# Package ‘ENMeval’

September 12, 2020

**Type** Package

**Title** Automated Runs and Evaluations of Ecological Niche Models

**Version** 0.3.1

**Date** 2020-09-08

**Author** Robert Muscarella, Peter J. Galante, Mariano Soley-Guardia, Robert A. Boria, Jamie M. Kass, Maria Uriarte and Robert P. Anderson

**Maintainer** Robert Muscarella <bob.muscarella@gmail.com>

**Description** Automatically partitions data into evaluation bins, executes ecological niche models across a range of settings, and calculates a variety of evaluation statistics. Current version only implements ENMs with Maxent (Phillips et al. 2006) or maxnet (Phillips et al. 2017).

**License** GPL

**Encoding** UTF-8

**Depends** methods, R (>= 3.4), dismo

**Imports** doParallel, foreach, utils, graphics, stats, grDevices, raster, maxnet

**Suggests** rJava (>= 0.5-0), parallel, knitr, rmarkdown, spocc, maptools, rgeos, sp

**VignetteBuilder** knitr

**RoxygenNote** 7.1.1

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2020-09-12 05:40:16 UTC

## R topics documented:

---

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

---

## Description

Automatically partitions data into bins for model training and testing, executes ecological niche models (ENMs) across a range of user-defined settings, and calculates evaluation metrics to help achieve a balance between goodness-of-fit and model complexity.

## Details

| | |
|---|---|
| Package: | ENMeval |
| Type: | Package |
| Version: | 0.3.1 |
| Date: | 2020-09-08 |
| License: | GNU 3.0 |

The **ENMeval** package (Muscarella *et al.* 2014) (1) automatically partitions data into training and testing bins using one of six methods (including several options for spatially independent partitions as well as user-defined bins), (2) executes a series of ENMs using Maxent (Phillips *et al.* 2006, Phillips *et al.* 2017) with a variety of user-defined settings (i.e., feature classes and regularization multipliers), conducting *k*-fold cross validation, and (3) calculates multiple evaluation metrics to aid in selecting model settings that balance model goodness-of-fit and complexity (i.e., "model tuning" or "smoothing").

`ENMevaluate` is the primary function of the **ENMeval** package, and multiple other functions highlighted below are called when it is run. The six options for partitioning occurrence data into training and testing (i.e., calibration and evaluation) bins are: *n*-1 jackknife, random *k*-fold, user-specified bins, and three explicit methods of masked geographically structured *k*-fold partitioning (see: `get.evaluation.bins`). After model training, these bins are used to calculate five metrics of model performance for each combination of settings: model discrimination (AUC of test localities), the difference between training and testing AUC, two different threshold-based omission rates, and the small sample-size corrected version of the Akaike information criterion (AICc), the latter using the unpartitioned dataset. A model prediction (as a raster layer) using the full (unpartitioned) dataset is generated for each combination of feature class and regularization multiplier settings. Similarity of these models in geographic space (i.e., "niche overlap") can be calculated to better understand how model settings change predictions (see `calc.niche.overlap`). The results of `ENMevaluate` are returned as an object of class `ENMevaluation-class`. A basic plotting function (`eval.plot`) can be used to visualize how evaluation metrics depend on model settings.

**- As of version 0.3.0 -**

The default `ENMevaluate` runs the the Maxent algorithm by calling the **maxnet** package (Phillips *et al.* 2017) instead of the previous implementation that relied on the 'maxent.jar' Java program called by the **dismo** package. This is controlled by the new argument, `algorithm='maxnet'`. A major advantage of this change is that it removes the reliance on Java and the **rJava** package, which is great but can sometimes cause confusing problems on different computers. Our team has done some fairly extensive testing to ensure this implementation gives the expected results but the maxnet implementation is relatively new (at the time of writing this) and we encourage users to scrutinize their results.

There are some differences between the 'maxnet' and 'maxent.jar' algorithms that may lead to slight numeric differences in the results (at least when `hinge` feature classes are used). See Phillips *et al.* (2017) and Phillips (2017) for more details. Additionally, the 'maxnet' algorithm does not provide information on variable importance (from the `var.importance()` function) because of differences in the underlying models. Users can still choose to use the 'maxent.jar' implementation by setting `algorithm='maxent.jar'` in the `ENMevaluate` function (also see note below).

**- As of version 0.2.0 -**

`ENMevaluate` includes an option for parallel computing. Setting `parallel = TRUE` can significantly speed up processing time, particularly for large analyses. For very small analyses, it may actually take longer than running with `parallel = FALSE`.

## Note

Currently, **ENMeval** only implements the Maxent algorithm (via either the 'maxent.jar' or 'maxnet' implementations), but we eventually plan to expand it to work with other algorithms. All calculations are based on the raw Maxent output (i.e., *not* logistic or cumulative transformations) and users can choose whether to use 'clamping' (see Maxent documentation for details on this option). Additionally, Maxent models are run with the arguments: `noaddsamplestobackground` and `noremoveDuplicates`. Users should consult Maxent documentation (Phillips *et al.* 2006) and other references (e.g., Phillips and Dudik 2008) for more information on these options. We note that interested users can edit the source code of `ENMeval` (in particular, the `make.args` and `tuning` functions) if they desire to change these or other options.

When using the 'maxent.jar' implementation (***not default as of version 0.3.0***), `ENMevaluate` directly uses several functions from the **dismo** package (Hijmans *et al.* 2011). Most importantly, the `maxent` function that runs the Maxent algorithm (Phillips *et al.* 2006) in Java. Before running this command, the user must first download Maxent from this website. Then, place the file 'maxent.jar' in the 'java' folder of the **dismo** package. The user can locate that folder by typing: `system.file("java",package="dismo")`. For additional details, users should consult the documentation of the **dismo** package (or just use the newer [default] **maxnet** implementation).

## Author(s)

Robert Muscarella, Peter J. Galante, Mariano Soley-Guardia, Robert A. Boria, Jamie M. Kass, Maria Uriarte and Robert P. Anderson

Maintainer: Robert Muscarella <bob.muscarella@gmail.com>

### References

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

Muscarella, R., Galante, P. J., Soley-Guardia, M., Boria, R. A., Kass, J. M., Uriarte, M., and 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.

Phillips, S. J. 2017. maxnet package for R. Available online at: `https://CRAN.R-project.org/package=maxnet`.

Phillips, S. J., Anderson, R. P., Dudík, M., Schapire, R. E. and Blair, M. E. 2017. Opening the black box: an open-source release of Maxent. *Ecography*, **40**: 887–893.

Phillips, S. J., Anderson, R. P., and Schapire, R. E. 2006. Maximum entropy modeling of species geographic distributions. *Ecological Modelling*, **190**: 231-259.

Phillips, S. J., and Dudik, M. 2008. Modeling of species distributions with Maxent: new extensions and a comprehensive evaluation. *Ecography*, **31**: 161-175.

### See Also

`maxnet` in the **maxnet** package

`maxent` in the **dismo** package

---

## `calc.aicc` *Calculate AICc from Maxent model prediction*

---

### Description

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

### Usage

```
calc.aicc(nparam, occ, predictive.maps)

get.params(model)
```

### Arguments

| | |
|---|---|
| `nparam` | The number of parameters in a model calculated with the `get.params` function. |
| `occ` | A data.frame of occurrence localities. |
| `predictive.maps` | A raster layer or RasterStack of predicted model surface(s). |
| `model` | A Maxent model object generated by the `maxent` function in the **dismo** package. |

## 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 parameters is determined by counting the number of non-zero parameters in the `maxent` lambda file. See Warren *et al.* (2014) for limitations of this approach, namely that the number of parameters is an estimate of the true degrees of freedom. For Maxent ENMs, AICc is calculated by standardizing the raw output such that all cells in the study extent sum to 1. The likelihood of the data for a given model is then calculated by taking the product of the raw output values for all grid cells that contain an occurrence locality (Warren and Seifert 2011).

## Value

A data.frame with four 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 parameters in the model (i.e., number of non-zero parameters in Maxent lambda file) and *n* is the number of occurrence localities. The *logLikelihood* is calculated as:

$$sum(log(vals/total))$$

where *vals* is a vector of Maxent raw values at occurrence localities and *total* is the sum of Maxent raw values across the entire study area.

`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).

`nparam` is the number of parameters in a Maxent model (number of non-zero parameters in the lambda file) and is used internally during a call of `calc.aicc` by `get.params`.

## Note

Returns all `NA`s if the number of parameters is larger than the number of observations (occurrence localities).

This function could produce erroneous results in version 0.1.0. when calculating AICc on multiple models simultaneously. This problem has been addressed in version 0.1.1.

## Author(s)

Robert Muscarella <bob.muscarella@gmail.com> and Jamie M. Kass <jkass@gc.cuny.edu>

### References

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

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.

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

Warren, D. L. and 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.

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.

### See Also

`maxent` in the **dismo** package.

---

## `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(predictive.maps, stat = "D", maxent.args)
```

### Arguments

`predictive.maps`
: A rasterStack of at least 2 Maxent predictive raster layers.

`stat`
: The statistic calculated by the `nicheOverlap` function of the **dismo** package. Defaults to Schoener's *D* (Schoener 1968) but can also accept `"I"` to calculate the *I* similarity statistic from Warren *et al.* (2008).

`maxent.args`
: A list of (1) feature classes, and (2) regularization multiplier values that describe model settings for each predictive map provided in `predictive.maps`. This is generated by the function `make.args(...,labels=TRUE)`.

### Value

A matrix with the lower triangle giving values of pairwise "niche overlap" in geographic space. Row and column names are given by the `make.args` argument when run by the `ENMevaluate` function.

### Author(s)

Based on **dismo**::`nicheOverlap`, which is based on **SDMTools**::`Istat`

Robert Muscarella <bob.muscarella@gmail.com>

### References

Hijmans, R. J., Phillips, S., Leathwick, J. and 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.

Warren, D. L., Glor, R. E., Turelli, M. and Funk, D. (2008) Environmental niche equivalency versus conservatism: quantitative approaches to niche evolution. *Evolution*, **62**: 2868-2883.

### See Also

`make.args`; `nicheOverlap` in the **dismo** package

---

## `corrected.var` *Calculate variance corrected for non-independence of* k*-fold iterations*

---

### Description

This function 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.

### Usage

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

### Arguments

| | |
|---|---|
| `x` | A numeric vector. |
| `nk` | Number of *k*-fold iterations. |

### 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.

Parr, W. C. (1985) Jackknifing differentiable statistical functionals. *Journal of the Royal Statistics Society, Series B*, **47**: 56-66.

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

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.

---

`ENMevaluate` *Tuning and evaluation of ENMs with Maxent*

---

## Description

`ENMevaluate` automatically executes Maxent (Phillips *et al*. 2006; Phillips and Dudik 2008) across a range of settings, returning a `data.frame` of evaluation metrics to aid in identifying settings that balance model fit and predictive ability. Since version 0.3.0, the default function uses the `maxnet` function in the **maxnet** package (Phillips *et al.* 2017) to implement the Maxent algorithm (*see notes*).

## Usage

```
ENMevaluate(occ, env, bg.coords = NULL, occ.grp = NULL,
bg.grp = NULL, RMvalues = seq(0.5, 4, 0.5),
fc = c("L", "LQ", "H", "LQH", "LQHP", "LQHPT"),
categoricals = NULL, n.bg = 10000, method = NULL,
algorithm = 'maxnet', overlap = FALSE,
aggregation.factor = c(2, 2), kfolds = NA,
bin.output = FALSE, clamp = TRUE, rasterPreds = TRUE,
parallel = FALSE, numCores = NULL, progbar = TRUE,
updateProgress = FALSE, ...)

tuning(occ, env, bg.coords, occ.grp, bg.grp, method,
algorithm, args, args.lab, categoricals,
aggregation.factor, kfolds, bin.output, clamp, alg,
rasterPreds, parallel, numCores, progbar,
updateProgress, userArgs)
```

## Arguments

- `occ` — Two-column matrix or data.frame of longitude and latitude (in that order) of occurrence localities.
- `env` — RasterStack of model predictor variables (environmental layers).
- `bg.coords` — Two-column matrix or data.frame of longitude and latitude (in that order) of background localities (required for 'user' method).
- `occ.grp` — Vector of bins of occurrence localities (required for 'user' method).
- `bg.grp` — Vector of bins of background localities (required for 'user' method).
- `RMvalues` — Vector of (non-negative) values to use for the regularization multiplier.
- `fc` — Character vector of feature class combinations to be included in analysis.
- `algorithm` — Character vector. Use `'maxnet'` to use the **maxnet** package [default] or `'maxent.jar'` to use the **dismo** package and the 'maxent.jar' Java program. See details for more information on these different implementations.
- `alg` — Character vector. Use `'maxnet'` to use the **maxnet** package [default] or `'maxent.jar'` to use the **dismo** package and the 'maxent.jar' Java program. See details for more information on these different implementations.
- `categoricals` — Vector indicating which (if any) of the input environmental layers are categorical.
- `n.bg` — The number of random background localities to draw from the study extent.
- `method` — Character string designating the method used for data partitioning. Choices are: `"jackknife","randomkfold","user","block","checkerboard1","checkerboard2"`. See details and `get.evaluation.bins` for more information.
- `overlap` — logical; If `TRUE`, provides pairwise metric of niche overlap (see details and `calc.niche.overlap`).
- `aggregation.factor` — List giving the factor by which the original input grid should be aggregated for checkerboard partitioning methods (see details and `get.evaluation.bins`).
- `kfolds` — Number of bins to use in the *k*-fold random method of data partitioning.
- `bin.output` — logical; If `TRUE`, appends evaluations metrics for each evaluation bin to results table (i.e., in addition to the average values across bins).
- `args` — Arguments to pass to Maxent that are generated by the `make.args` function
- `args.lab` — Character labels describing feature classes and regularization multiplier values for Maxent runs provided by the `make.args` function.
- `clamp` — logical; If `TRUE`, 'clamping' is used (see Maxent documentation and tutorial for more details).
- `rasterPreds` — logical; If `TRUE`, the `predict` function from `dismo` is used to predict each full model across the extent of the input environmental variables. Note that AICc (and associated values) are *NOT* calculated if `rasterPreds=FALSE` because these calculations require the predicted surfaces. However, setting to `FALSE` can significantly reduce run time.
- `parallel` — logical; If `TRUE`, parallel processing is used to execute tuning function.

| | |
|---|---|
| numCores | numeric; indicates the number of cores to use if running in parallel. If `parallel=TRUE` and this is not specified, the total number of available cores are used. |
| progbar | logical; used internally. |
| updateProgress | logical; used internally. |
| ... | character vector; use this to pass other arguments (e.g., prevalence) to the 'maxent' call. Note that not all options are functional or relevant. |
| userArgs | character vector; use this to pass other arguments (e.g., prevalence) to the 'maxent' call. Note that not all options are functional or relevant. |

## Details

`ENMevaluate` is the primary function for general use in the **ENMeval** package; the `tuning` function is used internally.

Since version 0.3.0, the default `ENMevaluate` runs the the Maxent algorithm by calling the **maxnet** package (Phillips *et al.* 2017) instead of the previous implementation (still available) that relies on the 'maxent.jar' Java program called by the **dismo** package. This choice is controlled by the argument `algorithm='maxnet'`. A major advantage of this change is that it removes the reliance on Java and the **rJava** package, which is great but can sometimes cause confusing problems on different computers. There are some differences between the 'maxnet' and 'maxent.jar' algorithms that may lead to slight numeric differences in the results (at least when `hinge` feature classes are used). See Phillips *et al.* (2017) and Phillips (2017) for more details. Additionally, the 'maxnet' algorithm does not provide information on variable importance (from the `var.importance()` function) because of differences in the underlying models. Users can still choose to use the 'maxent.jar' implementation by setting `algorithm='maxent.jar'` in the `ENMevaluate` function (also see note below). Our team has done some fairly extensive testing to ensure this implementation gives the expected results but the maxnet implementation is relatively new (at the time of writing this) and we encourage users to scrutinize their results.

*Maxent settings:* In the current default implementation of Maxent, the combination of feature classes (`fcs`) allowed depends on the number of occurrence localities, and the value for the regularization multiplier (`RM`) is 1.0. `ENMevaluate` provides an automated way to execute ecological niche models in Maxent across a user-specified range of (`RM`) values and (`fc`) combinations, regardless of sample size. Acceptable values for the `fc` argument include: L=linear, Q=quadratic, P=product, T=threshold, and H=hinge (see Maxent help documentation, Phillips *et al.* (2006), Phillips and Dudik (2008), Elith *et al.* (2011), and Merow *et al.* (2013) for additional details on `RM` and `fcs`). Categorical feature classes (C) are specified by the `categoricals` argument.

*Methods for partitioning data:* `ENMevaluate` includes six methods to partition occurrence and background localities into bins for training and testing (`'jackknife'`,`'randomkfold'`,`'user'`,`'block'`, `'checkerboard1'`,`'checkerboard2'`). The `jackknife` method is a special case of *k*-fold cross validation where the number of folds (*k*) is equal to the number of occurrence localities (*n*) in the dataset. The `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. The `user` method enables users to define bins *a priori*. For this method, the user is required to provide background coordinates (`bg.coords`) and bin designations for both occurrence localities (`occ.grp`) and background localities (`bg.grp`). The `block` method partitions the data into four bins according to the lines of latitude and longitude that divide the occurrence localities into bins of as equal number as possible. The `checkerboard1` (and `checkerboard2`) methods partition data into two (or four) bins based on one (or two) checkerboard patterns with grain size defined as one (or two) aggregation factor(s) of the original environmental layers. Although the `checkerboard1` (and `checkerboard2`) methods are designed to partition occurrence localities into two (and four) evaluation bins, they may give fewer bins depending on the location of occurrence localities with respect to the checkerboard grid(s) (e.g., all records happen to fall in the "black" squares). A warning is given if the number of bins is < 4 for the `checkerboard2` method, and an error is given if all localities fall in a single evaluation bin. Additional details can be found in `get.evaluation.bins`.

*Evaluation metrics:* Four evaluation metrics are calculated using the partitioned dataset, and one additional metric is provided based on the full dataset. `ENMevaluate` uses the same background localities and evaluation bin designations for each of the *k* iterations (for each unique combination of `RM` and `fc`) to facilitate valid comparisons among model settings.

`avg.test.AUC` is the area under the curve of the receiver operating characteristic plot made based on the testing data (i.e., AUCtest), averaged across *k* bins. In each iteration, as currently implemented, the AUCtest value is calculated with respect to the full set of background localities to enable comparisons across the *k* iterations (Radosavljevic and Anderson 2014). As a relative measure for a given study species and region, high values of `avg.test.AUC` are associated with the degree to which a model can successfully discriminate occurrence from background localities. This rank-based non-parametric metric, however, does not reveal the model goodness-of-fit (Lobo *et al.* 2008; Peterson *et al.* 2011).

To quantify the degree of overfitting, `ENMevaluate` calculates three metrics. The first is the difference between training and testing AUC, averaged across *k* bins (`avg.diff.AUC`) (Warren and Seifert 2011). `avg.diff.AUC` is expected to be high for models overfit to the training data. `ENMevaluate` also calculates two threshold-dependent omission rates that quantify overfitting when compared with the omission rate expected by the threshold employed: the proportion of testing localities with Maxent output values lower than the value associated with (1) the training locality with the lowest value (i.e., the minimum training presence, MTP; = 0 percent training omission) (`avg.test.orMTP`) and (2) the value that excludes the 10 percent of training localities with the lowest predicted suitability (`avg.test.or10pct`) (Pearson *et al.* 2007). `ENMevaluate` uses `corrected.var` to calculate the variance for each of these metrics across *k* bins (i.e., variances are corrected for non-independence of cross-validation iterations; see Shcheglovitova and Anderson 2013). The value of these metrics for each of the individual *k* bins is returned if `bin.output = TRUE`.

Based on the unpartitioned (full) dataset, `ENMevaluate` uses `calc.aicc` to calculate the AICc value for each model run and provides delta.AIC, AICc weights, as well as the number of parameters for each model (Warren and Seifert 2011). Note that AICc (and associated values) are *NOT* calculated if `rasterPreds=FALSE` because these calculations require the predicted surfaces. The AUCtrain value for the full model is also returned (`train.AUC`).

To quantify how resulting predictions differ in geographic space depending on the settings used, `ENMevaluate` includes an option to compute pairwise niche overlap between all pairs of full models (i.e., using the unpartitioned dataset) with Schoeners *D* statistic (Schoener 1968; Warren *et al.* 2009).

## Value

An object of class `ENMevaluation` with named slots:

`@results` data.frame of evaluation metrics. If `bin.output=TRUE`, evaluation metrics calculated separately for each evaluation bin are included in addition to the averages and corrected variances (see `corrected.var`) across *k* bins. Note that the names of some columns changed as of Version 0.3.0.

`@predictions` RasterStack of full model predictions with each layer named as: `fc_RM` (e.g., `L_1`). This will be an empty RasterStack if the `rasterPreds=FALSE`.

`@models` List of objects of class `"MaxEnt"` from the **dismo** package. Each of these entries include slots for lambda values and the original Maxent results table. See Maxent documentation for more information.

`@partition.method` character vector with the method used for data partitioning.

`@occ.pts` data.frame of the latitude/longitude of input occurrence localities.

`@occ.grp` vector identifying the bin for each occurrence locality.

`@bg.pts` data.frame of the latitude/longitude of input background localities.

`@bg.grp` vector identifying the bin for each background locality.

`@overlap` matrix of pairwise niche overlap (blank if `overlap = FALSE`).

## Author(s)

Uses the `maxent` function in the **dismo** package (Hijmans *et al.* 2011, Phillips *et al.* 2006)

Robert Muscarella <bob.muscarella@gmail.com> and Jamie M. Kass <jkass@gc.cuny.edu>

## References

Elith, J., Phillips, S. J., Hastie, T., Dudik, M., Chee, Y. E., and Yates, C. J. (2011) A statistical explanation of MaxEnt for ecologists. *Diversity and Distributions*, **17**: 43-57.

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

Lobo, J. M., Jimenez-Valverde, A., and Real, R. (2008) AUC: A misleading measure of the performance of predictive distribution models. *Global Ecology and Biogeography*, **17**: 145-151.

Muscarella, R., Galante, P.J., Soley-Guardia, M., Boria, R.A., Kass, J., Uriarte, M. and Anderson, R.P. (2014) ENMeval: An R package for conducting spatially independent evaluations and estimating optimal model complexity for ecological niche models. *Methods in Ecology and Evolution*, **5**: 1198-1205.

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.

Peterson, A. T., Soberon, J., Pearson, R. G., Anderson, R. P., Martinez-Meyer, E., Nakamura, M. and Araujo, M. B. (2011) *Ecological Niches and Geographic Distributions*. Monographs in Population Biology, 49. Princeton University Press, Princeton, NJ.

Phillips, S. J. 2017. maxnet package for R. Available online at: https://CRAN.R-project.org/package=maxnet.

Phillips, S. J., Anderson, R. P., Dudík, M., Schapire, R. E. and Blair, M. E. 2017. Opening the black box: an open-source release of Maxent. *Ecography*, **40**: 887–893.

Phillips, S. J., Anderson, R. P., and Schapire, R. E. (2006) Maximum entropy modeling of species geographic distributions. *Ecological Modelling*, **190**: 231-259.

Phillips, S. J. and Dudik, M. (2008) Modeling of species distributions with Maxent: new extensions and a comprehensive evaluation. *Ecography*, **31**: 161-175.

Merow, C., Smith, M., and Silander, J. A. (2013) A practical guide to Maxent: what it does, and why inputs and settings matter. *Ecography*, **36**: 1-12.

Radosavljevic, A. and Anderson, R. P. 2014. Making better Maxent models of species distributions: complexity, overfitting and evaluation. *Journal of Biogeography*, **41**: 629-643.

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

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.

Warren, D. L., Glor, R. E., Turelli, M. and Funk, D. (2009) Environmental niche equivalency versus conservatism: quantitative approaches to niche evolution. *Evolution*, **62**: 2868-2883; *Erratum: Evolution*, **65**: 1215.

Warren, D.L. and 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.

## See Also

`maxnet` in the **maxnet** package

`maxent` in the **dismo** package

## Examples

```
require(raster)

### Simulated data environmental covariates
set.seed(1)
r1 <- raster(matrix(nrow=50, ncol=50, data=runif(10000, 0, 25)))
r2 <- raster(matrix(nrow=50, ncol=50, data=rep(1:100, each=100), byrow=TRUE))
r3 <- raster(matrix(nrow=50, ncol=50, data=rep(1:100, each=100)))
r4 <- raster(matrix(nrow=50, ncol=50, data=c(rep(1,1000),rep(2,500)),byrow=TRUE))
values(r4) <- as.factor(values(r4))
env <- stack(r1,r2,r3,r4)

### Simulate occurrence localities
nocc <- 50
x <- (rpois(nocc, 2) + abs(rnorm(nocc)))/11
y <- runif(nocc, 0, .99)
occ <- cbind(x,y)

## Not run:
### This call gives the results loaded below
enmeval_results <- ENMevaluate(occ, env, method="block", n.bg=500,
  categoricals=4, algorithm='maxent.jar')

## End(Not run)

data(enmeval_results)
enmeval_results
```

```
### See table of evaluation metrics
enmeval_results@results

### Plot prediction with lowest AICc
plot(enmeval_results@predictions[[which (enmeval_results@results$delta.AICc == 0) ]])
points(enmeval_results@occ.pts, pch=21, bg=enmeval_results@occ.grp)

### Niche overlap statistics between model predictions
enmeval_results@overlap
```

---

ENMevaluation-class *Class* "ENMevaluation"

---

## Description

Objects of this class are generated by a call of ENMevaluate.

## Objects from the Class

Objects can be created by calls of the form new("ENMevaluation",...).

## Slots

algorithm**:** Object of class "character". The algorithm used for the analysis.

results**:** Object of class "data.frame". The full results table.

predictions**:** Object of class "RasterStack". Model predictions in geographic space.

models**:** List of objects of class "maxnet" from the **maxnet** package or "MaxEnt" from the **dismo** package (depending on which algorithm was used). For "Maxnet", see **maxnet** package documentation for more information. For "MaxEnt", each of these entries include slots for lambda values and the original Maxent results table. See **dismo** package documentation for more information.

partition.method**:** Object of class "character". Indicates the method used for data partitioning.

occ.pts**:** Object of class "data.frame". The original presence coordinates.

occ.grp**:** Object of class "numeric". The evaluation bin assignment for each occurrence point.

bg.pts**:** Object of class "data.frame". The background coordinates used for analysis.

bg.grp**:** Object of class "numeric". The evaluation bin assignment for each background point.

overlap**:** Object of class "matrix". Niche overlap statistic between models of different settings.

## Author(s)

Jamie M. Kass <jkass@gc.cuny.edu> and Robert Muscarella <bob.muscarella@gmail.com>

## Examples

```
showClass("ENMevaluation")
```

enmeval_results *An object of class "ENMevaluation"*

## Description

An example results file based on a call of ENMevaluate (see example).

## Usage

```
data(enmeval_results)
```

## Format

An object of class 'ENMevaluation' with nine slots:

@ `results` : data.frame of evaluation metrics

@ `predictions` : RasterStack of model predictions

@ `models`: list of MaxEnt model objects (see `MaxEnt` documentation for details)

@ `partition.method`: character giving method of data partitioning

@ `occ.pts` : data.frame of latitude and longitude of occurrence localities

@ `occ.grp` : data.frame of bins for occurrence localities

@ `bg.pts` : data.frame of latitude and longitude of background localities

@ `bg.grp` : data.frame of bins for background localities

@ `overlap` : matrix of pairwise niche overlap

## Details

The dataset is based on the simulated dataset and call of `ENMevaluate` shown in the example section below.

## Examples

```
require(raster)

### Simulated data environmental covariates
set.seed(1)
r1 <- raster(matrix(nrow=50, ncol=50, data=runif(10000, 0, 25)))
r2 <- raster(matrix(nrow=50, ncol=50, data=rep(1:100, each=100), byrow=TRUE))
r3 <- raster(matrix(nrow=50, ncol=50, data=rep(1:100, each=100)))
r4 <- raster(matrix(nrow=50, ncol=50, data=c(rep(1,1000),rep(2,500)),byrow=TRUE))
values(r4) <- as.factor(values(r4))
env <- stack(r1,r2,r3,r4)

### Simulate occurrence localities
nocc <- 50
x <- (rpois(nocc, 2) + abs(rnorm(nocc)))/11
y <- runif(nocc, 0, .99)
```

```
occ <- cbind(x,y)

## Not run:
### This gives the results that are loaded below:
enmeval_results <- ENMevaluate(occ, env, method="block", n.bg=500,
  categoricals=4, algorithm='maxent.jar')

## End(Not run)

data(enmeval_results)
enmeval_results

### See table of evaluation metrics
enmeval_results@results

### Plot prediction with lowest AICc
plot(enmeval_results@predictions[[which (enmeval_results@results$delta.AICc == 0) ]])
points(enmeval_results@occ.pts, pch=21, bg= enmeval_results@occ.grp)

### Niche overlap statistics between model predictions
enmeval_results@overlap
```

---

`eval.plot` *Generate Basic Plot for ENMevaluate Output*

---

## Description

This function can be used to generate a basic plot of evaluation metrics generated by a call of `ENMevaluate`.

## Usage

```
eval.plot(results, value = "delta.AICc", variance = NULL, legend = TRUE,
legend.position = "topright")
```

## Arguments

| | |
|---|---|
| `results` | A data.frame of results from `ENMevaluate`. |
| `value` | Character string of the column of `results` to use for plotting. |
| `variance` | Character string of the column of `results` to be used for error bars. |
| `legend` | logical; If `TRUE` (default), includes legend in plot with `fcs`. |
| `legend.position` | Character string for the placement of the legend. |

## Author(s)

Robert Muscarella <bob.muscarella@gmail.com>

## Examples

```
data(enmeval_results)

par(mfrow=c(2,2))
eval.plot(enmeval_results@results, legend.position="topright")
eval.plot(enmeval_results@results, "Mean.AUC", )
eval.plot(enmeval_results@results, "Mean.AUC.DIFF", variance="Var.AUC.DIFF")
eval.plot(enmeval_results@results, "Mean.ORmin")
```

---

# eval2 *An object of class "ENMevaluation"*

---

## Description

An example results file based on a call of ENMevaluate for use in the ENMeval vignette.

## Usage

```
data(eval2)
```

## Format

An object of class 'ENMevaluation' with nine slots:

@ `results` : data.frame of evaluation metrics

@ `predictions` : RasterStack of model predictions

@ `models`: list of `MaxEnt` model objects (see `MaxEnt` documentation for details)

@ `partition.method`: character giving method of data partitioning

@ `occ.pts` : data.frame of latitude and longitude of occurrence localities

@ `occ.grp` : data.frame of bins for occurrence localities

@ `bg.pts` : data.frame of latitude and longitude of background localities

@ `bg.grp` : data.frame of bins for background localities

@ `overlap` : matrix of pairwise niche overlap

## Details

The dataset is used for the ENMeval vignette.

get.evaluation.bins *Methods to partition data for evaluation*

## Description

ENMeval provides six methods to partition occurrence and background localities into bins for training and testing (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.

## Usage

```
get.block (occ, bg.coords)
get.checkerboard1(occ, env, bg.coords, aggregation.factor)
get.checkerboard2(occ, env, bg.coords, aggregation.factor)
get.jackknife(occ, bg.coords)
get.randomkfold(occ, bg.coords, kfolds)
get.user(occ.grp, bg.grp)
```

## Arguments

| | |
|---|---|
| `occ` | Two-column matrix or data.frame of longitude and latitude (in that order) of occurrence localities. |
| `bg.coords` | Two-column matrix or data.frame of longitude and latitude (in that order) of background localities. |
| `env` | RasterStack of environmental predictor variables. |
| `aggregation.factor` | A vector or list of 1 or 2 numbers giving the scale for aggregation used for the `get.checkerboard1` and `get.checkerboard2` methods. If a single number is given and `get.checkerboard2` partitioning method is used, the single value is used for both scales of aggregation. |
| `kfolds` | Number of random *k*-folds for `get.randomkfold` method. |
| `occ.grp` | Vector of user-defined bins for occurrence localities for `get.user` method. |
| `bg.grp` | Vector of user-defined bins for background localities for `get.user` method. |

## Details

These functions are used internally to partition data during a call of `ENMevaluate`.

The `get.block` method partitions occurrence localities by finding the latitude and longitude that divide the occurrence localities into four groups of (insofar as possible) equal numbers. 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.checkerboard1` and `get.checkerboard2` methods are variants of a checkerboard approach to partition occurrence localities. These methods use the `gridSample` function of the **dismo** package (Hijmans *et al.* 2011) to partition records according to checkerboard grids across the study extent. The spatial grain of these grids is determined by resampling (or aggregating) the original environmental input grids based on the user-defined `aggregation factor` (e.g., an aggregation factor of 2 results in a checkerboard with grid cells four times as large in area as the original input grids). The `get.checkerboard1` method partitions data into two groups according to a single checkerboard pattern, and the `get.checkerboard2` method partitions data into four groups according to two nested checkerboard grids. In contrast to the `get.block` method, both the `get.checkerboard1` and `get.checkerboard2` methods subdivide geographic space equally but do not ensure a balanced number of occurrence localities in each group. The two `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 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 valiation currently provided by Maxent.

The `get.user` method is flexible and enables users to define evaluation bins *a priori*. With this method, occurrence and background localities, as well as evaluation bin designation for each locality, are supplied by the user.

## Value

A named list of two items:

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

## Note

The `checkerboard1` and `checkerboard2` methods are designed to partition occurrence localities into two and four evaluation bins, respectively. 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 the "black" squares). A warning is given if the number of bins is < 4 for the `checkerboard2` 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 <jkass@gc.cuny.edu>

## References

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

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.

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.

## Examples

```
require(raster)

set.seed(1)

### Create environmental extent (raster)
env <- raster(matrix(nrow=25, ncol=25))

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

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

### Show points
plot(env)
points(bg.pts)
points(occ.pts, pch=21, bg=2)

### Block partitioning method
blk.pts <- get.block(occ.pts, bg.pts)
plot(env)
points(occ.pts, pch=23, bg=blk.pts$occ.grp)
plot(env)
points(bg.pts, pch=21, bg=blk.pts$bg.grp)

### Checkerboard1 partitioning method
chk1.pts <- get.checkerboard1(occ.pts, env, bg.pts, 4)
plot(env)
points(occ.pts, pch=23, bg=chk1.pts$occ.grp)
plot(env)
points(bg.pts, pch=21, bg=chk1.pts$bg.grp)

### Checkerboard2 partitioning method
chk2.pts <- get.checkerboard2(occ.pts, env, bg.pts, c(2,2))
plot(env)
points(occ.pts, pch=23, bg=chk2.pts$occ.grp)
plot(env)
```

```
points(bg.pts, pch=21, bg=chk2.pts$bg.grp)

### Random k-fold partitions
# Note that k random does not partition the background
krandom.pts <- get.randomkfold(occ.pts, bg.pts, 4)
plot(env)
points(occ.pts, pch=23, bg=krandom.pts$occ.grp)
plot(env)
points(bg.pts, pch=21, bg=krandom.pts$bg.grp)

### k-1 jackknife partitions
# Note background is not partitioned
jack.pts <- get.jackknife(occ.pts, bg.pts)
plot(env)
points(occ.pts, pch=23, bg=rainbow(length(jack.pts$occ.grp)))
plot(env)
points(bg.pts, pch=21, bg=jack.pts$bg.grp)

### User-defined partitions
# Note background is not partitioned
occ.grp <- c(rep(1, 10), rep(2, 5), rep(3, 10))
bg.grp <- c(rep(1, 200), rep(2, 100), rep(3, 200))
user.pts <- get.user(occ.grp, bg.grp)
plot(env)
points(occ.pts, pch=23, bg=user.pts$occ.grp)
plot(env)
points(bg.pts, pch=21, bg=user.pts$bg.grp)
```

---

make.args *Generate arguments for Maxent*

---

## Description

This function generates a list of arguments to pass to Maxent or to use as convenient labels for plotting.

## Usage

```
make.args(RMvalues = seq(0.5, 4, 0.5),
fc = c("L", "LQ", "H", "LQH", "LQHP", "LQHPT"),
labels = FALSE)
```

## Arguments

| | |
|---|---|
| RMvalues | Vector of (non-negative) values to use for the regularization multiplier. |
| fc | Character vector of feature class combinations to be included in analysis. |
| labels | logical; If FALSE (default), provides arguments to pass directly to Maxent; if TRUE, provides more intuitive labels to use, for example, in plotting. |

## Details

When `labels = FALSE`, the following additional arguments are added:

`noaddsamplestobackground,noremoveDuplicates,noautofeature`.

For details on these arguments, see Phillips *et al.* (2006) and the help documentation and tutorial of the Maxent software and the tutorial that can be downloaded from this website.

## Value

If `labels = FALSE`, a `list` the length of the total number of unique combinations of feature class(es) and regularization multiplier(s).

If `labels = TRUE`, a `list` of two items:

`$` character vector of feature class combinations in the same order they were provided.

`$` numeric vector of regularization multiplier values in the same order they were provided.

## Author(s)

Robert Muscarella <bob.muscarella@gmail.com> and Jamie M. Kass <jkass@gc.cuny.edu>

## References

Phillips, S. J., Anderson, R. P. and Schapire, R. E. 2006. Maximum entropy modeling of species geographic distributions. *Ecological Modelling*, **190**: 231-259.

## See Also

`maxent` in the **dismo** package.

## Examples

```
make.args(RMvalues=c(1:3), fc=c("L","LQ"))

make.args(RMvalues=c(1:3), fc=c("L","LQ"), labels=TRUE)
```

---

`maxnet.functions` *Functions for compatability with maxnet package*

---

## Description

These functions are used internally for compatability with the **maxnet** package (Phillips *et al.* 2017, Phillips 2017).

## Usage

```
maxentJARversion ()
modelTune.maxentJar (pres, bg, env, nk, group.data,
                      args.i, userArgs, rasterPreds, clamp, categoricals)
modelTune.maxnet (pres, bg, env, nk, group.data, args.i, rasterPreds, clamp)
maxnet.predictRaster (mod, env, type, clamp)
```

## Arguments

| | |
|---|---|
| pres | Occurrence points. |
| bg | Background points. |
| env | Environmental predictor variables. |
| nk | Number of k-fold partitions. |
| group.data | Input data grouped for k-fold evaluations (output of the get.evaluation.bins functions). |
| args.i | Internal arguments. |
| userArgs | User arguments. |
| rasterPreds | Raster(s) of model predictions. |
| clamp | Logical. |
| categoricals | Vector indicating which (if any) of the input environmental layers are categorical. |
| mod | maxnet model object. |
| type | see **maxnet** package documentation. |

## Details

These functions are used internally for compatability with the **maxnet** package.

## Value

Depends on which function is used.

## Author(s)

Robert Muscarella <bob.muscarella@gmail.com> and Jamie M. Kass <jkass@gc.cuny.edu>

## References

Phillips, S. J. 2017. maxnet package for R. Available online at: https://CRAN.R-project.org/package=maxnet.

Phillips, S. J., Anderson, R. P., Dudík, M., Schapire, R. E. and Blair, M. E. 2017. Opening the black box: an open-source release of Maxent. *Ecography*, **40**: 887–893.

`var.importance` *Extract percent contribution and permutation importance from a Maxent model*

## Description

Extract the percent contribution and permutation importance metrics generated by a Maxent model.

## Usage

```
var.importance(mod)
```

## Arguments

`mod` A Maxent model object.

## Details

Maxent provides two metrics to determine the importance of input variables in the final model: **percent contribution** and **permutation importance**. This function extracts both metrics from the results slot of a maxent model object and places them into a `data.frame`.

*According to Phillips (2006), the **percent contribution** of each variable is calculated as follows:*

"While the Maxent model is being trained, it keeps track of which environmental variables are contributing to fitting the model. Each step of the Maxent algorithm increases the gain of the model by modifying the coefficient for a single feature; the program assigns the increase in the gain to the environmental variable(s) that the feature depends on. Converting to percentages at the end of the training process, we get the percent contribution."

"The percent contribution values are only heuristically defined: they depend on the particular path that the Maxent code uses to get to the optimal solution, and a different algorithm could get to the same solution via a different path, resulting in different percent contribution values. In addition, when there are highly correlated environmental variables, the percent contributions should be interpreted with caution."

*Also according to Phillips (2006), the **permutation importance** of each variable is calculated as follows:*

"...for each environmental variable in turn, the values of that variable on training presence and background data are randomly permuted. The model is reevaluated on the permuted data, and the resulting drop in training AUC is shown in the table, normalized to percentages."

"The permutation importance measure depends only on the final Maxent model, not the path used to obtain it. The contribution for each variable is determined by randomly permuting the values of that variable among the training points (both presence and background) and measuring the resulting decrease in training AUC. A large decrease indicates that the model depends heavily on that variable. Values are normalized to give percentages."

## Value

A `data.frame` with the percent contribution and permutation importance for each variable included in a Maxent model.

## Note

Both metrics should be interpreted with caution when the predictor variables are correlated (Phillips 2006).

## Author(s)

Jamie M. Kass <jkass@gc.cuny.edu> and Robert Muscarella <bob.muscarella@gmail.com>

## References

Phillips, S. (2006) A brief tutorial on Maxent. AT&T Research. Available at: https://www.cs.princeton.edu/~schapire/maxent/

## Examples

```
data(enmeval_results)

# Select model with lowest AICc

aic.mod <- enmeval_results@models[[which(enmeval_results@results$delta.AICc==0)]]
var.importance(aic.mod)

# See the variable importance metrics for the first 3 models

lapply(enmeval_results@models, var.importance)[1:3]
```

# Index