Exercise 10: Model Evaluation - Metrics

Adam B. Smith, Danielle Svehla, & Camilo Sanín

July 31, 2017

Thus far we've been using the "non-parametric eyeball" test to assess how well our models were performing. This may seem ill-advised given how statistically quantitative ecology has become. **However, I firmly believe visual inspection of output maps is one of the most important kinds of model evaluation.** In many cases I've found model evaluation metrics are very similar between models that produce very different maps. Likewise, model metrics can only tell you how good the model is in locations where you have test sites. So don't eschew your eyeball test!

We'll be looking at two kinds of evaluation metrics:

1. **Threshold-dependent metrics (TDMs)** which use a cut-off value to divide predictions into presences (1s) and absence (0s), and
2. **Threshold-independent metrics (TIMs)** which generally evaluate the ability of the raw predictions (ranging between 0 to 1) to reflect the propensity of the species to be present or absent at sites.

# Threshold-dependent metrics of model performance

TDMs require a rule to divide predictions into presences (1s) and absences (0s). *A priori* you might assume that a value of 0.5 is best, but recall that the prediction from a model trained using background sites at best can only be guaranteed to correlate with the true probability of presence. This means a predicted value of 0.5 does not necessarily correspond to a probability of presence of 0.5. There are many ways to choose thresholds.

Thresholds are often evaluated according to their ability to produce good presence and absence predictions. **Sensitivity** is the proportion of test presences that are correctly predicted to be present. **Specificity** is the proportion of test absences (or background sites) correctly predicted to be absences. Note that specificity is fairly meaningless if background sites are used in place of true absences. We'll be exploring how well three of the most commonly-used thresholds perform in terms of their sensitivity (and specificity).

First, though, we'll extract predictions at presences and (random) background sites to use in calculating thresholds and model performance. Note we're using the "new baseline" map (target background, smooth, tuned) we created in the last tutorial.

predPres <- extract(tunedMap, cbind(records$longitude, records$latitude))  
predBg <- extract(tunedMap, cbind(randomBg$longitude, randomBg$latitude))

## Minimum training presence threshold

Description: The smallest predicted value across all of the training presences.

Notes: This threshold guarantees every training presence is predicted to be "present". However, this threshold is often very optimistic and over-predicts the species' range.

Example:

minTrainPresThreshold <- min(predPres)  
minTrainPresThreshold

## [1] 0.07852979

# sensitivity (true positive rate)  
sum(predPres >= minTrainPresThreshold) / length(predPres)

## [1] 1

# specificity (true negative rate)  
sum(predBg < minTrainPresThreshold) / length(predBg)

## [1] 0.6782035

## Threshold that equalizes sensitivity and specificity (or at least minimizes their difference)

Description: This threshold occurs when the rate of correct presence predictions is as close as possible to the rate of correct absence predictions.

Notes: As noted, "correct absence" predictions are dubious when background sites are used instead of true absences.

Example:

# calculate an "evaluation" object  
eval <- evaluate(p=as.vector(predPres), a=as.vector(predBg), tr=seq(0, 1, by=0.01))  
  
# see some evaluation statistics  
eval

## class : ModelEvaluation   
## n presences : 29   
## n absences : 9997   
## AUC : 0.8960912   
## cor : 0.09754886   
## max TPR+TNR at : 0.31

# see even more statistics  
# t = threshold values (from 0 to 1)  
# TPR = True Presence Rate = sensitivity at each threshold value  
# TNR = True Negative Rate = specificity at each threshold value  
str(eval)

## Formal class 'ModelEvaluation' [package "dismo"] with 22 slots  
## ..@ presence : num [1:29] 0.372 0.646 0.979 0.999 0.997 ...  
## ..@ absence : num [1:9997] 1.66e-07 1.59e-05 6.93e-05 8.52e-02 1.51e-01 ...  
## ..@ np : int 29  
## ..@ na : int 9997  
## ..@ auc : num 0.896  
## ..@ pauc : num(0)   
## ..@ cor : Named num 0.0975  
## .. ..- attr(\*, "names")= chr "cor"  
## ..@ pcor : num 1.25e-22  
## ..@ t : num [1:101] 0 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 ...  
## ..@ confusion : int [1:101, 1:4] 29 29 29 29 29 29 29 29 28 28 ...  
## .. ..- attr(\*, "dimnames")=List of 2  
## .. .. ..$ : NULL  
## .. .. ..$ : chr [1:4] "tp" "fp" "fn" "tn"  
## ..@ prevalence: num [1:101] 0.00289 0.00289 0.00289 0.00289 0.00289 ...  
## ..@ ODP : num [1:101] 0.997 0.997 0.997 0.997 0.997 ...  
## ..@ CCR : num [1:101] 0.00289 0.44883 0.51656 0.56723 0.60273 ...  
## ..@ TPR : num [1:101] 1 1 1 1 1 ...  
## ..@ TNR : num [1:101] 0 0.447 0.515 0.566 0.602 ...  
## ..@ FPR : num [1:101] 1 0.553 0.485 0.434 0.398 ...  
## ..@ FNR : num [1:101] 0 0 0 0 0 ...  
## ..@ PPP : num [1:101] 0.00289 0.00522 0.00595 0.00664 0.00723 ...  
## ..@ NPP : num [1:101] NaN 1 1 1 1 ...  
## ..@ MCR : num [1:101] 0.997 0.551 0.483 0.433 0.397 ...  
## ..@ OR : num [1:101] NaN Inf Inf Inf Inf ...  
## ..@ kappa : num [1:101] 0 0.00466 0.00611 0.00749 0.00866 ...

equalSeSpThreshold <- eval@t[which.min(abs(eval@TPR - eval@TNR))]  
equalSeSpThreshold

## [1] 0.39

# sensitivity (true positive rate)  
sum(predPres >= equalSeSpThreshold) / length(predPres)

## [1] 0.8275862

# specificity (true negative rate)  
sum(predBg < equalSeSpThreshold) / length(predBg)

## [1] 0.8275483

## Threshold that maximizes the sum of sensitivity and specificity

Description: This threshold maximizes the correct prediction rate regardless of whether correct predictions are mainly from presences or absences/background sites

Notes: Assuming test presences and background sites are sampled from the landscape in a representative manner (i.e., unbiased), the threshold value that maximizes the sum of sensitivity and specificity will be the same regardless of whether true absences or background sites are use. Pretty cool! See [Liu, C., White, M., and Newell, G. 2013. Selecting thresholds for the prediction of species occurrence with presence-only data. Journal of Biogeography 40:788-789.](http://dx.doi.org/10.1111/jbi.12058)

Example:

maxSeSpThreshold <- eval@t[which.max(eval@TPR + eval@TNR)]  
maxSeSpThreshold

## [1] 0.31

# sensitivity (true positive rate)  
sum(predPres >= maxSeSpThreshold) / length(predPres)

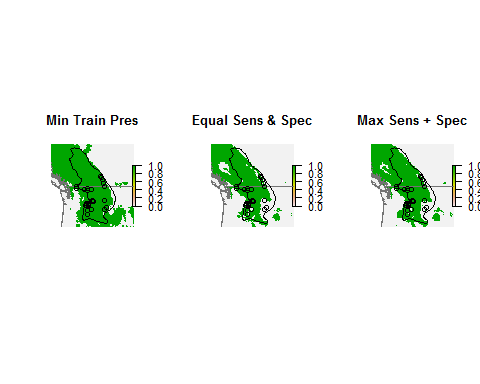
## [1] 0.9310345

# specificity (true negative rate)  
sum(predBg < maxSeSpThreshold) / length(predBg)

## [1] 0.8088427

Now, let's look at what the maps look like when they're thresholded.

par(mfrow=c(1, 3), pty='s')  
  
plot(rangeMap, main='Min Train Pres')  
thresholdedMap <- tunedMap >= minTrainPresThreshold  
plot(thresholdedMap, add=TRUE)  
sp::plot(countries, add=TRUE, border='gray45')  
plot(rangeMap, add=TRUE)  
points(records$longitude, records$latitude)  
  
plot(rangeMap, main='Equal Sens & Spec')  
thresholdedMap <- tunedMap >= equalSeSpThreshold  
plot(thresholdedMap, add=TRUE)  
sp::plot(countries, add=TRUE, border='gray45')  
plot(rangeMap, add=TRUE)  
points(records$longitude, records$latitude)  
  
plot(rangeMap, main='Max Sens + Spec')  
thresholdedMap <- tunedMap >= maxSeSpThreshold  
plot(thresholdedMap, add=TRUE)  
sp::plot(countries, add=TRUE, border='gray45')  
plot(rangeMap, add=TRUE)  
points(records$longitude, records$latitude)



## Reflection

Which threshold seems best? This is just a single example, but based on what you see, do you think thresholding in general is advised?

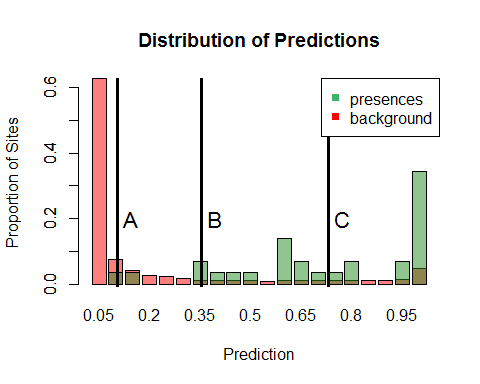
# Threshold-independent metrics (TIMs)

TIMs compare raw model predictions to determine how good a model performs. In general they are probably better indicators of true performance since TDMs erase much of the information output by the model when the threshold is applied. However, there is no "perfect" TIM (or TDM), and it is probable that most or even all can be severely skewed if your test sites are biased. This is an ongoing area of research, so we'll just explore two TIMs even though they may not be perfect.

## Area under the receiver-operator characteristic curve (AUC)

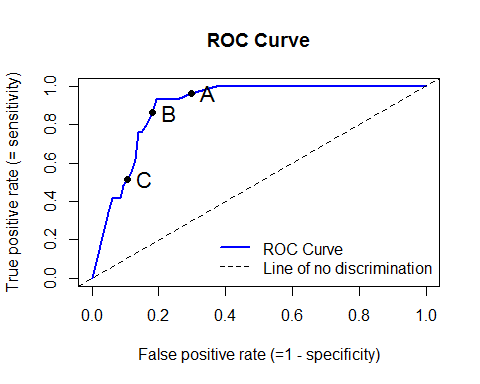
AUC was developed in World War II to measure the effectiveness of radar in detecting enemy planes. It is essentially a series of TDMs summed across multiple thresholds. We'll explain AUC in terms of its calculation with presences and background sites (versus absences), but it can also be calculated with presences and absences. To see how it works, consider the frequency of predictions at presence and background sites for our species:

# tally frequency of predictions at presence and background sites in bins  
presDistrib <- hist(predPres, plot=FALSE, breaks=seq(0, 1, length.out=21))$counts  
bgDistrib <- hist(predBg, plot=FALSE, breaks=seq(0, 1, length.out=21))$counts  
  
# convert to proportion of sites  
presDistrib <- presDistrib / sum(presDistrib)  
bgDistrib <- bgDistrib / sum(bgDistrib)  
  
# plot distribution of predictions at presences and background sites  
labels <- round(seq(0, 1, length.out=21), 2)  
barplot(bgDistrib,  
 col=alpha('red', 0.5),  
 names.arg=labels[2:21],  
 ylab='Proportion of Sites',  
 xlab='Prediction',  
 main='Distribution of Predictions'  
)  
barplot(presDistrib, col=alpha('forestgreen', 0.5), add=TRUE)  
  
# vertical bars for three example thresholds  
abline(v=c(2, 8, 17), lwd=3)  
text(x=c(2, 8, 17) + 1, y=c(0.2, 0.2, 0.2), labels=LETTERS[1:3], cex=1.4)  
  
legend('topright', legend=c('presences', 'background'),  
 col=c('mediumseagreen', 'red'), pch=15)



You can see that the predicted values for the presences and background sites overlap to some extent. Note that any threshold value (we've drawn three of them here at A, B, and C) would cause the presences to the left of a line to be incorrectly designated as absences and some background sites to the right of the line to be designated as presences. Thus we can calculate the sensitivity (true positive rate) and false positive rate (= 1 - specificity) for a series of thresholds. We can then plot these two values (sensitivity and 1 - specificity) to get an indication of how well the model predicts across multiple thresholds.

# will store sensitivity and false positive rate  
sensitivity <- FPR <- numeric()  
  
# use 21 threshold values from 0 to 1  
for (threshold in seq(0, 1, length.out=21)) {  
 thisSens <- sum(predPres >= threshold) / length(predPres)  
 thisFPR <- sum(predBg >= threshold) / length(predBg)  
 sensitivity <- c(sensitivity, thisSens)  
 FPR <- c(FPR, thisFPR)  
}  
  
# plot ROC curve  
plot(FPR,  
 sensitivity,  
 type='l',  
 xlab='False positive rate (=1 - specificity)',  
 ylab='True positive rate (= sensitivity)',  
 main='ROC Curve',  
 lwd=2,  
 col='blue'  
)  
  
abline(a=0, b=1, lty='dashed')  
  
legend('bottomright',  
 legend=c('ROC Curve', 'Line of no discrimination'),  
 lwd=c(2, 1),  
 lty=c('solid', 'dashed'),  
 col=c('blue', 'black'),  
 bty='n'  
)  
  
# add location three example thresholds from previous graph  
points(c(FPR[which.min(abs(seq(0, 1, length.out=21) - 0.12))],  
 FPR[which.min(abs(seq(0, 1, length.out=21) - 0.35))],  
 FPR[which.min(abs(seq(0, 1, length.out=21) - 0.70))]),  
 c(sensitivity[which.min(abs(seq(0, 1, length.out=21) - 0.12))],  
 sensitivity[which.min(abs(seq(0, 1, length.out=21) - 0.35))],  
 sensitivity[which.min(abs(seq(0, 1, length.out=21) - 0.70))]), pch=16)  
  
text(c(FPR[which.min(abs(seq(0, 1, length.out=21) - 0.12))],  
 FPR[which.min(abs(seq(0, 1, length.out=21) - 0.35))],  
 FPR[which.min(abs(seq(0, 1, length.out=21) - 0.70))]) + 0.05,  
 c(sensitivity[which.min(abs(seq(0, 1, length.out=21) - 0.12))],  
 sensitivity[which.min(abs(seq(0, 1, length.out=21) - 0.35))],  
 sensitivity[which.min(abs(seq(0, 1, length.out=21) - 0.70))]),  
 labels=LETTERS[1:3], cex=1.4)



The area under this curve (AUC) is used as a metric of model performance. This area has a range from 0 to 1:

0 to 0.5: Model performs *worse* than random  
0.5: Model performs no better or worse than random  
0.5 to 1: Model performs *better* than random

The AUC also has a probabilistic interpretation: it represents the probability that a randomly drawn presence site has a higher predicted value than a randomly drawn absence/background site. It is thus inherently a rank-based statistic because it doesn't care how much higher the prediction of a presence is over an absence. In this respect AUC is a **discriminative measure of model performance** because it tells you how well the model differentiates between presences and absences.

Downside: Though easily the most commonly-reported model performance metric, when calculated using background sites in place of true absences it can be very misleading. For good models it has a maximum value of 1 - *a*/2, where *a* is the proportion of the landscape occupied by the species. Typically you will not know this, so an AUC value of 0.6, which is just above random, may or may not be "high" depending on *a*. This also means you can't compare AUC calculated with background sites across species since *a* will vary by species.

More perniciously, bad models can cause AUC to surpass the threshold 1 - *a*/2. This would seem to be a good thing--higher values of AUC are better. But AUC calculated with background sites is *apparent* AUC... the AUC you're really interested in AUC calculated with presences and true absences (which are rarely available). For bad models as apparent AUC increases *real* AUC declines... so your model can actually become worse as it appears to become better! See [Smith, A.B. 2013. On evaluating species distribution models with random background sites in place of absences when test presences disproportionately sample suitable habitat. Diversity and Distributions 19:867-872.](http://dx.doi.org/10.1111/ddi.12031)

Note, though that AUC calculated with presences and true absences should be OK!

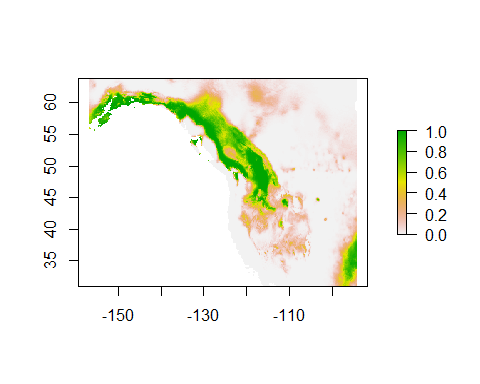
## The Continuous Boyce Index (CBI)

A lesser-used but likely more sensitive measure of model performance is the Continuous Boyce Index, which is a variant of the (non-continuous) Boyce Index. CBI is a measure of **calibration performance** which indicates the ability of a model to predict the actual probability of occurrence. This is different from discriminative measures like AUC because the latter simply indicate how well models differentiate between presences and absences regardless of the degree of difference between them.

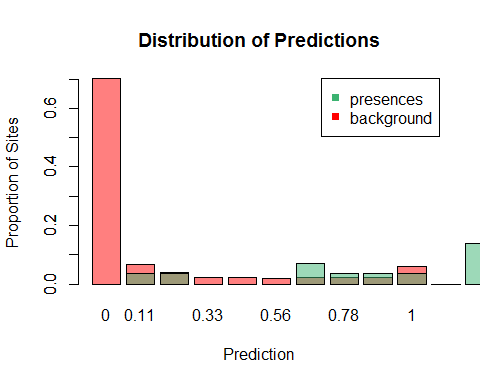
Original Boyce Index: [Boyce, M.S., Vernier, P.R., Nielsen, S.E., and Schmiegelow, F.K.A. 2002. Evaluating resource selection functions. Ecological Modeling 157:281-300.](http://dx.doi.org/10.1016/S0304-3800(02)00200-4)

Continuous Boyce Index: [Hirzel, A.H., Le Lay, G., Helfer, V., Randin, C., and Guisan, A. 2006. Evaluating the ability of habitat suitability models to predict species presences. Ecological Modeling 199:142-152.](http://dx.doi.org/10.1016/j.ecolmodel.2006.05.017)

We'll explain CBI using somewhat inaccurate language then tell you what we really meant when you understand how it is calculated. CBI first examines the distribution of predictions taken from across the landscape. Ergo, it takes this:



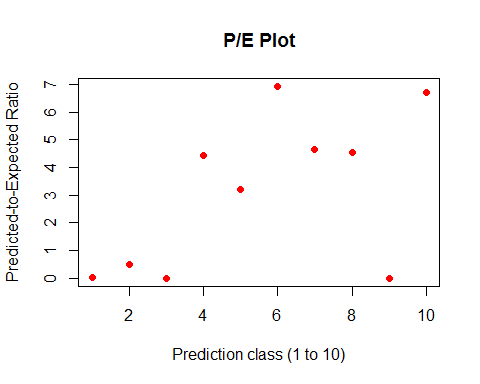
...and calculates the histogram of predictions across the landscape (this is the same graph as above):



Here we're using the distribution of predictions across background sites as an estimate of distributor of predictions across all cells in the landscape. CBI then calculates the ratio of observed predictions at test presence sites (=P) to the expected distribution across the landscape (=E). CBI uses this ratio as a metric of accuracy. Consider a very simple case in which the model predicted only two regions of suitability that each covered 50% of the landscape. If the model is no better than random we'd expect half of the test presences to be in either section. If it is better than random then the portion with a higher prediction should have proportionately more test presences in it.

CBI breaks the predictions across the landscape into bins, calculates the P/E ratio for each, then calculates the rank correlation coefficient between P/E and the rank of the bin (from the first bin with the lowest prediction class to the highest bin with the highest prediction class).

# get proportion of presences or background sites in each bin  
presDistrib <- hist(predPres, plot=FALSE, breaks=seq(0, 1, length.out=11))$counts  
bgDistrib <- hist(predBg, plot=FALSE, breaks=seq(0, 1, length.out=11))$counts  
  
# convert to proportion of sites  
presDistrib <- presDistrib / sum(presDistrib)  
bgDistrib <- bgDistrib / sum(bgDistrib)  
  
# P/E plot  
pe <- presDistrib / bgDistrib  
  
plot(pe,  
 xlab='Prediction class (1 to 10)',  
 ylab='Predicted-to-Expected Ratio', main='P/E Plot',  
 col='red', pch=16  
)



The incorrect part of what we've said so far is that CBI really uses partially overlapping bins for its calculation. Its' just easier to explain it in terms of non-overlapping bins. Generally 10 bins are used.

# from the enmSdm package  
cbi <- contBoyce(pres=predPres, bg=predBg)

cbi

## [1] 0.7575758

## Underview

So which measure of model performance should you use? In general it's a good idea to use several since they each reflect a different kind of performance.