

Package ‘SDMtune’

September 2, 2019

Type Package

Title Species Distribution Model Selection

Version 0.2.0

Date 2019-09-02

Description User-friendly framework that enables the training and the evaluation of species distribution models (SDMs). The package implements functions for data driven variable selection and model tuning and includes numerous utilities to display the results. All the functions used to select variables or to tune model hyperparameters have an interactive real-time chart displayed in the 'RStudio' viewer pane during their execution.

License GPL-3

URL <https://consbiol-unibern.github.io/SDMtune/>

BugReports <https://github.com/ConsBiol-unibern/SDMtune/issues>

Depends R (>= 3.2.0)

Imports cli (>= 1.1.0), crayon (>= 1.3.4), dismo (>= 1.1-4), gbm (>= 2.1.5), ggplot2 (>= 3.2.0), htmltools (>= 0.3.6), jsonlite (>= 1.6), kableExtra (>= 1.1.0), maxnet (>= 0.1.2), methods, nnet (>= 7.3-12), plotROC (>= 2.2.1), progress (>= 1.2.2), randomForest (>= 4.6-14), raster (>= 2.9-5), rasterVis (>= 0.45), Rcpp (>= 1.0.1), reshape2 (>= 1.4.3), rstudioapi (>= 0.10), scales (>= 1.0.0), stringr (>= 1.4.0), whisker (>= 0.3-2)

Encoding UTF-8

LazyData true

LinkingTo Rcpp

RoxygenNote 6.1.1

Suggests covr, knitr (>= 1.23), maps (>= 3.3.0), pkgdown (>= 1.3.0), rgdal (>= 1.4-4), rmarkdown (>= 1.13), roxygen2 (>= 6.1.1), snow (>= 0.4-3), spelling, testthat (>= 2.2.1), zeallot (>= 0.1.0)

VignetteBuilder knitr

Collate 'ANN-class.R' 'BRT-class.R' 'Maxent-class.R' 'Maxnet-class.R'
 'RF-class.R' 'RcppExports.R' 'SWD-class.R' 'SDMmodel-class.R'
 'SDMmodel2MaxEnt.R' 'SDMmodelCV-class.R' 'SDMtune-class.R'
 'SDMtune.R' 'aicc.R' 'auc.R' 'chart-utils.R' 'confMatrix.R'
 'convertFolds.R' 'corVar.R' 'doJk.R' 'getSubsample.R'
 'gridSearch.R' 'help-functions.R' 'maxentTh.R' 'maxentVarImp.R'
 'mergeSWD.R' 'modelReport.R' 'optimizeModel.R' 'plotCor.R'
 'plotJk.R' 'plotPA.R' 'plotPred.R' 'plotROC.R' 'plotResponse.R'
 'plotVarImp.R' 'predict-ANN.R' 'predict-BRT.R'
 'predict-Maxent.R' 'predict-RF.R' 'predict-SDMmodel.R'
 'prepareSWD.R' 'randomFolds.R' 'randomSearch.R' 'reduceVar.R'
 'swd2csv.R' 'thinData.R' 'thresholds.R' 'train.R' 'trainANN.R'
 'trainBRT.R' 'trainMaxent.R' 'trainMaxnet.R' 'trainRF.R'
 'trainValTest.R' 'tss.R' 'utils.R' 'varImp.R' 'varSel.R'
 'virtualSp.R' 'zzz.R'

Language en-US

NeedsCompilation yes

Author Sergio Vignali [aut, cre] (<<https://orcid.org/0000-0002-3390-5442>>),
 Arnaud Barras [aut] (<<https://orcid.org/0000-0003-0850-6965>>),
 Veronika Braunsch [aut] (<<https://orcid.org/0000-0001-7035-4662>>),
 Conservation Biology - University of Bern [fnd]

Maintainer Sergio Vignali <sergio.vignali@iee.unibe.ch>

Repository CRAN

Date/Publication 2019-09-02 13:10:03 UTC

R topics documented:

aicc	3
ANN-class	5
auc	5
BRT-class	7
confMatrix	8
corVar	9
doJk	10
getSubsample	12
get_tunable_args	13
gridSearch	14
Maxent-class	15
maxentTh	16
maxentVarImp	17
Maxnet-class	18
mergeSWD	19
modelReport	20
old2NewSDMmodel	21
old2NewSDMmodelCV	22
old2NewSDMtune	22

old2NewSWD	23
optimizeModel	23
plot,SDMtune,missing-method	25
plotCor	27
plotJk	28
plotPA	29
plotPred	30
plotResponse	31
plotROC	33
plotVarImp	34
predict,ANN-method	35
predict,BRT-method	36
predict,Maxent-method	36
predict,RF-method	37
predict,SDMmodel-method	38
prepareSWD	40
randomFolds	41
randomSearch	42
reduceVar	44
RF-class	45
SDMmodel-class	46
SDMmodel2MaxEnt	46
SDMmodelCV-class	47
SDMtune-class	48
SDMtune-pkg	48
SWD-class	49
swd2csv	49
thinData	50
thresholds	51
train	52
trainValTest	56
tss	57
varImp	59
varSel	60
virtualSp	63
Index	64

aicc

AICc

Description

Compute the Akaike Information Criterion corrected for small samples size (Warren and Seifert, 2011).

Usage

```
aicc(model, env, parallel = FALSE)
```

Arguments

`model` [SDMmodel](#) object.
`env` [stack](#) containing the environmental variables.
`parallel` logical, if TRUE it uses parallel computation, default is FALSE.

Details

The function is available only for **Maxent** and **Maxnet** methods. Parallel computation increases the speed only for large datasets due to the time necessary to create the cluster.

Value

The computed AICc

Author(s)

Sergio Vignali

References

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

Examples

```
# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                   pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare presence and background locations
p_coords <- virtualSp$presence
bg_coords <- virtualSp$background

# Create SWD object
data <- prepareSWD(species = "Virtual species", p = p_coords, a = bg_coords,
                  env = predictors, categorical = "biome")

# Train a model
model <- train(method = "Maxnet", data = data, fc = "1")

# Compute the AICc
aicc(model, predictors)
```

```
# Compute the AICc using parallel computation. This reduces the time only for
# large datasets, in this case it takes longer than the previous example due
# to the time used to start and stop a cluster
aicc(model, predictors, parallel = TRUE)
```

ANN-class

Artificial Neural Network

Description

This Class represents an Artificial Neural Network model objects and hosts all the information related to the model.

Details

See [nnet](#) for the meaning of the slots.

Slots

size integer. Number of the units in the hidden layer.
 decay numeric. Weight decay.
 rang numeric. Initial random weights.
 maxit integer. Maximum number of iterations.
 model [nnet](#). The randomForest model object.

Author(s)

Sergio Vignali

auc

AUC

Description

Compute the AUC using the Man-Whitney U Test formula.

Usage

```
auc(model, test = NULL, a = NULL)
```

Arguments

model [SDMmodel](#) or [SDMmodelCV](#) objects.
 test [SWD](#) test object for [SDMmodel](#) objects or logical for [SDMmodelCV](#) objects, if not provided it computes the train AUC, default is NULL.
 a Deprecated.

Details

If the model is a `SDMmodelCV` object, the function computes the mean of the training or testing AUC values of the different replicates.

Value

The value of the AUC.

Author(s)

Sergio Vignali

References

Mason, S. J. and Graham, N. E. (2002), Areas beneath the relative operating characteristics (ROC) and relative operating levels (ROL) curves: Statistical significance and interpretation. *Q.J.R. Meteorol. Soc.*, 128: 2145-2166.

Examples

```
# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                   pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare presence and background locations
p_coords <- virtualSp$presence
bg_coords <- virtualSp$background

# Create SWD object
data <- prepareSWD(species = "Virtual species", p = p_coords, a = bg_coords,
                  env = predictors, categorical = "biome")

# Split presence locations in training (80%) and testing (20%) datasets
datasets <- trainValTest(data, test = 0.2, only_presence = TRUE)
train <- datasets[[1]]
test <- datasets[[2]]

# Train a model
model <- train(method = "Maxnet", data = train, fc = "1")

# Compute the training AUC
auc(model)

# Compute the testing AUC
auc(model, test = test)

# Same example but using cross validation instead of training and testing
# datasets
# Create the folds
```

```

folds <- randomFolds(data, k = 4, only_presence = TRUE)
model <- train(method = "Maxnet", data = data, fc = "1", folds = folds)

# Compute the training AUC
auc(model)

# Compute the testing AUC
auc(model, test = TRUE)

```

BRT-class

Boosted Regression Tree

Description

This Class represents a Boosted Regression Tree model objects and hosts all the information related to the model.

Details

See [gbm](#) for the meaning of the slots.

Slots

`distribution` character. Name of the used distribution.

`n.trees` integer. Maximum number of grown trees.

`interaction.depth` integer. Maximum depth of each tree.

`shrinkage` numeric. The shrinkage parameter.

`bag.fraction` numeric. Random fraction of data used in the tree expansion.

`model` [gbm](#). The Boosted Regression Tree model object.

Author(s)

Sergio Vignali

confMatrix	<i>Confusion Matrix</i>
------------	-------------------------

Description

Computes Confusion Matrixes for threshold values varying from 0 to 1.

Usage

```
confMatrix(model, test = NULL, th = NULL, type = NULL)
```

Arguments

model	SDMmodel object.
test	SWD test locations, if not provided it uses the train dataset, default is NULL.
th	numeric vector, if provided it computes the evaluation at the given thresholds, default is NULL and it computes the evaluation for the unique predicted values at presence and absence/background locations.
type	character. The output type used for "Maxent" and "Maxnet" methods, possible values are "cloglog" and "logistic", default is NULL.

Details

- For models trained with the **Maxent** method the argument type can be: "raw", "logistic" and "cloglog".
- For models trained with the **Maxnet** method the argument type can be: "link", "exponential", "logistic" and "cloglog", see [maxnet](#) for more details.

Value

The Confusion Matrix for all the used thresholds.

Author(s)

Sergio Vignali

Examples

```
# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                  pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare presence and background locations
p_coords <- virtualSp$presence
bg_coords <- virtualSp$background
```



```

# Create SWD object
data <- prepareSWD(species = "Virtual species", p = p_coords, a = bg_coords,
                  env = predictors, categorical = "biome")

# Train a model
model <- train(method = "Maxnet", data = data, fc = "1")

# Get the confusion matrix for thresholds ranging from 0 to 1
cm <- confMatrix(model, type = "cloglog")
head(cm)
tail(cm)

# Get the confusion matrix for a specific threshold
confMatrix(model, type = "logistic", th = 0.6)

```

corVar

Print Correlated Variables

Description

Utility that prints the name of correlated variables and the relative correlation coefficient value.

Usage

```
corVar(bg, method = "spearman", cor_th = NULL, order = TRUE,
       remove_diagonal = TRUE)
```

Arguments

bg	SWD object with the locations used to compute the correlation between environmental variables.
method	character. The method used to compute the correlation matrix, default is "spearman".
cor_th	numeric. If provided it prints only the variables whose correlation coefficient is higher or lower than the given threshold, default is NULL.
order	logical, if TRUE the variable are ordered from the most to the less highly correlated, default is TRUE.
remove_diagonal	logical, if TRUE the values in the diagonal are, removed, default is TRUE.

Value

The name of the correlated variables.

Author(s)

Sergio Vignali

Examples

```
# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                  pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare background locations
bg_coords <- dismo::randomPoints(predictors, 10000)

# Create SWD object
bg <- prepareSWD(species = "Virtual species", a = bg_coords,
                env = predictors, categorical = "biome")

# Get the correlation among all the environmental variables
corVar(bg, method = "spearman")

# Get the environmental variables that have a correlation greater or equal to
# the given threshold
corVar(bg, method = "pearson", cor_th = 0.8)
```

doJk

Jackknife Test

Description

Run the Jackknife test for variable importance removing one variable at time.

Usage

```
doJk(model, metric, variables = NULL, test = NULL, with_only = TRUE,
     env = NULL, parallel = FALSE, return_models = FALSE)
```

Arguments

model	SDMmodel or SDMmodelCV object.
metric	character. The metric used to evaluate the models, possible values are: "auc", "tss" and "aicc".
variables	vector. Variables used for the test, if not provided it takes all the variables used to train the model, default is NULL.
test	SWD . If provided it reports the result also for the test dataset. Not used for aicc and SDMmodelCV .
with_only	logical. If TRUE it runs the test also for each variable in isolation, default is TRUE.
env	stack containing the environmental variables, used only with "aicc", default is NULL.
parallel	logical, if TRUE it uses parallel computation, default is FALSE. Used only with AICc.
return_models	logical, if TRUE returns all the models together with the test result, default is FALSE.

Details

Parallel computation increases the speed only for large datasets due to the time necessary to create the cluster.

Value

A data frame with the test results. If "return_model = TRUE" it returns a list containing the test results together with the models.

Author(s)

Sergio Vignali

Examples

```
# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                    pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare presence and background locations
p_coords <- virtualSp$presence
bg_coords <- virtualSp$background

# Create SWD object
data <- prepareSWD(species = "Virtual species", p = p_coords, a = bg_coords,
                  env = predictors, categorical = "biome")

# Split presence locations in training (80%) and testing (20%) datasets
datasets <- trainValTest(data, test = 0.2, only_presence = TRUE)
train <- datasets[[1]]
test <- datasets[[2]]

# Train a model
model <- train(method = "Maxnet", data = train, fc = "lq")

# Execute the Jackknife test only for the environmental variables "bio1" and
# "bio12", using the metric AUC
doJk(model, metric = "auc", variables = c("bio1", "bio12"), test = test)

# Execute the Jackknife test only for the environmental variables "bio1" and
# "bio12", using the metric TSS but without running the test for one single
# variable
doJk(model, metric = "tss", variables = c("bio1", "bio12"), test = test,
      with_only = FALSE)

# Execute the Jackknife test only for the environmental variables "bio1" and
# "bio12", using the metric AICc but without running the test for one single
# variable
doJk(model, metric = "aicc", variables = c("bio1", "bio12"),
      with_only = FALSE, env = predictors)
```

```
# Execute the Jackknife test for all the environmental variables using the
# metric AUC and returning all the trained models
jk <- doJk(model, metric = "auc", test = test, return_models = TRUE)
jk$results
jk$models_without
jk$models_withonly
```

getSubsample

Get Dataset Subsample, Deprecated

Description

Get a random subsample of an [SWD](#) object. This function is deprecated, don't use it with the an [SWD](#) object that contains presence and absence/background locations together.

Usage

```
getSubsample(dataset, size, seed = NULL)
```

Arguments

dataset	SWD object.
size	numeric. The size of the sub sample.
seed	numeric. The value used to set the seed in order to have consistent results, default is NULL.

Value

The sub sample as [SWD](#) object.

Author(s)

Sergio Vignali

Examples

```
# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                  pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare background locations
bg_coords <- dismo::randomPoints(predictors, 5000)

# Create SWD object
bg <- prepareSWD(species = "Virtual species", a = bg_coords, env = predictors,
                categorical = "biome")
```

```
# Get a subsample
getSubsample(bg, size = 2000, seed = 25)
```

get_tunable_args *Get Tunable Arguments*

Description

Returns the name of all function arguments that can be tuned for a given model.

Usage

```
get_tunable_args(model)
```

Arguments

model [SDMmodel](#) or [SDMmodelCV](#) object.

Value

character vector

Author(s)

Sergio Vignali

Examples

```
# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                   pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare presence and background locations
p_coords <- virtualSp$presence
bg_coords <- virtualSp$background

# Create SWD object
data <- prepareSWD(species = "Virtual species", p = p_coords, a = bg_coords,
                  env = predictors, categorical = "biome")

# Train a Maxent model and get tunable hyperparameters
model <- train(method = "Maxnet", data = data, fc = "1")
get_tunable_args(model)

# Train a Maxnet model and get tunable hyperparameters
model <- train(method = "Maxent", data = data, fc = "1")
get_tunable_args(model)
```

gridSearch

*Grid Search***Description**

Given a set of possible hyperparameter values, the function trains models with all the possible combinations of hyperparameters.

Usage

```
gridSearch(model, hypers, metric, test = NULL, bg4test = NULL,
  env = NULL, parallel = FALSE, save_models = TRUE, seed = NULL)
```

Arguments

model	SDMmodel or codeSDMmodelCV object.
hypers	named list containing the values of the hyperparameters that should be tuned, see details.
metric	character. The metric used to evaluate the models, possible values are: "auc", "tss" and "aicc".
test	codeSWD object. Test dataset used to evaluate the model, not used with aicc and codeSDMmodelCV objects, default is NULL.
bg4test	Deprecated.
env	stack containing the environmental variables, used only with "aicc", default is NULL.
parallel	logical, if TRUE it uses parallel computation, default is FALSE.
save_models	logical, if FALSE the models are not saved and the output contains only a data frame with the metric values for each hyperparameter combination. Default is TRUE, set it to FALSE when there are many combinations to avoid R crashing for memory overload.
seed	Deprecated.

Details

To know which hyperparameters can be tune you can use the output of the function [get_tunable_args](#). Hyperparameters not included in the hypers argument take the value that they have in the passed model. Parallel computation increases the speed only for large datasets due to the time necessary to create the cluster.

Value

[codeSDMtune](#) object.

Author(s)

Sergio Vignali

See Also

[randomSearch](#) and [optimizeModel](#).

Examples

```
# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                  pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare presence and background locations
p_coords <- virtualSp$presence
bg_coords <- virtualSp$background

# Create SWD object
data <- prepareSWD(species = "Virtual species", p = p_coords, a = bg_coords,
                  env = predictors, categorical = "biome")

# Split presence locations in training (80%) and testing (20%) datasets
datasets <- trainValTest(data, test = 0.2, only_presence = TRUE)
train <- datasets[[1]]
test <- datasets[[2]]

# Train a model
model <- train(method = "Maxnet", data = train, fc = "1")

# Define the hyperparameters to test
h <- list(reg = 1:2, fc = c("lqp", "lqph"))

# Run the function using the AUC as metric
output <- gridSearch(model, hypers = h, metric = "auc", test = test)
output@results
output@models
# Order results by highest test AUC
head(output@results[order(-output@results$test_AUC), ])

# Run the function using the AICc as metric and without saving the trained
# models, helpful when numerous hyperparameters are tested to avoid memory
# problems
output <- gridSearch(model, hypers = h, metric = "aicc", env = predictors,
                  save_models = FALSE)
output@results
```

Maxent-class

Maxent

Description

This Class represents a MaxEnt model objects and hosts all the information related to the model.

Slots

results matrix. The result that usually MaxEnt provide as a csv file.
reg numeric. The value of the regularization multiplier used to train the model.
fc character. The feature class combination used to train the model.
iter numeric. The number of iterations used to train the model.
extra_args character. Extra arguments used to run MaxEnt.
lambdas vector. The lambdas parameters of the model.
coeff data.frame. The lambda coefficients of the model.
formula formula. The formula used to make prediction.
lpn numeric. Linear Predictor Normalizer.
dn numeric. Density Normalizer.
entropy numeric. The entropy value.
min_max data.frame. The minimum and maximum values of the continuous variables, used for clamping.

Author(s)

Sergio Vignali

maxentTh

MaxEnt Thresholds

Description

Returns the value of the thresholds generated by the MaxEnt software.

Usage

```
maxentTh(model)
```

Arguments

model [SDMmodel](#) object trained using the "Maxent" method.

Value

data.frame with the thresholds.

Author(s)

Sergio Vignali

See Also

[maxentVarImp](#)

Examples

```
# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                  pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare presence and background locations
p_coords <- virtualSp$presence
bg_coords <- virtualSp$background

# Create SWD object
data <- prepareSWD(species = "Virtual species", p = p_coords, a = bg_coords,
                  env = predictors, categorical = "biome")

# Train a Maxent model
model <- train(method = "Maxent", data = data, fc = "1")
maxentTh(model)
```

maxentVarImp

Maxent Variable Importance

Description

Shows the percent contribution and permutation importance of the environmental variables used to train the model.

Usage

```
maxentVarImp(model)
```

Arguments

model [SDMmodel](#) or [SDMmodelCV](#) object trained using the "Maxent" method.

Details

When an [SDMmodelCV](#) object is passed to the function, the output is the average of the variable importance of each model trained during the cross validation.

Value

A data frame with the variable importance.

Author(s)

Sergio Vignali

See Also[maxentTh](#)**Examples**

```
# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                  pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare presence and background locations
p_coords <- virtualSp$presence
bg_coords <- virtualSp$background

# Create SWD object
data <- prepareSWD(species = "Virtual species", p = p_coords, a = bg_coords,
                  env = predictors, categorical = "biome")

# Train a Maxent model
model <- train(method = "Maxent", data = data, fc = "1")
maxentVarImp(model)
```

Maxnet-class

Maxnet

Description

This Class represents a Maxnet model objects and hosts all the information related to the model.

Slots

`reg` numeric. The value of the regularization multiplier used to train the model.

`fc` character. The feature class combination used to train the model.

`model` maxnet. The maxnet model object.

Author(s)

Sergio Vignali

`mergeSWD`*Merge SWD Objects*

Description

Merge two [SWD](#) objects.

Usage

```
mergeSWD(swd1, swd2)
```

Arguments

`swd1` [SWD](#) object.
`swd2` [SWD](#) object.

Details

- In case the two [SWD](#) objects have different columns, only the common columns are used in the merged object.
- The [SWD](#) object is created in a way that the presence locations are always before than the absence/background locations.

Value

The merged [SWD](#) object.

Author(s)

Sergio Vignali

Examples

```
# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                   pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare presence and background locations
p_coords <- virtualSp$presence
bg_coords <- virtualSp$background

# Create SWD object
data <- prepareSWD(species = "Virtual species", p = p_coords, a = bg_coords,
                  env = predictors, categorical = "biome")

# Split presence locations in training (80%) and testing (20%) datasets
datasets <- trainValTest(data, test = 0.2, only_presence = TRUE)
```

```

train <- datasets[[1]]
test <- datasets[[2]]

# Merge the training and the testing datasets together
merged <- mergeSWD(train, test)

```

modelReport

Model Report

Description

Make a report that shows the main results.

Usage

```

modelReport(model, folder, test = NULL, type = NULL,
  response_curves = FALSE, only_presence = FALSE, jk = FALSE,
  env = NULL, clamp = TRUE, permut = 10)

```

Arguments

model	SDMmodel object.
folder	character. The name of the folder in which to save the output. The folder is created in the working directory.
test	SWD object with the test locations, default is NULL.
type	character. The output type used for "Maxent" and "Maxnet" methods, possible values are "cloglog" and "logistic", default is NULL.
response_curves	logical, if TRUE it plots the response curves in the html output, default is FALSE.
only_presence	logical, if TRUE it uses only the range of the presence location for the marginal response, default is FALSE.
jk	logical, if TRUE it runs the jackknife test, default FALSE.
env	stack . If provided it computes and adds a prediction map to the output, default is NULL.
clamp	logical for clumping during prediction, used for response curves and for the prediction map, default is TRUE.
permut	integer. Number of permutations, default is 10.

Details

The function produces a report similar to the one created by MaxEnt software.

Author(s)

Sergio Vignali

Examples

```
# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                   pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare presence and background locations
p_coords <- virtualSp$presence
bg_coords <- virtualSp$background

# Create SWD object
data <- prepareSWD(species = "Virtual species", p = p_coords, a = bg_coords,
                  env = predictors, categorical = "biome")

# Split presence locations in training (80%) and testing (20%) datasets
datasets <- trainValTest(data, test = 0.2, only_presence = TRUE)
train <- datasets[[1]]
test <- datasets[[2]]

# Train a model
model <- train(method = "Maxent", data = train, fc = "1")

# Create the report
modelReport(model, type = "cloglog", folder = "my_folder", test = test,
            response_curves = TRUE, only_presence = TRUE, jk = TRUE,
            env = predictors)
```

old2NewSDMmodel

Convert Old SDMmodel object

Description

This is an help function to convert SDMmodel objects created with an old version of **SDMtune** (v <= 0.2.0) into the new format.

Usage

```
old2NewSDMmodel(model, fold = NULL)
```

Arguments

model	old SDMmodel object.
fold	logical, used internally, don't use it directly.

Value

An `SDMmodel` object in the new format.

Author(s)

Sergio Vignali

old2NewSDMmodelCV *Convert Old SDMmodelCV object*

Description

This is an help function to convert SDMmodelCV objects created with an old version of **SDMtune** (v <= 0.2.0) into the new format.

Usage

```
old2NewSDMmodelCV(model)
```

Arguments

model old SDMmodelCV object.

Value

An [SDMmodelCV](#) object in the new format.

Author(s)

Sergio Vignali

old2NewSDMtune *Convert Old SDMtune object*

Description

This is an help function to convert SDMtune objects created with an old version of **SDMtune** (v <= 0.2.0) into the new format.

Usage

```
old2NewSDMtune(object)
```

Arguments

object old SDMtune object.

Value

An [SDMtune](#) object in the new format.

Author(s)

Sergio Vignali

old2NewSWD	<i>Convert Old SWD object</i>
------------	-------------------------------

Description

This is an help function to convert SWD objects created with an old version of **SDMtune** (v <= 0.2.0) into the new format.

Usage

```
old2NewSWD(p, a, fold = NULL)
```

Arguments

p	Old SWD object with presence or test locations.
a	Old SWD object with absence/background locations.
fold	logical, used internally, don't use it directly.

Value

An [SWD](#) object in the new format.

Author(s)

Sergio Vignali

optimizeModel	<i>Optimize Model</i>
---------------	-----------------------

Description

The function uses a Genetic Algorithm implementation to optimize the model hyperparameter configuration according to the chosen metric.

Usage

```
optimizeModel(model, hypers, metric, test = NULL, bg4test = NULL,
  pop = 20, gen = 5, env = NULL, parallel = FALSE,
  keep_best = 0.4, keep_random = 0.2, mutation_chance = 0.4,
  seed = NULL)
```

Arguments

model	SDMmodel or SDMmodelCV object.
hypers	named list containing the values of the hyperparameters that should be tuned, see details.
metric	character. The metric used to evaluate the models, possible values are: "auc", "tss" and "aicc".
test	SWD object. Test dataset used to evaluate the model, not used with aicc and SDMmodelCV objects, default is NULL.
bg4test	Deprecated.
pop	numeric. Size of the population, default is 20.
gen	numeric. Number of generations, default is 20.
env	stack containing the environmental variables, used only with "aicc", default is NULL.
parallel	logical, if TRUE it uses parallel computation, default is FALSE.
keep_best	numeric. Percentage of the best models in the population to be retained during each iteration, expressed as decimal number. Default is 0.4.
keep_random	numeric. Probability of retaining the excluded models during each iteration, expressed as decimal number. Default is 0.2.
mutation_chance	numeric. Probability of mutation of the child models, expressed as decimal number. Default is 0.4.
seed	numeric. The value used to set the seed to have consistent results, default is NULL.

Details

To know which hyperparameters can be tuned you can use the output of the function [get_tunable_args](#). Hyperparameters not included in the hypers argument take the value that they have in the passed model. Parallel computation increases the speed only for large datasets due to the time necessary to create the cluster. Part of the code is inspired by [this post](#).

Value

[SDMtune](#) object.

Author(s)

Sergio Vignali

See Also

[gridSearch](#) and [randomSearch](#).

Examples

```

# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                   pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare presence and background locations
p_coords <- virtualSp$presence
bg_coords <- virtualSp$background

# Create SWD object
data <- prepareSWD(species = "Virtual species", p = p_coords, a = bg_coords,
                  env = predictors, categorical = "biome")

# Split presence locations in training (80%) and testing (20%) datasets
datasets <- trainValTest(data, test = 0.2, only_presence = TRUE)
train <- datasets[[1]]
test <- datasets[[2]]

# Train a model
model <- train(method = "Maxent", data = train, fc = "l")

# Define the hyperparameters to test
h <- list(reg = 1:3, fc = c("lqp", "lqph", "lh"), iter = seq(300, 700, 100))

# Run the function using as metric the AUC
output <- optimizeModel(model, hypers = h, metric = "auc", test = test,
                       seed = 25)

output@results
output@models
output@models[[1]] # Best model

```

plot,SDMtune,missing-method

Plot SDMtune object

Description

Plot an [SDMtune](#) object. Use the interactive argument to create an interactive chart.

Usage

```

## S4 method for signature 'SDMtune,missing'
plot(x, title = "", interactive = FALSE)

```

Arguments

`x` [SDMtune](#) object.

`title` character. The title of the plot, by default is an empty string.

`interactive` logical, if TRUE plot an interactive chart, default is FALSE.

Value

If `interactive = FALSE` the function returns a [ggplot](#) object otherwise it returns an `SDMtuneChart` object that contains the path of the temporary folder where the necessary files to create the chart are saved. In both cases the objects are returned as invisible.

Author(s)

Sergio Vignali

Examples

```
# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                   pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare presence and background locations
p_coords <- virtualSp$presence
bg_coords <- virtualSp$background

# Create SWD object
data <- prepareSWD(species = "Virtual species", p = p_coords, a = bg_coords,
                  env = predictors, categorical = "biome")

# Split presence locations in training (80%) and testing (20%) datasets
datasets <- trainValTest(data, test = 0.2, only_presence = TRUE)
train <- datasets[[1]]
test <- datasets[[2]]

# Train a model
model <- train(method = "Maxnet", data = train, fc = "1")

# Define the hyperparameters to test
h <- list(reg = 1:5, fc = c("lqp", "lqph"))

# Run the gridSearch function using as metric the AUC
output <- gridSearch(model, hypers = h, metric = "auc", test = test)

# Plot the output
plot(output, title = "My experiment")

# Plot the interactive chart
p <- plot(output, title = "My experiment", interactive = TRUE)
# Print the temporary folder that stores the files used to create the chart
```

```
str(p)
```

plotCor	<i>Plot Correlation</i>
---------	-------------------------

Description

Plot a correlation matrix heat map with the value of the correlation coefficients according with the given method. If `cor_th` is passed then it prints only the coefficients that are higher or lower than the given threshold.

Usage

```
plotCor(bg, method = "spearman", cor_th = NULL)
```

Arguments

<code>bg</code>	codeSWD object used to compute the correlation matrix.
<code>method</code>	character. The method used to compute the correlation matrix, default is "spearman".
<code>cor_th</code>	numeric. If provided it prints only the coefficients that are higher or lower than the given threshold, default is NULL.

Value

A `ggplot` object.

Author(s)

Sergio Vignali

Examples

```
# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                   pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare background locations
bg_coords <- dismo::randomPoints(predictors, 10000)

# Create SWD object
bg <- prepareSWD(species = "Virtual species", a = bg_coords,
                 env = predictors, categorical = "biome")

# Plot heat map
plotCor(bg, method = "spearman")
```

```
# Plot heat map showing only values higher than given threshold
plotCor(bg, method = "spearman", cor_th = 0.8)
```

plotJk

Plot Jackknife Test

Description

Plot the Jackknife Test for variable importance.

Usage

```
plotJk(jk, type = c("train", "test"), ref = NULL)
```

Arguments

jk	data.frame with the output of the doJk function.
type	character, "train" or "test" to plot the result of the test on the train or testing dataset.
ref	numeric. The value of the chosen metric for the model trained using all the variables. If provided it plots a vertical line showing the reference value. Default is NULL.

Value

The [ggplot](#) object.

Author(s)

Sergio Vignali

Examples

```
# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                   pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare presence and background locations
p_coords <- virtualSp$presence
bg_coords <- virtualSp$background

# Create SWD object
data <- prepareSWD(species = "Virtual species", p = p_coords, a = bg_coords,
                  env = predictors, categorical = "biome")
```

```

# Split presence locations in training (80%) and testing (20%) datasets
datasets <- trainValTest(data, test = 0.2, only_presence = TRUE)
train <- datasets[[1]]
test <- datasets[[2]]

# Train a model
model <- train(method = "Maxnet", data = train, fc = "lq")

# Execute the Jackknife test for all the environmental variables using the
# metric AUC
jk <- doJk(model, metric = "auc", test = test)

# Plot Jackknife test result for training
plotJk(jk, type = "train", ref = auc(model))

#' # Plot Jackknife test result for testing
plotJk(jk, type = "test", ref = auc(model, test = test))

```

plotPA

Plot Presence Absence Map

Description

Plot a presence absence map using the given threshold.

Usage

```
plotPA(map, th, colors = NULL, hr = FALSE, filename = NULL,
       format = "GTiff", ...)
```

Arguments

map	raster object with the prediction.
th	numeric. The threshold used to convert the output in a presence/absence map.
colors	vector. Colors to be used, default is NULL and uses red and blue.
hr	logical, if TRUE produces an output with high resolution, default is FALSE.
filename	character, if provided the raster map is saved in a file, default is NULL.
format	character. The output format, see writeRaster for all the options, default is Geotiff.
...	Additional arguments, see writeRaster for all the options.

Value

A [ggplot](#) object.

Author(s)

Sergio Vignali

See Also

[plotPred](#)

Examples

```
map <- raster::raster(matrix(runif(400, 0, 1), 20, 20))
plotPA(map, th = 0.8)
# Custom colors
plotPA(map, th = 0.5, colors = c("#d8b365", "#018571"))
# Save the file
plotPA(map, th = 0.7, filename = "my_map", format = "ascii")
```

plotPred

Plot Prediction

Description

Plot Prediction output.

Usage

```
plotPred(map, lt = "", colorramp = NULL, hr = FALSE)
```

Arguments

map	raster object with the prediction.
lt	character. Legend title, default is an empty string.
colorramp	vector. A custom color ramp given as a vector of colors (see example), default is NULL and uses a color ramp similar to the original MaxEnt output.
hr	logical, if TRUE produces an output with high resolution, default is FALSE.

Value

A [ggplot](#) object.

Author(s)

Sergio Vignali

See Also

[plotPA](#)

Examples

```
map <- raster::raster(matrix(runif(400, 0, 1), 20, 20))
plotPred(map, lt = "Habitat suitability \ncloglog")
# Custom colors
plotPred(map, lt = "Habitat suitability",
          colorramp = c("#2c7bb6", "#ffffbf", "#d7191c"))
```

plotResponse	<i>Plot Response Curve</i>
--------------	----------------------------

Description

Plot the Response Curve of the given environmental variable.

Usage

```
plotResponse(model, var, type = NULL, only_presence = FALSE,
             marginal = FALSE, fun = mean, rug = FALSE, color = "red")
```

Arguments

model	SDMmodel or SDMmodelCV object.
var	character. Name of the variable to be plotted.
type	character. The output type used for "Maxent" and "Maxnet" methods, possible values are "cloglog" and "logistic", default is NULL.
only_presence	logical, if TRUE it uses only the range of the presence location for the marginal response, default is FALSE.
marginal	logical, if TRUE it plots the marginal response curve, default is FALSE.
fun	function used to compute the level of the other variables for marginal curves, default is mean.
rug	logical, if TRUE it adds the rug plot for the presence and absence/background locations, available only for continuous variables, default is FALSE.
color	The color of the curve, default is "red".

Details

Note that fun is not a character argument, you must use mean and not "mean".

Value

A [ggplot](#) object.

Author(s)

Sergio Vignali

Examples

```
# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                   pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare presence and background locations
p_coords <- virtualSp$presence
bg_coords <- virtualSp$background

# Create SWD object
data <- prepareSWD(species = "Virtual species", p = p_coords, a = bg_coords,
                  env = predictors, categorical = "biome")

# Train a model
model <- train(method = "Maxnet", data = data, fc = "l")

# Plot cloglog response curve for a continuous environmental variable (bio1)
plotResponse(model, var = "bio1", type = "cloglog")

# Plot marginal cloglog response curve for a continuous environmental
# variable (bio1)
plotResponse(model, var = "bio1", type = "cloglog", marginal = TRUE)

# Plot logistic response curve for a continuous environmental variable
# (bio12) adding the rugs and giving a custom color
plotResponse(model, var = "bio12", type = "logistic", rug = TRUE,
             color = "blue")

# Plot response curve for a categorical environmental variable (biome) giving
# a custom color
plotResponse(model, var = "biome", type = "logistic", color = "green")

# Train a model with cross validation
model <- train(method = "Maxnet", p = presence, a = bg, fc = "lq", rep = 4)

# Plot cloglog response curve for a continuous environmental variable (bio17)
plotResponse(model, var = "bio1", type = "cloglog")

# Plot logistic response curve for a categorical environmental variable
# (biome) giving a custom color
plotResponse(model, var = "biome", type = "logistic", color = "green")
```

`plotROC`*Plot ROC curve*

Description

Plot the ROC curve of the given model and print the AUC value.

Usage

```
plotROC(model, test = NULL, val = NULL)
```

Arguments

<code>model</code>	<code>SDMmodel</code> object.
<code>test</code>	<code>SWD</code> object. The testing dataset, default is <code>NULL</code> .
<code>val</code>	deprecated.

Value

The plot object.

Author(s)

Sergio Vignali

Examples

```
# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                  pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare presence and background locations
p_coords <- virtualSp$presence
bg_coords <- virtualSp$background

# Create SWD object
data <- prepareSWD(species = "Virtual species", p = p_coords, a = bg_coords,
                  env = predictors, categorical = "biome")

# Split presence locations in training (80%) and testing (20%) datasets
datasets <- trainValTest(data, test = 0.2, only_presence = TRUE)
train <- datasets[[1]]
test <- datasets[[2]]

# Train a model
model <- train(method = "Maxnet", data = train, fc = "1")
```

```
# Plot the training ROC curve
plotROC(model)

# Plot the training and testing ROC curves
plotROC(model, test = test)
```

plotVarImp

Plot Variable Importance

Description

Plot the variable importance as a bar plot.

Usage

```
plotVarImp(df, color = "grey")
```

Arguments

`df` data.frame. A data.frame containing the the name of the variables as first column and the value of the variable importance as second column.

`color` character. The color of the bar plot, default is grey.

Value

A `ggplot` object

Author(s)

Sergio Vignali

Examples

```
# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                  pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare presence and background locations
p_coords <- virtualSp$presence
bg_coords <- virtualSp$background

# Create SWD object
data <- prepareSWD(species = "Virtual species", p = p_coords, a = bg_coords,
                  env = predictors, categorical = "biome")
```

```
# Train a model
model <- train(method = "Maxnet", data = data, fc = "1")

# Compute variable importance
vi <- varImp(model, permut = 1)

# Plot variable importance
plotVarImp(vi)

# Plot variable importance with custom color
plotVarImp(vi, color = "red")
```

predict,ANN-method	<i>Predict ANN</i>
--------------------	--------------------

Description

Predict the output for a new dataset from a trained [ANN](#) model.

Usage

```
## S4 method for signature 'ANN'
predict(object, data, type, clamp)
```

Arguments

object	BRT object.
data	data.frame with the data for the prediction.
type	Not used.
clamp	Not used.

Details

Used by the [predict,SDMmodel-method](#), not exported.

Value

A vector with the predicted values.

Author(s)

Sergio Vignali

`predict,BRT-method` *Predict BRT*

Description

Predict the output for a new dataset from a trained [BRT](#) model.

Usage

```
## S4 method for signature 'BRT'
predict(object, data, type, clamp)
```

Arguments

<code>object</code>	BRT object.
<code>data</code>	data.frame with the data for the prediction.
<code>type</code>	Not used.
<code>clamp</code>	Not used.

Details

Used by the [predict,SDMmodel-method](#), not exported.

The function uses the number of tree defined to train the model and the "response" type output.

Value

A vector with the predicted values.

Author(s)

Sergio Vignali

`predict,Maxent-method` *Predict Maxent*

Description

Predict the output for a new dataset from a trained [Maxent](#) model.

Usage

```
## S4 method for signature 'Maxent'
predict(object, data, type = c("cloglog", "logistic",
  "raw"), clamp = TRUE)
```

Arguments

object	Maxent object.
data	data.frame with the data for the prediction.
type	character MaxEnt output type, possible values are "cloglog", "logistic" and "raw", default is "cloglog".
clamp	logical for clumping during prediction, default is TRUE.

Details

Used by the [predict,SDMmodel-method](#), not exported.

The function performs the prediction in **R** without calling the **MaxEnt** Java software. This results in a faster computation for large datasets. The results might differ slightly from the Java software output.

Value

A vector with the prediction

Author(s)

Sergio Vignali

References

Wilson P.D., (2009). Guidelines for computing MaxEnt model output values from a lambdas file.

predict,RF-method	<i>Predict RF</i>
-------------------	-------------------

Description

Predict the output for a new dataset from a trained **RF** model.

Usage

```
## S4 method for signature 'RF'  
predict(object, data, type, clamp)
```

Arguments

object	RF object.
data	data.frame with the data for the prediction.
type	Not used.
clamp	Not used.

Details

Used by the [predict,SDMmodel-method](#), not exported.

Value

A vector with the predicted probabilities of class 1.

Author(s)

Sergio Vignali

predict,SDMmodel-method

Predict

Description

Predict the output for a new dataset given a trained [SDMmodel](#) model.

Usage

```
## S4 method for signature 'SDMmodel'
predict(object, data, type = NULL, clamp = TRUE,
        filename = "", format = "GTiff", extent = NULL, parallel = FALSE,
        progress = "", ...)
```

Arguments

object	SDMmodel object.
data	data.frame, SWD or raster stack with the data for the prediction.
type	character. Output type, see details, used only for Maxent and Maxnet methods, default is NULL.
clamp	logical for clumping during prediction, used only for Maxent and Maxnet methods, default is TRUE.
filename	character. Output file name for the prediction map, if provided the output is saved in a file.
format	character. The output format, see writeRaster for all the options, default is "GTiff".
extent	Extent object, if provided it restricts the prediction to the given extent, default is NULL.
parallel	logical to use parallel computation during prediction, default is FALSE.
progress	character to display a progress bar: "text", "window" or "" (default) for no progress bar.
...	Additional arguments to pass to the writeRaster function.

Details

- For models trained with the **Maxent** method the argument type can be: "raw", "logistic" and "cloglog".
- For models trained with the **Maxnet** method the argument type can be: "link", "exponential", "logistic" and "cloglog", see [maxnet](#) for more details.
- For models trained with the **ANN** method the function uses the "raw" output type.
- For models trained with the **RF** method the output is the probability of class 1.
- For models trained with the **BRT** method the function uses the number of trees defined to train the model and the "response" output type.
- Parallel computation increases the speed only for large datasets due to the time necessary to create the cluster. For **Maxent** models the function performs the prediction in **R** without calling the **MaxEnt** Java software. This results in a faster computation for large datasets and might result in slightly different results compare to the Java software.

Value

A vector with the prediction or a Raster object if data is a raster [stack](#).

Author(s)

Sergio Vignali

References

Wilson P.D., (2009). Guidelines for computing MaxEnt model output values from a lambdas file.

Examples

```
# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                  pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare presence and background locations
p_coords <- virtualSp$presence
bg_coords <- virtualSp$background

# Create SWD object
data <- prepareSWD(species = "Virtual species", p = p_coords, a = bg_coords,
                  env = predictors, categorical = "biome")

# Split presence locations in training (80%) and testing (20%) datasets
datasets <- trainValTest(data, test = 0.2, only_presence = TRUE)
train <- datasets[[1]]
test <- datasets[[2]]

# Train a model
model <- train(method = "Maxnet", data = train, fc = "1")
```

```
# Make cloglog prediction for the test dataset
predict(model, data = test, type = "cloglog")

# Make logistic prediction for the all study area
predict(model, data = predictors, type = "logistic")

# Make logistic prediction for the all study area and save it in a file
predict(model, data = predictors, type = "logistic", filename = "my_map")
```

prepareSWD

Prepare an SWD object

Description

Given the coordinates, the species' name and the environmental variables, the function creates an [SWD](#) object (sample with data).

Usage

```
prepareSWD(species, p = NULL, a = NULL, coords = NULL, env,
            categorical = NULL)
```

Arguments

species	character. The name of the species.
p	data.frame. The coordinates of the presence locations.
a	data.frame. The coordinates of the absence/background locations.
coords	Deprecated.
env	stack containing the environmental variables used to extract the values at coordinate locations.
categorical	vector indicating which of the environmental variable are categorical, default is NULL.

Details

The [SWD](#) object is created in a way that the presence locations are always before than the absence/background locations.

Value

An [SWD](#) object.

Author(s)

Sergio Vignali

Examples

```
# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                   pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare presence and background locations
p_coords <- virtualSp$presence
bg_coords <- virtualSp$background

# Create the SWD object
data <- prepareSWD(species = "Virtual species", p = p_coords, a = bg_coords,
                  env = predictors, categorical = "biome")

data
```

randomFolds

Create Random Folds

Description

Create random folds for cross validation.

Usage

```
randomFolds(data, k, only_presence = FALSE, seed = NULL)
```

Arguments

data	SWD object that will be used to train the model.
k	integer. Number of fold used to create the partition.
only_presence	logical, if TRUE the random folds are created only for the presence locations and all the background locations are included in each fold, used manly for presence-only methods, default is FALSE.
seed	integer. The value used to set the seed for the fold partition, default is NULL.

Value

list with two matrices, the first for the training and the second for the testing dataset. Each column of one matrix represents a fold with TRUE for the locations included in and FALSE excluded from the partition.

When `only_presence = FALSE`, the proportion of presence and absence is preserved.

Author(s)

Sergio Vignali

Examples

```
# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                  pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare presence and background locations
p_coords <- virtualSp$presence
bg_coords <- virtualSp$background

data <- prepareSWD(species = "Virtual species", p = p_coords, a = bg_coords,
                  env = predictors, categorical = "biome")

# Create 4 random folds splitting presence and absence locations
folds <- randomFolds(data, k = 4)

# Create 4 random folds splitting only the presence locations
folds <- randomFolds(data, k = 4, only_presence = TRUE)
```

randomSearch

Random Search

Description

The function performs a random search in the hyperparameters space, creating a population of random models each one with a random combination of the provided hyperparameters values.

Usage

```
randomSearch(model, hypers, metric, test = NULL, bg4test = NULL,
             pop = 20, env = NULL, parallel = FALSE, seed = NULL)
```

Arguments

model	SDMmodel or SDMmodelCV object.
hypers	named list containing the values of the hyperparameters that should be tuned, see details.
metric	character. The metric used to evaluate the models, possible values are: "auc", "tss" and "aicc".
test	SWD object. Test dataset used to evaluate the model, not used with aicc and SDMmodelCV objects, default is NULL.
bg4test	Deprecated.
pop	numeric. Size of the population, default is 20.
env	stack containing the environmental variables, used only with "aicc", default is NULL.
parallel	logical, if TRUE it uses parallel computation, default is FALSE.
seed	numeric. The value used to set the seed to have consistent results, default is NULL.

Details

To know which hyperparameters can be tuned you can use the output of the function `get_tunable_args`. Hyperparameters not included in the `hypers` argument take the value that they have in the passed model. Parallel computation increases the speed only for large datasets due to the time necessary to create the cluster.

Value

`SDMtune` object.

Author(s)

Sergio Vignali

Examples

```
# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                    pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare presence and background locations
p_coords <- virtualSp$presence
bg_coords <- virtualSp$background

# Create SWD object
data <- prepareSWD(species = "Virtual species", p = p_coords, a = bg_coords,
                  env = predictors, categorical = "biome")

# Split presence locations in training (80%) and testing (20%) datasets
datasets <- trainValTest(data, test = 0.2, only_presence = TRUE)
train <- datasets[[1]]
test <- datasets[[2]]

# Train a model
model <- train(method = "Maxnet", data = train, fc = "1")

# Define the hyperparameters to test
h <- list(reg = seq(0.2, 3, 0.2), fc = c("lqp", "lqph", "lh"))

# Run the function using as metric the AUC
output <- randomSearch(model, hypers = h, metric = "auc", test = test,
                      pop = 10, seed = 25)

output@results
output@models
# Order results by highest test AUC
output@results[order(-output@results$test_AUC), ]
```

reduceVar

*Reduce Variables***Description**

Remove variables whose importance is less than the given threshold. The function removes one variable at time and after trains a new model to get the new variable contribution rank. If `use_jk` is TRUE the function checks if after removing the variable the model performance decreases (according to the given metric and based on the starting model). In this case the function stops removing the variable even if the contribution is lower than the given threshold.

Usage

```
reduceVar(model, th, metric, test = NULL, env = NULL,
          parallel = FALSE, use_jk = FALSE, permut = 10, use_pc = FALSE)
```

Arguments

<code>model</code>	SDMmodel or SDMmodelCV object.
<code>th</code>	numeric. The contribution threshold used to remove variables.
<code>metric</code>	character. The metric used to evaluate the models, possible values are: "auc", "tss" and "aicc", used only if <code>use_jk</code> is TRUE.
<code>test</code>	SWD object containing the test dataset used to evaluate the model, not used with <code>aicc</code> , and if <code>use_jk = FALSE</code> , default is NULL.
<code>env</code>	stack containing the environmental variables, used only with "aicc", default is NULL.
<code>parallel</code>	logical, if TRUE it uses parallel computation, default is FALSE. Used only with <code>AICc</code> .
<code>use_jk</code>	Flag to use the Jackknife AUC test during the variable selection, if FALSE the function uses the percent variable contribution, default is FALSE.
<code>permut</code>	integer. Number of permutations, used if <code>use_pc</code> is FALSE, default is 10.
<code>use_pc</code>	logical, use percent contribution. If TRUE and the model is trained using the Maxent method, the algorithm uses the percent contribution computed by Maxent software to score the variable importance, default is FALSE.

Value

The model trained using the selected variables.

Author(s)

Sergio Vignali

Examples

```

# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                   pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare presence and background locations
p_coords <- virtualSp$presence
bg_coords <- virtualSp$background

# Create SWD object
data <- prepareSWD(species = "Virtual species", p = p_coords, a = bg_coords,
                  env = predictors, categorical = "biome")

# Split presence locations in training (80%) and testing (20%) datasets
datasets <- trainValTest(data, test = 0.2, only_presence = TRUE)
train <- datasets[[1]]
test <- datasets[[2]]

# Train a Maxnet model
model <- train(method = "Maxnet", data = train, fc = "lq")

# Remove all variables with permutation importance lower than 2%
output <- reduceVar(model, th = 2, metric = "auc", test = test, permut = 1)

# Remove variables with permutation importance lower than 2% only if testing
# TSS doesn't decrease
output <- reduceVar(model, th = 2, metric = "tss", test = test, permut = 1,
                   use_jk = TRUE)

# Remove variables with permutation importance lower than 2% only if AICc
# doesn't increase
output <- reduceVar(model, th = 2, metric = "aicc", permut = 1,
                   use_jk = TRUE, env = predictors)

# Train a Maxent model
model <- train(method = "Maxent", data = train, fc = "lq")

# Remove all variables with percent contribution lower than 2%
output <- reduceVar(model, th = 2, metric = "auc", test = test,
                   use_pc = TRUE)

```

RF-class

*Random Forest***Description**

This Class represents a Random Forest model objects and hosts all the information related to the model.

Details

See [randomForest](#) for the meaning of the slots.

Slots

`mtry` integer. Number of variable randomly sampled.
`ntree` integer. Number of grown trees.
`nodesize` integer. Minimum size of terminal nodes.
`model` [randomForest](#). The randomForest model object.

Author(s)

Sergio Vignali

SDMmodel-class

SDMmodel

Description

This Class represents an SDMmodel model object and hosts all the information related to the model.

Slots

`data` [SWD](#) object. The data used to train the model.
`model` [Maxent](#) or [Maxnet](#) object.

Author(s)

Sergio Vignali

SDMmodel2MaxEnt

SDMmodel2MaxEnt

Description

Converts an [SDMmodel](#) object containing a [Maxent](#) model into a dismo [MaxEnt](#) object.

Usage

```
SDMmodel2MaxEnt(model)
```

Arguments

`model` [SDMmodel](#) object to be converted.

Value

The converted dismo [MaxEnt](#) object.

Author(s)

Sergio Vignali

Examples

```
# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                  pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare presence and background locations
p_coords <- virtualSp$presence
bg_coords <- virtualSp$background

# Create SWD object
data <- prepareSWD(species = "Virtual species", p = p_coords, a = bg_coords,
                  env = predictors, categorical = "biome")

# Train a Maxent model
model <- train(method = "Maxent", data = data, fc = "1")

dismo_model <- SDMmodel2MaxEnt(model)
dismo_model
```

SDMmodelCV-class

SDMmodelCV

Description

This Class represents a SDMmodel model object with replicates and hosts all the models trained during the cross validation.

Slots

`models` list. A list containing all the models trained during the cross validation.

`data` [SWD](#) object. Full dataset used to make the partitions.

`folds` list with two matrices, the first for the training and the second for the testing dataset. Each column of one matrix represents a fold with TRUE for the locations included in and FALSE excluded from the partition.

Author(s)

Sergio Vignali

SDMtune-class*SDMtune class*

Description

Class used to save the results of one of the following functions: [gridSearch](#), [randomSearch](#) or [optimizeModel](#).

Slots

`results` data.frame. Results with the evaluation of the models.

`models` list. List of [SDMmodel](#) or [SDMmodelCV](#) objects.

Author(s)

Sergio Vignali

SDMtune-pkg*SDMtune: A package for tuning Species Distribution Models.*

Description

SDMtune provides a user-friendly framework that enables the training and the evaluation of species distribution models (SDMs). The package implements functions for data driven variable selection and model tuning and includes numerous utilities to display the results. All the functions used to select variables or to tune model hyperparameters have an interactive real-time chart displayed in the RStudio viewer pane during their execution. SDMtune uses its own script to predict MaxEnt models, resulting in much faster predictions for large datasets compared to native predictions from the use of the Java software. This reduces considerably the computation time when tuning the model using the AICc. At the moment only the Maximum Entropy method is available using the Java implementation through the [dismo](#) package and the R implementation through the [maxnet](#) package. Visit the [package website](#) and learn how to use **SDMtune** starting from the first article [Prepare data for the analysis](#).

SWD-class	<i>Sample With Data</i>
-----------	-------------------------

Description

Object similar to the MaxEnt SWD format that hosts the species name, the coordinates of the locations and the value of the environmental variables at the location places.

Details

The object can contains presence/absence, presence/background, presence only or absence/background only data. Use the [prepareSWD](#) function to create the object.

Slots

`species` character. Name of the species.

`coords` data.frame. Coordinates of the locations.

`data` data.frame. Value of the environmental variables at location sites.

`pa` numeric. Vector with 1 for presence and 0 for absence/background locations.

Author(s)

Sergio Vignali

swd2csv	<i>SWD to csv</i>
---------	-------------------

Description

Save an [SWD](#) object as csv file.

Usage

```
swd2csv(swd, file_name)
```

Arguments

`swd` [SWD](#) object.

`file_name` character. The name of the file in which to save the object, see details.

Details

- The `file_name` argument should include the extension (i.e. `my_file.csv`).
- If `file_name` is a single name the function saves the presence absence/background locations in a single file, adding the column **pa** with 1s for presence and 0s for absence/background locations. If `file_name` is a vector with two names, it saves the object in two files: the first name is used for the presence locations and the second for the absence/background locations.

Author(s)

Sergio Vignali

Examples

```
# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                   pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare presence and background locations
p_coords <- virtualSp$presence
bg_coords <- virtualSp$background

# Create SWD object
data <- prepareSWD(species = "Virtual species", p = p_coords, a = bg_coords,
                  env = predictors, categorical = "biome")

# Save the SWD object as a single csv file
swd2csv(data, "train_data.csv")

# Save the SWD object in two separate csv files
swd2csv(data, c("presence.csv", "absence.csv"))
```

`thinData`*Thin Data*

Description

Remove all but one location per raster cell. The function removes NAs and if more than one location falls within the same raster cell it selects randomly one.

Usage

```
thinData(coords, env)
```

Arguments

`coords` data.frame or matrix with the coordinates.
`env` [stack](#) containing the environmental variables, or a single [raster](#) layer.

Details

`coords` and `env` must have the same coordinate reference system.

Value

a data frame with the thinned locations.

Author(s)

Sergio Vignali

Examples

```
# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                   pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare background locations
bg_coords <- dismo::randomPoints(predictors, 9000)

nrow(bg_coords)

# Thin the locations
thinned_bg <- thinData(bg_coords, env = predictors)
nrow(thinned_bg)
```

thresholds

Thresholds

Description

Compute three threshold values: minimum training presence, equal training sensitivity and specificity and maximum training sensitivity plus specificity together with fractional predicted area and the omission rate. If a test dataset is provided it returns also the equal test sensitivity and specificity and maximum test sensitivity plus specificity thresholds and the p-values of the one-tailed binomial exact test.

Usage

```
thresholds(model, type = NULL, test = NULL)
```

Arguments

model	SDMmodel object.
type	character. The output type used for "Maxent" and "Maxnet" methods, possible values are "cloglog" and "logistic", default is NULL.
test	SWD test locations, if not provided it returns the training and test thresholds, default is NULL.

Details

The equal training sensitivity and specificity minimizes the difference between sensitivity and specificity. The one-tailed binomial test checks that test points are predicted no better than by a random prediction with the same fractional predicted area.

Value

data.frame with the thresholds.

Author(s)

Sergio Vignali

Examples

```
# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                   pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare presence and background locations
p_coords <- virtualSp$presence
bg_coords <- virtualSp$background

# Create SWD object
data <- prepareSWD(species = "Virtual species", p = p_coords, a = bg_coords,
                  env = predictors, categorical = "biome")

# Split presence locations in training (80%) and testing (20%) datasets
datasets <- trainValTest(data, test = 0.2, only_presence = TRUE)
train <- datasets[[1]]
test <- datasets[[2]]

# Train a model
model <- train(method = "Maxnet", data = train, fc = "1")

# Get the cloglog thresholds
thresholds(model, type = "cloglog")

# Get the logistic thresholds passing the test dataset
thresholds(model, type = "logistic", test = test)
```

train

Train

Description

Train a model using one of the following methods: Artificial Neural Networks, Boosted Regression Trees, Maxent, Maxnet or Random Forest.

Usage

```
train(method, data, folds = NULL, verbose = TRUE, p = NULL,
      a = NULL, rep = NULL, seed = NULL, ...)
```

Arguments

method	character. Method used to train the model, possible values are "ANN", "BRT", "Maxent", "Maxnet" or "RF".
data	SWD object with presence and absence/background locations.
folds	list. Output of the function randomFolds or folds object created with other packages, see details, default is NULL.
verbose	logical, if TRUE shows a progress bar during cross validation, default is TRUE.
p	Deprecated.
a	Deprecated.
rep	Deprecated.
seed	Deprecated.
...	Arguments passed to the relative method, see details.

Details

- For the ANN method possible arguments are (for more details see [nnet](#)):
 - size: integer. Number of the units in the hidden layer.
 - decay numeric. Weight decay, default is 0.
 - rang numeric. Initial random weights, default is 0.7.
 - maxit integer. Maximum number of iterations, default is 100.
- For the BRT method possible arguments are (for more details see [gbm](#)):
 - distribution: character. Name of the distribution to use, default is "bernoulli".
 - n.trees: integer. Maximum number of tree to grow, default is 100.
 - interaction.depth: integer. Maximum depth of each tree, default is 1.
 - shrinkage: numeric. The shrinkage parameter, default is 0.1.
 - bag.fraction: numeric. Random fraction of data used in the tree expansion, default is 0.5.
- For the RF method the model is trained as classification. Possible arguments are (for more details see [randomForest](#)):
 - mtry: integer. Number of variable randomly sampled at each split, default is $\text{floor}(\sqrt{\text{number of variables}})$.
 - ntree: integer. Number of tree to grow, default is 500.
 - nodesize: integer. Minimum size of terminal nodes.
- For the Maxent method, possible arguments are:
 - reg: numeric. The value of the regularization multiplier, default is 1.
 - fc: character. The value of the feature classes, possible values are combinations of "l", "q", "p", "h" and "t", default is "lqph".
 - iter: numeric. Number of iterations used by the MaxEnt algorithm, default is 500.

- `extra_args`: vector. Extra arguments used to run MaxEnt, default is "removeduplicates=false" and "addsamplestobackground=false". In case this is not your expected behavior you can assign `extra_args = ""` or you can change or add any other additional arguments extending the default settings (e.g. `extra_args = c("removeduplicates=true,addsamplestobackground=true)**`)
- For the Maxnet method, possible arguments are (for more details see [maxnet](#)):
 - `reg`: numeric. The value of the regularization intensity, default is 1.
 - `fc`: character. The value of the feature classes, possible values are combinations of "l", "q", "p", "h" and "t", default is "lqph".

The `fold` argument accepts also objects created with other packages: **ENMeval** or **blockCV**. In this case the function converts internally the folds into a format valid for **SDMtune**.

Value

An `SDMmodel` or `SDMmodelCV` object.

Author(s)

Sergio Vignali

References

Venables, W. N. & Ripley, B. D. (2002) Modern Applied Statistics with S. Fourth Edition. Springer, New York. ISBN 0-387-95457-0.

Brandon Greenwell, Bradley Boehmke, Jay Cunningham and GBM Developers (2019). `gbm`: Generalized Boosted Regression Models. <https://CRAN.R-project.org/package=gbm>.

A. Liaw and M. Wiener (2002). Classification and Regression by `randomForest`. R News 2(3), 18–22.

Hijmans, Robert J., Steven Phillips, John Leathwick, and Jane Elith. 2017. `dismo`: Species Distribution Modeling. <https://cran.r-project.org/package=dismo>.

Steven Phillips (2017). `maxnet`: Fitting 'Maxent' Species Distribution Models with 'glmnet'. <https://CRAN.R-project.org/package=maxnet>.

Muscarella, R., Galante, P.J., Soley-Guardia, M., Boria, R.A., Kass, J., Uriarte, M. and R.P. Anderson (2014). `ENMeval`: An R package for conducting spatially independent evaluations and estimating optimal model complexity for ecological niche models. *Methods in Ecology and Evolution*.

Roosbeh Valavi, Jane Elith, José Lahoz-Monfort and Gurutzeta Guillera-Arroita (2018). `blockCV`: Spatial and environmental blocking for k-fold cross-validation. <https://github.com/rvalavi/blockCV>.

See Also

[randomFolds](#)

Examples

```

# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                    pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare presence and background locations
p_coords <- virtualSp$presence
bg_coords <- virtualSp$background

# Create SWD object
data <- prepareSWD(species = "Virtual species", p = p_coords, a = bg_coords,
                  env = predictors, categorical = "biome")

# Train a Maxent model
model <- train(method = "Maxent", data = data, fc = "l", reg = 1.5,
              iter = 700)

# Train a Maxnet model
model <- train(method = "Maxnet", data = data, fc = "lq", reg = 1.5)

# Cross Validation
# Create 4 random folds splitting only the presence data
folds <- randomFolds(data, k = 4, only_presence = TRUE)
model <- train(method = "Maxnet", data = data, fc = "l", reg = 0.8,
              folds = folds)

# Block partition using the ENMeval package
# library(ENMeval)
block_folds <- get.block(occ = data@data[data@pa == 1, ],
                       bg.coords = data@data[data@pa == 0, ])
model <- train(method = "Maxnet", data = data, fc = "l", reg = 0.8,
              folds = block_folds)

# Checkerboard1 partition using the ENMeval package
cb_folds <- get.checkerboard1(occ = data@data[data@pa == 1, ],
                             env = predictors,
                             bg.coords = data@data[data@pa == 0, ],
                             aggregation.factor = 4)
model <- train(method = "Maxnet", data = data, fc = "l", reg = 0.8,
              folds = cb_folds)

# Environmental block using the blockCV package
# library(blockCV)
# Create spatial points data frame
library(raster)
sp_df <- SpatialPointsDataFrame(data@coords, data = as.data.frame(data@pa),
                              proj4string = crs(predictors))
e_folds <- envBlock(rasterLayer = predictors,
                  speciesData = sp_df,
                  species = "data@pa",

```

```

        k = 4,
        standardization = "standard",
        rasterBlock = FALSE,
        numLimit = 100)
model <- train(method = "Maxnet", data = data, fc = "1", reg = 0.8,
              folds = e_folds)

# Train presence absence models
# Prepare presence and absence locations
p_coords <- virtualSp$presence
a_coords <- virtualSp$absence#'
# Create SWD object
data <- prepareSWD(species = "Virtual species", p = p_coords, a = a_coords,
                  env = predictors[[1:5]])

# Train an Artificial Neural Network model
model <- train("ANN", data = data, size = 10)

# Train a Random Forest model
model <- train("RF", data = data, ntree = 300)

# Train a Boosted Regression Tree model
model <- train("BRT", data = data, n.trees = 300, shrinkage = 0.001)

```

trainValTest

Train, Validation and Test datasets

Description

Split a dataset randomly in training and testing datasets or training, validation and testing datasets.

Usage

```
trainValTest(x, test, val = 0, only_presence = FALSE, seed = NULL)
```

Arguments

x	SWD object containing the data that have to be split in training, validation and testing datasets.
test	numeric. The percentage of data withhold for testing.
val	numeric. The percentage of data withhold for validation, default is 0.
only_presence	logical, if TRUE the split is done only for the presence locations and all the background locations are included in each partition, used manly for presence-only methods, default is FALSE.
seed	numeric. The value used to set the seed in order to have consistent results, default is NULL.

Details

When `only_presence = FALSE`, the proportion of presence and absence is preserved.

Value

A list with the training, validation and testing or training and testing SWD objects accordingly.

Author(s)

Sergio Vignali

Examples

```
# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                  pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare presence and background locations
p_coords <- virtualSp$presence
bg_coords <- virtualSp$background

# Create SWD object
data <- prepareSWD(species = "Virtual species", p = p_coords, a = bg_coords,
                  env = predictors, categorical = "biome")

# Split presence locations in training (80%) and testing (20%) datasets
# and splitting only the presence locations
datasets <- trainValTest(data, test = 0.2, only_presence = TRUE)
train <- datasets[[1]]
test <- datasets[[2]]

# Split presence locations in training (60%), validation (20%) and testing
# (20%) datasets and splitting the presence and the absence locations
datasets <- trainValTest(data, val = 0.2, test = 0.2)
train <- datasets[[1]]
val <- datasets[[2]]
test <- datasets[[3]]
```

tss

True Skill Statistics

Description

Compute the max TSS of a given model.

Usage

```
tss(model, test = NULL)
```

Arguments

model	SDMmodel or SDMmodelCV object.
test	SWD test locations for SDMmodel objects or logical for SDMmodelCV objects, if not provided it computes the training AUC, default is NULL.

Details

If the model is a [SDMmodelCV](#) object, the function computes the mean of the training or testing TSS values of the different replicates.

Value

The value of the TSS of the given model.

Author(s)

Sergio Vignali

References

Allouche O., Tsoar A., Kadmon R., (2006). Assessing the accuracy of species distribution models: prevalence, kappa and the true skill statistic (TSS). *Journal of Applied Ecology*, 43(6), 1223–1232.

Examples

```
# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                  pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare presence and background locations
p_coords <- virtualSp$presence
bg_coords <- virtualSp$background

# Create SWD object
data <- prepareSWD(species = "Virtual species", p = p_coords, a = bg_coords,
                  env = predictors, categorical = "biome")

# Split presence locations in training (80%) and testing (20%) datasets
datasets <- trainValTest(data, test = 0.2, only_presence = TRUE)
train <- datasets[[1]]
test <- datasets[[2]]

# Train a model
model <- train(method = "Maxnet", data = train, fc = "1")

# Compute the training TSS
tss(model)

# Compute the testing TSS
tss(model, test)
```

```
# Same example but using cross validation instead of training and testing
# datasets
# Create 4 random folds splitting only the presence locations
folds = randomFolds(data, k = 4, only_presence = TRUE)
model <- train(method = "Maxnet", p = presence, a = bg, fc = "1",
               folds = folds)

# Compute the training TSS
TSS(model)

# Compute the testing TSS
TSS(model, test = TRUE)
```

varImp

Variable Importance

Description

The function randomly permutes one variable at time (using training and absence/background datasets) and computes the decrease in training AUC. The result is normalized to percentages. Same implementation of MaxEnt java software but with the additional possibility of running several permutations to obtain a better estimate of the permutation importance. In case of more than one permutation (default is 10) the average of the decrease in training AUC is computed.

Usage

```
varImp(model, permut = 10)
```

Arguments

model [SDMmodel](#) or [SDMmodelCV](#) object.
permut integer. Number of permutations, default is 10.

Details

Note that it could return values slightly different from MaxEnt Java software due to a different random permutation.

For [SDMmodelCV](#) objects the function returns the average and the standard deviation of the permutation importances of the single models.

Value

data.frame with the ordered permutation importance.

Author(s)

Sergio Vignali

Examples

```

# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                   pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare presence and background locations
p_coords <- virtualSp$presence
bg_coords <- virtualSp$background

# Create SWD object
data <- preparesWD(species = "Virtual species", p = p_coords, a = bg_coords,
                  env = predictors, categorical = "biome")

# Split presence locations in training (80%) and testing (20%) datasets
datasets <- trainValTest(data, test = 0.2, only_presence = TRUE)
train <- datasets[[1]]
test <- datasets[[2]]

# Train a model
model <- train(method = "Maxnet", data = train, fc = "1")

# Compute variable importance
vi <- varImp(model, permut = 5)
vi

# Same example but using cross validation instead of training and testing
# datasets
# Create 4 random folds splitting only the presence locations
folds = randomFolds(data, k = 4, only_presence = TRUE)
model <- train(method = "Maxnet", p = presence, a = bg, fc = "1",
              folds = folds)
# Compute variable importance
vi <- varImp(model, permut = 5)
vi

```

varSel

Variable Selection

Description

The function performs a data-driven variable selection. Starting from the provided model it iterates through all the variables starting from the one with the highest contribution (permutation importance or maxent percent contribution). If the variable is correlated with other variables (according to the

given method and threshold) it performs a Jackknife test and among the correlated variables it removes the one that results in the best performing model when removed (according to the given metric for the training dataset). The process is repeated until the remaining variables are not highly correlated anymore.

Usage

```
varSel(model, metric, bg4cor, test = NULL, env = NULL,
       parallel = FALSE, method = "spearman", cor_th = 0.7, permut = 10,
       use_pc = FALSE)
```

Arguments

model	SDMmodel or SDMmodelCV object.
metric	character. The metric used to evaluate the models, possible values are: "auc", "tss" and "aicc".
bg4cor	SWD object. Background locations used to test the correlation between environmental variables.
test	SWD . Test dataset used to evaluate the model, not used with aicc and SDMmodelCV objects, default is NULL.
env	stack containing the environmental variables, used only with "aicc", default is NULL.
parallel	logical, if TRUE it uses parallel computation, default is FALSE. Used only with AICc.
method	character. The method used to compute the correlation matrix, default "spearman".
cor_th	numeric. The correlation threshold used to select highly correlated variables, default is 0.7.
permut	integer. Number of permutations, default is 10.
use_pc	logical, use percent contribution. If TRUE and the model is trained using the Maxent method, the algorithm uses the percent contribution computed by Maxent software to score the variable importance, default is FALSE.

Details

Parallel computation increases the speed only for large datasets due to the time necessary to create the cluster.

To find highly correlated variables the following formula is used:

$$|coeff| \leq cor_{th}$$

Value

The [SDMmodel](#) or [SDMmodelCV](#) object trained using the selected variables.

Author(s)

Sergio Vignali

Examples

```

# Acquire environmental variables
files <- list.files(path = file.path(system.file(package = "dismo"), "ex"),
                  pattern = "grd", full.names = TRUE)
predictors <- raster::stack(files)

# Prepare presence and background locations
p_coords <- virtualSp$presence
bg_coords <- virtualSp$background

# Create SWD object
data <- prepareSWD(species = "Virtual species", p = p_coords, a = bg_coords,
                  env = predictors, categorical = "biome")

# Split presence locations in training (80%) and testing (20%) datasets
datasets <- trainValTest(data, test = 0.2, only_presence = TRUE)
train <- datasets[[1]]
test <- datasets[[2]]

# Train a model
model <- train(method = "Maxnet", data = train, fc = "1")

# Prepare background locations to test autocorrelation
bg_coords <- dismo::randomPoints(predictors, 10000)
bg <- prepareSWD(species = "Virtual species", a = bg_coords,
                 env = predictors, categorical = "biome")

# Remove variables with correlation higher than 0.7 accounting for the AUC,
# in the following example the variable importance is computed as permutation
# importance
vs <- varSel(model, metric = "auc", bg4cor = bg, test = test, cor_th = 0.7,
            permut = 1)
vs

# Remove variables with correlation higher than 0.7 accounting for the TSS,
# in the following example the variable importance is the MaxEnt percent
# contribution
vs <- varSel(model, metric = "tss", bg4cor = bg, test = test, cor_th = 0.7,
            use_pc = TRUE)
vs

# Remove variables with correlation higher than 0.7 accounting for the aicc,
# in the following example the variable importance is the MaxEnt percent
# contribution
vs <- varSel(model, metric = "aicc", bg4cor = bg, cor_th = 0.7,
            use_pc = TRUE, env = predictors)
vs

```

`virtualSp`*Virtual Species*

Description

Dataset containing a random generated virtual species. The purpose of this dataset is to demonstrate the use of the functions included in the package.

Usage

```
virtualSp
```

Format

A list with five elements:

presence 400 random generated coordinates for the presence locations.

absence 300 random generated coordinates for the absence locations.

background 5000 random generated coordinates for the background locations.

pa_map Presence absence map used to extract the presence and absence locations.

prob_map Probability map of the random generated virtual species.

Details

The random species has been generated using the package **virtualsepecies**.

References

Leroy, B. , Meynard, C. N., Bellard, C. and Courchamp, F. (2016), virtualespecies, an R package to generate virtual species distributions. *Ecography*, 39: 599-607. doi:10.1111/ecog.01388

Index

*Topic **datasets**

virtualSp, 63

aicc, 3, 14

ANN, 35

ANN (ANN-class), 5

ANN-class, 5

auc, 5

BRT, 35, 36

BRT (BRT-class), 7

BRT-class, 7

confMatrix, 8

corVar, 9

dismo, 48

doJk, 10

Extent, 38

gbm, 7, 53

get_tunable_args, 13, 14, 24, 43

getSubsample, 12

ggplot, 26–31, 34

gridSearch, 14, 24, 48

MaxEnt, 46, 47

Maxent, 36, 37, 44, 46, 61

Maxent (Maxent-class), 15

Maxent-class, 15

maxentTh, 16, 18

maxentVarImp, 16, 17

Maxnet, 46

Maxnet (Maxnet-class), 18

maxnet, 8, 39, 48, 54

Maxnet-class, 18

mergeSWD, 19

modelReport, 20

nnet, 5, 53

old2NewSDMmodel, 21

old2NewSDMmodelCV, 22

old2NewSDMtune, 22

old2NewSWD, 23

optimizeModel, 15, 23, 48

plot, SDMtune, missing-method, 25

plotCor, 27

plotJk, 28

plotPA, 29, 30

plotPred, 30, 30

plotResponse, 31

plotROC, 33

plotVarImp, 34

predict, ANN-method, 35

predict, BRT-method, 36

predict, Maxent-method, 36

predict, RF-method, 37

predict, SDMmodel-method, 38

prepareSWD, 40, 49

randomFolds, 41, 53, 54

randomForest, 46, 53

randomSearch, 15, 24, 42, 48

raster, 29, 30, 50

reduceVar, 44

RF, 37

RF (RF-class), 45

RF-class, 45

SDMmodel, 4, 5, 8, 10, 13, 14, 16, 17, 20, 21,
24, 31, 33, 38, 42, 44, 46, 48, 51, 54,
58, 59, 61

SDMmodel (SDMmodel-class), 46

SDMmodel-class, 46

SDMmodel2MaxEnt, 46

SDMmodelCV, 5, 6, 10, 13, 14, 17, 22, 24, 31,
42, 44, 48, 54, 58, 59, 61

SDMmodelCV (SDMmodelCV-class), 47

SDMmodelCV-class, 47

SDMtune, [14](#), [22](#), [24–26](#), [43](#)
SDMtune (SDMtune-class), [48](#)
SDMtune-class, [48](#)
SDMtune-pkg, [48](#)
SDMtune-pkg-package (SDMtune-pkg), [48](#)
stack, [4](#), [10](#), [14](#), [20](#), [24](#), [38–40](#), [42](#), [44](#), [50](#), [61](#)
SWD, [5](#), [8–10](#), [12](#), [14](#), [19](#), [20](#), [23](#), [24](#), [27](#), [33](#), [38](#),
[40–42](#), [44](#), [46](#), [47](#), [49](#), [51](#), [53](#), [56–58](#),
[61](#)
SWD (SWD-class), [49](#)
SWD-class, [49](#)
swd2csv, [49](#)

thinData, [50](#)
thresholds, [51](#)
train, [52](#)
trainValTest, [56](#)
tss, [57](#)

varImp, [59](#)
varSel, [60](#)
virtualSp, [63](#)

writeRaster, [29](#), [38](#)