "kuenm2")

## Prepare projection data using fitted models to check variables
pr <- prepare_projection(models = fitted_model_maxnet,
                         present_dir = out_dir_current,
                         future_dir = out_dir_future,
                         future_period = "2041-2060",
                         future_pscen = "ssp126",
                         future_gcm = c("ACCESS-CM2", "MIROC6"),
                         raster_pattern = ".tif*")

# Step 4: Run multiple model projections
## A folder to save projection results
out_dir <- file.path(tempdir(), "Projection_results/maxnet3")
dir.create(out_dir, recursive = TRUE)

## Project selected models to multiple scenarios
p <- project_selected(models = fitted_model_maxnet, projection_data = pr,
                      out_dir = out_dir)

# Step 5: Compute variance from distinct sources
v <- projection_variability(model_projections = p, from_replicates = FALSE)
```

```
#terra::plot(v$Present$from_replicates)  # Variance from replicates, present projection
terra::plot(v$Present$from_parameters)  # From models with distinct parameters
#terra::plot(v$`Future_2041-2060_ssp126`$from_replicates) # From replicates in future projection
terra::plot(v$`Future_2041-2060_ssp126`$from_parameters)  # From models
terra::plot(v$`Future_2041-2060_ssp126`$from_gcms)  # From GCMs
```

`response_curve` *Variable response curves for fitted models*

Description

Plot variable responses for fitted models. Responses based on single or multiple models can be plotted.

Usage

```
# Single variable response curves
response_curve(models, variable, modelID = NULL, n = 100,
               show_variability = FALSE, show_lines = FALSE, data = NULL,
               new_data = NULL, averages_from = "pr_bg", extrapolate = TRUE,
               extrapolation_factor = 0.1, add_points = FALSE, p_col = NULL,
               l_limit = NULL, u_limit = NULL, xlab = NULL,
               ylab = "Suitability", col = "darkblue", ...)

# Response curves for all variables in all or individual models
all_response_curves(models, modelID = NULL, n = 100, show_variability = FALSE,
                    show_lines = FALSE, data = NULL, new_data = NULL,
                    averages_from = "pr_bg", extrapolate = TRUE,
                    extrapolation_factor = 0.1, add_points = FALSE,
                    p_col = NULL, l_limit = NULL, u_limit = NULL,
                    xlab = NULL, ylab = "Suitability", col = "darkblue",
                    ylim = NULL, mfrow = NULL, ...)
```

Arguments

`models`	an object of class `fitted_models` returned by the `fit_selected()` function.
`variable`	(character) name of the variable to be plotted.
`modelID`	(character) ModelID(s) to be considered. By default all IDs in `models` are included. Default = NULL.
`n`	(numeric) an integer guiding the number of breaks to produce the curve. Default = 100.
`show_variability`	(logical) if `modelID` is defined, shows variability in response curves considering replicates. If `modelID` is not defined, the default, FALSE, always shows variability from multiple models if present in `models`.

show_lines	(logical) whether to show variability by plotting lines for all models or replicates. The default = FALSE, uses a GAM to characterize a median trend and variation among modes or replicates. Ignored if show_variability = FALSE and modelID is defined.
data	data.frame or matrix of data used in the model calibration step. The default, NULL, uses data stored in models.
new_data	a SpatRaster, data.frame, or matrix with values for variables representing an area or scenario of interest for model projection. Default = NULL.
averages_from	(character) specifies how the averages or modes of the other variables are calculated when producing responses for the variable of interest. Options are "pr" (from the presences) or "pr_bg" (from presences and background). Default is "pr_bg". See details.
extrapolate	(logical) whether to allow extrapolation of the response outside training conditions. Ignored if new_data is defined. Default = TRUE.
extrapolation_factor	(numeric) a value used to calculate how much to expand the training region for extrapolation. Larger values produce extrapolation farther from training limits. Default = 0.1.
add_points	(logical) if TRUE, adds the original observed points (0/1) to the plot. This also sets ylim = c(0, 1), unless these limits are defined as part of .... Default = FALSE.
p_col	(character) color for the observed points when add_points = TRUE. Any valid R color name or hexadecimal code. Default = "black".
l_limit	(numeric) directly specifies the lower limit for the variable. Default = NULL, meaning the lower limit will be calculated from existing data. (if extrapolation = TRUE).
u_limit	(numeric) directly specifies the upper limit for the variable. Default = NULL, meaning the upper limit will be calculated from existing data. (if extrapolation = TRUE).
xlab	(character) a label for the x axis. The default, NULL, uses the name defined in variable.
ylab	(character) a label for the y axis. Default = "Suitability".
col	(character) color for lines. Default = "darkblue".
...	additional arguments passed to plot.
ylim	(numeric) vector of length two with limits for the y axis. Directly used in all_response_curves. Default = NULL.
mfrow	(numeric) a vector specifying the number of rows and columns in the plot layout, e.g., c(rows, columns). Default is NULL, meaning the grid will be arranged automatically based on the number of plots.

Details

The response curve for a variable of interest is generated with all other variables set to their mean values (or mode for categorical variables), calculated either from the presence records (if averages_from = "pr") or the combined set of presence and background records (if averages_from = "pr_bg").

For categorical variables, a bar plot is generated with error bars showing variability across models (if multiple models are included).

Value

For response_curve(), a plot with the response curve for a `variable`. For all_response_curves(), a multipanel plot with response curves fro all variables in `models`.

See Also

`bivariate_response()`, `partition_response_curves()`

Examples

```
# Example with maxnet
# Import example of fitted_models (output of fit_selected())
data(fitted_model_maxnet, package = "kuenm2")

# Response curves for one variable at a time
response_curve(models = fitted_model_maxnet, variable = "bio_1")
response_curve(models = fitted_model_maxnet, variable = "bio_1",
               add_points = TRUE)
response_curve(models = fitted_model_maxnet, variable = "bio_1",
               show_lines = TRUE)

response_curve(models = fitted_model_maxnet, variable = "bio_1",
               modelID = "Model_192", show_variability = TRUE)
response_curve(models = fitted_model_maxnet, variable = "bio_1",
               modelID = "Model_192", show_variability = TRUE,
               show_lines = TRUE)

# Example with maxnet
# Import example of fitted_models (output of fit_selected())
data(fitted_model_maxnet, package = "kuenm2")

# Response curves for all variables at once
all_response_curves(fitted_model_maxnet)
all_response_curves(fitted_model_maxnet, show_lines = TRUE)
all_response_curves(fitted_model_maxnet, show_lines = TRUE,
                    add_points = TRUE)

all_response_curves(fitted_model_maxnet, modelID = "Model_192",
                    show_variability = TRUE, show_lines = TRUE)
all_response_curves(fitted_model_maxnet, modelID = "Model_192",
                    show_variability = TRUE, show_lines = TRUE,
                    add_points = TRUE)
```

`select_models` *Select models that perform the best among candidates*

Description

This function selects the best models according to user-defined criteria, evaluating statistical significance (partial ROC), predictive ability (omission rates), and model complexity (AIC).

Usage

```
select_models(calibration_results = NULL, candidate_models = NULL, data = NULL,
              algorithm = NULL, compute_proc = FALSE,
              addsamplestobackground = TRUE, weights = NULL,
              remove_concave = FALSE, omission_rate = NULL,
              allow_tolerance = TRUE, tolerance = 0.01,
              significance = 0.05, delta_aic = 2, parallel = FALSE,
              ncores = NULL, progress_bar = FALSE,verbose = TRUE)
```

Arguments

`calibration_results`
: an object of class `calibration_results` returned by the `calibration()` function. Default is NULL.

`candidate_models`
: (data.frame) a summary of the evaluation metrics for each candidate model. Required only if `calibration_results` is NULL. In the output of the `calibration()`, this data.frame is located in `$calibration_results$Summary`. Default is NULL.

`data`
: an object of class `prepared_data` returned by the `prepare_data()` function. Required only if `calibration_results` is NULL and `compute_proc` is TRUE.

`algorithm`
: (character) model algorithm, either "glm" or "maxnet". The default, NULL, uses the one defined as part of `calibration_results`, or `data`. If those arguments are not used, `algorithm` must be defined.

`compute_proc`
: (logical) whether to compute partial ROC tests for the selected models. This is required when partial ROC is not calculated for all candidate models during calibration. Default is FALSE.

`addsamplestobackground`
: (logical) whether to add to the background any presence sample that is not already there. Required only if `compute_proc` is TRUE and `calibration_results` is NULL.Default is TRUE.

`weights`
: (numeric) a numeric vector specifying weights for the occurrence records. Required only if `compute_proc` is TRUE and `calibration_results` is NULL. Default is NULL.

`remove_concave`
: (logical) whether to remove candidate models presenting concave curves. Default is FALSE.

`omission_rate` (numeric) the maximum omission rate a candidate model can have to be considered as a potentially selected model. The default, NULL, uses the value provided as part of `calibration_results`. For purposes of selection in existing results of evaluation, this value must match one of the values used in omission tests, and must be manually defined.

`allow_tolerance` (logical) whether to allow selection of models with minimum values of omission rates even if their omission rate surpasses the `omission_rate`. This is only applicable if all candidate models have omission rates higher than the `omission_rate`. Default is TRUE.

`tolerance` (numeric) The value added to the minimum omission rate if it exceeds the `omission_rate`. If `allow_tolerance = TRUE`, selected models will have an omission rate equal to or less than the minimum rate plus this tolerance. Default is 0.01.

`significance` (numeric) the significance level to select models based on the partial ROC (pROC). Default is 0.05. See Details.

`delta_aic` (numeric) the value of delta AIC used as a threshold to select models. Default is 2.

`parallel` (logical) whether to calculate the PROC of the candidate models in parallel. Default is FALSE.

`ncores` (numeric) number of cores to use for parallel processing. Default is NULL and uses available cores - 1. This is only applicable if `parallel = TRUE`.

`progress_bar` (logical) whether to display a progress bar during processing. Default is TRUE.

`verbose` (logical) whether to display messages during processing. Default is TRUE.

Details

Partial ROC is calculated following Peterson et al. (2008).

Value

If calibration_results is provided, it returns a new calibration_results with the new selected models and summary. If calibration_results is NULL, it returns a list containing the following elements:

- selected_models: data frame with the ID and the summary of evaluation metrics for the selected models.
- summary: A list containing the delta AIC values for model selection, and the ID values of models that failed to fit, had concave curves, non-significant pROC values, omission rates above the threshold, delta AIC values above the threshold, and the selected models.

Examples

```
# Import example of calibration results (output of calibration function)
## GLM
data(calib_results_glm, package = "kuenm2")

#Select new best models based on another value of omission rate
```

```
new_best_model <- select_models(calibration_results = calib_results_glm,
                                algorithm = "glm", compute_proc = TRUE,
                                omission_rate = 10)  # Omission error of 10

# Compare with best models selected previously
calib_results_glm$summary$Selected  # Model 86 selected
new_best_model$summary$Selected  # Models 64, 73 and 86 selected
```

`single_mop` *Analysis of extrapolation risks using the MOP metric (for single scenario)*

Description

Calculates the mobility-oriented parity metric and other sub-products to represent dissimilarities and non-analogous conditions when comparing a set of reference conditions (M) against another set of scenario conditions (G).

Usage

```
single_mop(data, new_variables, subset_variables = FALSE,
           mask = NULL, type = "basic", na_in_range = TRUE,
           calculate_distance = FALSE, where_distance = "in_range",
           distance = "euclidean", scale = FALSE, center = FALSE,
           fix_NA = TRUE, percentage = 1, comp_each = 2000, tol = NULL,
           rescale_distance = FALSE, parallel = FALSE, ncores = NULL,
           progress_bar = TRUE, write_files = FALSE, out_dir = NULL,
           overwrite = FALSE)
```

Arguments

`data`	an object of class `fitted_models` returned by the `fit_selected()` function, an object of class `prepared_data` returned by the `prepare_data()` function, or an object of class `calibration_results` returned by the `calibration()` function.
`new_variables`	a SpatRaster or data.frame of predictor variables. The names of these variables must match those used to prepare the date or calibrate the models provided in `data`.
`subset_variables`	(logical) whether to include in the analysis only the variables present in the selected models. Only applicable if `data` is a `fitted_models` object. Default is FALSE
`mask`	(SpatRaster, SpatVector, or SpatExtent) spatial object used to mask the variables (optional). Default is NULL.
`type`	(character) type of MOP analysis to be performed. Options available are "basic", "simple" and "detailed". See Details for further information.

`na_in_range` (logical) whether to assign `NA` to regions within the projected area (G) where environmental conditions fall within the range of the calibration data (M). If `TRUE` (default), these regions are assigned `NA`. If `FALSE`, they are assigned `0` in the simple and basic MOP outputs, and `"within ranges"` in the detailed MOP output.

`calculate_distance` (logical) whether to calculate distances (dissimilarities) between m and g. The default, FALSE, runs rapidly and does not assess dissimilarity levels.

`where_distance` (character) where to calculate distances, considering how conditions in g are positioned in comparison to the range of conditions in m. Options available are "in_range", "out_range" and "all". Default is "in_range".

`distance` (character) which distances are calculated, euclidean or mahalanobis. Only applicable if calculate_distance = TRUE.

`scale` (logical or numeric) whether to scale as in `scale`. Default is FALSE.

`center` (logical or numeric) whether to center as in `scale`. Default is FALSE.

`fix_NA` (logical) whether to fix layers so cells with NA values are the same in all layers. Setting to FALSE may save time if the rasters are big and have no NA matching problems. Default is TRUE.

`percentage` (numeric) percentage of `m` closest conditions used to derive mean environmental distances to each combination of conditions in `g`.

`comp_each` (numeric) number of combinations in `g` to be used for distance calculations at a time. Increasing this number requires more RAM

`tol` (numeric) tolerance to detect linear dependencies when calculating Mahalanobis distances. The default, NULL, uses `.Machine$double.eps`.

`rescale_distance` (logical) whether to re-scale distances 0-1. Re-scaling prevents comparisons of dissimilarity values obtained from runs with different values of `percentage`.

`parallel` (logical) whether to compute MOP in parallel. Default is FALSE.

`ncores` (numeric) number of cores to use for parallel processing. Default is NULL and uses available cores - 1. This is only applicable if `parallel = TRUE`.

`progress_bar` (logical) whether to display a progress bar during distance calculation. Only applicable if calculate_distance is TRUE. Default is TRUE.

`write_files` (logical) whether to save the MOP results (SpatRasters and data.frames) to disk. Default is FALSE.

`out_dir` (character) directory path where results will be saved. Only relevant if `write_files = TRUE`.

`overwrite` (logical) whether to overwrite SpatRasters if they already exist. Only applicable if `write_files = TRUE`. Default is FALSE.

Details

`type` options return results that differ in the detail of how non-analogous conditions are identified.

- **basic** - makes calculation as proposed by Owens et al. (2013) doi:10.1016/j.ecolmodel.2013.04.011.

- **simple** - calculates how many variables in the set of interest are non-analogous to those in the reference set.
- **detailed** - calculates five additional extrapolation metrics. See `mop_detailed` under `Value` below for full details.

`where_distance` options determine what values should be used to calculate dissimilarity

- **in_range** - only conditions inside `m` ranges
- **out_range** - only conditions outside `m` ranges
- **all** - all conditions

When the variables used to represent conditions have different units, scaling and centering are recommended. This step is only valid when Euclidean distances are used.

Value

An object of class `mop_results` containing:

- **summary** - a data.frame with details of the data used in the analysis:
  - *variables* - names of variables considered.
  - *type* - type of MOP analysis performed.
  - *scale* - value according to the argument `scale`.
  - *center* - value according to the argument `center`.
  - *calculate_distance* - value according to the argument `calculate_distance`.
  - *distance* - option regarding distance used.
  - *percentage* - percentage of `m` used as reference for distance calculation.
  - *rescale_distance* - value according to the argument `rescale_distance`.
  - *fix_NA* - value according to the argument `fix_NA`.
  - *N_m* - total number of elements (cells with values or valid rows) in `m`.
  - *N_g* - total number of elements (cells with values or valid rows) in `g`.
  - *m_ranges* - the range (minimum and maximum values) of the variable in reference conditions (`m`)
- **mop_distances** - if `calculate_distance = TRUE`, a SpatRaster with distance values for the set of interest (`g`). Higher values represent greater dissimilarity compared to the set of reference (`m`).
- **mop_basic** - a SpatRaster for the set of interest representing conditions in which at least one of the variables is non-analogous to the set of reference. Values should be: 1 for non-analogous conditions, and NA for conditions inside the ranges of the reference set.
- **mop_simple** - a SpatRaster, for the set of interest, representing how many variables in the set of interest are non-analogous to those in the reference set. NA is used for conditions inside the ranges of the reference set.
- **mop_detailed** - a list containing:
  - *interpretation_combined* - a data.frame to help identify combinations of variables in *towards_low_combined* and *towards_high_combined* that are non-analogous to `m`.
  - *towards_low_end* - a SpatRaster for all variables representing where non-analogous conditions were found towards low values of each variable.

- *towards_high_end* - a SpatRaster for all variables representing where non-analogous conditions were found towards high values of each variable.
- *towards_low_combined* - a SpatRaster with values representing the identity of the variables found to have non-analogous conditions towards low values.
- *towards_high_combined* - a SpatRaster with values representing the identity of the variables found to have non-analogous conditions towards high values.

See Also

`projection_mop()`

Examples

```
# Import an example of fitted models (output of fit_selected())
data("fitted_model_maxnet", package = "kuenm2")

# Import variables under a new set of conditions
# Here, future climate data
future_scenario <- terra::rast(system.file("extdata",
                                           "wc2.1_10m_bioc_ACCESS-CM2_ssp585_2081-2100.tif",
                                              package = "kuenm2"))



# Rename variables to match the variable names in the fitted models
names(future_scenario) <- sub("bio0", "bio", names(future_scenario))
names(future_scenario) <- sub("bio", "bio_", names(future_scenario))

# Run MOP
sm <- single_mop(data = fitted_model_maxnet, new_variables = future_scenario,
                 type = "detailed")
```

---

sp_swd *Prepared Data for maxnet models*

Description

A `prepared_data` object resulted from `prepare_data()` to calibrate models using 'glm' algorithm.

Usage

```
data("sp_swd")
```

Format

A `prepared_data` object with the following elements:

species Species names

calibration_data A data.frame containing the variables extracted for presence and background points

formula_grid A data.frame with the ID, formulas, and regularization multipliers of each candidate model

part_data A list with the partition data, where each element corresponds to a replicate and contains the **indices of the test points** for that replicate

partition_method A character indicating the partition method

n_replicates A numeric value indicating the number of replicates or k-folds

train_proportion A numeric value indicating the proportion of occurrences used as train points when the partition method is 'subsample' or 'bootstrap'

data_xy A data.frame with the coordinates of the occurrence and background points

continuous_variables A character indicating the names of the continuous variables

categorical_variables A character indicating the names of the categorical variables

weights A numeric value specifying weights for the occurrence records. It's NULL, meaning it was not set weights.

pca A prcomp object storing PCA information. Is NULL, meaning PCA was not performed

algorithm A character indicating the algorithm (glm)

---

sp_swd_glm *Prepared Data for glm models*

Description

A prepared_data object resulted from prepare_data() to calibrate models using 'glm' algorithm.

Usage

```
data("sp_swd_glm")
```

Format

A prepared_data object with the following elements:

species Species names

calibration_data A data.frame containing the variables extracted for presence and background points

formula_grid A data.frame with the ID, formulas, and regularization multipliers of each candidate model

part_data A list with the partition data, where each element corresponds to a replicate and contains the **indices of the test points** for that replicate

partition_method A character indicating the partition method

n_replicates A numeric value indicating the number of replicates or k-folds

train_proportion A numeric value indicating the proportion of occurrences used as train points when the partition method is 'subsample' or 'bootstrap'

data_xy A data.frame with the coordinates of the occurrence and background points

continuous_variables A character indicating the names of the continuous variables

categorical_variables A character indicating the names of the categorical variables

weights A numeric value specifying weights for the occurrence records. It's NULL, meaning it was not set weights.

pca A prcomp object storing PCA information. Is NULL, meaning PCA was not performed

algorithm A character indicating the algorithm (glm)

---

swd_spatial_block *Prepared data with spatial blocks created with ENMeval*

Description

A prepared_data object resulted from prepare_data() to calibrate models using 'glmnet' algorithm. In this object, the original partitioning was replaced with spatial blocks generated using the get.block() method from the ENMeval R package.

Usage

```
data("sp_swd")
```

Format

An object of class prepared_data of length 13.

---

user_data *User Custom Calibration Data*

Description

A data.frame containing presence and background records along with environmental variables used to demonstrate data preparation with user-supplied data.

Usage

```
data("user_data")
```

Format

A `data.frame` with the following columns:

pr_bg Column indicating presences (1) and background (0).

bio_1 The extracted values for the variable bio_1 at presence and background points.

bio_7 The extracted values for the variable bio_12 at presence and background points.

bio_12 The extracted values for the variable bio_12 at presence and background points.

bio_15 The extracted values for the variable bio_15 at presence and background points.

bio_15 The extracted values for the variable soilType at presence and background points.

`var` *SpatRaster Representing present-day Conditions (WorldClim)*

Description

Raster layer containing bioclimatic variables representing present-day climatic conditions. The variables were obtained at a 10 arc-minute resolution and masked using the `m` region provided in the package. Data sourced from WorldClim: https://worldclim.org/data/worldclim21.html

Format

A `SpatRaster` object.

Value

No return value. Used with function `rast` to bring raster variables to analysis.

Examples

```
var <- terra::rast(system.file("extdata",
                               "Current_variables.tif",
                                package = "kuenm2"))
terra::plot(var)
```

variable_importance *Variable importance*

Description

Variable importance

Usage

```
variable_importance(models, modelID = NULL, by_terms = FALSE,
                    parallel = FALSE, ncores = NULL,
                    progress_bar = TRUE, verbose = TRUE)
```

Arguments

models	an object of class fitted_models returned by the fit_selected() function.
modelID	(character). Default = NULL.
by_terms	(logical) whether to calculate importance by model terms (e.g., bio1, I(bio1^2), hinge(bio1)) instead of aggregating by variable. Default = FALSE.
parallel	(logical) whether to calculate importance in parallel. Default is FALSE.
ncores	(numeric) number of cores to use for parallel processing. Default is NULL and uses available cores - 1. This is only applicable if parallel = TRUE.
progress_bar	(logical) whether to display a progress bar during processing. Default is TRUE.
verbose	(logical) whether to display detailed messages during processing. Default is TRUE.

Value

A data.frame containing the relative contribution of each variable (or term if by_terms = TRUE). An identification for distinct models is added if fitted contains multiple models.

See Also

plot_importance()

Examples

```
# Example with maxnet
# Import example of fitted_models (output of fit_selected())
data(fitted_model_maxnet, package = "kuenm2")

# Variable importance
imp_maxnet <- variable_importance(models = fitted_model_maxnet)

# Plot
plot_importance(imp_maxnet)
```

```
# Example with glm
# Import example of fitted_models (output of fit_selected())
data(fitted_model_glm, package = "kuenm2")

# Variable importance
imp_glm <- variable_importance(models = fitted_model_glm)

# Plot
plot_importance(imp_glm)
```

`world` *World country polygons from Natural Earth*

Description

A spatial vector of the world countries. This is a simplified version of the `countries110` from rnaturalearth R package.

Format

A `SpatVector` object.

Value

No return value. Used with function `vect` to bring vector variables to analysis.

Examples

```
m <- terra::vect(system.file("extdata",
                             "world.gpkg",
                              package = "kuenm2"))
terra::plot(m)
```

Index