Package ‘ENMeval’

May 1, 2025

Type Package

Title Automated Tuning and Evaluations of Ecological Niche Models

Version 2.0.5.2

Date 2025-05-01

Maintainer Jamie M. Kass <jamie.m.kass@gmail.com>

Description Runs ecological niche models over all combinations of user-defined settings (i.e., tuning), performs cross validation to evaluate models, and returns data tables to aid in selection of optimal model settings that balance goodness-of-fit and model complexity. Also has functions to partition data spatially (or not) for cross validation, to plot multiple visualizations of results, to run null models to estimate significance and effect sizes of performance metrics, and to calculate range overlap between model predictions, among others. The package was originally built for Maxent models (Phillips et al. 2006, Phillips et al. 2017), but the current version allows possible extensions for any modeling algorithm. The extensive vignette, which guides users through most package functionality but unfortunately has a file size too big for CRAN, can be found here on the package's Github Pages website: <https://jamiemkass.github.io/ENMeval/articles/ENMeval-2.0-vignette.html>.

License GPL-3

Encoding UTF-8

Depends methods, R (>= 4.1.0)

Imports terra, maxnet, predicts (>= 0.1-16), parallel, foreach, utils, stats, grDevices, dplyr, tidyr, ggplot2, glmnet, rangeModelMetadata, rlang

RoxygenNote 7.3.2

LazyData true

Suggests rmarkdown, graphics, testthat, knitr, rJava (>= 0.5-0), spocc, RColorBrewer, sf, blockCV, devtools, tibble, ecospat, geodata, usdm

URL https://jamiemkass.github.io/ENMeval/

NeedsCompilation no

Author Jamie M. Kass [aut, cre],
Robert Muscarella [aut],
Peter J. Galante [aut],
Corentin Bohl [aut],
Gonzalo E. Buitrago-Pinilla [aut],
Robert A. Boria [aut],
Mariano Soley-Guardia [aut],
Robert P. Anderson [aut]

Repository CRAN

Date/Publication 2025-05-01 13:50:07 UTC

Contents

`ENMeval-package` *Automated runs and evaluations of ecological niche models*

Description

Runs ecological niche models over all combinations of user-defined settings (i.e., tuning), performs cross validation to evaluate models, and returns data tables to aid in selection of optimal model settings that balance goodness-of-fit and model complexity. Also has functions to partition data spatially (or not) for cross validation, to plot multiple visualizations of results, to run null models to estimate significance and effect sizes of performance metrics, and to calculate range overlap between model predictions, among others. The package was originally built for Maxent models (Phillips et al. 2006, Phillips et al. 2017), but the current version allows possible extensions for any modeling algorithm. The extensive vignette, which guides users through most package functionality but unfortunately has a file size too big for CRAN, can be found here on the package's Github Pages website: `https://jamiemkass.github.io/ENMeval/articles/ENMeval-2.0-vignette.html`.

Details

See README for details.

Author(s)

Maintainer: Jamie M. Kass <`jamie.m.kass@gmail.com`>

Authors:

- Robert Muscarella
- Peter J. Galante
- Corentin Bohl
- Gonzalo E. Buitrago-Pinilla
- Robert A. Boria
- Mariano Soley-Guardia
- Robert P. Anderson

See Also

Useful links:

- `https://jamiemkass.github.io/ENMeval/`

`aic.maxent` *Calculate AICc from Maxent model prediction*

Description

This function calculates AICc for Maxent models based on Warren and Seifert (2011).

Usage

```
aic.maxent(p.occs, ncoefs, p = NULL)
```

Arguments

`p.occs`	data frame: raw (maxent.jar) or exponential (maxnet) predictions for the occurrence localities based on one or more models
`ncoefs`	numeric: number of non-zero model coefficients
`p`	SpatRaster: raw (maxent.jar) or exponential (maxnet) model predictions; if NULL, AICc will be calculated based on the background points, which already have predictions that sum to 1 and thus need no correction – this assumes that the background points represent a good sample of the study extent

Details

As motivated by Warren and Seifert (2011) and implemented in ENMTools (Warren *et al.* 2010), this function calculates the small sample size version of Akaike Information Criterion for ENMs (Akaike 1974). We use AICc (instead of AIC) regardless of sample size based on the recommendation of Burnham and Anderson (1998, 2004). The number of coefficients is determined by counting the number of non-zero coefficients in the `maxent` lambda file (`m@lambdas` for maxent.jar and `m$betas` for maxnet. See Warren *et al.* (2014) for limitations of this approach, namely that the number of non-zero coefficients is an estimate of the true degrees of freedom. For Maxent ENMs, AICc is calculated by first standardizing the raw predictions such that all cells in the study extent sum to 1, then extracting the occurrence record predictions. The predictions of the study extent may not sum to 1 if the background does not cover every grid cell – as the background predictions sum to 1 by definition, extra predictions for grid cells not in the training data will add to this sum. When no raster data is provided, the raw predictions of the occurrence records are used to calculate AICc without standardization, with the assumption that the background records have adequately represented the occurrence records. The standardization is not necessary here because the background predictions sum to 1 already, and the occurrence data is a subset of the background. This will not be true if the background does not adequately represent the occurrence records, in which case the occurrences are not a subset of the background and the raster approach should be used instead. The likelihood of the data for a given model is then calculated by taking the product of the raw occurrence predictions (Warren and Seifert 2011), or the sum of their logs, as is implemented here.

Value

data frame with three columns: `AICc` is the Akaike Information Criterion corrected for small sample sizes calculated as:

$$(2 * K - 2 * logLikelihood) + (2 * K) * (K + 1)/(n - K - 1)$$

where *K* is the number of non-zero coefficients in the model and *n* is the number of occurrence localities. The *logLikelihood* is calculated as:

$$sum(log(vals))$$

where *vals* is a vector of Maxent raw/exponential values at occurrence localities and the sum of these values across the study extent is equal to 1. `delta.AICc` is the difference between the AICc of a given model and the AICc of the model with the lowest AICc. `w.AICc` is the Akaike weight (calculated as the relative likelihood of a model (exp(-0.5 * `delta.AICc`)) divided by the sum of the likelihood values of all models included in a run. These can be used for model averaging (Burnham and Anderson 2002).

Note

Returns all `NA`s if the number of non-zero coefficients is larger than the number of observations (occurrence localities).

References

Akaike, H. (1974) A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, **19**: 716-723. doi:10.1109/TAC.1974.1100705

Burnham, K. P. and Anderson, D. R. (1998) Model selection and multimodel inference: a practical information-theoretic approach. Springer, New York.

Burnham, K. P. and Anderson, D. R. (2004) Multimodel inference: understanding AIC and BIC in model selection. *Sociological Methods and Research*, **33**: 261-304. doi:10.1177/0049124104268644

Warren, D. L., Glor, R. E, and Turelli, M. (2010) ENMTools: a toolbox for comparative studies of environmental niche models. *Ecography*, **33**: 607-611. doi:10.1111/j.16000587.2009.06142.x

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**: 335-342. doi:10.1890/101171.1

Warren, D. L., Wright, A. N., Seifert, S. N., and Shaffer, H. B. (2014). Incorporating model complexity and sampling bias into ecological niche models of climate change risks faced by 90 California vertebrate species of concern. *Diversity and Distributions*, **20**: 334-343. doi:10.1111/ddi.12160

See Also

`MaxEnt` for the **predicts** package.

`buildRMM` *Build metadata object from ENMeval results*

Description

Builds a `rangeModelMetadata` object from the output of `ENMevaluate`. See Merow *et al.* (2019) for more details on the nature of the metadata and the `rangeModelMetadata` package. To improve reproducibility of the study, this metadata object can be used as supplemental information for a manuscript, shared with collaborators, etc.

Usage

```
buildRMM(e, envs, rmm = NULL)
```

Arguments

`e`	ENMevaluation object
`envs`	SpatRaster: environmental predictor variables used in analysis; needed to pull information on the predictor variables not included in the ENMevaluation object
`rmm`	rangeModelMetadata object: if included, fields are appended to this RMM object as opposed to returning a new RMM object

References

Merow, C., Maitner, B. S., Owens, H. L., Kass, J. M., Enquist, B. J., Jetz, W., & Guralnick, R. (2019). Species' range model metadata standards: RMMS. *Global Ecology and Biogeography*, **28**: 1912-1924. doi:10.1111/geb.12993

`bvariegatus` *Example occurrence dataset.*

Description

An example occurrence dataset for the Brown Throated Sloth (*Bradypus variegatus*) downloaded from GBIF with the `spocc` package.

Usage

```
bvariegatus
```

Format

A data frame with 476 rows and 2 variables:

longitude The longitude coordinate (x).

latitude The latitude coordinate (y).

Source

https://www.gbif.org/

calc.10p.trainThresh *Calculate 10 percentile threshold*

Description

Function to calculate the 10 percentile threshold from training predictions

Usage

```
calc.10p.trainThresh(pred.train)
```

Arguments

`pred.train`	numeric vector: training occurrence predictions

calc.niche.overlap *Calculate Similarity of ENMs in Geographic Space*

Description

Compute pairwise "niche overlap" in geographic space for Maxent predictions. The value ranges from 0 (no overlap) to 1 (identical predictions). The function uses the `nicheOverlap` function of the **dismo** package (Hijmans *et al.* 2011).

Usage

```
calc.niche.overlap(predictors, overlapStat, quiet = FALSE)
```

Arguments

`predictors`	SpatRaster: at least 2 Maxent raster predictions
`overlapStat`	character: either "D" or "I", the statistic calculated by the `nicheOverlap` function of the **dismo** package (default: "D"), which we updated for **terra** as no correlate currently exists in the new **predicts** package
`quiet`	boolean: if TRUE, silence all function messages (but not errors)

Details

"D" refers to Schoeners *D* (Schoener 1968), while "I" refers to the *I* similarity statistic from Warren *et al.* (2008).

Value

A matrix with the lower triangle giving values of pairwise "niche overlap" in geographic space. Row and column names correspond to the results table output by `ENMevaluate()`.

Author(s)

Based on **dismo**::`nicheOverlap`, which is based on **SDMTools**::`Istat`, updated for **terra** package Robert Muscarella <bob.muscarella@gmail.com>

References

Hijmans, R. J., Phillips, S., Leathwick, J. & Elith, J. (2011) dismo package for R. Available online at: `https://cran.r-project.org/package=dismo`.

Schoener, T. W. (1968) The *Anolis* lizards of Bimini: resource partitioning in a complex fauna. *Ecology*, **49**: 704-726. doi:10.2307/1935534

Warren, D. L., Glor, R. E., Turelli, M. & Funk, D. (2008) Environmental niche equivalency versus conservatism: quantitative approaches to niche evolution. *Evolution*, **62**: 2868-2883. doi:10.1111/j.15585646.2008.00482.x

See Also

'nicheOverlap' in the **dismo** package

`clamp.vars` *Clamp predictor variables*

Description

This function restricts the values of one or more predictor variable rasters to stay within the bounds of the input occurrence and background data (argument "ref.vals"). This is termed "clamping", and is mainly used to avoid making extreme extrapolations when making model predictions to environmental conditions outside the range of the occurrence / background data used to train the model. Clamping can be done on variables of choice on one or both tails of their distributions (i.e., arguments "left" and "right" for minimum and maximum clamps, respectively). If "left" and/or "right" are not specified and left at the default NULL, the function will clamp all variables for that tail (thus, the function default is to clamp all variables on both sides). To turn off clamping for one side, enter "none" for either "left" or "right".

Categorical variables need to be declared with the argument "categoricals". These variables are excluded from the clamping analysis, but are put back into the SpatRaster that is returned.

Usage

```
clamp.vars(orig.vals, ref.vals, left = NULL, right = NULL, categoricals = NULL)
```

Arguments

`orig.vals`	SpatRaster / matrix / data frame: environmental predictor variables (must be in same geographic projection as occurrence data), or predictor variables values for the original records
`ref.vals`	matrix / data frame: predictor variable values for the reference records (not including coordinates), used to determine the minimums and maximums – this should ideally be the occurrences + background (can be made with terra::extract())
`left`	character vector: names of variables to get a minimum clamp; can be "none" to turn off minimum clamping
`right`	character vector: names of variables to get a maximum clamp, can be "none" to turn off maximum clamping
`categoricals`	character vector: name or names of categorical environmental variables

Value

The clamped SpatRaster object.

Author(s)

Stephen J. Phillips, Jamie M. Kass, Gonzalo Pinilla-Buitrago

`corrected.var` *Corrected variance function*

Description

Calculate variance corrected for non-independence of *k*-fold iterations

Usage

```
corrected.var(x, nk)
```

Arguments

`x`	numeric vector: input values
`nk`	numeric: number of *k*-fold iterations

Details

‘corrected.var‘ calculates variance corrected for non-independence of *k*-fold iterations. See Appendix of Shcheglovitova & Anderson (2013) and other references (Miller 1974; Parr 1985; Shao and Wu 1989) for additional details. This function calculates variance that is corrected for the non-independence of *k* cross-validation iterations. Following Shao and Wu (1989):

$$SumOfSquares * ((n-1)/n)$$

where *n* = the number of *k*-fold iterations.

Value

A numeric value of the corrected variance.

Author(s)

Robert Muscarella <bob.muscarella@gmail.com>

References

Miller, R. G. (1974) The jackknife - a review. *Biometrika*, **61**: 1-15. doi:10.1093/biomet/61.1.1

Parr, W. C. (1985) Jackknifing differentiable statistical functionals. *Journal of the Royal Statistics Society, Series B*, **47**: 56-66. doi:10.1111/j.25176161.1985.tb01330.x

Shao J. and Wu, C. F. J. (1989) A general theory for jackknife variance estimation. *Annals of Statistics*, **17**: 1176-1197. doi:10.1214/aos/1176347263

Shcheglovitova, M. and Anderson, R. P. (2013) Estimating optimal complexity for ecological niche models: a jackknife approach for species with small sample sizes. *Ecological Modelling*, **269**: 9-17. doi:10.1016/j.ecolmodel.2013.08.011

`emp.bg` *emp.bg generic for ENMnull object*

Description

emp.bg generic for ENMnull object

Usage

```
emp.bg(x)

## S4 method for signature 'ENMnull'
emp.bg(x)
```

Arguments

`x`	ENMnull object

`emp.bg.grp` *emp.bg.grp generic for ENMnull object*

Description

emp.bg.grp generic for ENMnull object

Usage

```
emp.bg.grp(x)

## S4 method for signature 'ENMnull'
emp.bg.grp(x)
```

Arguments

`x`	ENMnull object

`emp.occs` *emp.occs generic for ENMnull object*

Description

emp.occs generic for ENMnull object

Usage

```
emp.occs(x)

## S4 method for signature 'ENMnull'
emp.occs(x)
```

Arguments

`x`	ENMnull object

emp.occs.grp — *emp.occs.grp generic for ENMnull object*

Description

emp.occs.grp generic for ENMnull object

Usage

```
emp.occs.grp(x)

## S4 method for signature 'ENMnull'
emp.occs.grp(x)
```

Arguments

`x`	ENMnull object

enm.args — *enm.args generic for ENMdetails object*

Description

enm.args generic for ENMdetails object

Usage

```
enm.args(x)

enm.args(x) <- value

## S4 method for signature 'ENMdetails'
enm.args(x)

## S4 replacement method for signature 'ENMdetails'
enm.args(x) <- value
```

Arguments

`x`	ENMdetails object
`value`	input value

`enm.bioclim` *ENMdetails bioclim*

Description

This is the ENMdetails implementation for the BIOCLIM climate envelope model, implemented by predicts.

Usage

```
enm.bioclim
```

Format

An object of class `ENMdetails` of length 1.

`enm.errors` *enm.errors generic for ENMdetails object*

Description

enm.errors generic for ENMdetails object

Usage

```
enm.errors(x)

enm.errors(x) <- value

## S4 method for signature 'ENMdetails'
enm.errors(x)

## S4 replacement method for signature 'ENMdetails'
enm.errors(x) <- value
```

Arguments

`x`	ENMdetails object
`value`	input value

enm.fun *enm.fun generic for ENMdetails object*

Description

enm.fun generic for ENMdetails object

Usage

```
enm.fun(x)

enm.fun(x) <- value

## S4 method for signature 'ENMdetails'
enm.fun(x)

## S4 replacement method for signature 'ENMdetails'
enm.fun(x) <- value
```

Arguments

`x`	ENMdetails object
`value`	input value

enm.maxent.jar *ENMdetails maxent.jar*

Description

This is the ENMdetails implementation for maxent.jar, the Java version of the Maxent algorithm. The configuration for running the model differs slightly from that in previous versions of ENMeval (0.3.0 and before) in that this version (>=2.0.0) uses the default of adding presences to the background for model training, while previous versions had turned this off. Specifically, previous versions ran maxent() with "noaddsamplestobackground" in the "args" vector argument, while this version does not.

Usage

```
enm.maxent.jar
```

Format

An object of class `ENMdetails` of length 1.

`enm.maxnet` *ENMdetails maxnet*

Description

This is the ENMdetails implementation for maxnet, the R version of the Maxent algorithm. The configuration for running the model now includes addsamplestobackground = TRUE, which explicitly adds presences to the background for model training, though as the current version of maxnet has this set to TRUE as default, behavior between ENMeval versions should not differ.

Usage

```
enm.maxnet
```

Format

An object of class `ENMdetails` of length 1.

`enm.msgs` *enm.msgs generic for ENMdetails object*

Description

enm.msgs generic for ENMdetails object

Usage

```
enm.msgs(x)

enm.msgs(x) <- value

## S4 method for signature 'ENMdetails'
enm.msgs(x)

## S4 replacement method for signature 'ENMdetails'
enm.msgs(x) <- value
```

Arguments

`x`	ENMdetails object
`value`	input value

enm.name *eval.name generic for ENMdetails object*

Description

eval.name generic for ENMdetails object

Usage

```
enm.name(x)

enm.name(x) <- value

## S4 method for signature 'ENMdetails'
enm.name(x)

## S4 replacement method for signature 'ENMdetails'
enm.name(x) <- value
```

Arguments

x	ENMdetails object
value	input value

enm.ncoefs *enm.ncoefs generic for ENMdetails object*

Description

enm.ncoefs generic for ENMdetails object

Usage

```
enm.ncoefs(x)

enm.ncoefs(x) <- value

## S4 method for signature 'ENMdetails'
enm.ncoefs(x)

## S4 replacement method for signature 'ENMdetails'
enm.ncoefs(x) <- value
```

Arguments

x	ENMdetails object
value	input value

enm.predict *enm.predict generic for ENMdetails object*

Description

enm.predict generic for ENMdetails object

Usage

```
enm.predict(x)

enm.predict(x) <- value

## S4 method for signature 'ENMdetails'
enm.predict(x)

## S4 replacement method for signature 'ENMdetails'
enm.predict(x) <- value
```

Arguments

x	ENMdetails object
value	input value

enm.variable.importance *enm.variable.importance generic for ENMdetails object*

Description

enm.variable.importance generic for ENMdetails object

Usage

```
enm.variable.importance(x)

enm.variable.importance(x) <- value

## S4 method for signature 'ENMdetails'
enm.variable.importance(x)

## S4 replacement method for signature 'ENMdetails'
enm.variable.importance(x) <- value
```

Arguments

`x`	ENMdetails object
`value`	input value

`ENMdetails-class` *ENMdetails class*

Description

An S4 class that details packages, functions, messages associated with a specific species distribution model (SDM) or ecological niche model (ENM). Objects of this class are generated by `ENMdetails()`. For examples, look in the package's R folder for scripts beginning with "enm" – these are pre-made ENMdetails object specifications that work with ENMeval out of the box.

Usage

```
## S4 method for signature 'ENMdetails'
show(object)
```

Arguments

`object`	ENMdetails object

Slots

- `name` character: name of algorithm
- `fun` function: function that runs the algorithm
- `errors` function: returns errors chosen by the user to prevent any malfunction in the analysis. The available arguments are: occs, envs, bg, tune.args, partitions, algorithm, partition.settings, other.settings, categoricals, doClamp, clamp.directions.
- `msgs` function: prints messages showing the package version number, etc., and those related to the input tuning parameters `tune.args`. The available arguments are: tune.args, other.settings.
- `args` function: returns the parameters needed to run the algorithm function. The available arguments are: occs.z, bg.z, tune.tbl.i, other.settings (where x.z is a data.frame of the envs values at coordinates of x, and tune.tbl.i is a single set of tuning parameters).
- `predict` function: specifies how to calculate a model prediction for a Raster* or a data frame. The available arguments are: mod, envs, other.settings.
- `ncoefs` function: counts the number of non-zero model coefficients. The available arguments are: mod.
- `variable.importance` function: generates a data frame of variable importance from the model object (if functionality is available). The available arguments are: mod.

Author(s)

Jamie M. Kass, `<jamie.m.kass@gmail.com>`

`ENMevaluate` *Tune ecological niche model (ENM) settings and calculate evaluation statistics*

Description

`ENMevaluate()` is the primary function for the **ENMeval** package. This function builds ecological niche models iteratively across a range of user-specified tuning settings. Users can choose to evaluate models with cross validation or a full-withheld testing dataset. `ENMevaluate()` returns an `ENMevaluation` object with slots containing evaluation statistics for each combination of settings and for each cross validation fold therein, as well as raster predictions for each model when raster data is input. The evaluation statistics in the results table should aid users in identifying model settings that balance fit and predictive ability. See the extensive vignette for fully worked examples: <https://jamiemkass.github.io/ENMeval/articles/ENMeval-2.0-vignette.html>.

Usage

```
ENMevaluate(
  occs,
  envs = NULL,
  bg = NULL,
  tune.args = NULL,
  partitions = NULL,
  algorithm = NULL,
  partition.settings = NULL,
  other.settings = list(),
  categoricals = NULL,
  doClamp = TRUE,
  raster.preds = TRUE,
  clamp.directions = NULL,
  user.enm = NULL,
  user.grp = NULL,
  occs.testing = NULL,
  taxon.name = NULL,
  n.bg = 10000,
  overlap = FALSE,
  overlapStat = c("D", "I"),
  user.val.grps = NULL,
  user.eval = NULL,
  rmm = NULL,
  parallel = FALSE,
  numCores = NULL,
  updateProgress = FALSE,
  quiet = FALSE
)
```

Arguments

`occs`	matrix / data frame: occurrence records with two columns for longitude and latitude of occurrence localities, in that order. If specifying predictor variable values assigned to presence/background localities (without inputting raster data), this table should also have one column for each predictor variable. See Note for important distinctions between running the function with and without rasters.
`envs`	SpatRaster: environmental predictor variables. These should be in same geographic projection as occurrence data.
`bg`	matrix / data frame: background records with two columns for longitude and latitude of background (or pseudo-absence) localities, in that order. If NULL, points will be randomly sampled across `envs` with the number specified by argument `n.bg`. If specifying predictor variable values assigned to presence/background localities (without inputting raster data), this table should also have one column for each predictor variable. See Details for important distinctions between running the function with and without rasters.
`tune.args`	named list: model settings to be tuned (i.e., for Maxent models: `list(fc = c("L","Q"), rm = 1:3))`
`partitions`	character: name of partitioning technique. Currently available options are the nonspatial partitions "randomkfold" and "jackknife", the spatial partitions "block" and "checkerboard", "testing" for partitioning with fully withheld data (see argument occs.testing), the "user" option (see argument user.grp), and "none" for no partitioning (see `?partitions` for details).
`algorithm`	character: name of the algorithm used to build models. Currently one of "maxnet", "maxent.jar", or "bioclim", else the name from a custom ENMdetails implementation.
`partition.settings`	named list: used to specify certain settings for partitioning schema. See Details and ?partitions for descriptions of these settings.
`other.settings`	named list: used to specify extra settings for the analysis. All of these settings have internal defaults, so if they are not specified the analysis will be run with default settings. See Details for descriptions of these settings, including how to specify arguments for maxent.jar.
`categoricals`	character vector: name or names of categorical environmental variables. If not specified, all predictor variables will be treated as continuous unless they are factors. If categorical variables are already factors, specifying names of such variables in this argument is not needed.
`doClamp`	boolean: if TRUE (default), model prediction extrapolations will be restricted to the upper and lower bounds of the predictor variables. Clamping avoids extreme predictions for environment values outside the range of the training data. If free extrapolation is a study aim, this should be set to FALSE, but for most applications leaving this at the default of TRUE is advisable to avoid unrealistic predictions. When predictor variables are input, they are clamped internally before making model predictions when clamping is on. When no predictor variables are input and data frames of coordinates and variable values are used instead (SWD format), validation data is clamped before making model predictions when clamping is on.

`raster.preds`	boolean: if TRUE (default), return model prediction rasters. If this is FALSE, the predictions slot in the ENMevaluation object will be empty, which is the same as if no raster data is input. You can still make model prediction rasters using the model objects in the models slot with the predict() function.
`clamp.directions`	named list: specifies the direction ("left" for minimum, "right" for maximum) of clamping for predictor variables – (e.g., `list(left = c("bio1","bio5"), right = c("bio10","bio15"))`).
`user.enm`	ENMdetails object: a custom ENMdetails object used to build models. This is an alternative to specifying `algorithm` with a character string.
`user.grp`	named list: specifies user-defined partition groups, where `occs.grp` = vector of partition group (fold) for each occurrence locality, intended for user-defined partitions, and `bg.grp` = same vector for background (or pseudo-absence) localities.
`occs.testing`	matrix / data frame: a fully withheld testing dataset with two columns for longitude and latitude of occurrence localities, in that order when `partitions = "testing"`. These occurrences will be used only for evaluation but not for model training, and thus no cross validation will be performed.
`taxon.name`	character: name of the focal species or taxon. This is used primarily for annotating the ENMevaluation object and output metadata (rmm), but not necessary for analysis.
`n.bg`	numeric: the number of background (or pseudo-absence) points to randomly sample over the environmental raster data (default: 10000) if background records were not already provided.
`overlap`	boolean: if TRUE, calculate range overlap statistics (Warren *et al.* 2008).
`overlapStat`	character: range overlap statistics to be calculated – "D" (Schoener's D) and or "I" (Hellinger's I) – see ?calc.niche.overlap for more details.
`user.val.grps`	matrix / data frame: user-defined validation record coordinates and predictor variable values. This is used internally by `ENMnulls()` to force each null model to evaluate with empirical validation data, and does not have any current use when running `ENMevaluate()` independently.
`user.eval`	function: custom function for specifying performance metrics not included in **ENMeval**. The function must first be defined and then input as the argument `user.eval`. This function should have a single argument called `vars`, which is a list that includes different data that can be used to calculate the metric. See Details below and the vignette for a worked example.
`rmm`	rangeModelMetadata object: if specified, `ENMevaluate()` will write metadata details for the analysis into this object, but if not, a new `rangeModelMetadata` object will be generated and included in the output `ENMevaluation` object.
`parallel`	boolean: if TRUE, run with parallel processing.
`numCores`	numeric: number of cores to use for parallel processing. If NULL, all available cores will be used.
`updateProgress`	boolean: if TRUE, use shiny progress bar. This is only for use in shiny apps.
`quiet`	boolean: if TRUE, silence all function messages (but not errors).

Details

There are a few methodological details in the implementation of ENMeval >=2.0.0 that are important to mention. There is also a brief discussion of some points relevant to null models in ?ENMnulls.

1. By default, validation AUC is calculated with respect to the full background (training + validation). This approach follows Radosavljevic & Anderson (2014).This setting can be changed by assigning other.settings$validation.bg to "partition", which will calculate AUC with respect to the validation background only. The default value for other.settings$validation.bg is "full". NOTE: When examining validation AUC and other discrimination metrics, the "full" option will likely result in higher performance than for the "partition" option because more and varied background data should lead to higher discriminatory power for the model. Users should thus make sure they are correctly interpreting the evaluation results.

2. The continuous Boyce index (always) and AICc (when no raster is provided) are not calculated using the predicted values of the SpatRaster delineating the full study extent, but instead using the predicted values for the background records. This decision to use the background only for calculating the continuous Boyce index was made to simplify the code and improve running time. The decision for AICc was made in order to allow AICc calculations for datasets that do not include raster data. See ?calc.aicc for more details, and for caveats when calculating AICc without raster data (mainly, that if the background does not adequately represent the occurrence records, users should use the raster approach, for reasons explained in the calc.aicc documentation). For both metrics, if the background records are a good representation of the study extent, there should not be much difference between this approach using the background data and the approach that uses rasters.

3. When running `ENMevaluate()` without raster data, and instead adding the environmental predictor values to the occurrence and background data tables, users may notice some differences in the results. Occurrence records that share a raster grid cell are automatically removed when raster data is provided, but without raster data this functionality cannot operate, and thus any such duplicate occurrence records can remain in the training data. The Java implementation of Maxent (maxent.jar implemented with `MaxEnt()` from the R package `predicts`) should automatically remove these records, but the R implementation `maxnet` does not, and the `envelope()` function from the R package `predicts` does not as well. Therefore, it is up to the user to remove such records before running `ENMevaluate()` when raster data are not included.

Below are descriptions of the parameters used in the other.settings, partition.settings, and user.eval arguments.

For other.settings, the options are:
* path - character: the folder path designating where maxent.jar files should be saved
* removeduplicates - boolean: whether or not to remove grid-cell duplicates for occurrences (this controls behavior for maxent.jar and ENMeval)
* addsamplestobackground - boolean: whether or not to add occurrences to the background when modeling with maxnet – the default is TRUE.
* abs.auc.diff - boolean: if TRUE, take absolute value of AUCdiff (default: TRUE)
* pred.type - character: specifies which prediction type should be used to generate maxnet or maxent.jar prediction rasters (default: "cloglog").
* validation.bg - character: either "full" to calculate training and validation AUC and CBI for cross-validation with respect to the full background (default), or "partition" (meant for spatial partitions only) to calculate each with respect to the partitioned background only (i.e., training occurrences are

compared to training background, and validation occurrences compared to validation background).
* other.args - named list: any additional model arguments not specified for tuning; this can include arguments for maxent.jar, which are described in the software's Help file, such as "jackknife=TRUE" for a variable importance jackknife plot or "responsecurves=TRUE" for response curve plots – note the the "path" must be specified (see above).

For partition.settings, the current options are:
* orientation - character: one of "lat_lon" (default), "lon_lat", "lat_lat", or "lon_lon" (required for block partition).
* aggregation.factor - numeric vector: one or two numbers specifying the factor with which to aggregate the envs (default: 2) raster to assign partitions (required for the checkerboard partitions).
* kfolds - numeric: the number of folds (i.e., partitions) for random partitions (default: 5).

For the block partition, the orientation specifications are abbreviations for "latitude" and "longitude", and they determine the order and orientations with which the block partitioning function creates the partition groups. For example, "lat_lon" will split the occurrence localities first by latitude, then by longitude. For the checkerboard partitions, the aggregation factor specifies how much to aggregate the existing cells in the envs raster to make new spatial partitions. For example, 'basic' checkerboard with an aggregation factor value of 2 will make squares 4 times larger than the input rasters and assign occurrence and background records to partition groups based on which square they fall in. Using two aggregation factors makes the checkerboard partitions hierarchical, where squares are first aggregated to define groups as in the 'basic' checkerboard, but a second aggregation is then made to separate the resulting two bins into four bins (see ?partitions for more details).

For user.eval, the variables you have access to in order to run your custom function are below. See the vignette for a worked example.
* enm - ENMdetails object
* occs.train.z - data frame: predictor variable values for training occurrences
* occs.val.z - data frame: predictor variable values for validation occurrences
* bg.train.z - data frame: predictor variable values for training background
* bg.val.z - data frame: predictor variable values for validation background
* mod.k - Model object for current partition (k)
* nk - numeric: number of folds (i.e., partitions)
* other.settings - named list: other settings specified in ENMevaluate()
* partitions - character: name of the partition method (e.g., "block")
* occs.train.pred - numeric: predictions made by mod.k for training occurrences
* occs.val.pred - numeric: predictions made by mod.k for validation occurrences
* bg.train.pred - numeric: predictions made by mod.k for training background
* bg.val.pred - numeric: predictions made by mod.k for validation background

Value

An ENMevaluation object. See ?ENMevaluation for details and description of the columns in the results table.

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**: 1198-1205. doi:10.1111/2041210X.12261

Warren, D. L., Glor, R. E., Turelli, M. & Funk, D. (2008) Environmental niche equivalency versus conservatism: quantitative approaches to niche evolution. *Evolution*, **62**: 2868-2883. doi:10.1111/j.15585646.2008.00482.x

Examples

```
## Not run:
library(terra)
library(ENMeval)

occs <- read.csv(file.path(system.file(package="predicts"),
"/ex/bradypus.csv"))[,2:3]
envs <- rast(list.files(path=paste(system.file(package="predicts"),
"/ex", sep=""), pattern="tif$", full.names=TRUE))
occs.z <- cbind(occs, extract(envs, occs, ID = FALSE))
occs.z$biome <- factor(occs.z$biome)
bg <- as.data.frame(predicts::backgroundSample(envs, n = 10000))
names(bg) <- names(occs)
bg.z <- cbind(bg, extract(envs, bg, ID = FALSE))
bg.z$biome <- factor(bg.z$biome)

# set other.settings -- pred.type is only for Maxent models
os <- list(abs.auc.diff = FALSE, pred.type = "cloglog",
validation.bg = "partition")
# set partition.settings -- here's an example for the block method
# see Details for the required settings for other partition methods
ps <- list(orientation = "lat_lat")

# here's a run with maxnet -- note the tune.args for feature classes (fc)
# and regularization multipliers (rm), as well as the designation of the
# categorical variable we are using (this can be a vector if multiple
# categorical variables are used)
e.maxnet <- ENMevaluate(occs, envs, bg,
tune.args = list(fc = c("L","LQ","LQH","H"), rm = 1:5),
partitions = "block", other.settings = os, partition.settings = ps,
algorithm = "maxnet", categoricals = "biome", overlap = TRUE)

# print the tuning results
eval.results(e.maxnet)

# you can plot the marginal response curves of a maxnet object with plot(),
# and you can also extract the data for plotting to make your own custom plots
mods.maxnet <- eval.models(e.maxnet)
m <- mods.maxnet$fc.LQH_rm.2
plot(m, type = "cloglog")
rcurve_data <- maxnet::response.plot(m, "bio1", type = "cloglog", plot = FALSE)

# there is currently no native function to make raster model predictions for
# maxnet models, but ENMeval can be used to make them like this:
# here's an example where we make a prediction based on the L2 model
# (feature class: Linear, regularization multiplier: 2) for our envs data
pred.LQH2 <- maxnet.predictRaster(m, envs)
plot(pred.L2)

# here's a run with maxent.jar -- note that if the R package rJava cannot
# install or load, or if you have other issues with Java on your computer,
# maxent.jar will not function
e.maxent.jar <- ENMevaluate(occs, envs, bg,
tune.args = list(fc = c("L","LQ","LQH","H"), rm = 1:5),
partitions = "block", other.settings = os, partition.settings = ps,
algorithm = "maxent.jar", categoricals = "biome", overlap = TRUE)

# here's a run of maxent.jar with a path specified for saving the html and
# plot files -- you can also turn on jackknife variable importance or
# response curves, etc., to have these plots saved there
e.maxent.jar <- ENMevaluate(occs, envs, bg,
tune.args = list(fc = c("L","LQ","LQH","H"), rm = 1:5),
partitions = "block", partition.settings = ps,
algorithm = "maxent.jar", categoricals = "biome", overlap = TRUE,
other.settings = list(path = "analyses/mxnt_results",
other.args = c("jackknife=TRUE", "responsecurves=TRUE")))

# print the tuning results
eval.results(e.maxent.jar)

# raster predictions can be made for maxent.jar models with predicts or
# ENMeval
mods.maxent.jar <- eval.models(e.maxent.jar)
pred.L2 <- predict(mods.maxent.jar$fc.L_rm.2, envs,
args = "outputform=cloglog")
pred.L2 <- maxnet.predictRaster(mods.maxent.jar$fc.L_rm.2, envs, os)
plot(pred.L2)

# this will give you the percent contribution (not deterministic) and
# permutation importance (deterministic) values of variable importance for
# Maxent models, and it only works with maxent.jar
eval.variable.importance(e.maxent.jar)

# here's a run with BIOCLIM. Note that we need to remove the categorical
# variable here because this algorithm only takes continuous variables. We
# also should point out that the way BIOCLIM is tuned is by comparing
# performance for different ways to make predictions (as opposed to comparing
# performance for models fit in different ways like for maxnet or maxent.jar).
# Namely, BIOCLIM can ignore different tails of the distribution when making
# predictions, and this is what is tuned in ENMevaluate (see
# ?predicts::envelope).

# print the tuning results
eval.results(e.bioclim)
# make raster predictions with predicts or ENMeval
mods.bioclim <- eval.models(e.bioclim)
# note: the models for low, high, and both are actually all the same, and
# the only difference for tuning is how they are predicted during
# cross-validation
pred.both <- predict(mods.bioclim$tails.both, envs, tails = "both")
plot(pred.both)

# please see the vignette for more examples of model tuning,
# partitioning, plotting functions, and null models
# https://jamiemkass.github.io/ENMeval/articles/ENMeval-2.0-vignette.html

## End(Not run)
```

`ENMevaluation-class` *ENMevaluation class*

Description

An S4 class that contains the ENMevaluate results.

Usage

```
## S4 method for signature 'ENMevaluation'
show(object)
```

Arguments

`object`	ENMevaluation object

Details

The following are brief descriptions of the columns in the results table, which prints when accessing ‘e@results‘ or ‘results(e)‘ if ‘e‘ is the ENMevaluation object. Those columns that represent evaluations of validation data (__.val.__) end in either "avg" (average of the metric across the models trained on withheld data during cross-validation) or "sd" (standard deviation of the metric across these models).
* fc = feature class
* rm = regularization multiplier
* tune.args = combination of arguments that define the complexity settings used for tuning (i.e., fc and rm for Maxent)
* auc.train = AUC calculated on the full dataset
* cbi.train = Continuous Boyce Index calculated on the full dataset
* auc.val = average/sd AUC calculated on the validation datasets (the data withheld during cross-validation)
* auc.diff = average/sd difference between auc.train and auc.val
* or.mtp = average/sd omission rate with threshold as the minimum suitability value across occurrence records
* or.10p = average/sd omission rate with threshold as the minimum suitability value across occurrence records after removing the lowest 10 cbi.val = average/sd Continuous Boyce Index calculated on the validation datasets (the data withheld during cross-validation)

* AICc = AIC corrected for small sample sizes
* delta.AICc = highest AICc value across all models minus this model's AICc value, where lower values mean higher performance and 0 is the highest performing model
* w.AIC = AIC weights, calculated by exp(-0.5 * delta.AIC), where higher values mean higher performance
* ncoef = number of non-zero beta values (model coefficients)

Slots

`algorithm` character: algorithm used

`tune.settings` data frame: settings that were tuned

`partition.method` character: partition method used

`partition.settings` list: partition settings used (i.e., value of *k* or aggregation factor)

`other.settings` list: other modeling settings used (i.e., decisions about clamping, AUC diff calculation)

`doClamp` logical: whether or not clamping was used

`clamp.directions` list: the clamping directions specified

`results` data frame: evaluation summary statistics

`results.partitions` data frame: evaluation k-fold statistics

`models` list: model objects

`variable.importance` list: variable importance data frames (when available)

`predictions` SpatRaster: model predictions

`taxon.name` character: the name of the focal taxon (optional)

`occs` data frame: occurrence coordinates and predictor variable values used for model training

`occs.testing` data frame: when provided, the coordinates of the fully-withheld testing records

`occs.grp` vector: partition groups for occurrence points

`bg` data frame: background coordinates and predictor variable values used for model training

`bg.grp` vector: partition groups for background points

`overlap` list: matrices of pairwise niche overlap statistics

`rmm` list: the rangeModelMetadata objects for each model

Author(s)

Jamie M. Kass, <`jamie.m.kass@gmail.com`>, Bob Muscarella, <`bob.muscarella@gmail.com`>

References

For references on performance metrics, see the following:

In general for ENMeval:

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**: 1198-1205. doi:10.1111/2041210X.12261

AUC

Fielding, A. H., & Bell, J. F. (1997). A review of methods for the assessment of prediction errors in conservation presence/absence models. *Environmental Conservation*, **24**: 38-49. doi:10.1017/S0376892997000088

Jimenez-Valverde, A. (2012). Insights into the area under the receiver operating characteristic curve (AUC) as a discrimination measure in species distribution modelling. *Global Ecology and Biogeography*, **21**: 498-507. doi:10.1111/j.14668238.2011.00683.x

AUC diff

Warren, D. L., Glor, R. E., Turelli, M. & Funk, D. (2008) Environmental niche equivalency versus conservatism: quantitative approaches to niche evolution. *Evolution*, **62**: 2868-2883. doi:10.1111/j.15585646.2008.00482.x

Radosavljevic, A., & Anderson, R. P. (2014). Making better Maxent models of species distributions: complexity, overfitting and evaluation. *Journal of Biogeography*, **41**(4), 629-643. doi:10.1111/jbi.12227

Omission rates

Radosavljevic, A., & Anderson, R. P. (2014). Making better Maxent models of species distributions: complexity, overfitting and evaluation. *Journal of Biogeography*, **41**(4), 629-643. doi:10.1111/jbi.12227

Continuous Boyce Index

Hirzel, A. H., Le Lay, G., Helfer, V., Randin, C., & Guisan, A. (2006). Evaluating the ability of habitat suitability models to predict species presences. *Ecological Modelling*, **199**: 142-152. doi:10.1016/j.ecolmodel.2006.05.017

`ENMevaluation_convert` *Convert old ENMevaluation objects to new ones*

Description

Converts ENMevaluation objects made with version <=0.3.1 to new ones made with version >=2.0.0.

Usage

```
ENMevaluation_convert(e, envs)
```

Arguments

`e`	ENMevaluation object: the old object to convert
`envs`	SpatRaster: the original predictor variables used to generate the old ENMevaluation object (these are used to make the new occs and bg slots which contain the predictor variable values)

Note

If bin.output was set to TRUE, `` `e@results` `` will be equivalent to the new results.partitions slot. Some slots are unable to be filled in because previous versions of ENMeval did not record them in ENMevaluation objects: variable.importance, partition.settings, other.settings, doClamp (set to TRUE arbitrarily to avoid errors, but may actually have been FALSE), clamp.directions, taxon.name, and rmm.

`enmeval_results` *Example ENMevaluation object.*

Description

An example ENMevaluation object produced after running ENMevaluate() with feature classes L, LQ, LQH, H and regularization multipliers 1 through 5. The SpatRaster data in the predictions slot is wrapped with ‘terra::wrap‘ to preserve the connections to the raster data. To use it, do the following: ‘e@predictions <- terra::unwrap(e@predictions)‘.

Usage

```
enmeval_results
```

Format

An ENMevaluation object

`ENMnull-class` *ENMnull class*

Description

An S4 class that contains the ENMnulls results.

Usage

```
## S4 method for signature 'ENMnull'
show(object)
```

Arguments

`object`	ENMnull object

Slots

`null.algorithm` character: algorithm used

`null.mod.settings` data frame: model settings used

`null.partition.method` character: partition method used

`null.partition.settings` list: partition settings used (i.e., value of *k* or aggregation factor)

`null.doClamp` logical: whether to clamp model predictions or not

`null.other.settings` list: other modeling settings used (i.e., decisions about clamping, AUC diff calculation)

`null.no.iter` numeric: number of null model iterations

`null.results` data frame: evaluation summary statistics for null models

`null.results.partitions` data frame: evaluation k-fold statistics for null models

`null.emp.results` data frame: evaluation summary statistics for the empirical model, means for all null models, z-scores, and p-values

`emp.occs` data frame: occurrence coordinates and predictor variable values used for model training (empirical model)

`emp.occs.grp` vector: partition groups for occurrence points (empirical model)

`emp.bg` data frame: background coordinates and predictor variable values used for model training (empirical model)

`emp.bg.grp` vector: partition groups for background points (empirical model)

Author(s)

Jamie M. Kass, <`jamie.m.kass@gmail.com`>, Corentin Bohl, <`corentinbohl@gmail.com`>

`ENMnulls` *Generate null ecological niche models (ENMs) and compare null with empirical performance metrics*

Description

`ENMnulls()` iteratively builds null ENMs for a single set of user-specified model settings based on an input ENMevaluation object, from which all other analysis settings are extracted. Summary statistics of the performance metrics for the null ENMs are taken (averages and standard deviations) and effect sizes and *p*-values are calculated by comparing these summary statistics to the empirical values of the performance metrics (i.e., from the model built with the empirical data). See the references below for more details on this method.

Usage

```
ENMnulls(
  e,
  mod.settings,
  no.iter,
  eval.stats = c("auc.val", "auc.diff", "cbi.val", "or.mtp", "or.10p"),
  user.enm = NULL,
  user.eval = NULL,
  user.eval.type = NULL,
  userStats.signs = NULL,
  removeMxTemp = TRUE,
  parallel = FALSE,
  numCores = NULL,
  quiet = FALSE
)
```

Arguments

`e`	ENMevaluation object
`mod.settings`	named list: one set of model settings with which to build null ENMs.
`no.iter`	numeric: number of null model iterations.
`eval.stats`	character vector: the performance metrics that will be used to calculate null model statistics.
`user.enm`	ENMdetails object: if implementing a user-specified model.
`user.eval`	function: custom function for specifying performance metrics not included in **ENMeval**. The function must first be defined and then input as the argument `user.eval`.
`user.eval.type`	character: if implementing a user-specified model, specify here which evaluation type to use – either "knonspatial", "kspatial", "testing", or "none".
`userStats.signs`	named list: user-defined evaluation statistics attributed with either 1 or -1 to designate whether the expected difference between empirical and null models is positive or negative; this is used to calculate the p-value of the z-score. For example, for AUC, the difference should be positive (the empirical model should have a higher score), whereas for omission rate it should be negative (the empirical model should have a lower score).
`removeMxTemp`	boolean: if TRUE, delete all temporary data generated when using maxent.jar for modeling.
`parallel`	boolean: if TRUE, use parallel processing.
`numCores`	numeric: number of cores to use for parallel processing; if NULL, all available cores will be used.
`quiet`	boolean: if TRUE, silence all function messages (but not errors).

Details

This null ENM technique is based on the implementation in Bohl *et al.* (2019), which follows the original methodology of Raes & ter Steege (2007) but makes an important modification: instead of evaluating each null model on random validation data, here we evaluate the null models on the same withheld validation data used to evaluate the empirical model. Bohl *et al.* (2019) demonstrates this approach using a single defined withheld partition group, but Kass *et al.* (2020) extended it to use spatial partitions by drawing null occurrences from the area of the predictor raster data defining each partition. Please see the vignette for a brief example.

This function avoids using raster data to speed up each iteration, and instead samples null occurrences from the partitioned background records. Thus, you should avoid running this when your background records are not well sampled across the study extent, as this limits the extent that null occurrences can be sampled from.

Value

An `ENMnull` object with slots containing evaluation summary statistics for the null models and their cross-validation results, as well as differences in results between the empirical and null models. This comparison table includes z-scores of these differences and their associated p-values (under a normal distribution). See ?ENMnull for more details.

References

Bohl, C. L., Kass, J. M., & Anderson, R. P. (2019). A new null model approach to quantify performance and significance for ecological niche models of species distributions. *Journal of Biogeography*, **46**: 1101-1111. doi:10.1111/jbi.13573

Kass, J. M., Anderson, R. P., Espinosa-Lucas, A., Juarez-Jaimes, V., Martinez-Salas, E., Botello, F., Tavera, G., Flores-Martinez, J. J., & Sanchez-Cordero, V. (2020). Biotic predictors with phenological information improve range estimates for migrating monarch butterflies in Mexico. *Ecography*, **43**: 341-352. doi:10.1111/ecog.04886

Raes, N., & ter Steege, H. (2007). A null-model for significance testing of presence-only species distribution models. *Ecography*, **30**: 727-736. doi:10.1111/j.2007.09067590.05041.x

Examples

```
## Not run:
library(ENMeval)

# first, let's tune some models
occs <- read.csv(file.path(system.file(package="predicts"),
"/ex/bradypus.csv"))[,2:3]
envs <- rast(list.files(path=paste(system.file(package="predicts"),
                                    "/ex", sep=""), pattern="tif$",
                                    full.names=TRUE))
bg <- as.data.frame(predicts::backgroundSample(envs, n = 10000))
names(bg) <- names(occs)

ps <- list(orientation = "lat_lat")

# as an example, let's use two user-specified evaluation metrics
```

```
conf.and.cons <- function(vars) {
  observations <- c(
    rep(x = 1, times = length(vars$occs.train.pred)),
    rep(x = 0, times = length(vars$bg.train.pred)),
    rep(x = 1, times = length(vars$occs.val.pred)),
    rep(x = 0, times = length(vars$bg.val.pred))
  )
  predictions <- c(vars$occs.train.pred, vars$bg.train.pred,
  vars$occs.val.pred, vars$bg.val.pred)
  evaluation_mask <- c(
    rep(x = FALSE, times = length(vars$occs.train.pred) +
    length(vars$bg.train.pred)),
    rep(x = TRUE, times = length(vars$occs.val.pred) +
    length(vars$bg.val.pred))
  )
  measures <- confcons::measures(observations = observations,
  predictions = predictions,
  evaluation_mask = evaluation_mask, df = TRUE)
  measures.metrics <- measures[, c("CPP_eval", "DCPP")]
  colnames(measures.metrics) <- c("confidence", "consistency")
  return(measures.metrics)
}

e <- ENMevaluate(occs, envs, bg,
                 tune.args = list(fc = c("L","LQ","LQH"), rm = 2:4),
                 partitions = "block", partition.settings = ps,
                 algorithm = "maxnet", categoricals = "biome",
                 user.eval  = conf.and.cons, parallel = TRUE)

d <- eval.results(e)

# here, we will choose an optimal model based on validation CBI, but you can
# choose yourself what evaluation statistics to use
opt <- d |> filter(cbi.val.avg == max(cbi.val.avg))

# now we can run our null models, and we can specify to include estimates for
# our user-specified variables too, but we need to make sure we note what
# sign we expect these statistics to be
# NOTE: you should use at least 100 iterations in practice -- this is just an
# example
nulls <- ENMnulls(e,
                  mod.settings = list(fc = opt$fc, rm = opt$rm),
                  no.iter = 10,
                  user.eval = conf.and.cons,
                  eval.stats = c("cbi.val", "confidence", "consistency"),
                  userStats.signs = c("confidence" = 1, "consistency" = 1))

# here are the results of all the null iterations
null.results(nulls)
# and here are the comparisons between the null and empirical values for
# the evaluation statistics, including the z-score and p-value
# for more details, see Bohl et al. 2019
null.emp.results(nulls)
```

```
## End(Not run)
```

`eval.algorithm` *eval.algorithm generic for ENMevaluation object*

Description

eval.algorithm generic for ENMevaluation object

Usage

```
eval.algorithm(x)

## S4 method for signature 'ENMevaluation'
eval.algorithm(x)
```

Arguments

`x`	ENMevaluation object

`eval.bg` *eval.bg generic for ENMevaluation object*

Description

eval.bg generic for ENMevaluation object

Usage

```
eval.bg(x)

## S4 method for signature 'ENMevaluation'
eval.bg(x)
```

Arguments

`x`	ENMevaluation object

eval.bg.grp *eval.bg.grp generic for ENMevaluation object*

Description

eval.bg.grp generic for ENMevaluation object

Usage

```
eval.bg.grp(x)

## S4 method for signature 'ENMevaluation'
eval.bg.grp(x)
```

Arguments

x	ENMevaluation object

eval.clamp.directions *eval.clamp.directions generic for ENMevaluation object*

Description

eval.clamp.directions generic for ENMevaluation object

Usage

```
eval.clamp.directions(x)

## S4 method for signature 'ENMevaluation'
eval.clamp.directions(x)
```

Arguments

x	ENMevaluation object

`eval.doClamp` *eval.doClamp generic for ENMevaluation object*

Description

eval.doClamp generic for ENMevaluation object

Usage

```
eval.doClamp(x)

## S4 method for signature 'ENMevaluation'
eval.doClamp(x)
```

Arguments

`x`	ENMevaluation object

`eval.models` *eval.models generic for ENMevaluation object*

Description

eval.models generic for ENMevaluation object

Usage

```
eval.models(x)

## S4 method for signature 'ENMevaluation'
eval.models(x)

## S4 method for signature 'ENMevaluation'
eval.variable.importance(x)
```

Arguments

`x`	ENMevaluation object

eval.occs *eval.occs generic for ENMevaluation object*

Description

eval.occs generic for ENMevaluation object

Usage

```
eval.occs(x)

## S4 method for signature 'ENMevaluation'
eval.occs(x)
```

Arguments

x	ENMevaluation object

eval.occs.grp *eval.occs.grp generic for ENMevaluation object*

Description

eval.occs.grp generic for ENMevaluation object

Usage

```
eval.occs.grp(x)

## S4 method for signature 'ENMevaluation'
eval.occs.grp(x)
```

Arguments

x	ENMevaluation object

`eval.occs.testing` *eval.occs.testing generic for ENMevaluation object*

Description

eval.occs.testing generic for ENMevaluation object

Usage

```
eval.occs.testing(x)

## S4 method for signature 'ENMevaluation'
eval.occs.testing(x)
```

Arguments

`x`	ENMevaluation object

`eval.other.settings` *eval.other.settings generic for ENMevaluation object*

Description

eval.other.settings generic for ENMevaluation object

Usage

```
eval.other.settings(x)

## S4 method for signature 'ENMevaluation'
eval.other.settings(x)
```

Arguments

`x`	ENMevaluation object

eval.overlap *eval.overlap generic for ENMevaluation object*

Description

eval.overlap generic for ENMevaluation object

Usage

```
eval.overlap(x)

## S4 method for signature 'ENMevaluation'
eval.overlap(x)
```

Arguments

x	ENMevaluation object

eval.partition.method *eval.partition.method generic for ENMevaluation object*

Description

eval.partition.method generic for ENMevaluation object

Usage

```
eval.partition.method(x)

## S4 method for signature 'ENMevaluation'
eval.partition.method(x)
```

Arguments

x	ENMevaluation object

`eval.partition.settings` *eval.partition.settings generic for ENMevaluation object*

Description

eval.partition.settings generic for ENMevaluation object

Usage

```
eval.partition.settings(x)

## S4 method for signature 'ENMevaluation'
eval.partition.settings(x)
```

Arguments

`x`	ENMevaluation object

`eval.predictions` *eval.predictions generic for ENMevaluation object*

Description

eval.predictions generic for ENMevaluation object

Usage

```
eval.predictions(x)

## S4 method for signature 'ENMevaluation'
eval.predictions(x)
```

Arguments

`x`	ENMevaluation object

`eval.results` *eval.results generic for ENMevaluation object*

Description

eval.results generic for ENMevaluation object

Usage

```
eval.results(x)

## S4 method for signature 'ENMevaluation'
eval.results(x)
```

Arguments

`x`	ENMevaluation object

`eval.results.partitions` *eval.results.partitions generic for ENMevaluation object*

Description

eval.results.partitions generic for ENMevaluation object

Usage

```
eval.results.partitions(x)

## S4 method for signature 'ENMevaluation'
eval.results.partitions(x)
```

Arguments

`x`	ENMevaluation object

`eval.rmm` *eval.rmm generic for ENMevaluation object*

Description

eval.rmm generic for ENMevaluation object

Usage

```
eval.rmm(x)

## S4 method for signature 'ENMevaluation'
eval.rmm(x)
```

Arguments

`x`	ENMevaluation object

`eval.taxon.name` *eval.taxon.name generic for ENMevaluation object*

Description

eval.taxon.name generic for ENMevaluation object

Usage

```
eval.taxon.name(x)

## S4 method for signature 'ENMevaluation'
eval.taxon.name(x)
```

Arguments

`x`	ENMevaluation object

`eval.tune.settings` *eval.tune.settings generic for ENMevaluation object*

Description

eval.tune.settings generic for ENMevaluation object

Usage

```
eval.tune.settings(x)

## S4 method for signature 'ENMevaluation'
eval.tune.settings(x)
```

Arguments

`x`	ENMevaluation object

`eval.variable.importance` *eval.variable.importance (variable importance) generic for ENMevaluation object*

Description

eval.variable.importance (variable importance) generic for ENMevaluation object

Usage

```
eval.variable.importance(x)
```

Arguments

`x`	ENMevaluation object

`evalplot.envSim.hist` *Similarity histogram plots for partition groups*

Description

Plots environmental similarity of reference partitions (occurrences or background) to remaining data (occurrence and background for all other partitions). This function does not use raster data, and thus only calculates similarity values for data used in model training. Further, this function does not calculate similarity for categorical variables.

Usage

```
evalplot.envSim.hist(
  e = NULL,
  occs.z = NULL,
  bg.z = NULL,
  occs.grp = NULL,
  bg.grp = NULL,
  ref.data = "occs",
  envs.vars = NULL,
  occs.testing.z = NULL,
  hist.bins = 30,
  return.tbl = FALSE,
  quiet = FALSE
)
```

Arguments

`e`	ENMevaluation object
`occs.z`	data frame: longitude, latitude, and environmental predictor variable values for occurrence records, in that order (optional); the first two columns must be named "longitude" and "latitude"
`bg.z`	data frame: longitude, latitude, and environmental predictor variable values for background records, in that order (optional); the first two columns must be named "longitude" and "latitude"
`occs.grp`	numeric vector: partition groups for occurrence records (optional)
`bg.grp`	numeric vector: partition groups for background records (optional)
`ref.data`	character: the reference to calculate MESS based on occurrences ("occs") or background ("bg"), with default "occs" these must be specified as this function was intended for use with continuous data only; these must be specified when inputting tabular data instead of an ENMevaluation object
`envs.vars`	character vector: names of a predictor variable to plot similarities for; if left NULL, calculations are done with respect to all variables (optional)
`occs.testing.z`	data frame: longitude, latitude, and environmental predictor variable values for fully withheld testing records, in that order; this is for use only with the "testing" partition option when an ENMevaluation object is not input (optional)

`hist.bins`	numeric: number of histogram bins for histogram plots; default is 30
`return.tbl`	boolean: if TRUE, return the data frames of similarity values used to make the ggplot instead of the plot itself
`quiet`	boolean: if TRUE, silence all function messages (but not errors)

Details

When fully withheld testing groups are used, make sure to input either an ENMevaluation object or the argument occs.testing.z. In the resulting plot, partition 1 refers to the training data, while partition 2 refers to the fully withheld testing group.

Histograms are plotted showing the environmental similarity estimates for each partition group in relation to the others. The similarity between environmental values associated with the validation occurrence or background records per partition group and those associated with the remaining data (training occurrences and background) are calculated with the MESS algorithm, and the minimum similarity per grid cell is returned. Higher negative values indicate a greater environmental difference between the validation occurrences and the study extent, and higher positive values indicate greater similarity. This function uses the ‘mess()‘ function from the package ‘predicts‘. Please see the below reference for details on MESS.

Value

A ggplot of environmental similarities between the occurrence or background data for each partition and the rest of the data (all other occurrences and background data).

References

Baumgartner J, Wilson P (2021). _rmaxent: Tools for working with Maxent in R_. R package version 0.8.5.9000, <URL: https://github.com/johnbaums/rmaxent>. Elith, J., Kearney, M., and Phillips, S. (2010) The art of modelling range-shifting species. *Methods in Ecology and Evolution*, **1**: 330-342. doi:10.1111/j.2041210X.2010.00036.x

Examples

```
## Not run:
# first, let's tune some models
occs <- read.csv(file.path(system.file(package="predicts"),
"/ex/bradypus.csv"))[,2:3]
envs <- rast(list.files(path=paste(system.file(package="predicts"),
"/ex", sep=""), pattern="tif$", full.names=TRUE))
bg <- as.data.frame(predicts::backgroundSample(envs, n = 10000))
names(bg) <- names(occs)

ps <- list(orientation = "lat_lat")

e <- ENMevaluate(occs, envs, bg,
               tune.args = list(fc = c("L","LQ","LQH"), rm = 1:5),
               partitions = "block", partition.settings = ps,
               algorithm = "maxnet", categoricals = "biome",
               parallel = TRUE)
```

```
# now, plot the environmental similarity of each partition to the others
evalplot.envSim.hist(e)

## End(Not run)
```

---

evalplot.envSim.map *Similarity maps for partition groups*

Description

Maps environmental similarity of reference partitions (occurrences or background) to all cells with values in the predictor variable rasters. This function uses raster data, and thus cannot map similarity values using only tables of environmental values f or occurrences or background. Further, this function does not calculate similarity for categorical variables.

Usage

```
evalplot.envSim.map(
  e = NULL,
  envs,
  occs.z = NULL,
  bg.z = NULL,
  occs.grp = NULL,
  bg.grp = NULL,
  ref.data = "occs",
  envs.vars = NULL,
  bb.buf = 0,
  occs.testing.z = NULL,
  plot.bg.pts = FALSE,
  sim.palette = NULL,
  pts.size = 1.5,
  gradient.colors = c("red", "white", "blue"),
  na.color = "gray",
  return.tbl = FALSE,
  return.ras = FALSE,
  quiet = FALSE
)
```

Arguments

`e`	ENMevaluation object (optional)
`envs`	SpatRaster: environmental predictor variables used to build the models in "e"; categorical variables will be removed before internally as they cannot be used to calculate MESS
`occs.z`	data frame: longitude, latitude, and environmental predictor variable values for occurrence records, in that order (optional); the first two columns must be named "longitude" and "latitude"

`bg.z`	data frame: longitude, latitude, and environmental predictor variable values for background records, in that order (optional); the first two columns must be named "longitude" and "latitude"
`occs.grp`	numeric vector: partition groups for occurrence records (optional)
`bg.grp`	numeric vector: partition groups for background records (optional)
`ref.data`	character: the reference to calculate MESS based on occurrences ("occs") or background ("bg"), with default "occs"
`envs.vars`	character vector: names of a predictor variable to plot similarities for; if left NULL, calculations are done with respect to all variables (optional)
`bb.buf`	numeric: distance used to buffer (extend) the mapping extent in map units; for latitude/longitude, this is in degrees (optional)
`occs.testing.z`	data frame: longitude, latitude, and environmental predictor variable values for fully withheld testing records, in that order; this is for use only with the "testing" partition option when an ENMevaluation object is not input (optional)
`plot.bg.pts`	boolean: if TRUE, plot background points when using ref.data = "bg"
`sim.palette`	character: RColorBrewer palette name to use for plotting discrete variables; if NULL, default is "Set1"
`pts.size`	numeric: custom point size for ggplot
`gradient.colors`	character vector: colors used for ggplot2::scale_fill_gradient2
`na.color`	character: color used for NA values
`return.tbl`	boolean: if TRUE, return the data frames of similarity values used to make the ggplot instead of the plot itself
`return.ras`	boolean: if TRUE, return the SpatRaster of similarity values used to make the ggplot instead of the plot itself
`quiet`	boolean: if TRUE, silence all function messages (but not errors)

Details

When fully withheld testing groups are used, make sure to input either an ENMevaluation object or the argument occs.testing.z. In the resulting plot, partition 1 refers to the training data, while partition 2 refers to the fully withheld testing group.

Rasters are plotted showing the environmental similarity estimates for each partition group in relation to the others. The similarity between environmental values associated with the validation occurrence or background records per partition group and those associated with the entire study extent (specified by the extent of the input SpatRaster "envs") are calculated with the MESS algorithm, and the minimum similarity per grid cell is returned. Higher negative values indicate greater environmental difference between the validation occurrences and the study extent, and higher positive values indicate greater similarity. This function uses the ‘mess()‘ function from the package ‘predicts‘ to calculate the similarities. Please see the below reference for details on MESS.

Value

A ggplot of environmental similarities between the occurrence or background data for each partition and all predictor variable values in the extent.

References

Baumgartner J, Wilson P (2021). _rmaxent: Tools for working with Maxent in R_. R package version 0.8.5.9000, <URL: https://github.com/johnbaums/rmaxent>. Elith, J., Kearney, M., and Phillips, S. (2010) The art of modelling range-shifting species. *Methods in Ecology and Evolution*, **1**: 330-342. doi:10.1111/j.2041210X.2010.00036.x

Examples

```
## Not run:
library(terra)
library(ENMeval)

# first, let's tune some models
occs <- read.csv(file.path(system.file(package="predicts"),
"/ex/bradypus.csv"))[,2:3]
envs <- rast(list.files(path=paste(system.file(package="predicts"),
"/ex", sep=""), pattern="tif$", full.names=TRUE))
bg <- as.data.frame(predicts::backgroundSample(envs, n = 10000))
names(bg) <- names(occs)

ps <- list(orientation = "lat_lat")

e <- ENMevaluate(occs, envs, bg,
               tune.args = list(fc = c("L","LQ","LQH"), rm = 1:5),
               partitions = "block", partition.settings = ps,
               algorithm = "maxnet", categoricals = "biome",
               parallel = TRUE)

# now, plot the environmental similarity of each partition to the others
evalplot.envSim.map(e, envs)

## End(Not run)
```

---

evalplot.grps *Partition group plots*

Description

Plot occurrence partition groups over an environmental predictor raster.

Usage

```
evalplot.grps(
  e = NULL,
  envs,
  pts = NULL,
  pts.grp = NULL,
  ref.data = "occs",
  pts.size = 1.5,
  return.tbl = FALSE
)
```

Arguments

e	ENMevaluation object
envs	SpatRaster: environmental predictor variable used to build the models in "e"
pts	matrix / data frame: coordinates for occurrence or background data
pts.grp	numeric vector: partition groups corresponding to data in "pts"
ref.data	character: plot occurrences ("occs") or background ("bg"), with default "occs"
pts.size	numeric: custom point size for ggplot
return.tbl	boolean: if TRUE, return the data frames used to make the ggplot instead of the plot itself

Details

This function serves as a quick way to visualize occurrence or background partitions over the extent of an environmental predictor raster. It can be run with an existing ENMevaluation object, or alternatively with occurrence or background coordinates and the corresponding partitions.

Examples

```
## Not run:
library(terra)
library(ENMeval)
occs <- read.csv(file.path(system.file(package="predicts"),
                           "/ex/bradypus.csv"))[,2:3]
envs <- rast(list.files(path=paste(system.file(package="predicts"),
                                   "/ex", sep=""), pattern="tif$", full.names=TRUE))
bg <- as.data.frame(predicts::backgroundSample(envs, n = 10000))
names(bg) <- names(occs)

parts <- get.block(occs, bg, orientation = "lat_lon")

# now, plot the partition groups for occurrence and background points
evalplot.grps(envs = envs, pts = occs, pts.grp = parts$occs.grp)
evalplot.grps(envs = envs, pts = bg, pts.grp = parts$bg.grp)

# you can also plot with an ENMevaluation object
ps <- list(orientation = "lat_lon")
e <- ENMevaluate(occs, envs, bg,
                 tune.args = list(fc = c("L","LQ"), rm = 1:3),
                 partitions = "block", partition.settings = ps,
                 algorithm = "maxnet", categoricals = "biome",
                 parallel = TRUE)

evalplot.grps(e = e, envs = envs, ref.data = "occs")
evalplot.grps(e = e, envs = envs, ref.data = "bg")
```

```
## End(Not run)
```

`evalplot.nulls` *ENMnulls statistics plot*

Description

Plot histogram of evaluation statistics for null ENM simulations

Usage

```
evalplot.nulls(
  e.null,
  stats,
  plot.type,
  facet.labels = NULL,
  metric.levels = NULL,
  return.tbl = FALSE
)
```

Arguments

`e.null`	ENMnull object
`stats`	character vector: metrics from results table to be plotted; examples are "auc.val" or "or.10p"; if more than one statistic is specified, the histogram plot will be faceted
`plot.type`	character: either "violin" or "histogram"
`facet.labels`	named list: custom names for the metric facets, in the form list(old_name = "new_name", ...)
`metric.levels`	character vector: custom factor levels for metrics; this controls the order that metric statistics are plotted
`return.tbl`	boolean: if TRUE, return the data frames of null results used to make the ggplot instead of the plot itself

Details

There are two variations for this plot, but both show null quantiles (0.01, 0.05, 0.5, 0.95, 0.99). For violin plots, the null distribution is displayed as a vertical shape (i.e., the violin) with horizontal lines showing the quantiles and the empirical value is plotted as a red point along the vertical axis. For histogram plots, the null distribution is displayed as a histogram with vertical lines showing the quantiles and the empirical value as a vertical red line on the distribution.

Value

A ggplot of null model statistics.

Examples

```
## Not run:
# first, let's tune some models
occs <- read.csv(file.path(system.file(package="predicts"),
                           "/ex/bradypus.csv"))[,2:3]
envs <- rast(list.files(path=paste(system.file(package="predicts"),
                                   "/ex", sep=""), pattern="tif$", full.names=TRUE))
bg <- as.data.frame(predicts::backgroundSample(envs, n = 10000))
names(bg) <- names(occs)

ps <- list(orientation = "lat_lat")

e <- ENMevaluate(occs, envs, bg,
                tune.args = list(fc = c("L","LQ","LQH"), rm = 2:4), partitions = "block",
                 partition.settings = ps, algorithm = "maxnet", categoricals = "biome",
                 parallel = TRUE)

d <- eval.results(e)

# here, we will choose an optimal model based on validation CBI, but you can
# choose yourself what evaluation statistics to use
opt <- d |> filter(cbi.val.avg == max(cbi.val.avg))

# now we can run our null models
# NOTE: you should use at least 100 iterations in practice -- this is just an
# example
nulls <- ENMnulls(e, mod.settings = list(fc = opt$fc, rm = opt$rm), no.iter = 10)

# let's plot the null model results
evalplot.nulls(nulls, stats = c("cbi.val", "auc.val"), plot.type = "violin")

## End(Not run)
```

`evalplot.stats` *ENMevaluation statistics plot*

Description

Plot evaluation statistics over tuning parameter ranges to visualize differences in performance

Usage

```
evalplot.stats(
  e,
  stats,
  x.var,
  color.var,
  dodge = NULL,
  error.bars = TRUE,
  facet.labels = NULL,
  metric.levels = NULL,
  return.tbl = FALSE
)
```

Arguments

`e`	ENMevaluation object
`stats`	character vector: names of statistics from results table to be plotted; if more than one statistic is specified, the plot will be faceted
`x.var`	character: variable to be plotted on x-axis
`color.var`	character: variable used to assign symbology colors
`dodge`	numeric: dodge width for points and lines; this improves visibility when there is high overlap (optional)
`error.bars`	boolean: if TRUE, plot error bars
`facet.labels`	character vector: custom names for the metric facets
`metric.levels`	character vector: custom factor levels for metrics; this controls the order that metric statistics are plotted
`return.tbl`	boolean: if TRUE, return the data frames of results used to make the ggplot instead of the plot itself

Details

In this plot, the x-axis represents a tuning parameter range, while the y-axis represents the average of a statistic over all partitions. Different colors represent the categories or values of another tuning parameter. Error bars represent the standard deviation of a statistic around the mean. Currently, this function can only plot two tuning parameters at a time.

Value

A ggplot of evaluation statistics.

Examples

```
## Not run:
# first, let's tune some models
occs <- read.csv(file.path(system.file(package="predicts"),
"/ex/bradypus.csv"))[,2:3]
envs <- rast(list.files(path=paste(system.file(package="predicts"),
"/ex", sep=""), pattern="tif$", full.names=TRUE))
bg <- as.data.frame(predicts::backgroundSample(envs, n = 10000))
names(bg) <- names(occs)

ps <- list(orientation = "lat_lat")
e <- ENMevaluate(occs, envs, bg,
              tune.args = list(fc = c("L","LQ","LQH"), rm = 1:5),
              partitions = "block", partition.settings = ps,
                algorithm = "maxnet", categoricals = "biome",
                parallel = TRUE)

evalplot.stats(e, c("cbi.val", "or.mtp"), x.var = "rm", color.var = "fc",
             error.bars = FALSE)

## End(Not run)
```

`loadENMevaluation` *Load ENMevaluation object*

Description

Load an ENMevaluation object as an .rds file. This is necessary to use instead of `readRDS()` because wrapped terra SpatRasters require `unwrap()` after loading for the raster data. This convenience function does that for you.

Usage

```
loadENMevaluation(filename)
```

Arguments

`filename`	character: path to the .rds file to load

`lookup.enm` *Look up ENMdetails abject*

Description

Internal function to look up ENMdetails objects.

Usage

```
lookup.enm(algorithm)
```

Arguments

`algorithm`	character: algorithm name (must be implemented as ENMdetails object)

`maxentJARversion` *Look up version of maxent.jar*

Description

Internal function to look up the version of the maxent.jar being used.

Usage

```
maxentJARversion()
```

`maxnet.predictRaster` *Predict raster for maxnet*

Description

As maxnet does not have native functionality for making prediction rasters, this function does it. It is a wrapper for the internal enm.maxnet@predict function, which is not really intended for use outside the package.

Usage

```
maxnet.predictRaster(mod, envs, pred.type = "cloglog", doClamp = TRUE, ...)
```

Arguments

`mod`	maxnet object
`envs`	SpatRaster
`pred.type`	character: the type of maxnet prediction to make; the default is "cloglog"
`doClamp`	Boolean: whether to clamp predictions or not
`...`	any additional parameters

null.algorithm — *null.algorithm generic for ENMnull object*

Description

null.algorithm generic for ENMnull object

Usage

```
null.algorithm(x)

## S4 method for signature 'ENMnull'
null.algorithm(x)
```

Arguments

x	ENMnull object

null.doClamp — *null.doClamp generic for ENMnull object*

Description

null.doClamp generic for ENMnull object

Usage

```
null.doClamp(x)

## S4 method for signature 'ENMnull'
null.doClamp(x)
```

Arguments

x	ENMnull object

`null.emp.results` *null.emp.results generic for ENMnull object*

Description

null.emp.results generic for ENMnull object

Usage

```
null.emp.results(x)

## S4 method for signature 'ENMnull'
null.emp.results(x)
```

Arguments

`x`	ENMnull object

`null.mod.settings` *null.mod.settings generic for ENMnull object*

Description

null.mod.settings generic for ENMnull object

Usage

```
null.mod.settings(x)

## S4 method for signature 'ENMnull'
null.mod.settings(x)
```

Arguments

`x`	ENMnull object

`null.no.iter` *null.no.iter generic for ENMnull object*

Description

null.no.iter generic for ENMnull object

Usage

```
null.no.iter(x)

## S4 method for signature 'ENMnull'
null.no.iter(x)
```

Arguments

`x`	ENMnull object

`null.other.settings` *null.other.settings generic for ENMnull object*

Description

null.other.settings generic for ENMnull object

Usage

```
null.other.settings(x)

## S4 method for signature 'ENMnull'
null.other.settings(x)
```

Arguments

`x`	ENMnull object

`null.partition.method` *null.partition.method generic for ENMnull object*

Description

null.partition.method generic for ENMnull object

Usage

```
null.partition.method(x)

## S4 method for signature 'ENMnull'
null.partition.method(x)
```

Arguments

`x`	ENMnull object

`null.partition.settings` *null.partition.settings generic for ENMnull object*

Description

null.partition.settings generic for ENMnull object

Usage

```
null.partition.settings(x)

## S4 method for signature 'ENMnull'
null.partition.settings(x)
```

Arguments

`x`	ENMnull object

`null.results` *null.results generic for ENMnull object*

Description

null.results generic for ENMnull object

Usage

```
null.results(x)

## S4 method for signature 'ENMnull'
null.results(x)
```

Arguments

`x`	ENMnull object

`null.results.partitions` *null.results.partitions generic for ENMnull object*

Description

null.results.partitions generic for ENMnull object

Usage

```
null.results.partitions(x)

## S4 method for signature 'ENMnull'
null.results.partitions(x)
```

Arguments

`x`	ENMnull object

parse_lambdas *Parse Maxent lambdas information*

Description

NOTICE: This function was borrowed from the rmaxent package written by John Baumgartner (https://github.com/johnbaums/rmaxent/). (dependencies on Github-only packages are not allowed for CRAN).

Parse Maxent .lambdas files to extract the types, weights, minima and maxima of features, as well as the fitted model's entropy and other values required for predicting to new data.

Usage

```
parse_lambdas(lambdas)
```

Arguments

`lambdas`	Either a ‘MaxEnt‘ fitted model object (fitted with the ‘maxent‘ function in the ‘dismo‘ package), or a file path to a Maxent .lambdas file.

Value

A list (of class ‘lambdas‘) with five elements: * ‘lambdas‘: a ‘data.frame‘ describing the features used in a Maxent model, including their weights (lambdas), maxima, minima, and type; * ‘linearPredictorNormalizer‘: a constant that ensures the linear predictor (the sum of clamped features multiplied by their respective feature weights) is always negative (for numerical stability); * ‘densityNormalizer‘: a scaling constant that ensures Maxent's raw output sums to 1 over background points; * ‘numBackgroundPoints‘: the number of background points used in model training; and * ‘entropy‘: the entropy of the fitted model.

References

* Wilson, P. W. (2009) [_Guidelines for computing MaxEnt model output values from a lambdas file_](http://gis.humboldt.edu/OLM/Courses/GSP_570/Learning%20Modules/10%20BlueSpray_Maxent_Uncertinaty/MaxI
* _Maxent software for species habitat modeling, version 3.3.3k_ help file (software freely available [here](https://www.cs.princeton.edu/~schapire/maxent/)).

Examples

```
## Not run:
# Below we use the predicts::MaxEnt example to fit a model:
library(predicts)
occs <- read.csv(file.path(system.file(package="predicts"),
"/ex/bradypus.csv"))[,2:3]
predictors <- rast(file.path(system.file(package='predicts'), '/ex/bio.tif'))
me <- MaxEnt(predictors, occs)
# ... and then parse the lambdas information:
lam <- parse_lambdas(me)
```

```
lam
str(lam, 1)

## End(Not run)
```

`partitions` *Methods to partition data for evaluation*

Description

ENMeval provides several ways to partition occurrence and background localities into bins for training and validation (or, evaluation and calibration). Users should carefully consider the objectives of their study and the influence of spatial bias when deciding on a method of data partitioning.

These functions are used internally to partition data during a call of `ENMevaluate` but can also be used independently to generate data partitions. For user-specified partitions, users can simply define groups of occurrence records and background points directly with `ENMevaluate`.

The `get.block` method partitions occurrence localities by finding the latitude and/or longitude lines that divide the occurrence localities into four groups of (insofar as possible) equal numbers. The order and nature of the divisions can be controlled with the "orientation" parameter. The default is "lat_lon", which divides first by a latitudinal line, then second by longitudinal lines. This method is based on the spatial partitioning technique described in Radosavljevic & Anderson (2014), where the "lon_lon" option was used. Background localities are assigned to each of the four groups based on their position with respect to these lines. While the `get.block` method results in (approximately) equal division of occurrence localities among four groups, the number of background localities (and, consequently, environmental and geographic space) in each group depends on the distribution of occurrence localities across the study area.

The `get.checkerboard` methods are variants of a checkerboard approach to partition occurrence localities. These methods use the `spatSample` function of the **terra** package (Hijmans 2023) to partition records according to checkerboard squares generated based on the input rasters. The spatial grain of these squares is determined by resampling (or aggregating) the original environmental input grids based on the user-defined `aggregation factor` (e.g., an aggregation factor with value 2 results in a checkerboard with grid cells four times the area of the original input rasters). With one input aggregation factor, `get.checkerboard` partitions data into two groups according to a 'basic' checkerboard pattern. With two aggregation factors, `get.checkerboard` partitions data into four groups according to 'hierarchical', or nested, checkerboard squares (see Muscarella et al. 2014). In contrast to the `get.block` method, the `get.checkerboard` methods subdivide geographic space equally but do not ensure a balanced number of occurrence localities in each group. The `get.checkerboard` methods give warnings (and potentially errors) if zero points (occurrence or background) fall in any of the expected bins.

The `get.jackknife` method is a special case of *k*-fold cross validation where the number of bins (*k*) is equal to the number of occurrence localities (*n*) in the dataset. It is suggested for occurrence datasets of relatively small sample size (generally < 25 localities) (Pearson *et al.* 2007; Shcheglovitova and Anderson 2013).

The `get.randomkfold` method partitions occurrence localities randomly into a user-specified number of (*k*) bins. This is equivalent to the method of *k*-fold cross validation currently provided by Maxent.

Users can also define custom partitions for occurrence and background data in the call to ‘ENMevaluate‘ with the "user.grp" parameter.

Usage

```
get.block(occs, bg, orientation = "lat_lon")

get.checkerboard(occs, envs, bg, aggregation.factor, gridSampleN = 10000)

get.checkerboard1(occs, envs, bg, aggregation.factor, gridSampleN = 10000)

get.checkerboard2(occs, envs, bg, aggregation.factor, gridSampleN = 10000)

get.jackknife(occs, bg)

get.randomkfold(occs, bg, kfolds)
```

Arguments

`occs`	matrix / data frame: longitude and latitude (in that order) of occurrence localities
`bg`	matrix / data frame: longitude and latitude (in that order) of background localities
`orientation`	character vector: the order of spatial partitioning for the `get.block` method; the first direction bisects the points into two groups, and the second direction bisects each of these further into two groups each, resulting in four groups; options are "lat_lon" (default), "lon_lat", "lon_lon", and "lat_lat"
`envs`	SpatRaster: environmental predictor variables
`aggregation.factor`	numeric or numeric vector: the scale of aggregation for `get.checkerboard`; can have one value (for ’basic’) or two values (for ’hierarchical’) – see Details.
`gridSampleN`	numeric: the number of points sampled from the input raster using gridSample() by the checkerboard partitioning functions
`kfolds`	numeric: number of random *k*-folds for `get.randomkfold` method

Value

A named list of two items:

`$occs.grp`	A vector of bin designation for occurrence localities in the same order they were provided.
`$bg.grp`	A vector of bin designation for background localities in the same order they were provided.

Note

The `checkerboard` methods are designed to partition occurrence localities into spatial evaluation bins: two (’basic’, for one aggregation factor) or four (’hierarchical’, for two aggregation factors). They may give fewer bins, however, depending on where the occurrence localities fall with respect to the grid cells (e.g., all records happen to fall in one group of checkerboard squares). A warning is given if the number of bins is < 4 for the hierarchical method, and an error is given if all localities fall into a single evaluation bin.

Author(s)

Robert Muscarella <bob.muscarella@gmail.com> and Jamie M. Kass <jamie.m.kass@gmail.com>

References

Hijmans, R. J. (2023). terra: Spatial Data Analysis. Available online at: `https://cran.r-project.org/package=terra`.

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(11), 1198-1205. doi:10.1111/2041210X.12945

Pearson, R. G., Raxworthy, C. J., Nakamura, M. and Peterson, A. T. (2007). Predicting species distributions from small numbers of occurrence records: a test case using cryptic geckos in Madagascar. *Journal of Biogeography*, **34**: 102-117. doi:10.1111/j.13652699.2006.01594.x

Radosavljevic, A., & Anderson, R. P. (2014). Making better Maxent models of species distributions: complexity, overfitting and evaluation. *Journal of Biogeography*, **41**: 629-643. doi:10.1111/jbi.12227

Shcheglovitova, M. and Anderson, R. P. (2013). Estimating optimal complexity for ecological niche models: a jackknife approach for species with small sample sizes. *Ecological Modelling*, **269**: 9-17. doi:10.1016/j.ecolmodel.2013.08.011

Examples

```
library(terra)

set.seed(1)

### Create environmental extent (raster)
envs <- rast(nrow = 25, ncol = 25, xmin = 0, xmax = 1, ymin = 0, ymax = 1)

### Create occurrence localities
set.seed(1)
nocc <- 25
xocc <- rnorm(nocc, sd=0.25) + 0.5
yocc <- runif(nocc, 0, 1)
occs <- as.data.frame(cbind(xocc, yocc))

### Create background points
nbg <- 500
xbg <- runif(nbg, 0, 1)
ybg <- runif(nbg, 0, 1)
bg <- as.data.frame(cbind(xbg, ybg))

### Plot points on environmental raster
plot(envs)
```

```
points(bg)
points(occs, pch=21, bg=2)

### Block partitioning method (default orientation is "lat_lon"))
blk.latLon <- get.block(occs, bg)
plot(envs)
points(occs, pch=23, bg=blk.latLon$occs.grp)
plot(envs)
points(bg, pch=21, bg=blk.latLon$bg.grp)
# Can partition with other orientations
blk.latLat <- get.block(occs, bg, orientation = "lat_lat")
plot(envs)
points(occs, pch=23, bg=blk.latLat$occs.grp)
plot(envs)
points(bg, pch=21, bg=blk.latLat$bg.grp)

### Checkerboard partitioning method with aggregation factor of 4
chk.ag4 <- get.checkerboard(occs, envs, bg, aggregation.factor = 4)
plot(envs)
points(occs, pch=23, bg=chk.ag4$occs.grp)
plot(envs)
points(bg, pch=21, bg=chk.ag4$bg.grp)
# Higher aggregation factors result in bigger checkerboard blocks
chk.ag8 <- get.checkerboard(occs, envs, bg, aggregation.factor = 8)
plot(envs)
points(occs, pch=23, bg=chk.ag8$occs.grp)
plot(envs)
points(bg, pch=21, bg=chk.ag8$bg.grp)

### Hierarchical checkerboard partitioning method with aggregation factors
### of 2 and 2
chk.ag2_2 <- get.checkerboard(occs, envs, bg, c(2,2))
plot(envs)
points(occs, pch=23, bg=chk.ag2_2$occs.grp)
plot(envs)
points(bg, pch=21, bg=chk.ag2_2$bg.grp)
# Higher aggregation factors result in bigger checkerboard blocks,
# and can vary between hierarchical levels
chk.ag4_6 <- get.checkerboard(occs, envs, bg, c(3,4))
plot(envs)
points(occs, pch=23, bg=chk.ag4_6$occs.grp)
plot(envs)
points(bg, pch=21, bg=chk.ag4_6$bg.grp)

### Random partitions with 4 folds
# Note that get.randomkkfold does not partition the background
krandom <- get.randomkfold(occs, bg, 4)
plot(envs)
points(occs, pch=23, bg=krandom$occs.grp)
plot(envs)
points(bg, pch=21, bg=krandom$bg.grp)

### k-1 jackknife partitions
```

```
# Note that get.jackknife does not partition the background
jack <- get.jackknife(occs, bg)
plot(envs)
points(occs, pch=23, bg=rainbow(length(jack$occs.grp)))
plot(envs)
points(bg, pch=21, bg=jack$bg.grp)
```

saveENMevaluation *Save ENMevaluation object*

Description

Save an ENMevaluation object as an .rds file. This is necessary to use instead of `saveRDS()` because terra SpatRasters require `wrap()` before saving to preserve the connections to the raster data. This convenience function does that for you.

Usage

```
saveENMevaluation(e, filename)
```

Arguments

`e`	ENMevaluation object
`filename`	character: path to the file to create with .rds extension

tune.enm *Iterate tuning of ENMs*

Description

Internal functions to tune and summarize results for ecological niche models (ENMs) iteratively across a range of user-specified tuning settings. See ENMevaluate for descriptions of shared arguments. Function `tune.parallel()` tunes ENMs with parallelization. Function `cv.enm()` calculates training and validation evaluation statistics for one set of specified tuning parameters.

Validation CBI is calculated here with background values, not raster data, in order to standardize the methodology for both training and validation data for spatial partitions, as ENMeval does not mask rasters to partition areas and hence does not have partitioned raster data. Further, predictions for occurrence and background localities are combined as input for the parameter "fit" in `ecospat::ecospat_boyce()` because the interval is determined from "fit" only, and if test occurrences all have higher predictions than the background, the interval will be cut short.

Usage

```
tune.train(
  enm,
  occs.z,
  bg.z,
  mod.full,
  tune.tbl.i,
  other.settings,
  partitions,
  quiet
)

tune.validate(
  enm,
  occs.train.z,
  occs.val.z,
  bg.train.z,
  bg.val.z,
  mod.k,
  nk,
  tune.tbl.i,
  other.settings,
  partitions,
  user.eval,
  quiet
)

tune(
  d,
  enm,
  partitions,
  tune.tbl,
  doClamp,
  other.settings,
  partition.settings,
  user.val.grps,
  occs.testing.z,
  numCores,
  parallel,
  user.eval,
  algorithm,
  updateProgress,
  quiet
)

cv.enm(
  d,
  enm,
  partitions,
  tune.tbl.i,
  doClamp,
  other.settings,
  partition.settings,
  user.val.grps,
  occs.testing.z,
  user.eval,
  algorithm,
  quiet
)
```

Arguments

`enm`	ENMdetails object
`occs.z`	data.frame: the envs values for the coordinates at the full dataset occurrence records
`bg.z`	data.frame: the envs values for the coordinates at the full dataset background records
`mod.full`	model object: the model trained on the full dataset
`tune.tbl.i`	vector: single set of tuning parameters
`other.settings`	named list: used to specify extra settings for the analysis. All of these settings have internal defaults, so if they are not specified the analysis will be run with default settings. See Details for descriptions of these settings, including how to specify arguments for maxent.jar.
`partitions`	character: name of partitioning technique (see `?partitions`)
`quiet`	boolean: if TRUE, silence all function messages (but not errors).
`occs.train.z`	data.frame: the envs values for the coordinates at the training occurrence records
`occs.val.z`	data.frame: the envs values for the coordinates at the validation occurrence records
`bg.train.z`	data.frame: the envs values for the coordinates at the training background records
`bg.val.z`	data.frame: the envs values for the coordinates at the validation background records
`mod.k`	model object: the model trained on the training dataset that becomes evaluated on the validation data
`nk`	numeric: the number of folds (i.e., partitions) – will be equal to `kfolds` for random partitions
`user.eval`	function: custom function for specifying performance metrics not included in **ENMeval**. The function must first be defined and then input as the argument `user.eval`. This function should have a single argument called `vars`, which is a list that includes different data that can be used to calculate the metric. See Details below and the vignette for a worked example.
`d`	data frame: data frame from `ENMevaluate()` with occurrence and background coordinates (or coordinates plus predictor variable values) and partition group values

`tune.tbl`	data frame: all combinations of tuning parameters
`doClamp`	boolean: if TRUE (default), model prediction extrapolations will be restricted to the upper and lower bounds of the predictor variables. Clamping avoids extreme predictions for environment values outside the range of the training data. If free extrapolation is a study aim, this should be set to FALSE, but for most applications leaving this at the default of TRUE is advisable to avoid unrealistic predictions. When predictor variables are input, they are clamped internally before making model predictions when clamping is on. When no predictor variables are input and data frames of coordinates and variable values are used instead (SWD format), validation data is clamped before making model predictions when clamping is on.
`partition.settings`	named list: used to specify certain settings for partitioning schema. See Details and ?partitions for descriptions of these settings.
`user.val.grps`	matrix / data frame: user-defined validation record coordinates and predictor variable values. This is used internally by `ENMnulls()` to force each null model to evaluate with empirical validation data, and does not have any current use when running `ENMevaluate()` independently.
`occs.testing.z`	data.frame: when fully withheld testing data is provided, the envs values for the coordinates at the testing occurrence records
`numCores`	numeric: number of cores to use for parallel processing. If NULL, all available cores will be used.
`parallel`	boolean: if TRUE, run with parallel processing.
`algorithm`	character: name of the algorithm used to build models. Currently one of "maxnet", "maxent.jar", or "bioclim", else the name from a custom ENMdetails implementation.
`updateProgress`	boolean: if TRUE, use shiny progress bar. This is only for use in shiny apps.

Index