Package ‘biomod2’

May 2, 2024

Type Package

Title Ensemble Platform for Species Distribution Modeling

Version 4.2-5

Date 2024-04-30

Author Wilfried Thuiller [aut],
       Damien Georges [aut],
       Maya Gueguen [aut, cre],
       Robin Engler [aut],
       Frank Breiner [aut],
       Bruno Lafourcade [aut],
       Remi Patin [aut],
       Helene Blancheteau [aut]

Maintainer Maya Gueguen <maya.gueguen@univ-grenoble-alpes.fr>

Contact Wilfried Thuiller <wilfried.thuiller@univ-grenoble-alpes.fr>,
       Maya Gueguen <maya.gueguen@univ-grenoble-alpes.fr>, Helene
       Blancheteau <helene.blancheteau@univ-grenoble-alpes.fr>

BugReports https://github.com/biomodhub/biomod2/issues

URL https://biomodhub.github.io/biomod2/

Description Functions for species distribution modeling, calibration and evaluation,
ensemble of models, ensemble forecasting and visualization. The package permits to run
consistently up to 10 single models on a presence/absences (resp presences/pseudo-absences)
dataset and to combine them in ensemble models and ensemble projections. Some bench of other
evaluation and visualisation tools are also available within the package.

Depends R (>= 4.1)

Imports stats, utils, methods, terra (>= 1.6-33), sp, reshape,
       reshape2, abind, foreach, ggplot2, gbm (>= 2.1.3), rpart, MASS,
       pROC (>= 1.15.0), PresenceAbsence, dplyr

Suggests Hmisc, gam, mgcv, earth, maxnet, mda, nnet, randomForest,
       xgboost, car, caret, dismo, ENMeval, doParallel, raster,
       ggpubr, testthat, knitr, markdown, tidyterra, ggtext

License GPL-3
RoxygenNote 7.3.1
Encoding UTF-8
VignetteBuilder knitr
Collate 'biomod2-package.R' 'biomod2_globalVariables.R'
  'biomod2_classes_0.R' 'biomod2_classes_1.R'
  'biomod2_classes_2.R' 'biomod2_classes_3.R'
  'biomod2_classes_4.R' 'biomod2_classes_5.R'
  'biomod2_internal.R' 'biomod2_data.R'
  'BIOMOD_EnsembleForecasting.R' 'BIOMOD_EnsembleModeling.R'
  'BIOMOD_FormatingData.R' 'BIOMOD_LoadModels.R'
  'BIOMOD_Modeling.R' 'BIOMOD_Projection.R' 'BIOMOD_RangeSize.R'
  'DEPRECATED.R' 'bm_BinaryTransformation.R'
  'bm_CrossValidation.R' 'bm_FindOptimStat.R' 'bm_MakeFormula.R'
  'bm_ModelingOptions.R' 'bm_PlotEvalBoxplot.R'
  'bm_PlotEvalMean.R' 'bm_PlotRangeSize.R'
  'bm_PlotResponseCurves.R' 'bm_PlotVarImpBoxplot.R'
  'bm_PseudoAbsences.R' 'bm_RunModelsLoop.R' 'bm_SRE.R'
  'bm_SampleBinaryVector.R' 'bm_SampleFactorLevels.R'
  'bm_Tuning.R' 'bm_VariablesImportance.R' 'zzz.R'
LazyData true
NeedsCompilation no
Repository CRAN
Date/Publication 2024-05-02 13:52:59 UTC

R topics documented:

bioclim_current ............................................. 3
bioclim_future ............................................. 4
BIOMOD.ensemble.models.out ............................... 4
BIOMOD.formated.data .................................... 7
BIOMOD.formated.data.PA ................................ 11
BIOMOD.models.options .................................. 15
BIOMOD.models.out ....................................... 16
BIOMOD.options.dataset ................................ 18
BIOMOD.options.default ................................ 20
BIOMOD.projection.out .................................. 21
BIOMOD.stored.data ..................................... 24
biomod2_ensemble_model ................................ 25
biomod2_model ............................................ 27
BIOMOD_EnsembleForecasting ............................. 29
BIOMOD_EnsembleModeling ................................ 33
BIOMOD_FormatingData .................................. 40
BIOMOD_LoadModels ....................................... 47
BIOMOD_Modeling .......................................... 49
BIOMOD_Projection ....................................... 56
BIOMOD_RangeSize ........................................ 60
bioclim_current

A SpatRaster with 5 bioclimatic variables commonly used for SDM and describing current climate. Additional information available at worldclim

Usage

bioclim_current
**Format**

A `SpatRaster` with 5 layers:

- `bio3` Isothermality
- `bio4` Temperature Seasonality
- `bio7` Temperature Annual Range
- `bio11` Mean Temperature of Coldest Quarter
- `bio12` Annual Precipitation

---

**Description**

A `SpatRaster` with 5 bioclimatic variables commonly used for SDM and describing future climate based on old RCP scenarios at the horizon 2080.

**Usage**

`bioclim_future`
Arguments

object a BIOMOD.ensemble.models.out object

Slots

modeling.id a character corresponding to the name (ID) of the simulation set
dir.name a character corresponding to the modeling folder
sp.name a character corresponding to the species name
expl.var.names a vector containing names of explanatory variables
models.out a BIOMOD.stored.models.out-class object containing informations from BIOMOD_Modeling object
em.by a character corresponding to the way kept models have been combined to build the ensemble models, must be among PA+run, PA+algo, PA, algo, all
em.computed a vector containing names of ensemble models
em.failed a vector containing names of failed ensemble models
em.models_kept a list containing single models for each ensemble model
models.evaluation a BIOMOD.stored.data.frame-class object containing models evaluation variables.importance a BIOMOD.stored.data.frame-class object containing variables importance
models.prediction a BIOMOD.stored.data.frame-class object containing models predictions
models.prediction.eval a BIOMOD.stored.data.frame-class object containing models predictions for evaluation data
link a character containing the file name of the saved object

Author(s)

Damien Georges

See Also

BIOMOD_EnsembleModeling, BIOMOD_LoadModels, BIOMOD_PresenceOnly, bm_VariablesImportance, bm_PlotEvalMean, bm_PlotEvalBoxplot, bm_PlotVarImpBoxplot, bm_PlotResponseCurves

Other Toolbox objects: BIOMOD.formated.data, BIOMOD.formated.data.PA, BIOMOD.models.options, BIOMOD.models.out, BIOMOD.options.dataset, BIOMOD.options.default, BIOMOD.projection.out, BIOMOD.stored.data, biomod2_ensemble_model, biomod2_model

Examples

showClass("BIOMOD.ensemble.models.out")

## ---------------------------------------------------------------
library(terra)

# Load species occurrences (6 species available)
# Select the name of the studied species
myRespName <- 'GuloGulo'

# Get corresponding presence/absence data
myResp <- as.numeric(DataSpecies[, myRespName])

# Get corresponding XY coordinates
myRespXY <- DataSpecies[, c('X_WGS84', 'Y_WGS84')]

# Load environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
data(bioclim_current)
myExpl <- terra::rast(bioclim_current)

## ----------------------------------------------------------------------- 
file.out <- paste0(myRespName, '/', myRespName, '.AllModels.models.out')
if (file.exists(file.out)) {
  myBiomodModelOut <- get(load(file.out))
} else {
  # Format Data with true absences
  myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
                                        expl.var = myExpl,
                                        resp.xy = myRespXY,
                                        resp.name = myRespName)

  # Model single models
  myBiomodModelOut <- BIOMOD_Modeling(bm.format = myBiomodData,
                                        modeling.id = 'AllModels',
                                        models = c('RF', 'GLM'),
                                        CV.strategy = 'random',
                                        CV.nb.rep = 2,
                                        CV.perc = 0.8,
                                        OPT.strategy = 'bigboss',
                                        metric.eval = c('TSS', 'ROC'),
                                        var.import = 3,
                                        seed.val = 42)
}

## ----------------------------------------------------------------------- 
# Model ensemble models
myBiomodEM <- BIOMOD_EnsembleModeling(bm.mod = myBiomodModelOut,
                                        models.chosen = 'all',
                                        em.by = 'all',
                                        em.algo = c('EMmean', 'EMca'),
                                        metric.select = c('TSS'),
                                        metric.select.thresh = c(0.7),
                                        metric.eval = c('TSS', 'ROC'),
                                        ...
myBiomodEM

BIOMOD.formated.data BIOMOD_FormatingData() output object class

Description

Class returned by BIOMOD_FormatingData, and used by bm_Tuning, bm_CrossValidation and BIOMOD_Modeling

Usage

## S4 method for signature 'numeric,data.frame'
BIOMOD.formated.data(
  sp,
  env,
  xy = NULL,
  dir.name = ".",
  sp.name = NULL,
  eval.sp = NULL,
  eval.env = NULL,
  eval.xy = NULL,
  na.rm = TRUE,
  data.mask = NULL,
  shared.eval.env = FALSE,
  filter.raster = FALSE
)

## S4 method for signature 'data.frame,ANY'
BIOMOD.formated.data(
  sp,
  env,
  xy = NULL,
  dir.name = ".",
  sp.name = NULL,
  eval.sp = NULL,
  eval.env = NULL,
  eval.xy = NULL,
  na.rm = TRUE,
  filter.raster = FALSE
)

## S4 method for signature 'numeric,matrix'

var.import = 3,
seed.val = 42)
BIOMOD.formated.data(
  sp,
  env,
  xy = NULL,
  dir.name = ".",
  sp.name = NULL,
  eval.sp = NULL,
  eval.env = NULL,
  eval.xy = NULL,
  na.rm = TRUE,
  filter.raster = FALSE
)

## S4 method for signature 'numeric,SpatRaster'
BIOMOD.formated.data(
  sp,
  env,
  xy = NULL,
  dir.name = ".",
  sp.name = NULL,
  eval.sp = NULL,
  eval.env = NULL,
  eval.xy = NULL,
  na.rm = TRUE,
  shared.eval.env = FALSE,
  filter.raster = FALSE
)

## S4 method for signature 'BIOMOD.formated.data'
show(object)

Arguments

**sp**
A vector, a **SpatVector** without associated data (if presence-only), or a **SpatVector** object containing binary data (0: absence, 1: presence, NA: indeterminate) for a single species that will be used to build the species distribution model(s).

*Note that old format from sp are still supported such as SpatialPoints (if presence-only) or SpatialPointsDataFrame object containing binary data.*

**env**
a matrix, data.frame, **SpatVector** or **SpatRaster** object containing the explanatory variables (in columns or layers) that will be used to build the species distribution model(s).

*Note that old format from raster and sp are still supported such as RasterStack and SpatialPointsDataFrame objects.*

**xy**
(optional, default NULL)
If resp.var is a vector, a 2-columns matrix or data.frame containing the corresponding X and Y coordinates that will be used to build the species distribution model(s)

**dir.name**
a character corresponding to the modeling folder
sp.name a character corresponding to the species name
eval.sp (optional, default NULL)
A vector, a SpatVector without associated data (if presence-only), or a SpatVector object containing binary data (0: absence, 1: presence, NA: indeterminate) for a single species that will be used to evaluate the species distribution model(s) with independent data.

Note that old format from sp are still supported such as SpatialPoints (if presence-only) or SpatialPointsDataFrame object containing binary data.

eval.env (optional, default NULL)
A matrix, data.frame, SpatVector or SpatRaster object containing the explanatory variables (in columns or layers) that will be used to evaluate the species distribution model(s) with independent data.

Note that old format from raster and sp are still supported such as RasterStack and SpatialPointsDataFrame objects.

eval.xy (optional, default NULL)
If resp.var is a vector, a 2-columns matrix or data.frame containing the corresponding X and Y coordinates that will be used to evaluate the species distribution model(s) with independent data.

na.rm (optional, default TRUE)
A logical value defining whether points having one or several missing values for explanatory variables should be removed from the analysis or not.

data.mask (optional, default NULL)
A SpatRaster object containing the mask of the studied area.

shared.eval.env (optional, default FALSE)
A logical value defining whether the explanatory variables used for the evaluation dataset are the same than the ones for calibration (if eval.env not provided for example) or not.

filter.raster (optional, default FALSE)
If env is of raster type, a logical value defining whether sp is to be filtered when several points occur in the same raster cell.

object a BIOMOD.formated.data object

Slots

dir.name a character corresponding to the modeling folder
sp.name a character corresponding to the species name
coord a 2-columns data.frame containing the corresponding X and Y coordinates
data.species a vector containing the species observations (0, 1 or NA)
data.env.var a data.frame containing explanatory variables
data.mask a SpatRaster object containing the mask of the studied area
has.data.eval a logical value defining whether evaluation data is given

eval.coord (optional, default NULL)
A 2-columns data.frame containing the corresponding X and Y coordinates for evaluation data.
eval.data.species (optional, default NULL)
   A vector containing the species observations (0, 1 or NA) for evaluation data

eval.data.env.var (optional, default NULL)
   A data.frame containing explanatory variables for evaluation data

Author(s)
Damien Georges

See Also
BIOMOD_FormatingData, bm_Tuning, bm_CrossValidation, BIOMOD_Modeling, bm_RunModelsLoop

Other Toolbox objects: BIOMOD.ensemble.models.out, BIOMOD.formated.data.PA, BIOMOD.models.options, BIOMOD.models.out, BIOMOD.options.dataset, BIOMOD.options.default, BIOMOD.projection.out, BIOMOD.stored.data, biomod2_ensemble_model, biomod2_model

Examples

showClass("BIOMOD.formated.data")

## ----------------------------------------------------------------------- #
library(terra)

# Load species occurrences (6 species available)
data(DataSpecies)
head(DataSpecies)

# Select the name of the studied species
myRespName <- 'GuloGulo'

# Get corresponding presence/absence data
myResp <- as.numeric(DataSpecies[, myRespName])

# Get corresponding XY coordinates
myRespXY <- DataSpecies[, c("X_WGS84", "Y_WGS84")]

# Load environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
data(bioclim_current)
myExpl <- terra::rast(bioclim_current)

## ----------------------------------------------------------------------- #

# Format Data with true absences
myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
   expl.var = myExpl,
   resp.xy = myRespXY,
   resp.name = myRespName)

myBiomodData
plot(myBiomodData)
BIOMOD.formated.data.PA

summary(myBiomodData)

BIOMOD.formated.data.PA

BIOMOD_FormatingData() output object class (with pseudo-absences)

Description

Class returned by `BIOMOD_FormatingData`, and used by `bm_Tuning`, `bm_CrossValidation` and `BIOMOD_Modeling`

Usage

```r
## S4 method for signature 'numeric, data.frame'
BIOMOD.formated.data.PA(
  sp,
  env,
  xy = NULL,
  dir.name = ".",
  sp.name = NULL,
  eval.sp = NULL,
  eval.env = NULL,
  eval.xy = NULL,
  PA.nb.rep = 1,
  PA.strategy = "random",
  PA.nb.absences = NULL,
  PA.dist.min = 0,
  PA.dist.max = NULL,
  PA.sre.quant = 0.025,
  PA.user.table = NULL,
  na.rm = TRUE,
  filter.raster = FALSE
)

## S4 method for signature 'numeric, SpatRaster'
BIOMOD.formated.data.PA(
  sp,
  env,
  xy = NULL,
  dir.name = ".",
  sp.name = NULL,
  eval.sp = NULL,
  eval.env = NULL,
  eval.xy = NULL,
```

PA.nb.rep = 1,
PA.strategy = "random",
PA.nb.absences = NULL,
PA.dist.min = 0,
PA.dist.max = NULL,
PA.sre.quant = 0.025,
PA.user.table = NULL,
na.rm = TRUE,
filter.raster = FALSE
)

Arguments

sp  A vector, a SpatVector without associated data (if presence-only), or a SpatVector object containing binary data (0: absence, 1: presence, NA: indeterminate) for a single species that will be used to build the species distribution model(s)

Note that old format from sp are still supported such as SpatialPoints (if presence-only) or SpatialPointsDataFrame object containing binary data.

eval.sp  A vector, a SpatVector without associated data (if presence-only), or a SpatVector object containing binary data (0: absence, 1: presence, NA: indeterminate) for a single species that will be used to evaluate the species distribution model(s) with independent data

Note that old format from sp are still supported such as SpatialPoints (if presence-only) or SpatialPointsDataFrame object containing binary data.

xy  If resp.var is a vector, a 2-columns matrix or data.frame containing the corresponding X and Y coordinates that will be used to build the species distribution model(s)

dir.name  a character corresponding to the modeling folder

sp.name  a character corresponding to the species name

eval.env  A matrix, data.frame, SpatVector or SpatRaster object containing the explanatory variables (in columns or layers) that will be used to build the species distribution model(s) with independent data

Note that old format from raster and sp are still supported such as RasterStack and SpatialPointsDataFrame objects.

eval.xy  If resp.var is a vector, a 2-columns matrix or data.frame containing the corresponding X and Y coordinates that will be used to evaluate the species distribution model(s) with independent data

Note that old format from raster and sp are still supported such as RasterStack and SpatialPointsDataFrame objects.
PA.nb.rep *(optional, default 0)*
If pseudo-absence selection, an integer corresponding to the number of sets (repetitions) of pseudo-absence points that will be drawn

PA.strategy *(optional, default NULL)*
If pseudo-absence selection, a character defining the strategy that will be used to select the pseudo-absence points. Must be random, sre, disk or user.defined (see Details)

PA.nb.absences *(optional, default 0)*
If pseudo-absence selection, and PA.strategy = 'random' or PA.strategy = 'sre' or PA.strategy = 'disk', an integer (or a vector of integer the same size as PA.nb.rep) corresponding to the number of pseudo-absence points that will be selected for each pseudo-absence repetition (true absences included)

PA.dist.min *(optional, default 0)*
If pseudo-absence selection and PA.strategy = 'disk', a numeric defining the minimal distance to presence points used to make the disk pseudo-absence selection (in meters, see Details)

PA.dist.max *(optional, default 0)*
If pseudo-absence selection and PA.strategy = 'disk', a numeric defining the maximal distance to presence points used to make the disk pseudo-absence selection (in meters, see Details)

PA.sre.quant *(optional, default 0)*
If pseudo-absence selection and PA.strategy = 'sre', a numeric between 0 and 0.5 defining the half-quantile used to make the sre pseudo-absence selection (see Details)

PA.user.table *(optional, default NULL)*
If pseudo-absence selection and PA.strategy = 'user.defined', a matrix or data.frame with as many rows as resp.var values, as many columns as PA.nb.rep, and containing TRUE or FALSE values defining which points will be used to build the species distribution model(s) for each repetition (see Details)

na.rm *(optional, default TRUE)*
A logical value defining whether points having one or several missing values for explanatory variables should be removed from the analysis or not

filter.raster *(optional, default FALSE)*
If env is of raster type, a logical value defining whether sp is to be filtered when several points occur in the same raster cell

**Slots**

dir.name a character corresponding to the modeling folder

sp.name a character corresponding to the species name

data.species a vector containing the species observations (0, 1 or NA)

filter.raster a SpatRaster object containing the mask of the studied area

has.data.eval a logical value defining whether evaluation data is given
eval.coord (optional, default NULL)
A 2-columns data.frame containing the corresponding X and Y coordinates for evaluation data

eval.data.species (optional, default NULL)
A vector containing the species observations (0, 1 or NA) for evaluation data

eval.data.env.var (optional, default NULL)
A data.frame containing explanatory variables for evaluation data

PA.strategy a character corresponding to the pseudo-absence selection strategy

PA.table a data.frame containing the corresponding table of selected pseudo-absences (indicated by TRUE or FALSE) from the pa.tab list element returned by the bm_PseudoAbsences function

Author(s)
Damien Georges

See Also

BIOMOD_FormatingData, bm_PseudoAbsences, bm_Tuning, bm_CrossValidation, BIOMOD_Modeling, bm_RunModelsLoop

Other Toolbox objects: BIOMOD.ensemble.models.out, BIOMOD.formated.data, BIOMOD.models.options, BIOMOD.models.out, BIOMOD.options.dataset, BIOMOD.options.default, BIOMOD.projection.out, BIOMOD.stored.data, biomod2_ensemble_model, biomod2_model

Examples

showClass("BIOMOD.formated.data.PA")

## ----------------------------------------------------------------------- #
library(terra)

# Load species occurrences (6 species available)
data(DataSpecies)
head(DataSpecies)

# Select the name of the studied species
myRespName <- 'GuloGulo'

# Keep only presence informations
DataSpecies <- DataSpecies[which(DataSpecies[, myRespName] == 1), ]

# Get corresponding presence/absence data
data <- as.numeric(DataSpecies[, myRespName])

# Get corresponding XY coordinates
data <- DataSpecies[, c('X_WGS84', 'Y_WGS84')]

# Load environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
data(bioclim_current)
myExpl <- terra::rast(bioclim_current)

# Format Data with pseudo-absences: random method
myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
                                      expl.var = myExpl,
                                      resp.xy = myRespXY,
                                      resp.name = myRespName,
                                      PA.nb.rep = 4,
                                      PA.strategy = 'random',
                                      PA.nb.absences = 1000)

myBiomodData
plot(myBiomodData)

---

**BIOMOD.models.options**  
**bm_ModelingOptions**  
**output object class**

**Description**

Class returned by `bm_ModelingOptions` and used by `BIOMOD_Modeling`

**Usage**

```r
## S4 method for signature 'BIOMOD.models.options'
show(object)
```

```r
## S4 method for signature 'BIOMOD.models.options'
print(x, dataset = "_allData_allRun")
```

**Arguments**

- `object`: a `BIOMOD.models.options` object
- `x`: a `BIOMOD.models.options` object
- `dataset`: a character corresponding to the name of a dataset contained in the `arg.values` slot of the `BIOMOD.options.dataset` object for each model

**Slots**

- `models`: a vector containing model names for which options have been retrieved and defined, must be `algo.datatype.package.function`
- `options`: a list containing `BIOMOD.options.dataset` object for each model

**Author(s)**

Maya Gueguen
See Also

BIOMOD.options.default, BIOMOD.options.dataset, bm_ModelingOptions, bm_Tuning, BIOMOD_Modeling

Other Toolbox objects: BIOMOD.ensemble.models.out, BIOMOD.formated.data, BIOMOD.formated.data.PA, BIOMOD.models.out, BIOMOD.options.dataset, BIOMOD.options.default, BIOMOD.projection.out, BIOMOD.stored.data, biomod2_ensemble_model, biomod2_model

Examples

showClass("BIOMOD.models.options")

<table>
<thead>
<tr>
<th>BIOMOD.models.out</th>
<th>BIOMOD_Modeling()</th>
</tr>
</thead>
<tbody>
<tr>
<td>output object class</td>
<td></td>
</tr>
</tbody>
</table>

Description

Class returned by BIOMOD_Modeling, and used by BIOMOD_LoadModels, BIOMOD_PresenceOnly, BIOMOD_Projection and BIOMOD_EnsembleModeling

Usage

## S4 method for signature 'BIOMOD.models.out'

show(object)

Arguments

object a BIOMOD.models.out object

Slots

modeling.id a character corresponding to the name (ID) of the simulation set
dir.name a character corresponding to the modeling folder
sp.name a character corresponding to the species name
expl.var.names a vector containing names of explanatory variables
models.computed a vector containing names of computed models
models.failed a vector containing names of failed models
has.evaluation.data a logical value defining whether evaluation data is given
scale.models a logical value defining whether models have been rescaled or not
formated.input.data a BIOMOD.stored.formated.data-class object containing informations from BIOMOD_FormatingData object
calib.lines a BIOMOD.stored.data.frame-class object containing calibration lines
models.options a `BIOMOD.stored.options-class` object containing informations from `bm_ModelingOptions` object
models.evaluation a `BIOMOD.stored.data.frame-class` object containing models evaluation
variables.importance a `BIOMOD.stored.data.frame-class` object containing variables importance
models.prediction a `BIOMOD.stored.data.frame-class` object containing models predictions
models.prediction.eval a `BIOMOD.stored.data.frame-class` object containing models predictions for evaluation data
link a character containing the file name of the saved object

Author(s)
Damien Georges

See Also
`BIOMOD_Modeling`, `BIOMOD_LoadModels`, `BIOMOD_PresenceOnly`, `BIOMOD_Projection`, `BIOMOD_EnsembleModeling`,
`bm_VariablesImportance`, `bm_PlotEvalMean`, `bm_PlotEvalBoxplot`, `bm_PlotVarImpBoxplot`,
`bm_PlotResponseCurves`

Other Toolbox objects: `BIOMOD.ensemble.models.out`, `BIOMOD.formated.data`, `BIOMOD.formated.data.PA`,
`BIOMOD.models.options`, `BIOMOD.options.dataset`, `BIOMOD.options.default`, `BIOMOD.projection.out`,
`BIOMOD.stored.data`, `biomod2_ensemble_model`, `biomod2_model`

Examples

```r
showClass("BIOMOD.models.out")

# Load species occurrences (6 species available)
data(DataSpecies)
head(DataSpecies)

# Select the name of the studied species
myRespName <- 'GuloGulo'

# Get corresponding presence/absence data
myResp <- as.numeric(DataSpecies[, myRespName])

# Get corresponding XY coordinates
myRespXY <- DataSpecies[, c('X_WGS84', 'Y_WGS84')]

# Load environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
data(bioclim_current)
myExpl <- terra::rast(bioclim_current)
```
## Format Data with true absences

myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
    expl.var = myExpl,
    resp.xy = myRespXY,
    resp.name = myRespName)

## Model single models

myBiomodModelOut <- BIOMOD_Modeling(bm.format = myBiomodData,
    modeling.id = 'AllModels',
    models = c('RF', 'GLM'),
    CV.strategy = 'random',
    CV.nb.rep = 2,
    CV.perc = 0.8,
    OPT.strategy = 'bigboss',
    metric.eval = c('TSS', 'ROC'),
    var.import = 3,
    seed.val = 42)

myBiomodModelOut

---

**BIOMOD.options.dataset**

**bm_ModelingOptions output object class**

**Description**

Class returned by `bm_ModelingOptions` (a list of `BIOMOD.options.dataset` more exactly), and used by `BIOMOD_Modeling`

**Usage**

```r
# S4 method for signature 'character'
BIOMOD.options.dataset(
    mod,
    typ,
    pkg,
    fun,
    strategy,
    user.val = NULL,
    user.base = NULL,
    tuning.fun = NULL,
    bm.format = NULL,
    calib.lines = NULL
)
```
## S4 method for signature 'BIOMOD.options.dataset'
show(object)

## S4 method for signature 'BIOMOD.options.dataset'
print(x, dataset = "_allData_allRun")

### Arguments

- **mod**: a character corresponding to the model name to be computed, must be either ANN, CTA, FDA, GAM, GBM, GLM, MARS, MAXENT, MAXNET, RF, SRE, XGBOOST
- **typ**: a character corresponding to the data type to be used, must be either binary, binary.PA, abundance, compositional
- **pkg**: a character corresponding to the package containing the model function to be called
- **fun**: a character corresponding to the model function name to be called
- **strategy**: a character corresponding to the method to select models' parameters values, must be either default, bigboss, user.defined, tuned
- **user.val**: (optional, default NULL)
  - A list containing parameters values
- **user.base**: (optional, default NULL)
  - A character, default or bigboss used when strategy = 'user.defined'. It sets the bases of parameters to be modified by user defined values.
- **tuning.fun**: (optional, default NULL)
  - A character corresponding to the model function name to be called through train function for tuning parameters
- **bm.format**: (optional, default NULL)
  - A BIOMOD.formated.data or BIOMOD.formated.data.PA object returned by the BIOMOD_FormatingData function
- **calib.lines**: (optional, default NULL)
  - A data.frame object returned by get_calib_lines or bm_CrossValidation functions, to explore the distribution of calibration and validation datasets
- **object**: a BIOMOD.options.dataset object
- **x**: a BIOMOD.options.dataset object
- **dataset**: a character corresponding to the name of a dataset contained in the arg.values slot

### Slots

- **model**: a character corresponding to the model
- **type**: a character corresponding to the data type (binary, binary.PA, abundance, compositional)
- **package**: a character corresponding to the package containing the model function to be called
- **func**: a character corresponding to the model function name to be called
- **args.names**: a vector containing character corresponding to the model function arguments
BIOMOD.options.default

args.default a list containing for each dataset the default values for all arguments listed in args.names

args.values a list containing for each dataset the to-be-used values for all arguments listed in args.names

Author(s)

Maya Gueguen

See Also

BIOMOD.options.default, bm_ModelingOptions, bm_Tuning, BIOMOD_Modeling, bm_RunModelsLoop

Other Toolbox objects: BIOMOD.ensemble.models.out, BIOMOD.formated.data, BIOMOD.formated.data.PA, BIOMOD.models.options, BIOMOD.models.out, BIOMOD.options.default, BIOMOD.projection.out, BIOMOD.stored.data, biomod2_ensemble_model, biomod2_model

Examples

showClass("BIOMOD.options.dataset")

```
BIOMOD.options.default

bm_ModelingOptions output object class

Description

Class returned by bm_ModelingOptions (a list of BIOMOD.options.dataset more exactly), and used by BIOMOD_Modeling

Usage

## S4 method for signature 'character,character'

BIOMOD.options.default(mod, typ, pkg, fun)

Arguments

mod a character corresponding to the model name to be computed, must be either ANN, CTA, FDA, GAM, GBM, GLM, MARS, MAXENT, MAXNET, RF, SRE, XGBOOST

typ a character corresponding to the data type to be used, must be either binary, binary.PA, abundance, compositional

pkg a character corresponding to the package containing the model function to be called

fun a character corresponding to the model function name to be called
Slots

- model: a character corresponding to the model
- type: a character corresponding to the data type (binary, binary.PA, abundance, compositional)
- package: a character corresponding to the package containing the model function to be called
- func: a character corresponding to the model function name to be called
- args.names: a vector containing the character corresponding to the model function arguments
- args.default: a list containing for each dataset the default values for all arguments listed in args.names

Author(s)

Maya Gueguen

See Also

- BIOMOD.options.dataset, bm_ModelingOptions, bm_Tuning, BIOMOD_Modeling, bm_RunModelsLoop
- Other Toolbox objects: BIOMOD.ensemble.models.out, BIOMOD.formated.data, BIOMOD.formated.data.PA, BIOMOD.models.options, BIOMOD.models.out, BIOMOD.options.dataset, BIOMOD.projection.out, BIOMOD.stored.data, biomod2_ensemble_model, biomod2_model

Examples

showClass("BIOMOD.options.default")

Description

Class returned by BIOMOD_Projection, and used by BIOMOD_EnsembleForecasting

Usage

```r
## S4 method for signature 'BIOMOD.projection.out,missing'
plot(
x,
coord = NULL,
plot.output,
do.plot = TRUE,
std = TRUE,
scales,
size,
```
maxcell = 5e+05, 
...
)

## S4 method for signature 'BIOMOD.projection.out'
show(object)

Arguments

x a BIOMOD.projection.out object
coord a 2-columns data.frame containing the corresponding X and Y
plot.output (optional, default facet) a character determining the type of output: with plot.output = 'list' the function will return a list of plots (one plot per model); with 'facet' ; with plot.output = 'facet' the function will return a single plot with all asked projections as facet.
do.plot (optional, default TRUE) a boolean determining whether the plot should be displayed or just returned.
std (optional, default TRUE) a boolean controlling the limits of the color scales. With std = TRUE color scales are displayed between 0 and 1 (or 1000). With std = FALSE color scales are displayed between 0 and the maximum value observed.
scales (optional, default fixed) a character determining whether x and y scales are shared among facet. Argument passed to facet_wrap. Possible values: 'fixed', 'free_x', 'free_y', 'free'.
size (optional, default 0.75) a numeric determining the size of points on the plots and passed to geom_point.
maxcell maximum number of cells to plot. Argument transmitted to plot.
... additional parameters to be passed to get_predictions to select the models that will be plotted
object a BIOMOD.projection.out object

Slots

modeling.id a character corresponding to the name (ID) of the simulation set
proj.name a character corresponding to the projection name
dir.name a character corresponding to the modeling folder
sp.name a character corresponding to the species name
expl.var.names a vector containing names of explanatory variables
coord a 2-columns matrix or data.frame containing the corresponding X and Y coordinates used to project the species distribution model(s)
scale.models a logical value defining whether models have been rescaled or not
models.projected a vector containing names of projected models
models.out a BIOMOD.stored.data object
type a character corresponding to the class of the val slot of the proj.out slot
proj.out a BIOMOD.stored.data object
showClass("BIOMOD.projection.out")

## ----------------------------------------------------------------------- #
library(terra)

# Load species occurrences (6 species available)
data(DataSpecies)
head(DataSpecies)

# Select the name of the studied species
myRespName <- 'GuloGulo'

# Get corresponding presence/absence data
myResp <- as.numeric(DataSpecies[, myRespName])

# Get corresponding XY coordinates
myRespXY <- DataSpecies[, c('X_WGS84', 'Y_WGS84')]

# Load environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
data(bioclim_current)
myExpl <- terra::rast(bioclim_current)

## ----------------------------------------------------------------------- #
file.out <- paste0(myRespName, '/', myRespName, '.AllModels.models.out')
if (file.exists(file.out)) {
  myBiomodModelOut <- get(load(file.out))
} else {
  # Format Data with true absences
  myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
                                        expl.var = myExpl,
                                        resp.xy = myRespXY,
                                        resp.name = myRespName)

  # Model single models
  myBiomodModelOut <- BIOMOD_Modeling(bm.format = myBiomodData,
modeling.id = 'AllModels',
models = c('RF', 'GLM'),
CV.strategy = 'random',
CV.nb.rep = 2,
CV.perc = 0.8,
OPT.strategy = 'bigboss',
metric.eval = c('TSS', 'ROC'),
var.import = 3,
seed.val = 42)
}

myBiomodProj <- BIOMOD_Projection(bm.mod = myBiomodModelOut,
proj.name = 'Current',
new.env = myExpl,
models.chosen = 'all',
metric.binary = 'all',
metric.filter = 'all',
built.clamping.mask = TRUE)

myBiomodProj
plot(myBiomodProj)

---

## BIOMOD.stored.data

### BIOMOD_Modeling and BIOMOD_EnsembleModeling output object class

#### Description

Classes used by BIOMOD_Modeling and BIOMOD_EnsembleModeling to build their output object (see BIOMOD.models.out objects)

#### Details

BIOMOD.stored.data is the basic object containing the slots inMemory and link. All listed classes below are derived from BIOMOD.stored.data, and contain a val slot of specific type:

- • BIOMOD.stored.data.frame: val is a data.frame
- • BIOMOD.stored.SpatRaster: val is a PackedSpatRaster
- • BIOMOD.stored.files: val is a character
- • BIOMOD.stored.formated.data: val is a BIOMOD.formated.data object
- • BIOMOD.stored.options: val is a BIOMOD.models.options object
- • BIOMOD.stored.models.out: val is a BIOMOD.models.out object
Slots

- `inMemory` a logical defining whether the `val` slot has been loaded in memory or not
- `link` a character containing the file name of the saved `val` slot
- `val` an object of type depending on the `BIOMOD.stored.[...]` class (see Details)

Author(s)

Damien Georges

See Also

- `BIOMOD.formated.data`, `BIOMOD.models.out`, `BIOMOD_Modeling`, `BIOMOD_EnsembleModeling`, `BIOMOD_Projection`, `BIOMOD_EnsembleForecasting`

Other Toolbox objects: `BIOMOD.ensemble.models.out`, `BIOMOD.formated.data`, `BIOMOD.formated.data.PA`, `BIOMOD.models.options`, `BIOMOD.models.out`, `BIOMOD.options.dataset`, `BIOMOD.options.default`, `BIOMOD.projection.out`, `biomod2_ensemble_model`, `biomod2_model`

Examples

```r
showClass("BIOMOD.stored.data")
showClass("BIOMOD.stored.data.frame")
showClass("BIOMOD.stored.SpatRaster")
showClass("BIOMOD.stored.files")
showClass("BIOMOD.stored.formated.data")
showClass("BIOMOD.stored.options")
showClass("BIOMOD.stored.models.out")
```

---

**biomod2_ensemble_model**

*Ensemble model output object class (when running BIOMOD_EnsembleModeling())*

Description

Class created by `BIOMOD_EnsembleModeling`

Usage

```r
# S4 method for signature 'biomod2_ensemble_model'
show(object)
```

Arguments

- **object** a `biomod2_ensemble_model` object
**biomod2_ensemble_model**

Details

biomod2_model is the basic object for biomod2 ensemble species distribution models. All listed classes below are derived from biomod2_model, and have a model_class slot specific value:

- biomod2_ensemble_model: model_class is EM
- EMmean_biomod2_model: model_class is EMmean
- EMmedian_biomod2_model: model_class is EMmedian
- EMcv_biomod2_model: model_class is EMcv
- EMci_biomod2_model: model_class is EMci
- EMca_biomod2_model: model_class is EMca
- EMwmean_biomod2_model: model_class is EMwmean

Slots

- modeling.id a character corresponding to the name (ID) of the simulation set
- model_name a character corresponding to the model name
- model_class a character corresponding to the model class
- model_options a list containing the model options
- model the corresponding model object
- scaling_model the corresponding scaled model object
- dir_name a character corresponding to the modeling folder
- resp_name a character corresponding to the species name
- expl_var_names a vector containing names of explanatory variables
- expl_var_type a vector containing classes of explanatory variables
- expl_var_range a list containing ranges of explanatory variables
- model_evaluation a data.frame containing the model evaluations
- model_variables_importance a data.frame containing the model variables importance

Author(s)

Damien Georges

See Also

biomod2_model, BIOMOD_EnsembleModeling

Other Toolbox objects: BIOMOD.ensemble.models.out, BIOMOD.formated.data, BIOMOD.formated.data.PA, BIOMOD.models.options, BIOMOD.models.out, BIOMOD.options.dataset, BIOMOD.options.default, BIOMOD.projection.out, BIOMOD.stored.data, biomod2_model
biomod2_model

Examples

showClass("biomod2_ensemble_model")
showClass("EMmean_biomod2_model")
showClass("EMmedian_biomod2_model")
showClass("EMcv_biomod2_model")
showClass("EMci_biomod2_model")
showClass("EMca_biomod2_model")
showClass("EMwmean_biomod2_model")

biomod2_model Single model output object class (when running BIOMOD_Modeling())

Description

Class created by BIOMOD_Modeling and bm_RunModel

Usage

## S4 method for signature 'biomod2_model'
show(object)

Arguments

object a biomod2_model object

Details

biomod2_model is the basic object for biomod2 single species distribution models. All listed classes below are derived from biomod2_model, and have a model_class slot specific value :

- ANN_biomod2_model: model_class is ANN
- CTA_biomod2_model: model_class is CTA
- FDA_biomod2_model: model_class is FDA
- GBM_biomod2_model: model_class is GBM
- GLM_biomod2_model: model_class is GLM
- MARS_biomod2_model: model_class is MARS
- MAXENT_biomod2_model: model_class is MAXENT
- MAXNET_biomod2_model: model_class is MAXNET
- RF_biomod2_model: model_class is RF
- SRE_biomod2_model: model_class is SRE
Slots

model_name a character corresponding to the model name
model_class a character corresponding to the model class
model_options a list containing the model options
model the corresponding model object
scaling_model the corresponding scaled model object
dir_name a character corresponding to the modeling folder
resp_name a character corresponding to the species name
expl_var_names a vector containing names of explanatory variables
expl_var_type a vector containing classes of explanatory variables
expl_var_range a list containing ranges of explanatory variables
model_evaluation a data.frame containing the model evaluations
model_variables_importance a data.frame containing the model variables importance

Author(s)

Damien Georges

See Also

BIOMOD_Modeling, bm_RunModel

Other Toolbox objects: BIOMOD.ensemble.models.out, BIOMOD.formated.data, BIOMOD.formated.data.PA, BIOMOD.models.options, BIOMOD.models.out, BIOMOD.options.dataset, BIOMOD.options.default, BIOMOD.projection.out, BIOMOD.stored.data, biomod2_ensemble_model

Examples

showClass("biomod2_model")
showClass("ANN_biomod2_model")
showClass("CTA_biomod2_model")
showClass("FDA_biomod2_model")
showClass("GAM_biomod2_model")
showClass("GBM_biomod2_model")
showClass("GLM_biomod2_model")
showClass("MARS_biomod2_model")
showClass("MAXENT_biomod2_model")
showClass("MAXNET_biomod2_model")
showClass("RF_biomod2_model")
showClass("SRE_biomod2_model")
**Description**

This function allows to project ensemble models built with the `BIOMOD_EnsembleModeling` function onto new environmental data *(which can represent new areas, resolution or time scales for example).*

**Usage**

```r
BIOMOD_EnsembleForecasting(
  bm.em,
  bm.proj = NULL,
  proj.name = NULL,
  new.env = NULL,
  new.env.xy = NULL,
  models.chosen = "all",
  metric.binary = NULL,
  metric.filter = NULL,
  compress = TRUE,
  nb.cpu = 1,
  na.rm = TRUE,
  ...
)
```

**Arguments**

- **bm.em** a BIOMOD.ensemble.models.out object returned by the BIOMOD_EnsembleModeling function
- **bm.proj** a BIOMOD.projection.out object returned by the BIOMOD_Projection function
- **proj.name** *(optional, default NULL)*
  - If `bm.proj = NULL`, a character corresponding to the name (ID) of the projection set *(a new folder will be created within the simulation folder with this name)*
- **new.env** *(optional, default NULL)*
  - If `bm.proj = NULL`, a matrix, data.frame or SpatRaster object containing the new explanatory variables (in columns or layers, with names matching the variables names given to the BIOMOD_FormatingData function to build `bm.mod`) that will be used to project the species distribution model(s)
  - *Note that old format from raster are still supported such as RasterStack objects.*
new.env.xy  
(optional, default NULL) 
If new.env is a matrix or a data.frame, a 2-columns matrix or data.frame containing the corresponding X and Y coordinates that will be used to project the ensemble species distribution model(s).

models.chosen  
a vector containing model names to be kept, must be either all or a sub-selection of model names that can be obtained with the get_built_models function.

metric.binary  
(optional, default NULL) 
A vector containing evaluation metric names to be used to transform prediction values into binary values based on models evaluation scores obtained with the BIOMOD_Modeling function. Must be among all (same evaluation metrics than those of modeling.output) or ROC, TSS, KAPPA, ACCURACY, BIAS, POD, FAR, POFD, SR, CSI, ETS, HK, HSS, OR, ORSS.

metric.filter  
(optional, default NULL) 
A vector containing evaluation metric names to be used to transform prediction values into filtered values based on models evaluation scores obtained with the BIOMOD_Modeling function. Must be among all (same evaluation metrics than those of modeling.output) or ROC, TSS, KAPPA, ACCURACY, BIAS, POD, FAR, POFD, SR, CSI, ETS, HK, HSS, OR, ORSS.

compress  
(optional, default TRUE) 
A logical or a character value defining whether and how objects should be compressed when saved on hard drive, must be either TRUE, FALSE, xz or gzip (see Details).

nb.cpu  
(optional, default 1) 
An integer value corresponding to the number of computing resources to be used to parallelize the single models computation.

na.rm  
(optional, default TRUE) 
A boolean defining whether Ensemble Model projection should ignore NA in Individual Model projection. Argument ignored by EWmean ensemble algorithm.

...  
(optional, see Details)

Details

If models.chosen = 'all', projections are done for all calibration and pseudo absences runs if applicable. These projections may be used later by the BIOMOD_EnsembleForecasting function.

If build.clamping.mask = TRUE, a raster file will be saved within the projection folder. This mask values will correspond to the number of variables in each pixel that are out of their calibration/validation range, identifying locations where predictions are uncertain.

... can take the following values:

• on_0_1000: a logical value defining whether 0 - 1 probabilities are to be converted to 0 - 1000 scale to save memory on backup
• do.stack: a logical value defining whether all projections are to be saved as one SpatRaster object or several SpatRaster files (the default if projections are too heavy to be all loaded at once in memory)
• keep.in.memory: a logical value defining whether all projections are to be kept loaded at once in memory, or only links pointing to hard drive are to be returned
• output.format: a character value corresponding to the projections saving format on hard drive, must be either .grd, .img, .tif or .RData (the default if new.env is given as matrix or data.frame)

Value

A BIOMOD.projection.out object containing models projections, or links to saved outputs. Models projections are stored out of R (for memory storage reasons) in proj.name folder created in the current working directory:

1. the output is a data.frame if new.env is a matrix or a data.frame
2. it is a SpatRaster if new.env is a SpatRaster (or several SpatRaster objects, if new.env is too large)
3. raw projections, as well as binary and filtered projections (if asked), are saved in the proj.name folder

Author(s)

Wilfried Thuiller, Damien Georges, Robin Engler

See Also

BIOMOD_FormatingData, bm_ModelingOptions, BIOMOD_Modeling, BIOMOD_EnsembleModeling, BIOMOD_RangeSize
Other Main functions: BIOMOD_EnsembleModeling(), BIOMOD_FormatingData(), BIOMOD_LoadModels(), BIOMOD_Modeling(), BIOMOD_Projection(), BIOMOD_RangeSize()

Examples

library(terra)
# Load species occurrences (6 species available)
data(DataSpecies)
head(DataSpecies)

# Select the name of the studied species
myRespName <- 'GuloGulo'

# Get corresponding presence/absence data
myResp <- as.numeric(DataSpecies[, myRespName])

# Get corresponding XY coordinates
myRespXY <- DataSpecies[, c('X_WGS84', 'Y_WGS84')]  
# Load environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
data(bioclim_current)
myExpl <- terra::rast(bioclim_current)

# --------------------------------------------------------------- #
file.out <- paste0(myRespName, "/", myRespName, ".AllModels.models.out")
if (file.exists(file.out)) {
  myBiomodModelOut <- get(load(file.out))
} else {

  # Format Data with true absences
  myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
    expl.var = myExpl,
    resp.xy = myRespXY,
    resp.name = myRespName)

  # Model single models
  myBiomodModelOut <- BIOMOD_Modeling(bm.format = myBiomodData,
    modeling.id = 'AllModels',
    models = c('RF', 'GLM'),
    CV.strategy = 'random',
    CV.nb.rep = 2,
    CV.perc = 0.8,
    OPT.strategy = 'bigboss',
    metric.eval = c('TSS', 'ROC'),
    var.import = 3,
    seed.val = 42)
}

file.proj <- paste0(myRespName, "/proj_Current/", myRespName, ".Current.projection.out")
if (file.exists(file.proj)) {
  myBiomodProj <- get(load(file.proj))
} else {

  # Project single models
  myBiomodProj <- BIOMOD_Projection(bm.mod = myBiomodModelOut,
    proj.name = 'Current',
    new.env = myExpl,
    models.chosen = 'all',
    build.clamping.mask = TRUE)
}

file.EM <- paste0(myRespName, "/", myRespName, ".AllModels.ensemble.models.out")
if (file.exists(file.EM)) {
  myBiomodEM <- get(load(file.EM))
} else {

  # Model ensemble models
  myBiomodEM <- BIOMOD_EnsembleModeling(bm.mod = myBiomodModelOut,
BIOMOD_EnsembleModeling

models.chosen = 'all',
em.by = 'all',
em.algo = c('EMmean', 'EMca'),
metric.select = c('TSS'),
metric.select.thresh = c(0.7),
metric.eval = c('TSS', 'ROC'),
var.import = 3,
seed.val = 42)

# Project ensemble models (from single projections)
myBiomodEMProj <- BIOMOD_EnsembleForecasting(bm.em = myBiomodEM,
    bm.proj = myBiomodProj,
    models.chosen = 'all',
    metric.binary = 'all',
    metric.filter = 'all')

# Project ensemble models (building single projections)
myBiomodEMProj <- BIOMOD_EnsembleForecasting(bm.em = myBiomodEM,
    proj.name = 'CurrentEM',
    new.env = myExpl,
    models.chosen = 'all',
    metric.binary = 'all',
    metric.filter = 'all')

myBiomodEMProj
plot(myBiomodEMProj)

BIOMOD_EnsembleModeling

Create and evaluate an ensemble set of models and predictions

Description

This function allows to combine a range of models built with the BIOMOD_Modeling function in one (or several) ensemble model. Modeling uncertainty can be assessed as well as variables importance, ensemble predictions can be evaluated against original data, and created ensemble models can be projected over new conditions (see Details).

Usage

BIOMOD_EnsembleModeling(
    bm.mod,
    models.chosen = "all",
    em.by = "PA+run",
    em.algo,
metric.select = "all",
metric.select.thresh = NULL,
metric.select.table = NULL,
metric.select.dataset = NULL,
metric.eval = c("KAPPA", "TSS", "ROC"),
var.import = 0,
EMci.alpha = 0.05,
EMwmean.decay = "proportional",
nb.cpu = 1,
seed.val = NULL,
do.progress = TRUE,
prob.mean,
prob.median,
prob.cv,
prob.ci,
committee.averaging,
prob.mean.weight,
prob.mean.weight.decay,
prob.ci.alpha
)

Arguments

bm.mod  a BIOMOD.models.out object returned by the BIOMOD.Modeling function
models.chosen  a vector containing model names to be kept, must be either all or a sub-
selection of model names that can be obtained with the get_built_models
function
em.by  a character corresponding to the way kept models will be combined to build
the ensemble models, must be among all, algo, PA, PA+algo, PA+run
em.algo  a vector corresponding to the ensemble models that will be computed, must be
among 'EMmean', 'EMmedian', 'EMcv', 'EMci', 'EMca', 'EMwmean'
metric.select  a vector containing evaluation metric names to be used together with metric.select.thresh
to exclude single models based on their evaluation scores (for ensemble methods
like probability weighted mean or committee averaging). Must be among all
(same evaluation metrics than those of bm.mod), user.defined (and defined
through metric.select.table) or ROC, TSS, KAPPA, ACCURACY, BIAS, POD, FAR,
POFD, SR, CSI, ETS, HK, HSS, OR, ORSS
metric.select.thresh
  (optional, default NULL)
  A vector of numeric values corresponding to the minimum scores (one for
each metric.select) below which single models will be excluded from the
ensemble model building
metric.select.table
  (optional, default NULL)
  If metric.select = 'user.defined', a data.frame containing evaluation scores
calculated for each single models and that will be compared to metric.select.thresh
values to exclude some of them from the ensemble model building, with metric.select
rownames, and models.chosen colnames
metric.select.dataset
(optional, default 'validation' if possible). A character determining which dataset should be used to filter and/or weigh the ensemble models should be among 'evaluation', 'validation' or 'calibration'.

metric.eval
A vector containing evaluation metric names to be used, must be among ROC, TSS, KAPPA, ACCURACY, BIAS, POD, FAR, POFD, SR, CSI, ETS, HK, HSS, OR, ORSS

var.import
(optional, default NULL)
An integer corresponding to the number of permutations to be done for each variable to estimate variable importance

EMci.alpha
(optional, default 0.05)
A numeric value corresponding to the significance level to estimate confidence interval

EMwmean.decay
(optional, default proportional)
A value defining the relative importance of the weights (if 'EMwmean' was given to argument em.algo). A high value will strongly discriminate good models from the had ones (see Details), while proportional will attribute weights proportionally to the models evaluation scores

nb.cpus
(optional, default 1)
An integer value corresponding to the number of computing resources to be used to parallelize the single models predictions and the ensemble models computation

seed.val
(optional, default NULL)
An integer value corresponding to the new seed value to be set

do.progress
(optional, default TRUE)
A logical value defining whether the progress bar is to be rendered or not

prob.mean
(deprecated, please use em.algo instead)
A logical value defining whether to compute the mean probabilities across predictions or not

prob.median
(deprecated, please use em.algo instead)
A logical value defining whether to compute the median probabilities across predictions or not

prob.cv
(deprecated, please use em.algo instead)
A logical value defining whether to compute the coefficient of variation across predictions or not

prob.ci
(deprecated, please use em.algo instead)
A logical value defining whether to compute the confidence interval around the prob.mean ensemble model or not

committee.averaging
(deprecated, please use em.algo instead)
A logical value defining whether to compute the committee averaging across predictions or not

prob.mean.weight
(deprecated, please use em.algo instead)
A logical value defining whether to compute the weighted sum of probabilities across predictions or not
prob.mean.weight.decay
   \textit{(deprecated, please use EMwmean.decay instead)}
\textit{old argument name for EMwmean.decay}

prob.ci.alpha \textit{(deprecated, please use EMci.alpha instead)}
\textit{old argument name for EMci.alpha}

Details

\textbf{Models sub-selection} (models.chosen) Applying \texttt{get_built_models} function to the \texttt{bm.mod} object gives the names of the single models created with the \texttt{BIOMOD_Modeling} function. The \texttt{models.chosen} argument can take either a sub-selection of these single model names, or the all default value, to decide which single models will be used for the ensemble model building.

\textbf{Models assembly rules} (em.by) Single models built with the \texttt{BIOMOD_Modeling} function can be combined in 5 different ways to obtain ensemble models:

\begin{itemize}
  \item PA+run : each combination of pseudo-absence and repetition datasets is done, \textit{merging} algorithms together
  \item PA+algo : each combination of pseudo-absence and algorithm datasets is done, \textit{merging} repetitions together
  \item PA : pseudo-absence datasets are considered individually, \textit{merging} algorithms and repetitions together
  \item algo : algorithm datasets are considered individually, \textit{merging} pseudo-absence and repetitions together
  \item all : all models are combined into one
\end{itemize}

Hence, depending on the chosen method, the number of ensemble models built will vary.

Be aware that if no evaluation data was given to the \texttt{BIOMOD_FormatingData} function, some ensemble model evaluations may be biased due to difference in data used for single model evaluations. Be aware that all of these combinations are allowed, but some may not make sense depending mainly on how pseudo-absence datasets have been built and whether all of them have been used for all single models or not (see \texttt{PA.nb.absences} and \texttt{models.pa} parameters in \texttt{BIOMOD_FormatingData} and \texttt{BIOMOD_Modeling} functions respectively).

\textbf{Evaluation metrics} \begin{itemize}
  \item metric.select : the selected metrics must be chosen among the ones used within the \texttt{BIOMOD_Modeling} function to build the \texttt{model.output} object, unless metric.select = 'user.defined' and therefore values will be provided through the \texttt{metric.select.table} parameter.
  
  In the case of the selection of several metrics, they will be used at different steps of the ensemble modeling function:
  \begin{enumerate}
    \item remove \textit{low quality} single models, having a score lower than metric.select.thresh
    \item perform the binary transformation needed if 'EMca' was given to argument \texttt{em.algo}
    \item weight models if 'EMwmean' was given to argument \texttt{em.algo}
  \end{enumerate}
  \item metric.select.thresh : as many values as evaluation metrics selected with the metric.select parameter, and defining the corresponding quality thresholds below which the single models will be excluded from the ensemble model building.
  \item metric.select.table : a data.frame must be given if metric.select = 'user.defined' to allow the use of evaluation metrics other than those calculated within \texttt{biomod2}. The data.frame must contain as many columns as models.chosen with matching names, and as many rows as evaluation metrics to be used. The number of rows must match the
length of the metric.select.thresh parameter. The values contained in the data.frame will be compared to those defined in metric.select.thresh to remove low quality single models from the ensemble model building.

- **metric.select.dataset**: a character determining the dataset which evaluation metric should be used to filter and/or weigh the ensemble models. Should be among evaluation, validation or calibration. By default BIOMOD_EnsembleModeling will use the validation dataset unless no validation is available in which case calibration dataset are used.

- **metric.eval**: the selected metrics will be used to validate/evaluate the ensemble models built

**Ensemble-models algorithms** The set of models to be calibrated on the data.

6 modeling techniques are currently available:

- **EMmean**: Mean of probabilities over the selected models. Old name: prob.mean
- **EMmedian**: Median of probabilities over the selected models
  The median is less sensitive to outliers than the mean, however it requires more computation time and memory as it loads all predictions (on the contrary to the mean or the weighted mean). Old name: prob.median
- **EMcv**: Coefficient of variation (sd / mean) of probabilities over the selected models
  This model is not scaled. It will be evaluated like all other ensemble models although its interpretation will be obviously different. CV is a measure of uncertainty rather a measure of probability of occurrence. If the CV gets a high evaluation score, it means that the uncertainty is high where the species is observed (which might not be a good feature of the model). The lower is the score, the better are the models. CV is a nice complement to the mean probability. Old name: prob.cv
- **EMci & EMci.alpha**: Confidence interval around the mean of probabilities of the selected models
  It is also a nice complement to the mean probability. It creates 2 ensemble models:
  - **LOWER**: there is less than 100 * EMci.alpha / 2 % of chance to get probabilities lower than the given ones
  - **UPPER**: there is less than 100 * EMci.alpha / 2 % of chance to get probabilities upper than the given ones
  These intervals are calculated with the following function:
  \[
  I_c = [\bar{x} - \frac{t_n sd}{\sqrt{n}}; \bar{x} + \frac{t_n sd}{\sqrt{n}}]
  \]
  Old parameter name: prob.ci & prob.ci.alpha
- **EMca**: Probabilities from the selected models are first transformed into binary data according to the thresholds defined when building the model.output object with the BIOMOD_Modeling function, maximizing the evaluation metric score over the testing dataset. The committee averaging score is obtained by taking the average of these binary predictions. It is built on the analogy of a simple vote:
  - each single model votes for the species being either present (1) or absent (0)
  - the sum of 1 is then divided by the number of single models voting
  The interesting feature of this measure is that it gives both a prediction and a measure of uncertainty. When the prediction is close to 0 or 1, it means that all models agree to predict 0 or 1 respectively. When the prediction is around 0.5, it means that half the
models predict 1 and the other half 0.
Old parameter name: committee.averaging

• EM\text{wmean} \& EM\text{wmean.decay} : Probabilities from the selected models are weighted according to their evaluation scores obtained when building the model output object with the BIOMOD_Modeling function (better a model is, more importance it has in the ensemble) and summed.
Old parameter name: prob.mean.weight \& prob.mean.weight.decay

The EM\text{wmean.decay} is the ratio between a weight and the next or previous one. The formula is : 
\[ W = W(-1) \times EM\text{wmean.decay} \]
For example, with the value of 1.6 and 4 weights wanted, the relative importance of the weights will be 
\[ 1/1.6/2.56(-1.6*1.6)/4.096(-2.56*1.6) \] from the weakest to the strongest, and gives 
\[ 0.11/0.17/0.275/0.445 \] considering that the sum of the weights is equal to one. The lower the EM\text{wmean.decay}, the smoother the differences between the weights enhancing a weak discrimination between models.

If EM\text{wmean.decay} = 'proportional', the weights are assigned to each model proportionally to their evaluation scores. The discrimination is fairer than using the decay method where close scores can have strongly diverging weights, while the proportional method would assign them similar weights.

It is also possible to define the EM\text{wmean.decay} parameter as a function that will be applied to single models scores and transform them into weights. For example, if EM\text{wmean.decay} = function(x) \( x^2 \), the squared of evaluation score of each model will be used to weight the models predictions.

Value

A BIOMOD.ensemble.models.out object containing models outputs, or links to saved outputs.
Models outputs are stored out of R (for memory storage reasons) in 2 different folders created in the current working directory :

1. a models folder, named after the resp.name argument of BIOMOD_FormatingData, and containing all ensemble models
2. a hidden folder, named .BIOMOD_DATA, and containing outputs related files (original dataset, calibration lines, pseudo-absences selected, predictions, variables importance, evaluation values...), that can be retrieved with get [...] or load functions, and used by other biomod2 functions, like BIOMOD_EnsembleForecasting

Author(s)

Wilfried Thuiller, Damien Georges, Robin Engler

See Also

BIOMOD_FormatingData, bm_ModelingOptions, bm_CrossValidation, bm_VariablesImportance, BIOMOD_Modeling, BIOMOD_EnsembleForecasting, bm_PlotEvalMean, bm_PlotEvalBoxplot, bm_PlotVarImpBoxplot, bm_PlotResponseCurves

Other Main functions: BIOMOD_EnsembleForecasting(), BIOMOD_FormatingData(), BIOMOD_LoadModels(), BIOMOD_Modeling(), BIOMOD_Projection(), BIOMOD_RangeSize()
Examples

```r
library(terra)
# Load species occurrences (6 species available)
data(DataSpecies)
head(DataSpecies)

# Select the name of the studied species
myRespName <- 'GuloGulo'

# Get corresponding presence/absence data
myResp <- as.numeric(DataSpecies[, myRespName])

# Get corresponding XY coordinates
myRespXY <- DataSpecies[, c('X_WGS84', 'Y_WGS84')]

# Load environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
data(bioclim_current)
myExpl <- terra::rast(bioclim_current)

## ----------------------------------------------------------------------- #
file.out <- paste0(myRespName, '/', myRespName, '.AllModels.models.out')
if (file.exists(file.out)) {
  myBiomodModelOut <- get(load(file.out))
} else {

  # Format Data with true absences
  myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
                                        expl.var = myExpl,
                                        resp.xy = myRespXY,
                                        resp.name = myRespName)

  # Model single models
  myBiomodModelOut <- BIOMOD_Modeling(bm.format = myBiomodData,
                                       modeling.id = 'AllModels',
                                       models = c('RF', 'GLM'),
                                       CV.strategy = 'random',
                                       CV.nb.rep = 2,
                                       CV.perc = 0.8,
                                       OPT.strategy = 'bigboss',
                                       metric.eval = c('TSS', 'ROC'),
                                       var.import = 3,
                                       seed.val = 42)
}

## ----------------------------------------------------------------------- #
# Model ensemble models
myBiomodEM <- BIOMOD_EnsembleModeling(bm.mod = myBiomodModelOut,
                                       models.chosen = 'all',
                                       em.by = 'all',
                                       
```

em.algo = c('EMmean', 'EMca'),
metric.select = c('TSS'),
metric.select.thresh = c(0.7),
metric.eval = c('TSS', 'ROC'),
var.import = 3,
seed.val = 42)

myBiomodEM

# Get evaluation scores & variables importance
get_evaluations(myBiomodEM)
get_variables_importance(myBiomodEM)

# Represent evaluation scores
bm_PlotEvalMean(bm.out = myBiomodEM, dataset = 'calibration')
bm_PlotEvalBoxplot(bm.out = myBiomodEM, group.by = c('algo', 'algo'))

# Represent variables importance
# bm_PlotVarImpBoxplot(bm.out = myBiomodEM, group.by = c('expl.var', 'algo', 'algo'))
# bm_PlotVarImpBoxplot(bm.out = myBiomodEM, group.by = c('expl.var', 'algo', 'merged.by.PA'))
# bm_PlotVarImpBoxplot(bm.out = myBiomodEM, group.by = c('algo', 'expl.var', 'merged.by.PA'))

# Represent response curves
# bm_PlotResponseCurves(bm.out = myBiomodEM,
# models.chosen = get_built_models(myBiomodEM),
# fixed.var = 'median')
# bm_PlotResponseCurves(bm.out = myBiomodEM,
# models.chosen = get_built_models(myBiomodEM),
# fixed.var = 'min')
# bm_PlotResponseCurves(bm.out = myBiomodEM,
# models.chosen = get_built_models(myBiomodEM, algo = 'EMmean'),
# fixed.var = 'median',
# do.bivariate = TRUE)

BIOMOD_FormatingData Format input data, and select pseudo-absences if wanted, for usage in biomod2

Description

This function gathers together all input data needed (xy, presences/absences, explanatory variables, and the same for evaluation data if available) to run biomod2 models. It allows to select pseudo-absences if no absence data is available, with different strategies (see Details).

Usage

BIOMOD_FormatingData(
  resp.name,
  resp.var,
expl.var,
dir.name = ".",
resp.xy = NULL,
eval.resp.var = NULL,
eval.expl.var = NULL,
eval.resp.xy = NULL,
PA.nb.rep = 0,
PA.nb.absences = 1000,
PA.strategy = NULL,
PA.dist.min = 0,
PA.dist.max = NULL,
PA.sre.quant = 0.025,
PA.user.table = NULL,
na.rm = TRUE,
filter.raster = FALSE
)

Arguments

resp.name

a character corresponding to the species name

resp.var

a vector, a SpatVector without associated data (if presence-only), or a SpatVector object containing binary data (0: absence, 1: presence, NA: indeterminate) for a single species that will be used to build the species distribution model(s)

Note that old format from sp are still supported such as SpatialPoints (if presence-only) or SpatialPointsDataFrame object containing binary data.

eval.resp.var

a matrix, data.frame, SpatVector or SpatRaster object containing the explanatory variables (in columns or layers) that will be used to build the species distribution model(s)

Note that old format from raster and sp are still supported such as RasterStack and SpatialPointsDataFrame objects.

dir.name

(optional, default .)

A character corresponding to the modeling folder

resp.xy

(optional, default NULL)

If resp.var is a vector, a 2-columns matrix or data.frame containing the corresponding X and Y coordinates that will be used to build the species distribution model(s)

eval.resp.var

(optional, default NULL)

A vector, a SpatVector without associated data (if presence-only), or a SpatVector object containing binary data (0: absence, 1: presence, NA: indeterminate) for a single species that will be used to evaluate the species distribution model(s) with independent data

Note that old format from sp are still supported such as SpatialPoints (if presence-only) or SpatialPointsDataFrame object containing binary data.

eval.expl.var

(optional, default NULL)

A matrix, data.frame, SpatVector or SpatRaster object containing the explanatory variables (in columns or layers) that will be used to evaluate the species distribution model(s) with independent data.
Note that old format from raster and sp are still supported such as RasterStack and SpatialPointsDataFrame objects.

eval.resp.xy (optional, default NULL)
If resp.var is a vector, a 2-columns matrix or data.frame containing the corresponding X and Y coordinates that will be used to evaluate the species distribution model(s) with independent data

PA.nb.rep (optional, default 0)
If pseudo-absence selection, an integer corresponding to the number of sets (repetitions) of pseudo-absence points that will be drawn

PA.nb.absences (optional, default 0)
If pseudo-absence selection, and PA.strategy = 'random' or PA.strategy = 'sre' or PA.strategy = 'disk', an integer corresponding to the number of pseudo-absence points that will be selected for each pseudo-absence repetition (true absences included). It can also be a vector of the same length as PA.nb.rep containing integer values corresponding to the different numbers of pseudo-absences to be selected

PA.strategy (optional, default NULL)
If pseudo-absence selection, a character defining the strategy that will be used to select the pseudo-absence points. Must be random, sre, disk or user.defined (see Details)

PA.dist.min (optional, default 0)
If pseudo-absence selection and PA.strategy = 'disk', a numeric defining the minimal distance to presence points used to make the disk pseudo-absence selection (in meters, see Details)

PA.dist.max (optional, default 0)
If pseudo-absence selection and PA.strategy = 'disk', a numeric defining the maximal distance to presence points used to make the disk pseudo-absence selection (in meters, see Details)

PA.sre.quant (optional, default 0)
If pseudo-absence selection and PA.strategy = 'sre', a numeric between 0 and 0.5 defining the half-quantile used to make the sre pseudo-absence selection (see Details)

PA.user.table (optional, default NULL)
If pseudo-absence selection and PA.strategy = 'user.defined', a matrix or data.frame with as many rows as resp.var values, as many columns as PA.nb.rep, and containing TRUE or FALSE values defining which points will be used to build the species distribution model(s) for each repetition (see Details)

na.rm (optional, default TRUE)
A logical value defining whether points having one or several missing values for explanatory variables should be removed from the analysis or not

filter.raster (optional, default FALSE)
If expl.var is of raster type, a logical value defining whether resp.var is to be filtered when several points occur in the same raster cell
Details

This function gathers and formats all input data needed to run biomod2 models. It supports different kind of inputs (e.g. matrix, SpatVector, SpatRaster) and provides different methods to select pseudo-absences if needed.

Concerning explanatory variables and XY coordinates:

- if SpatRaster, RasterLayer or RasterStack provided for expl.var or eval.expl.var, biomod2 will extract the corresponding values from XY coordinates provided:
  - either through resp.xy or eval.resp.xy respectively
  - or resp.var or eval.resp.var, if provided as SpatVector or SpatialPointsDataFrame
  Be sure to give the objects containing XY coordinates in the same projection system than the raster objects!

- if data.frame or matrix provided for expl.var or eval.expl.var, biomod2 will simply merge it (cbind) with resp.var without considering XY coordinates.
  Be sure to give explanatory and response values in the same row order!

Concerning pseudo-absence selection (see bm_PseudoAbsences):

- if both presence and absence data are available, and there is enough absences: set PA.nb.rep = 0 and no pseudo-absence will be selected.
- if no absence data is available, several pseudo-absence repetitions are recommended (to estimate the effect of pseudo-absence selection), as well as high number of pseudo-absence points.
  Be sure not to select more pseudo-absence points than maximum number of pixels in the studied area!

- it is possible now to create several pseudo-absence repetitions with different number of points, BUT with the same sampling strategy.

Response variable biomod2 models single species at a time (no multi-species). Hence, resp.var must be a uni-dimensional object (either a vector, a one-column matrix, data.frame, a SpatVector (without associated data - if presence-only), a SpatialPoints (if presence-only), a SpatialPointsDataFrame or SpatVector object), containing values among:

- 1 : presences
- 0 : true absences (if any)
- NA : no information point (might be used to select pseudo-absences if any)

If no true absences are available, pseudo-absence selection must be done.
If resp.var is a non-spatial object (vector, matrix or data.frame), XY coordinates must be provided through resp.xy.
If pseudo-absence points are to be selected, NA points must be provided in order to select pseudo-absences among them.
Explanatory variables  Factorial variables are allowed, but might lead to some pseudo-absence strategy or models omissions (e.g. sre).

Evaluation data  Although biomod2 provides tools to automatically divide dataset into calibration and validation parts through the modeling process (see CV.[…]parameters in BIOMOD_Modeling function ; or bm_CrossValidation function), it is also possible (and strongly advised) to directly provide two independent datasets, one for calibration/validation and one for evaluation.

Pseudo-absence selection (see bm_PseudoAbsences)  If no true absences are available, pseudo-absences must be selected from the background data, meaning data there is no information whether the species of interest occurs or not. It corresponds either to the remaining pixels of the expl.var (if provided as a SpatRaster or RasterSatck) or to the points identified as NA in resp.var (if expl.var provided as a matrix or data.frame). Several methods are available to do this selection:

- **random** all points of initial background are pseudo-absence candidates. PA.nb.absences are drawn randomly, for each PA.nb.rep requested.
- **sre** pseudo-absences have to be selected in conditions (combination of explanatory variables) that differ in a defined proportion (PA.sre.quant) from those of presence points. A Surface Range Envelop model is first run over the species of interest (see bm_SRE), and pseudo-absences are selected outside this envelop. This case is appropriate when all the species climatic niche has been sampled, otherwise it may lead to over-optimistic model evaluations and predictions!
- **disk** pseudo-absences are selected within circles around presence points defined by PA.dist.min and PA.dist.max distance values (in meters). It allows to select pseudo-absence points that are not too close to (avoid same niche and pseudo-replication) or too far (localized sampling strategy) from presences.
- **user.defined** pseudo-absences are defined in advance and given as data.frame through the PA.user.table parameter.

Value

A BIOMOD.formated.data object that can be used to build species distribution model(s) with the BIOMOD_Modeling function. print/show, plot and summary functions are available to have a summary of the created object.

Author(s)

Damien Georges, Wilfried Thuiller

See Also

- bm_PseudoAbsences, BIOMOD_Modeling
- Other Main functions: BIOMOD_EnsembleForecasting(), BIOMOD_EnsembleModeling(), BIOMOD_LoadModels(), BIOMOD_Modeling(), BIOMOD_Projection(), BIOMOD_RangeSize()

Examples

library(terra)

# Load species occurrences (6 species available)
data(DataSpecies)
head(DataSpecies)

# Select the name of the studied species
myRespName <- 'GuloGulo'

# Get corresponding presence/absence data
myResp <- as.numeric(DataSpecies[, myRespName])

# Get corresponding XY coordinates
myRespXY <- DataSpecies[, c('X_WGS84', 'Y_WGS84')]

# Load environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
data(bioclim_current)
myExpl <- terra::rast(bioclim_current)

# Format Data with true absences
myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
expl.var = myExpl,
resp.xy = myRespXY,
resp.name = myRespName)

myBiomodData
summary(myBiomodData)
plot(myBiomodData)

# Transform true absences into potential pseudo-absences
myResp.PA <- ifelse(myResp == 1, 1, NA)

# Format Data with pseudo-absences:
# random method
# myBiomodData.r <- BIOMOD_FormatingData(resp.var = myResp.PA,
# expl.var = myExpl,
# resp.xy = myRespXY,
# resp.name = myRespName,
# PA.nb.rep = 4,
# PA.nb.absences = 1000,
# PA.strategy = 'random')
#
# disk method
# myBiomodData.d <- BIOMOD_FormatingData(resp.var = myResp.PA,
# expl.var = myExpl,
# resp.xy = myRespXY,
# resp.name = myRespName,
# PA.nb.rep = 4,
# PA.nb.absences = 500,
# PA.strategy = 'disk',
# PA.dist.min = 5,
# PA.dist.max = 35)
```r
# # Format Data with pseudo-absences: SRE method
# myBiomodData.s <- BIOMOD_FormatingData(resp.var = myResp.PA,
#   expl.var = myExpl,
#   resp.xy = myRespXY,
#   resp.name = myRespName,
#   PA.nb.rep = 4,
#   PA.nb.absences = 1000,
#   PA.strategy = 'sre',
#   PA.sre.quant = 0.025)
#
# # Format Data with pseudo-absences: user.defined method
# myPAtable <- data.frame(PA1 = ifelse(myResp == 1, TRUE, FALSE),
#   PA2 = ifelse(myResp == 1, TRUE, FALSE))
# for (i in 1:ncol(myPAtable)) myPAtable[sample(which(myPAtable[, i] == FALSE), 500), i] = TRUE
# myBiomodData.u <- BIOMOD_FormatingData(resp.var = myResp.PA,
#   expl.var = myExpl,
#   resp.xy = myRespXY,
#   resp.name = myRespName,
#   PA.strategy = 'user.defined',
#   PA.user.table = myPAtable)
#
# myBiomodData.r
# myBiomodData.d
# myBiomodData.s
# myBiomodData.u
# plot(myBiomodData.r)
# plot(myBiomodData.d)
# plot(myBiomodData.s)
# plot(myBiomodData.u)

# Select multiple sets of pseudo-absences
# Transform true absences into potential pseudo-absences
# myResp.PA <- ifelse(myResp == 1, 1, NA)

# # Format Data with pseudo-absences: random method
# myBiomodData.multi <- BIOMOD_FormatingData(resp.var = myResp.PA,
#   expl.var = myExpl,
#   resp.xy = myRespXY,
#   resp.name = myRespName,
#   PA.nb.rep = 4,
#   PA.nb.absences = c(1000, 500, 500, 200),
#   PA.strategy = 'random')
# myBiomodData.multi
# summary(myBiomodData.multi)
# plot(myBiomodData.multi)
```
**BIOMOD_LoadModels**  
*Load species distribution models built with biomod2*

**Description**  
This function loads individual models built with `BIOMOD_Modeling` or `BIOMOD_EnsembleModeling` functions.

**Usage**  
```r
BIOMOD_LoadModels(
  bm.out,
  full.name = NULL,
  PA = NULL,
  run = NULL,
  algo = NULL,
  merged.by.PA = NULL,
  merged.by.run = NULL,
  merged.by.algo = NULL,
  filtered.by = NULL
)
```

**Arguments**

- `bm.out`  
  a `BIOMOD.models.out` or `BIOMOD.ensemble.models.out` object that can be obtained with the `BIOMOD_Modeling` or `BIOMOD_EnsembleModeling` functions

- `full.name` *(optional, default NULL)*  
  A vector containing model names to be kept, must be either all or a sub-selection of model names that can be obtained with the `get_built_models` function

- `PA` *(optional, default NULL)*  
  A vector containing pseudo-absence set to be loaded, must be among PA1, PA2, ... , allData

- `run` *(optional, default NULL)*  
  A vector containing repetition set to be loaded, must be among RUN1, RUN2, ... , allRun

- `algo` *(optional, default NULL)*  
  A character containing algorithm to be loaded, must be either ANN, CTA, FDA, GAM, GBM, GLM, MARS, MAXENT, MAXNET, RF, SRE, XGBOOST

- `merged.by.PA` *(optional, default NULL)*  
  A vector containing merged pseudo-absence set to be loaded, must be among PA1, PA2, ... , mergedData

- `merged.by.run` *(optional, default NULL)*  
  A vector containing merged repetition set to be loaded, must be among RUN1, RUN2, ... , mergedRun
merged.by.algo  *(optional, default NULL)*
A character containing merged algorithm to be loaded, must be among ANN, CTA, FDA, GAM, GBM, GLM, MARS, MAXENT, MAXNET, RF, SRE, XGBOOST, mergedAlgo

filtered.by  *(optional, default NULL)*
A vector containing evaluation metric selected to filter single models to build the ensemble models, must be among ROC, TSS, KAPPA, ACCURACY, BIAS, POD, FAR, POFD, SR, CSI, ETS, HK, HSS, OR, ORSS

**Details**

This function might be of particular use to load models and make response plot analyses.

Running the function providing only `bm.out` argument will load all models built by the `BIOMOD_Modeling` or `BIOMOD_EnsembleModeling` function, but a subselection of models can be done using the additional arguments (`full.name`, `PA`, `run`, `algo`, `merged.by.PA`, `merged.by.run`, `merged.by.algo`, `filtered.by`).

**Value**

A vector containing the names of the loaded models.

**Author(s)**

Damien Georges

**See Also**

`BIOMOD_Modeling`, `BIOMOD_EnsembleModeling`

Other Main functions: `BIOMOD_EnsembleForecasting()`, `BIOMOD_EnsembleModeling()`, `BIOMOD_FormatingData()`, `BIOMOD_Modeling()`, `BIOMOD_Projection()`, `BIOMOD_RangeSize()`

**Examples**

```r
library(terra)

# Load species occurrences (6 species available)
data(DataSpecies)
head(DataSpecies)

# Select the name of the studied species
myRespName <- 'GuloGulo'

# Get corresponding presence/absence data
myResp <- as.numeric(DataSpecies[, myRespName])

# Get corresponding XY coordinates
myRespXY <- DataSpecies[, c('X_WGS84', 'Y_WGS84')]

# Load environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
```
```r
data(bioclim_current)
myExpl <- terra::rast(bioclim_current)

# ---------------------------------------------------------------
file.out <- paste0(myRespName, "/", myRespName, ".AllModels.models.out")
if (file.exists(file.out)) {
  myBiomodModelOut <- get(load(file.out))
} else {

  # Format Data with true absences
  myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
                                        expl.var = myExpl,
                                        resp.xy = myRespXY,
                                        resp.name = myRespName)

  # Model single models
  myBiomodModelOut <- BIOMOD_Modeling(bm.format = myBiomodData,
                                       modeling.id = "AllModels",
                                       models = c("RF", "GLM"),
                                       CV.strategy = "random",
                                       CV.nb.rep = 2,
                                       CV.perc = 0.8,
                                       OPT.strategy = "bigboss",
                                       metric.eval = c("TSS", "ROC"),
                                       var.import = 3,
                                       seed.val = 42)
}

# ---------------------------------------------------------------
# Loading some models built
BIOMOD_LoadModels(bm.out = myBiomodModelOut, algo = "RF")
```

---

### BIOMOD_Modeling

**Run a range of species distribution models**

### Description

This function allows to calibrate and evaluate a range of modeling techniques for a given species distribution. The dataset can be split up in calibration/validation parts, and the predictive power of the different models can be estimated using a range of evaluation metrics (see Details).

### Usage

```r
BIOMOD_Modeling(
  bm.format,
```
modeling.id = as.character(format(Sys.time(), "%s")),
models = c("ANN", "CTA", "FDA", "GAM", "GBM", "GLM", "MARS", "MAXENT", "MAXNET", "RF", "SRE", "XGBOOST"),
models.pa = NULL,
CV.strategy = "random",
CV.nb.rep = 1,
CV.perp = NULL,
CV.k = NULL,
CV.balance = NULL,
CV.env.var = NULL,
CV.strat = NULL,
CV.do.full.models = TRUE,
OPT.data.type = "binary",
OPT.strategy = "default",
OPT.user.val = NULL,
OPT.user.base = "bigboss",
OPT.user = NULL,
bm.options,
nb.rep,
data.split.perc,
data.split.table,
do.full.models,
weights = NULL,
prevalence = NULL,
metric.eval = c("KAPPA", "TSS", "ROC"),
var.import = 0,
scale.models = FALSE,
nb.cpu = 1,
seed.val = NULL,
do.progress = TRUE
)

Arguments

**bm.format**

A BIOMOD.formated.data or BIOMOD.formated.data.PA object returned by the BIOMOD_FormatingData function

**modeling.id**

A character corresponding to the name (ID) of the simulation set *(a random number by default)*

**models**

A vector containing model names to be computed, must be among ANN, CTA, FDA, GAM, GBM, GLM, MARS, MAXENT, MAXNET, RF, SRE, XGBOOST

**models.pa** *(optional, default NULL)*

A list containing for each model a vector defining which pseudo-absence datasets are to be used, must be among colnames(bm.format@PA.table)

**CV.strategy**

A character corresponding to the cross-validation selection strategy, must be among random, kfold, block, strat, env or user.defined

**CV.nb.rep** *(optional, default 0)*
If `strategy = 'random'` or `strategy = 'kfold'`, an integer corresponding to the number of sets (repetitions) of cross-validation points that will be drawn.

**CV.perc** *(optional, default 0)*
If `strategy = 'random'`, a numeric between 0 and 1 defining the percentage of data that will be kept for calibration.

**CV.k** *(optional, default 0)*
If `strategy = 'kfold'` or `strategy = 'strat'` or `strategy = 'env'`, an integer corresponding to the number of partitions.

**CV.balance** *(optional, default 'presences')*
If `strategy = 'strat'` or `strategy = 'env'`, a character corresponding to how data will be balanced between partitions, must be either `presences` or `absences`.

**CV.env.var** *(optional)*
If `strategy = 'env'`, a character corresponding to the environmental variables used to build the partition. `k` partitions will be built for each environmental variables. By default the function uses all environmental variables available.

**CV.strat** *(optional, default 'both')*
If `strategy = 'env'`, a character corresponding to how data will partitioned along gradient, must be among `x`, `y`, `both`.

**CV.user.table** *(optional, default NULL)*
If `strategy = 'user.defined'`, a matrix or data.frame defining for each repetition (in columns) which observation lines should be used for models calibration (TRUE) and validation (FALSE).

**CV.do.full.models** *(optional, default TRUE)*
A logical value defining whether models should be also calibrated and validated over the whole dataset (and pseudo-absence datasets) or not.

**OPT.data.type**
a character corresponding to the data type to be used, must be either `binary`, `binary.PA`, `abundance`, `compositional`.

**OPT.strategy**
a character corresponding to the method to select models’ parameters values, must be either `default`, `bigboss`, `user.defined`, `tuned`.

**OPT.user.val** *(optional, default NULL)*
A list containing parameters values for some (all) models.

**OPT.user.base** *(optional, default bigboss)*
A character, `default` or `bigboss` used when `OPT.strategy = 'user.defined'`. It sets the bases of parameters to be modified by user defined values.

**OPT.user** *(optional, default TRUE)*
A BIOMOD.models.options object returned by the `bm_ModelingOptions` function.

**bm.options**
a BIOMOD.models.options object returned by the `bm_ModelingOptions` function.

**nb.rep**
Deprecated, now called `CV.nb.rep`.

**data.split.perc**
Deprecated, now called `CV.perc`.
data.split.table
    deprecated, now called CV.user.table

do.full.models
    deprecated, now called CV.do.full.models

weights
    (optional, default NULL)
    A vector of numeric values corresponding to observation weights (one per
    observation, see Details)

prevalence
    (optional, default NULL)
    A numeric between 0 and 1 corresponding to the species prevalence to build
    'weighted response weights' (see Details)

metric.eval
    a vector containing evaluation metric names to be used, must be among ROC,
    TSS, KAPPA, ACCURACY, BIAS, POD, FAR, POFD, SR, CSI, ETS, HK, HSS, OR, ORSS,
    BOYCE, MPA

var.import
    (optional, default NULL)
    An integer corresponding to the number of permutations to be done for each
    variable to estimate variable importance

scale.models
    (optional, default FALSE)
    A logical value defining whether all models predictions should be scaled with
    a binomial GLM or not

nb.cpu
    (optional, default 1)
    An integer value corresponding to the number of computing resources to be
    used to parallelize the single models computation

seed.val
    (optional, default NULL)
    An integer value corresponding to the new seed value to be set

do.progress
    (optional, default TRUE)
    A logical value defining whether the progress bar is to be rendered or not

Details

**bm.format** If pseudo absences have been added to the original dataset (see `BIOMOD_FormatingData`),
PA.nb.rep *(nb.rep + 1) models will be created.

**models** The set of models to be calibrated on the data. 12 modeling techniques are currently
available:

- ANN: Artificial Neural Network (`nnet`)
- CTA: Classification Tree Analysis (`rpart`)
- FDA: Flexible Discriminant Analysis (`fda`)
- GAM: Generalized Additive Model (`gam`, `gam` or `bam`)
  (see `bm_ModelingOptions` for details on algorithm selection)
- GBM: Generalized Boosting Model, or usually called Boosted Regression Trees (`gbm`)
- GLM: Generalized Linear Model (`glm`)
- MARS: Multiple Adaptive Regression Splines (`earth`)
- MAXENT: Maximum Entropy (https://biodiversityinformatics.amnh.org/open-source/maxent/)
- MAXNET: Maximum Entropy (`maxnet`)
- RF: Random Forest (`randomForest`)
models.pa Different models might respond differently to different numbers of pseudo-absences. It is possible to create sets of pseudo-absences with different numbers of points (see BIOMOD_FormatingData) and to assign only some of these datasets to each single model.

CV[... parameters] Different methods are available to calibrate/validate the single models (see bm_CrossValidation).

OPT.[... parameters] Different methods are available to parameterize the single models (see bm_ModelingOptions and BIOMOD.options.dataset). Note that only binary data type is allowed currently.

weights & prevalence More or less weight can be given to some specific observations.

metric.eval simple • POD: Probability of detection (hit rate)
  • FAR: False alarm ratio
  • POFD: Probability of false detection (fall-out)
  • SR: Success ratio
  • ACCURACY: Accuracy (fraction correct)
  • BIAS: Bias score (frequency bias)

metric.eval complex • ROC: Relative operating characteristic
  • TSS: True skill statistic (Hanssen and Kuipers discriminant, Peirce’s skill score)
  • KAPPA: Cohen’s Kappa (Heidke skill score)
  • OR: Odds Ratio
  • ORSS: Odds ratio skill score (Yule’s Q)
  • CSI: Critical success index (threat score)
  • ETS: Equitable threat score (Gilbert skill score)

presence-only • BOYCE: Boyce index
• MPA: Minimal predicted area (cutoff optimising MPA to predict 90% of presences)

Optimal value of each method can be obtained with the `get_optim_value` function. Several
evaluation metrics can be selected. Please refer to the CAWRC website (section "Methods for
dichotomous forecasts") to get detailed description of each metric. Results after modeling can
be obtained through the `get_evaluations` function.

Evaluation metrics are calculated on the calibrating data (column calibration), on the
cross-validation data (column validation) or on the evaluation data (column evaluation).

For cross-validation data, see CV.[...]. For evaluation data, see eval.[...].

`var.import` A value characterizing how much each variable has an impact on each model predic-
tions can be calculated by randomizing the variable of interest and computing the correlation
between original and shuffled variables (see `bm_VariablesImportance`).

`scale.models` This parameter is quite experimental and it is recommended not to use it. It
may lead to reduction in projection scale amplitude. Some categorical models always have
to be scaled (FDA, ANN), but it may be interesting to scale all computed models to ensure
comparable predictions (0~1000 range). It might be particularly useful when doing ensemble
forecasting to remove the scale prediction effect (the more extended projections are, the more
they influence ensemble forecasting results).

**Value**

A BIOMOD.models.out object containing models outputs, or links to saved outputs.
Models outputs are stored out of R (for memory storage reasons) in 2 different folders created in
the current working directory:

1. a models folder, named after the resp.name argument of `BIOMOD_FormatingData`, and con-
taining all calibrated models for each repetition and pseudo-absence run
2. a hidden folder, named .BIOMOD_DATA, and containing outputs related files (original dataset,
calibration lines, pseudo-absences selected, predictions, variables importance, evaluation val-
ues...), that can be retrieved with `get_[...]` or load functions, and used by other biomod2
functions, like `BIOMOD_Projection` or `BIOMOD_EnsembleModeling`.

**Author(s)**

Wilfried Thuiller, Damien Georges, Robin Engler

**See Also**

`glm, gam, gam, bam, gbm, rpart, nnet, fda, earth, randomForest, maxnet, xgboost, BIOMOD_FormatingData,
bm_ModelingOptions, bm_Tuning, bm_CrossValidation, bm_VariablesImportance, BIOMOD_Projection,
BIOMOD_EnsembleModeling, bm_PlotEvalMean, bm_PlotEvalBoxplot, bm_PlotVarImpBoxplot,
bm_PlotResponseCurves`

Other Main functions: `BIOMOD_EnsembleForecasting()`, `BIOMOD_EnsembleModeling()`, `BIOMOD_FormatingData()
`, `BIOMOD_LoadModels()`, `BIOMOD_Projection()`, `BIOMOD_RangeSize()`

**Examples**

```r
library(terra)
```
# Load species occurrences (6 species available)
data(DataSpecies)
head(DataSpecies)

# Select the name of the studied species
myRespName <- 'GuloGulo'

# Get corresponding presence/absence data
myResp <- as.numeric(DataSpecies[, myRespName])

# Get corresponding XY coordinates
myRespXY <- DataSpecies[, c('X_WGS84', 'Y_WGS84')]

# Load environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
data(bioclim_current)
myExpl <- terra::rast(bioclim_current)

# Format Data with true absences
myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
                                      expl.var = myExpl,
                                      resp.xy = myRespXY,
                                      resp.name = myRespName)

# Model single models
myBiomodModelOut <- BIOMOD_Modeling(bm.format = myBiomodData,
                                      modeling.id = 'AllModels',
                                      models = c('RF', 'GLM'),
                                      CV.strategy = 'random',
                                      CV.nb.rep = 2,
                                      CV.perc = 0.8,
                                      OPT.strategy = 'bigboss',
                                      metric.eval = c('TSS', 'ROC'),
                                      var.import = 2,
                                      seed.val = 42)

# Get evaluation scores & variables importance
get_evaluations(myBiomodModelOut)
get_variables_importance(myBiomodModelOut)

# Represent evaluation scores
bm_PlotEvalMean(bm.out = myBiomodModelOut, dataset = 'calibration')
bm_PlotEvalMean(bm.out = myBiomodModelOut, dataset = 'validation')
bm_PlotEvalBoxplot(bm.out = myBiomodModelOut, group.by = c('algo', 'run'))

# Represent variables importance
# bm_PlotVarImpBoxplot(bm.out = myBiomodModelOut, group.by = c('expl.var', 'algo', 'algo'))
# bm_PlotVarImpBoxplot(bm.out = myBiomodModelOut, group.by = c('expl.var', 'algo', 'run'))
# BIOMOD_Projection

## BIOMOD_Projection

Project a range of calibrated species distribution models onto new environment

### Description

This function allows to project a range of models built with the `BIOMOD_Modeling` function onto new environmental data (which can represent new areas, resolution or time scales for example).

### Usage

```r
BIOMOD_Projection(
  bm.mod,  
  proj.name,  
  new.env,  
  new.env.xy = NULL,  
  models.chosen = "all",  
  metric.binary = NULL,  
  metric.filter = NULL,  
  compress = TRUE,  
  build.clamping.mask = TRUE,  
  nb.cpu = 1,  
  seed.val = NULL,  
  ...
)
```

### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bm.mod</code></td>
<td>A <code>BIOMOD.models.out</code> object returned by the <code>BIOMOD_Modeling</code> function</td>
</tr>
<tr>
<td><code>proj.name</code></td>
<td>A character corresponding to the name (ID) of the projection set (a new folder will be created within the simulation folder with this name)</td>
</tr>
</tbody>
</table>
new.env  A matrix, data.frame or SpatRaster object containing the new explanatory variables (in columns or layers, with names matching the variables names given to the BIOMOD_formatingData function to build bm.mod) that will be used to project the species distribution model(s).

Note that old format from raster are still supported such as RasterStack objects.

new.env.xy  (optional, default NULL)
If new.env is a matrix or a data.frame, a 2-columns matrix or data.frame containing the corresponding X and Y coordinates that will be used to project the species distribution model(s).

models.chosen  a vector containing model names to be kept, must be either all or a sub-selection of model names that can be obtained with the get_built_models function.

metric.binary  (optional, default NULL)
A vector containing evaluation metric names to be used to transform prediction values into binary values based on models evaluation scores obtained with the BIOMOD_Modeling function. Must be among all (same evaluation metrics than those of bm.mod) or ROC, TSS, KAPPA, ACCURACY, BIAS, POD, FAR, POFD, SR, CSI, ETS, HK, HSS, OR, ORSS

metric.filter  (optional, default NULL)
A vector containing evaluation metric names to be used to transform prediction values into filtered values based on models evaluation scores obtained with the BIOMOD_Modeling function. Must be among all (same evaluation metrics than those of bm.mod) or ROC, TSS, KAPPA, ACCURACY, BIAS, POD, FAR, POFD, SR, CSI, ETS, HK, HSS, OR, ORSS

compress  (optional, default TRUE)
A logical or character value defining whether and how objects should be compressed when saved on hard drive. Must be either TRUE, FALSE, xz or gzip (see Details)

build.clamping.mask  (optional, default TRUE)
A logical value defining whether a clamping mask should be built and saved on hard drive or not (see Details)

nb.cpu  (optional, default 1)
An integer value corresponding to the number of computing resources to be used to parallelize the single models computation

seed.val  (optional, default NULL)
An integer value corresponding to the new seed value to be set

...  (optional, see Details)

Details
If models.chosen = 'all', projections are done for all calibration and pseudo absences runs if applicable.
These projections may be used later by the BIOMOD_EnsembleForecasting function.
If `build.clamping.mask = TRUE`, a raster file will be saved within the projection folder. This mask values will correspond to the number of variables in each pixel that are out of their calibration/validation range, identifying locations where predictions are uncertain.

... can take the following values:

- `omit.na`: a logical value defining whether all not fully referenced environmental points will get NA as predictions or not
- `on_0_1000`: a logical value defining whether 0 - 1 probabilities are to be converted to 0 - 1000 scale to save memory on backup
- `do.stack`: a logical value defining whether all projections are to be saved as one `SpatRaster` object or several `SpatRaster` files (the default if projections are too heavy to be all loaded at once in memory)
- `keep.in.memory`: a logical value defining whether all projections are to be kept loaded at once in memory, or only links pointing to hard drive are to be returned
- `output.format`: a character value corresponding to the projections saving format on hard drive, must be either `.grd`, `.img`, `.tif` or `.RData` (the default if `new.env` is given as matrix or data.frame)

Value

A `BIOMOD.projection.out` object containing models projections, or links to saved outputs. Models projections are stored out of R (for memory storage reasons) in `proj.name` folder created in the current working directory:

1. the output is a data.frame if `new.env` is a matrix or a data.frame
2. it is a `SpatRaster` if `new.env` is a `SpatRaster` (or several `SpatRaster` objects, if `new.env` is too large)
3. raw projections, as well as binary and filtered projections (if asked), are saved in the `proj.name` folder

Author(s)

Wilfried Thuiller, Damien Georges

See Also

`BIOMOD_Modeling`, `BIOMOD_EnsembleModeling`, `BIOMOD_RangeSize`

Other Main functions: `BIOMOD_EnsembleForecasting()`, `BIOMOD_EnsembleModeling()`, `BIOMOD_FormatingData()`, `BIOMOD_LoadModels()`, `BIOMOD_Modeling()`, `BIOMOD_RangeSize()`

Examples

```r
library(terra)

# Load species occurrences (6 species available)
data(DataSpecies)
```
head(DataSpecies)

# Select the name of the studied species
myRespName <- 'GuloGulo'

# Get corresponding presence/absence data
myResp <- as.numeric(DataSpecies[, myRespName])

# Get corresponding XY coordinates
myRespXY <- DataSpecies[, c('X_WGS84', 'Y_WGS84')]

# Load environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
data(bioclim_current)
myExpl <- terra::rast(bioclim_current)

# ---------------------------------------------------------------#
file.out <- paste0(myRespName, '/', myRespName, '.AllModels.models.out')
if (file.exists(file.out)) {
  myBiomodModelOut <- get(load(file.out))
} else {
  # Format Data with true absences
  myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
                                          expl.var = myExpl,
                                          resp.xy = myRespXY,
                                          resp.name = myRespName)

  # Model single models
  myBiomodModelOut <- BIOMOD_Modeling(bm.format = myBiomodData,
                                       modeling.id = 'AllModels',
                                       models = c('RF', 'GLM'),
                                       CV.strategy = 'random',
                                       CV.nb.rep = 2,
                                       CV.perc = 0.8,
                                       OPT.strategy = 'bigboss',
                                       metric.eval = c('TSS', 'ROC'),
                                       var.import = 3,
                                       seed.val = 42)
}

# ---------------------------------------------------------------#
# Project single models
file.proj <- paste0(myRespName, '/proj_Current/', myRespName, '.Current.projection.out')
if (file.exists(file.proj)) {
  myBiomodProj <- get(load(file.proj))
} else {
  myBiomodProj <- BIOMOD_Projection(bm.mod = myBiomodModelOut,
                                     proj.name = 'Current',
                                     new.env = myExpl,
                                     models.chosen = 'all')
BIOMOD_RangeSize

**Description**

This function allows to calculate the absolute number of locations (pixels) lost, stable and gained, as well as the corresponding relative proportions, between two (or more) binary projections of (ensemble) species distribution models (which can represent new time scales or environmental scenarios for example).

**Usage**

```r
BIOMOD_RangeSize(proj.current, proj.future)
```

### S4 method for signature 'data.frame, data.frame'

```r
BIOMOD_RangeSize(proj.current, proj.future)
```

### S4 method for signature 'SpatRaster, SpatRaster'

```r
BIOMOD_RangeSize(proj.current, proj.future)
```

**Arguments**

- `proj.current`: a `data.frame`, `RasterLayer` or `SpatRaster` object containing the initial binary projection(s) of the (ensemble) species distribution model(s)
- `proj.future`: a `data.frame`, `RasterLayer` or `SpatRaster` object containing the final binary projection(s) of the (ensemble) species distribution model(s)

**Details**

Note that this function is only relevant to compare binary projections, made on the same area with the same resolution.

Comparison between `proj.current` and `proj.future` depends on the number of projection in both objects:

- **1 projection** (e.g. `data.frame with 1 column`, `SpatRaster with 1 layer`)
- **n projections** (e.g. `data.frame with n column`, `SpatRaster with n layer`)

- **1 projection** (e.g. `data.frame with 1 column`, `SpatRaster with 1 layer`)
- **n projections** (e.g. `data.frame with n column`, `SpatRaster with n layer`)

```r
myBiomodProj
plot(myBiomodProj)
```
Diff.By.Pixel object is obtained by applying the simple following formula:

\[ \text{proj.future} - 2 \times \text{proj.current} \]

Value

A list containing two objects:

- **Compt.By.Species** a data.frame containing the summary of range change for each comparison
  - Loss: number of pixels predicted to be lost
  - Stable0: number of pixels not currently occupied and not predicted to be
  - Stable1: number of pixels currently occupied and predicted to remain occupied
  - Gain: number of pixels predicted to be gained
  - PercLoss: percentage of pixels currently occupied and predicted to be lost \((\text{Loss} / (\text{Loss} + \text{Stable1}))\)
  - PercGain: percentage of pixels predicted to be gained compare to the number of pixels currently occupied \((\text{Gain} / (\text{Loss} + \text{Stable1}))\)
  - SpeciesRangeChange: percentage of pixels predicted to change (loss or gain) compare to the number of pixels currently occupied \((\text{PercGain} - \text{PercLoss})\)
  - CurrentRangeSize: number of pixels currently occupied
  - FutureRangeSize0Disp: number of pixels predicted to be occupied, assuming no migration
  - FutureRangeSize1Disp: number of pixels predicted to be occupied, assuming migration

- **Diff.By.Pixel** an object in the same form than the input data (\text{proj.current} and \text{proj.future}) and containing a value for each point/pixel of each comparison among:
  - -2: predicted to be lost
  - -1: predicted to remain occupied
  - 0: predicted to remain unoccupied
  - 1: predicted to be gained

Author(s)

Wilfried Thuiller, Damien Georges, Bruno Lafourcade

See Also

- \text{BIOMOD_Projection}, \text{BIOMOD_EnsembleForecasting}, \text{bm_PlotRangeSize}
- Other Main functions: \text{BIOMOD_EnsembleForecasting()}, \text{BIOMOD_EnsembleModeling()}, \text{BIOMOD_FormatingData()}, \text{BIOMOD_LoadModels()}, \text{BIOMOD_Modeling()}, \text{BIOMOD_Projection()}

Examples

```r
library(terra)

# Load species occurrences (6 species available)
data(DataSpecies)
```
head(DataSpecies)

# Select the name of the studied species
myRespName <- 'GuloGulo'

# Get corresponding presence/absence data
myResp <- as.numeric(DataSpecies[, myRespName])

# Get corresponding XY coordinates
myRespXY <- DataSpecies[, c('X_WGS84', 'Y_WGS84')]

# Load environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
data(bioclim_current)
myExpl <- terra::rast(bioclim_current)

# --------------------------------------------------------------- #
file.out <- paste0(myRespName, '/', myRespName, '.AllModels.models.out')
if (file.exists(file.out)) {
  myBiomodModelOut <- get(load(file.out))
} else {
  # Format Data with true absences
  myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
                                         expl.var = myExpl,
                                         resp.xy = myRespXY,
                                         resp.name = myRespName)

  # Model single models
  myBiomodModelOut <- BIOMOD_Modeling(bm.format = myBiomodData,
                                       modeling.id = 'AllModels',
                                       models = c('RF', 'GLM'),
                                       CV.strategy = 'random',
                                       CV.nb.rep = 2,
                                       CV.perc = 0.8,
                                       OPT.strategy = 'bigboss',
                                       metric.eval = c('TSS', 'ROC'),
                                       var.import = 3,
                                       seed.val = 42)
}

models.proj <- get_built_models(myBiomodModelOut, algo = "RF")
# Project single models
myBiomodProj <- BIOMOD_Projection(bm.mod = myBiomodModelOut,
                                   proj.name = 'CurrentRangeSize',
                                   new.env = myExpl,
                                   models.chosen = models.proj,
                                   metric.binary = 'all',
                                   build.clamping.mask = TRUE)

# --------------------------------------------------------------- #
# Load environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
data(bioclim_future)
myExplFuture <- terra::rast(bioclim_future)

# Project onto future conditions
myBiomodProjectionFuture <- BIOMOD_Projection(bm.mod = myBiomodModelOut,
                                             proj.name = 'FutureRangeSize',
                                             new.env = myExplFuture,
                                             models.chosen = models.proj,
                                             metric.binary = 'TSS')

# Load current and future binary projections
CurrentProj <- get_predictions(myBiomodProj,
                                metric.binary = "TSS",
                                model.as.col = TRUE)
FutureProj <- get_predictions(myBiomodProjectionFuture,
                                metric.binary = "TSS",
                                model.as.col = TRUE)

# Compute differences
myBiomodRangeSize <- BIOMOD_RangeSize(proj.current = CurrentProj, proj.future = FutureProj)
myBiomodRangeSize$Compt.By.Models
plot(myBiomodRangeSize$Diff.By.Pixel)

# Represent main results
bm_PlotRangeSize(bm.range = myBiomodRangeSize)

---

**bm_BinaryTransformation**

*Convert probability values into binary values using a predefined threshold*

**Description**

This internal biomod2 function allows to convert probability (not necessary between 0 and 1) values into binary presence-absence (0 or 1) values according to a predefined threshold (see Details).

**Usage**

bm_BinaryTransformation(data, threshold, do.filtering = FALSE)

## S4 method for signature 'data.frame'

bm_BinaryTransformation(data, threshold, do.filtering = FALSE)

## S4 method for signature 'matrix'

bm_BinaryTransformation(data, threshold, do.filtering = FALSE)
bm_BinaryTransformation

## S4 method for signature 'numeric'

```
bm_BinaryTransformation(data, threshold, do.filtering = FALSE)
```

## S4 method for signature 'SpatRaster'

```
bm_BinaryTransformation(data, threshold, do.filtering = FALSE)
```

### Arguments

- **data**: a vector, a matrix, data.frame, or a `SpatRaster` containing the data to be converted
- **threshold**: a numeric or a vector of numeric corresponding to the threshold used to convert the given data
- **do.filtering** *(optional, default FALSE)*: A logical value defining whether filtered data should be returned, or binary one (see Details)

### Details

If `data` is a vector, `threshold` should be a single numeric value.
If `data` is a matrix, data.frame or `SpatRaster`, `threshold` should be a vector containing as many values as the number of columns or layers contained in `data`. If only one numeric value is given, the same threshold will be applied to all columns or layers.

If `do.filtering = FALSE`, binary (0 or 1) values are returned.
If `do.filtering = TRUE`, values will be filtered according to `threshold`, meaning that:

- `data < threshold` will return 0
- `data >= threshold` will return the actual values of `data` (not transformed in 1)

### Value

An object of the same class than `data` and containing either binary (0 or 1) values, or filtered values.

### Author(s)

Wilfried Thuiller, Damien Georges

### See Also

`BIOMOD_Projection`, `BIOMOD_EnsembleForecasting`

## Examples

```r
# Generate a 0-1000 vector (normal distribution)
vec.d <- rnorm(100, 500, 100)

# From continuous to binary / filtered vector
vec.d_bin <- bm_BinaryTransformation(data = vec.d, threshold = 500)
vec.d_filt <- bm_BinaryTransformation(data = vec.d, threshold = 500, do.filtering = TRUE)
cbind(vec.d, vec.d_bin, vec.d_filt)
```

## Description

This internal `biomod2` function allows to build a cross-validation table according to 6 different methods: random, kfold, block, strat, env or user.defined (see Details).

## Usage

```r
bm_CrossValidation(
  bm.format,
  strategy = "random",
  nb.rep = 0,
  perc = 0.8,
  k = 0,
  balance = "presences",
  env.var = NULL,
  strat = "both",
  user.table = NULL,
  do.full.models = FALSE
)
```

```r
bm_CrossValidation_user.defined(bm.format, ...)
```

```r
# S4 method for signature 'BIOMOD.formated.data'
bm_CrossValidation_user.defined(bm.format, user.table)
```

```r
# S4 method for signature 'BIOMOD.formated.data.PA'
bm_CrossValidation_user.defined(bm.format, user.table)
```

```r
bm_CrossValidation_random(bm.format, ...)
```

```r
# S4 method for signature 'BIOMOD.formated.data'
bm_CrossValidation_random(bm.format, nb.rep, perc)
```
bm_CrossValidation

## S4 method for signature 'BIOMOD.formated.data.PA'

bm_CrossValidation_random(bm.format, nb.rep, perc)

bm_CrossValidation_kfold(bm.format, ...

## S4 method for signature 'BIOMOD.formated.data'

bm_CrossValidation_kfold(bm.format, nb.rep, k)

## S4 method for signature 'BIOMOD.formated.data.PA'

bm_CrossValidation_kfold(bm.format, nb.rep, k)

bm_CrossValidation_block(bm.format, ...

## S4 method for signature 'BIOMOD.formated.data'

bm_CrossValidation_block(bm.format)

## S4 method for signature 'BIOMOD.formated.data.PA'

bm_CrossValidation_block(bm.format)

bm_CrossValidation_strat(bm.format, ...

## S4 method for signature 'BIOMOD.formated.data'

bm_CrossValidation_strat(bm.format, balance, strat, k)

## S4 method for signature 'BIOMOD.formated.data.PA'

bm_CrossValidation_strat(bm.format, balance, strat, k)

bm_CrossValidation_env(bm.format, ...

## S4 method for signature 'BIOMOD.formated.data'

bm_CrossValidation_env(bm.format, balance, k, env.var)

## S4 method for signature 'BIOMOD.formated.data.PA'

bm_CrossValidation_env(bm.format, balance, k, env.var)

### Arguments

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bm.format</td>
<td>a <code>BIOMOD.formated.data</code> or <code>BIOMOD.formated.data.PA</code> object returned by the <code>BIOMOD_FormatingData</code> function</td>
</tr>
<tr>
<td>strategy</td>
<td>a character corresponding to the cross-validation selection strategy, must be among random, kfold, block, strat, env or user_defined</td>
</tr>
<tr>
<td>nb.rep</td>
<td>(optional, default 0) If strategy = 'random' or strategy = 'kfold', an integer corresponding to the number of sets (repetitions) of cross-validation points that will be drawn</td>
</tr>
<tr>
<td>perc</td>
<td>(optional, default 0) If strategy = 'random', a numeric between 0 and 1 defining the percentage of data that will be kept for calibration</td>
</tr>
</tbody>
</table>
k
(optional, default 0)
If strategy = 'kfold' or strategy = 'strat' or strategy = 'env', an integer corresponding to the number of partitions

balance
(optional, default 'presences')
If strategy = 'strat' or strategy = 'env', a character corresponding to how data will be balanced between partitions, must be either presences or absence

env.var
(optional)
If strategy = 'env', a character corresponding to the environmental variables used to build the partition. k partitions will be built for each environmental variables. By default the function uses all environmental variables available.

strat
(optional, default 'both')
If strategy = 'env', a character corresponding to how data will partitioned along gradient, must be among x, y, both

user.table
(optional, default NULL)
If strategy = 'user.defined', a matrix or data.frame defining for each repetition (in columns) which observation lines should be used for models calibration (TRUE) and validation (FALSE)

do.full.models
(optional, default TRUE)
A logical value defining whether models should be also calibrated and validated over the whole dataset (and pseudo-absence datasets) or not

Details

Several parameters are available within the function and some of them can be used with different cross-validation strategies:

<table>
<thead>
<tr>
<th>.......</th>
<th>random</th>
<th>kfold</th>
<th>block</th>
<th>strat</th>
<th>env</th>
</tr>
</thead>
<tbody>
<tr>
<td>nb.rep.</td>
<td>x......</td>
<td>x....</td>
<td>.....</td>
<td>.....</td>
<td>...</td>
</tr>
<tr>
<td>perc...</td>
<td>x......</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>...</td>
</tr>
<tr>
<td>k......</td>
<td>.....</td>
<td>x....</td>
<td>x....</td>
<td>x...</td>
<td>x..</td>
</tr>
<tr>
<td>balance</td>
<td>.....</td>
<td>.....</td>
<td>x....</td>
<td>x....</td>
<td>x..</td>
</tr>
<tr>
<td>strat..</td>
<td>.....</td>
<td>.....</td>
<td>x....</td>
<td>.....</td>
<td>...</td>
</tr>
</tbody>
</table>

Concerning column names of matrix output:
The number of columns depends on the strategy selected. The column names are given a posteriori of the selection, ranging from 1 to the number of columns. If do.full.models = TRUE, columns merging runs (and/or pseudo-absence datasets) are added at the end.

Concerning cross-validation strategies:
**random**  Most simple method to calibrate and validate a model is to split the original dataset in two datasets: one to calibrate the model and the other one to validate it. The splitting can be repeated `nb.rep` times.

**k-fold**  The k-fold method splits the original dataset in `k` datasets of equal sizes: each part is used successively as the validation dataset while the other `k-1` parts are used for the calibration, leading to `k` calibration/validation ensembles. This multiple splitting can be repeated `nb.rep` times.

**block**  It may be used to test for model overfitting and to assess transferability in geographic space. *Block* stratification was described in *Muscarella et al. 2014* (see References). Four bins of equal size are partitioned (bottom-left, bottom-right, top-left and top-right).

**stratified**  It may be used to test for model overfitting and to assess transferability in geographic space. *x* and *y* stratification was described in *Wenger and Olden 2012* (see References). *y* stratification uses `k` partitions along the *y*-gradient, *x* stratification does the same for the *x*-gradient. Both returns `2k` partitions: `k` partitions stratified along the *x*-gradient and `k` partitions stratified along the *y*-gradient.

**environmental**  It may be used to test for model overfitting and to assess transferability in environmental space. It returns `k` partitions for each variable given in `env.var`.

**user-defined**  Allow the user to give its own crossvalidation table. For a presence-absence dataset, column names must be formatted as: `_allData_RUNx` with `x` an integer. For a presence-only dataset for which several pseudo-absence dataset were generated, column names must be formatted as: `_PAx_RUNy` with `x` an integer and `PAx` an existing pseudo-absence dataset and `y` an integer

**Concerning balance parameter:**

If `balance = ‘presences’`, presences are divided (balanced) equally over the partitions (e.g. *Fig. 1b in Muscarella et al. 2014*). Absences or pseudo-absences will however be unbalanced over the partitions especially if the presences are clumped on an edge of the study area.

If `balance = ‘absences’`, absences (resp. pseudo-absences or background) are divided (balanced) as equally as possible between the partitions (geographical balanced bins given that absences are spread over the study area equally, approach similar to *Fig. 1 in Wenger et Olden 2012*). Presences will however be unbalanced over the partitions especially if the presences are clumped on an edge of the study area.

**Value**

A matrix or data.frame defining for each repetition (in columns) which observation lines should be used for models calibration (TRUE) and validation (FALSE).

**Author(s)**

Frank Breiner, Maya Gueguen
References


See Also

get.block, kfold, BIOMOD_FormatingData, BIOMOD_Modeling

Other Secondary functions: bm_BinaryTransformation(), bm_FindOptimStat(), bm_MakeFormula(), bm_ModelingOptions(), bm_PlotEvalBoxplot(), bm_PlotEvalMean(), bm_PlotRangeSize(), bm_PlotResponseCurves(), bm_PlotVarImpBoxplot(), bm_PseudoAbsences(), bm_RunModelsLoop(), bm_SRE(), bm_SampleBinaryVector(), bm_SampleFactorLevels(), bm_Tuning(), bm_VariablesImportance()

Examples

```r
library(terra)
# Load species occurrences (6 species available)
data(DataSpecies)
head(DataSpecies)

# Select the name of the studied species
myRespName <- 'GuloGulo'

# Get corresponding presence/absence data
myResp <- as.numeric(DataSpecies[, myRespName])

# Get corresponding XY coordinates
myRespXY <- DataSpecies[, c('X_WGS84', 'Y_WGS84')]

# Load environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
data(bioclim_current)
myExpl <- terra::rast(bioclim_current)

# Format Data with true absences
myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
                                      expl.var = myExpl,
                                      resp.xy = myRespXY,
                                      resp.name = myRespName)

# Create the different validation datasets
# random selection
```

bm_CrossValidation(bm.format = myBiomodData, 
strategy = "random", 
nb.rep = 3, 
k = 0.8)

# k-fold selection
bm_CrossValidation(bm.format = myBiomodData, 
strategy = "kfold", 
nb.rep = 2, 
k = 3)

# block selection
bm_CrossValidation(bm.format = myBiomodData, 
strategy = "block")

# stratified selection (geographic)
bm_CrossValidation(bm.format = myBiomodData, 
strategy = "strat", 
k = 2, 
balance = "presences", 
strat = "x")

# stratified selection (environmental)
bm_CrossValidation(bm.format = myBiomodData, 
strategy = "env", 
k = 2, 
balance = "presences")

head(cv.r)
apply(cv.r, 2, table)
head(cv.k)
apply(cv.k, 2, table)
head(cv.b)
apply(cv.b, 2, table)
head(cv.s)
apply(cv.s, 2, table)
head(cv.e)
apply(cv.e, 2, table)

bm_FindOptimStat  Calculate the best score according to a given evaluation method

Description

This internal biomod2 function allows the user to find the threshold to convert continuous values into binary ones leading to the best score for a given evaluation metric.
Usage

```r
bm_FindOptimStat(
  metric.eval = "TSS",
  obs,
  fit,
  nb.thresh = 100,
  threshold = NULL,
  boyce.bg.env = NULL,
  mpa.perc = 0.9
)
```

```r
get_optim_value(metric.eval)
```

```r
bm_CalculateStat(misc, metric.eval = "TSS")
```

Arguments

- `metric.eval` : a character corresponding to the evaluation metric to be used, must be either POD, FAR, POFD, SR, ACCURACY, BIAS, ROC, TSS, KAPPA, OR, ORSS, CSI, ETS, BOYCE, MPA
- `obs` : a vector of observed values (binary, 0 or 1)
- `fit` : a vector of fitted values (continuous)
- `nb.thresh` : an integer corresponding to the number of thresholds to be tested over the range of fitted values
- `threshold` : (optional, default NULL) A numeric corresponding to the threshold used to convert the given data
- `boyce.bg.env` : (optional, default NULL) A matrix, data.frame, SpatVector or SpatRaster object containing values of environmental variables (in columns or layers) extracted from the background (if presences are to be compared to background instead of absences or pseudo-absences selected for modeling) Note that old format from raster and sp are still supported such as RasterStack and SpatialPointsDataFrame objects.
- `mpa.perc` : a numeric between 0 and 1 corresponding to the percentage of correctly classified presences for Minimal Predicted Area (see ecospat.mpa() in ecospat)
- `misc` : a matrix corresponding to a contingency table

Details

- `simple`
  - POD : Probability of detection (hit rate)
  - FAR : False alarm ratio
  - POFD : Probability of false detection (fall-out)
  - SR : Success ratio
  - ACCURACY : Accuracy (fraction correct)
  - BIAS : Bias score (frequency bias)
complex

• ROC: Relative operating characteristic
• TSS: True skill statistic (Hanssen and Kuipers discriminant, Peirce’s skill score)
• KAPPA: Cohen’s Kappa (Heidke skill score)
• OR: Odds Ratio
• ORSS: Odds ratio skill score (Yule’s Q)
• CSI: Critical success index (threat score)
• ETS: Equitable threat score (Gilbert skill score)

presence-only

• BOYCE: Boyce index
• MPA: Minimal predicted area (cutoff optimising MPA to predict 90% of presences)

Optimal value of each method can be obtained with the get_optim_value function.

Please refer to the CAWRC website (section "Methods for dichotomous forecasts") to get detailed description of each metric.

Note that if a value is given to threshold, no optimisation will be done, and only the score for this threshold will be returned.

The Boyce index returns NA values for SRE models because it can not be calculated with binary predictions.
This is also the reason why some NA values might appear for GLM models if they do not converge.

Value

A 1 row x 5 columns data.frame containing:

• metric.eval: the chosen evaluation metric
• cutoff: the associated cut-off used to transform the continuous values into binary
• sensitivity: the sensibility obtained on fitted values with this threshold
• specificity: the specificity obtained on fitted values with this threshold
• best.stat: the best score obtained for the chosen evaluation metric

Note

In order to break dependency loop between packages biomod2 and ecospat, code of ecospat.boyce() and ecospat.mpa() in ecospat) functions have been copied within this file from version 3.2.2 (august 2022).

Author(s)

Damien Georges

References

See Also
ecospat.boyce() and ecospat.mpa() in ecospat, BIOMOD_Modeling, bm_RunModelsLoop, BIOMOD_EnsembleModeling
Other Secondary functions: bm_BinaryTransformation(), bm_CrossValidation(), bm_MakeFormula(), bm_ModelingOptions(), bm_PlotEvalBoxplot(), bm_PlotEvalMean(), bm_PlotRangeSize(), bm_PlotResponseCurves(), bm_PlotVarImpBoxplot(), bm_PseudoAbsences(), bm_RunModelsLoop(), bm_SRE(), bm_SampleBinaryVector(), bm_SampleFactorLevels(), bm_Tuning(), bm_VariablesImportance()

Examples

## Generate a binary vector
vec.a <- sample(c(0, 1), 100, replace = TRUE)

## Generate a 0-1000 vector (random drawing)
vec.b <- runif(100, min = 0, max = 1000)

## Generate a 0-1000 vector (biased drawing)
BiasedDrawing <- function(x, m1 = 300, sd1 = 200, m2 = 700, sd2 = 200) {
  return(ifelse(x < 0.5, rnorm(1, m1, sd1), rnorm(1, m2, sd2)))
}
vec.c <- sapply(vec.a, BiasedDrawing)
vec.c[which(vec.c < 0)] <- 0
vec.c[which(vec.c > 1000)] <- 1000

## Find optimal threshold for a specific evaluation metric
bm_FindOptimStat(metric.eval = 'TSS', fit = vec.b, obs = vec.a)
bm_FindOptimStat(metric.eval = 'TSS', fit = vec.c, obs = vec.a, nb.thresh = 100)
bm_FindOptimStat(metric.eval = 'TSS', fit = vec.c, obs = vec.a, threshold = 280)
Arguments

- **resp.name**: a character corresponding to the response variable name
- **expl.var**: a matrix or data.frame containing the explanatory variables that will be used at the modeling step
- **type**: a character corresponding to the wanted type of formula, must be simple, quadratic, polynomial or s_smoother
- **interaction.level**: an integer corresponding to the interaction level depth between explanatory variables
- **k** *(optional, default NULL)*: An integer corresponding to the smoothing parameter value of s or s arguments *(used only if type = 's_smoother')*

Details

It is advised to give only a subset of expl.var table to avoid useless memory consuming. If some explanatory variables are factorial, expl.var must be a data.frame whose corresponding columns are defined as factor.

Value

A `formula` class object that can be directly given to most of R statistical models.

Author(s)

Damien Georges

See Also

`formula`, `s`, `bm_ModelingOptions`, `bm_Tuning`, `bm_RunModelsLoop`

Other Secondary functions: `bm_BinaryTransformation()`, `bm_CrossValidation()`, `bm_FindOptimStat()`, `bm_ModelingOptions()`, `bm_PlotEvalBoxplot()`, `bm_PlotEvalMean()`, `bm_PlotRangeSize()`, `bm_PlotResponseCurves()`, `bm_PlotVarImpBoxplot()`, `bm_PseudoAbsences()`, `bm_RunModelsLoop()`, `bm_SRE()`, `bm_SampleBinaryVector()`, `bm_SampleFactorLevels()`, `bm_Tuning()`, `bm_VariablesImportance()`

Examples

```r
## Create simple simulated data
myResp.s <- sample(c(0, 1), 20, replace = TRUE)
myExpl.s <- data.frame(var1 = sample(c(0, 1), 100, replace = TRUE),
                        var2 = rnorm(100),
                        var3 = 1:100)

## Generate automatic formula
bm_MakeFormula(resp.name = 'myResp.s',
               expl.var = head(myExpl.s),
               type = 'quadratic',
               interaction.level = 0)
```
bm_ModelingOptions

Configure the modeling options for each selected model

Description

Parameterize and/or tune biomod2’s single models options.

Usage

bm_ModelingOptions(
  data.type,
  models = c("ANN", "CTA", "FDA", "GAM", "GBM", "GLM", "MARS", "MAXENT", "MAXNET", "RF", "SRE", "XGBOOST"),
  strategy,
  user.val = NULL,
  user.base = "bigboss",
  bm.format = NULL,
  calib.lines = NULL
)

Arguments

data.type       a character corresponding to the data type to be used, must be either binary, binary.PA, abundance, compositional
models          a vector containing model names to be computed, must be among ANN, CTA, FDA, GAM, GBM, GLM, MARS, MAXENT, MAXNET, RF, SRE, XGBOOST
strategy        a character corresponding to the method to select models’ parameters values, must be either default, bigboss, user.defined, tuned
user.val        (optional, default NULL)
                A list containing parameters values for some (all) models
user.base       (optional, default bigboss)
                A character, default or bigboss used when strategy = 'user.defined'. It sets the bases of parameters to be modified by user defined values.
bm.format       (optional, default NULL)
                A BIOMOD.formated.data or BIOMOD.formated.data.PA object returned by the BIOMOD_FormatingData function
calib.lines     (optional, default NULL)
                A data.frame object returned by get_calib_lines or bm_CrossValidation functions
Details

This function creates a `BIOMOD.models.options` object containing parameter values for each single model that can be run within `biomod2` through `BIOMOD_Modeling` function. 12 models are currently available, and are listed within the `ModelsTable` dataset.

Different strategies are available to set those parameters, through the `strategy` argument:

- **default** all parameters names and values are directly retrieve from functions to be called through `formalArgs` and `formals` functions respectively
- **bigboss** default parameter values are updated with values predefined by `biomod2` team
- **user.defined** default parameter values are updated with values provided by the user
- **tuned** default parameter values are updated by calling `bm_Tuning` function

Value

A `BIOMOD.models.options` of object that can be used to build species distribution model(s) with the `BIOMOD_Modeling` function.

Note

MAXENT being the only external model (not called through a R package), default parameters, and their values, are the following:

- `path_to_maxent.jar = getwd()` : a character corresponding to path to maxent.jar file
- `memory_allocated = 512` : an integer corresponding to the amount of memory (in Mo) reserved for java to run MAXENT, must be either 64, 128, 256, 512, 1024... or NULL to use default java memory limitation parameter
- `initial_heap_size = NULL` : a character corresponding to initial heap space (shared memory space) allocated to java (argument `-Xms` when calling java), must be either 1024K, 4096M, 10G ... or NULL to use default java parameter. Used in `BIOMOD_Projection` but not in `BIOMOD_Modeling`.
- `max_heap_size = NULL` : a character corresponding to maximum heap space (shared memory space) allocated to java (argument `-Xmx` when calling java), must be either 1024K, 4096M, 10G ... or NULL to use default java parameter, and must be larger than `initial_heap_size`. Used in `BIOMOD_Projection` but not in `BIOMOD_Modeling`.
- `background_data_dir = 'default'` : a character corresponding to path to folder where explanatory variables are stored as ASCII files (raster format). If specified, MAXENT will generate its own background data from rasters of explanatory variables ('default' value). Otherwise `biomod2` pseudo-absences will be used (see `BIOMOD_FormatingData`).
- `visible = FALSE` : a logical value defining whether MAXENT user interface is to be used or not
- `linear = TRUE` : a logical value defining whether linear features are to be used or not
- `quadratic = TRUE` : a logical value defining whether quadratic features are to be used or not
- `product = TRUE` : a logical value defining whether product features are to be used or not
- `threshold = TRUE` : a logical value defining whether threshold features are to be used or not
• hinge = TRUE: a logical value defining whether hinge features are to be used or not
• l2lqthreshold = 10: an integer corresponding to the number of samples at which quadratic features start being used
• lq2lqptthreshold = 80: an integer corresponding to the number of samples at which product and threshold features start being used
• hingethreshold = 15: an integer corresponding to the number of samples at which hinge features start being used
• beta_lqp = -1.0: a numeric corresponding to the regularization parameter to be applied to all linear, quadratic and product features (negative value enables automatic setting)
• beta_threshold = -1.0: a numeric corresponding to the regularization parameter to be applied to all threshold features (negative value enables automatic setting)
• beta_hinge = -1.0: a numeric corresponding to the regularization parameter to be applied to all hinge features (negative value enables automatic setting)
• beta_categorical = -1.0: a numeric corresponding to the regularization parameter to be applied to all categorical features (negative value enables automatic setting)
• betamultiplier = 1: a numeric corresponding to the number by which multiply all automatic regularization parameters (higher number gives a more spread-out distribution)
• defaultprevalence = 0.5: a numeric corresponding to the default prevalence of the modelled species (probability of presence at ordinary occurrence points)

Author(s)
Damien Georges, Wilfried Thuiller, Maya Gueguen

See Also
ModelsTable, BIOMOD.models.options, bm_Tuning, BIOMOD_Modeling

Other Secondary functions: bm_BinaryTransformation(), bm_CrossValidation(), bm_FindOptimStat(), bm_MakeFormula(), bm_PlotEvalBoxplot(), bm_PlotEvalMean(), bm_PlotRangeSize(), bm_PlotResponseCurves(), bm_PlotVarImpBoxplot(), bm_PseudoAbsences(), bm_RunModelsLoop(), bm_SRE(), bm_SampleBinaryVector(), bm_SampleFactorLevels(), bm_Tuning(), bm_VariablesImportance()

Examples
library(terra)

# Load species occurrences (6 species available)
data(DataSpecies)
head(DataSpecies)

# Select the name of the studied species
myRespName <- 'GuloGulo'

# Get corresponding presence/absence data
myResp <- as.numeric(DataSpecies[, myRespName])

# Get corresponding XY coordinates
myRespXY <- DataSpecies[, c("X_WGS84", "Y_WGS84")]

# Load environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
data(bioclim_current)
myExpl <- terra::rast(bioclim_current)

# Format Data with true absences
myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
expl.var = myExpl,
resp.xy = myRespXY,
resp.name = myRespName)

# k-fold selection
cv.k <- bm_CrossValidation(bm.format = myBiomodData,
strategy = "kfold",
 nb.rep = 2,
k = 3)

# default parameters
opt.d <- bm_ModelingOptions(data.type = "binary",
 models = allModels,
 strategy = "default")

# providing formated data
opt.df <- bm_ModelingOptions(data.type = "binary",
 models = allModels,
 strategy = "default",
 bm.format = myBiomodData,
calib.lines = cv.k)

opt.d
opt.d@models
opt.d@options$ANN.binary.nnet.nnet
names(opt.d@options$ANN.binary.nnet.nnet@args.values)

opt.df@options$ANN.binary.nnet.nnet
names(opt.df@options$ANN.binary.nnet.nnet@args.values)

# bigboss parameters
opt.b <- bm_ModelingOptions(data.type = "binary",
 models = allModels,
 strategy = "bigboss")
# user defined parameters
user.SRE <- list('_allData_allRun' = list(quant = 0.01))
user.XGBOOST <- list('_allData_allRun' = list(nrounds = 10))
user.val <- list(SRE.binary.biomod2_bm.SRE = user.SRE
                    , XGBOOST.binary.xgboost.xgboost = user.XGBOOST)

opt.u <- bm_ModelingOptions(data.type = 'binary',
                            models = c('SRE', 'XGBOOST'),
                            strategy = 'user.defined',
                            user.val = user.val)

opt.b
opt.u

## Not run:
# tuned parameters with formated data
opt.t <- bm_ModelingOptions(data.type = 'binary',
                            models = c('SRE', 'XGBOOST'),
                            strategy = 'tuned',
                            bm.format = myBiomodData)

opt.t

## End(Not run)

---

**bm_PlotEvalBoxplot**  
*Plot boxplot of evaluation scores*

**Description**

This function represents boxplot of evaluation scores of species distribution models, from `BIOMOD.models.out` or `BIOMOD.ensemble.models.out` objects that can be obtained from `BIOMOD_Modeling` or `BIOMOD_EnsembleModeling` functions. Scores are represented according to 2 grouping methods (see Details).

**Usage**

```r
bm_PlotEvalBoxplot(
  bm.out,
  dataset = "calibration",
  group.by = c("algo", "run"),
  do.plot = TRUE,
  ...
)
```

**Arguments**

- `bm.out` a `BIOMOD.models.out` or `BIOMOD.ensemble.models.out` object that can be obtained with the `BIOMOD_Modeling` or `BIOMOD_EnsembleModeling` functions.
dataset: a character corresponding to the dataset upon which evaluation metrics have been calculated and that is to be represented, must be among calibration, validation, evaluation

group.by: a 2-length vector containing the way kept models will be represented, must be among full.name, PA, run, algo (if bm.out is a BIOMOD.models.out object), or full.name, merged.by.PA, merged.by.run, merged.by.algo (if bm.out is a BIOMOD.ensemble.models.out object)

do.plot: (optional, default TRUE)
A logical value defining whether the plot is to be rendered or not

... some additional arguments (see Details)

Details
... can take the following values:

• main: a character corresponding to the graphic title

• scales: a character corresponding to the scales argument of the facet_wrap function, must be either fixed, free_x, free_y or free

Value
A list containing a data.frame with evaluation scores and the corresponding ggplot object representing them in boxplot.

Author(s)
Damien Georges, Maya Gueguen

See Also
BIOMOD.models.out, BIOMOD.ensemble.models.out, BIOMOD_Modeling, BIOMOD_EnsembleModeling, get_evaluations

Other Secondary functions: bm_BinaryTransformation(), bm_CrossValidation(), bm_FindOptimStat(), bm_MakeFormula(), bm_ModelingOptions(), bm_PlotEvalMean(), bm_PlotRangeSize(), bm_PlotResponseCurves(), bm_PlotVarImpBoxplot(), bm_PseudoAbsences(), bm_RunModelsLoop(), bm_SRE(), bm_SampleBinaryVector(), bm_SampleFactorLevels(), bm_Tuning(), bm_VariablesImportance()

Other Plot functions: bm_PlotEvalMean(), bm_PlotRangeSize(), bm_PlotResponseCurves(), bm_PlotVarImpBoxplot()

Examples

library(terra)

# Load species occurrences (6 species available)
data(DataSpecies)
head(DataSpecies)

# Select the name of the studied species
myRespName <- 'GuloGulo'
# Get corresponding presence/absence data
myResp <- as.numeric(DataSpecies[, myRespName])

# Get corresponding XY coordinates
myRespXY <- DataSpecies[, c('X_WGS84', 'Y_WGS84')]

# Load environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
data(bioclim_current)
myExpl <- terra::rast(bioclim_current)

# ---------------------------------------------------------------
file.out <- paste0(myRespName, '/', myRespName, '.AllModels.models.out')
if (file.exists(file.out)) {
  myBiomodModelOut <- get(load(file.out))
} else {

  # Format Data with true absences
  myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
                                        expl.var = myExpl,
                                        resp.xy = myRespXY,
                                        resp.name = myRespName)

  # Model single models
  myBiomodModelOut <- BIOMOD_Modeling(bm.format = myBiomodData,
                                       modeling.id = 'AllModels',
                                       models = c('RF', 'GLM'),
                                       CV.strategy = 'random',
                                       CV.nb.rep = 2,
                                       CV.perc = 0.8,
                                       OPT.strategy = 'bigboss',
                                       metric.eval = c('TSS', 'ROC'),
                                       var.import = 3,
                                       seed.val = 42)
}

# ---------------------------------------------------------------
# Get evaluation scores
get_evaluations(myBiomodModelOut)

# Represent evaluation scores
bm_PlotEvalBoxplot(bm.out = myBiomodModelOut, group.by = c('algo', 'run'))
bm_PlotEvalMean

Description

This function represents mean evaluation scores (and their standard deviation) of species distribution models, from `BIOMOD.models.out` or `BIOMOD.ensemble.models.out` objects that can be obtained from `BIOMOD_Modeling` or `BIOMOD_EnsembleModeling` functions. Scores are represented according to 2 different evaluation methods, and models can be grouped (see Details).

Usage

```r
bm_PlotEvalMean(
  bm.out,
  metric.eval = NULL,
  dataset = "calibration",
  group.by = "algo",
  do.plot = TRUE,
  ...
)
```

Arguments

- `bm.out` a `BIOMOD.models.out` or `BIOMOD.ensemble.models.out` object that can be obtained with the `BIOMOD_Modeling` or `BIOMOD_EnsembleModeling` functions
- `metric.eval` a vector containing evaluation metric names to be used, must be among ROC, TSS, KAPPA, ACCURACY, BIAS, POD, FAR, POFD, SR, CSI, ETS, HK, HSS, OR, ORSS
- `dataset` a character corresponding to the dataset upon which evaluation metrics have been calculated and that is to be represented, must be among calibration, validation, evaluation
- `group.by` a character corresponding to the way kept models will be combined to compute mean and sd evaluation scores, must be among full.name, PA, run, algo (if `bm.out` is a `BIOMOD.models.out` object), or full.name, merged.by.PA, merged.by.run, merged.by.algo (if `bm.out` is a `BIOMOD.ensemble.models.out` object)
- `do.plot` (optional, default TRUE)
  A logical value defining whether the plot is to be rendered or not
  ...
  some additional arguments (see Details)

Details

... can take the following values:

- `xlim`: an integer corresponding to the x maximum limit to represent
- `ylim`: an integer corresponding to the y maximum limit to represent
- `main`: a character corresponding to the graphic title
- `col`: a vector containing new color values

Value

A list containing a `data.frame` with mean and standard deviation of evaluation scores and the corresponding `ggplot` object representing them according to 2 different evaluation methods.
Author(s)
Damien Georges, Maya Gueguen

See Also
BIOMOD.models.out, BIOMOD.ensemble.models.out, BIOMOD_Modeling, BIOMOD_EnsembleModeling, get_evaluations

Other Secondary functions: bm_BinaryTransformation(), bm_CrossValidation(), bm_FindOptimStat(), bm_MakeFormula(), bm_ModelingOptions(), bm_PlotEvalBoxplot(), bm_PlotRangeSize(), bm_PlotResponseCurves(), bm_PlotVarImpBoxplot(), bm_PseudoAbsences(), bm_RunModelsLoop(), bm_SRE(), bm_SampleBinaryVector(), bm_SampleFactorLevels(), bm_Tuning(), bm_VariablesImportance()

Other Plot functions: bm_PlotEvalBoxplot(), bm_PlotRangeSize(), bm_PlotResponseCurves(), bm_PlotVarImpBoxplot()

Examples

library(terra)

# Load species occurrences (6 species available)
data(DataSpecies)
head(DataSpecies)

# Select the name of the studied species
myRespName <- 'GuloGulo'

# Get corresponding presence/absence data
myResp <- as.numeric(DataSpecies[, myRespName])

# Get corresponding XY coordinates
myRespXY <- DataSpecies[, c('X_WGS84', 'Y_WGS84')]

# Load environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
data(bioclim_current)
myExpl <- terra::rast(bioclim_current)

# ---------------------------------------------------------------
file.out <- paste0(myRespName, '/', myRespName, '.AllModels.models.out')
if (file.exists(file.out)) {
  myBiomodModelOut <- get(load(file.out))
} else {

  # Format Data with true absences
  myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
                                        expl.var = myExpl,
                                        resp.xy = myRespXY,
                                        resp.name = myRespName)

  # Model single models
  myBiomodModelOut <- BIOMOD_Modeling(bm.format = myBiomodData,
modeling_id = 'AllModels',
models = c('RF', 'GLM'),
CV.strategy = 'random',
CV.nb.rep = 2,
CV.perc = 0.8,
OPT.strategy = 'bigboss',
metric.eval = c('TSS', 'ROC'),
var.import = 3,
seed.val = 42)
}

# ---------------------------------------------------------------
# Get evaluation scores
get_evaluations(myBiomodModelOut)

# Represent mean evaluation scores
bm_PlotEvalMean(bm.out = myBiomodModelOut)

bm_PlotRangeSize

Plot species range change

Description
This function represents species range change from object that can be obtained from \texttt{BIOMOD\_RangeSize} function. Several graphics can be obtained, representing global counts or proportions of gains / losses, as well as spatial representations (see Details).

Usage

\begin{verbatim}
bm_PlotRangeSize(
  bm.range,
  do.count = TRUE,
  do.perc = TRUE,
  do.maps = TRUE,
  do.mean = TRUE,
  do.plot = TRUE,
  row.names = c("Species", "Dataset", "Run", "Algo")
)
\end{verbatim}

Arguments

- \texttt{bm.range} \hspace{1cm} an object returned by the \texttt{BIOMOD\_RangeSize} function
- \texttt{do.count} \hspace{1cm} \texttt{(optional, default TRUE)}
  A logical value defining whether the count plot is to be computed or not
- \texttt{do.perc} \hspace{1cm} \texttt{(optional, default TRUE)}
  A logical value defining whether the percentage plot is to be computed or not
do.maps  
(optional, default TRUE)  
A logical value defining whether the maps plot is to be computed or not

do.mean  
(optional, default TRUE)  
A logical value defining whether the mean maps plot is to be computed or not

do.plot  
(optional, default TRUE)  
A logical value defining whether the plots are to be rendered or not

row.names  
(optional, default c('Species', 'Dataset', 'Run', 'Algo'))  
A vector containing tags matching bm.range$Compt.By.Models rownames splitted by '_' character

Details

4 plots can be obtained with this function:

**Count barplot** representing absolute number of locations (pixels) lost, stable and gained

**Percentage barplot** representing percentage of locations (pixels) lost, stable, and the corresponding Species Range Change (PercGain - PercLoss)

**SRC models maps** representing spatially locations (pixels) lost, stable and gained for each single distribution model

**SRC community averaging maps** representing spatially locations (pixels) lost, stable and gained, taking the majority value across single distribution models (and representing the percentage of models' agreement)

*Please see BIOMOD_RangeSize function for more details about the values.*

Value

A list containing one or several data.frame and the corresponding ggplot object representing species range change.

Author(s)

Maya Gueguen

See Also

BIOMOD_RangeSize

Other Secondary functions: bm_BinaryTransformation(), bm_CrossValidation(), bm_FindOptimStat(), bm_MakeFormula(), bm_ModelingOptions(), bm_PlotEvalBoxplot(), bm_PlotEvalMean(), bm_PlotResponseCurves(), bm_PlotVarImpBoxplot(), bm_PseudoAbsences(), bm_RunModelsLoop(), bm_SRE(), bm_SampleBinaryVector(), bm_SampleFactorLevels(), bm_Tuning(), bm_VariablesImportance()

Other Plot functions: bm_PlotEvalBoxplot(), bm_PlotEvalMean(), bm_PlotResponseCurves(), bm_PlotVarImpBoxplot()
Examples

library(terra)

# Load species occurrences (6 species available)
data(DataSpecies)
head(DataSpecies)

# Select the name of the studied species
myRespName <- 'GuloGulo'

# Get corresponding presence/absence data
myResp <- as.numeric(DataSpecies[, myRespName])

# Get corresponding XY coordinates
myRespXY <- DataSpecies[, c('X_WGS84', 'Y_WGS84')]

# Load environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
data(bioclim_current)
myExpl <- terra::rast(bioclim_current)

# ---------------------------------------------------------------#
file.out <- paste0(myRespName, '/', myRespName, '.AllModels.models.out')
if (file.exists(file.out)) {
  myBiomodModelOut <- get(load(file.out))
} else {
  # Format Data with true absences
  myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
    expl.var = myExpl,
    resp.xy = myRespXY,
    resp.name = myRespName)

  # Model single models
  myBiomodModelOut <- BIOMOD_Modeling(bm.format = myBiomodData,
    modeling.id = 'AllModels',
    models = c('RF', 'GLM'),
    CV.strategy = 'random',
    CV.nb.rep = 2,
    CV.perc = 0.8,
    OPT.strategy = 'bigboss',
    metric.eval = c('TSS', 'ROC'),
    var.import = 3,
    seed.val = 42)
}

models.proj <- get_built_models(myBiomodModelOut, algo = "RF")
# Project single models
myBiomodProj <- BIOMOD_Projection(bm.mod = myBiomodModelOut,
  proj.name = 'CurrentRangeSize',
  new.env = myExpl,
models.chosen = models.proj,
metric.binary = 'all')

# Load environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
data(bioclim_future)
myExplFuture <- terra::rast(bioclim_future)

# Project onto future conditions
myBiomodProjectionFuture <- BIOMOD_Projection(bm.mod = myBiomodModelOut,
  proj.name = 'FutureRangeSize',
  new.env = myExplFuture,
  models.chosen = models.proj,
  metric.binary = 'TSS')

# Load current and future binary projections
CurrentProj <- get_predictions(myBiomodProj,
  metric.binary = "TSS",
  model.as.col = TRUE)
FutureProj <- get_predictions(myBiomodProjectionFuture,
  metric.binary = "TSS",
  model.as.col = TRUE)

# Compute differences
myBiomodRangeSize <- BIOMOD_RangeSize(proj.current = CurrentProj, proj.future = FutureProj)

# Represent main results
bm_PlotRangeSize(bm.range = myBiomodRangeSize)

bm_PlotResponseCurves  Plot response curves

Description

This function represents response curves of species distribution models, from BIOMOD.models.out or BIOMOD.ensemble.models.out objects that can be obtained from BIOMOD_Modeling or BIOMOD_EnsembleModeling functions. Response curves can be represented in either 2 or 3 dimensions (meaning 1 or 2 explanatory variables at a time, see Details).
Usage

\begin{verbatim}
bm_PlotResponseCurves(
bm.out,
models.chosen = "all",
new.env = get_formal_data(bm.out, "expl.var"),
show.variables = get_formal_data(bm.out, "expl.var.names"),
fixed.var = "mean",
do.bivariate = FALSE,
do.plot = TRUE,
do.progress = TRUE,
...)
\end{verbatim}

Arguments

- **bm.out**: a BIOMOD.models.out or BIOMOD.ensemble.models.out object that can be obtained with the BIOMOD_Modeling or BIOMOD_EnsembleModeling functions
- **models.chosen**: a vector containing model names to be kept, must be either all or a sub-selection of model names that can be obtained with the get_built_models function
- **new.env**: a matrix, data.frame or SpatRaster object containing the new explanatory variables (in columns or layers, with names matching the variables names given to the BIOMOD_FormatingData function to build bm.out) that will be used to project the species distribution model(s)
  
  Note that old format from raster are still supported such as RasterStack objects.
- **show.variables**: a vector containing the names of the explanatory variables present into new.env parameter and to be plotted
- **fixed.var**: a character corresponding to the statistic to be used to fix as constant the remaining variables other than the one used to predict response, must be either mean, median, min, max
- **do.bivariate**: (optional, default FALSE)
  
  A logical value defining whether the response curves are to be represented in 3 dimensions (meaning 2 explanatory variables at a time) or not (meaning only 1)
- **do.plot**: (optional, default TRUE)
  
  A logical value defining whether the plot is to be rendered or not
- **do.progress**: (optional, default TRUE)
  
  A logical value defining whether the progress bar is to be rendered or not
- ... some additional arguments (see Details)

Details

This function is an adaptation of the Evaluation Strip method proposed by Elith et al. (2005). To build the predicted response curves:
• n-1 variables are set constant to a fixed value determined by the fixed.var parameter (in the case of categorical variable, the most represented class is taken)
• the remaining variable is made to vary throughout its range given by the new.env parameter
• predicted values are computed with these n-1 fixed variables, and this studied variable varying

If do.bivariate = TRUE, 2 variables are varying at the same time.

The response curves obtained show the sensibility of the model to the studied variable. Note that this method does not account for interactions between variables.

... can take the following values:
• main: a character corresponding to the graphic title

Value
A list containing a data.frame with variables and predicted values and the corresponding ggplot object representing response curves.

Author(s)
Damien Georges, Maya Gueguen

References

See Also
BIOMOD.models.out, BIOMOD.ensemble.models.out, BIOMOD_Modeling, BIOMOD_EnsembleModeling
Other Secondary functions: bm_BinaryTransformation(), bm_CrossValidation(), bm_FindOptimStat(), bm_MakeFormula(), bm_ModelingOptions(), bm_PlotEvalBoxplot(), bm_PlotEvalMean(), bm_PlotRangeSize(), bm_PlotVarImpBoxplot(), bm_PseudoAbsences(), bm_RunModelsLoop(), bm_SRE(), bm_SampleBinaryVector(), bm_SampleFactorLevels(), bm_Tuning(), bm_VariablesImportance()

Other Plot functions: bm_PlotEvalBoxplot(), bm_PlotEvalMean(), bm_PlotRangeSize(), bm_PlotVarImpBoxplot()

Examples

library(terra)

# Load species occurrences (6 species available)
data(DataSpecies)
head(DataSpecies)

# Select the name of the studied species
myRespName <- 'GuloGulo'

# Get corresponding presence/absence data
myResp <- as.numeric(DataSpecies[, myRespName])

# Get corresponding XY coordinates
myRespXY <- DataSpecies[, c('X_WGS84', 'Y_WGS84')]

# Load environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
data(bioclim_current)
myExpl <- terra::rast(bioclim_current)

# ---------------------------------------------------------------#
file.out <- paste0(myRespName, '/', myRespName, '.AllModels.models.out')
if (file.exists(file.out)) {
  myBiomodModelOut <- get(load(file.out))
} else {

  # Format Data with true absences
  myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
                                        expl.var = myExpl,
                                        resp.xy = myRespXY,
                                        resp.name = myRespName)

  # Model single models
  myBiomodModelOut <- BIOMOD_Modeling(bm.format = myBiomodData,
                                       modeling.id = 'AllModels',
                                       models = c('RF', 'GLM'),
                                       CV.strategy = 'random',
                                       CV.nb.rep = 2,
                                       CV.perc = 0.8,
                                       OPT.strategy = 'bigboss',
                                       metric.eval = c('TSS', 'ROC'),
                                       var.import = 3,
                                       seed.val = 42)
}

# ---------------------------------------------------------------#
# Represent response curves
mods <- get_built_models(myBiomodModelOut, run = 'RUN1')
bm_PlotResponseCurves(bm.out = myBiomodModelOut,
                       models.chosen = mods,
                       fixed.var = 'median')
## fixed.var can also be set to 'min', 'max' or 'mean'
# bm_PlotResponseCurves(bm.out = myBiomodModelOut,
#                       models.chosen = mods,
#                       fixed.var = 'min')

# Bivariate case (one model)
# variables can be selected with argument 'show.variables'
# models can be selected with argument 'models.chosen'
mods <- get_built_models(myBiomodModelOut, full.name = 'GuloGulo_allData_RUN2_RF')
bm_PlotResponseCurves(bm.out = myBiomodModelOut,
    show.variables = c("bio4","bio12","bio11"),
    models.chosen = mods,
    fixed.var = 'median',
    do.bivariate = TRUE)

--

bm_PlotVarImpBoxplot  Plot boxplot of variables importance

Description

This function represents boxplot of variables importance of species distribution models, from BIOMOD.models.out or BIOMOD.ensemble.models.out objects that can be obtained from BIOMOD_Modeling or BIOMOD_EnsembleModeling functions. Scores are represented according to 3 grouping methods (see Details).

Usage

bm_PlotVarImpBoxplot(
    bm.out,
    group.by = c("run", "expl.var", "algo"),
    do.plot = TRUE,
    ...
)

Arguments

bm.out  a BIOMOD.models.out or BIOMOD.ensemble.models.out object that can be obtained with the BIOMOD_Modeling or BIOMOD_EnsembleModeling functions

group.by  a 3-length vector containing the way kept models will be represented, must be among full.name, PA, run, algo, expl.var (if bm.out is a BIOMOD.models.out object), or full.name, merged.by.PA, merged.by.run, merged.by.algo, expl.var (if bm.out is a BIOMOD.ensemble.models.out object)

do.plot  (optional, default TRUE)

A logical value defining whether the plot is to be rendered or not

...  some additional arguments (see Details)

Details

... can take the following values:

- main: a character corresponding to the graphic title
Value

A list containing a data.frame with variables importance and the corresponding ggplot object representing them in boxplot.

Author(s)

Damien Georges, Maya Gueguen

See Also

BIOMOD.models.out, BIOMOD.ensemble.models.out, BIOMOD_Modeling, BIOMOD_EnsembleModeling, get_variables_importance

Other Secondary functions: bm_BinaryTransformation(), bm_CrossValidation(), bm_FindOptimStat(), bm_MakeFormula(), bm_ModelingOptions(), bm_PlotEvalBoxplot(), bm_PlotEvalMean(), bm_PlotRangeSize(), bm_PlotResponseCurves(), bm_PseudoAbsences(), bm_RunModelsLoop(), bm_SRE(), bm_SampleBinaryVector(), bm_SampleFactorLevels(), bm_Tuning(), bm_VariablesImportance()

Other Plot functions: bm_PlotEvalBoxplot(), bm_PlotEvalMean(), bm_PlotRangeSize(), bm_PlotResponseCurves()

Examples

library(terra)

# Load species occurrences (6 species available)
data(DataSpecies)
head(DataSpecies)

# Select the name of the studied species
myRespName <- "GuloGulo"

# Get corresponding presence/absence data
myResp <- as.numeric(DataSpecies[, myRespName])

# Get corresponding XY coordinates
myRespXY <- DataSpecies[, c("X_WGS84", "Y_WGS84")]

# Load environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
data(bioclim_current)
myExpl <- terra::rast(bioclim_current)

# ---------------------------------------------------------------
file.out <- paste0(myRespName, "/", myRespName, ".AllModels.models.out")
if (file.exists(file.out)) {
  myBiomodModelOut <- get(load(file.out))
} else {

  # Format Data with true absences
  myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
                                        expl.var = myExpl,
                                        myBiomodModelOut <- get(load(file.out))
  )
}

# Format Data with true absences
myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
                                        expl.var = myExpl,
bm_PseudoAbsences

This internal **biomod2** function allows to select pseudo-absences according to 4 different methods: random, sre, disk or user.defined (see Details).

**Usage**

```r
bm_PseudoAbsences(
  resp.var,  
  expl.var,  
  nb.rep = 1,  
  strategy = "random",  
  nb.absences = NULL,  
  sre.quant = 0,  
  dist.min = 0,  
  dist.max = NULL,  
  user.table = NULL
)
```
bm_PseudoAbsences_user.defined(resp.var, expl.var, ...)  

## S4 method for signature 'ANY,SpatVector'
bm_PseudoAbsences_user.defined(resp.var, expl.var, user.table)

## S4 method for signature 'ANY,SpatRaster'
bm_PseudoAbsences_user.defined(resp.var, expl.var, user.table)

bm_PseudoAbsences_random(resp.var, expl.var, ...)

## S4 method for signature 'ANY,SpatVector'
bm_PseudoAbsences_random(resp.var, expl.var, nb.absences, nb.rep)

## S4 method for signature 'ANY,SpatRaster'
bm_PseudoAbsences_random(resp.var, expl.var, nb.absences, nb.rep)

bm_PseudoAbsences_sre(resp.var, expl.var, ...)

## S4 method for signature 'ANY,SpatVector'
bm_PseudoAbsences_sre(resp.var, expl.var, sre.quant, nb.absences, nb.rep)

## S4 method for signature 'ANY,SpatRaster'
bm_PseudoAbsences_sre(resp.var, expl.var, sre.quant, nb.absences, nb.rep)

bm_PseudoAbsences_disk(resp.var, expl.var, ...)

## S4 method for signature 'ANY,SpatVector'
bm_PseudoAbsences_disk(
    resp.var,
    expl.var,
    dist.min,
    dist.max,
    nb.absences,
    nb.rep
  )

## S4 method for signature 'ANY,SpatRaster'
bm_PseudoAbsences_disk(
    resp.var,
    expl.var,
    dist.min,
    dist.max,
    nb.absences,
    nb.rep
  )
Arguments

resp.var  a vector, SpatialPoints or SpatialPointsDataFrame object containing binary data (0: absence, 1: presence, NA: indeterminate) for a single species that will be used to find the pseudo-absences

expl.var  a matrix, data.frame, SpatialPointsDataFrame or SpatRaster object containing the explanatory variables (in columns or layers) that will be used to find the pseudo-absences

nb.rep   an integer corresponding to the number of sets (repetitions) of pseudo-absence points that will be drawn

strategy  a character corresponding to the pseudo-absence selection strategy, must be among random, sre, disk or user.defined

nb.absences  (optional, default NULL)
   If strategy = 'random' or strategy = 'sre' or strategy = 'disk', an integer corresponding to the number of pseudo-absence points that will be selected for each pseudo-absence repetition (true absences included)

sre.quant  (optional, default 0)
   If strategy = 'sre', a numeric between 0 and 0.5 defining the half-quantile used to make the sre pseudo-absence selection (see bm_SRE)

dist.min  (optional, default 0)
   If strategy = 'disk', a numeric defining the minimal distance to presence points used to make the disk pseudo-absence selection (in meters)

dist.max  (optional, default NULL)
   If strategy = 'disk', a numeric defining the maximal distance to presence points used to make the disk pseudo-absence selection (in meters)

user.table  (optional, default NULL)
   If strategy = 'user.defined', a matrix or data.frame with as many rows as resp.var values, as many columns as nb.rep, and containing TRUE or FALSE values defining which points will be used to build the species distribution model(s) for each repetition

...  (optional, one or several of the above arguments depending on the selected method)

Details

Concerning random selection:

The idea is to select pseudo-absences randomly in spatial locations where the species has not been sampled. This method is the simplest one and the most appropriate if lacking information about the presence sampling (non-exhaustive, biased sampling, etc).

Concerning SRE selection (see bm_SRE):

The idea is to select pseudo-absences in spatial locations whose environmental conditions are different from those of the presence points. This method is appropriate when most of the environmental space of the species has been sampled.
Concerning disk selection:
The idea is to select pseudo-absences, not too close from presence points, but not too far away either. This method is appropriate when most of the spatial range of the species has been sampled.

Concerning user defined selection:
The user can provide pseudo-absences locations through a table containing spatial locations in rows, pseudo-absences repetitions in columns, and TRUE/FALSE values indicating whether each point is to be considered as pseudo-absence or not for each dataset.

Value
A list containing the following elements:
• xy: the coordinates of the species observations
• sp: the values of the species observations (0, 1 or NA)
• env: the explanatory variables
• pa.tab: the corresponding table of selected pseudo-absences (indicated by TRUE or FALSE)

Author(s)
Wilfried Thuiller, Damien Georges

See Also
bm_SRE, BIOMOD.formated.data.PA, BIOMOD_FormatingData

Other Secundary functions: bm_BinaryTransformation(), bm_CrossValidation(), bm_FindOptimStat(),
bm_MakeFormula(), bm_ModelingOptions(), bm_PlotEvalBoxplot(), bm_PlotEvalMean(), bm_PlotRangeSize(),
bm_PlotResponseCurves(), bm_PlotVarImpBoxplot(), bm_RunModelsLoop(), bm_SRE(), bm_SampleBinaryVector(),
bm_SampleFactorLevels(), bm_Tuning(), bm_VariablesImportance()

Examples
library(terra)

# Load species occurrences (6 species available)
data(DataSpecies)
head(DataSpecies)

# Select the name of the studied species
myRespName <- 'GuloGulo'

# Get corresponding presence/absence data
myResp <- as.numeric(DataSpecies[, myRespName])

# Get corresponding XY coordinates
myRespXY <- DataSpecies[, c('X_WGS84', 'Y_WGS84')]

# Load environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
data(bioclim_current)
myExpl <- terra::rast(bioclim_current)

# Create the different pseudo-absence datasets

# Transform true absences into potential pseudo-absences
myResp.PA <- ifelse(myResp == 1, 1, NA)
myResp.PA.vect <- vect(cbind(myRespXY, myResp.PA), geom = c("X_WGS84","Y_WGS84"))

# random method
PA.r <- bm_PseudoAbsences(resp.var = myResp.PA.vect,
                           expl.var = myExpl,
                           nb.rep = 4,
                           nb.absences = 1000,
                           strategy = 'random')

# disk method
PA.d <- bm_PseudoAbsences(resp.var = myResp.PA.vect,
                           expl.var = myExpl,
                           nb.rep = 4,
                           nb.absences = 500,
                           strategy = 'disk',
                           dist.min = 5,
                           dist.max = 35)

# SRE method
PA.s <- bm_PseudoAbsences(resp.var = myResp.PA.vect,
                           expl.var = myExpl,
                           nb.rep = 4,
                           nb.absences = 1000,
                           strategy = 'sre',
                           sre.quant = 0.025)

# user.defined method
myPAtable <- data.frame(PA1 = ifelse(myResp == 1, TRUE, FALSE),
                        PA2 = ifelse(myResp == 1, TRUE, FALSE))
for (i in 1:ncol(myPAtable)) myPAtable[sample(which(myPAtable[, i] == FALSE), 500), i] = TRUE
PA.u <- bm_PseudoAbsences(resp.var = myResp.PA.vect,
                           expl.var = myExpl,
                           strategy = 'user.defined',
                           user.table = myPAtable)

str(PA.r)
head(PA.r$pa.tab)
apply(PA.r$pa.tab, 2, table)
bm_RunModelsLoop

Loop to compute all single species distribution models

Description

This internal biomod2 function allows the user to compute all single species distribution models (asked by the BIOMOD_Modeling function).

Usage

bm_RunModelsLoop(  bm.format,  weights,  calib.lines,  modeling.id,  models,  models.pa,  bm.options,  metric.eval,  var.import,  scale.models = TRUE,  nb.cpu = 1,  seed.val = NULL,  do.progress = TRUE)
bm_RunModelsLoop

bm_RunModel(
  model,
  run.name,
  dir.name = ".",
  modeling.id = ",",
  bm.options,
  Data,
  weights.vec,
  calib.lines.vec,
  eval.data = NULL,
  metric.eval = c("ROC", "TSS", "KAPPA"),
  var.import = 0,
  scale.models = TRUE,
  nb.cpu = 1,
  seed.val = NULL,
  do.progress = TRUE
)

Arguments

bm.format a BIOMOD.formated.data or BIOMOD.formated.data.PA object returned by the BIOMOD_FormatingData function
weights a matrix containing observation weights for each pseudo-absence (or allData) dataset
calib.lines a matrix containing calibration / validation lines for each pseudo-absence (or allData) x repetition (or allRun) combination that can be obtained with the bm_CrossValidation function
modeling.id a character corresponding to the name (ID) of the simulation set (a random number by default)
models a vector containing model names to be computed, must be among ANN, CTA, FDA, GAM, GBM, GLM, MARS, MAXENT, MAXNET, RF, SRE, XGBOOST
models.pa (optional, default NULL) A list containing for each model a vector defining which pseudo-absence datasets are to be used, must be among colnames(bm.format@PA.table)
bm.options a BIOMOD.models.options object returned by the bm_ModelingOptions function
metric.eval a vector containing evaluation metric names to be used, must be among ROC, TSS, KAPPA, ACCURACY, BIAS, POD, FAR, POFD, SR, CSI, ETS, HK, HSS, OR, ORSS
var.import (optional, default NULL) An integer corresponding to the number of permutations to be done for each variable to estimate variable importance
scale.models (optional, default FALSE) A logical value defining whether all models predictions should be scaled with a binomial GLM or not
nb.cpu (optional, default 1) An integer value corresponding to the number of computing resources to be used to parallelize the single models computation
seed.val  (optional, default NULL)
An integer value corresponding to the new seed value to be set

do.progress  (optional, default TRUE)
A logical value defining whether the progress bar is to be rendered or not

model  a character corresponding to the model name to be computed, must be either
ANN, CTA, FDA, GAM, GBM, GLM, MARS, MAXENT, MAXNET, RF, SRE, XGBOOST

run.name  a character corresponding to the model to be run (sp.name + pa.id + run.id)

dir.name  (optional, default .)
A character corresponding to the modeling folder

Data  a data.frame containing observations, coordinates and environmental variables
that can be obtained with the get_species_data function

weights.vec  a vector containing observation weights the concerned pseudo-absence (or allData)

calib.lines.vec  a vector containing calibration / validation lines for the concerned pseudo-absence (or allData) x repetition (or allRun) combination

eval.data  (optional, default NULL)
A data.frame containing validation observations, coordinates and environmental variables that can be obtained with the get_eval_data function

Value
A list containing for each model a list containing the following elements:

- model : the name of correctly computed model
- calib.failure : the name of incorrectly computed model
- pred : the prediction outputs for calibration data
- pred.eval : the prediction outputs for evaluation data
- evaluation : the evaluation outputs returned by the bm_FindOptimStat function
- var.import : the mean of variables importance returned by the bm_VariablesImportance function

Author(s)
Damien Georges

See Also
rpart, prune, gbm, nnet, earth, fda, mars, maxnet, randomForest, xgboost, bm_ModelingOptions,
BIOMOD_Modeling, bm_MakeFormula, bm_SampleFactorLevels, bm_FindOptimStat, bm_VariablesImportance

Other Secondary functions: bm_BinaryTransformation(), bm_CrossValidation(), bm_FindOptimStat(),
bm_MakeFormula(), bm_ModelingOptions(), bm_PlotEvalBoxplot(), bm_PlotEvalMean(), bm_PlotRangeSize(),
bm_PlotResponseCurves(), bm_PlotVarImpBoxplot(), bm_PseudoAbsences(), bm_SRE(), bm_SampleBinaryVector(),
bm_SampleFactorLevels(), bm_Tuning(), bm_VariablesImportance()
**bm_SampleBinaryVector**  

Sample binary vector

---

**Description**

This internal *biomod2* function allows the user to sample a binary vector keeping the same proportion of 0 and 1 as the initial vector.

**Usage**

```r
bm_SampleBinaryVector(obs, ratio, as.logical = FALSE, seedval = NULL)
```

**Arguments**

- **obs**
  - a vector containing binary values (either 0 or 1)
- **ratio**
  - a numeric between 0 and 1 corresponding to the proportion of obs values to sample
- **as.logical**  
  - *(optional, default FALSE)*
  - A logical value defining whether output should be returned as a vector of TRUE/FALSE values or integer values corresponding to the indices of obs elements to be kept
- **seedval**  
  - *(optional, default NULL)*
  - An integer value corresponding to the new seed value to be set

**Value**

A list containing the following elements:

- **calibration**: elements selected for calibration
- **validation**: elements selected for validation (complementary to the calibration set)

**Author(s)**

Damien Georges

**See Also**

Other Secondary functions: `bm_BinaryTransformation()`, `bm_CrossValidation()`, `bm_FindOptimStat()`, `bm_MakeFormula()`, `bm_ModelingOptions()`, `bm_PlotEvalBoxplot()`, `bm_PlotEvalMean()`, `bm_PlotRangeSize()`, `bm_PlotResponseCurves()`, `bm_PlotVarImpBoxplot()`, `bm_PseudoAbsences()`, `bm_RunModelsLoop()`, `bm_SRE()`, `bm_SampleFactorLevels()`, `bm_Tuning()`, `bm_VariablesImportance()`
Examples

```r
## Generate a binary vector
vec.a <- sample(c(0, 1), 100, replace = TRUE)

## Generate calibration / validation datasets
bm_SampleBinaryVector(obs = vec.a, ratio = 0.7)
```

---

### bm_SampleFactorLevels

Sample all levels of a factorial variable

**Description**

This internal `biomod2` function allows the user to sample all levels of all the factorial variables contained in a `data.frame` or `SpatRaster` object.

**Usage**

```r
bm_SampleFactorLevels(expl.var, mask.out = NULL, mask.in = NULL)
```

**Arguments**

- `expl.var` a `data.frame` or `SpatRaster` object containing the explanatory variables (in columns or layers)
- `mask.out` a `data.frame` or `SpatRaster` object containing the area that has already been sampled (factor levels within this mask will not be sampled)
- `mask.in` a `data.frame` or `SpatRaster` object containing areas where factor levels are to be sampled in priority. *Note that if after having explored these masks, some factor levels remain unsampled, they will be sampled in the reference input object expl.var.*

**Details**

The `expl.var`, `mask.out` and `mask.in` parameters must be coherent in terms of dimensions:

- same number of rows for `data.frame` objects
- same resolution, projection system and number of cells for `SpatRaster` objects

If `mask.in` contains several columns (`data.frame`) or layers (`SpatRaster`), then their order matters: they will be considered successively to sample missing factor levels.

- Values in `data.frame` will be understood as:
---

**bm_SampleFactorLevels**

- FALSE : out of mask
- TRUE : in mask

- Values in *SpatRaster* will be understood as:
  - NA : out of mask
  - not NA : in mask

**Value**

A vector of numeric values corresponding to either row (data.frame) or cell (*SpatRaster*) numbers, each referring to a single level of a single factorial variable.

In case no factorial variable is found in the input object, NULL is returned.

**Author(s)**

Damien Georges

**See Also**

`bm_PseudoAbsences, bm_CrossValidation`

Other Secondary functions: `bm_BinaryTransformation(), bm_CrossValidation(), bm_FindOptimStat(), bm_MakeFormula(), bm_ModelingOptions(), bm_PlotEvalBoxplot(), bm_PlotEvalMean(), bm_PlotRangeSize(), bm_PlotResponseCurves(), bm_PlotVarImpBoxplot(), bm_PseudoAbsences(), bm_RunModelsLoop(), bm_SRE(), bm_SampleBinaryVector(), bm_Tuning(), bm_VariablesImportance()`

**Examples**

```r
library(terra)

## Create raster data
ras.1 <- ras.2 <- mask.out <- rast(nrows = 10, ncols = 10)
ras.1[] <- as.factor(rep(c(1, 2, 3, 4, 5), each = 20))
ras.1 <- as.factor(ras.1)
ras.2[] <- rnorm(100)
stk <- c(ras.1, ras.2)
names(stk) <- c("varFact", "varNorm")

## define a mask for already sampled points
mask.out[1:40] <- 1

## define a list of masks where we want to sample in priority
mask.in <- list(ras.1, ras.1)
mask.in[[1]][1:80] <- NA # only level 5 should be sampled in this mask
mask.in[[1]][21:80] <- NA # only levels 1 and 5 should be sampled in this mask

## Sample all factor levels
samp1 <- bm_SampleFactorLevels(expl.var = stk, mask.out = mask.out)
samp2 <- bm_SampleFactorLevels(expl.var = stk, mask.in = mask.in)
samp3 <- bm_SampleFactorLevels(expl.var = stk, mask.out = mask.out, mask.in = mask.in)
```
bm_SRE

Surface Range Envelope

Description

This internal biomod2 function allows the user to run a rectilinear surface range envelop (SRE) (equivalent to BIOCLIM) using the extreme percentiles (as recommended by Nix or Busby, see References and Details).

Usage

bm_SRE(
  resp.var = NULL,
  expl.var = NULL,
  new.env = NULL,
  quant = 0.025,
  do.extrem = FALSE
)

Arguments

resp.var a vector, a SpatVector without associated data (if presence-only), or a SpatVector object containing binary data (0: absence, 1: presence, NA: indeterminate) for a single species that will be used to build the species distribution model(s)
Note that old format from sp are still supported such as SpatialPoints (if presence-only) or SpatialPointsDataFrame object containing binary data.

expl.var a matrix, data.frame, SpatVector or SpatRaster object containing the explanatory variables (in columns or layers) that will be used to build the SRE model
Note that old format from raster and sp are still supported such as RasterStack and SpatialPointsDataFrame objects.

new.env a matrix, data.frame, SpatVector or SpatRaster object containing the explanatory variables (in columns or layers) that will be used to predict the SRE model
Note that old format from raster and sp are still supported such as RasterStack and SpatialPointsDataFrame objects.

quant a numeric between 0 and 0.5 defining the half-quantile corresponding to the most extreme value for each variable not to be taken into account for determining the tolerance boundaries of the considered species (see Details)

do.extrem (optional, default FALSE)
A logical value defining whether a matrix containing extreme conditions supported should be returned or not
Details

*Please refer to References to get more information about surface range envelop models.*

This method is highly influenced by the extremes of the data input. Whereas a linear model can discriminate the extreme values from the main tendency, the SRE considers them as important as any other data point leading to changes in predictions.

*The more (non-collinear) variables, the more restrictive the model will be.*

Predictions are returned as binary (0 or 1) values, a site being either potentially suitable for all the variables, or out of bounds for at least one variable and therefore considered unsuitable.

quant determines the threshold from which the data will be taken into account for calibration. The default value of 0.05 induces that the 5% most extreme values will be avoided for each variable on each side of its distribution along the gradient, meaning that a total of 10% of the data will not be considered.

Value

A vector or a `SpatRaster` object, containing binary (0 or 1) values.

Author(s)

Wilfried Thuiller, Bruno Lafourcade, Damien Georges

References


See Also

`bm_PseudoAbsences`, `BIOMOD_FormatingData`, `bm_ModelingOptions`, `bm_Tuning`, `bm_RunModelsLoop`, `BIOMOD_Modeling`.

Other Secondary functions: `bm_BinaryTransformation()`, `bm_CrossValidation()`, `bm_FindOptimStat()`, `bm_MakeFormula()`, `bm_ModelingOptions()`, `bm_PlotEvalBoxplot()`, `bm_PlotEvalMean()`, `bm_PlotRangeSize()`, `bm_PlotResponseCurves()`, `bm_PlotVarImpBoxplot()`, `bm_PseudoAbsences()`, `bm_RunModelsLoop()`, `bm_SampleBinaryVector()`, `bm_SampleFactorLevels()`, `bm_Tuning()`, `bm_VariablesImportance()`
Examples

```r
library(terra)
## Load real data
data(DataSpecies)
myResp.r <- as.numeric(DataSpecies[, 'GuloGulo'])

data(bioclim_current)
myExpl.r <- rast(bioclim_current)
myRespXY <- DataSpecies[which(myResp.r == 1), c('X_WGS84', 'Y_WGS84')]
myResp.v <- classify(subset(myExpl.r, 1),
  matrix(c(-Inf, Inf, 0), ncol = 3, byrow = TRUE))
myResp.v[cellFromXY(myResp.v, myRespXY)] <- 1

## Compute SRE for several quantile values
sre.100 <- bm_SRE(resp.var = myResp.v,
  expl.var = myExpl.r,
  new.env = myExpl.r,
  quant = 0)
sre.095 <- bm_SRE(resp.var = myResp.v,
  expl.var = myExpl.r,
  new.env = myExpl.r,
  quant = 0.025)
sre.090 <- bm_SRE(resp.var = myResp.v,
  expl.var = myExpl.r,
  new.env = myExpl.r,
  quant = 0.05)

## Visualize results
res <- c(myResp.v, sre.100, sre.095, sre.090)
names(res) <- c("Original distribution", "Full data calibration",
  "Over 95 percent", "Over 90 percent")
plot(res)
```

---

### bm_Tuning

**Tune models parameters**

**Description**

This internal `biomod2` function allows to tune single model parameters and select more efficient ones based on an evaluation metric.

**Usage**

```r
bm_Tuning(
  model,
```
tuning.fun,  
do.formula = FALSE,  
do.stepAIC = FALSE,  
params.train = list(ANN.size = c(2, 4, 6, 8), ANN.decay = c(0.001, 0.01, 0.05, 0.1),  
ANN.bag = FALSE, FDA.degree = 1:2, FDA.nprune = 2:38, GAM.select = c(TRUE, FALSE),  
GAM.method = c("GCV.Cp", "GACV.Cp", "REML", "P-REML", "ML", "P-ML"), GAM.span =  
c(0.3, 0.5, 0.7), GAM.degree = 1, GBM.n.trees = c(500, 1000, 2500),  
GBM.interaction.depth = seq(2, 8, by = 3), GBM.shrinkage = c(0.001, 0.01, 0.1),  
GBM.n.minobsinnode = 10, MARS.degree = 1:2, MARS.nprune = 2:max(38, 2 *  
ncol(bm.format@data.env.var) + 1), MAXENT.algorithm = "maxnet",  
MAXENT.parallel = TRUE, RF.mtry = 1:min(10, ncol(bm.format@data.env.var)), SRE.quant = c(0, 0.0125,  
0.025, 0.05, 0.1), XGBOOST.nrounds = 50, XGBOOST.max_depth = 1, XGBOOST.eta = c(0.3,  
0.4), XGBOOST.gamma = 0, XGBOOST.ccolsample_bytree = c(0.6, 0.8),  
XGBOOST.min_child_weight = 1, XGBOOST.subsample = 0.5)

Arguments

model a character corresponding to the algorithm to be tuned, must be either ANN,  
CTA, FDA, GAM, GBM, GLM, MARS, MAXENT, MAXNET, RF, SRE, XGBOOST

tuning.fun a character corresponding to the model function name to be called through  
train function for tuning parameters (see ModelsTable dataset)

do.formula (optional, default FALSE)  
A logical value defining whether formula is to be optimized or not

do.stepAIC (optional, default FALSE)  
A logical value defining whether variables selection is to be performed for GLM  
and GAM models or not

bm.options a BIOMOD.options.default or BIOMOD.options.dataset object returned by  
the bm_ModelingOptions function

bm.format a BIOMOD.formated.data or BIOMOD.formated.data.PA object returned by  
the BIOMOD_FormatingData function

calib.lines (optional, default NULL)  
A data.frame object returned by get_calib_lines or bm_CrossValidation  
functions

metric.eval a character corresponding to the evaluation metric to be used, must be either  
AUC, Kappa or TSS for SRE only ; auc.val.avg, auc.diff.avg, or.mtp.avg,  
or.10p.avg, AICc for MAXENT only ; ROC or TSS for all other models

metric.AIC a character corresponding to the AIC metric to be used, must be either AIC or  
BIC
weights (optional, default NULL)
A vector of numeric values corresponding to observation weights (one per observation, see Details)

ctrl.train (optional, default NULL)
A trainControl object

params.train a list containing values of model parameters to be tested (see Details)

Details

Concerning ctrl.train parameter:
Set by default to:

```r
ctrl.train <- caret::trainControl(method = "repeatedcv", repeats = 3, number = 10, summaryFunction = caret::twoClassSummary, classProbs = TRUE, returnData = FALSE)
```

Concerning params.train parameter:
All elements of the list must have names matching model.parameter_name format, parameter_name being one of the parameter of the tuning.fun function called by caret package and that can be found through the getModelInfo function.
Currently, the available parameters to be tuned are the following:

- ANN size, decay, bag
- CTA maxdepth
- FDA degree, nprune
- GAM.gam span, degree
- GAM.mgcv select, method
- GBM n.trees, interaction.depth, shrinkage, n.minobsinnode
- MARS degree, nprune
- MAXENT algorithm, parallel
- RF mtry
- SRE quant
- XGBOOST nrounds, max_depth, eta, gamma, colsampl_bytree, min_child_weight, subsample

The expand.grid function is used to build a matrix containing all combinations of parameters to be tested.

Value
A BIOMOD.models.options object (see bm_ModelingOptions) with optimized parameters
Note

- No tuning for GLM and MAXNET
- MAXENT is tuned through `ENMevaluate` function which is calling either:
  - maxnet (by defining `MAXENT.algorithm = 'maxnet'`) (default)
  - Java version of Maxent defined in dismo package (by defining `MAXENT.algorithm = 'maxent.jar'`)
- SRE is tuned through `bm_SRE` function
- All other models are tuned through `train` function
- No optimization of formula for MAXENT, MAXNET, SRE and XGBOOST
- No interaction included in formula for CTA
- Variables selection only for GAM, gam and GLM

Author(s)

Frank Breiner, Maya Gueguen, Helene Blancheteau

See Also

`trainControl, train, ENMevaluate, ModelsTable, BIOMOD.models.options, bm_ModelingOptions, BIOMOD_Modeling`

Other Secondary functions: `bm_BinaryTransformation(), bm_CrossValidation(), bm_FindOptimStat(), bm_MakeFormula(), bm_ModelingOptions(), bm_PlotEvalBoxplot(), bm_PlotEvalMean(), bm_PlotRangeSize(), bm_PlotResponseCurves(), bm_PlotVarImpBoxplot(), bm_PseudoAbsences(), bm_RunModelsLoop(), bm_SRE(), bm_SampleBinaryVector(), bm_SampleFactorLevels(), bm_VariablesImportance()`

Examples

```r
library(terra)

# Load species occurrences (6 species available)
data(DataSpecies)
head(DataSpecies)

# Select the name of the studied species
myRespName <- 'GuloGulo'

# Get corresponding presence/absence data
myResp <- as.numeric(DataSpecies[, myRespName])

# Get corresponding XY coordinates
myRespXY <- DataSpecies[, c('X_WGS84', 'Y_WGS84')]

# Load environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
data(bioclim_current)
myExpl <- terra::rast(bioclim_current)
```
# Format Data with true absences
myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
                                        expl.var = myExpl,
                                        resp.xy = myRespXY,
                                        resp.name = myRespName)

# List of all models currently available in `biomod2` (and their related package and function)
# Some of them can be tuned through the `train` function of the `caret` package
# (and corresponding training function to be used is indicated)
data(ModelsTable)
ModelsTable

allModels <- c('ANN', 'CTA', 'FDA', 'GAM', 'GBM', 'GLM',
               'MARS', 'MAXENT', 'MAXNET', 'RF', 'SRE', 'XGBOOST')

# default parameters
opt.d <- bm_ModelingOptions(data.type = 'binary',
                             models = allModels,
                             strategy = 'default')

# tune parameters for Random Forest model
tuned.rf <- bm_Tuning(model = 'RF',
                      tuning.fun = 'rf', ## see in ModelsTable
                      do.formula = FALSE,
                      bm.options = opt.d@options$RF.binary.randomForest.randomForest,
                      bm.format = myBiomodData)
tuned.rf

## Not run:
# tune parameters for GAM (from mgcv package) model
tuned.gam <- bm_Tuning(model = 'GAM',
                       tuning.fun = 'gam', ## see in ModelsTable
                       do.formula = TRUE,
                       do.stepAIC = TRUE,
                       bm.options = opt.d@options$GAM.binary.mgcv.gam,
                       bm.format = myBiomodData)
tuned.gam

## End(Not run)
**bm_VariablesImportance**

**Description**

This internal **biomod2** function allows the user to compute a variable importance value for each variable involved in the given model.

**Usage**

```r
bm_VariablesImportance(
  bm.model,  
  expl.var,  
  variables = NULL, 
  method = "full_rand", 
  nb.rep = 1,  
  seed.val = NULL,  
  do.progress = TRUE,  
  temp.workdir = NULL
)
```

**Arguments**

- `bm.model` : a **biomod2_model** object (or nnet, rpart, fda, gam, glm, lm, gbm, mars, randomForest, xgb.Booster) that can be obtained with the `get_formal_model` function
- `expl.var` : a **data.frame** containing the explanatory variables that will be used to compute the variables importance
- `variables` : *(optional, default NULL)*
  A vector containing the names of the explanatory variables that will be considered
- `method` : a character corresponding to the randomisation method to be used, must be `full_rand` *(only method available so far)*
- `nb.rep` : an integer corresponding to the number of permutations to be done for each variable
- `seed.val` : *(optional, default NULL)*
  An integer value corresponding to the new seed value to be set
- `do.progress` : *(optional, default TRUE)*
  A logical value defining whether the progress bar is to be rendered or not
- `temp.workdir` : *(optional, default NULL)*
  A character value corresponding to the folder name containing temporal prediction files when using MAXENT

**Details**

For each variable to be evaluated:

1. shuffle the original variable
2. compute model prediction with shuffled variable
3. calculate Pearson’s correlation between reference and shuffled predictions
4. return score as \( 1 - \text{cor} \)
The highest the value, the less reference and shuffled predictions are correlated, and the more influence the variable has on the model. A value of 0 assumes no influence of the variable on the model.

*Note that this calculation does not account for variables’ interactions.*

The same principle is used in `randomForest`.

**Value**

A 3 columns `data.frame` containing variable’s importance scores for each permutation run:

- `expl.var`: the considered explanatory variable (the one permuted)
- `rand`: the ID of the permutation run
- `var.imp`: the variable’s importance score

**Author(s)**

Damien Georges

**See Also**

`randomForest`, `bm_RunModelsLoop`, `BIOMOD_Modeling`, `BIOMOD_EnsembleModeling`, `bm_PlotVarImpBoxplot`, `get_variables_importance`

Other Secundary functions: `bm_BinaryTransformation()`, `bm_CrossValidation()`, `bm_FindOptimStat()`, `bm_MakeFormula()`, `bm_ModelingOptions()`, `bm_PlotEvalBoxplot()`, `bm_PlotEvalMean()`, `bm_PlotRangeSize()`, `bm_PlotResponseCurves()`, `bm_PseudoAbsences()`, `bm_RunModelsLoop()`, `bm_SRE()`, `bm_SampleBinaryVector()`, `bm_SampleFactorLevels()`, `bm_Tuning()`

**Examples**

```r
## Create simple simulated data
myResp.s <- sample(c(0, 1), 20, replace = TRUE)
myExpl.s <- data.frame(var1 = sample(c(0, 1), 100, replace = TRUE),
                        var2 = rnorm(100),
                        var3 = 1:100)

## Compute variables importance
mod <- glm(var1 ~ var2 + var3, data = myExpl.s)
bm_VariablesImportance(bm.model = mod, expl.var = myExpl.s[, c('Var2', 'Var3')],
                       method = "full_rand",
                       nb.rep = 3)
```

DataSpecies

 Presence-Absence data to build test SDM

Description

A dataset covering all the continent with presence/absence data for 6 mammal species. Presence/absence were derived from range maps downloaded at IUCN.

Usage

DataSpecies

Format

A data.frame object with 2488 rows and 10 variables:

X_WGS84 Longitude
Y_WGS84 Latitude
ConnochaetesGnou Presence (1) or Absence (0) for black wildebeest
GuloGulo Presence (1) or Absence (0) for wolverine
PantheraOnca Presence (1) or Absence (0) for jaguar
PteropusGiganteus Presence (1) or Absence (0) for indian flying fox
TenrecEcaudatus Presence (1) or Absence (0) for tailless tenrec
VulpesVulpes Presence (1) or Absence (0) for red fox

getters.bm

Functions to extract informations from biomod2_model objects

Description

These functions allow the user to easily retrieve single models (formal or scaled) from biomod2_model objects from the modeling step.

Usage

## S4 method for signature 'biomod2_model'
get_formal_model(object)

## S4 method for signature 'biomod2_model'
get_scaling_model(object)

Arguments

object a biomod2_model object
**Value**

get_formal_model an object from the model slot of a `biomod2_model` object

get_scaling_model an object from the scaling_model slot of a `biomod2_model` object

**Author(s)**

Damien Georges

**See Also**

`biomod2_model`

Other Toolbox functions: `getters.out`, `load_stored_object()`, `predict.bm`, `predict.em`, `predict2.bm`, `predict2.em`

---

**Description**

These functions allow the user to easily retrieve informations stored in the different `biomod2` objects from the different modeling steps, such as modeling options and formatted data, models used or not, predictions, evaluations, variables importance.

**Usage**

```r
## S4 method for signature 'BIOMOD.formated.data'
get_species_data(obj)

## S4 method for signature 'BIOMOD.formated.data.PA'
get_species_data(obj)

## S4 method for signature 'BIOMOD.formated.data'
get_eval_data(obj)

## S4 method for signature 'BIOMOD.models.out'
get_options(obj)

## S4 method for signature 'BIOMOD.models.out'
get_calib_lines(obj, as.data.frame = FALSE, PA = NULL, run = NULL)

## S4 method for signature 'BIOMOD.models.out'
get_formal_data(obj, subinfo = NULL)

## S4 method for signature 'BIOMOD.models.out'
get_predictions()
```
getters.out

obj,
evaluation = FALSE,
full.name = NULL,
PA = NULL,
run = NULL,
algo = NULL,
model.as.col = FALSE
)

## S4 method for signature 'BIOMOD.models.out'
get_built_models(obj, full.name = NULL, PA = NULL, run = NULL, algo = NULL)

## S4 method for signature 'BIOMOD.models.out'
get_evaluations(
  obj,
  full.name = NULL,
  PA = NULL,
  run = NULL,
  algo = NULL,
  metric.eval = NULL
)

## S4 method for signature 'BIOMOD.models.out'
get_variables_importance(
  obj,
  full.name = NULL,
  PA = NULL,
  run = NULL,
  algo = NULL,
  expl.var = NULL
)

## S4 method for signature 'BIOMOD.projection.out'
get_projected_models(  
  obj,
  full.name = NULL,
  PA = NULL,
  run = NULL,
  algo = NULL,
  merged.by.algo = NULL,
  merged.by.run = NULL,
  merged.by.PA = NULL,
  filtered.by = NULL
)

## S4 method for signature 'BIOMOD.projection.out'
free(obj)
## S4 method for signature 'BIOMOD.projection.out'
get_predictions(
  obj,
  metric.binary = NULL,
  metric.filter = NULL,
  full.name = NULL,
  PA = NULL,
  run = NULL,
  algo = NULL,
  merged.by.algo = NULL,
  merged.by.run = NULL,
  merged.by.PA = NULL,
  filtered.by = NULL,
  model.as.col = FALSE,
  ...
)

## S4 method for signature 'BIOMOD.ensemble.models.out'
get_formal_data(obj, subinfo = NULL)

## S4 method for signature 'BIOMOD.ensemble.models.out'
get_built_models(
  obj,
  full.name = NULL,
  merged.by.algo = NULL,
  merged.by.run = NULL,
  merged.by.PA = NULL,
  filtered.by = NULL,
  algo = NULL
)

## S4 method for signature 'BIOMOD.ensemble.models.out'
get_kept_models(obj)

## S4 method for signature 'BIOMOD.ensemble.models.out'
get_predictions(
  obj,
  evaluation = FALSE,
  full.name = NULL,
  merged.by.algo = NULL,
  merged.by.run = NULL,
  merged.by.PA = NULL,
  filtered.by = NULL,
  algo = NULL,
  model.as.col = FALSE
)

## S4 method for signature 'BIOMOD.ensemble.models.out'
get_evaluations(
  obj,
  full.name = NULL,
  merged.by.algo = NULL,
  merged.by.run = NULL,
  merged.by.PA = NULL,
  filtered.by = NULL,
  algo = NULL,
  metric.eval = NULL
)

## S4 method for signature 'BIOMOD.ensemble.models.out'
get_variables_importance(
  obj,
  full.name = NULL,
  merged.by.algo = NULL,
  merged.by.run = NULL,
  merged.by.PA = NULL,
  filtered.by = NULL,
  algo = NULL,
  expl.var = NULL
)

Arguments

obj a BIOMOD.formated.data, BIOMOD.formated.data.PA, BIOMOD.models.out, BIOMOD.projection.out or BIOMOD.ensemble.models.out object

as.data.frame a logical defining whether output should be returned as data.frame or array object

PA (optional, default NULL)
A vector containing pseudo-absence set to be loaded, must be among PA1, PA2, ..., allData

run (optional, default NULL)
A vector containing repetition set to be loaded, must be among RUN1, RUN2, ..., allRun

subinfo a character corresponding to the information to be extracted, must be among NULL, expl.var.names, resp.var, expl.var,MinMax, eval.resp.var, eval.expl.var (see Details)

evaluation a logical defining whether evaluation data should be used or not

full.name (optional, default NULL)
A vector containing model names to be kept, must be either all or a sub-selection of model names that can be obtained with the get_built_models function

algo (optional, default NULL)
A character containing algorithm to be loaded, must be either ANN, CTA, FDA, GAM, GBM, GLM, MARS, MAXENT, MAXNET, RF, SRE, XGBOOST
model.as.col  
(optional, default FALSE)  
A boolean given to get_predictions. If TRUE prediction are returned as a wide data.frame with each column containing predictions for a single model and corresponding to the old output given by biomod2 in version < 4.2-2. If FALSE predictions are returned as a long data.frame with many additional informations readily available.

metric.eval  
(optional, default NULL)  
A vector containing evaluation metric to be kept, must be among ROC, TSS, KAPPA, ACCURACY, BIAS, POD, FAR, POFD, SR, CSI, ETS, HK, HSS, OR, ORSS

expl.var  
(optional, default NULL)  
A vector containing explanatory variables to be kept, that can be obtained with the get_formal_data(obj, subinfo = 'expl.var.names') function

merged.by.algo  
(optional, default NULL)  
A character containing merged algorithm to be loaded, must be among ANN, CTA, FDA, GAM, GBM, GLM, MARS, MAXENT, MAXNET, RF, SRE, XGBOOST, mergedAlgo

merged.by.run  
(optional, default NULL)  
A vector containing merged repetition set to be loaded, must be among RUN1, RUN2, ..., mergedRun

merged.by.PA  
(optional, default NULL)  
A vector containing merged pseudo-absence set to be loaded, must be among PA1, PA2, ..., mergedData

filtered.by  
(optional, default NULL)  
A vector containing evaluation metric selected to filter single models to build the ensemble models, must be among ROC, TSS, KAPPA, ACCURACY, BIAS, POD, FAR, POFD, SR, CSI, ETS, HK, HSS, OR, ORSS

metric.binary  
(optional, default NULL)  
A vector containing evaluation metric selected to transform predictions into binary values, must be among ROC, TSS, KAPPA, ACCURACY, BIAS, POD, FAR, POFD, SR, CSI, ETS, HK, HSS, OR, ORSS

metric.filter  
(optional, default NULL)  
A vector containing evaluation metric to filter predictions, must be among ROC, TSS, KAPPA, ACCURACY, BIAS, POD, FAR, POFD, SR, CSI, ETS, HK, HSS, OR, ORSS

...  
(optional, one or several of the following arguments depending on the selected function)

Value

get_species_data  
a data.frame combining data.species, coord, data.env.var (and PA.table) slots of BIOMOD.formated.data (or BIOMOD.formated.data.PA) object

get_eval_data  
a data.frame combining eval.data.species, eval.coord, eval.data.env.var slots of BIOMOD.formated.data or BIOMOD.formated.data.PA object

get_options  
a BIOMOD.stored.options-class object from the models.options slot of a BIOMOD.models.out-class object

get_calib_lines  
a BIOMOD.stored.data.frame-class object from the calib.lines slot of a BIOMOD.models.out object
get_projected_models a vector from the models.projected slot of a BIOMOD.projection.out object
get_predictions a BIOMOD.stored.data object from the proj.out slot of a BIOMOD.models.out, BIOMOD.projection.out or BIOMOD.ensemble.models.out object
get_kept_models a vector containing names of the kept models of a BIOMOD.ensemble.models.out object
get_formal_data depending on the subinfo parameter:
   NULL a BIOMOD.stored.formated.data-class (or BIOMOD.stored.models.out-class) object from the formated.input.data (or models.out) slot of a BIOMOD.models.out (or BIOMOD.ensemble.models.out) object
   expl.var.names a vector from the expl.var.names slot of a BIOMOD.models.out or BIOMOD.ensemble.models.out object
   resp.var a vector from the data.species slot of the formated.input.data slot of a BIOMOD.models.out or BIOMOD.ensemble.models.out object
   expl.var a data.frame from the data.env.var slot of the formated.input.data slot of a BIOMOD.models.out or BIOMOD.ensemble.models.out object
   MinMax a list of minimum and maximum values (or levels if factorial) of variable contained in the data.env.var slot of the formated.input.data slot of a BIOMOD.models.out or BIOMOD.ensemble.models.out object
   eval.resp.var a vector from the eval.data.species slot of the formated.input.data slot of a BIOMOD.models.out or BIOMOD.ensemble.models.out object
   eval.expl.var a data.frame from the eval.data.env.var slot of the formated.input.data slot of a BIOMOD.models.out or BIOMOD.ensemble.models.out object
get_built_models a vector from the models.computed slot (or em.computed) of a BIOMOD.models.out (or BIOMOD.ensemble.models.out) object
get_evaluations a data.frame from the models.evaluation slot (or model_evaluation of each model in em.computed) of a BIOMOD.models.out (or BIOMOD.ensemble.models.out) object. Contains evaluation metric for different models and dataset. Evaluation metric are calculated on the calibrating data (column calibration), on the cross-validation data (column validation) or on the evaluation data (column evaluation).
For cross-validation data, see CV.[...] parameters in BIOMOD_Modeling function ; for evaluation data, see eval.[...] parameters in BIOMOD_FormatingData.
get_variables_importance a BIOMOD.stored.data.frame-class from the variables.importance slot (or model_variables_importance of each model in em.models) of a BIOMOD.models.out (or BIOMOD.ensemble.models.out) object

Author(s)
   Damien Georges

See Also
   BIOMOD.models.out, BIOMOD.projection.out, BIOMOD.ensemble.models.out
   Other Toolbox functions: getters.bm, load_stored_object(), predict.bm, predict.em, predict2.bm, predict2.em
load_stored_object  

Functions to load BIOMOD.stored.data objects

Description

This functions allow the user to load BIOMOD.stored.data objects into memory.

Usage

load_stored_object(obj, ...)

## S4 method for signature 'BIOMOD.stored.data'
load_stored_object(obj, layer = 1)

## S4 method for signature 'BIOMOD.stored.SpatRaster'
load_stored_object(obj, layer = 1)

Arguments

obj     a BIOMOD.stored.data object
...
layer   an integer corresponding to the layer ID to be extracted when multilayer object considered

Author(s)

Damien Georges

See Also

BIOMOD.stored.data
Other Toolbox functions: getters.bm, getters.out, predict.bm, predict.em, predict2.bm, predict2.em

ModelsTable  

Single models package and functions

Description

A data.frame containing for each single model available in biomod2 the package and functions to be called.

Usage

ModelsTable
OptionsBigboss

Format

A `data.frame` object with 12 rows and 5 variables:

- **model**: all single models that can be computed in `biomod2`
- **type**: data type associated to the models
- **package**: R package used
- **func**: function used in the R package
- **train**: function called by `caret` for the tuning

All single models available are the following:

- ANN (`nnet`)
- CTA (`rpart`)
- FDA (`fda`)
- GAM (`gam`, `gam` or `bam`)
- GBM (`gbm`)
- GLM (`glm`)
- MARS (`earth`)
- MAXENT ([https://biodiversityinformatics.amnh.org/open_source/maxent/](https://biodiversityinformatics.amnh.org/open_source/maxent/))
- MAXNET (`maxnet`)
- RF (`randomForest`)
- SRE (`bm_SRE`)
- XGBOOST (`xgboost`)

---

OptionsBigboss  Bigboss pre-defined parameter values for single models

Description

A `BIOMOD.models.options` object containing for each single model available in `biomod2` the parameter values pre-defined by `biomod2` team.

Usage

OptionsBigboss
A `BIOMOD.models.options` object with some changed values:

- **ANN.binary.nnet.nnet**
  - size = 5
  - decay = 5
  - trace = FALSE
  - rang = 0.1
  - maxit = 200

- **CTA.binary.rpart.rpart**
  - method = 'class'
  - control = list(xval = 5, minbucket = 5, minsplit = 5, cp = 0.001, maxdepth = 25)
  - cost = NULL

- **FDA.binary.mda.fda**
  - method = 'mars'

- **GAM.binary.gam.gam**
  - family = binomial(link = 'logit')
  - method = 'GCV.Cp'
  - control = list(epsilon = 1e-06, trace = FALSE, maxit = 100)

- **GBM.binary.gbm.gbm**
  - n.trees = 2500
  - interaction.depth = 7
  - n.minobsinnode = 5
  - shrinkage = 0.001
  - cv.folds = 3
  - keep.data = FALSE
  - n.cores = 1

- **GLM.binary.stats.glm**
  - family = binomial(link = 'logit')
  - mustart = 0.5
  - control = glm.control(maxit = 50)

- **MARS.binary.earth.earth**
  - glm = list(family = binomial(link = 'logit'))
  - n.cross = 0
  - nk = NULL
  - penalty = 2
  - thresp = 0.001
  - nprune = NULL
  - pmethod = 'backward'

- **MAXENT.binary.MAXENT.MAXENT**
  - path_to_maxent.jar = '.'

- **RF.binary.randomForest.randomForest**
  - type = 'classification'
  - ntree = 500
  - mtry = NULL
  - strata = factor(c(0, 1))
  - sampsize = NULL
  - nodesize = 5
plot.BIOMOD.formated.data,missing-method

- maxnodes = NULL

SRE.binary.biomod2.bm_SRE • do.extrem = TRUE

XGBOOST.binary.xgboost.xgboost • params = list(max_depth = 2, eta = 1)
  - nthread = 2
  - nrounds = 4
  - objective = 'binary:logistic'

Description

Plot the spatial distribution of presences, absences and pseudo-absences among the different potential dataset (calibration, validation and evaluation). Available only if coordinates were given to BIOMOD_FormatingData.

Usage

```r
## S4 method for signature 'BIOMOD.formated.data,missing'
plot(
  x,
  calib.lines = NULL,
  plot.type,
  plot.output,
  PA,
  run,
  plot.eval,
  point.size = 1.5,
  do.plot = TRUE
)
```

Arguments

- **x**
  a BIOMOD.formated.data or BIOMOD.formated.data.PA object. Coordinates must be available to be able to use plot.

- **calib.lines**
  (optional, default NULL)
  an data.frame object returned by get_calib_lines or bm_CrossValidation functions, to explore the distribution of calibration and validation datasets

- **plot.type**
  a character, either 'points' (default) or 'raster' (if environmental variables were given as a raster). With plot.type = 'points' occurrences will be represented as points (better when using fine-grained data). With plot.type = 'raster' occurrences will be represented as a raster (better when using coarse-grained data)
plot.output  a character, either 'facet' (default) or 'list'. plot.output determines whether plots are returned as a single facet with all plots or a list of individual plots (better when there are numerous graphics)

PA  (optional, default 'all')
    If x is a BIOMOD.formated.data.PA object, a vector containing pseudo-absence set to be represented

run  (optional, default 'all')
    If calib.lines provided, a vector containing repetition set to be represented

plot.eval  (optional, default TRUE)
    A logical defining whether evaluation data should be added to the plot or not

point.size  a numeric to adjust the size of points when plot.type = 'points'.

do.plot  (optional, default TRUE)
    A logical defining whether the plot is to be rendered or not

Value

  a list with the data used to generate the plot and a ggplot2 object

Author(s)

  Remi Patin

Examples

  library(terra)

  # Load species occurrences (6 species available)
  data(DataSpecies)
  head(DataSpecies)

  # Select the name of the studied species
  myRespName <- 'GuloGulo'

  # Get corresponding presence/absence data
  myResp <- as.numeric(DataSpecies[, myRespName])

  # Get corresponding XY coordinates
  myRespXY <- DataSpecies[, c('X_WGS84', 'Y_WGS84')]

  # Load environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
  data(bioclim_current)
  myExpl <- terra::rast(bioclim_current)

  ## ----------------------------------------------------------------------- #

  # Format Data with true absences
  myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
                                        expl.var = myExpl,
predict.bm

## S4 method for signature 'biomod2_model'

predict(object, newdata, ...)

Arguments

- **object**: a `biomod2_model` object
- **newdata**: a `data.frame` or `SpatRaster` object containing data for new predictions
- **...**: (optional)

Description

This function allows the user to predict single models from `biomod2_model` on (new) explanatory variables.

Usage

```r
## S4 method for signature 'biomod2_model'
predict(object, newdata, ...)
```

Author(s)

Damien Georges

See Also

- `biomod2_model`
- Other Toolbox functions: `getters.bm`, `getters.out`, `load_stored_object()`, `predict.em`, `predict2.bm`, `predict2.em`
**predict.em**

*Functions to get predictions from biomod2_ensemble_model objects*

**Description**

This function allows the user to predict single models from `biomod2_ensemble_model` on (new) explanatory variables.

**Arguments**

- **object**
  - `a biomod2_ensemble_model object`
- **newdata**
  - `a data.frame or SpatRaster object containing data for new predictions`
- **...**
  - `(optional)`

**Author(s)**

Damien Georges

**See Also**

`biomod2_ensemble_model`

Other Toolbox functions: `getters.bm`, `getters.out`, `load_stored_object()`, `predict.bm`, `predict2.bm`, `predict2.em`

**summary,BIOMOD.formated.data-method**

*summary method for BIOMOD.formated.data object class*

**Description**

Summarize the number of presences, absences and pseudo-absences among the different potential dataset (calibration, validation and evaluation).

**Usage**

```r
## S4 method for signature 'BIOMOD.formated.data'
summary(object, calib.lines = NULL)
```

**Arguments**

- **object**
  - `a BIOMOD.formated.data or BIOMOD.formated.data.PA object returned by the BIOMOD_FormatingData function`
- **calib.lines**
  - `(optional, default NULL)`
    - `an array object returned by get_calib_lines or bm_CrossValidation functions, to explore the distribution of calibration and validation datasets`
Value

a data.frame

Author(s)

Remi Patin

Examples

library(terra)

# Load species occurrences (6 species available)
data(DataSpecies)
head(DataSpecies)

# Select the name of the studied species
myRespName <- 'GuloGulo'

# Get corresponding presence/absence data
myResp <- as.numeric(DataSpecies[, myRespName])

# Get corresponding XY coordinates
myRespXY <- DataSpecies[, c('X_WGS84', 'Y_WGS84')]

# Load environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
data(bioclim_current)
myExpl <- terra::rast(bioclim_current)

## ----------------------------------------------------------------------- #
# Format Data with true absences
myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
  expl.var = myExpl,
  resp.xy = myRespXY,
  resp.name = myRespName)

myBiomodData
summary(myBiomodData)
Index

* ANN
  bm_RunModelsLoop, 98
* CTA
  bm_RunModelsLoop, 98
* FDA
  bm_RunModelsLoop, 98
* GAM
  bm_RunModelsLoop, 98
* GBM
  bm_RunModelsLoop, 98
* GLM
  bm_RunModelsLoop, 98
* MARS
  bm_RunModelsLoop, 98
* MAXENT
  bm_RunModelsLoop, 98
* Main functions
  BIOMOD_EnsembleForecasting, 29
  BIOMOD_EnsembleModeling, 33
  BIOMOD_FormatingData, 40
  BIOMOD_LoadModels, 47
  BIOMOD_Modeling, 49
  BIOMOD_Projection, 56
  BIOMOD_RangeSize, 60
* Pearson
  bm_VariablesImportance, 110
* Plot functions
  bm_PlotEvalBoxplot, 79
  bm_PlotEvalMean, 81
  bm_PlotRangeSize, 84
  bm_PlotResponseCurves, 87
  bm_PlotVarImpBoxplot, 91
* RF
  bm_RunModelsLoop, 98
* SRE
  bm_PseudoAbsences, 93
  bm_RunModelsLoop, 98
* Secondary functions
  bm_BinaryTransformation, 63
  bm_CrossValidation, 65
  bm_FindOptimStat, 70
  bm_MakeFormula, 73
  bm_ModelingOptions, 75
  bm_PlotEvalBoxplot, 79
  bm_PlotEvalMean, 81
  bm_PlotRangeSize, 84
  bm_PlotResponseCurves, 87
  bm_PlotVarImpBoxplot, 91
* Toolbox functions
  getters.bm, 113
  getters.out, 114
  load_stored_object, 120
  predict.bm, 125
  predict.em, 126
* Toolbox objects
  BIOMOD.ensemble.models.out, 4
  BIOMOD.formated.data, 7
  BIOMOD.formated.data.PA, 11
  BIOMOD.models.options, 15
  BIOMOD.models.out, 16
  BIOMOD.options.dataset, 18
  BIOMOD.options.default, 20
  BIOMOD.projection.out, 21
  BIOMOD.stored.data, 24
  biomod2_ensemble_model, 25
  biomod2_model, 27
* XGBOOST
  bm_RunModelsLoop, 98
* auc
  bm_FindOptimStat, 70
* binary
bm_PlotResponseCurves, 87

*b sample
  bm_SampleBinaryVector, 101
  bm_SampleFactorLevels, 102

*b shuffle
  bm_VariablesImportance, 110

*b species
  BIOMOD_RangeSize, 60
  bm_PlotRangeSize, 84

*b sre
  bm_SRE, 104

*b surface
  bm_SRE, 104

*b threshold
  bm_BinaryTransformation, 63

*b tree
  BIOMOD_Modeling, 49

*b tss
  bm_FindOptimStat, 70

*b weights
  BIOMOD_EnsembleModeling, 33

ANN_biomod2_model-class
  (biomod2_model), 27

bam, 52, 54, 121
bioclim_current, 3
bioclim_future, 4

BIOMOD.ensemble.models.out, 4, 5, 10, 14, 16, 17, 20, 21, 23, 25, 26, 28, 29, 38, 47, 79, 80, 82, 83, 87–89, 91, 92, 114, 117, 119

BIOMOD.ensemble.models.out-class
  (BIOMOD.ensemble.models.out), 4

BIOMOD.formated.data, 5, 7, 9, 14, 16, 17, 19–21, 23–26, 28, 44, 50, 66, 75, 99, 107, 117, 118, 123, 126

BIOMOD.formated.data,data.frame,ANY-method
  (BIOMOD.formated.data), 7

BIOMOD.formated.data,numerical.data.frame-method
  (BIOMOD.formated.data), 7

BIOMOD.formated.data,numerical,matrix-method
  (BIOMOD.formated.data), 7

BIOMOD.formated.data,numerical,SpatRaster-method
  (BIOMOD.formated.data), 7

BIOMOD.formated.data-class
  (BIOMOD.formated.data), 7

BIOMOD.formated.data.PA, 5, 10, 11, 16, 17, 19–21, 23, 25, 26, 28, 50, 66, 75, 96, 99, 107, 117, 118, 123, 124, 126

BIOMOD.formated.data.PA,numerical.data.frame-method
  (BIOMOD.formated.data.PA), 11

BIOMOD.formated.data.PA,numerical,SpatRaster-method
  (BIOMOD.formated.data.PA), 11

BIOMOD.formated.data.PA-class
  (BIOMOD.formated.data.PA), 11

BIOMOD.models.options, 5, 10, 14, 15, 17, 20, 21, 23–26, 28, 51, 53, 76, 77, 99, 108, 109, 121, 122

BIOMOD.models.options-class
  (BIOMOD.models.options), 15

BIOMOD.models.out, 5, 10, 14, 16, 20, 21, 23–26, 28, 34, 47, 54, 56, 79, 80, 82, 83, 87–89, 91, 92, 114, 117–119

BIOMOD.models.out-class
  (BIOMOD.models.out), 16

BIOMOD.options.dataset, 5, 10, 14–17, 18, 19, 21, 23, 25, 26, 28, 53, 107

BIOMOD.options.dataset,character-method
  (BIOMOD.options.dataset), 18

BIOMOD.options.dataset-class
  (BIOMOD.options.dataset), 18

BIOMOD.options.default, 5, 10, 14, 16, 17, 20, 20, 23, 26, 28, 107

BIOMOD.options.default,character,character-method
  (BIOMOD.options.default), 20

BIOMOD.options.default-class
  (BIOMOD.options.default), 20

BIOMOD.projection.out, 5, 10, 14, 16, 17, 20, 21, 22, 25, 26, 28, 29, 31, 58, 114, 117, 119

BIOMOD.projection.out-class
  (BIOMOD.projection.out), 21

BIOMOD.stored.data, 5, 10, 14, 16, 17, 20–23, 24, 26, 28, 119, 120

BIOMOD.stored.data-class
  (BIOMOD.stored.data), 24

BIOMOD.stored.data.frame-class
  (BIOMOD.stored.data), 24

BIOMOD.stored.files-class
  (BIOMOD.stored.data), 24

BIOMOD.stored.formated.data-class
  (BIOMOD.stored.data), 24

BIOMOD.stored.models.out-class
  (BIOMOD.stored.data), 24

BIOMOD.stored.options-class
  (BIOMOD.stored.data), 24
INDEX

BIOMOD.stored.SpatRaster-class
(BIOMOD.stored.data), 24
biomod2_ensemble_model, 5, 10, 14, 16, 17,
20, 21, 23, 25, 28, 126
biomod2_ensemble_model-class
(biomod2_ensemble_model), 25
biomod2_model-class (biomod2_model), 27
BIOMOD.EnsembleForecasting, 4, 21, 23, 25,
29, 30, 38, 44, 48, 54, 57, 58, 61, 64
BIOMOD.EnsembleModeling, 4, 5, 16, 17,
24–26, 29, 31, 33, 44, 47, 48, 54, 58,
61, 73, 79, 80, 82, 83, 87–89, 91, 92,
112
BIOMOD_FormatingData, 7, 10, 11, 14, 16, 19,
29, 31, 36, 38, 40, 48, 50, 52–54, 57,
58, 61, 66, 69, 75, 76, 88, 96, 99,
103, 107, 119, 123, 126
BIOMOD_LoadModels, 4, 5, 16, 17, 31, 38, 44,
47, 54, 58, 61
BIOMOD_Modeling, 5, 7, 10, 11, 14–18, 20, 21,
24, 25, 27, 28, 30, 31, 33, 34, 36, 38,
44, 47, 48, 49, 54, 56–58, 61, 69, 73,
76, 77, 79, 80, 82, 83, 87–89, 91, 92,
98, 100, 105, 109, 112, 119
BIOMOD_PresenceOnly, 4, 5, 16, 17
BIOMOD_Projection, 16, 17, 21, 23, 25, 29,
31, 38, 44, 48, 54, 56, 61, 64, 76
BIOMOD_RangeSize, 31, 38, 44, 48, 54, 58, 60,
84, 85
BIOMOD_RangeSize, data.frame, data.frame-method
(BIOMOD_RangeSize), 60
BIOMOD_RangeSize, SpatRaster, SpatRaster-method
(biomod2_RangeSize), 60
bm_BinaryTransformation, 63, 69, 73, 74,
77, 80, 83, 85, 89, 92, 96, 100, 101,
103, 105, 109, 112
bm_BinaryTransformation, data.frame-method
(bm_BinaryTransformation), 63
bm_BinaryTransformation, matrix-method
(bm_BinaryTransformation), 63
bm_BinaryTransformation, numeric-method
(bm_BinaryTransformation), 63
bm_BinaryTransformation, SpatRaster-method
(bm_BinaryTransformation), 63
bm_CalculateStat (bm_FindOptimStat), 70
bm_CrossValidation, 7, 10, 11, 14, 19, 38,
44, 53, 54, 64, 65, 73–75, 77, 80, 83,
85, 89, 92, 96, 99–101, 103, 105,
107, 109, 112, 123, 126
bm_CrossValidation_block
(bm_CrossValidation), 65
bm_CrossValidation_block, BIOMOD.formated.data-method
(bm_CrossValidation), 65
bm_CrossValidation_block, BIOMOD.formated.data.PA-method
(bm_CrossValidation), 65
bm_CrossValidation_env
(bm_CrossValidation), 65
bm_CrossValidation_env, BIOMOD.formated.data-method
(bm_CrossValidation), 65
bm_CrossValidation_env, BIOMOD.formated.data.PA-method
(bm_CrossValidation), 65
bm_CrossValidation_kfold
(bm_CrossValidation), 65
bm_CrossValidation_kfold, BIOMOD.formated.data-method
(bm_CrossValidation), 65
bm_CrossValidation_kfold, BIOMOD.formated.data.PA-method
(bm_CrossValidation), 65
bm_CrossValidation_random
(bm_CrossValidation), 65
bm_CrossValidation_random, BIOMOD.formated.data-method
(bm_CrossValidation), 65
bm_CrossValidation_random, BIOMOD.formated.data.PA-method
(bm_CrossValidation), 65
bm_CrossValidation_strat
(bm_CrossValidation), 65
bm_CrossValidation_strat, BIOMOD.formated.data-method
(bm_CrossValidation), 65
bm_CrossValidation_strat, BIOMOD.formated.data.PA-method
(bm_CrossValidation), 65
bm_CrossValidation_user_defined
(bm_CrossValidation), 65
bm_CrossValidation_user_defined, BIOMOD.formated.data-method
(bm_CrossValidation), 65
bm_CrossValidation_user_defined, BIOMOD.formated.data.PA-method
(bm_CrossValidation), 65
bm_FindOptimStat, 64, 69, 70, 74, 77, 80, 83,
85, 89, 92, 96, 100, 101, 103, 105,
109, 112
bm_MakeFormula, 64, 69, 73, 73, 77, 80, 83,
85, 89, 92, 96, 100, 101, 103, 105,
109, 112
bm_ModelingOptions, 15–18, 20, 21, 31, 38,
51–54, 64, 69, 73, 74, 75, 80, 83, 85,
89, 92, 96, 99–101, 103, 105,
107–109, 112
bm_PlotEvalBoxplot, 5, 17, 38, 54, 64, 69, 73, 74, 77, 79, 83, 85, 89, 92, 96, 100, 101, 103, 105, 109, 112
bm_PlotEvalMean, 5, 17, 38, 54, 64, 69, 73, 74, 77, 80, 81, 85, 89, 92, 96, 100, 101, 103, 105, 109, 112
bm_PlotRangeSize, 5, 17, 38, 54, 64, 69, 73, 74, 77, 80, 83, 84, 89, 92, 96, 100, 101, 103, 105, 109, 112
bm_PlotResponseCurves, 5, 17, 38, 54, 64, 69, 73, 74, 77, 80, 83, 85, 87, 92, 96, 100, 101, 103, 105, 109, 112
bm_PseudoAbsences, 14, 43, 44, 64, 69, 73, 74, 77, 80, 83, 85, 89, 91, 96, 100, 101, 103, 105, 109, 112
bm_PseudoAbsences_disk (bm_PseudoAbsences), 93
bm_PseudoAbsences_disk, ANY, SpatRaster-method (bm_PseudoAbsences), 93
bm_PseudoAbsences_disk, ANY, SpatVector-method (bm_PseudoAbsences), 93
bm_PseudoAbsences_random (bm_PseudoAbsences), 93
bm_PseudoAbsences_random, ANY, SpatRaster-method (bm_PseudoAbsences), 93
bm_PseudoAbsences_random, ANY, SpatVector-method (bm_PseudoAbsences), 93
bm_PseudoAbsences_sre (bm_PseudoAbsences), 93
bm_PseudoAbsences_sre, ANY, SpatRaster-method (bm_PseudoAbsences), 93
bm_PseudoAbsences_sre, ANY, SpatVector-method (bm_PseudoAbsences), 93
bm_PseudoAbsences_user_defined (bm_PseudoAbsences), 93
bm_PseudoAbsences_user_defined, ANY, SpatRaster-method (bm_PseudoAbsences), 93
bm_PseudoAbsences_user_defined, ANY, SpatVector-method (bm_PseudoAbsences), 93
bm_RunModel, 27, 28
bm_RunModel (bm_RunModelsLoop), 98
bm_RunModelsLoop, 10, 14, 20, 21, 64, 69, 73, 74, 77, 80, 83, 85, 89, 92, 96, 98, 101, 103, 105, 109, 112
bm_SampleBinaryVector, 64, 69, 73, 74, 77, 80, 83, 85, 89, 92, 96, 100, 101, 103, 105, 109, 112
bm_SampleFactorLevels, 64, 69, 73, 74, 77, 80, 83, 85, 89, 92, 96, 100, 101, 102, 105, 109, 112
bm_SRE, 44, 53, 64, 69, 73, 74, 77, 80, 83, 85, 89, 92, 95, 96, 100, 101, 103, 104, 109, 112, 121
bm_Tuning, 7, 10, 11, 14, 16, 20, 21, 53, 54, 64, 69, 73, 74, 76, 77, 80, 83, 85, 89, 92, 96, 100, 101, 103, 105, 106, 112
bm_VariablesImportance, 5, 17, 38, 54, 64, 69, 73, 74, 77, 80, 83, 85, 89, 92, 96, 100, 101, 103, 105, 109, 110
CTA_biomod2_model-class (biomod2_model), 27
DataSpecies, 113
earth, 52, 54, 100, 121
EMca_biomod2_model-class (biomod2_ensemble_model), 25
EMci_biomod2_model-class (biomod2_ensemble_model), 25
EMcv_biomod2_model-class (biomod2_ensemble_model), 25
EMmean_biomod2_model-class (biomod2_ensemble_model), 25
EMmedian_biomod2_model-class (biomod2_ensemble_model), 25
EMnew_biomod2_model-class (biomod2_ensemble_model), 25
ENMevaluate, 109
expand.grid, 108
facet_wrap, 22, 80
fda, 52, 54, 100, 121
FDAbiomod2_model-class (biomod2_model), 27
formalArgs, 76
formula, 74
free(getters.out), 114
free,BIOMOD.projection.out-method (getters.out), 114
gam, 52, 54, 121
<table>
<thead>
<tr>
<th>Function/Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>get_built_models</td>
<td>30, 34, 36, 47, 57, 88, 117</td>
</tr>
<tr>
<td>get_built_models (getters.out)</td>
<td>114</td>
</tr>
<tr>
<td>get_built_models (BIOMOD.ensemble.models.out-method)</td>
<td></td>
</tr>
<tr>
<td>get_calib_lines</td>
<td>19, 75, 107, 123, 126</td>
</tr>
<tr>
<td>get_calib_lines (getters.out)</td>
<td>114</td>
</tr>
<tr>
<td>get_calib_lines (BIOMOD.models.out-method)</td>
<td></td>
</tr>
<tr>
<td>get_eval_data</td>
<td>114</td>
</tr>
<tr>
<td>get_eval_data (getters.out)</td>
<td></td>
</tr>
<tr>
<td>get_eval_data (BIOMOD.formated.data-method)</td>
<td></td>
</tr>
<tr>
<td>get_evaluations</td>
<td>54, 80, 83</td>
</tr>
<tr>
<td>get_evaluations (getters.out)</td>
<td>114</td>
</tr>
<tr>
<td>get_evaluations (BIOMOD.ensemble.models.out-method)</td>
<td></td>
</tr>
<tr>
<td>get_evaluations (BIOMOD.models.out-method)</td>
<td></td>
</tr>
<tr>
<td>get_formal_data</td>
<td>118</td>
</tr>
<tr>
<td>get_formal_data (getters.out)</td>
<td>114</td>
</tr>
<tr>
<td>get_formal_data (BIOMOD.ensemble.models.out-method)</td>
<td></td>
</tr>
<tr>
<td>get_formal_data (BIOMOD.models.out-method)</td>
<td></td>
</tr>
<tr>
<td>get_formal_model</td>
<td>111</td>
</tr>
<tr>
<td>get_formal_model (getters.bm)</td>
<td>113</td>
</tr>
<tr>
<td>get_formal_model (biomod2_model-method)</td>
<td></td>
</tr>
<tr>
<td>get_kept_models</td>
<td>114</td>
</tr>
<tr>
<td>get_kept_models (getters.out)</td>
<td></td>
</tr>
<tr>
<td>get_kept_models (BIOMOD.ensemble.models.out-method)</td>
<td></td>
</tr>
<tr>
<td>get_optim_value</td>
<td>54, 72</td>
</tr>
<tr>
<td>get_optim_value (bm_FindOptimStat)</td>
<td>70</td>
</tr>
<tr>
<td>get_options</td>
<td>114</td>
</tr>
<tr>
<td>get_options (BIOMOD.models.out-method)</td>
<td></td>
</tr>
<tr>
<td>get_predictions</td>
<td>22, 118</td>
</tr>
<tr>
<td>get_predictions (getters.out)</td>
<td>114</td>
</tr>
<tr>
<td>get_predictions (BIOMOD.ensemble.models.out-method)</td>
<td></td>
</tr>
<tr>
<td>get_predictions (BIOMOD.projection.out-method)</td>
<td></td>
</tr>
<tr>
<td>get_projected_models</td>
<td>114</td>
</tr>
<tr>
<td>get_projected_models (BIOMOD.projection.out-method)</td>
<td></td>
</tr>
<tr>
<td>get_scaling_model</td>
<td>113</td>
</tr>
<tr>
<td>get_scaling_model (biomod2_model-method)</td>
<td></td>
</tr>
<tr>
<td>get_scaling_model (getters.bm)</td>
<td></td>
</tr>
<tr>
<td>get_species_data</td>
<td>114</td>
</tr>
<tr>
<td>get_species_data (BIOMOD.formated.data-method)</td>
<td></td>
</tr>
<tr>
<td>get_species_data (BIOMOD.formated.data.PA-method)</td>
<td></td>
</tr>
<tr>
<td>get_variables_importance</td>
<td>92, 112</td>
</tr>
<tr>
<td>get_variables_importance (getters.out)</td>
<td></td>
</tr>
<tr>
<td>get_variables_importance (BIOMOD.ensemble.models.out-method)</td>
<td></td>
</tr>
<tr>
<td>get_variables_importance (BIOMOD.models.out-method)</td>
<td></td>
</tr>
<tr>
<td>getSpeciesInfo</td>
<td>108</td>
</tr>
<tr>
<td>getters.bm</td>
<td>113, 119, 120, 125, 126</td>
</tr>
<tr>
<td>getters.out</td>
<td>114, 114, 120, 125, 126</td>
</tr>
<tr>
<td>glm</td>
<td>52, 54, 121</td>
</tr>
<tr>
<td>GLM_biomod2_model-class</td>
<td></td>
</tr>
<tr>
<td>glm</td>
<td></td>
</tr>
<tr>
<td>load</td>
<td>38, 54</td>
</tr>
<tr>
<td>load_stored_object</td>
<td>114, 119, 120, 125, 126</td>
</tr>
<tr>
<td>load_stored_object (BIOMOD.stored.data-method)</td>
<td></td>
</tr>
<tr>
<td>load_stored_object (BIOMOD.stored.SpatRaster-method)</td>
<td></td>
</tr>
<tr>
<td>load_stored_object (BIOMOD.stored.data)</td>
<td></td>
</tr>
<tr>
<td>load_stored_object (BIOMOD.stored.SpatRaster-method)</td>
<td></td>
</tr>
<tr>
<td>MARS_biomod2_model-class</td>
<td>100</td>
</tr>
<tr>
<td>MAXNET_biomod2_model-class</td>
<td>27</td>
</tr>
<tr>
<td>MAXENT_biomod2_model-class</td>
<td>27</td>
</tr>
<tr>
<td>maxnet</td>
<td>52, 54, 100, 121</td>
</tr>
<tr>
<td>MAXNET_biomod2_model-class</td>
<td></td>
</tr>
<tr>
<td>ModelsTable</td>
<td>76, 77, 107, 109, 120</td>
</tr>
<tr>
<td>nnet</td>
<td>52, 54, 100, 121</td>
</tr>
</tbody>
</table>
OptionsBigboss, 53, 121
PackedSpatRaster, 24
plot, 22
plot, BIOMOD.formated.data, missing-method, 123
plot, BIOMOD.projection.out, missing-method (BIOMOD.projection.out), 21
predict, biomod2_model-method (predict.bm), 125
predict.biomod2_model (predict.bm), 125
predict.bm, 114, 119, 120, 125, 126
predict2.bm, 114, 119, 120, 125, 126
predict2.em, 114, 119, 120, 125, 126
print, BIOMOD.models.options-method (BIOMOD.models.options), 15
print, BIOMOD.options.dataset-method (BIOMOD.options.dataset), 18
prune, 100
randomForest, 52, 54, 100, 112, 121
RasterLayer, 60
RF_biomod2_model-class (biomod2_model), 27
rpart, 52, 54, 100, 121
s, 74
show, BIOMOD.ensemble.models.out-method (BIOMOD.ensemble.models.out), 4
show, BIOMOD.formated.data-method (BIOMOD.formated.data), 7
show, BIOMOD.models.options-method (BIOMOD.models.options), 15
show, BIOMOD.models.out-method (BIOMOD.models.out), 16
show, BIOMOD.options.dataset-method (BIOMOD.options.dataset), 18
show, BIOMOD.projection.out-method (BIOMOD.projection.out), 21
show, biomod2_ensemble_model-method (biomod2_ensemble_model), 25
show, biomod2_model-method (biomod2_model), 27
SpatialPoints, 95
SpatialPointsDataFrame, 95
SpatRaster, 3, 4, 8, 9, 12, 13, 29, 31, 41, 43, 44, 57, 58, 60, 64, 71, 88, 95, 102–105, 125, 126
SpatVector, 8, 9, 12, 41, 43, 71, 104
SRE_biomod2_model-class (biomod2_model), 27
summary, BIOMOD.formated.data-method, 126
train, 19, 107, 109
trainControl, 108, 109
xgboost, 53, 54, 100, 121
XGBOOST_biomod2_model-class (biomod2_model), 27