Package 'biomod2'

March 1, 2016

Type Package

Title Ensemble Platform for Species Distribution Modeling

Version 3.3-7

Date 2016-03-01

Author Wilfried Thuiller [aut, cre],
Damien Georges [aut, cre],
Robin Engler [aut],
Frank Breiner [aut]

Maintainer Damien Georges <damien.georges2@gmail.com>

Contact Wilfried Thuiller <wilfried.thuiller@ujf-grenoble.fr>, Damien Georges <damien.georges2@gmail.com>

BugReports <https://r-forge.r-project.org/R/?group_id=302>

Description Functions for species distribution modeling, calibration and evaluation, ensemble of models.

Depends R (>= 3.2.1), stats, utils, sp, raster, parallel, reshape, ggplot2

Imports abind, rasterVis, pROC, nnet, gbm, mda, randomForest, rpart, MASS, methods, PresenceAbsence, dismo, maxent, earth

Suggests ade4, foreign, Hmisc, mgcv, gam, testthat, ecospat, caret

License GPL-2

RoxygenNote 5.0.1

Repository CRAN

Repository/R-Forge/Project biomod

Repository/R-Forge/Revision 716

Repository/R-Forge/DateTimeStamp 2016-03-01 10:01:06

Date/Publication 2016-03-01 18:40:06

NeedsCompilation no

R topics documented:

`biomod2-package` *Species Distribution Modeling within an Ensemble Forecasting framework*

Description

biomod2 is the updated object-oriented version of **BIOMOD** package.

biomod2 offers the possibility to run 10 state-of-the-art modeling techniques to describe and model the relationships between a given species and its environment. It is an attempt to define the ecological niche of a particular species using environmental variables (temperature, precipitation, ...) with the potential use of making, for instance, future projections under climate and land use change scenarios. Although it has been mostly developed for ecologists that aim to predict species distribution, **biomod2** can also be used to model any binomial data (for instance, gene, markers, ecosystem...) in function of any explanatory variables.

Details

Package:	biomod2
Type:	Package
Version:	1.0
Date:	2012-07-01
License:	GPL-2

Author(s)

Wilfried Thuiller and Damien Georges, with participation of Robin Engler <wilfried.thuiller@ujf-grenoble.fr>

`BinaryTransformation` *Convert species' probability of occurrence into binary presence-absence data using a predefined threshold*

Description

Function that converts an object containing probability values into a binary presence-absence object according to a pre-defined threshold(s).

Usage

```
BinaryTransformation(data, threshold)

## S4 method for signature 'data.frame'
BinaryTransformation(data, threshold)

## S4 method for signature 'matrix'
BinaryTransformation(data, threshold)

## S4 method for signature 'numeric'
BinaryTransformation(data, threshold)

## S4 method for signature 'array'
BinaryTransformation(data, threshold)

## S4 method for signature 'RasterLayer'
BinaryTransformation(data, threshold)

## S4 method for signature 'RasterStack'
BinaryTransformation(data, threshold)

## S4 method for signature 'RasterBrick'
BinaryTransformation(data, threshold)
```

Arguments

`data`	numeric vector, a `matrix`, a `data.frame`, a `RasterLayer` or a `RasterStack` containing the data to be converted
`threshold`	numeric value or a vector containing the threshold to be used for converting data.

Details

If data is a vector or a raster object, then the threshold should be a numeric value. If data is matrix,dataframe or rasterStack, then the threshold should have, in theory, as many values as the number of columns or layers to transform. In the particular case that the data to convert is a `matrix`/`data.frame` with several columns or a `RasterStack` with several layers and the threshold is a single numeric value, the same threshold will be applied to all columns (resp. layers).

Value

An object of the same class than `data` with binary (0 or 1) values, usually presence-absence.

Author(s)

Wilfried Thuiller, Damien Georges

Examples

```
xx <- rnorm(50,10)
yy <- BinaryTransformation(xx, 10)

cbind(xx,yy)
```

BIOMOD.EnsembleModeling.out-class
BIOMOD_EnsembleModeling() outputs objects class

Description

EnsembleModeling objects are created, used and returned by BIOMOD functions. It's contains information relative to an **biomod2** ensemble modeling procedure.

- output of: BIOMOD_EnsembleModeling
- input of: BIOMOD_EnsembleForecasting

Slots

`sp.name`: "character", species name

`expl.var.names`: "character", explanatory variables names

`models.out.obj`: "BIOMOD.stored.models.out", object which contains information on individuals models that have been combined

`eval.metric`: "character", evaluation metrics choosed for models selection

`eval.metric.quality.threshold`: "numeric", thresholds defined for models selection

`em.computed`: "character", ensemble models built names

`em.by`: "character", way models are combined

`em.models`: "ANY", list of built biomod2.ensemble.models objects

`modeling.id`: "character", the id of the whole modelling process

`link`: "character", the path to correspunding hard drive saved object

Author(s)

Damien Georges

See Also

BIOMOD_Projection, BIOMOD_Modeling, BIOMOD_EnsembleModeling, BIOMOD_EnsembleForecasting

Examples

```
showClass("BIOMOD.EnsembleModeling.out")
```

BIOMOD.EnsembleModeling.out-method

BIOMOD.EnsembleModeling.out getters

Description

Functions to get attributs of BIOMOD_EnsembleModeling outputs

Usage

```
get_kept_models(obj, ...)
get_needed_models(obj, ...)
```

Arguments

obj	"BIOMOD.EnsembleModeling.out" object
...	extra arguments (see details)

Details

... available values :

- **get_evaluations**
  - as.data.frame:"logical", (FALSE by default) if TRUE, a standardized data.frame will be produced else a list is returned
- **get_kept_models**
  - model"character" or "numeric" refering to model names (get_built_models()) or model id

Value

1. **get_built_models:** a character vector indicating set of ensemble-modeling algorithms ran
2. **get_built_models:** a character vector indicating the names of ensemble models computed
3. **get_evaluations:** an array, a data.frame or a list containing ensemble models evaluation scores
4. **get_kept_models:** an character vector indicating names of selected models for ensemble-models building
5. **get_needed_models:** an character vector indicating names of all needed models required to build all ensemble-models

Note

get_built_models & get_evaluations are also available for "BIOMOD.EnsembleModeling.out" object.

Author(s)

Wilfried Thuiller, Damien Georges

See Also

help

`BIOMOD.formated.data-class`

BIOMOD_FormatingData() outputs objects class

Description

BIOMOD.formated.data objects are created, used and returned by BIOMOD functions. It's contains the minimal set of data **biomod2** needs to work. Input data given to `BIOMOD_FormatingData` are rearanged to fit with `BIOMOD_Modeling` input format. All data are stored into matrix (even environemental raster) explaining why some objects appears to be quite heavy.

If you ask for pseudo absences selection in `BIOMOD_FormatingData`, you will get a `BIOMOD.formated.data.PA`, else you will get a `BIOMOD.formated.data.PA` object.

- output of: `BIOMOD_FormatingData`
- input of: `BIOMOD_Modeling`

Slots

BIOMOD.formated.data

`sp.name`: "character", species name

`coord`: "data.frame", species points XY coordinates

`data.species`: "numeric", species presences, absences and no information kept points

`data.env.var`: "data.frame", explanatory variables associated to species points

`has.data.eval`: "logical", was specific models evaluation dataset given ?

`eval.coord`: "data.frame", species models evaluation points XY coordinates

`eval.data.species`: "numeric", species presences, absences and no information models evaluation kept points

`eval.data.env.var`: "data.frame", explanatory variables associated to models evaluation species points

BIOMOD.formated.data

`sp.name`: "character", species name

`coord`: "data.frame", species points XY coordinates

`data.species`: "numeric", species presences, absences and no information kept points

`data.env.var`: "data.frame", explanatory variables associated to species points

`has.data.eval`: "logical", was specific models evaluation dataset given ?

`eval.coord`: "data.frame", species models evaluation points XY coordinates

`eval.data.species`: "numeric", species presences, absences and no information models evaluation kept points

`eval.data.env.var`: "data.frame", explanatory variables associated to models evaluation species points

`PA`: "data.frame", each column contains 1 or 0 indicating for each species points if it was select or not in associated PA dataset

Author(s)

Damien Georges

Examples

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

`BIOMOD.Model.Options-class`

BIOMOD_ModelingOptions() outputs objects class

Description

BIOMOD.Model.Options objects are created, used and returned by BIOMOD functions. These objects will contains for each model support within **biomod2**, a set of options that users can change. Please refer to `BIOMOD_ModelingOptions` for futher details.

- output of: `BIOMOD_ModelingOptions`
- input of: `BIOMOD_Modeling`

Slots

Please refer to `BIOMOD_ModelingOptions` for each model arguments supported.

`GLM`: "list", list of GLM supported options

`GBM`: "list", list of GBM supported options

`GAM`: "list", list of GAM supported options

`CTA`: "list", list of CTA supported options

`ANN`: "list", list of ANN supported options

`SRE`: "list", list of SRE supported options

`FDA`: "list", list of FDA supported options

`MARS`: "list", list of MARS supported options

`RF`: "list", list of RF supported options

`MAXENT.Phillips`: "list", list of MAXENT.Phillips supported options

`MAXENT.Tsuruoka`: "list", list of MAXENT.Tsuruoka supported options

Author(s)

Damien Georges

See Also

BIOMOD_ModelingOptions.

Examples

```
showClass("BIOMOD.Model.Options")
```

BIOMOD.models.out-class

BIOMOD_modelling() outputs objects class

Description

The BIOMOD.models.out objects are created, used and returned by BIOMOD functions.

- output of: BIOMOD_Modeling
- input of: BIOMOD_EnsembleModeling, BIOMOD_Projection

Slots

BIOMOD.models.out objects

`modeling.id`: "character", id of modeling procedure

`sp.name`: "character", species name

`expl.var.names`: "character", explanatory variables names

`has.evaluation.data`: "logical", is some data are reserved for evaluating models?

`models.computed`: "character", names of computed models

`models.failed`: "character", names of failed models

`models.evaluation`: "BIOMOD.stored.array", evaluations of each model computed according to selected evaluation methods

`variables.importances`: "BIOMOD.stored.array", models variable importances

`models.prediction`: "BIOMOD.stored.array", predictions of each models on their own calibration + validation dataset

`models.prediction.eval`: "BIOMOD.stored.array", predictions of each models on evaluation dataset (if defined)

`formated.input.data`: "BIOMOD.stored.formated.data", input data

`calib.lines`: "BIOMOD.stored.array", calibrations and evaluation lines selected for each run

`models.options`: "BIOMOD.stored.models.options", options used to build each model

`rescal.all.models`: "logical", is all models scaling done?

`link`: 'character', the path to object hard drive copy

Author(s)

Damien Georges

Examples

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

`BIOMOD.models.out-methods`
BIOMOD.models.out getters

Description

Functions to get attributs of `BIOMOD_Modeling` outputs

Usage

```
get_predictions(obj, ...)
get_formal_data(obj, ...)
get_evaluations(obj, ...)
get_calib_lines(obj, ...)
get_variables_importance(obj, ...)
get_options(obj, ...)
get_built_models(obj, ...)
```

Arguments

`obj`	"`BIOMOD.models.out`" object
`...`	extra arguments (see details)

Details

... available values :

- **get_evaluations**
  - `as.data.frame`:"logical", (FALSE by default) if TRUE, a standardized `data.frame` will be produced else an 4 dimentions `array` is returned
- **get_calib_lines**
- **get_predictions (for BIOMOD_Modeling() outputs only)**
  - `as.data.frame`:logical(default FALSE). If TRUE, models predictions will be returned as `data.frame` rather than `array`
  - `evaluation`:logical (default FALSE). If TRUE, model prediction over evaluation data will be returned
- **get_formal_data**

- subinfo:character. Flag defining a specific information to extract from "BIOMOD.formated.data" object. Supported values are:
  - NULL: (default) the whole "BIOMOD.formated.data" object is returned
  - 'MinMax': All explanatory variables ranges returned
  - 'resp.var': Response variables vector returned
  - 'eval.resp.var': Evaluation response variables vector returned
  - 'expl.var': Explanatory variables data.frame returned
  - 'eval.expl.var': Evaluation explanatory variables data.frame returned
  - 'expl.var.names': Explanatory variables names returned

Value

1. **get_predictions:** an array (or a data.frame) containing models predictions over calibrating and testing data (those used for evaluate models)
2. **get_calib_lines:** an array (or a data.frame) having the same dimention than the outpurt of **get_predictions()** of logical values. All lines containing TRUE have been used to calibrate the model
3. **get_evaluations:** an array, a data.frame or a list containing models evaluation scores
4. **get_variables_importance:** an array containing models variables importances
5. **get_options:** a "BIOMOD.Model.Options" reporting options used to build individual models
6. **get_formal_data:** a "BIOMOD.formated.data" object containing data used for models building and evavaluation, or a part of this object
7. **get_built_models:** a character vector giving the names of models succefully computed

Author(s)

Wilfried Thuiller, Damien Georges

See Also

help

BIOMOD.models.out-RemoveProperly

remove properly BIOMOD_Modeling outputs

Description

Functions to free properly a BIOMOD_Modeling outputs

Usage

```
RemoveProperly(obj, obj.name=deparse(substitute(obj)), ...)
```

Arguments

obj	"BIOMOD.models.out" object
obj.name	the name of object in current environment, automatically filled
...	extra arguments (not implemented yet)

Details

This function will remove all objects created during a biomod2 modeling run. It will free both objects saved in memory and objects saved on hard drive.

Author(s)

Wilfried Thuiller, Damien Georges

See Also

help

Examples

```
## Not run:
# species occurrences
DataSpecies <- read.csv(system.file("external/species/mammals_table.csv",
                                    package="biomod2"), row.names = 1)
head(DataSpecies)

# the name of studied species
myRespName <- 'VulpesVulpes'

# the presence/absences data for our species
myResp <- as.numeric(DataSpecies[,myRespName])

# the XY coordinates of species data
myRespXY <- DataSpecies[,c("X_WGS84","Y_WGS84")]


# Environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
myExpl = stack( system.file( "external/bioclim/current/bio3.grd",
                     package="biomod2"),
                system.file( "external/bioclim/current/bio4.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio7.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio11.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio12.grd",
                             package="biomod2"))

# 1. Formatting Data
myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
                                     expl.var = myExpl,
```

```
                                        resp.xy = myRespXY,
                                        resp.name = myRespName)

  # 2. Defining Models Options using default options.
  myBiomodOption <- BIOMOD_ModelingOptions()

  # 3. Doing Modelisation

  myBiomodModelOut <- BIOMOD_Modeling( myBiomodData,
                                        models = c('SRE'),
                                        models.options = myBiomodOption,
                                        NbRunEval=1,
                                        DataSplit=80,
                                        Prevalence=0.5,
                                        VarImport=0,
                                        models.eval.meth = c('TSS','ROC'),
                                        do.full.models=FALSE,
                                        modeling.id="test2")

  # files have been created on hard drive
  list.files(myRespName,all.files=TRUE,recursive=TRUE)

  # remove properly the modeling objects and all the file saved on hard drive
  RemoveProperly(myBiomodModelOut)

  # check files had been removed
  list.files(myRespName,all.files=TRUE,recursive=TRUE)

## End(Not run)
```

`BIOMOD.projection.out-class`

BIOMOD_Projection() outputs objects class

Description

BIOMOD.projection.out object is created, used and returned by **biomod2** functions. It contains information relative to projections of `BIOMOD_Modeling` outputs over a defined area. This object may be reused latter by `BIOMOD_EnsembleForecasting` function.

- output of: `BIOMOD_Projection`
- input of: `BIOMOD_EnsembleForecasting`

Slots

`proj.names`**:** "character", projection name (name of folder where projections are stored)

`sp.name`**:** "character", species name

`expl.var.names`**:** "character", explanatory variables names

`models.projected`: "character", models projected names

`scaled.models`: "logical", was projected models scaled?

`modeling.object`: "BIOMOD.stored.data", associated modeling objects (link to)

`modeling.object.id`: "character",id of associated modeling objects

`type`: "character", type of projection done (`array`, `data.frame`, `RasterStack`, `RasterLayer` ...)

`proj`: "BIOMOD.stored.data", object containing projected values

`xy.coord`: "matrix", coordinates of projected points

Author(s)

Damien Georges

See Also

`BIOMOD_Projection`, `BIOMOD_Modeling`, `BIOMOD_EnsembleModeling`, `BIOMOD_EnsembleForecasting`

Examples

```
showClass("BIOMOD.projection.out")
```

BIOMOD.projection.out-method

BIOMOD.projection.out getters

Description

Functions to get attributs of `BIOMOD_Projection` outputs

Usage

```
get_projected_models(obj, ...)
free(obj, ...)
```

Arguments

`obj`	"`BIOMOD.projection.out`" object
`...`	extra arguments (see details)

Details

. . . available values :

- get_predictions
  - `as.data.frame`:"logical", return projections into a `data.frame` (TRUE) or an `array` (FALSE, default)
  - `full.name`:NULL or "character", full names of models you want to extract projections
  - `model`:NULL or "character", *type name* of models you want to extract projections
  - `run.eval`:NULL or "character", *run name* of models you want to extract projections
  - `data.set`:NULL or "character", *dataset name* of models you want to extract projections

Value

1. **get_predictions:** an `array` or a `data.frame` containing models projections
2. **get_projected_models:** `character` containing names of models that have been projected
3. **free:** remove projection value from R memory and only keep the path to correspunding file on hard drive.

Note

`get_predictions` is also available for `"BIOMOD.projection.out"` object.

Author(s)

Wilfried Thuiller, Damien Georges

See Also

`help`

BIOMOD.stored.objects-class

BIOMOD.stored.xxx objects class

Description

All BIOMOD.stored.xxx objects are there to make some RAM savings by loading some objects only when they are needed. We just keep a link to the place were those objects are stored when we don't need them anymore. They are kind of pointers to objects created within **biomod2** and having a copy on hard drive. It can be consider just as a code trick.

Slots

Each BIOMOD.stored.xxx objects has 3 slot:

`inMemory`: "logical", is the object already loaded on RAM?

`link`: "character", path to the hard drive copy of the object

`val`: NULL or the object if loaded.. The object class is different for each BIOMOD.stored.xxx objects

Author(s)

Damien Georges

Examples

```
showClass("BIOMOD.stored.files")
```

BIOMOD.stored.objects-methods

BIOMOD.stored.objects functions

Description

Functions to play with attributs of `BIOMOD.stored.objects` objects

Usage

```
load_stored_object(obj, ...)
```

Arguments

`obj`	`BIOMOD.stored.objects` object
`...`	extra arguments (see details)

Value

returns the value stored within biomod2 storing object.

Author(s)

Damien Georges

`biomod2.deprecated.functions` *biomod2 deprecated functions*

Description

Set of functions that will be removed in a while. They have all an updated equivalent.

Usage

```
getEM_needed_models(obj,...)
getEMalgos(obj,...)
getEMbuiltModels(obj,...)
getEMeval(obj,...)
getEMkeptModels(obj,...)
getFormalModel(obj,...)
getModelsBuiltModels(obj,...)
getModelsEvaluations(obj,...)
getModelsInputData(obj,...)
getModelsOptions(obj,...)
getModelsPrediction(obj,...)
getModelsPredictionEval(obj,...)
getModelsVarImport(obj,...)
getProjection(obj,...)
getScalingModel(obj,...)
```

Arguments

`obj`	specific `biomod2` object
`...`	extra arguments

Author(s)

Wilfried Thuiller, Damien Georges

See Also

`help`

biomod2.inner-method *biomod2 internal functions*

Description

biomod2 internal functions.

Details

All this functions are low level functions. Interested people can go into source code for futher details.

Author(s)

Wilfried Thuiller, Damien Georges

See Also

help

biomod2_model-class *biomod2 models objects class and functions*

Description

This objects represent all biomod2 models as it was a 'classical' R model. It is produce during BIOMOD_Modeling step. It can be printed. predict function is also implemented for this object and supports data.frame and {RasterStack} inputs. You cen rise formal models with get_formal_model and the associated scaling GLM (if exists) with get_scaling_model. All this object have also their own predict function.

Slots

model_name: "character", model name

model_class: "character", type of model (e.g. 'GLM', 'RF', 'MAXENT.Phillips','EMmean')

model_options: "list", list of options used for model building

model: "ANY", the formal R model (if exists)

scaling_model: "ANY", the associated glm to scale prediction on 0-1 (if exists)

resp_name: "character", response variable name

expl_var_names: "character", explanatory variables names

model_evaluation: "matrix", model evaluation scores (if exist)

model_variables_importance: "matrix", model importance of variables (if exists)

model_output_dir: "character", path to model output directory

Author(s)

Damien Georges

Examples

```
showClass("ANN_biomod2_model")
```

BIOMOD_ConvertOldRun *Convert objects and outputs from BIOMOD.xx into biomod2.xx objects and outputs*

Description

This function converts workspace, modelling outputs, results and objects created with version xx of **BIOMOD** into **biomod2** objects and re-organized the directories to be used with **biomod2**

Usage

```
BIOMOD_ConvertOldRun(savedObj,
                     path = NULL)
```

Arguments

savedObj	a BIOMOD.1.xx workspace image. It's a .Rdata file named 'Biomod_run.RData' for plurispecific run or Your_Species_Name.RData if you have done monospecific modelling
path	Optional path to your 'savedObj' if you don't have give the full path to the object

Details

This function is useful to convert former **BIOMOD** runs into the new **biomod2** object structure. This is mostly interesting in the case users want to relaunched some projections or analyses within the **biomod2** new structure. Returned 'BIOMOD.models.out' objects can be then used as classic object for making projections for instance (BIOMOD_Projection). Be aware that because **biomod2** has strongly changed between the first and second version, some new additional functions and information could not be used with converted objects (i.e. Calibration Lines access, Maxent run, SRE projections...).

Value

A list of 'BIOMOD.models.out' (one per species modeled) containing information of your old run. Specific directories are also created on your hard drive (see BIOMOD_Modeling)

Author(s)

Damien Georges

See Also

BIOMOD_Modeling

Examples

```
## Not Done Yet ##
##
##
```

BIOMOD_cv *Custom models cross-validation procedure*

Description

This function creates a DataSplitTable which could be used to evaluate models in Biomod with repeated k-fold cross-validation (cv) or stratified cv instead of repeated split sample runs

Usage

```
BIOMOD_cv(data, k = 5, repetition = 5, do.full.models = TRUE,
  stratified.cv = FALSE, stratify = "both", balance = "pres")
```

Arguments

data	BIOMOD.formated.data object returned by BIOMOD_FormatingData
k	number of bins/partitions for k-fold cv
repetition	number of repetitions of k-fold cv (1 if stratified.cv=TRUE)
do.full.models	if true, models calibrated and evaluated with the whole dataset are done
stratified.cv	logical. run a stratified cv
stratify	stratification method of the cv. Could be "x", "y", "both" (default), "block" or the name of a predictor for environmental stratified cv.
balance	make balanced particions for "presences" (default) or "absences" (resp. pseudo-absences or background).

Details

Stratified cv could be used to test for model overfitting and for assessing transferability in geographic and environmental space. If balance = "presences" presences are divided (balanced) equally over the particions (e.g. Fig. 1b in Muscarelly et al. 2014). Pseudo-Absences will however be unbalanced over the particions 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 for the particions (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 particians. Be careful: If the presences are clumped on an edge of the study area it is possible that all presences are in one bin.

Value

DataSplitTable matrix with k*repetition (+ 1 for Full models if do.full.models = TRUE) columns for BIOMOD_Modeling function. Stratification "x" and "y" was described in Wenger and Olden 2012. While Stratification "y" uses k partitions along the y-gradient, "x" does the same for the x-gradient and "both" combines them. Stratification "block" was described in Muscarella et al. 2014. For bins of equal number are partitioned (bottom-left, bottom-right, top-left and top-right).

Author(s)

Frank Breiner

References

Muscarella, R., Galante, P.J., Soley-Guardia, M., Boria, R.A., Kass, J.M., Uriarte, M. & Anderson, R.P. (2014). ENMeval: An R package for conducting spatially independent evaluations and estimating optimal model complexity for Maxent ecological niche models. Methods in Ecology and Evolution, 5, 1198<80><93>1205. Wenger, S.J. & Olden, J.D. (2012). Assessing transferability of ecological models: an underappreciated aspect of statistical validation. Methods in Ecology and Evolution, 3, 260<80><93>267.

Examples

```
## Not run:
# species occurrences
DataSpecies <- read.csv(system.file("external/species/mammals_table.csv",
                                    package="biomod2"))
head(DataSpecies)

the name of studied species
myRespName <- 'GuloGulo'

# the presence/absences data for our species
myResp <- as.numeric(DataSpecies[,myRespName])

# the XY coordinates of species data
myRespXY <- DataSpecies[,c("X_WGS84","Y_WGS84")]


# Environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
myExpl = stack( system.file( "external/bioclim/current/bio3.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio4.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio7.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio11.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio12.grd",
                             package="biomod2"))

# 1. Formatting Data
```

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

# 2. Defining Models Options using default options.
myBiomodOption <- BIOMOD_ModelingOptions()


# 3. Creating DataSplitTable

DataSplitTable <- BIOMOD_cv(myBiomodData, k=5, rep=2, do.full.models=F)
DataSplitTable.y <- BIOMOD_cv(myBiomodData,stratified.cv=T, stratify="y", k=2)
colnames(DataSplitTable.y)[1:2] <- c("RUN11","RUN12")
DataSplitTable <- cbind(DataSplitTable,DataSplitTable.y)
head(DataSplitTable)

# 4. Doing Modelisation

myBiomodModelOut <- BIOMOD_Modeling( myBiomodData,
                                     models = c('RF'),
                                     models.options = myBiomodOption,
                                     DataSplitTable = DataSplitTable,
                                     VarImport=0,
                                     models.eval.meth = c('ROC'),
                                     do.full.models=FALSE,
                                     modeling.id="test")

## get cv evaluations
eval <- get_evaluations(myBiomodModelOut,as.data.frame=T)

eval$strat <- NA
eval$strat[grepl("13",eval$Model.name)] <- "Full"
eval$strat[!(grepl("11",eval$Model.name)|
             grepl("12",eval$Model.name)|
             grepl("13",eval$Model.name))] <- "Random"
eval$strat[grepl("11",eval$Model.name)|grepl("12",eval$Model.name)] <- "Strat"

boxplot(eval$Testing.data~ eval$strat, ylab="ROC AUC")

## End(Not run)
```

`BIOMOD_EnsembleForecasting` *Ensemble projections of species over space and time*

Description

This function use projections of 'individual models' and ensemble models from `BIOMOD_EnsembleModeling` to build an ensemble of species' projections over space and time.

Usage

```
BIOMOD_EnsembleForecasting( EM.output,
                            projection.output = NULL,
                            new.env = NULL,
                            xy.new.env = NULL,
                            selected.models = 'all',
                            proj.name = NULL,
                            binary.meth = NULL,
                            filtered.meth = NULL,
                            compress = TRUE,
                            ...)
```

Arguments

EM.output	a "BIOMOD.EnsembleModeling.out" returned by BIOMOD_EnsembleModeling
projection.output	a "BIOMOD.projection.out" returned by BIOMOD_Projection
new.env	a RasterStack or a data.frame object containing explanatory data for the studied area. Needed only if projection.output is NULL. Prefer to use "BIOMOD.projection.out" if you have ever done the calculations.
xy.new.env	the matching coordinates of new.env if new.env is defined and if it is a data.frame
selected.models	if not 'all', a character vector containing a subset of ensemble-models you want make projacion
proj.name	the projection name (results will be saved within proj_proj.name directory). Only needed if projection.output is NULL
binary.meth	vector specifying the names of evaluation metrics and associated thresholds to transform the probabilities of presence into presence and absence (binary transformation).
filtered.meth	vector specifying the names of evaluation metrics and associated thresholds to transform into 0 the probabilities of presence lower than the thresholds.
compress	boolean or character, the compression format of objects stored on your hard drive. May be one of 'TRUE', 'FALSE', 'xz' or 'gzip' (see save)
...	further arguments (see details)

Details

This function requires to have successfully run **biomod2** modeling, ensemble-modeling and projection steps. Ensemble projections will be created in respect to projection.output projections, which are combined following EM.output ensemble-modeling rules.

The 'total.consensus' projection is basically the mean of all projections (for having only one output).

... may be :

- on_0_1000:logical, if TRUE (default), 0 - 1 probabilities are converted into a 0 - 1000 integer scale. This implies a lot of memory saving. User that want to comeback on a 0 - 1 scale latter will just have to divide all projections by 1000

Value

Nothing returned but specific 'projection files' are saved on the hard drive projection folder. This files are either an `array` or a `RasterStack` depending the original projections data type. Load these created files to plot and work with them.

Author(s)

Wilfried Thuiller, Damien Georges, Robin Engler

See Also

BIOMOD_EnsembleModeling, BIOMOD_Projection

Examples

```
# 0. Load data & Selecting Data
# species occurrences
DataSpecies <- read.csv(system.file("external/species/mammals_table.csv",
                                    package="biomod2"), row.names = 1)
head(DataSpecies)

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

# the presence/absences data for our species
myResp <- as.numeric(DataSpecies[,myRespName])

# the XY coordinates of species data
myRespXY <- DataSpecies[,c("X_WGS84","Y_WGS84")]


# Environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
myExpl = stack( system.file( "external/bioclim/current/bio3.grd",
                     package="biomod2"),
                system.file( "external/bioclim/current/bio4.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio7.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio11.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio12.grd",
                             package="biomod2"))

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

# 2. Defining Models Options using default options.
myBiomodOption <- BIOMOD_ModelingOptions()
```

```
# 3. Running the models
myBiomodModelOut <- BIOMOD_Modeling( myBiomodData,
                                     models = c('RF'),
                                     models.options = myBiomodOption,
                                     NbRunEval=2,
                                     DataSplit=60,
                                     Yweights=NULL,
                                     VarImport=0,
                                     models.eval.meth = c('TSS'),
                                     SaveObj = TRUE,
                                     rescal.all.models = FALSE,
                                     do.full.models = FALSE)

# 4. Creating the ensemble models
myBiomodEM <- BIOMOD_EnsembleModeling(
                  modeling.output = myBiomodModelOut,
                  chosen.models = grep('_RF', get_built_models(myBiomodModelOut),
                                       value=TRUE),
                  em.by = 'algo',
                  eval.metric = c('TSS'),
                  eval.metric.quality.threshold = c(0.7),
                  prob.mean = TRUE,
                  prob.cv = FALSE,
                  prob.ci = FALSE,
                  prob.ci.alpha = 0.05,
                  prob.median = FALSE,
                  committee.averaging = FALSE,
                  prob.mean.weight = FALSE,
                  prob.mean.weight.decay = 'proportional' )

# 5. Individual models projections on current environmental conditions
myBiomodProjection <- BIOMOD_Projection(
                        modeling.output = myBiomodModelOut,
                        new.env = myExpl,
                        proj.name = 'current',
                        selected.models = grep('_RF', get_built_models(
                                                 myBiomodModelOut), value=TRUE),
                        compress = FALSE,
                        build.clamping.mask = FALSE)


# 4. Creating the ensemble projections
BIOMOD_EnsembleForecasting( projection.output = myBiomodProjection,
                            EM.output = myBiomodEM)
```

BIOMOD_EnsembleModeling *Create and evaluate an ensemble set of models and predictions*

Description

BIOMOD_EnsembleModeling combines models and make ensemble predictions built with BIOMOD_Modeling. The ensemble predictions can also be evaluated against the original data given to BIOMOD_Modeling. Biomod2 proposes a range of options to build ensemble models and predictions and to assess the modeling uncertainty. The created ensemble models can then be used to project distributions over space and time as classical **biomod2** models.

Usage

```
BIOMOD_EnsembleModeling( modeling.output,
                         chosen.models = 'all',
                         em.by = 'PA_dataset+repet',
                         eval.metric = 'all',
                         eval.metric.quality.threshold = NULL,
                         models.eval.meth = c('KAPPA','TSS','ROC'),
                         prob.mean = TRUE,
                         prob.cv = FALSE,
                         prob.ci = FALSE,
                         prob.ci.alpha = 0.05,
                         prob.median = FALSE,
                         committee.averaging = FALSE,
                         prob.mean.weight = FALSE,
                         prob.mean.weight.decay = 'proportional',
                         VarImport = 0)
```

Arguments

`modeling.output`
: a "BIOMOD.models.out" returned by BIOMOD_Modeling

`chosen.models`
: a character vector (either 'all' or a sub-selection of model names) that defines the models kept for building the ensemble models (might be useful for removing some non-preferred models)

`em.by`
: Character. Flag defining the way the models will be combined to build the ensemble models. Available values are `'PA_dataset+repet'` (default), `'PA_dataset+algo'`, `'PA_dataset'`, `'algo'` and `'all'`

`eval.metric`
: vector of names of evaluation metric used to build ensemble models. It is involved for formal models exclusion if `eval.metric.quality.threshold` is defined and/or for building ensemble models that are dependent of formal models evaluation scores (e.g. weighted mean and comitee averaging). If 'all', the same evaluation metrics than those of `modeling.output` will be automatically selected

`eval.metric.quality.threshold`
: If not `NULL`, the minimum scores below which models will be excluded of the ensemble-models building.

`models.eval.meth`
: the evaluation methods used to evaluate ensemble models (see "BIOMOD_Modeling" **models.eval.meth** section for more detailed informations)

prob.mean	Logical. Estimate the mean probabilities across predictions
prob.cv	Logical. Estimate the coefficient of variation across predictions
prob.ci	Logical . Estimate the confidence interval around the prob.mean
prob.ci.alpha	Numeric. Significance level for estimating the confidence interval. Default = 0.05
prob.median	Logical. Estimate the mediane of probabilities
committee.averaging	Logical. Estimate the committee averaging across predictions
prob.mean.weight	Logical. Estimate the weighted sum of probabilities
prob.mean.weight.decay	Define the relative importance of the weights. A high value will strongly discriminate the 'good' models from the 'bad' ones (see the details section). If the value of this parameter is set to 'proportional' (default), then the attributed weights are proportional to the evaluation scores given by 'weight.method'(eval.metric)
VarImport	Number of permutation to estimate variable importance

Details

1. **Models sub-selection** (chosen.models)

   Useful to exclude some models that have been selected in the previous steps (modeling.output). This vector of model names can be access applying get_built_models to your modeling.output data. It makes easier the selection of models. The default value (i.e. 'all') will kept all available models.

2. **Models assembly rules** (em.by)

   Please refer to EnsembleModelingAssembly vignette that is dedicated to this parameter.

   5 different ways to combine models can be considered. You can make ensemble models considering :

   - Dataset used for models building (Pseudo Absences dataset and repetitions done): 'PA_dataset+repet'
   - Dataset used and statistical models : 'PA_dataset+algo'
   - Pseudo-absences selection dataset : 'PA_dataset'
   - Statistical models : 'algo'
   - A total consensus model : 'all'

   The value chosen for this parameter will control the number of ensemble models built. If no evaluation data was given the at BIOMOD_FormatingData step, some ensemble models evaluation may be a bit unfair because the data that will be used for evaluating ensemble models could differ from those used for evaluate BIOMOD_Modeling models (in particular, some data used for 'basal models' calibration can be re-used for ensemble models evaluation). You have to keep it in mind ! (EnsembleModelingAssembly vignette for extra details)

3. **Evaluation metrics**

   - eval.metric

     The selected metrics here are necessary the ones chosen at the BIOMOD_Modeling step. If you select several, ensembles will be built according to each of them. The chosen metrics will be used at different stages in this function :

(a) to remove 'bad models' (having a score lower than `eval.metric.quality.threshold` (see bellow))

(b) to make the binary transformation needed for committee averaging computation

(c) to weight the models in the probability weighted mean model

(d) to test (and/or evaluate) your ensemble-models forecasting ability (at this step, each ensemble-model (ensemble will be evaluated according to each evaluation metric)

- `eval.metric.quality.threshold`
  You have to give as many threshold as `eval.metric` you have selected. If you have selected several evaluation metrics , take care to ensure that the order of thresholds matches the order of `eval.metric`. All models having a score lower than these quality thresholds will not be kept for building ensemble-models.

4. **Ensemble-models algorithms**

(a) **Mean of probabilities** (`prob.mean`)
This ensemble-model corresponds to the mean probabilities over the selected models.

(b) **Coefficient of variation of Probabilities** (`prob.cv`)
This ensemble-model corresponds to the coefficient of variation (i.e. sd / mean) of the probabilities over the selected models. This model is not scaled. It will be evaluated like all other ensemble-models although this interpretation is 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 models). The lower is the score, the better are the models. CV is a nice complement to the mean probability.

(c) **Confidence interval** (`prob.ci` **&** `prob.ci.alpha`) This is the confidence interval around the mean probability (see above). This is also a nice complement to the mean probability. Two ensemble-models will be built if `prob.ci` is `TRUE` :

- The upper one (there is less than a 100*`prob.ci.alpha`/2 % of chance to get probabilities upper than the given ones)
- The lower one (there is less than a 100*`prob.ci.alpha`/2 % of chance to get probabilities lower the than given ones)

These intervals are calculated with the following function :

$$I_c = [\bar{x} - \frac{t_\alpha sd}{\sqrt{n}}; \bar{x} + \frac{t_\alpha sd}{\sqrt{n}}]$$

(d) **Median of probabilities** (`prob.median`)
This ensemble-model corresponds to the median probability over the selected models. The median is less sensitive to outliers than the mean. In practical terms, calculating the median requires more time and memory than the mean (or even weighting mean) as it asks to load all predictions to then extract the median. It may need to be considered in case of large dataset.

(e) **Models committee averaging** (`committee.averaging`)
To do this model, the probabilities from the selected models are first transformed into binary data according to the thresholds defined at `BIOMOD_Modeling` step (maximizing evaluation metric score over 'testing dataset'). The committee averaging score is then the average of binary predictions. It is built on the analogy of a simple vote. Each model vote for the species being ether present or absent. For each site, the sum of 1 is then divided by the number of models. 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 and 1 respectively. When the prediction is around 0.5, it means that half the models predict 1 and the other half 0.

(f) **Weighted mean of probabilities (**`prob.mean.weight` **&** `prob.mean.weight.decay`**)**
This algorithm return the mean weighted (or more precisely this is the weighted sum) by the selected evaluation method scores (better a model is, more importance it has in the ensemble). The scores come from `BIOMOD_Modeling` step.
The `prob.mean.weight.decay` is the ratio between a weight and the following or prior one. The formula is : W = W(-1) * `prob.mean.weight.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 `prob.mean.weight.decay`, the smoother the differences between the weights enhancing a weak discrimination between models.
The value 'proportional' (default) is also possible for the `prob.mean.weight.decay`: the weights are awarded for each method proportionally to their evaluation scores. The advantage is that the discrimination is more fair than with the `prob.mean.weight.decay`. In the latter case, close scores can strongly diverge in the weights they are awarded, when the proportional method will consider them as being fairly similar in prediction quality and award them a similar weight. It is also possible to define a function as `prob.mean.weight.decay` argument. In this case the given function will be applied to models scores to transforme them into weights that will be used for wheigted mean ensemble model building. For instance if you specified `function(x){x^2}` as `prob.mean.weight.decay`, the squared of evaluation score of each model will be used to weight formal models predictions.

Value

A `"BIOMOD.EnsembleModeling.out"`. This object will be later given to `BIOMOD_EnsembleForecasting` if you want to make some projections of this ensemble-models.

You can access to evaluation scores with the `get_evaluations` function and to the built models names with the `get_built_models` function (see example).

Note

Models are now combined by repetition, other way to combine them (e.g. by Models, all together...) will be available soon

Author(s)

Damien Georges & Wilfried Thuiller with participation of Robin Engler

See Also

`BIOMOD_Modeling`, `BIOMOD_Projection`, `BIOMOD_EnsembleForecasting`

Examples

```
# species occurrences
```

```
DataSpecies <- read.csv(system.file("external/species/mammals_table.csv",
                                    package="biomod2"), row.names = 1)
head(DataSpecies)

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

# the presence/absences data for our species
myResp <- as.numeric(DataSpecies[,myRespName])

# the XY coordinates of species data
myRespXY <- DataSpecies[,c("X_WGS84","Y_WGS84")]


# Environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
myExpl = stack( system.file( "external/bioclim/current/bio3.grd",
                     package="biomod2"),
                system.file( "external/bioclim/current/bio4.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio7.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio11.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio12.grd",
                             package="biomod2"))

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

# 2. Defining Models Options using default options.
myBiomodOption <- BIOMOD_ModelingOptions()

# 3. Doing Modelisation

myBiomodModelOut <- BIOMOD_Modeling( myBiomodData,
                                       models = c('SRE','CTA','RF'),
                                       models.options = myBiomodOption,
                                       NbRunEval=1,
                                       DataSplit=80,
                                       Yweights=NULL,
                                       VarImport=3,
                                       models.eval.meth = c('TSS'),
                                       SaveObj = TRUE,
                                       rescal.all.models = FALSE,
                                       do.full.models = FALSE)

# 4. Doing Ensemble Modelling
myBiomodEM <- BIOMOD_EnsembleModeling( modeling.output = myBiomodModelOut,
                            chosen.models = 'all',
                            em.by = 'all',
```

```
                            eval.metric = c('TSS'),
                            eval.metric.quality.threshold = c(0.7),
                            models.eval.meth = c('TSS','ROC'),
                            prob.mean = TRUE,
                            prob.cv = FALSE,
                            prob.ci = FALSE,
                            prob.ci.alpha = 0.05,
                            prob.median = FALSE,
                            committee.averaging = FALSE,
                            prob.mean.weight = TRUE,
                            prob.mean.weight.decay = 'proportional' )

# print summary
myBiomodEM

# get evaluation scores
get_evaluations(myBiomodEM)
```

BIOMOD_FormatingData *Initialise the datasets for usage in* **biomod2**

Description

This function rearranges the user's input data to make sure they can be used within **biomod2**. The function allows to select pseudo-absences or background data in the case that true absences data are not available, or to add pseudo-asbence data to an existing set of absence (see details).

Usage

```
BIOMOD_FormatingData(resp.var,
                     expl.var,
                     resp.xy = NULL,
                     resp.name = NULL,
                     eval.resp.var = NULL,
                     eval.expl.var = NULL,
                     eval.resp.xy = NULL,
                     PA.nb.rep = 0,
                     PA.nb.absences = 1000,
                     PA.strategy = 'random',
                     PA.dist.min = 0,
                     PA.dist.max = NULL,
                     PA.sre.quant = 0.025,
                     PA.table = NULL,
                     na.rm = TRUE)
```

Arguments

resp.var	a vector, SpatialPointsDataFrame (or SpatialPoints if you work with 'only presences' data) containing species data (a single species) in binary format (ones for presences, zeros for true absences and NA for indeterminated) that will be **used to build the species distribution models**.
expl.var	a matrix, data.frame, SpatialPointsDataFrame or RasterStack containing your explanatory variables that will be **used to build your models**.
resp.xy	optional 2 columns matrix containing the X and Y coordinates of resp.var (only consider if resp.var is a vector) that will be **used to build your models**.
eval.resp.var	a vector, SpatialPointsDataFrame your species data (a single species) in binary format (ones for presences, zeros for true absences and NA for indeterminated) that will be **used to evaluate the models with independant data (or past data for instance)**.
eval.expl.var	a matrix, data.frame, SpatialPointsDataFrame or RasterStack containing your explanatory variables that will be **used to evaluate the models with independant data (or past data for instance)**.
eval.resp.xy	opional 2 columns matrix containing the X and Y coordinates of resp.var (only consider if resp.var is a vector) that will be **used to evaluate the modelswith independant data (or past data for instance)**.
resp.name	response variable name (character). The species name.
PA.nb.rep	number of required Pseudo Absences selection (if needed). 0 by Default.
PA.nb.absences	number of pseudo-absence selected for each repetition (when PA.nb.rep > 0) of the selection (true absences included)
PA.strategy	strategy for selecting the Pseudo Absences (must be 'random', 'sre', 'disk' or 'user.defined')
PA.dist.min	minimal distance to presences for 'disk' Pseudo Absences selection (in meters if the explanatory is a not projected raster (+proj=longlat) and in map units (typically also meters) when it is projected or when explanatory variables are stored within table)
PA.dist.max	maximal distance to presences for 'disk' Pseudo Absences selection(in meters if the explanatory is a not projected raster (+proj=longlat) and in map units (typically also meters) when it is projected or when explanatory variables are stored within table)
PA.sre.quant	quantile used for 'sre' Pseudo Absences selection
PA.table	a matrix (or a data.frame) having as many rows than resp.var values. Each column correspund to a Pseudo-absences selection. It contains TRUE or FALSE indicating which values of resp.var will be considered to build models. It must be used with 'user.defined' PA.strategy.
na.rm	locical, if TRUE, all points having one or several missing value for environmental data will be removed from analyse

Details

This function homogenises the initial data for making sure the modelling exercie will be completed with all the required data. It supports different kind of inputs.

IMPORTANT: When the explanatory data are given in `rasterLayer` or `rasterStack` objects, **biomod2** will be extract the variables onto the XY coordinates of the presence (and absence is any) vector. Be sure to give the XY coordinates ('resp.xy') in the same projection system than the raster objects. Same for the evaluation data in the case some sort of independant (or past) data are available ('eval.resp.xy'). When the explanatory variables are given in `SpatialPointsDataFrame`, the same requirements are asked than for the raster objects. The XY coordinates must be given to make sure biomod2 can extract the explanatory variables onto the presence (absence) data When the explanatory variables are stored in a data.frame, make sure there are in the same order than the response variable. **biomod2** will simply merge the datasets without considering the XY coordinates.

When both presence and absence data are available, and there is enough absences: set sQuotePA.nb.rep to 0. No pseudo-absence will be extracted.

When no true absences are given or when there are not numerous enough. It's advise to make several pseudo absences selections. That way the influence of the pseudo-absence selection could then be estimated later on. If the user do not want to run several repetition, make sure to select a relatively high number pseudo-absence. Make sure the number of pseudo-absence data is not higher than the maximum number of potential pseudo-absence (e.g. do not select 10,000 pseudo-absence when the rasterStack or data.frame do not contain more than 2000 pixels or rows).

1. **Response variable encoding**

   `BIOMOD_FormatingData` concerns a single species at a time so `resp.var` must be a unidimentional object.

   Response variable must be a `vector` or a one column `data.frame`/`matrix`/`SpatialPointsDataFrame` (`SpatialPoints` are also allowed if you work with 'only presences' data) object. As most of **biomod2** models need Presences AND Absences data, the response variable must contain some absences (if there are not, make sure to select pseudo-absence). In the input `resp.var` argument, the data should be coded in the following way :

   - Presences : 1
   - True Absesnces : 0 (if any)
   - No Information : NA (if any, might latter be used for pseudo-absence)

   If `resp.var` is a non-spatial object (`vector`, `matrix`/`data.frame`) and that some models requiring spatial data are being used (e.g. MAXENT.Phillips) and/or pseudo absences spatialy dependent (i.e 'disk'), make sure to give the XY coordinates of the sites/rows ('resp.xy').

2. **Explanatory variables encoding**

   Explanatory variables must be stored together in a multidimentional object. It may be a `matrix`, a `data.frame`, a `SpatialPointsDataFrame` or a `rasterStack` object. Factorial variables are allowed here even if that can lead to some models omissions.

3. **Evaluation Data**

   If you have data enough, we strongly recommand to split your dataset into 2 part : one for training/calibrating and testing the models and another to evaluate it. If you do it, fill the `eval.resp.var`, `eval.expl.var` and optionally the `eval.resp.xy` arguments whith this data. The advantage of working with a specific dataset for evaluating your models is that you will be able to evaluate more properly your 'ensemble modeled' models. That being said, this

argument is optional and you may prefer only to test (kind of evaluation) your models only with a 'cross-validation' procedure (see Models function). The best practice is to use one set of data for training/calibrating, one set of testing and one for evaluating. The calibration and testing of the data can be done automaticaly in **biomod2** in the Models function. The dataset for evaluation must be entered in `BIOMOD_FormatingData`.

4. **Pseudo Absences selection**

   The `PA.xxx`'s arguments let you parametrise your pseudo absences selection if you want some. It's an optional step.

   Pseudo absences will be selected within the 'background data' and might be constrained by a defined 'strategy'.

   (a) background data
   'Background data' represents data there is no information whether the species of interest occurs or not. It is defined by the 'No Information' data of your `resp.var` if you give some. If not, (i.e Only presences data or all cells with a define presence or absence state) the backgroud will be take into your `expl.var` object if it's a `RasterStack`.

   (b) strategy
   The strategy allows to constrain the choice of pseudo-absence within the 'background data'. 3 ways are currently implemented to select the pseudo-absences candidate cells (`PA.strategy` argument):

   - 'random': all cell of initial background are Pseudo absences candidates. The choice is made randomly given the number of pseudo-absence to select `PA.nb.absences`.
   - 'disk': you may define a minimal (`PA.dist.min`), respectively a maximal (`PA.dist.max`) distance to presences points for selecting your pseudo absences candidates. That may be usefull if you don't want to select pseudo-absences too close to your presences (same niche and to avoid pseudo-replication), respectively too far from your presences (localised sampling startegy).
   - 'sre': Pseudo absences candidates have to be selected in condition that differs from a defined proportion (`PA.sre.quant`) of presences data. It forces pseudo absences to be selected outside of the broadly defined environemental conditions for the species. It means that a surface range envelop model (sre, similar the BIOCLIM) is first carried out (using the specified quantile) on the species of interest, and then the pseudo-absence data are extracted outside of this envelop. This particular case may lead to over optimistic models evaluations.
   - 'user.defined': In this case, pseudo absences selection should have been done in a previous step. This pseudo absences have to be reference into a well formated `data.frame` (e.g. `PA.table` argument)

Value

A `'data.formated.Biomod.object'` for `BIOMOD_Modeling`. It is strongly advised to check whether this formated data corresponds to what was expected. A summary is easily printed by simply tipping the name of the object. A generic plot function is also available to display the different dataset in the geographic space.

Author(s)

Wilfried Thuiller, Damien Georges

See Also

BIOMOD_Modeling

Examples

```
# species occurrences
DataSpecies <- read.csv(system.file("external/species/mammals_table.csv",
                                    package="biomod2"), row.names = 1)
head(DataSpecies)

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

# the presence/absences data for our species
myResp <- as.numeric(DataSpecies[,myRespName])

# the XY coordinates of species data
myRespXY <- DataSpecies[,c("X_WGS84","Y_WGS84")]


# Environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
myExpl = stack( system.file( "external/bioclim/current/bio3.grd",
                     package="biomod2"),
                system.file( "external/bioclim/current/bio4.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio7.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio11.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio12.grd",
                             package="biomod2"))
# 1. Formatting Data
myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
                                     expl.var = myExpl,
                                     resp.xy = myRespXY,
                                     resp.name = myRespName)

myBiomodData
plot(myBiomodData)
```

BIOMOD_LoadModels *Load models built within BIOMOD_Modeling function*

Description

This function was implemented to help **biomod2** users to load individual models built during the BIOMOD_Modeling step.

Usage

```
BIOMOD_LoadModels(bm.out, ... )
```

Arguments

bm.out	a "BIOMOD.models.out" returned by BIOMOD_Modeling
...	additional arguments (see details)

Details

This function is particulary useful when you plane to make some response plot analyses. It will induce models, built at BIOMOD_Modeling step, loading in your working space.

If you run this function referencing only bm.out argument, all models built will be loaded. However, you can make a models subselection using the additional arguments (see below).

Additional arguments (...) : All the following arguments are optional.

- **models:** a character vector defining the names of models (e.g c('GLM', 'GAM', 'RF')) you want to load (models subselection)
- **run.eval:** a character vector defining the names of evaluation run (e.g c('RUN1', 'Full')) you want to load (repetition subselection)
- **data.set:** a character vector defining the names of data.set (e.g c('PA1', 'PA2')) you want to load (pseudo absences subselection)
- **path:** the path to file where the species folder is. To be filled if species folder is different from your working directory)

Value

A character vector filled with the loaded models names.

Note

SRE models are not supported yet. They will be automaticly excluded even if they are selected.

Author(s)

Damien Georges

See Also

BIOMOD_Modeling

Examples

```
# species occurrences
DataSpecies <- read.csv(system.file("external/species/mammals_table.csv",
                                    package="biomod2"), row.names = 1)
head(DataSpecies)

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

# the presence/absences data for our species
myResp <- as.numeric(DataSpecies[,myRespName])

# the XY coordinates of species data
myRespXY <- DataSpecies[,c("X_WGS84","Y_WGS84")]


# Environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
myExpl = stack( system.file( "external/bioclim/current/bio3.grd",
                      package="biomod2"),
                system.file( "external/bioclim/current/bio4.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio7.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio11.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio12.grd",
                             package="biomod2"))

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


# 2. Defining Models Options using default options.
myBiomodOption <- BIOMOD_ModelingOptions()

# 3. Doing Modelisation

myBiomodModelOut <- BIOMOD_Modeling( myBiomodData,
                                       models = c('RF'),
                                       models.options = myBiomodOption,
                                       NbRunEval=2,
                                       DataSplit=70,
                                       models.eval.meth = c('TSS'),
                                       SaveObj = TRUE,
                                       do.full.models = FALSE)

# 4. Loading some models built

myLoadedModels <- BIOMOD_LoadModels(myBiomodModelOut, models='RF')
```

```
myLoadedModels
```

BIOMOD_Modeling *Run a range of species distribution models*

Description

This function allows to calibrate and evaluate a range of species distribution models techniques run over a given species. Calibrations are made on the whole sample or a random subpart. The predictive power of the different models is estimated using a range of evaluation metrics.

Usage

```
BIOMOD_Modeling( data,
                 models = c('GLM','GBM','GAM','CTA','ANN',
                            'SRE','FDA','MARS','RF','MAXENT.Phillips',
                            "MAXENT.Tsuruoka"),
                 models.options = NULL,
                 NbRunEval=1,
                 DataSplit=100,
                 Yweights=NULL,
                 Prevalence=NULL,
                 VarImport=0,
                 models.eval.meth = c('KAPPA','TSS','ROC'),
                 SaveObj = TRUE,
                 rescal.all.models = FALSE,
                 do.full.models = TRUE,
                 modeling.id = as.character(format(Sys.time(), '%s')),
                 ...)
```

Arguments

data	BIOMOD.formated.data object returned by BIOMOD_FormatingData
models	vector of models names choosen among 'GLM', 'GBM', 'GAM', 'CTA', 'ANN', 'SRE', 'FDA', 'MARS', 'RF', 'MAXENT.Phillips' and "MAXENT.Tsuruoka"
models.options	BIOMOD.models.options object returned by BIOMOD_ModelingOptions
NbRunEval	Number of Evaluation run
DataSplit	% of data used to calibrate the models, the remaining part will be used for testing
Yweights	response points weights
Prevalence	either NULL (default) or a 0-1 numeric used to build 'weighted response weights'
VarImport	Number of permutation to estimate variable importance
models.eval.meth	vector of names of evaluation metric among 'KAPPA', 'TSS', 'ROC', 'FAR', 'SR', 'ACCURACY', 'BIAS', 'POD', 'CSI' and 'ETS'

SaveObj	keep all results and outputs on hard drive or not (NOTE: strongly recommended)
rescal.all.models	if true, all model prediction will be scaled with a binomial GLM
do.full.models	if true, models calibrated and evaluated with the whole dataset are done
modeling.id	character, the ID (=name) of modeling procedure. A random number by default.
...	further arguments : • DataSplitTable : a matrix, data.frame or a 3D array filled with TRUE/FALSE to specify which part of data must be used for models calibration (TRUE) and for models validation (FALSE). Each column correspund to a 'RUN'. If filled, args NbRunEval, DataSplit and do.full.models will be ignored.

Details

1. **data**

   If you have decide to add pseudo absences to your original dataset (see BIOMOD_FormatingData), NbPseudoAbsences * NbRunEval + 1 models will be created.

2. **models**

   The set of models to be calibrated on the data. 10 modeling techniques are currently available:

   - GLM : Generalized Linear Model (glm)
   - GAM : Generalized Additive Model (gam, gam or bam, see BIOMOD_ModelingOptions for details on algorith
   - GBM : Generalized Boosting Model or usually called Boosted Regression Trees (gbm)
   - CTA: Classification Tree Analysis (rpart)
   - ANN: Artificial Neural Network (nnet)
   - SRE: Surface Range Envelop or usually called BIOCLIM
   - FDA: Flexible Discriminant Analysis (fda)
   - MARS: Multiple Adaptive Regression Splines (earth)
   - RF: Random Forest (randomForest)
   - MAXENT.Phillips: Maximum Entropy (http://www.cs.princeton.edu/~schapire/maxent/)
   - MAXENT.Tsuruoka: low-memory multinomial logistic regression (maxent)

3. **NbRunEval & DataSplit**

   As already explained in the BIOMOD_FormatingData help file, the common trend is to split the original dataset into two subsets, one to calibrate the models, and another one to evaluate them. Here we provide the possibility to repeat this process (calibration and evaluation) N times (NbRunEval times). The proportion of data kept for calibration is determined by the DataSplit argument (100% - DataSplit will be used to evaluate the model). This sort of cross-validation allows to have a quite robust test of the models when independent data are not available. Each technique will also be calibrated on the complete original data. All the models produced by BIOMOD and their related informations are saved on the hard drive.

4. **Yweights & Prevalence**

   Allows to give more or less weight to some particular observations. If these arguments is kept to NULL (Yweights = NULL, Prevalence = NULL), each observation (presence or absence) has the same weight (independent of the number of presences and absences). If Prevalence = 0.5 absences will be weighted equally to the presences (i.e. the weighted

sum of presence equals the weighted sum of absences). If prevalence is set below or above 0.5 absences or presences are given more weight, respectively. In the particular case that pseudo-absence data have been generated `BIOMOD_FormatingData` (`PA.nb.rep > 0`), weights are by default (`Prevalence = NULL`) calculated such that prevalence is 0.5, meaning that the presences will have the same importance as the absences in the calibration process of the models. Automatically created `Yweights` will be composed of integers to prevent different modelling issues. Note that the `Prevalence` argument will always be ignored if `Yweights` are defined.

5. **models.eval.meth**

   The available evaluations methods are :

   - 'ROC' : Relative Operating Characteristic
   - 'KAPPA' : Cohen's Kappa (Heidke skill score)
   - 'TSS' : True kill statistic (Hanssen and Kuipers discriminant, Peirce's skill score)
   - 'FAR' : False alarm ratio
   - 'SR' : Success ratio
   - 'ACCURANCY' : Accuracy (fraction correct)
   - 'BIAS' : Bias score (frequency bias)
   - 'POD' : Probability of detection (hit rate)
   - 'CSI' : Critical success index (threat score)
   - 'ETS' : Equitable threat score (Gilbert skill score)

   Some of them are scaled to have all an optimum at 1. You can choose one of more (vector) evaluation metric. By Default, only 'KAPPA', 'TSS' and 'ROC' evaluation are done. Please refer to the CAWRC website (http://www.cawcr.gov.au/projects/verification/#Methods_for_dichotomous_forecasts) to get detailled description of each metric.

6. **SaveObj**

   If this argument is set to False, it may prevent the evaluation of the 'ensemble modelled' models in further steps. We strongly recommend to always keep this argument `TRUE` even it asks for free space onto the hard drive.

7. **rescal.all.models**

   This parameter is quite experimental and we adcise not to use it. It should lead to reduction in projection scale amplitude Some categorical models have to be scaled in every case ('FDA', 'ANN'). But It may be interesting to scale all model computed to ensure that they will produced comparable predictions (0-1000 ladder). That's particularly useful when you do some ensemble forecasting to remove the scale prediction effect (the more extended projections are, the more they influence ensemble forecasting results).

8. **do.full.models**

   Building models with all information available may be usefull in some particular cases (i.e. rare species with few presences points). The main drawback of this method is that, if you don't give separated data for models evaluation, your models will be evaluated with the same data that the ones used for calibration. Thats will lead to over-optimistic evaluation scores. Be carefull whith this '_Full' models interpretation.

Value

A BIOMOD.models.out object

See "BIOMOD.models.out" for details.

Additional objects are stored out of R in two different directories for memory storage purposes. They are created by the function directly on the root of your working directory set in R ("models" directory). This one contains each calibrated model for each repetition and pseudo-absence run. A hidden folder '.DATA_BIOMOD' contains some files (predictions, original dataset copy, pseudo absences chosen...) used by other functions like BIOMOD_Projection or BIOMOD_EnsembleModeling .

The models are currently stored as objects to be read exclusively in R. To load them back (the same stands for all objects stored on the hard disk) use the load function (see examples section below).

Author(s)

Wilfried Thuiller, Damien Georges, Robin Engler

See Also

BIOMOD_FormatingData, BIOMOD_ModelingOptions, BIOMOD_Projection

Examples

```
# species occurrences
DataSpecies <- read.csv(system.file("external/species/mammals_table.csv",
                                    package="biomod2"))
head(DataSpecies)

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

# the presence/absences data for our species
myResp <- as.numeric(DataSpecies[,myRespName])

# the XY coordinates of species data
myRespXY <- DataSpecies[,c("X_WGS84","Y_WGS84")]


# Environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
myExpl = stack( system.file( "external/bioclim/current/bio3.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio4.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio7.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio11.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio12.grd",
                             package="biomod2"))

# 1. Formatting Data
myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
                                     expl.var = myExpl,
                                     resp.xy = myRespXY,
```

```
                                    resp.name = myRespName)

# 2. Defining Models Options using default options.
myBiomodOption <- BIOMOD_ModelingOptions()

# 3. Doing Modelisation

myBiomodModelOut <- BIOMOD_Modeling( myBiomodData,
                                     models = c('SRE','RF'),
                                     models.options = myBiomodOption,
                                     NbRunEval=2,
                                     DataSplit=80,
                                     VarImport=0,
                                     models.eval.meth = c('TSS','ROC'),
                                     do.full.models=FALSE,
                                     modeling.id="test")

## print a summary of modeling stuff
myBiomodModelOut
```

BIOMOD_ModelingOptions

Configure the modeling options for each selected model

Description

Function to set the different options for each modeling technique.

Usage

```
BIOMOD_ModelingOptions( GLM = NULL,
                        GBM = NULL,
                        GAM = NULL,
                        CTA = NULL,
                        ANN = NULL,
                        SRE = NULL,
                        FDA = NULL,
                        MARS = NULL,
                        RF = NULL,
                        MAXENT.Phillips = NULL,
                        MAXENT.Tsuruoka = NULL)
```

Arguments

GLM	list of GLM options
GBM	list of GBM options

GAM	list of GAM options
CTA	list of CTA options
ANN	list of ANN options
SRE	list of SRE options
FDA	list of FDA options
MARS	list of MARS options
RF	list of RF options
MAXENT.Phillips	list of MAXENT.Phillips options
MAXENT.Tsuruoka	list of MAXENT.Tsuruoka options

Details

The aim of this function is to allow advanced user to change some default parameters of BIOMOD inner models. For each modeling technique, options can be set up.

Each argument have to be put in a list object.

The best way to use this function is to print defaut models options (`Print_Default_ModelingOptions`) or create a default 'BIOMOD.model.option object' and print it in your console. Then copy the output, change only the required parameters, and paste it as function arguments. (see example)

Here the detailed list of modifiable parameters. They correspond to the traditional parameters that could be setted out for each modeling technique (e.g. ?GLM)

Value

A `"BIOMOD.Model.Options"` object given to `BIOMOD_Modeling`

=-=-= GLM =-=-= (`glm`)

- `myFormula` : a typical formula object (see example). If not NULL, type and interaction.level args are switched off. You can choose to either:
  - generate automatically the GLM formula by using the type and interaction.level arguments type (default `'quadratic'`) : formula given to the model ('simple', 'quadratic' or 'polynomial'). interaction.level (default `0`) : integer corresponding to the interaction level between variables considered. Consider that interactions quickly enlarge the number of effective variables used into the GLM.
  - or construct specific formula
- `test` (default `'AIC'`) : Information criteria for the stepwise selection procedure: AIC for Akaike Information Criteria, and BIC for Bayesian Information Criteria ('AIC' or 'BIC'). 'none' is also a supported value which implies to concider only the full model (no stepwise selection). This can lead to convergence issu and strange results.
- `family` (default `binomial(link = 'logit')`) : a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See family for details of family functions.) . BIOMOD only runs on presence-absence data so far, so binomial family by default.

- control : a list of parameters for controlling the fitting process. For glm.fit this is passed to glm.control.

=-=-= GBM =-=-= (default gbm)

Please refer to gbm help file to get the meaning of this options.

- distribution (default 'bernoulli')
- n.trees (default 2500)
- interaction.depth (default 7)
- n.minobsinnode (default 5)
- shrinkage (default 0.001)
- bag.fraction (default 0.5)
- train.fraction (default 1)
- cv.folds (default 3)
- keep.data (default FALSE)
- verbose (default FALSE)
- perf.method (default 'cv')

=-=-= GAM =-=-= (gam or gam)

- algo : either "GAM_gam" (default), "GAM_mgcv" or "BAM_mgcv" defining the chosen GAM function (see gam, gam resp. bam for more details)
- myFormula : a typical formula object (see example). If not NULL, type and interaction.level args are switched off. You can choose to either:
  - generate automatically the GAM formula by using the type and interaction.level arguments type : the smother used to generate the formula. Only "s_smoother" available at time. interaction.level : integer corresponding to the interaction level between variables considered. Consider that interactions quickly enlarge the number of effective variables used into the GAM. Interaction are not considered if you choosed "GAM_gam" algo
  - or construct specific formula
- k (default -1 or 4): a smooth term in a formula argument to gam (see **gam** s or **mgcv** s)
- family (default binomial(link = 'logit')) : a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See family for details of family functions.) . BIOMOD only runs on presence-absence data so far, so binomial family by default.
- control : see gam.control or gam.control
- some extra "GAM_mgcv" specific options (ignored if algo = "GAM_gam")
  - method (default 'GCV.Cp')
  - optimizer (default c('outer','newton'))
  - select (default FALSE)
  - knots (default NULL)
  - paramPen (default NULL)

=-=-= CTA =-=-= (rpart)

Please refer to rpart help file to get the meaning of the following options.

- method (default 'class')
- parms (default 'default') : if 'default', default **rpart** parms value are kept
- cost (default NULL)
- control: see rpart.control

NOTE: for method and parms, you can give a 'real' value as described in the rpart help file or 'default' that implies default rpart values.

=-=-= ANN =-=-= (nnet)

- NbCV (default 5) : nb of cross validation to find best size and decay parameters
- size (default NULL) : number of units in the hidden layer. If NULL then size parameter will be optimised by cross validation based on model AUC (NbCv cross validation; tested size will be the following c(2,4,6, 8)). You can also specified a vector of size you want to test. The one giving the best model AUC will be then selected.
- decay (default NULL) : parameter for weight decay. If NULL then decay parameter will be optimised by cross validation on model AUC (NbCv cross validation; tested decay will be the following c(0.001, 0.01, 0.05, 0.1)). You can also specified a vector of decay you want to test. The one giving the best model AUC will be then selected.
- rang (default 0.1) : Initial random weights on [-rang, rang]
- maxit (default 200): maximum number of iterations.

=-=-= SRE =-=-= (sre)

- quant (default 0.025): quantile of 'extreme environmental variable' removed for selection of species envelops

=-=-= FDA =-=-= (fda)

Please refer to fda help file to get the meaning of these options.

- method (default 'mars')
- add_args (default NULL) : additional arguments to method given as a list of parameters (correspond to the ... options of fda function)

=-=-= MARS -=-= (earth)

Please refer to earth help file to get the meaning of these options.

- myFormula : a typical formula object (see example). If not NULL, type and interaction.level args are switched off. You can choose to either:
  - generate automatically the GLM formula by using the type and interaction.level arguments type (default 'simple') : formula given to the model ('simple', 'quadratic' or 'polynomial'). interaction.level (default 0) : integer corresponding to the interaction level between variables considered. Consider that interactions quickly enlarge the number of effective variables used into the GLM/MARS.

  - or construct specific formula

- nk (default NULL) : an optional integer specifying the maximum number of model terms. If NULL is given then default mars function value is used (i.e max(21, 2 * nb_expl_var + 1))
- penalty (default 2)
- thresh (default 0.001)
- nprune (default NULL)
- pmethod (default "backward")

=-=-= RF -=-=-= (randomForest)

- do.classif (default TRUE) : if TRUE classification random.forest computed else regression random.forest will be done
- ntree (default 500)
- mtry (default 'default')
- nodesize (default 5)
- maxnodes (default NULL)

NOTE: for mtry, you can give a 'real' value as described in randomForest help file or 'default' that implies default randomForest values

=-=-= MAXENT.Phillips -= http://www.cs.princeton.edu/~schapire/maxent/

- path_to_maxent.jar : character, the link to **maxent.jar** file (the working directory by default)
- memory_allocated : integer (default 512), the amount of memory (in Mo) reserved for java to run MAXENT.Phillips. should be 64, 128, 256, 512, 1024, 2048... or NULL if you want to use default java memory limitation parameter.
- maximumiterations : integer (default 200), maximum iteration done
- visible : logical (default FALSE), make the Maxent user interface visible
- linear : logical (default TRUE), allow linear features to be used
- quadratic : logical (default TRUE), allow quadratic features to be used
- product : logical (default TRUE), allow product features to be used
- threshold : logical (default TRUE), allow threshold features to be used
- hinge : logical (default TRUE), allow hinge features to be used
- lq2lqptthreshold : integer (default 80), number of samples at which product and threshold features start being used
- l2lqthreshold : integer (default 10), number of samples at which quadratic features start being used
- hingethreshold : integer (default 15), number of samples at which hinge features start being used
- beta_threshold : numeric (default -1.0), regularization parameter to be applied to all threshold features; negative value enables automatic setting

- beta_categorical : numeric (default -1.0), regularization parameter to be applied to all categorical features; negative value enables automatic setting
- beta_lqp : numeric (default -1.0), regularization parameter to be applied to all linear, quadratic and product features; negative value enables automatic setting
- beta_hinge : numeric (default -1.0), regularization parameter to be applied to all hinge features; negative value enables automatic setting
- betamultiplier : numeric (default 1), multiply all automatic regularization parameters by this number. A higher number gives a more spread-out distribution.
- defaultprevalence : numeric (default 0.5), default prevalence of the species: probability of presence at ordinary occurrence points

=-=-= MAXENT.Tsuruoka -=-=-= (maxent)

- l1_regularizer (default 0.0): An numeric turning on L1 regularization and setting the regularization parameter. A value of 0 will disable L1 regularization
- l2_regularizer (default 0.0): An numeric turning on L2 regularization and setting the regularization parameter. A value of 0 will disable L2 regularization
- use_sgd (default FALSE): A logical indicating that SGD parameter estimation should be used. Defaults to FALSE
- set_heldout (default 0): An integer specifying the number of documents to hold out. Sets a held-out subset of your data to test against and prevent overfitting
- verbose (default FALSE): A logical specifying whether to provide descriptive output about the training process

NOTE: if you use the set_heldout parameter then the data that will be held out will be taken in the calibration data pool. It can be penilizing in case of low number of occurences dataset.

Author(s)

Wilfried Thuiller, Damien Georges

Examples

```
# default BIOMOD.model.option object
myBiomodOptions <- BIOMOD_ModelingOptions()

# print the object
myBiomodOptions

# you can copy a part of the print, change it and custom your options
# here we want to compute quadratic GLM and select best model with 'BIC' criterium
myBiomodOptions <- BIOMOD_ModelingOptions(
                      GLM = list( type = 'quadratic',
                                  interaction.level = 0,
                                  myFormula = NULL,
                                  test = 'BIC',
                                  family = 'binomial',
                                  control = glm.control(epsilon = 1e-08,
                                              maxit = 1000,
```

```
                                                trace = FALSE) ))

# check changes was done
myBiomodOptions

# you can prefer to establish your own GLM formula
myBiomodOptions <- BIOMOD_ModelingOptions(
                    GLM = list( myFormula = formula("Sp277 ~ bio3 +
                    log(bio10) + poly(bio16,2) + bio19 + bio3:bio19")))

# check changes was done
myBiomodOptions

# you also can directly print default parameters and then follow the same processus
Print_Default_ModelingOptions()
```

BIOMOD_presenceonly *evaluate models with presences only metrics*

Description

This function enables to evaluate BIOMOD.models.out and BIOMOD.EnsembleModeling.out object with presence-only evaluation methods (Boyce index and Minimal Predicted Area MPA)

Usage

```
BIOMOD_presenceonly(modeling.output = NULL, EM.output = NULL,
  save.output = T)
```

Arguments

modeling.output	"BIOMOD.models.out" object produced by a BIOMOD_Modeling run
EM.output	a "BIOMOD.EnsembleModeling.out" returned by BIOMOD_EnsembleModeling
save.output	logical. If TRUE (Default) the output is saved to the ".BIOMOD_DATA" folder

Details

'em.by' of 'BIOMOD.EnsembleModeling' must be 'PA_dataset+repet' to have an ensemble for each RUN of the 'NbRunEval' argument (BIOMOD_Modeling funtion) for evaluation. The Boyce index returns NA values for 'SRE' models because it is not possible to be calculated with binary predictions. This is also the reason why there are sometimes NA values for 'GLM' models if they don not converge.

Value

data.frame containing evaluation scores for the evaluation metrics used for the BIOMOD_Modeling function and additional Boyce index and MPA

Author(s)

Frank Breiner

References

Engler, R., A. Guisan, and L. Rechsteiner. 2004. An improved approach for predicting the distribution of rare and endangered species from occurrence and pseudo-absence data. Journal of Applied Ecology.

See Also

ecospat.boyce, ecospat.mpa, BIOMOD_Modeling, BIOMOD_EnsembleModeling

Examples

```
## Not run:
require(PresenceAbsence)

# species occurrences
DataSpecies <- read.csv(system.file("external/species/mammals_table.csv",
                                    package="biomod2"), row.names = 1)
head(DataSpecies)

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

# the presence/absences data for our species
myResp <- as.numeric(DataSpecies[,myRespName])

# the XY coordinates of species data
myRespXY <- DataSpecies[,c("X_WGS84","Y_WGS84")]


# Environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
myExpl = stack( system.file( "external/bioclim/current/bio3.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio4.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio7.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio11.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio12.grd",
                             package="biomod2"))

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

# 2. Defining Models Options using default options.
```

```
myBiomodOption <- BIOMOD_ModelingOptions()

# 3. Doing Modelisation

myBiomodModelOut <- BIOMOD_Modeling( myBiomodData,
                                     models = c('SRE','CTA','RF'),
                                     models.options = myBiomodOption,
                                     NbRunEval=1,
                                     DataSplit=80,
                                     Yweights=NULL,
                                     VarImport=3,
                                     models.eval.meth = c('TSS','ROC'),
                                     SaveObj = TRUE,
                                     rescal.all.models = FALSE,
                                     do.full.models = FALSE)

# 4. Doing Ensemble Modelling
myBiomodEM <- BIOMOD_EnsembleModeling( modeling.output = myBiomodModelOut,
                                       chosen.models = 'all',
                                       em.by = 'PA_dataset+repet',
                                       eval.metric = c('TSS'),
                                       eval.metric.quality.threshold = c(0.7),
                                       models.eval.meth = c('TSS','ROC'),
                                       prob.mean = TRUE,
                                       prob.cv = FALSE,
                                       prob.ci = FALSE,
                                       prob.ci.alpha = 0.05,
                                       prob.median = FALSE,
                                       committee.averaging = FALSE,
                                       prob.mean.weight = TRUE,
                                       prob.mean.weight.decay = 'proportional' )

# evaluate Biomod models with the Boyce index and MPA
pres.only.eval <- BIOMOD_presenceonly(myBiomodModelOut, myBiomodEM)
pres.only.eval$eval

## End(Not run)
```

`BIOMOD_Projection` *Project the calibrated models within* **biomod2** *into new space or time*

Description

For all the models currently implemented, **biomod2** is able to project potential distributions of species in other areas, other resolutions or other time scales.

Usage

```
BIOMOD_Projection(modeling.output,
                  new.env,
```

```
                    proj.name,
                    xy.new.env = NULL,
                    selected.models = 'all',
                    binary.meth = NULL,
                    filtered.meth = NULL,
                    compress = TRUE,
                    build.clamping.mask = TRUE,
                    ...)
```

Arguments

modeling.output	"BIOMOD.models.out" object produced by a BIOMOD_Modeling run
new.env	A set of explanatory variables onto which models will be projected . It could be a data.frame, a matrix, or a rasterStack object. Make sure the column names (data.frame or matrix) or layer Names (rasterStack) perfectly match with the names of variables used to build the models in the previous steps.
proj.name	a character defining the projection name (a new folder will be created with this name)
xy.new.env	optional coordinates of new.env data. Ignored if new.env is a rasterStack
selected.models	'all' when all models have to be used to render projections or a subset vector of modelling.output models computed (accessing with the slot @models.computed of your "BIOMOD.models.out" object)
binary.meth	a vector of a subset of models evaluation method computed before (see BIOMOD_Modeling). If NULL then no binary transformation computed, else the given binary techniques will be used to transform the projection into 0/1 data.
filtered.meth	a vector of a subset of models evaluation method computed before (see BIOMOD_Modeling). if NULL then no filtering transformation computed, else the given binary techniques will be used to transform the projection by stetting to 0 the probability values below a specific threshold.
compress	Boolean or character, the compression format of objects stored on your hard drive. May be one of 'TRUE', 'FALSE', 'xz' or 'gzip' (see save)
build.clamping.mask	if TRUE, a clamping mask will be saved on hard drive different (see details)
...	Additional arguments (see details section)

Details

Projections are done for all selected models, that means (by default) for all evaluation run, and pseudo absences selections if applicable. This projections may be used later to compute ensemble forecasting.

If build.clamping.mask is set to TRUE a file (same type than new.env arg) will be saved in your projection folder. This mask will identifies locations where predictions are uncertain because the values of the variables are outside the range used for calibrating the models. The 'build.clamping.mask' values correspond to the number of variables that are out of their calibrating/training range. (see vignette for more details)

... may be :

- silent:logical, if TRUE, console outputs are turned off
- do.stack: logical, if TRUE, attempt to save all projections in a unique object i.e RasterStack. If FALSE or if objects are too heavy to be load all together in memory, projections will be stored into separated files.
- keep.in.memory:logical, if FALSE only the link pointing to a hard drive copy of projections are stored in output object. That can be useful to prevent memory issues.
- output.format:whether '.RData', '.grd' or '.img' defining projections saving format (on hard drive). If new.env argument is under table format (data.frame or matrix), the only choice you have is '.RData'
- omit.na:logical, if TRUE (default), all not fully referenced environmental points will get a NA as prediction. If FALSE, models that can produce predictions with incomplete data will return a prediction value for this points.
- on_0_1000:logical, if TRUE (default), 0 - 1 probabilities are converted into a 0 - 1000 integer scale. This implies a lot of memory saving. User that want to comeback on a 0 - 1 scale latter will just have to divide all projections by 1000

Value

Returns the projections for all selected model ("BIOMOD.projection.out" object), and stored in the hard drive on the specific directory names by the name of the projection. The data is a 4-dimensions array (see ...) if new.env is a matrix or a data.frame. It is a rasterStack if new.env is a rasterStack and or several rasterLayers if the rasterStack is too large.

A new folder is also created on your hard drive. This folder contains the created projection object (basic one and binary and filtered ones if selected). The object are loaded with the load function. The loaded object can be then plotted and analysed.

Author(s)

Wilfried Thuiller, Damien Georges

See Also

BIOMOD_Modeling, BIOMOD_FormatingData, BIOMOD_ModelingOptions

Examples

```
# species occurrences
DataSpecies <- read.csv(system.file("external/species/mammals_table.csv",
                                    package="biomod2"), row.names = 1)
head(DataSpecies)

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

# the presence/absences data for our species
myResp <- as.numeric(DataSpecies[,myRespName])
```

```
# the XY coordinates of species data
myRespXY <- DataSpecies[,c("X_WGS84","Y_WGS84")]


# Environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
myExpl = stack( system.file( "external/bioclim/current/bio3.grd",
                     package="biomod2"),
                system.file( "external/bioclim/current/bio4.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio7.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio11.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio12.grd",
                             package="biomod2"))
# 1. Formatting Data
myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
                                     expl.var = myExpl,
                                     resp.xy = myRespXY,
                                     resp.name = myRespName)

# 2. Defining Models Options using default options.
myBiomodOption <- BIOMOD_ModelingOptions()

# 3. Doing Modelisation

myBiomodModelOut <- BIOMOD_Modeling( myBiomodData,
                                      models = c('SRE','RF'),
                                      models.options = myBiomodOption,
                                      NbRunEval=1,
                                      DataSplit=70,
                                      models.eval.meth = c('TSS'),
                                      do.full.models = FALSE)


# 4.1 Projection on current environemental conditions

myBiomodProjection <- BIOMOD_Projection(modeling.output = myBiomodModelOut,
                                        new.env = myExpl,
                                        proj.name = 'current',
                                        selected.models = 'all',
                                        binary.meth = 'TSS',
                                        compress = FALSE,
                                        build.clamping.mask = FALSE)

## Not run:
# 4.2 Projection on future environemental conditions
myExplFuture = stack(system.file("external/bioclim/future/bio3.grd",package="biomod2"),
                     system.file("external/bioclim/future/bio4.grd",package="biomod2"),
                     system.file("external/bioclim/future/bio7.grd",package="biomod2"),
                     system.file("external/bioclim/future/bio11.grd",package="biomod2"),
                     system.file("external/bioclim/future/bio12.grd",package="biomod2"))
```

```
myBiomodProjectionFuture <- BIOMOD_Projection(modeling.output = myBiomodModelOut,
                                              new.env = myExplFuture,
                                              proj.name = 'future',
                                              selected.models = 'all',
                                              binary.meth = 'TSS',
                                              compress = FALSE,
                                              build.clamping.mask = TRUE)

# print summary and plot projections
myBiomodProjectionFuture
plot(myBiomodProjectionFuture)

## End(Not run)
```

BIOMOD_RangeSize *Analysis of the range size changes*

Description

This function allows to estimate the proportion and relative number of pixels (or habitat) lost, gained and stable for the time slice considered in species-climate modelling under future scenarios.

Usage

```
## S4 method for signature 'data.frame,data.frame'
BIOMOD_RangeSize(CurrentPred, FutureProj,  SpChange.Save=NULL)

## S4 method for signature 'array,array'
BIOMOD_RangeSize(CurrentPred, FutureProj,  SpChange.Save=NULL)

## S4 method for signature 'RasterStack,RasterStack'
BIOMOD_RangeSize(CurrentPred, FutureProj,  SpChange.Save=NULL)

## S4 method for signature 'RasterStack,RasterStack'
BIOMOD_RangeSize(CurrentPred, FutureProj,  SpChange.Save=NULL)

## S4 method for signature 'RasterLayer,RasterLayer'
BIOMOD_RangeSize(CurrentPred, FutureProj,  SpChange.Save=NULL)
```

Arguments

CurrentPred	a data.frame of n (number of models) columns OR an array formated as an output of BIOMOD_Projection function OR a RasterStack (a layer by model) giving the current state of the species in binary

FutureProj — a data.frame of n (number of models) columns OR an array formated as an output of BIOMOD_Projection function OR a RasterStack (a layer by model) giving the future projections of the species in binary according to a future scenario

SpChange.Save — the name given to the new object storing the results

Details

Note that this function is only relevant if you make projections on the same area with the same resolution and for a different time slice (past or future) than for the current data.

Value

A list of two items is created: Compt.By.Species and Diff.By.Pixel ; Compt.By.Species is the summary of this function; Diff.By.Pixel is in the same form than your input data.

Compt.By.Species
: stores the summary of range change for each species (sorted by rows). The first four columns are absolute values whereas the next 3 ones are relative values:

Disa — represents the number of pixels predicted to be lost by the given species.

Stable0 — is the number of pixels which are not currently occupied by the given species and not predicted to be.

Stable1 — represents the number of pixels currently occupied by the given species, and predicted to remain occupied into the future.

Gain — represents the number of pixels which are currently not occupied by the given species but predicted to be into the future.

PercLoss — corresponds to the percentage of currently occupied sites to be lost (Disa/(Disa+Stable1)

PercGain — corresponds to the percentage of new sites considering the species' current distribution size (Gain/(Disa+Stable1). For example, if the there are 30 sites currently occupied and 15 new sites are projected to be occupied in future, it makes PercGain=+50(%).

SpeciesRangeChange
: it is the overall projection outcome, equal to PercGain-PercLoss. It does not assess for any migration shifts as it strictly compares the range sizes between current and future states.

CurrentRangeSize
: represents the modelled current range size (number of pixels occupied) of the given species.

FutureRangeSize0Disp
: represents the future modelled range size assuming no migration of the given species.

FutureRangeSize1Disp
: represents the future modelled range size assuming migration of the given species (depending on the datasets given in input, if Migration has been used or not).

`Diff.By.Pixel` the summary of range change for each species (sorted by columns and with the pixel in rows). For each species, a pixel could have four different values : -2 if the given pixel is predicted to be lost by the species. -1 if the given pixel is predicted to be stable for the species. 0 is the given pixel was not occupied, and will not be in the future. 1 if the given pixel was not occupied, and is predicted to be into the future. This table could be easily plotted into GIS software in order to represent the pattern of change for the selected species (or even with the level.plot() function).

Author(s)

Wilfried Thuiller, Damien Georges, Bruno Lafourcade

Examples

```
# species occurrences
DataSpecies <- read.csv(system.file("external/species/mammals_table.csv",
                                    package="biomod2"), row.names = 1)
head(DataSpecies)

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

# the presence/absences data for our species
myResp <- as.numeric(DataSpecies[,myRespName])

# the XY coordinates of species data
myRespXY <- DataSpecies[,c("X_WGS84","Y_WGS84")]


# Environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
myExpl = stack( system.file( "external/bioclim/current/bio3.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio4.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio7.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio11.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio12.grd",
                             package="biomod2"))

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


# 2. Defining Models Options using default options.
myBiomodOption <- BIOMOD_ModelingOptions()
```

```
# 3. Doing Modelisation

myBiomodModelOut <- BIOMOD_Modeling( myBiomodData,
                                     models = c('CTA','RF'),
                                     models.options = myBiomodOption,
                                     models.eval.meth ='TSS',
                                     rescal.all.models=FALSE)


# 4.1 Projection on current environemental conditions

myBiomodProjection <- BIOMOD_Projection(modeling.output = myBiomodModelOut,
                                        new.env = myExpl,
                                        proj.name = 'current',
                                        selected.models = 'all',
                                        binary.meth = 'TSS',
                                        compress = FALSE,
                                        build.clamping.mask = FALSE)

# 4.2 Projection on future environemental conditions

myExplFuture = stack(system.file("external/bioclim/future/bio3.grd",package="biomod2"),
                     system.file("external/bioclim/future/bio4.grd",package="biomod2"),
                     system.file("external/bioclim/future/bio7.grd",package="biomod2"),
                     system.file("external/bioclim/future/bio11.grd",package="biomod2"),
                     system.file("external/bioclim/future/bio12.grd",package="biomod2"))

myBiomodProjectionFuture <- BIOMOD_Projection(modeling.output = myBiomodModelOut,
                                              new.env = myExplFuture,
                                              proj.name = 'future',
                                              selected.models = 'all',
                                              binary.meth = 'TSS',
                                              compress = FALSE,
                                              build.clamping.mask = TRUE)

# 5. Detect where our species occurances state is forecasted to change

# load binary projections
# here is rasters objects ('.grd')
currentPred <- stack("GuloGulo/proj_current/proj_current_GuloGulo_TSSbin.grd")
futurePred <- stack("GuloGulo/proj_future/proj_future_GuloGulo_TSSbin.grd")


# call the Range size function
myBiomodRangeSize <- BIOMOD_RangeSize(
  CurrentPred=currentPred,
  FutureProj=futurePred)

# see the results
myBiomodRangeSize$Compt.By.Models
plot(myBiomodRangeSize$Diff.By.Pixel)
```

BIOMOD_tuning *Tune models parameters*

Description

Function to tune biomod single models parameters

Usage

```
BIOMOD_tuning(data, models = c("GLM", "GBM", "GAM", "CTA", "ANN", "FDA",
  "MARS", "RF", "MAXENT.Phillips"), models.options = BIOMOD_ModelingOptions(),
  method.ANN = "avNNet", method.RF = "rf", method.MARS = "earth",
  method.GAM = "gam", method.GLM = "glmStepAIC", trControl = NULL,
  metric = "ROC", ctrl.CTA = NULL, ctrl.RF = NULL, ctrl.ANN = NULL,
  ctrl.MARS = NULL, ctrl.FDA = NULL, ctrl.GAM = NULL, ctrl.GBM = NULL,
  ctrl.GLM = NULL, tuneLength = 30, decay.tune.ANN = c(0.001, 0.01, 0.05,
  0.1), size.tune.ANN = c(2, 4, 6, 8), maxit.ANN = 500, MaxNWts.ANN = 10 *
  (ncol(data@data.env.var) + 1) + 10 + 1, type.GLM = "simple",
  cvmethod.ME = "randomkfold", overlap.ME = FALSE, bin.output.ME = TRUE,
  kfolds.ME = 10, n.bg.ME = 10000, env.ME = NULL, metric.ME = "ROC",
  clamp.ME = T)
```

Arguments

data	BIOMOD.formated.data object returned by BIOMOD_FormatingData
models	vector of models names choosen among 'GLM', 'GBM', 'GAM', 'CTA', 'ANN', 'FDA', 'MARS', 'RF' and 'MAXENT.Phillips'
models.options	BIOMOD.models.options object returned by BIOMOD_ModelingOptions. Default: BIOMOD_ModelingOptions()
method.ANN	which classification or regression model to use for artificial neural networks (default: "avNNet"). see http://topepo.github.io/caret/Neural_Network.html
method.RF	which classification or regression model to use for randomForest (default: "rf"). see http://topepo.github.io/caret/Random_Forest.html
method.MARS	which classification or regression model to use for mars (default: "earth"). see http://topepo.github.io/caret/Multivariate_Adaptive_Regression_Splines.html
method.GAM	which classification or regression model to use for GAM (default: "gam"). see http://topepo.github.io/caret/Generalized_Additive_Model.html
method.GLM	which classification or regression model to use for GLM: (default: 'glmStepAIC'). see http://topepo.github.io/caret/Generalized_Linear_Model.html
trControl	global control parameters for runing (default trainControl(method="cv",summaryFunction=twoClassSum = FALSE). for details see trainControl
metric	metric to select the optimal model (Default ROC). TSS (maximizing Sensitivity and Specificity) is also possible. see ?train
ctrl.CTA	specify control parameters only for CTA (default trControl)

ctrl.RF	specify control parameters only for RF (default trControl)
ctrl.ANN	specify control parameters only for ANN (default trControl)
ctrl.MARS	specify control parameters only for MARS (default trControl)
ctrl.FDA	specify control parameters only for FDA (default trControl)
ctrl.GAM	specify control parameters only for GAM (default trControl)
ctrl.GBM	specify control parameters only for GBM (default trControl)
ctrl.GLM	specify control parameters only for GLM (default trControl)
tuneLength	see ?train (default 30)
decay.tune.ANN	weight decay parameters used for tuning for ANN (default: c(0.001, 0.01, 0.05, 0.1)) Will be optimised by method specified in ctrl.ANN (if not available trControl).
size.tune.ANN	size parameters (number of units in the hidden layer) for ANN used for tuning (default: c(2,4,6,8)). Will be optimised using the method specified in ctrl.ANN (if not available trControl).
maxit.ANN	maximum number of iterations for ANN (default 500)
MaxNWts.ANN	The maximum allowable number of weights for ANN (default 10 * (ncol(myBiomodData'at'data.env.var) + 1) + 10 + 1).
type.GLM	either 'simple', 'quadratic', 'polynomial' or 's_smoother' defining the type of formula you want to build with GLM
cvmethod.ME	method used for data partitioning for MAXENT.Phillips (default: 'randomkfold')
overlap.ME	logical; Calculates pairwise metric of niche overlap if TRUE (Default: FALSE). (see ?calc.niche.overlap)
bin.output.ME	logical; If TRUE, appends evaluations metrics for each evaluation bin to results table (i.e., in addition to the average values across bins).
kfolds.ME	number of bins to use for k-fold cross-validation used for MAXENT.Phillips (Default: 10).
n.bg.ME	Number of Background points used to run MAXENT.Phillips (Default: 10000)
env.ME	RasterStack of model predictor variables
metric.ME	metric to select the optimal model for Maxent (Default: ROC). One out of Mean.AUC (or ROC), delta.AICc, Mean.AUC.DIFF. see ?ENMevaluate
clamp.ME	logical; If TRUE (Default) "Features are constrained to remain within the range of values in the training data" (Elith et al. 2011)

Details

Tuning ANN: if no parameters are specified for ANN, they are tuned internally in BIOMOD_Modelling using cross-validation. ANN parameters are therefore tuned in every case either within ecospat.BIOMOD.tunning or BIOMOD_Modelling

Value

BIOMOD.models.options object with optimized parameters

Author(s)

Frank Breiner

See Also

BIOMOD_ModelingOptions(), train

Examples

```
## Not run:
# species occurrences
DataSpecies <- read.csv(system.file("external/species/mammals_table.csv",
                                    package="biomod2"))
head(DataSpecies)

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

# the presence/absences data for our species
myResp <- as.numeric(DataSpecies[,myRespName])

# the XY coordinates of species data
myRespXY <- DataSpecies[,c("X_WGS84","Y_WGS84")]

# Environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
myExpl = stack( system.file( "external/bioclim/current/bio3.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio4.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio7.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio11.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio12.grd",
                             package="biomod2"))
# 1. Formatting Data
myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
                                     expl.var = myExpl,
                                     resp.xy = myRespXY,
                                     resp.name = myRespName)

# 2. Defining Models Options using default options.
### Duration for turing all models sequential with default settings
### on 3.4 GHz processor: approx. 45 min tuning all models in parallel
### (on 8 cores) using foreach loops runs much faster: approx. 14 min

#library(doParallel);cl<-makeCluster(8);registerDoParallel(cl)


time.seq<-system.time(Biomod.tuning <- BIOMOD_tuning(myBiomodData,
                                                     env.ME = myExpl,
                                                     n.bg.ME = ncell(myExpl)))
```

```
#stopCluster(cl)

myBiomodModelOut <- BIOMOD_Modeling( myBiomodData,
                                     models = c('RF','CTA'),
                                     models.options = Biomod.tuning$models.options,
                                     NbRunEval=1,
                                     DataSplit=100,
                                     VarImport=0,
                                     models.eval.meth = c('ROC'),
                                     do.full.models=FALSE,
                                     modeling.id="test")


#  eval.plot(Biomod.tuning$tune.MAXENT.Phillips at results)
par(mfrow=c(1,3))
plot(Biomod.tuning$tune.CTA.rpart)
plot(Biomod.tuning$tune.CTA.rpart2)
plot(Biomod.tuning$tune.RF)

## End(Not run)
```

`calculate.stat` *Calculate evaluation metrics based on a misclassification table*

Description

`calculate.stat` is an internal **biomod2** function to get scores, based on a misclassification table, of some referenced evaluation metrics.

Usage

```
calculate.stat( Misc,
                stat='TSS')
```

Arguments

`Misc`	a misclassification table
`stat`	either 'TSS', 'KAPPA', 'ACCURACY', 'BIAS', 'POD', 'FAR', 'POFD', 'SR', 'CSI', 'ETS', 'HK', 'HSS', 'OR' or 'ORSS'

Details

Please refer to `BIOMOD_Modeling` to get more information about this metrics.

Value

The `stat` score for the `Misc` table.

Author(s)

Damien Georges

See Also

BIOMOD_Modeling, getStatOptimValue, Find.Optim.Stat

Examples

```
a <- sample(c(0,1),100, replace=TRUE)
b <- sample(c(0,1),100, replace=TRUE)

miscTab_aa <- table(a,a)
miscTab_ab <- table(a,b)

# perfect score
calculate.stat( miscTab_aa, stat='TSS')
# random score
calculate.stat( miscTab_ab, stat='TSS')
```

`CustomIndexMaker` *Replace default package Index help file by a custom one.*

Description

This function replace default html index file by a custom one if defined

Usage

```
CustomIndexMaker()
```

Details

....

Value

A `logical` specifying if operation succed or not

Author(s)

Wilfried Thuiller, Damien Georges

Examples

```
## Automaticly done at buildinfg package state
# CustomIndexMaker()
```

DF_to_ARRAY *Convert a biomod2 data.frame (or list) into array*

Description

DF_to_ARRAY and LIST_to_ARRAY are **biomod2** internal functions that can be useful to help users to transform a standard **biomod2** `data.frame` or `list` output into the standard `array` one.

Usage

```
DF_to_ARRAY(df)
LIST_to_ARRAY(ll)
```

Arguments

`df`	a standard **biomod2** `data.frame` output
`ll`	a standard **biomod2** `list` output (e.g `evaluate` function output)

Details

This functions can be useful when you want to re-convert **biomod2** `data.frame` or `list` (e.g. projections, evaluations) into their initial `array` format (see `BIOMOD_Projection` for further details)

Value

A standard **biomod2** `array` output.

Author(s)

Damien Georges

See Also

`BIOMOD_Projection`

Examples

```
# species occurrences
DataSpecies <- read.csv(system.file("external/species/mammals_table.csv",
                                    package="biomod2"), row.names = 1)
head(DataSpecies)

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

# the presence/absences data for our species
myResp <- as.numeric(DataSpecies[,myRespName])
```

```
# the XY coordinates of species data
myRespXY <- DataSpecies[,c("X_WGS84","Y_WGS84")]


# Environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
myExpl = stack( system.file( "external/bioclim/current/bio3.grd",
                     package="biomod2"),
                system.file( "external/bioclim/current/bio4.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio7.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio11.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio12.grd",
                             package="biomod2"))

# Keep only points where we have info
myExpl <- extract(myExpl, myRespXY)

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

# 2. Defining Models Options using default options.
myBiomodOption <- BIOMOD_ModelingOptions()

# 3. Doing Modelisation

myBiomodModelOut <- BIOMOD_Modeling( myBiomodData,
                                       models = c('SRE','RF'),
                                       models.options = myBiomodOption,
                                       NbRunEval=1,
                                       DataSplit=70,
                                       Yweights=NULL,
                                       VarImport=0,
                                       models.eval.meth = c('ROC'),
                                       rescal.all.models = FALSE,
                                       do.full.models = FALSE)


# 4 Projection on current environemental conditions

myBiomodProjection <- BIOMOD_Projection(modeling.output = myBiomodModelOut,
                                          new.env = data.frame(myExpl),
                                          proj.name = 'current',
                                          selected.models = 'all')



# 5. Get projection under data.frame format
myProjDF <- get_predictions(myBiomodProjection, as.data.frame=TRUE)
```

```
class(myProjDF)
dim(myProjDF)
dimnames(myProjDF)

# 6. Transform data.frame into array
myProjArray <- DF_to_ARRAY(myProjDF)
class(myProjArray)
dim(myProjArray)
dimnames(myProjArray)
```

`evaluate` *biomod2 modelling outputs evaluation*

Description

This function will evaluate biomod2 modelling output for given metrics (e.g 'TSS', 'ROC'...) for a given dataset.

Usage

```
evaluate(model, data, stat, as.array=FALSE)
```

Arguments

`model`	the model you want evaluate (either `"BIOMOD.models.out"`, `"BIOMOD.EnsembleModeling.out"` or `"biomod2_model"`)
`data`	the `data.set` on which you want to perform analyses. Must be a dataset with first column containing the observed data for your species. The flowing columns must be the explanatory variables at observed points. Be sure that columns names of your dataset are the name of your species then the names of variables used for building models at previous steps.
`stat`	vector of statistic metrics names (e.g 'TSS','ROC') you want to perform. (see `BIOMOD_Modeling`) to get the list of all available metrics)
`as.array`	logical, (FALSE by default) if FALSE a list of evaluation tables is returned (one item by models). If TRUE, the output will be return under 'classical' biomod2 array objects (see `BIOMOD_Projection`)

Details

Given model predictive score is evaluated on the new data set. It is done comparing binary transformed model predictions (on data set) to species occurrences (first column of `data` arg). A list of available evaluation metrics is given in `BIOMOD_Modeling` help file. For metrics that compared binary/binary data, a set of threshold will be test to transform continuous model prediction within binary ones. The return scores are the ones obtained for the threshold optimising tested metric (best score).

Value

a `list` or an `array` containing for each evaluation metric the score, the threshold considered to transform continuous data into binary ones (for all metrics excepted 'ROC') and associated sensibility and specificity.

Author(s)

Damien Georges

See Also

BIOMOD_Modeling, BIOMOD_EnsembleModeling, variables_importance

Examples

```
# species occurrences
DataSpecies <- read.csv(system.file("external/species/mammals_table.csv",
                                    package="biomod2"), row.names = 1)
head(DataSpecies)

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

# the presence/absences data for our species
myResp <- as.numeric(DataSpecies[,myRespName])

# the XY coordinates of species data
myRespXY <- DataSpecies[,c("X_WGS84","Y_WGS84")]


# Environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
myExpl = stack( system.file( "external/bioclim/current/bio3.grd",
                     package="biomod2"),
                system.file( "external/bioclim/current/bio4.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio7.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio11.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio12.grd",
                             package="biomod2"))

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

# 2. Defining Models Options using default options.
myBiomodOption <- BIOMOD_ModelingOptions()
```

```
# 3. Doing Modelisation

myBiomodModelOut <- BIOMOD_Modeling( myBiomodData,
                                     models = c('SRE','CTA','RF'),
                                     models.options = myBiomodOption,
                                     NbRunEval=1,
                                     DataSplit=80,
                                     Yweights=NULL,
                                     VarImport=3,
                                     models.eval.meth = c('TSS'),
                                     SaveObj = TRUE,
                                     rescal.all.models = FALSE,
                                     do.full.models = FALSE,
                                     modeling.id='test')

# 4. Evaluate model over another dataset (here the full one)

## creation of suitable dataset
data <- cbind(GuloGulo=get_formal_data(myBiomodModelOut,'resp.var'),
              get_formal_data(myBiomodModelOut,'expl.var'))

## evaluation
evaluate(myBiomodModelOut, data=data, stat=c('ROC','TSS'))
```

FilteringTransformation

Convert species' probability of occurrence into binary presence-absence data using a predefined threshold

---

Description

Function that converts an object containing probability values into a filtered object according to a pre-defined threshold(s).

Usage

```
## S4 method for signature 'numeric'
FilteringTransformation(data, threshold)

## S4 method for signature 'matrix'
FilteringTransformation(data, threshold)

## S4 method for signature 'data.frame'
FilteringTransformation(data, threshold)

## S4 method for signature 'array'
FilteringTransformation(data, threshold)
```

```
## S4 method for signature 'RasterLayer'
FilteringTransformation(data, threshold)

## S4 method for signature 'RasterStack'
FilteringTransformation(data, threshold)
```

Arguments

data	a numeric vector, a matrix, a data.frame, a RasterLayer or a RasterStack containing the data to be converted
threshold	a numeric value or a vector containing the threshold to be used for converting data.

Details

If data is a vector or a raster object, then the threshold should be a numeric value. If data is matrix,dataframe or rasterStack, then the threshold should have, in theory, as many values as the number of columns or layers to transform. In the particular case that the data to convert is a matrix/data.frame with several columns or a RasterStack with several layers and the threshold is a single numeric value, the same threshold will be applied to all columns (resp. layers).

Value

An object of the same class than data with the values of data if suprerior to threshold and 0 if not.

Methods

```
signature(data = "data.frame")
signature(data = "matrix")
signature(data = "numeric")
signature(data = "RasterBrick")
signature(data = "RasterLayer")
signature(data = "RasterStack")
```

Author(s)

Wilfried Thuiller, Damien Georges

See Also

help

Examples

```
xx <- rnorm(50,10)
yy <- FilteringTransformation(xx, 10)

cbind(xx,yy)
```

`Find.Optim.Stat` *Calculate the best score according to a given evaluation method*

Description

`Find.Optim.Stat` is an internal **biomod2** function to find the threshold to convert continuous values into binary ones leading to the best score for a given evaluation metric.

Usage

```
Find.Optim.Stat(Stat='TSS',
                Fit,
                Obs,
                Nb.thresh.test = 100,
                Fixed.thresh = NULL)
```

Arguments

`Stat`	either 'ROC', TSS', 'KAPPA', 'ACCURACY', 'BIAS', 'POD', 'FAR', 'POFD', 'SR', 'CSI', 'ETS', 'HK', 'HSS', 'OR' or 'ORSS'
`Fit`	vector of fitted values (continuous)
`Obs`	vector of observed values (binary)
`Nb.thresh.test`	integer, the numer of thresholds tested over the range of fitted value
`Fixed.thresh`	integer, if not `NULL`, the only threshold value tested

Details

Please refer to `BIOMOD_Modeling` to get more information about this metrics. If you give a `Fixed.thresh`, no optimisation will be done. Only the score for this threshold will be returned.

Value

A 1 row x 4 column `matrix` :

- `best.iter`: the best score obtained for chosen statistic
- `cutoff`: the associated cut-off used for transform fitted vector into binary
- `sensibility`: the sensibility with this threshold
- `specificity`: the specificity with this threshold

Author(s)

Damien Georges

See Also

BIOMOD_Modeling, getStatOptimValue, calculate.stat

Examples

```
a <- sample(c(0,1),100, replace=TRUE)

##' random drawing
b <- runif(100,min=0,max=1000)
Find.Optim.Stat(Stat='TSS',
                Fit=b,
                Obs=a)

##' 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)))
}

c <- sapply(a,BiasedDrawing)

Find.Optim.Stat(Stat='TSS',
                Fit=c,
                Obs=a,
                Nb.thresh.test = 100)
```

---

full_suffling *data set shuffling tool*

Description

This function is developper tool to shuffle elegantly data-set. This function suffle one column of a given data.set

Usage

```
full_suffling(x,id=NULL)
```

Arguments

x	a data.frame you want to suffle
id	the columms you want to shuffle

Value

a data.frame with selected columns suffled compared to the original table.

Author(s)

Damien Georges

See Also

variables_importance

Examples

```
xx <- matrix(rep(1:10,3),10,3)
full_suffling(xx,c(1,2))
```

---

getStatOptimValue *get the optimal score of evaluation statistical metrics*

Description

getStatOptimValue is an internal **biomod2** function to get the best value that some referenced evaluation statistical metrics.

Usage

```
getStatOptimValue(stat)
```

Arguments

stat	either 'TSS', 'KAPPA', 'ACCURACY', 'BIAS', 'POD', 'FAR', 'POFD', 'SR', 'CSI', 'ETS', 'HK', 'HSS', 'OR' or 'ORSS'

Details

Please refer to BIOMOD_Modeling to get more information about this metrics.

Value

The best value that stat could rise.

Author(s)

Damien Georges

See Also

BIOMOD_Modeling, calculate.stat, Find.Optim.Stat

Examples

```
getStatOptimValue('TSS')
getStatOptimValue('KAPPA')
getStatOptimValue('POFD')
```

`level.plot` *Plot 2-dimensional data for visualizing distribution of species or environment*

Description

Enables to plot data with 2-dimensional geographical coordinates

Usage

```
level.plot( data.in,
            XY,
            color.gradient = "red",
            cex = 1,
            level.range = c(min(data.in), max(data.in)),
            show.scale = TRUE,
            title = "level plot",
            SRC=FALSE,
            save.file="no",
            ImageSize="small",
            AddPresAbs=NULL,
            PresAbsSymbol=c(cex*0.8,16,4),
            ...)
```

Arguments

`data.in`	the data you want to visualize
`XY`	a 2 columns matrix of the same length as data.in giving the coordinates of the data points
`color.gradient`	available : red, grey and blue
`cex`	to change the point size : >1 will increase size, <1 will decrease it
`level.range`	the range of values for which you want the color gradient to be used, usefull to increase the resolution of the graph especially if you have extreme values (see examples section)
`show.scale`	a feature for just keeping the graph without the scale (if set to False)
`title`	the title wanted for the graph
`SRC`	if TRUE, the function recognizes the values as being 'Species Range Change' (see XXX) values and associates the appropriate colors to them
`save.file`	can be set to "pdf", "jpeg" or "tiff" to save the plot. Pdf options can be changed by setting the default values of pdf.options().

ImageSize	The image size for JPEG and TIFF files saved to disk. Available : 'small', 'standard' and 'large'
AddPresAbs	Optional: adds the presences and absences used for calibration to the plot. Data must be entered as a matrix/dataframe with 3 columns (in this order): X-coordinate, Y-coordinate, Presence(1) or Absence(0). X and Y coordinates must be in the same system as the plot.
PresAbsSymbol	Optional: a 3 element vector giving the symbols to be used by the AddPresAbs argument for plotting. The elements of the vector must be in this order: size of presence/absence symbols given as a multiplication factor of the 'cex' value entered in the function (e.g. a value of 0.5 means that the symbols will be drawn at a size = 0.5*cex value entered in the function), symbol (in PCH code) to be used for presences, symbol (in PCH code) to be used for absences. An example of input vector for this parameter is 'c(0.4,16,4)'
...	extra args

Author(s)

Bruno Lafourcade

See Also

multiple.plot

Examples

```
## Not run:
# species occurrences
DataSpecies <- read.csv(system.file("external/species/mammals_table.csv",
                                    package="biomod2"), row.names = 1)

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

# the presence/absences data for our species
myResp <- as.numeric(DataSpecies[,myRespName])

# the XY coordinates of species data
myRespXY <- DataSpecies[,c("X_WGS84","Y_WGS84")]


level.plot(data.in=myResp, XY=myRespXY)

## End(Not run)
```

`makeFormula` *Standardized formula maker*

Description

makeFormula is an internal **biomod2** function that can be useful to help users to build easily some standardized formula used later by statistical models.

Usage

```
makeFormula(respName,
            explVar,
            type = 'simple',
            interaction.level = 0,
            ...)
```

Arguments

`respName`	a `character` indicating the response variable name
`explVar`	a `matrix` or a `data.frame`, the explanatory variables table that will be considered at modelling step
`type`	either 'simple', 'quadratic', 'polynomial' or 's_smoother' defining the type of formula you want to build
`interaction.level`	an `integer`, the interaction level depth between explanatory variables
`...`	some additional arguments (see details)

Details

It's advise to give only a subset of `explVar` table to avoid useless memory consuming. If some explanatory variables are factorial ones, you have to give a `data.frame` for `explVar` where associated columns are define as `factor`.

`...` argument availables values are :

- k the smooting parameter value (used only if `type = 's_smoother'`) corespunding to `k` parameter of **mgcv** `s` or `df` **gam** `s` arguments.

Value

A `link[stats]{formula}` class object that can be directly given to most of R statistical models.

Author(s)

Damien Georges

See Also

BIOMOD_ModelingOptions, link[stats]{formula}

Examples

```
# create simulated data
myResp <- sample(c(0,1),20, replace=TRUE)
myExpl <- matrix(runif(60), ncol=3, dimnames=list(NULL,c('var1','var2','var3')))

# create a formula
myFormula <- makeFormula( respName = 'myResp',
                          explVar = head(myExpl),
                          type = 'quadratic',
                          interaction.level = 0)
# show formula created
myFormula
```

---

models_scores_graph *Produce models evaluation bi-dimensional graph*

Description

This function is a graphic tool to represent evaluation scores of models produced with biomod2 according to 2 different evaluation methods. Models can be grouped in several ways (by algo, by CV run, ...) to highlight potential differences in models quality due to chosen models, cross validation sampling bias,... Each point represents the average evaluation score across each group. Lines represents standard deviation of evaluation scores of the group.

Usage

```
models_scores_graph(obj,
                    metrics = NULL,
                    by = 'models',
                    plot = TRUE,
                    ... )
```

Arguments

obj	a "BIOMOD.models.out" (returned by BIOMOD_Modeling) or a "BIOMOD.EnsembleModeling.out" (returned by BIOMOD_EnsembleModeling)
metrics	character vector of 2 chosen metrics (e.g c("ROC", "TSS")); if not filled the two first evaluation methods computed at modeling stage will be selected.
by	character ('models'), the way evaluation scores are grouped. Should be one of 'models', 'algos', 'CV_run' or 'data_set' (see detail section)
plot	logical (TRUE), does plot should be produced
...	additional graphical arguments (see details)

Details

by argument description :

by arg refers to the way models scores will be combined to compute mean and sd. It should take the following values:

- models : group evaluation scores according to top level models. Top level models should be for instance GLM, GAM, RF, SRE... in "BIOMOD.models.out" input case whereas it should be EMcaByTSS (committee averaging using TSS score), EMwmeanByROC (weighted mean using ROC scores),... or whatever in "BIOMOD.EnsembleModeling.out" input case.
- algos : If you work with "BIOMOD.models.out" then algos is equivalent to models. If you work with "BIOMOD.EnsembleModeling.out" then it refer to formal models i.e. GLM, GAM, RF, SRE... (should also be mergedAlgo in this case).
- cv_run : the cross validation run e.g. run1, run2,... Should be mergedRun in EnsembleModels input case.
- data_set : the data set (resp. pseudo absences data set) used to group scores e.g PA1, PA2,... if pseudo absences sampling have been computed or AllData inf not. Should also be mergedData in EnsembleModels case.

Additional arguments (...) :

Additional graphical parameters should be.

- xlim the graphical range represent for the first evaluation metric
- ylim the graphical range represent for the second evaluation metric
- main main plot title

Value

A ggplot2 plotting object is return. It means that user should then easily customize this plot (see example)

Note

This function have been instigate by *Elith*, J., H. Graham*, C., P. Anderson, R., Dudik, M., Ferrier, S., Guisan, A., J. Hijmans, R., Huettmann, F., R. Leathwick, J., Lehmann, A., Li, J., G. Lohmann, L., A. Loiselle, B., Manion, G., Moritz, C., Nakamura, M., Nakazawa, Y., McC. M. Overton, J., Townsend Peterson, A., J. Phillips, S., Richardson, K., Scachetti-Pereira, R., E. Schapire, R., Soberon, J., Williams, S., S. Wisz, M. and E. Zimmermann, N. (2006), Novel methods improve prediction of species distributions from occurrence data. Ecography, 29: 129-151. doi: 10.1111/j.2006.0906-7590.04596.x* (fig 3)

Author(s)

Damien Georges

See Also

BIOMOD_Modeling, BIOMOD_EnsembleModeling

Examples

```
## this example is based on BIOMOD_Modeling function example
example(BIOMOD_Modeling)

## plot evaluation models score graph

### by models
gg1 <- models_scores_graph( myBiomodModelOut,
                            by = 'models',
                            metrics = c('ROC','TSS') )
## we see a influence of model selected on models capabilities
## e.g. RF are much better than SRE

### by cross validation run
gg2 <- models_scores_graph( myBiomodModelOut,
                            by = 'cv_run',
                            metrics = c('ROC','TSS') )
## there is no difference in models quality if we focus on
## cross validation sampling

### some graphical customisations
gg1_custom <-
  gg1 +
  ggtitle("Diff between RF and SRE evaluation scores") + ## add title
  scale_colour_manual(values=c("green", "blue")) ## change colors

gg1_custom
```

`multiple.plot` *Plot and compare prediction maps within BIOMOD*

Description

This function allows a direct comparison of the predictions across models. The maps produced are the same than with the `level.plot` function

Usage

```
multiple.plot(Data,
              coor,
              color.gradient='red',
              plots.per.window=9,
              cex=1,
              save.file="no",
              name="multiple plot",
              ImageSize = "small",
              AddPresAbs = NULL,
              PresAbsSymbol = c(cex * 0.8, 16, 4))
```

Arguments

Data	the data to visualise
coor	a 2 columns matrix of the same length as data.in giving the coordinates of the data points
color.gradient	available : 'red', 'grey' and 'blue'
plots.per.window	the number of plots to visualise per graphic window
cex	to change the point size : >1 will increase size, <1 will decrease it
save.file	can be set to "pdf", "jpeg" or "tiff" to save the plot. Pdf options can be changed by setting the default values of pdf.options().
name	the name of the file produced if save.file is different to "no" (extensions are already included)
ImageSize	The image size for JPEG and TIFF files saved to disk. Available : 'small', 'standard' and 'large'
AddPresAbs	Optional: adds the presences and absences used for calibration to the plot. Data must be entered as a matrix/dataframe with 3 columns (in this order): X-coordinate, Y-coordinate, Presence(1) or Absence(0). X and Y coordinates must be in the same system as the plot.
PresAbsSymbol	Optional: a 3 element vector giving the symbols to be used by the AddPresAbs argument for plotting. The elements of the vector must be in this order: size of presence/absence symbols given as a multiplication factor of the 'cex' value entered in the function (e.g. a value of 0.5 means that the symbols will be drawn at a size = 0.5*cex value entered in the function), symbol (in PCH code) to be used for presences, symbol (in PCH code) to be used for absences. An example of input vector for this parameter is 'c(0.4,16,4)'

Author(s)

Wilfried Thuiller, Bruno Lafourcade

See Also

level.plot

Examples

```
## Not run:
# species occurrences
DataSpecies <- read.csv(system.file("external/species/mammals_table.csv",
                                    package="biomod2"), row.names = 1)

# the name of studied species
myRespName <- c("ConnochaetesGnou", "GuloGulo", "PantheraOnca",
                "PteropusGiganteus", "TenrecEcaudatus", "VulpesVulpes")

# the presence/absences data for our species
myResp <- DataSpecies[,myRespName]
```

```
# the XY coordinates of species data
myRespXY <- DataSpecies[,c("X_WGS84","Y_WGS84")]

multiple.plot(Data = myResp,
              coor = myRespCoord )

## End(Not run)
```

`Print_Default_ModelingOptions` *Get default values of BIOMOD inner models' options*

Description

This function print the default parameters used to build each model.

Usage

```
Print_Default_ModelingOptions()
```

Details

This function is useful to parameterize the selected models. It gives a formated list of all parameters the user can modify. You can copy this function output, modify all parameters you want in a text editor and paste the modified string as argument to (BIOMOD_ModelingOptions) function.

Value

Nothing returned.

Author(s)

Wilfried Thuiller, Damien Georges

Examples

```
# print default models options
Print_Default_ModelingOptions()
```

ProbDensFunc *Probability Density Function*

Description

Using a variety of parameters in modelling will inevitably bring variability in predictions, especially when it comes to making future predictions. This function enables an overall viewing of the future predictions range per species and gives the likelihood of range shift estimations. It will calculate the optimal way for condensing a difined proportion (50, 75, 90 and 95% per default) of the data.

Usage

```
ProbDensFunc(initial,
             projections,
             groups = NULL,
             plothist = TRUE,
             cvsn = TRUE,
             resolution = 5,
             filename = NULL, ...)
```

Arguments

initial	a vector (resp. a SpatialPointsDataFrame) in a binary format (ones and zeros) representing the current distribution of a species which will be used as a reference for the range change calculations
projections	a matrix (resp; a rasterStack) grouping all the predictions where each column is a single prediction. Make sure you keep projections in the same order as the initial vector (line1=site1, line2=site2, etc.).
plothist	set to TRUE to plot the range change histogram
cvsn	stands for "current vs new". If true, the range change calculations will be of two types: the percentage of cells currently occupied by the species to be lost, and the relative percentage of cells currently unoccupied but projected to be, namely 'new' cells, compared to current surface range.
groups	an option for ungrouping the projections enabling a separated visualisation of the prediction range per given group. A matrix is expected where each column is a single prediction and each line is giving details of one parameter (See the examples section).
resolution	the step used for classes of prediction in graphics. The default value is 5
filename	the name of file (with extension) where plots will be stored. If not NULL, no ploting windows will be open
...	futher arguments: • lim: ordered numeric vector indicating the proportion of data to consider for histogramme representation (by default : c(0.5,0.75,0.9,0.95)) • nb.points.max: the maximum number of points to sample, 25000 by default (usefull for huge raster* objects)

Details

The future range changes are calculated as a percentage of the species' present state. For example, if a species currently occupies 100 cells and is estimated by a model to cover 120 cells in the future, the range change will be + 20%.

Resolution : Note that modifying the resolution will directly influence the probability scale. Bigger classes will cumulate a greater number of predictions and therefore represent a greater fraction of the total predictions. The probability is in fact that of the class and not of isolated events.

Value

This is a plotting function, no objects are returned or created.

Author(s)

Wilfried Thuiller, Bruno Lafourcade, Damien Georges

See Also

BIOMOD_Projection, BIOMOD_EnsembleForecasting

Examples

```
## Not run:
DataSpecies <- read.csv(system.file("external/species/mammals_table.csv",
                                    package="biomod2"), row.names = 1)
head(DataSpecies)

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

# the presence/absences data for our species
myResp <- as.numeric(DataSpecies[,myRespName])

# remove all 0 from response vector to work with
# presence only data (Pseudo Absences selections)
rm_id <- which(myResp==0)
myResp <- myResp[-rm_id]


# the XY coordinates of species data
myRespXY <- DataSpecies[-rm_id,c("X_WGS84","Y_WGS84")]


# Environmental variables extracted from BIOCLIM
myExpl = stack( system.file( "external/bioclim/current/bio3.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio4.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio7.grd",
                             package="biomod2"),
```

```
                system.file( "external/bioclim/current/bio11.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio12.grd",
                             package="biomod2"))

# 1. Formatting Data
myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
                                     expl.var = myExpl,
                                     resp.xy = myRespXY,
                                     resp.name = myRespName,
                                     PA.nb.rep=3)

# 2. Defining Models Options using default options.
myBiomodOption <- BIOMOD_ModelingOptions()

# 3. Doing Modelisation
myBiomodModelOut <- BIOMOD_Modeling( myBiomodData,
                                     models = c('CTA','RF','GLM','GAM','ANN','MARS'),
                                     models.options = myBiomodOption,
                                     NbRunEval=5,
                                     DataSplit=70,
                                     Prevalence=0.5,
                                     models.eval.meth = c('TSS'),
                                     do.full.models = FALSE,
                                     rescal.all.models=T,
                                     modeling.id='test')

# 4. Build ensemble-models that will be taken as reference
myBiomodEM <- BIOMOD_EnsembleModeling( modeling.output = myBiomodModelOut,
                                       chosen.models = 'all',
                                       em.by = 'all',
                                       eval.metric = c('TSS'),
                                       eval.metric.quality.threshold = c(0.7),
                                       prob.mean = TRUE,
                                       prob.median = TRUE)

# 5. Projection on future environmental conditions

## load future environmental conditions from biomod2 package
myExpl_fut <- stack( system.file( "external/bioclim/future/bio3.grd",
                                  package="biomod2"),
                     system.file( "external/bioclim/future/bio4.grd",
                                  package="biomod2"),
                     system.file( "external/bioclim/future/bio7.grd",
                                  package="biomod2"),
                     system.file( "external/bioclim/future/bio11.grd",
                                  package="biomod2"),
                     system.file( "external/bioclim/future/bio12.grd",
                                  package="biomod2"))

myBiomodProjection <- BIOMOD_Projection(modeling.output = myBiomodModelOut,
                                        new.env = myExpl_fut,
                                        proj.name = 'future',
```

```
                                        selected.models = 'all',
                                        binary.meth = 'TSS',
                                        compress = FALSE,
                                        build.clamping.mask = TRUE)

BIOMOD_EnsembleForecasting(projection.output=myBiomodProjection,
                           EM.output=myBiomodEM,
                           binary.meth='TSS')

# 6. load binary projections
consensusBin <- stack('GuloGulo/proj_future/proj_future_GuloGulo_ensemble_TSSbin.grd')
projectionsBin <- stack('GuloGulo/proj_future/proj_future_GuloGulo_TSSbin.grd')

# 7. build a ref state based on ensemble-models
ref <- sampleRandom(subset(consensusBin, 1, drop=T), size=5000, sp=T, na.rm=T)

# 8. autoatic creation of groups matrix
find_groups <- function(diff_by_pix){
  data.set <- sapply(names(diff_by_pix),biomod2:::.extractModelNamesInfo,info='data.set')
  run.eval <- sapply(names(diff_by_pix),biomod2:::.extractModelNamesInfo,info='run.eval')
   models <- sapply(names(diff_by_pix),biomod2:::.extractModelNamesInfo,info='models')
   return(rbind(data.set,run.eval,models))
}

groups <- find_groups(projectionsBin)

# 9. plot ProbDensFunct graphs
ProbDensFunc(initial = ref,
             projections = projectionsBin,
             plothist=TRUE,
             cvsn=TRUE,
             groups=groups,
             resolution=2,
             filename=NULL,
             lim=c(0.5,0.8,0.95))

## 3 plots should be produced.. Should be convenient to save it within a device
## supporting multiple plots.


## End(Not run)
```

`randomise_data` *data set shuffling tool*

Description

This function is wrapper to shuffle elegantly data-set. This function shuffle selected columns of a given data.set according to a defined shuffling algorithm.

Usage

```
randomise_data(data,variable,method)
```

Arguments

data	a data.frame you want to shuffle
variable	the variables names or columns id you want to shuffle
method	the randomisation method (only 'full_rand' available yet)

Value

a data.frame with selected columns shuffled compared to the original table.

Author(s)

Damien Georges

See Also

variables_importance, full_suffling

Examples

```
xx <- data.frame(a=1:10,b=11:20,c=21:30)
randomise_data(data=xx, variable='b', method='full_rand')
```

---

response.plot *Analysis of the response curves of a model within Biomod*

Description

Depreciated function, please use response.plot2 instead

Usage

```
response.plot(model,
              Data,
              show.variables=seq(1:ncol(Data)),
              save.file="no",
              name="response_curve",
              ImageSize=480,
              plot=TRUE)
```

Arguments

model	the model for which you want the response curves to be plotted. Compatible with GAM, GBM, GLM, ANN, CTA, RF, FDA and MARS.
Data	the variables for which you want the response curves to be plotted. A data frame is wanted with one column per vriable. They have to have the same names as the ones used to calibrate the model.
show.variables	give in the column numbers of 'Data' for selecting the variables that are wanted for plotting
save.file	can be set to "pdf", "jpeg" or "tiff" to save the plot. Pdf options can be changed by setting the default values of pdf.options().
name	the name of the file produced if save.file is different to "no" (extensions are already included)
ImageSize	the size of the image in pixels if save.file is different to "no". Affects "jpeg" and "tiff" outputs only. Default if 480 pixels which is the R default.
plot	if TRUE (the default) then a plot is produced. If not, an array containing predictions is returned (see details)

Details

Depreciated function, please use `response.plot2` instead.

Author(s)

Wilfried Thuiller

References

Elith, J., Ferrier, S., Huettmann, FALSE. & Leathwick, J. R. 2005 The evaluation strip: A new and robust method for plotting predicted responses from species distribution models. Ecological Modelling 186, 280-289.

---

response.plot2 *Function for for plotting predicted responses from species distribution models in 2 or 3 dimensions*

Description

Adaptation of the Evaluation Strip method proposed by Elith et al.(2005). This function enables to plot the response curves of a model independently of the algorithm used for building the model. It therefore permits a direct comparison of predicted responses from the different statistical approaches on the same data.

Usage

```
response.plot2( models,
                Data,
                show.variables=seq(1:ncol(Data)),
                do.bivariate = FALSE,
                fixed.var.metric = 'mean',
                save.file="no",
                name="response_curve",
                ImageSize=480,
                plot=TRUE,
                ...)
```

Arguments

models	a character vector specifying the models for which the response curves have to be plotted. Compatible with GAM, GBM, GLM, ANN, CTA, RF, FDA, MARS and MAXENT.
Data	a dataframe containing the variables for which the response curves have to be plotted. They must have the same names as the ones used to calibrate the model. RasterStack are also supported.
show.variables	the names or the column numbers of 'Data' for selecting the variables to be plotted. By default all columns are taken
do.bivariate	'logical', if FALSE (default), the predicted response curves are plotted for every singe variable independently (2-D dimension). If TRUE, the predicted response curves are represented in 3-D for all pairs of variables
fixed.var.metric	either 'mean' (default), 'median', 'min' or 'max' specifying the statistic used to fix as constant the remaining variables when the predicted response is estimated for one of the variables
save.file	can be set to "pdf", "jpeg" or "tiff" to save the plot. Pdf options can be changed by setting the default values of pdf.options().
name	the name of the file produced if save.file is different to "no" (extensions are already included)
ImageSize	the size of the image in pixels if save.file is different to "no". Affects "jpeg" and "tiff" outputs only. Default if 480 pixels which is the R default.
plot	if TRUE (the default) then a plot is produced
...	further arguments :

- data_species : vector containing data species occurances. Have to match with Data. If given, the statistic used to fix variables value will be calculated only over presences points. (Considered only if Data is under table format)
- col : vector of colors to be used (only for univariate case)
- lty : vector of lines types to be used
- main : character, the title of the graph (default one based on first model class is automatically produced if not refered)

- `display_title` : logical, display or not graph title
- `legend` : logical, add legend to plot (only for univariate case)

Details

For building the predicted response curves, n-1 variables are set constant to a fixed value (mean, median, min or max i.e `fixed.var.metric` arg) and only the remaining one (resp. 2 for 3D response plot) is varying across its whole range (given by `Data`). n the case of categorical variable, the most represented class is taken. The variations observed and the curve thus obtained shows the sensibility of the model to that specific variable. This method does not account for interactions between variables. In the evaluation strip initially proposed by Elith et al. 2005 the remaining variables are set to the mean.

Value

a 4D array is returned. It contains the necessary outputs to produce the plots. This is useful to make your own custom response plot graphics.

Array returned structure :

- First dimension: the dimension of the predictions
- Second dimension: 2 or 3 columns: The first one (resp. the first two) is (are) the explanatory variable(s) to plot, the last one, the probability of occurrence
- Third dimension: The set of environmental variables for which the response.plot was asked to run.
- Fourth dimension: the selected models

Author(s)

Wilfried Thuiller, Damien Georgs

References

Elith, J., Ferrier, S., Huettmann, FALSE. & Leathwick, J. R. 2005 The evaluation strip: A new and robust method for plotting predicted responses from species distribution models. Ecological Modelling 186, 280-289.

See Also

BIOMOD_Modeling

Examples

```
## Not run:
# species occurrences
DataSpecies <- read.csv(system.file("external/species/mammals_table.csv",
                                    package="biomod2"), row.names = 1)
head(DataSpecies)

# the name of studied species
```

```
myRespName <- 'VulpesVulpes'

# the presence/absences data for our species
myResp <- as.numeric(DataSpecies[,myRespName])

# the XY coordinates of species data
myRespXY <- DataSpecies[,c("X_WGS84","Y_WGS84")]


# Environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
myExpl = stack( system.file( "external/bioclim/current/bio3.grd",
                     package="biomod2"),
                system.file( "external/bioclim/current/bio4.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio7.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio11.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio12.grd",
                             package="biomod2"))

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

# 2. Defining Models Options using default options.
myBiomodOption <- BIOMOD_ModelingOptions()

# 3. Doing Modelisation

myBiomodModelOut <- BIOMOD_Modeling( myBiomodData,
                                       models = c('RF','GLM'),
                                       models.options = myBiomodOption,
                                       NbRunEval=2,
                                       DataSplit=80,
                                       Prevalence=0.5,
                                       VarImport=0,
                                       models.eval.meth = c('TSS'),
                                       SaveObj = TRUE,
                                       do.full.models=FALSE)


# 4. Plot response curves

# 4.1 Load the models for which we want to extract the predicted response curves
myGLMs <- BIOMOD_LoadModels(myBiomodModelOut, models='GLM')

# 4.2 plot 2D response plots
myRespPlot2D <- response.plot2(models  = myGLMs,
                               Data = get_formal_data(myBiomodModelOut,'expl.var'),
                         show.variables= get_formal_data(myBiomodModelOut,'expl.var.names'),
```

```
                                do.bivariate = FALSE,
                                fixed.var.metric = 'median',
                                col = c("blue", "red"),
                                legend = TRUE,
                              data_species = get_formal_data(myBiomodModelOut,'resp.var'))

# 4.2 plot 3D response plots
## here only for a lone model (i.e "VulpesVulpes_PA1_AllData_GLM")
myRespPlot3D <- response.plot2(models  = myGLMs[1],
                                Data = get_formal_data(myBiomodModelOut,'expl.var'),
                            show.variables= get_formal_data(myBiomodModelOut,'expl.var.names'),
                                do.bivariate = TRUE,
                                fixed.var.metric = 'median',
                              data_species = get_formal_data(myBiomodModelOut,'resp.var'),
                                display_title=FALSE)

### all the values used to produce this plot are stored into the returned object
### you can redo plots by yourself and customised them
dim(myRespPlot2D)
dimnames(myRespPlot2D)

dim(myRespPlot3D)
dimnames(myRespPlot3D)

## End(Not run)
```

`sample.factor.levels` *Tool to ensure the sampling of all levels of a factorial variable*

Description

This function will sample randomly an element of each level of all the factorial variables contains in a Raster* object or a data.frame

Usage

```
sample.factor.levels(x, mask.out = NULL, mask.in = NULL)
```

Arguments

`x`	a Raster* object or a data.frame
`mask.out`	a Raster/data.frame mask containing area that have already been sampled. The factor levels within this mask will not be sampled.
`mask.in`	a Raster/list of Raster/data.frame mask (potentially a stack of masks) indicating areas were we want to sample our factor level in priority. Note that if after having explore this masks some levels of the considered factorial varialble remains unsampled, this levels will be sampled in the reference input object (here 'x')

Details

In case any factorial variable is found in the input object then NULL is returned.

Value

a numeric vector the number (cell number for Raster* objects or row number for data.frame) where each will refer to a single level of a single factorial variable.

Note

- The x/mask.out/mask.in should be coherent in term of dimention (same number of rows for data.frame and same number of rows, column, identic resolution and projection coordinates system for Raster* objects) - If mask.in contains several masks (RasterStack or multi-column data.frame) then the order of the mask matter. The mask will be considered successively. The first will be use prioritarly to sample our variable factor levels and so on. - Raster* masks will be understood as: - NA: out of of mask - not NA: in mask - data.frame masks will be understood as: - FALSE: out of mask - TRUE: in mask

Author(s)

damien g.

Examples

```
## example with raster* object ----------
## create a factorial raster
r1 <- raster()
r1[] <- 1; r1[1] <- 2; r1[2:3] <- 3
r1 <- as.factor(r1)
## create a continuous raster
r2 <- raster()
r2[] <- rnorm(ncell(r2))
## pull the raster into a RasterStack
stk <- stack(r1, r2)
is.factor(stk)

## define a mask for already sampled points
mask.out <- r1
mask.out[] <- NA; mask.out[2:3] <- 1

## define a list of mask where we want to sample in priority
mask.in.1 <- mask.in.2 <- r1
mask.in.1[1:10] <- NA ## only level 1 should be sampled in this mask
mask.in.2[1] <- NA ## only levels 1 and 3 should be sampled in this mask
mask.in <- list(mask.in.1 = mask.in.1,
                mask.in.2 = mask.in.2)

## test different version of the function
sample.factor.levels(stk, mask.out = mask.out)
sample.factor.levels(stk, mask.in = mask.in)
sample.factor.levels(stk, mask.out = mask.out, mask.in = mask.in)
```

SampleMat2 *Sample binary vector*

Description

SampleMat2 is an internal **biomod2** function that can help user to sample a binary vector keeping the same proportion of 0s and 1s than in the initial vector.

Usage

```
SampleMat2(ref, ratio, as.logi=FALSE)
```

Arguments

ref	a binary vector
ratio	the proportion of ref to sample
as.logi	logical, if FALSE (default) id of cell will be return; if TRUE, logical vector of same length than ref will be return

Details

This function can be useful to help users to select a part of initial dataset that will be only kept for all validation procedures.

Value

A list of 2 elements is returned :

- calibration Ids of cells selected for calibration (1st sample)
- evaluation Ids of cells selected for evaluation (1st sample complementary)

Author(s)

Damien Georges

See Also

BIOMOD_FormatingData

Examples

```
a <- sample(c(0,1),100, replace=TRUE)
SampleMat2(ref=a, ratio=0.7)
```

sre *Surface Range Envelope*

Description

Run a rectilinear surface range envelop (equivalent to BIOCLIM) using the extreme percentiles as recommended by Nix or Busby. The SRE performs a simple analysis of within which range of each variable the data is recorded and renders predictions.

Usage

```
sre(Response = NULL,
    Explanatory = NULL,
    NewData = NULL,
    Quant=0.025,
    return_extremcond = FALSE)
```

Arguments

Response	a vector (resp. a matrix/data.frame, a SpatialPointsDataFrame or a RasterLayer) giving your species' presences and absences data
Explanatory	a matrix (resp. a data.frame, a SpatialPointsDataFrame or a RasterStack) containing the environmental variables for the sites given in Response. It must have as many rows as there are elements in Response.
NewData	The data for which you want to render predictions with the sre. It must be a matrix (resp. a data.frame, a SpatialPointsDataFrame or a RasterStack) of the same type as the one given in Explanatory and with precisely the same variable names.
Quant	the value defines the most extreme values for each variable not to be taken into account for determining the tolerance boundaries for the considered species.
return_extremcond	boolean, if TRUE a matrix containing extrem conditions supported is returned

Details

The more variables you put in, the more restrictive your model will be (if non-colinear variables).

This method is very much influenced by the data input, and more specifically by the extremes. Where a linear model can discriminate the extreme values from the main tendency, the SRE considers it equal as any other data point which leads to notable changes in predictions. Note that, as a consequence of its functioning, the predictions are directly given in binary, a site being either potentially suitable for all the variables, either out of bounds for at least one variable and therefore considered unsuitable.

The quants argument determines the threshold at which the data will be taken into account for calibration : the default 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. So it in fact takes 5% away at each end of the variables distribution, giving a total of 10% of data not considered.

Value

A vector (resp. a RasterLayer) of the same length as there are rows in NewData giving the prediction in binary (1=presence, 0=absence)

Author(s)

Wilfried Thuiller, Bruno Lafourcade, Damien Georges

See Also

BIOMOD_Modeling, BIOMOD_ModelingOptions, BIOMOD_Projection

Examples

```
# species occurrences
DataSpecies <- read.csv(system.file("external/species/mammals_table.csv",
                                    package="biomod2"), row.names = 1)
head(DataSpecies)

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

# the presence/absences data for our species
myResp <- as.numeric(DataSpecies[,myRespName])

# the XY coordinates of species data
myRespXY <- DataSpecies[which(myResp==1),c("X_WGS84","Y_WGS84")]

# Environmental variables extracted from BIOCLIM (bio_3, bio_4, bio_7, bio_11 & bio_12)
myExpl = stack( system.file( "external/bioclim/current/bio3.grd",
                     package="biomod2"),
                system.file( "external/bioclim/current/bio4.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio7.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio11.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio12.grd",
                             package="biomod2"))

# we build a raster layer based on environmental rasters for our response variable
myResp <- reclassify(subset(myExpl,1,drop=TRUE), c(-Inf,Inf,0))
myResp[cellFromXY(myResp,myRespXY)] <- 1


# Compute some SRE for several quantile values
g <- sre(Response = myResp, Explanatory = myExpl, NewData=myExpl, Quant=0)
gg <- sre(Response = myResp, Explanatory = myExpl, NewData=myExpl, Quant=0.025)
ggg <- sre(Response = myResp, Explanatory = myExpl, NewData=myExpl, Quant=0.05)

# visualise results
```

```
par(mfrow=c(2,2),mar=c(6, 5, 5, 3))
plot(myResp, main=paste(myRespName,"original distrib."))
plot(g, main="full data calibration")
plot(gg, main="Perc025")
plot(ggg, main="Perc05")
```

`update_objects` *update biomod2 objects*

Description

This function is wrapper to update objects construct with a old version of biomod2 to a current one

Usage

```
update_objects(obj, recursive=TRUE)
```

Arguments

`obj`	a `biomod2` object you want to update (e.g. 'BIOMOD.formated.data', 'biomod2_model' object)
`recursive`	logical, if TRUE all objects on which updated object is based will be also updtaed

Details

This function will add/change/delete all object slots that have evolved between 2 versions of the package. If necessary, objects stored on hard drive will also be updated.

Value

the updated version of the biomod2 object is return

Author(s)

Damien Georges

See Also

`variables_importance`, `full_suffling`

Examples

```
## not done yet ##
```

variables_importance *Variables importance calculation*

Description

This function will return a variable importance value for each variable involved within your model.

Usage

```
variables_importance(model, data, method="full_rand", nb_rand=1, ...)
```

Arguments

model	the model you want to study variables importance (one of the models supported within biomod2, ensemble models are also supported)
data	the data.set on which you want to perform analyses
method	the randomisation method (only 'full_rand' available yet)
nb_rand	the number of permutation done for each variable
...	additional args (not implemented yet)

Details

It's more or less base on the same principle than randomForest variables importance algorithm. The principle is to shuffle a single variable of the given data. Make model prediction with this 'shuffled' data.set. Then we compute a simple correlation (Pearson's by default) between references predictions and the 'shuffled' one. The return score is 1-cor(pred_ref,pred_shuffled). The highest the value, the more influence the variable has on the model. A value of this 0 assumes no influence of that variable on the model. Note that this technique does not account for interactions between the variables.

Value

a list of class "BIOMOD_variables_importances" which contains:

- mata data.frame containing variables importance scores for each permutation run.

Author(s)

Damien Georges

See Also

randomise_data, full_suffling

Examples

```
xx <- data.frame( a=sample(c(0,1),100,replace=TRUE),
                  b=rnorm(100),
                  c=1:100)

mod <- glm(a~b+c, data=xx)

variables_importance(model=mod, data=xx[,c('b','c')], method="full_rand", nb_rand=3)
```

Index