ENMeval Vignette

Robert Muscarella, Jamie M. Kass, and Peter Galante

2017-01-09

Introduction

`ENMeval` is an R package that performs automated runs and evaluations of ecological niche models, and currently only implements Maxent. `ENMeval` was made for those who want to "tune" their models to maximize predictive ability and avoid overfitting, or in other words, optimize model complexity to balance goodness-of-fit and predictive ability. The primary function, `ENMevaluate`, does all the heavy lifting and returns several items including a table of evaluation statistics and, for each setting combination (here, colloquially: *runs*), a model object and a raster layer showing the model prediction across the study extent. There are also options for calculating niche overlap between predictions, running in parallel to speed up computation, and more. For a more detailed description of the package, check out the open-access publication:

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.

Data Acquisition & Pre-processing

In this vignette, we briefly demonstrate acquisition and pre-processing of input data for `ENMeval`. There are a number of other excellent tutorials on these steps, some of which we compiled in the Resources section.

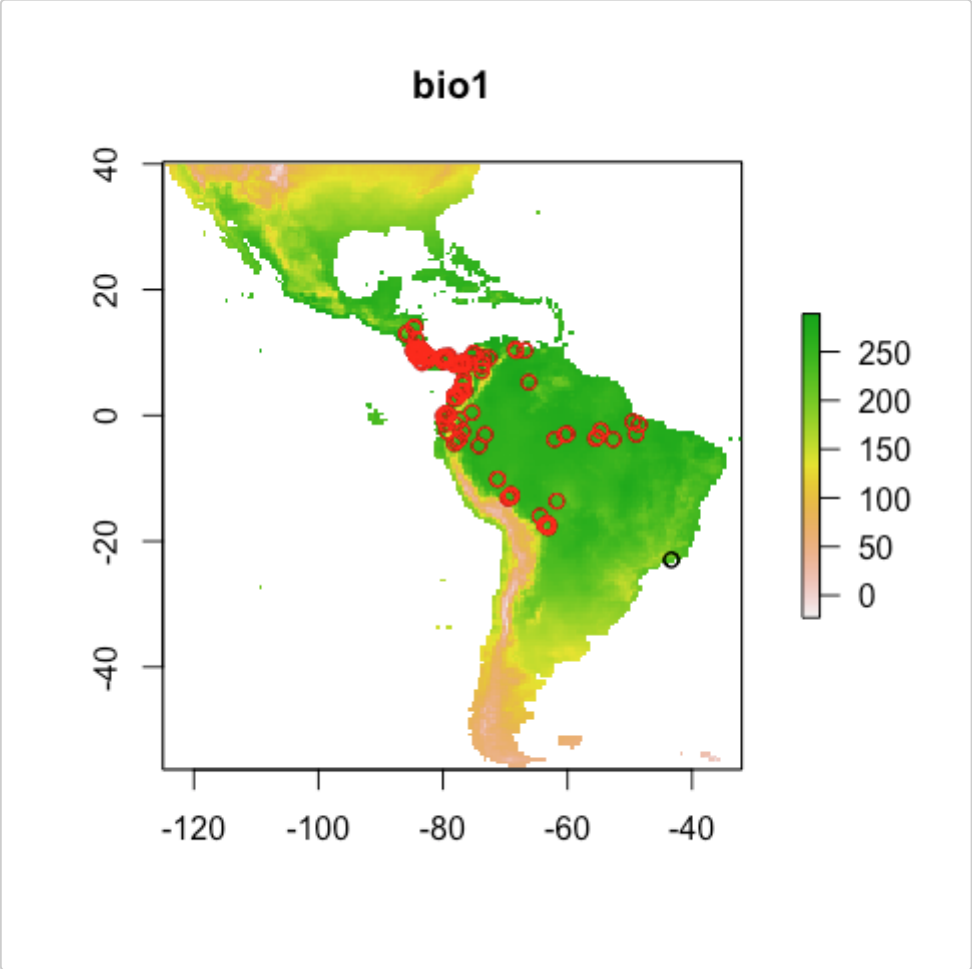
We'll start by downloading an occurrence dataset for *Bradypus variegatus*, the Brown-throated sloth. We'll go ahead and load the `ENMeval` and `spocc` packages. (We are using `spocc` to download occurrence records).

```
if (!require('spocc')) install.packages('spocc', repos="http://cran.us.r-project.org")
if (!require('ENMeval')) install.packages('ENMeval', repos="http://cran.us.r-project.org")
library(spocc)
library(ENMeval)

# Search GBIF for occurrence data.
bv <- occ('Bradypus variegatus', 'gbif', limit=300, has_coords=TRUE)

# Get the latitude/coordinates for each locality. Also convert the tibble that occ() outputs
# to a data frame for compatibility with ENMeval functions.
occs <- as.data.frame(bv$gbif$data$Bradypus_variegatus[,2:3])
```

```
# Remove duplicate rows (Note that you may or may not want to do this).
occs <- occs[!duplicated(occs),]
```

We are going to model the climatic niche suitability for our focal species using climate data from WorldClim. WorldClim has a range of variables available at various resolutions; for simplicity, here we'll use the 9 bioclimatic variables at 10 arcmin resolution (about 20 km across at the equator) included in the `dismo` package. These climatic data are based on 50-year averages from 1950-2000. Now's also a good time to load the package, as it includes all the downstream dependencies (`raster`, `dismo`, etc.).

```
# First, load some predictor rasters from the dismo folder:
files <- list.files(path=paste(system.file(package='dismo'), '/ex', sep=''), pattern='grd',
full.names=TRUE)

# Put the rasters into a RasterStack:
envs <- stack(files)

# Plot first raster in the stack, bio1
plot(envs[[1]], main=names(envs)[1])

# Add points for all the occurrence points onto the raster
points(occs)

# There are some points all the way to the south-east, far from all others. Let's say we know that
this represents a subpopulation that we don't want to include, and want to remove these points from
the analysis. We can find them by first sorting the occs table by latitude.
head(occs[order(occs$latitude),])
#>     longitude  latitude
#> 21  -43.40630 -23.00829
#> 83  -43.23764 -22.93862
#> 110 -63.16667 -17.78333
#> 1   -63.06480 -17.77664
#> 78  -63.17032 -17.74494
#> 2   -63.65988 -17.46079

# We see there are two such points, and we can find them by specifying a logical statement that says
to find all records with latitude less than -20.
index <- which(occs$latitude < (-20))

# Next, let's subset our dataset to remove them by using the negative assignment on the index vector.
occs <- occs[-index,]

# Let's plot our new points over the old ones to see what a good job we did.
points(occs, col='red')
```

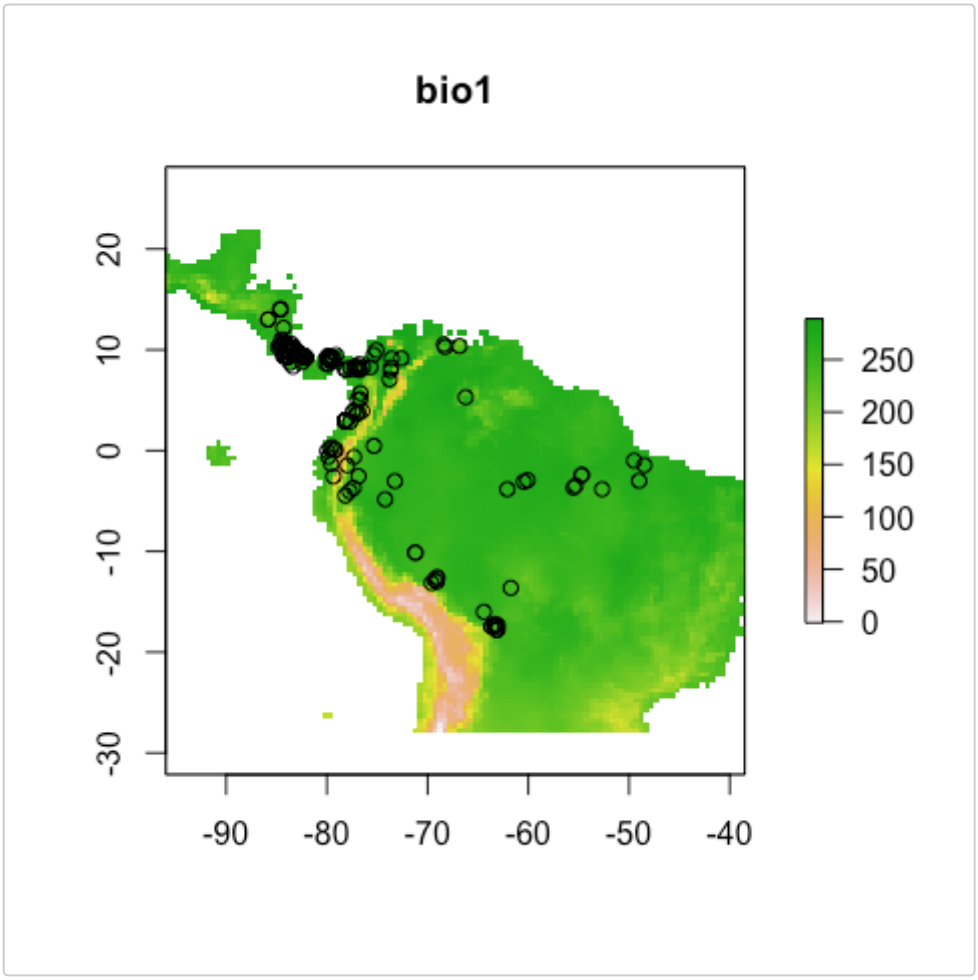


Next, we will specify the background extent by cropping (or "clipping" in ArcGIS terms) our global predictor variable rasters to a smaller region. Since our models will compare the environment at occurrence (or, presence) localities to the environment at background localities, we need to sample random points from a background extent. To help ensure we don't include areas that are suitable for our species but are unoccupied due to limitations like dispersal constraints, we will conservatively define the background extent as an area surrounding our occurrence localities. We will do this by buffering a bounding box that includes all occurrence localities. Some other methods of background extent delineation (e.g., minimum convex hulls) are more conservative because they better characterize the geographic space holding the points. In any case, this is one of the many things that you will need to carefully consider for your own study.

```
# Make a SpatialPoints object
occs.sp <- SpatialPoints(occs)

# Get the bounding box of the points
bb <- bbox(occs.sp)

# Add 5 degrees to each bound by stretching each bound by 10, as the resolution is 0.5 degree.
bb.buf <- extent(bb[1]-10, bb[3]+10, bb[2]-10, bb[4]+10)

# Crop environmental layers to match the study extent
envs.backg <- crop(envs, bb.buf)
```

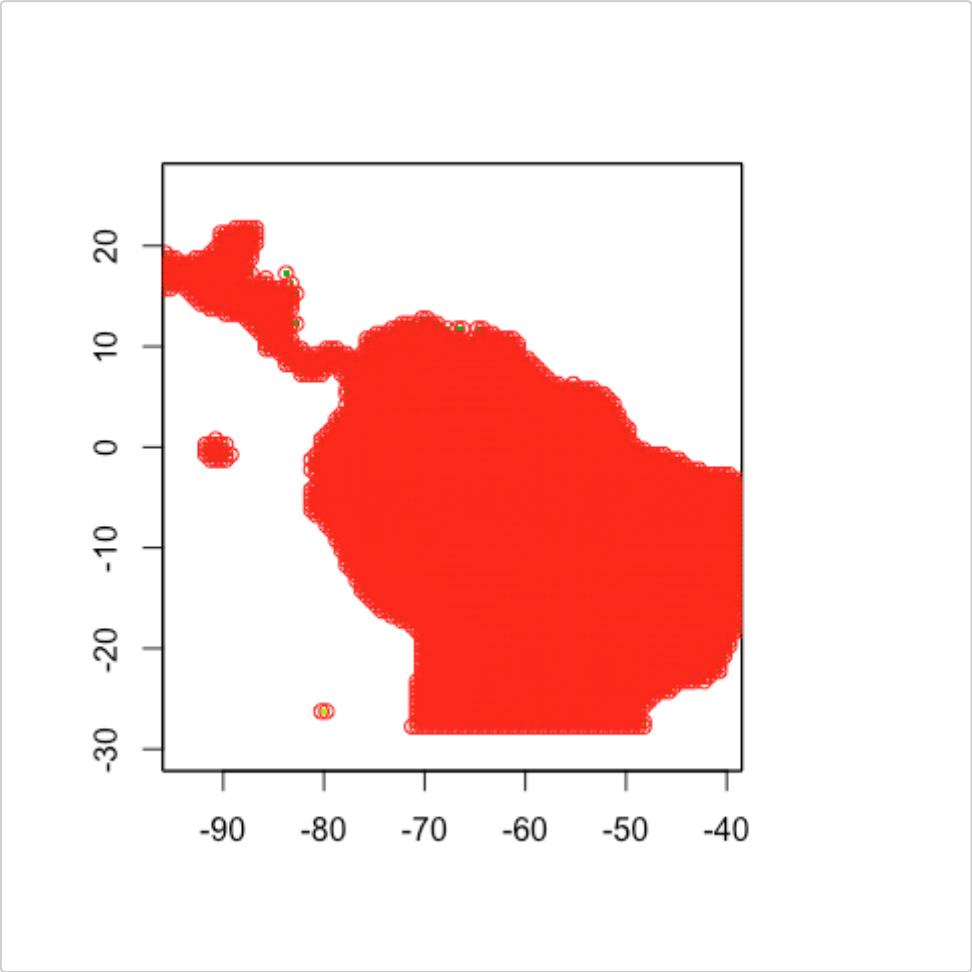
We may also, however, want to remove the Caribbean islands (for example) from our background extent. For this, we can use tools from the `maptools` package, which is not automatically loaded with `ENMeval`.

```
if (!require('maptools')) install.packages('maptools', repos="http://cran.us.r-project.org")
if (!require('rgeos')) install.packages('rgeos', repos="http://cran.us.r-project.org")
library(maptools)
library(rgeos)

# Get a simple world countries polygon
data(wrld_simpl)

# Get polygons for Central and South America
ca.sa <- wrld_simpl[wrld_simpl@data$SUBREGION==5 | wrld_simpl@data$SUBREGION==13,]

# Both spatial objects have the same geographic coordinate system with slightly different
specifications, so just name the coordinate reference system (crs) for ca.sa with that of
# envs.backg to ensure smooth geoprocessing.
crs(envs.backg) <- crs(ca.sa)

# Mask envs by this polygon after buffering a bit to make sure not to lose coastline.
ca.sa <- gBuffer(ca.sa, width=1)
envs.backg <- mask(envs.backg, ca.sa)

# Let's check our work. We should see Central and South America without the Carribbean.
plot(envs.backg[[1]], main=names(envs.backg)[1])
points(occs)
```



In the next step, we'll sample 10,000 random points from the background (note that the number of background points is also a consideration you should make with respect to your own study).

```
# Randomly sample 10,000 background points from one background extent raster (only one per cell
without replacement). Note: Since the raster has <10,000 pixels, you'll get a warning and all pixels
will be used for background.
bg <- randomPoints(envs.backg[[1]], n=10000)
bg <- as.data.frame(bg)

# Notice how we have pretty good coverage (every cell).
plot(envs.backg[[1]], legend=FALSE)
points(bg, col='red')
```



Partitioning Occurrences for Evaluation

A run of ENMevaluate begins by using one of six methods to partition occurrence localities into testing and training bins (folds) for k-fold cross-validation (Fielding and Bell 1997; Peterson et al. 2011). Generally, the data partitioning step is done within the main 'ENMevaluate' function call. In this section, we illustrate the different options.

1. Block
2. Checkerboard1
3. Checkerboard2
4. k-1 Jackknife

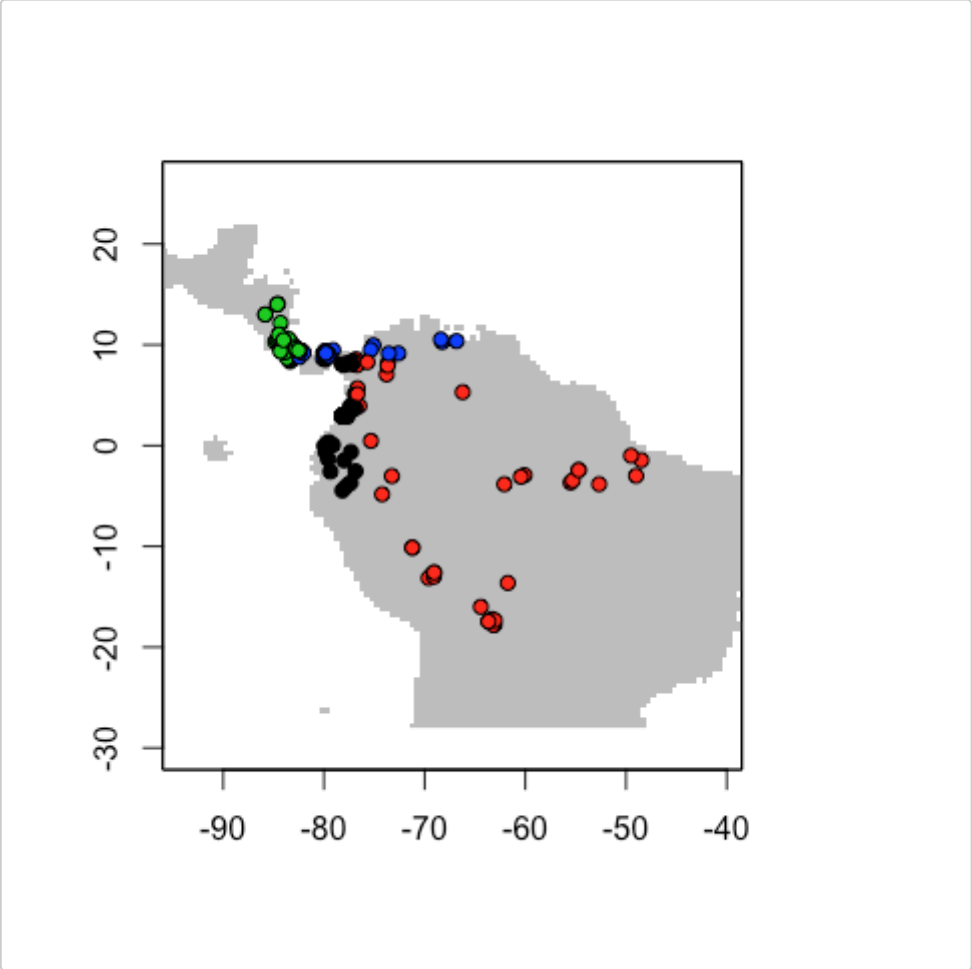
5. Random k-fold
6. User-defined

The first three partitioning methods are variations of what Radosavljevic and Anderson (2014) referred to as ‘masked geographically structured’ data partitioning. Basically, these methods partition both occurrence records and background points into evaluation bins based on some spatial rules. The intention is to reduce spatial-autocorrelation between points that are included in the testing and training bins, which can overinflate model performance, at least for data sets that result from biased sampling (Veloz 2009; Hijmans 2012; Wenger and Olden 2012).

1. Block

First, the ‘block’ method partitions data according to the latitude and longitude lines that divide the occurrence localities into four bins of (insofar as possible) equal numbers. Both occurrence and background localities are assigned to each of the four bins based on their position with respect to these lines. The resulting object is a list of two vectors that supply the bin designation for each occurrence and background point.

```
blocks <- get.block(occs, bg)
str(blocks)
#> List of 2
#>  $ occ.grp: num [1:172] 2 2 2 3 3 4 1 4 3 4 ...
#>  $ bg.grp : num [1:5381] 2 2 2 4 4 2 2 2 1 2 ...

plot(envs.backg[[1]], col='gray', legend=FALSE)
points(occs, pch=21, bg=blocks$occ.grp)
```

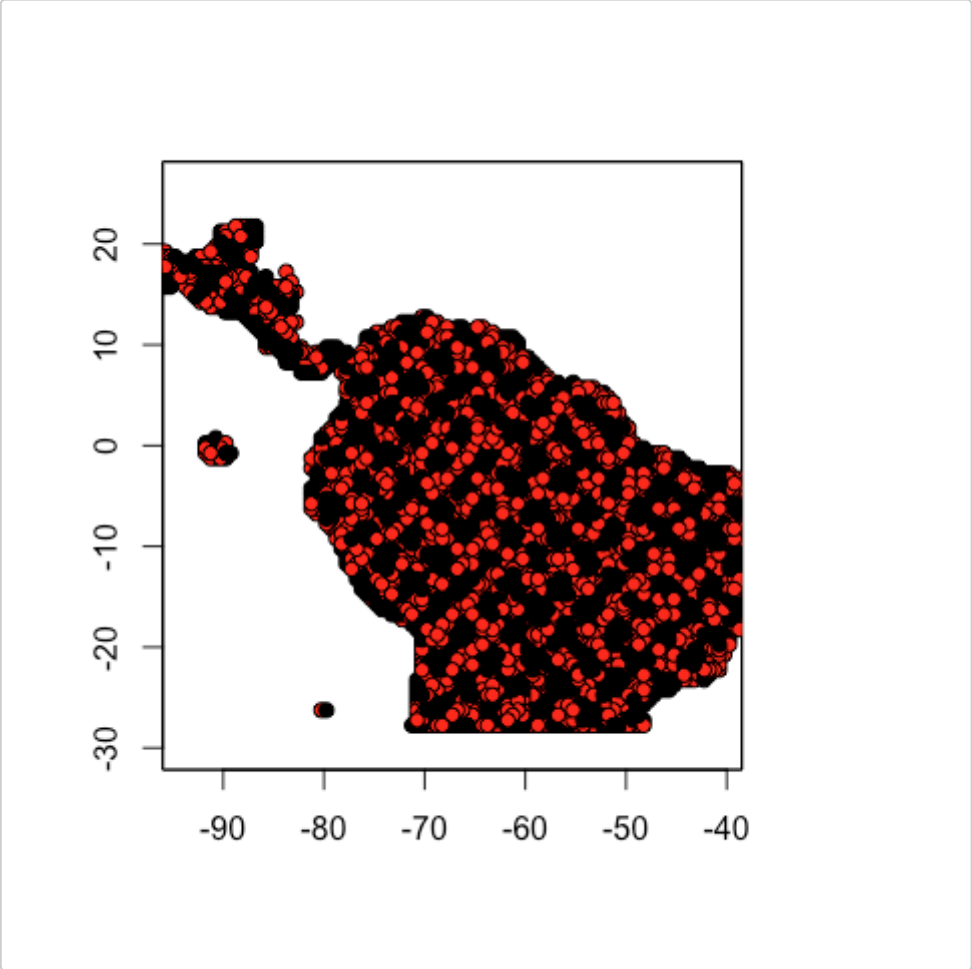


2. Checkerboard1

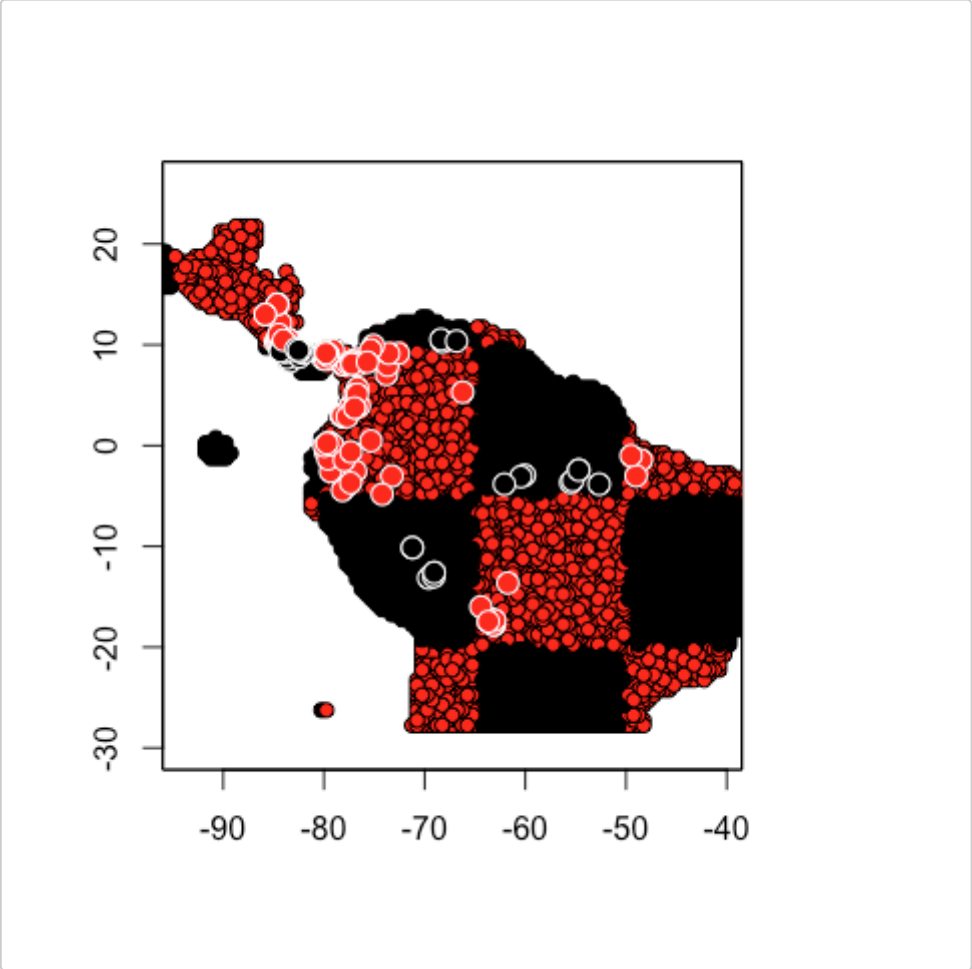
The next two partitioning methods are variants of a 'checkerboard' approach to partition occurrence localities. These generate checkerboard grids across the study extent and partition the localities into bins based on where they fall in the checkerboard. In contrast to the block method, both checkerboard methods subdivide geographic space equally but do not ensure a balanced number of occurrence localities in each bin. For these methods, the user needs to provide a raster layer on which to base the underlying checkerboard pattern. Here we simply use the predictor variable RasterStack. Additionally, the user needs to define an *aggregation.factor*. This value tells the number of grids cells to aggregate when making the underlying checkerboard pattern.

The Checkerboard1 method partitions the points into k=2 bins using a simple checkerboard pattern.

```
check1 <- get.checkerboard1(occs, envs, bg, aggregation.factor=5)

plot(envs.backg[[1]], col='gray', legend=FALSE)
points(occs, pch=21, bg=check1$occ.grp)

# The partitioning method is more clearly illustrated by looking at the background points:
points(bg, pch=21, bg=check1$bg.grp)
```



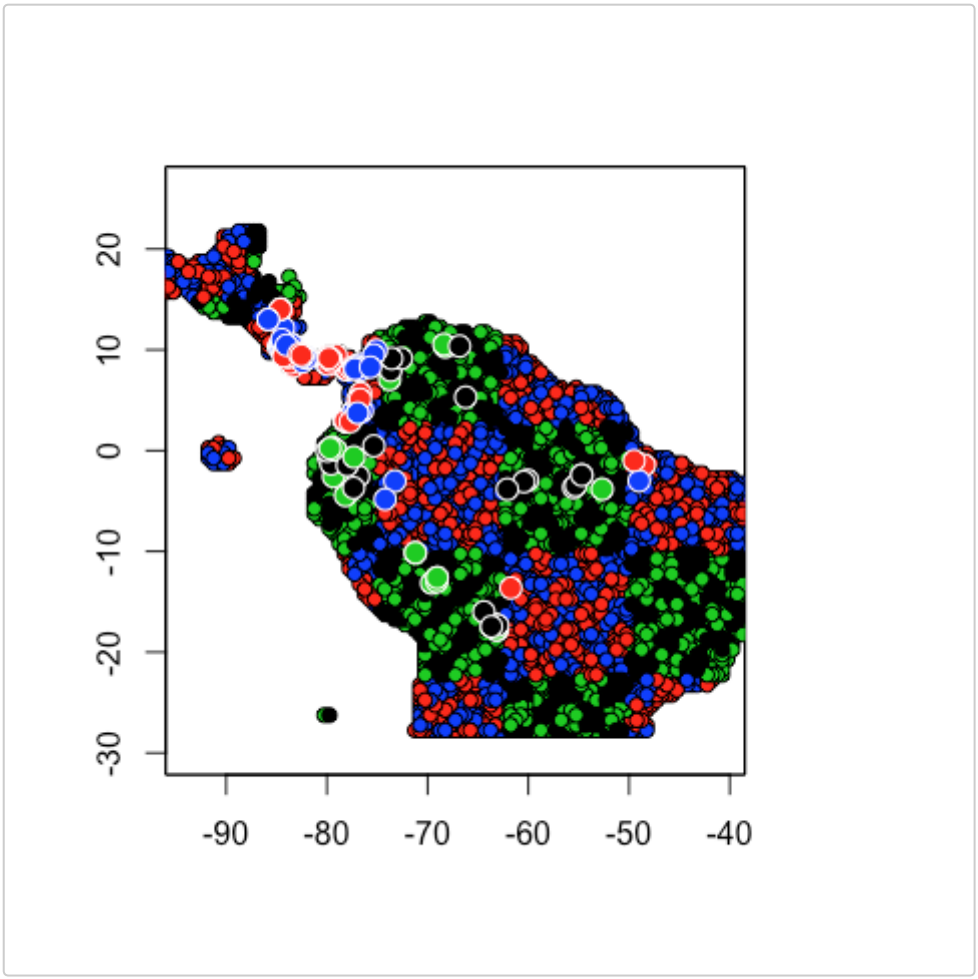

```
# We can change the aggregation factor to better illustrate how this partitioning method works:
check1.large <- get.checkerboard1(occs, envs, bg, aggregation.factor=30)
plot(envs.backg[[1]], col='gray', legend=FALSE)
points(bg, pch=21, bg=check1.large$bg.grp)
points(occs, pch=21, bg=check1.large$occ.grp, col='white', cex=1.5)
```



3. Checkerboard2

The Checkerboard2 method partitions the data into k=4 bins. This is done by aggregating the input raster at two scales. Presence and background points are assigned to a bin with respect to where they fall in checkerboards of both scales.

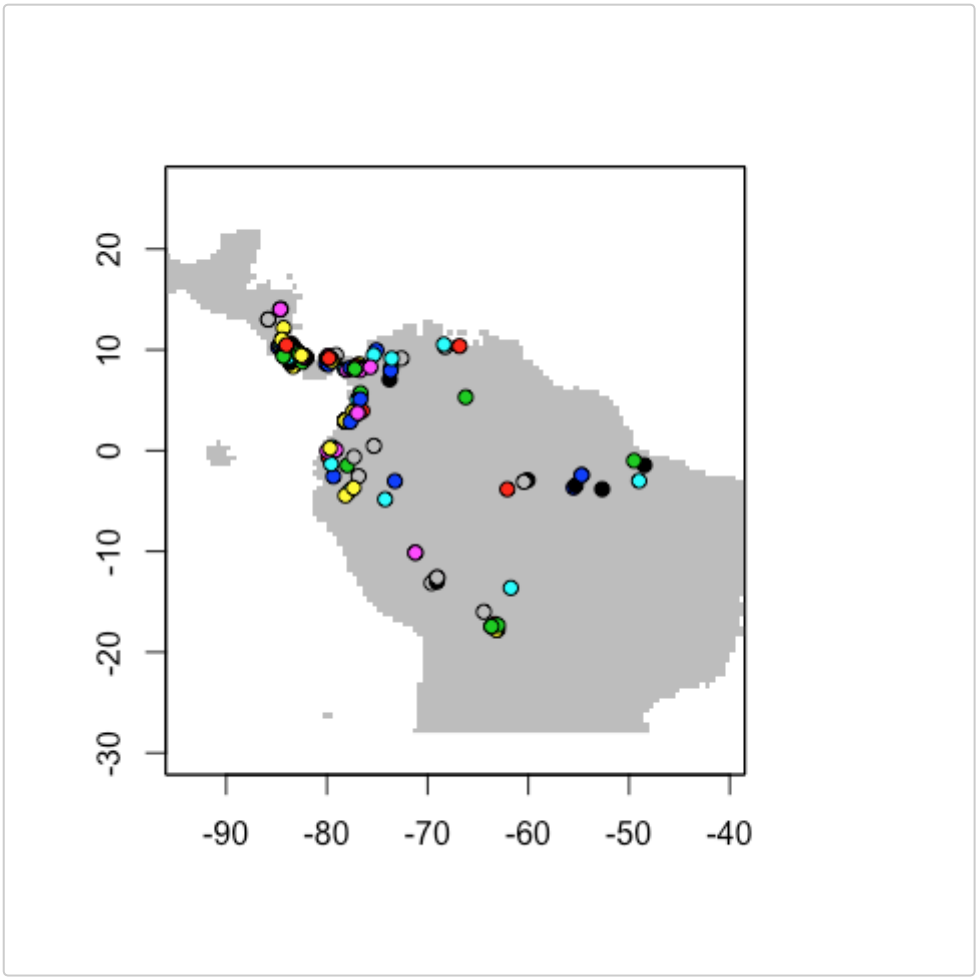
```
check2 <- get.checkerboard2(occs, envs, bg, aggregation.factor=c(5,5))

plot(envs.backg[[1]], col='gray', legend=FALSE)
points(bg, pch=21, bg=check2$bg.grp)
points(occs, pch=21, bg=check2$occ.grp, col='white', cex=1.5)
```



4. k-1 Jackknife

The next two methods differ from the first three in that (i) they do not partition the background points into different groups, and (ii) they do not account for spatial autocorrelation between testing and training localities. Primarily when working with relatively small data sets (e.g. < ca. 25 presence localities), users may choose 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 data set (Pearson et al. 2007; Shcheglovitova and Anderson 2013). This is referred to as the k-1 jackknife. This method will take prohibitively long times for computation when the number of presence localities is medium to large.

```
jack <- get.jackknife(occs, bg)

plot(envs.backg[[1]], col='gray', legend=FALSE)
points(occs, pch=21, bg=jack$occ.grp)  # note that colors are repeated here
```

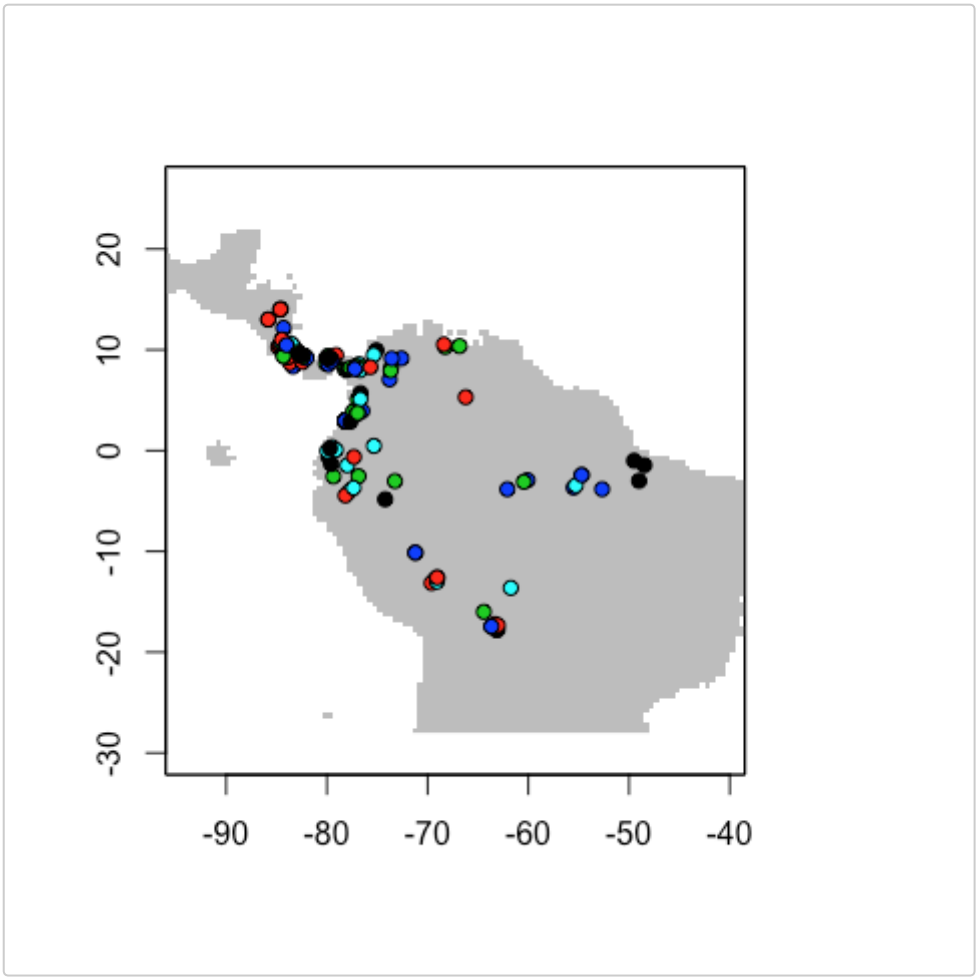


5. Random k-fold

The ‘random k-fold’ method partitions occurrence localities randomly into a user specified number of (k) bins. This method is equivalent to the ‘cross-validate’ partitioning scheme available in the current version of the Maxent software GUI.

For instance, let’s partition the data into five evaluation bins:

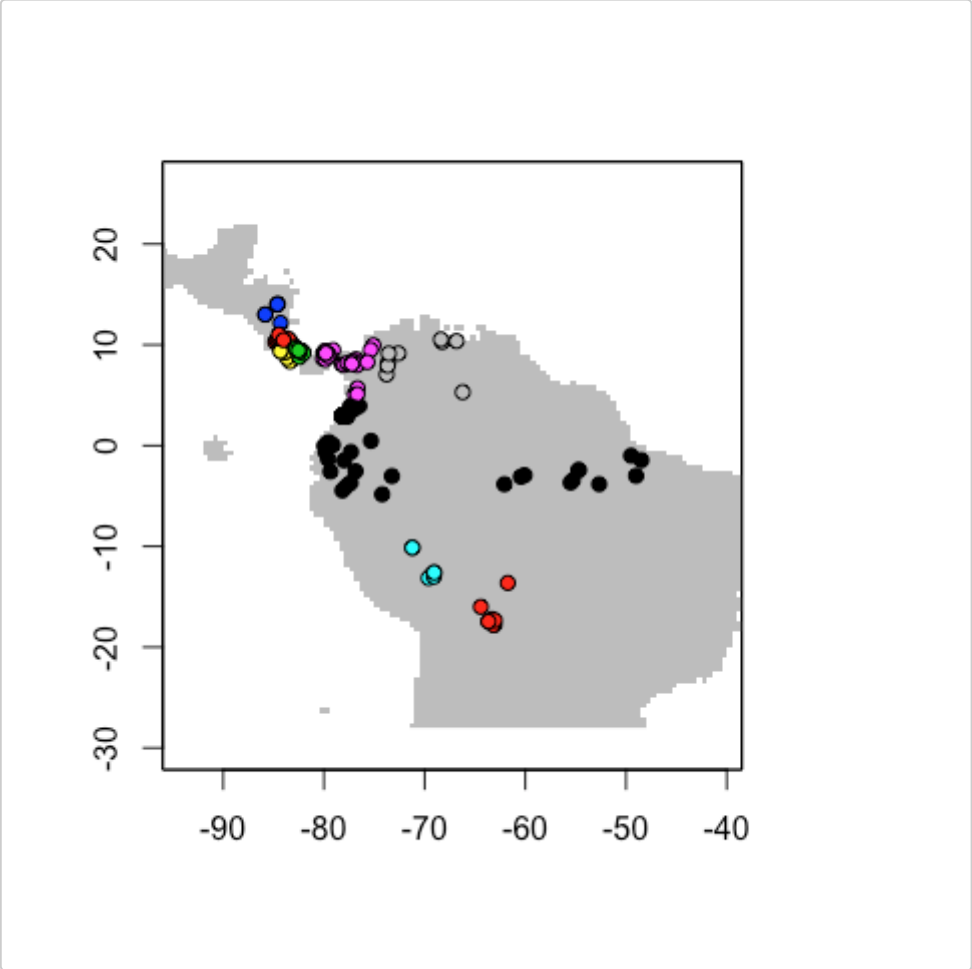
```
random <- get.randomkfold(occs, bg, k=5)

plot(envs.backg[[1]], col='gray', legend=FALSE)
points(occs, pch=21, bg=random$occ.grp)
```



6. User-defined

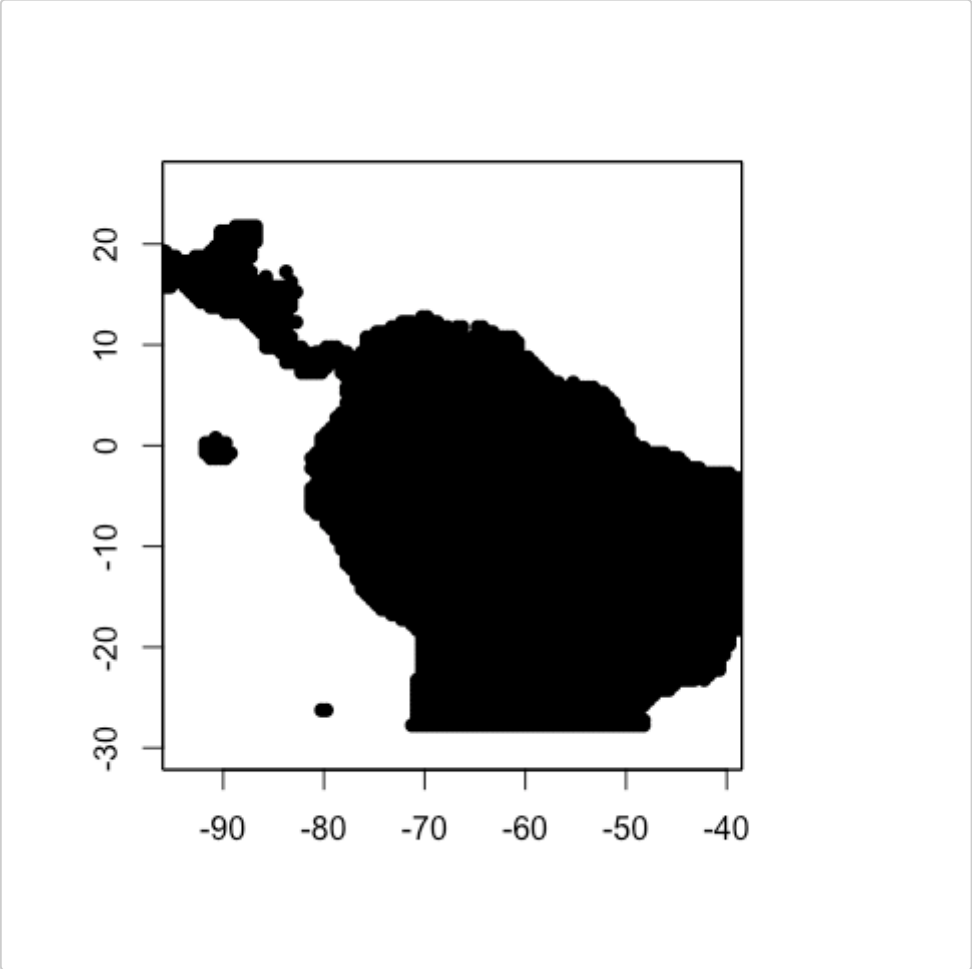
For maximum flexibility, the last partitioning method is designed so that users can define *a priori* partitions. This provides a flexible way to conduct spatially-independent cross-validation with background masking. For example, perhaps we would like to partition points based on a k-means clustering routine.

```
ngrps <- 10
kmeans <- kmeans(occs, ngrps)
occ.grp <- kmeans$cluster

plot(envs.backg[[1]], col='gray', legend=FALSE)
points(occs, pch=21, bg=occ.grp)
```



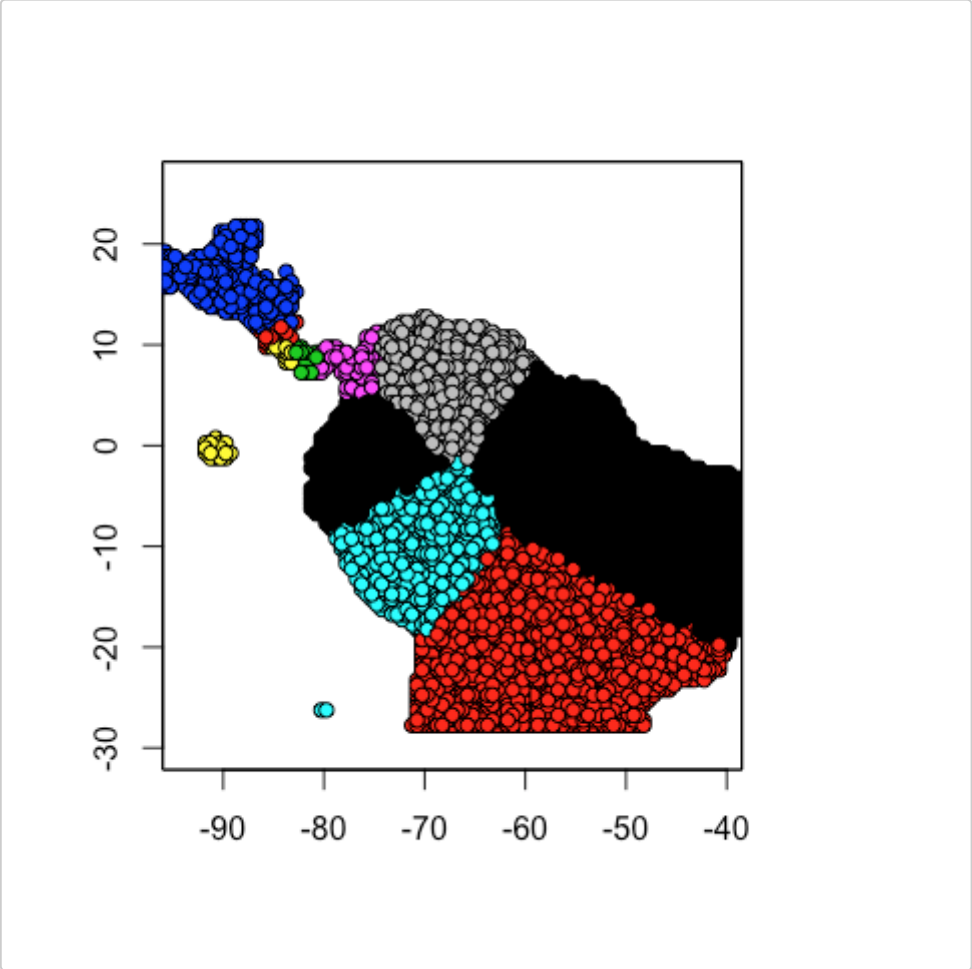
When using the user-defined partitioning method, we need to supply ENMevaluate with group identifiers for both occurrence points AND background points. If we want to use all background points for each group, we can set the background to zero.

```
bg.grp <- rep(0, nrow(bg))

plot(envs.backg[[1]], col='gray', legend=FALSE)
points(bg, pch=16, bg=bg.grp)
```



Alternatively, we may think of various ways to partition background data. This depends on the goals of the study but we might, for example, find it reasonable to partition background by clustering around the centroids of the occurrence clusters.

```
centers <- kmeans$center
d <- pointDistance(bg, centers, lonlat=T)
bg.grp <- apply(d, 1, function(x) which(x==min(x)))

plot(envs.backg[[1]], col='gray', legend=FALSE)
points(bg, pch=21, bg=bg.grp)
```



Choosing among these data partitioning methods depends on the research objectives and the characteristics of the study system. Refer to the Resources section for additional considerations on appropriate partitioning for evaluation.

Running ENMeval

Once you decide which method of data partitioning you would like to use, you are ready to start building models. We now move on to the main function in ENMeval: `ENMevaluate`.

- Initial considerations
- Exploring the results (the ENMevaluate object)

Initial considerations

The two main parameters to define when calling `ENMevaluate` are (1) the range of regularization multiplier values and (2) the combinations of feature class to consider. The ***regularization multiplier*** (RM) determines the penalty for adding parameters to the model. Higher RM values impose a stronger penalty on model complexity and thus result in simpler (*flatter*) model predictions. The ***feature classes*** determine the potential shape of the response curves. A model that is only allowed to include linear feature classes will most likely be simpler than a model that is allowed to include all possible feature classes. Much more description of these parameters is available in the Resources section. For the purposes of this vignette, we demonstrate simply how to adjust these parameters. The following section deals with comparing the outputs of each model.

Unless you supply the function with background points (which is recommended in many cases), you will need to define how many background points should be used with the 'n.bg' argument. If any of your predictor

variables are categorical (e.g., biomes), you will need to define which layer(s) these are using the 'categoricals' argument.

ENMevaluate builds a separate model for each unique combination of RM values and feature class combinations. For example, the following call will build and evaluate 2 models. One with RM=1 and another with RM=2, both allowing only linear features.

```
eval1 <- ENMevaluate(occs, envs, bg, method='checkerboard2', RMvalues=c(1,2), fc=c('L'))
```

We may, however, want to compare a wider range of models that can use a wider variety of feature classes:

```
eval2 <- ENMevaluate(occ=occs, env=envs, bg.coords=bg, method='checkerboard2', RMvalues=c(1,2),
fc=c('L','LQ','LQP'))
```

When building many models, the command may take a long time to run. Of course this depends on the size of your dataset and the computer you are using. When working on big projects, running the command in parallel can be faster.

```
eval2.par <- ENMevaluate(occs, envs, bg, method='checkerboard2', RMvalues=c(1,2),
fc=c('L','LQ','LQP'), parallel=TRUE)
```

Another way to save time at this stage is to turn off the option that generates model predictions across the full study extent (rasterPreds). Note, however, that the full model predictions are needed for calculating AICc values so those are returned as NA in the results table when the `rasterPreds` argument is set to FALSE.

```
eval3 <- ENMevaluate(occs, envs, bg, method='checkerboard2', RMvalues=c(1,2), fc=c('L','LQ','LQP'),
rasterPreds=FALSE)
```

We can also calculate one of two niche overlap statistics while running `ENMevaluate` by setting the `niche.overlap` argument, which supports Moran's I or Schoener's D. Note that you can also calculate this value at a later stage using the separate `calc.niche.overlap` function.

```
overlap <- calc.niche.overlap(eval2@predictions, stat='D')
```

```
overlap
#>             L_1      LQ_1     LQP_1       L_2      LQ_2 LQP_2
#> L_1          NA        NA        NA        NA        NA    NA
#> LQ_1  0.8557208        NA        NA        NA        NA    NA
#> LQP_1 0.8383703 0.9445758        NA        NA        NA    NA
#> L_2   0.9908049 0.8531787 0.8351710        NA        NA    NA
#> LQ_2  0.8628724 0.9894438 0.9420927 0.8604812        NA    NA
#> LQP_2 0.8529408 0.9436047 0.9719879 0.8501070 0.9445745    NA
```

The `bin.output` argument determines if separate evaluation statistics for each testing bin are included in the results file. If `bin.output=FALSE`, only the mean and variance of evaluation statistics across k bins is returned.

Exploring the results

Now let's take a look at the ENMeval object in more detail. It contains the following slots:

- A data.frame holding the model evaluation statistics
- A RasterStack of the model predictions
- A list of maxent model objects
- A data.frame of the original occurrence coordinates
- A vector of the evaluation bins used for the occurrence points
- A data.frame of the background coordinates
- A vector of the evaluation bins used for the background points
- (if `overlap=T`) A matrix of the pairwise niche overlap metric

Let's first examine the structure of the object:

```
eval2
#> An object of class: ENMevaluation
#>  Occurrence/background points: 172/5381
#>  Partition method: checkerboard2
#>  Feature classes: L, LQ, LQP
#>  Regularization multipliers: 1, 2
#>  @results     : data.frame of evaluation results
#>  @predictions : RasterStack of model predictions
#>  @models      : list of model objects
#>  @occ.pts     : data.frame of occurrence coordinates
#>  @occ.grp     : vector of bins for occurrence points
#>  @bg.pts      : data.frame of background coordinates
#>  @bg.grp      : vector of bins for background points
#>

str(eval2, max.level=3)
#> Formal class 'ENMevaluation' [package "ENMeval"] with 9 slots
#>   ..@ results         :'data.frame': 6 obs. of  16 variables:
#>   .. ..$ settings     : Factor w/ 6 levels "L_1","L_2","LQ_1",..: 1 3 5 2 4 6
#>   .. ..$ features     : Factor w/ 3 levels "L","LQ","LQP": 1 2 3 1 2 3
#>   .. ..$ rm           : num [1:6] 1 1 1 2 2 2
#>   .. ..$ full.AUC     : num [1:6] 0.863 0.871 0.871 0.863 0.871 ...
#>   .. ..$ Mean.AUC     : num [1:6] 0.842 0.849 0.829 0.842 0.847 ...
#>   .. ..$ Var.AUC      : num [1:6] 0.00835 0.00954 0.01588 0.00817 0.00972 ...
#>   .. ..$ Mean.AUC.DIFF: num [1:6] 0.0347 0.0356 0.0549 0.0348 0.036 ...
#>   .. ..$ Var.AUC.DIFF : num [1:6] 0.0106 0.0126 0.0197 0.0104 0.0127 ...
#>   .. ..$ Mean.OR10    : num [1:6] 0.101 0.116 0.144 0.101 0.116 ...
#>   .. ..$ Var.OR10     : num [1:6] 0.00491 0.0091 0.02273 0.00491 0.0091 ...
#>   .. ..$ Mean.ORmin   : num [1:6] 0.0445 0.0445 0.0548 0.0292 0.0445 ...
#>   .. ..$ Var.ORmin    : num [1:6] 0.00457 0.00457 0.00391 0.0015 0.00457 ...
#>   .. ..$ AICc         : num [1:6] 2637 2587 2614 2639 2589 ...
#>   .. ..$ delta.AICc   : num [1:6] 50.09 0 27.08 51.19 2.11 ...
#>   .. ..$ w.AIC        : num [1:6] 9.85e-12 7.42e-01 9.79e-07 5.69e-12 2.58e-01 ...
#>   .. ..$ nparam       : num [1:6] 7 11 26 7 11 28
#>   ..@ predictions     :Formal class 'RasterStack' [package "raster"] with 11 slots
#>   ..@ models          :List of 6
#>   .. ..$ :Formal class 'MaxEnt' [package "dismo"] with 7 slots
#>   .. ..$ :Formal class 'MaxEnt' [package "dismo"] with 7 slots
#>   .. ..$ :Formal class 'MaxEnt' [package "dismo"] with 7 slots
#>   .. ..$ :Formal class 'MaxEnt' [package "dismo"] with 7 slots
#>   .. ..$ :Formal class 'MaxEnt' [package "dismo"] with 7 slots
```

```
#>   .. ..$ :Formal class 'MaxEnt' [package "dismo"] with 7 slots
#>   ..@ partition.method: chr "checkerboard2"
#>   ..@ occ.pts         :'data.frame': 172 obs. of  2 variables:
#>   .. ..$ LON: num [1:172] -63.1 -63.7 -63.4 -84.8 -84 ...
#>   .. ..$ LAT: num [1:172] -17.8 -17.5 -17.4 10.3 10.5 ...
#>   ..@ occ.grp         : num [1:172] 1 1 1 3 3 4 2 3 2 4 ...
#>   ..@ bg.pts          :'data.frame': 5381 obs. of  2 variables:
#>   .. ..$ LON: num [1:5381] -63.8 -75.2 -90.8 -62.2 -49.8 ...
#>   .. ..$ LAT: num [1:5381] -9.25 0.25 -0.25 1.75 -7.25 ...
#>   ..@ bg.grp          : num [1:5381] 1 2 2 1 1 4 3 3 3 4 ...
#>   ..@ overlap         : num[0 , 0 ]
```

The first slot holds the table of evaluation metrics. We can use this to, for example, select the 'best' model based on one or more of our evaluation criteria. Let's find the model settings that resulted in the lowest AICc.

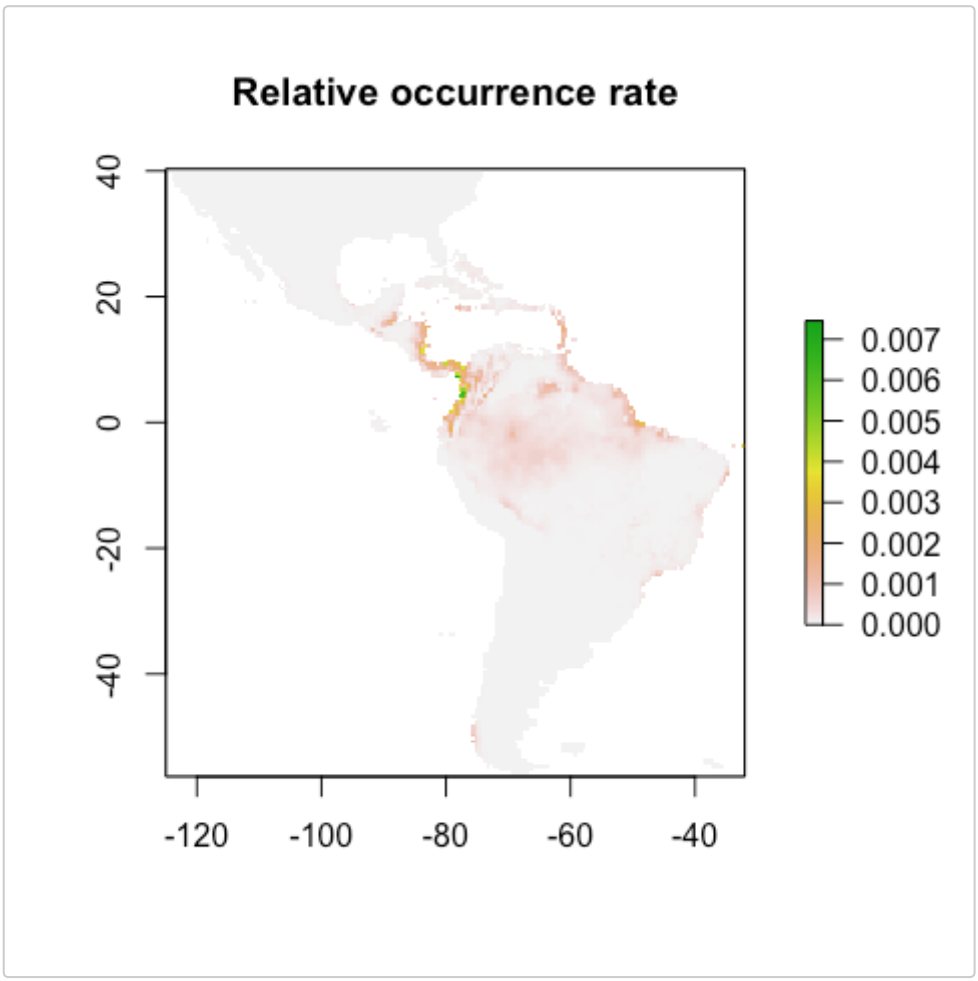
```
eval2@results
#>   settings features rm full.AUC  Mean.AUC     Var.AUC Mean.AUC.DIFF
#> 1      L_1        L  1   0.8627 0.8419877 0.008350469    0.03469096
#> 2     LQ_1       LQ  1   0.8709 0.8490394 0.009538650    0.03564626
#> 3    LQP_1      LQP  1   0.8708 0.8292492 0.015884705    0.05487862
#> 4      L_2        L  2   0.8627 0.8417072 0.008166510    0.03481578
#> 5     LQ_2       LQ  2   0.8705 0.8473323 0.009723896    0.03596288
#> 6    LQP_2      LQP  2   0.8699 0.8288280 0.013631901    0.05424220
#>   Var.AUC.DIFF Mean.OR10    Var.OR10 Mean.ORmin   Var.ORmin     AICc
#> 1   0.01063267 0.1013024 0.004912416 0.04448622 0.004574406 2637.447
#> 2   0.01260206 0.1159707 0.009098218 0.04448622 0.004574406 2587.357
#> 3   0.01968855 0.1436068 0.022727712 0.05482456 0.003907403 2614.433
#> 4   0.01036024 0.1013024 0.004912416 0.02918009 0.001496376 2638.544
#> 5   0.01268704 0.1159707 0.009098218 0.04448622 0.004574406 2589.467
#> 6   0.01731231 0.1644401 0.018800484 0.07013068 0.008437679 2624.870
#>   delta.AICc        w.AIC nparam
#> 1  50.090538 9.845409e-12      7
#> 2   0.000000 7.417470e-01     11
#> 3  27.076708 9.786408e-07     26
#> 4  51.187624 5.688588e-12      7
#> 5   2.110145 2.582520e-01     11
#> 6  37.513592 5.300084e-09     28

eval2@results[which(eval2@results$delta.AICc==0),]
#>   settings features rm full.AUC  Mean.AUC    Var.AUC Mean.AUC.DIFF
#> 2     LQ_1       LQ  1   0.8709 0.8490394 0.00953865    0.03564626
#>   Var.AUC.DIFF Mean.OR10    Var.OR10 Mean.ORmin   Var.ORmin     AICc
#> 2   0.01260206 0.1159707 0.009098218 0.04448622 0.004574406 2587.357
#>   delta.AICc    w.AIC nparam
#> 2          0 0.741747     11
```

Now let's access the RasterStack of the model predictions. Note that these predictions are in the 'raw' output format.

```
eval2@predictions
#> class       : RasterStack
#> dimensions  : 192, 186, 35712, 6  (nrow, ncol, ncell, nlayers)
#> resolution  : 0.5, 0.5  (x, y)
#> extent      : -125, -32, -56, 40  (xmin, xmax, ymin, ymax)
#> coord. ref. : +proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0
#> names       :          L_1,         LQ_1,        LQP_1,          L_2,         LQ_2,        LQP_2
#> min values  : 2.109031e-12, 3.138558e-14, 3.773544e-14, 2.216246e-12, 5.345332e-14, 2.350602e-13
#> max values  :  0.017159045,  0.007454745,  0.007906989,  0.016996421,  0.007288013,  0.008368275
```

Now plot the model with the lowest AICc:

```
plot(eval2@predictions[[which(eval2@results$delta.AICc==0)]], main="Relative occurrence rate")
```



We can also access a list of Maxent model objects, which (as all lists) can be subset with double brackets (e.g. `results@eval2[[1]]`). The Maxent model objects provide access to various elements of the model (including the lambda file). The model objects can also be used for predicting models into other time periods or geographic areas. Note that the html file that is created when Maxent is run is **not** kept.

Let's look at the model object for our "AICc optimal" model:

```
aic.opt <- eval2@models[[which(eval2@results$delta.AICc==0)]]

aic.opt
```

```
#> class    : MaxEnt
#> variables: bio1 bio12 bio16 bio17 bio5 bio6 bio7 bio8 biome
#> output html file no longer exists
```

The “results” slot shows the Maxent model statistics:

```
aic.opt@results
#>                                                                        [,1]
#> X.Training.samples                                                172.0000
#> Regularized.training.gain                                           1.1228
#> Unregularized.training.gain                                         1.2140
#> Iterations                                                        500.0000
#> Training.AUC                                                        0.8709
#> X.Background.points                                            5379.0000
#> bio1.contribution                                                   0.1732
#> bio12.contribution                                                 11.3920
#> bio16.contribution                                                 11.8783
#> bio17.contribution                                                  6.4576
#> bio5.contribution                                                   5.6670
#> bio6.contribution                                                   8.4474
#> bio7.contribution                                                  35.4122
#> bio8.contribution                                                   3.2099
#> biome.contribution                                                 17.3624
#> bio1.permutation.importance                                         0.0000
#> bio12.permutation.importance                                       30.2544
#> bio16.permutation.importance                                        0.0677
#> bio17.permutation.importance                                        4.6323
#> bio5.permutation.importance                                        12.0550
#> bio6.permutation.importance                                         4.1121
#> bio7.permutation.importance                                        29.5807
#> bio8.permutation.importance                                        16.4224
#> biome.permutation.importance                                        2.8753
#> Entropy                                                             7.4686
#> Prevalence..average.of.logistic.output.over.background.sites.       0.1619
#> Fixed.cumulative.value.1.cumulative.threshold                       1.0000
#> Fixed.cumulative.value.1.logistic.threshold                         0.0277
#> Fixed.cumulative.value.1.area                                       0.6577
#> Fixed.cumulative.value.1.training.omission                          0.0058
#> Fixed.cumulative.value.5.cumulative.threshold                       5.0000
#> Fixed.cumulative.value.5.logistic.threshold                         0.1083
#> Fixed.cumulative.value.5.area                                       0.4458
#> Fixed.cumulative.value.5.training.omission                          0.0814
#> Fixed.cumulative.value.10.cumulative.threshold                     10.0000
#> Fixed.cumulative.value.10.logistic.threshold                        0.1881
#> Fixed.cumulative.value.10.area                                      0.3529
#> Fixed.cumulative.value.10.training.omission                         0.1221
#> Minimum.training.presence.cumulative.threshold                      0.9069
#> Minimum.training.presence.logistic.threshold                        0.0259
#> Minimum.training.presence.area                                      0.6689
#> Minimum.training.presence.training.omission                         0.0000
#> X10.percentile.training.presence.cumulative.threshold               7.4085
```

```
#> X10.percentile.training.presence.logistic.threshold                                0.1532
#> X10.percentile.training.presence.area                                              0.3939
#> X10.percentile.training.presence.training.omission                                 0.0988
#> Equal.training.sensitivity.and.specificity.cumulative.threshold                   21.7045
#> Equal.training.sensitivity.and.specificity.logistic.threshold                      0.2875
#> Equal.training.sensitivity.and.specificity.area                                    0.2296
#> Equal.training.sensitivity.and.specificity.training.omission                       0.2267
#> Maximum.training.sensitivity.plus.specificity.cumulative.threshold                 25.5091
#> Maximum.training.sensitivity.plus.specificity.logistic.threshold                   0.3149
#> Maximum.training.sensitivity.plus.specificity.area                                 0.2008
#> Maximum.training.sensitivity.plus.specificity.training.omission                    0.2442
#> Balance.training.omission..predicted.area.and.threshold.value.cumulative.threshold 2.4120
#> Balance.training.omission..predicted.area.and.threshold.value.logistic.threshold   0.0529
#> Balance.training.omission..predicted.area.and.threshold.value.area                 0.5468
#> Balance.training.omission..predicted.area.and.threshold.value.training.omission    0.0058
#> Equate.entropy.of.thresholded.and.original.distributions.cumulative.threshold      12.0558
#> Equate.entropy.of.thresholded.and.original.distributions.logistic.threshold        0.2076
#> Equate.entropy.of.thresholded.and.original.distributions.area                      0.3257
#> Equate.entropy.of.thresholded.and.original.distributions.training.omission         0.1395
```

You can use the `var.importance` function to get a data.frame of two variable importance metrics: percent contribution and permutation importance. See the function help file for more information on these metrics.

```
var.importance(aic.opt)
#>   variable percent.contribution permutation.importance
#> 1     bio1               0.1732                 0.0000
#> 2    bio12              11.3920                30.2544
#> 3    bio16              11.8783                 0.0677
#> 4    bio17               6.4576                 4.6323
#> 5     bio5               5.6670                12.0550
#> 6     bio6               8.4474                 4.1121
#> 7     bio7              35.4122                29.5807
#> 8     bio8               3.2099                16.4224
#> 9    biome              17.3624                 2.8753
```

The “lambdas” slot shows which variables were included in the model. After the variable name, the next number is the variable coefficient, then the min and max of that variable for the inut data. If the coefficient is 0, that variable was not included in the model. You will likely find the syntax to be cryptic and the information is not stored in a very user-friendly way. Fortunately, John Baumgartner has developed a useful function to parse this file into a more user-friendly data.frame. See the `parse_lambdas.R` function in his `rmaxent` package for more details.

```
aic.opt@lambdas
#>  [1] "bio1, 0.0, -1.0, 289.0"
#>  [2] "bio12, 17.407991363812382, 0.0, 7682.0"
#>  [3] "bio16, 0.0, 0.0, 2458.0"
#>  [4] "bio17, -11.41332780283899, 0.0, 1496.0"
#>  [5] "bio5, 0.0, 97.0, 363.0"
#>  [6] "bio6, 0.0, -128.0, 236.0"
#>  [7] "bio7, -12.31658264255844, 64.0, 325.0"
```

```
#>  [8] "bio8, 13.578185430909318, 27.0, 290.0"
#>  [9] "biome, -5.1094467716509495, 1.0, 14.0"
#> [10] "bio12^2, -4.807539155874979, 0.0, 5.9013124E7"
#> [11] "bio16^2, -3.8903550909683493, 0.0, 6041764.0"
#> [12] "bio17^2, 6.66326827008995, 0.0, 2238016.0"
#> [13] "bio5^2, -7.71897483159865, 9409.0, 131769.0"
#> [14] "bio6^2, -4.465888692592631, 0.0, 55696.0"
#> [15] "biome^2, 3.823435363203316, 1.0, 196.0"
#> [16] "linearPredictorNormalizer, 5.989266795505559"
#> [17] "densityNormalizer, 134.1427416404159"
#> [18] "numBackgroundPoints, 5379"
#> [19] "entropy, 7.468605185099883"
```

Finally, the ENMevaluate object also remembers which occurrence partitioning method you used:

```
eval2@partition.method
#> [1] "checkerboard2"
```

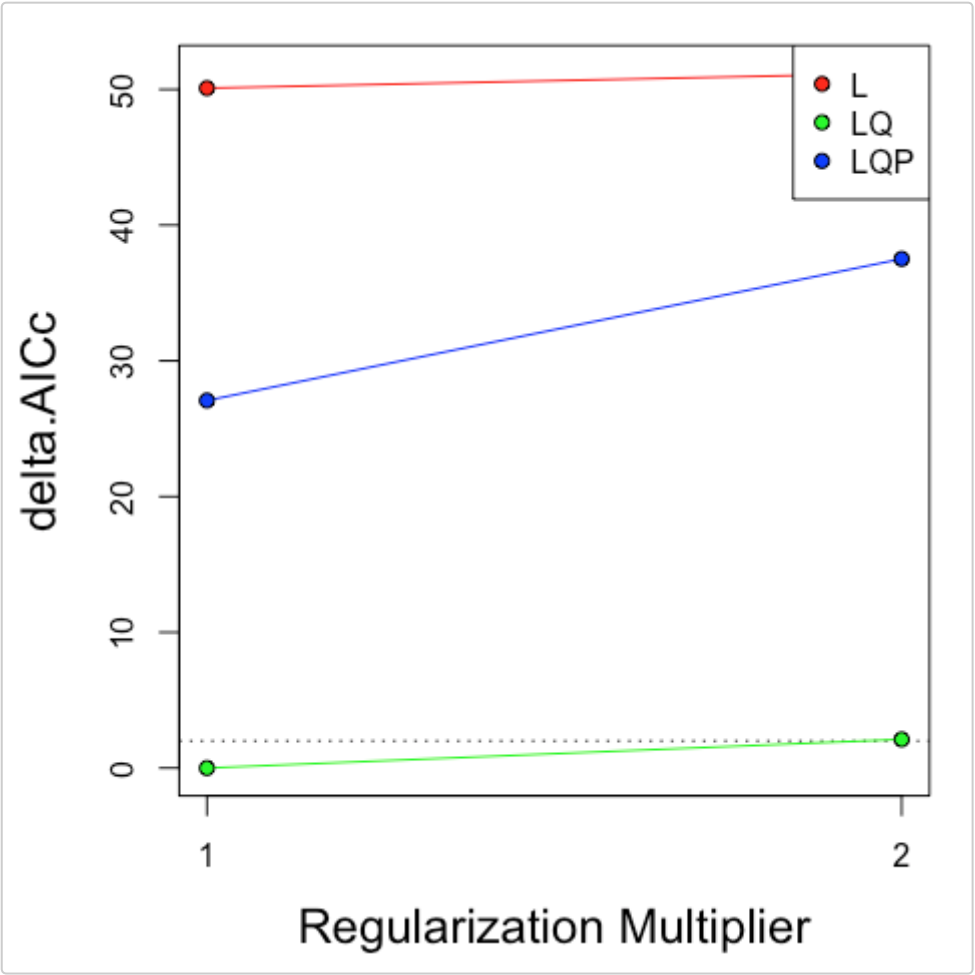
Plotting results

Plotting options in R are extremely flexible and here we demonstrate some key tools to explore the results of an ENMevaluate object graphically.

- Plotting model predictions
- Plotting response curves

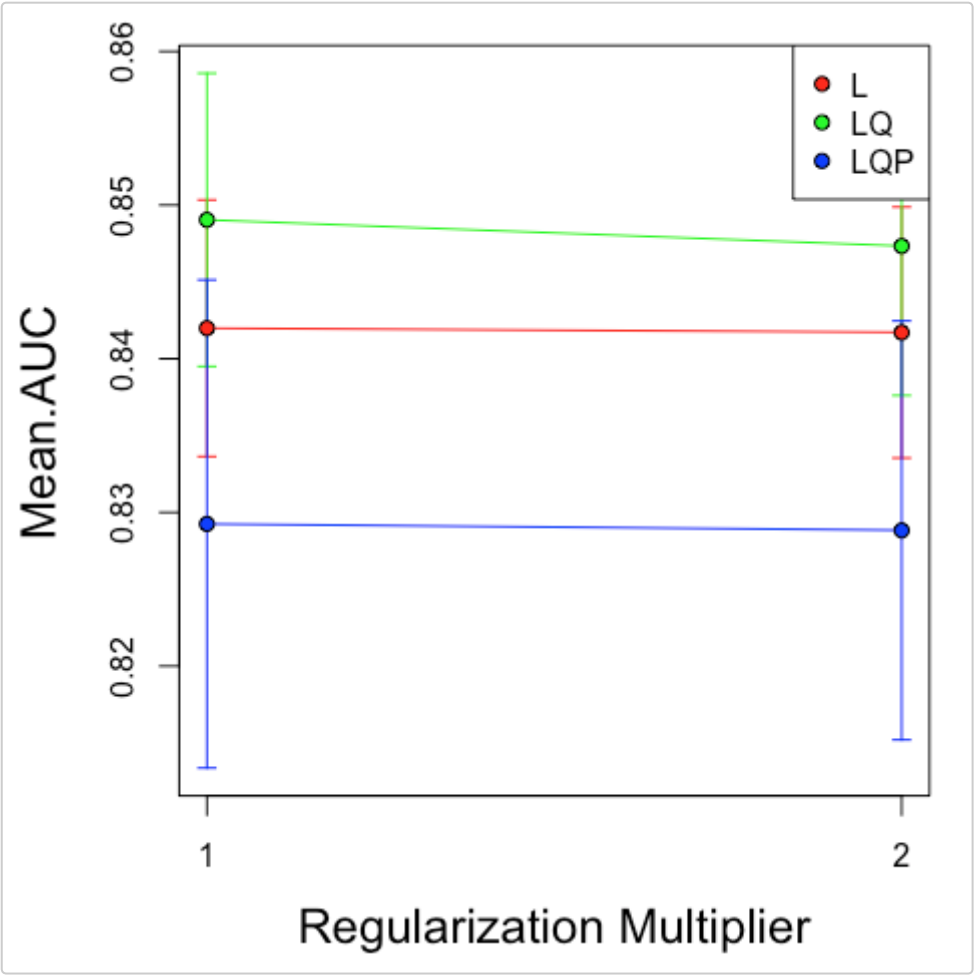
ENMeval has a built-in plotting function (`eval.plot`) to visualize the results of different models. It requires the results table of the ENMevaluation object. By default, it plots delta.AICc values.

```
eval.plot(eval2@results)
```



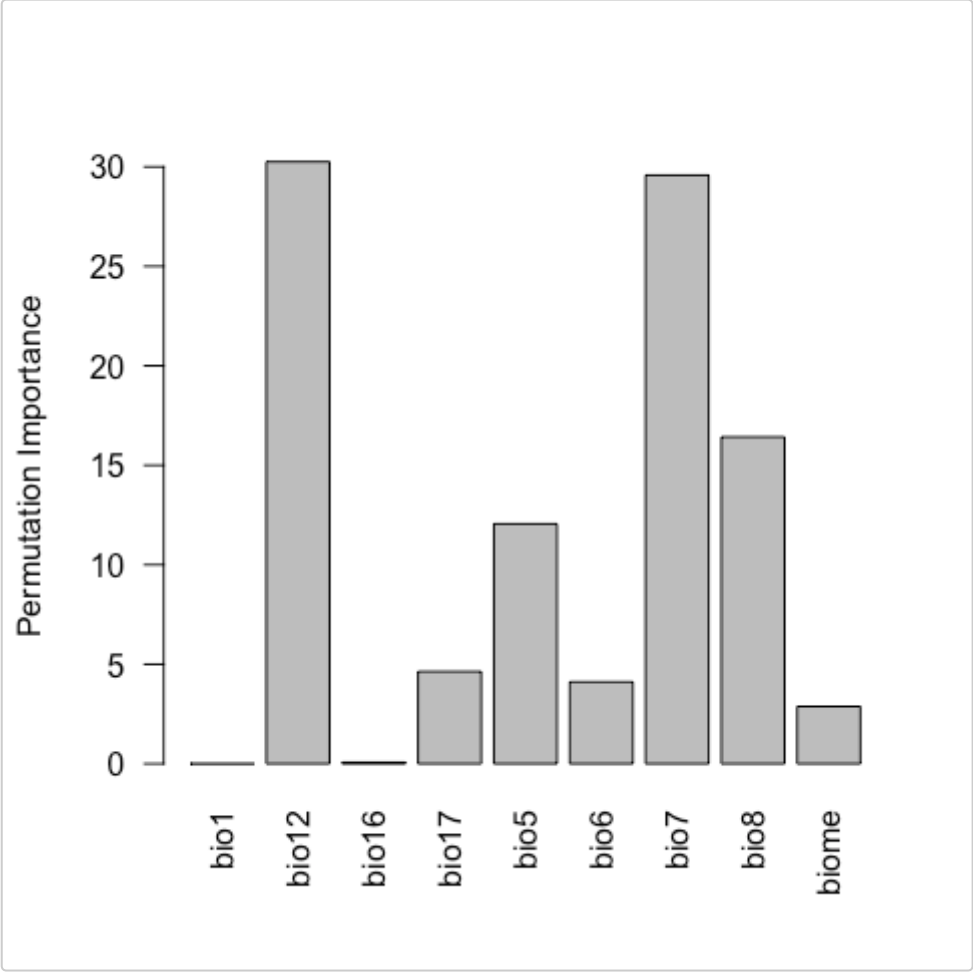
You can choose which evaluation metric to plot, and you can also include error bars, if relevant.

```
eval.plot(eval2@results, 'Mean.AUC', var='Var.AUC')
```



You can also plot the permutation importance or percent contribution.

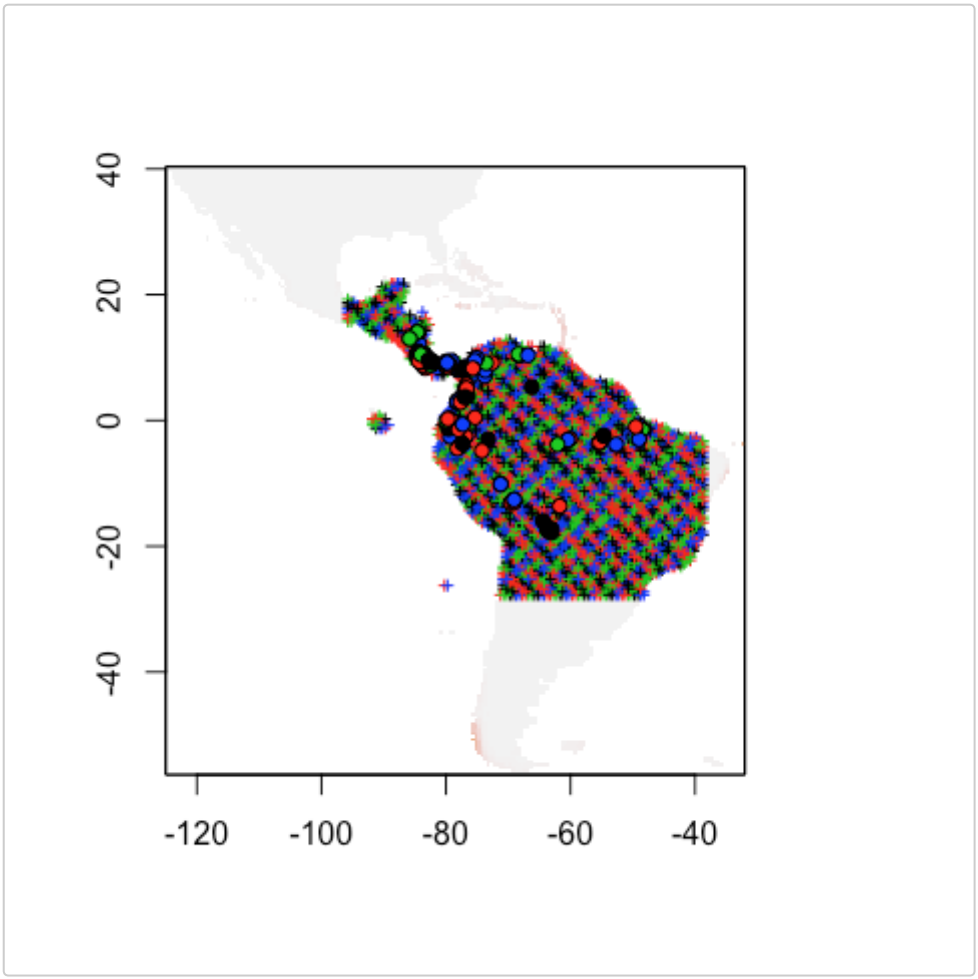
```
df <- var.importance(aic.opt)
barplot(df$permutation.importance, names.arg=df$variable, las=2, ylab="Permutation Importance")
```



Plotting model predictions

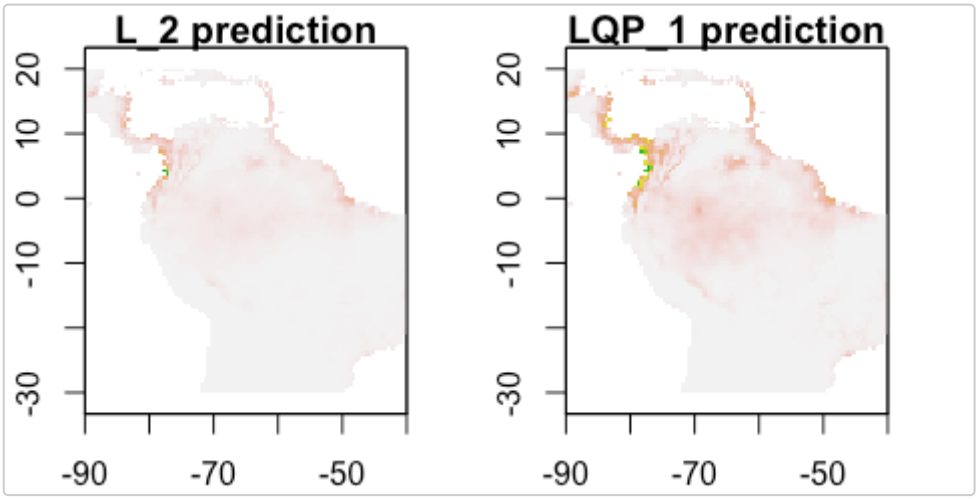
If you generated raster predictions of the models (i.e., `rasterpreds=T`), you can easily plot them. For example, let's look at the first two models included in our analysis - remember that the output values are in Maxent's 'raw' units.

```
plot(eval2@predictions[[1]], legend=F)

# Now add the occurrence and background points, colored by evaluation bins:
points(eval2@bg.pts, pch=3, col=eval2@bg.grp, cex=0.5)
points(eval2@occ.pts, pch=21, bg=eval2@occ.grp)
```



Let's see how model complexity changes the predictions in our example. We'll compare the model predictions of the model with only linear feature classes and with the highest regularization multiplier value we used (i.e., fc='L', RM=2) versus the model with all feature class combination and the lowest regularization multiplier value we used (i.e., fc='LQP', RM=1).

```
# bisect the plotting area to make two columns
par(mfrow=c(1,2), mar=c(2,2,1,0))

plot(eval2@predictions[['L_2']], ylim=c(-30,20), xlim=c(-90,-40), legend=F, main='L_2 prediction')

plot(eval2@predictions[['LQP_1']], ylim=c(-30,20), xlim=c(-90,-40), legend=F, main='LQP_1
prediction')
```



Plotting response curves

We can also plot the response curves of our model to see how different input variables influence our model predictions. (Note that, as with the `dismo::maxent` function, using this function requires that the maxent.jar file be installed in the `dismo` package java folder).

```
response(eval2@models[[1]])
```

Downstream Analyses (*under construction*)

Below is a running list of other things we plan to add to this vignette. Feel free to let us know if there are particular things you would like to see added.

- Extracting model results from object (various thresholds)
- Use model object to make a new prediction (e.g., if you want a logistic prediction)
- Make a projection to a new extent
- Do MESS map (Use `mess()` in the `dismo` package)

Resources (*under construction*)

Web Resources

- Hijmans, R. and Elith, J. (2016) Species distribution modeling with R. dismo vignette.
- Phillips, S. J. (2006) Phillips, S. (2006) A brief tutorial on Maxent. AT&T Research. Available at: http://www.cs.princeton.edu/~schapire/maxent/tutorial/tutorial.doc
- Yoder, J. (2013) Species distribution models in R. The Molecular Ecologist.
- Maxent Google Group

General Guides

- 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.
- Peterson, A.T., Soberón, J., Pearson, R.G., Anderson, R.P., Martínez-Meyer, E., Nakamura, M., and Araújo, M.B. (2011) Ecological Niches and Geographic Distributions. Monographs in Population Biology, 49. Princeton University Press.
- Renner, I.W., Elith, J., Baddeley, A., Fithian, W., Hastie, T., Phillips, S.J., . . . Warton, D.I. (2015) Point process models for presence-only analysis. Methods in Ecology and Evolution 6, 366-379.

Model Evaluation

- Aiello-Lammens, M.E., Boria, R.A., Radosavljevic, A., Vilela, B., and Anderson, R.P. (2015) spThin: an R package for spatial thinning of species occurrence records for use in ecological niche models. Ecography 38, 541-545.
- Fielding, A.H. and Bell, J.F. (1997) A review of methods for the assessment of prediction errors in conservation presence-absence models. Environmental Conservation 24, 38-49.
- Hijmans, R.J. (2012) Cross-validation of species distribution models: removing spatial sorting bias and calibration with a null model. Ecology 93, 679-688.
- 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 Ecol Evol, 5: 1198–1205.
- Radosavljevic, A. and Anderson, R.P. (2014) Making better Maxent models of species distributions: complexity, overfitting and evaluation. Journal of Biogeography 41, 629-643.
- Shcheglovitova, M. and Anderson, R.P. (2013) Estimating optimal complexity for ecological niche models: A jackknife approach for species with small sample sizes. Ecol. Model. 269, 9-17.
- Veloz, S.D. (2009) Spatially autocorrelated sampling falsely inflates measures of accuracy for presence-only niche models. Journal of Biogeography 36, 2290-2299.
- Wenger, S.J. and Olden, J.D. (2012) Assessing transferability of ecological models: an underappreciated aspect of statistical validation. Methods in Ecology and Evolution 3, 260-267.

Some Empirical Examples

- 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.