Exercise 11: Model Evaluation - Cross-Validation

Adam B. Smith, Danielle Svehla, & Camilo Sanín

July 6, 2018

In the previous exercise we calculated model performance metrics using the same presences and background sites used to train the model. This is called **re-substitution** because it uses the same data to train and test the model. Re-substitution yields overly-optimistic measures of performance--after all, a model should be able to recreate the data on which it was trained. A more reliable measure of performance uses separate test data in a procedure called **cross-validation** which splits the data into sets and uses each to train and test the model.

Here you will learn how to:  
  
1. Evaluate models using **random cross-validation**  
2. Evaluate models using **geographic cross-validation** which is generally more robust than random cross-validation

# Random cross-validation

Under random cross-validation, presences are broken into mutually exclusive groups. Each group usually constitutes 20% or so of the data set (in which case you'd have 5 mutually exclusive groups). These groups are called **k-folds**. You then train a model using all presences except one of the groups, then test the model against this group using some performance measures. Model performance is averaged across groups and presumably better reflects a model's ability to predict to new data. Let's illustrate how to do this with an example using our "tuned" model.

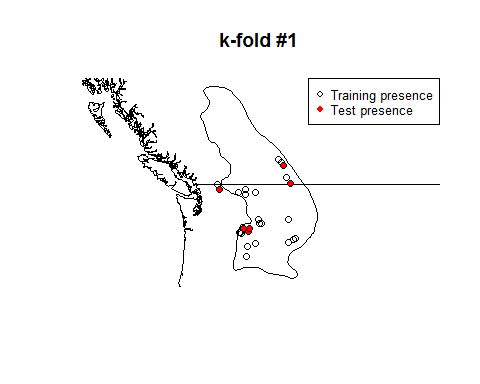
# calculate k-folds for presences and background sites  
kPres <- kfold(x=records, k=5)  
kBg <- kfold(x=targetBg, k=5)  
  
head(kPres)

## [1] 4 1 4 2 2 3

head(kBg)

## [1] 5 4 4 2 4 4

# map  
plot(rangeMap, main='k-fold #1')  
sp::plot(countries, add=TRUE)  
points(records$longitude, records$latitude)  
points(records$longitude[kPres==1],  
 records$latitude[kPres==1],  
 bg='red',  
 pch=21  
)  
  
legend('topright',  
 legend=c('Training presence', 'Test presence'),  
 pch=c(1, 16),  
 col=c('black', 'red'),  
 bg='white',  
 cex=0.8  
)



This illustrates k-fold #1. The model will be trained on the black points and tested on the red. Note your map may look different from mind because folds are chosen at random.

# create output directory for model object and rasters  
dir.create('./Models/Model 09 Model Evaluation - Random K-Folds', recursive=TRUE, showWarnings=FALSE)  
  
# for storing AUC and CBI  
aucRandom <- cbiRandom <- rep(NA, 5)  
  
# cycle through each k-fold  
for (i in 1:5) {  
  
 say('K-fold ', i, ':', post=0)  
   
 # make training data frame with predictors and vector of 1/0 for  
 # presence/background... using only points not in this k-fold  
 envData <- rbind(  
 records[kPres!=i, predictors],  
 targetBg[kBg!=i, predictors]  
 )  
  
 presBg <- c(rep(1, sum(kPres!=i)), rep(0, sum(kBg!=i)))  
   
 trainData <- cbind(presBg, envData)  
   
 # tuned model: just using one regularization value for expdiancy!  
 model <- trainMaxNet(  
 data=trainData,  
 regMult=1,  
 classes='lqp',  
 verbose=FALSE  
 )  
  
 model <- model$model  
  
 # save  
 save(model,  
 file=paste0('./Models/Model 09 Model Evaluation - Random K-Folds/Model ', i, '.Rdata'),  
 compress=TRUE)  
  
 # predict to presences and background sites  
 predPres <- raster::predict(model, newdata=records[kPres==i, ], type='cloglog')  
 predBg <- raster::predict(model, newdata=randomBg[kBg==i, ], type='cloglog')  
   
 # evaluate and remember result  
 thisEval <- evaluate(p=as.vector(predPres), a=as.vector(predBg))  
   
 thisAuc <- thisEval@auc  
 thisCbi <- contBoyce(pres=predPres, bg=predBg)  
   
 say(': AUC = ', round(thisAuc, 2), ' | CBI = ', round(thisCbi, 2))  
  
 aucRandom[i] <- thisAuc  
 cbiRandom[i] <- thisCbi  
  
}

## K-fold 1: : AUC = 0.88 | CBI = 0.61   
## K-fold 2: : AUC = 0.84 | CBI = 0.64   
## K-fold 3: : AUC = 0.72 | CBI = -0.29   
## K-fold 4: : AUC = 0.87 | CBI = 0.95   
## K-fold 5: : AUC = 0.87 | CBI = 0.9

say('Mean AUC:', round(mean(aucRandom), 2))

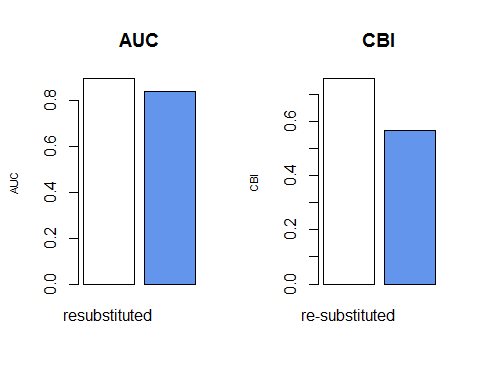
## Mean AUC:0.84

say('Mean CBI:', round(mean(cbiRandom), 2))

## Mean CBI:0.56

Notice how the metrics vary across k-folds. Normally we'd report the average of these plus a measure of variation like the range or standard deviation (not standard error). Let's compare these values to a case where we used re-substitution to evaluate the model (test presences - training presences).

## predict to presences and background sites of non-k-folded sites  
predPres <- predict(tunedModel, records, type='cloglog')  
predBg <- predict(tunedModel, randomBg, type='cloglog')  
  
# evaluate and remember result  
thisEval <- evaluate(p=as.vector(predPres), a=as.vector(predBg))  
  
aucResub <- thisEval@auc  
cbiResub <- contBoyce(pres=predPres, bg=predBg, numBins=100)  
  
# plot  
par(mfrow=c(1, 2), pty='m')  
barplot(c(aucResub, mean(aucRandom)),  
 names.arg=c('resubstituted', 'k-fold'),  
 cex.lab=0.7,  
 col=c('white', 'cornflowerblue'), ylab='AUC', main='AUC'  
)  
  
barplot(c(cbiResub, mean(cbiRandom)),  
 names.arg=c('re-substituted', 'k-fold'),  
 cex.lab=0.7,  
 col=c('white', 'cornflowerblue'), ylab='CBI', main='CBI'  
)



You will probably see that the re-substituted model has better apparent performance than the cross-validated models, although perhaps not (for reasons explored in this next section)! You will also probably see that CBI is a more sensitive measure than AUC.

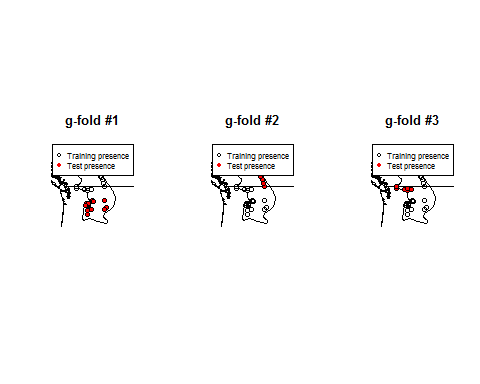
# Geographic cross-validation

Random cross-validation is a good start, but if test sites are close to training sites then they probably don't really represent very independent test sites. As a result, model performance will be falsely elevated. An alternative to randomly splitting test sites from training sites is to use **geographic g-folds** which divide sites spatially into sections. This helps increase the independence between them and increases the reliability of the evaluation metric.

Relevant citation: [Bahn, V. and McGill, B.J. 2013. Testing the predictive performance of distribution models. Oikos 122:321-331.](http://dx.doi.org/10.1111/j.1600-0706.2012.00299.x)

We'll use a function I wrote to generate g-folds, train models on all but each g-fold, and evaluate them. Note that in the previous section when using random k-folds we used 5 folds. Here we'll use just 3 because it's computationally hard to find 4 or more geographic divisions between these particular records while keeping the records geographically cohesive.

# calculate g-folds  
gPres <- geoFold(  
 x=records,  
 k=3,  
 minIn=5,  
 minOut=10,  
 longLat=c('longitude', 'latitude')  
)  
  
# maps  
par(mfrow=c(1, 3), pty='s')  
for (i in 1:3) {  
  
 plot(rangeMap, main=paste0('g-fold #', i))  
 sp::plot(countries, add=TRUE)  
 points(records$longitude, records$latitude)  
 points(records$longitude[gPres==i],  
 records$latitude[gPres==i],  
 bg='red',  
 pch=21  
 )  
  
 legend('topright',  
 legend=c('Training presence', 'Test presence'),  
 pch=c(1, 16),  
 col=c('black', 'red'),  
 bg='white',  
 cex=0.8  
 )  
   
}



You can see the g-folds are spatially segregated. Now we'll divide the background sites into g-folds based on their proximity to presences--i.e., a background site will be assigned to a particular g-fold if its closest presence is from that g-fold.

# initialize vectors to store g-fold assignments  
gTrainBg <- rep(NA, nrow(targetBg))  
gTestBg <- rep(NA, nrow(randomBg))  
  
# for each TRAINING presence, find closest test presence and use it's g-fold designation  
for (i in 1:nrow(targetBg)) {  
 dists <- distCosine(cbind(targetBg$longitude[i], targetBg$latitude[i]),  
 cbind(records$longitude, records$latitude))  
 closest <- which.min(dists)  
 gTrainBg[i] <- gPres[closest]  
}  
  
# for each TEST presence, find closest test presence and use it's g-fold designation  
for (i in 1:nrow(randomBg)) {  
 dists <- distCosine(cbind(randomBg$longitude[i], randomBg$latitude[i]),  
 cbind(records$longitude, records$latitude))  
 closest <- which.min(dists)  
 gTestBg[i] <- gPres[closest]  
}

Now that we've divided the presences and background sites into g-folds, let's train 3 models and evaluate each using the presences and background sites that are not in the g-folds used to train the model.

# create output directory  
dirCreate('./Models/Model 10 Model Evaluation - Geographic K-Folds')  
  
# for storing model evaluation metrics for each k-fold  
aucGeog <- cbiGeog <- numeric()  
  
# cycle through each k-fold  
for (i in 1:3) {  
  
 say('G-fold ', i, ':', post=0)  
   
 # make training data frame with predictors and vector of 1/0  
 # for presence/background  
 envData <- rbind(  
 records[gPres!=i, predictors],  
 targetBg[gTrainBg!=i, predictors]  
 )  
  
 presBg <- c(rep(1, sum(gPres!=i)), rep(0, sum(gTrainBg!=i)))  
   
 trainData <- cbind(presBg, envData)  
   
 # tuned model  
 model <- trainMaxNet(  
 data=trainData,  
 regMult=1,  
 classes='lpq',  
 out='model'  
 )  
  
 # save  
 save(model,  
 file=paste0('./Models/Model 10 Model Evaluation - Geographic K-Folds/Model ', i, '.Rdata'),  
 compress=TRUE)  
  
 # predict to presences and background sites  
 predPres <- predict(model, records[gPres==i, ], type='cloglog')  
 predBg <- predict(model, randomBg[gTestBg==i, ], type='cloglog')  
   
 # evaluate and remember result  
 thisEval <- evaluate(p=as.vector(predPres), a=as.vector(predBg))  
   
 thisAuc <- thisEval@auc  
 thisCbi <- contBoyce(pres=predPres, bg=predBg, numBins=100)  
   
 say(': AUC (geog) = ', round(thisAuc, 2), ' | CBI (geog) = ', round(thisCbi, 2))  
  
 aucGeog <- c(aucGeog, thisAuc)  
 cbiGeog <- c(cbiGeog, thisCbi)  
  
}

## G-fold 1: : AUC (geog) = 0.8 | CBI (geog) = 0.94   
## G-fold 2: : AUC (geog) = 0.96 | CBI (geog) = 0.6   
## G-fold 3: : AUC (geog) = 0.71 | CBI (geog) = 0.75

say('Mean AUC (geog): ', round(mean(aucGeog), 2))

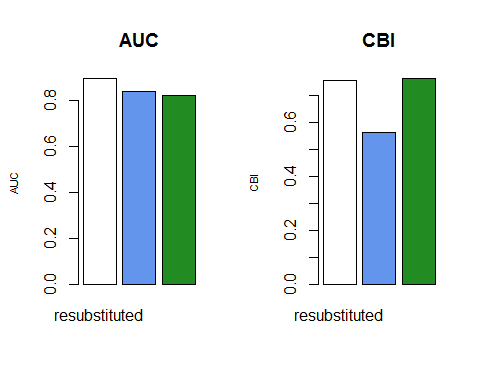
## Mean AUC (geog): 0.82

say('Mean CBI (geog): ', round(mean(cbiGeog), 2))

## Mean CBI (geog): 0.76

How do the three methods of evaluation--re-substitution, random k-folding, and g-folding--compare?

# plot  
par(mfrow=c(1, 2), pty='m')  
barplot(c(aucResub, mean(aucRandom), mean(aucGeog)),  
 names.arg=c('resubstituted', 'k-fold', 'geo-fold'),  
 cex.lab=0.7,  
 col=c('white', 'cornflowerblue', 'forestgreen'),  
 ylab='AUC',  
 main='AUC'  
)  
  
barplot(c(cbiResub, mean(cbiRandom), mean(cbiGeog)),  
 names.arg=c('resubstituted', 'k-fold', 'geo-fold'),  
 cex.lab=0.7,  
 col=c('white', 'cornflowerblue', 'forestgreen'),  
 ylab='CBI',  
 main='CBI'  
)



## Reflection

1. How did apparent model performance vary between re-substitution, random k-folds, and geographic g-folds?
2. Was AUC or CBI a more sensitive measure of model performance? Which had more variance within k-folds/g-folds?