Step4_Model_validation

setwd(paste0(direct,folder))

```
library(raster)
```

```
## Warning: package 'raster' was built under R version 4.1.2
```

```
## Loading required package: sp
```

```
require(rgdal)
```

```
## Loading required package: rgdal
```

```
## Warning: package 'rgdal' was built under R version 4.1.2
```

```
## Please note that rgdal will be retired by the end of 2023,
## plan transition to sf/stars/terra functions using GDAL and PROJ
## at your earliest convenience.
##
## rgdal: version: 1.5-29, (SVN revision 1165M)
## Geospatial Data Abstraction Library extensions to R successfully loaded
## Loaded GDAL runtime: GDAL 3.4.0, released 2021/11/04
## Path to GDAL shared files: /Library/Frameworks/R.framework/Versions/4.1/Resources/library/rgdal/gdal
## GDAL binary built with GEOS: TRUE
## Loaded PROJ runtime: Rel. 8.1.1, September 1st, 2021, [PJ_VERSION: 811]
## Path to PROJ shared files: /Library/Frameworks/R.framework/Versions/4.1/Resources/library/rgdal/proj
## PROJ CDN enabled: FALSE
## Linking to sp version:1.4-6
## To mute warnings of possible GDAL/OSR exportToProj4() degradation,
## use options("rgdal_show_exportToProj4_warnings"="none") before loading sp or rgdal.
## Overwritten PROJ_LIB was /Library/Frameworks/R.framework/Versions/4.1/Resources/library/rgdal/proj
```

```
direct<- "/Users/s2992269/Documents/Use_cases"
folder <- "/SDM_in_R"

#this sets your working director for all subsequent chunks of code in your R Markdown script
knitr::opts_knit$set(root.dir = paste0(direct,folder))

# if you are not working R Markdown, simply use this instead
setwd(paste0(direct,folder))
```

step 4 evaluate the model

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.1.2
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(dismo)
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```
library(raster)
library(sp)
library(jpeg)
library(galah)
```

```
## Warning: package 'galah' was built under R version 4.1.2
```

```
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 4.1.2
```

```
##
## Attaching package: 'tidyr'
```

```
## The following object is masked from 'package:raster':
##
##     extract
```

```
setwd(paste0(direct,folder))
getwd()
```

```
## [1] "/Users/s2992269/Documents/Use_cases/SDM_in_R"
```

```
## [1] "AWAP.asc"                 "Bioclim05.asc"
## [3] "Bioclim06.asc"            "Bioclim12.asc"
## [5] "EVI.asc"                  "wetland_connectivity.asc"
```

```
## class      : RasterStack
## dimensions : 6228, 4366, 27191448, 6  (nrow, ncol, ncell, nlayers)
## resolution : 0.002349, 0.002349  (x, y)
## extent     : 144.279, 154.5347, -29.90543, -15.27585  (xmin, xmax, ymin, ymax)
## crs        : +proj=longlat +datum=WGS84 +no_defs
## names      : AWAP, Bioclim05, Bioclim06, Bioclim12, EVI, wetland_connectivity
```

```
## Coordinate Reference System:
## Deprecated Proj.4 representation: +proj=longlat +datum=WGS84 +no_defs
## WKT2 2019 representation:
## GEOGCRS["unknown",
##     DATUM["World Geodetic System 1984",
##         ELLIPSOID["WGS 84",6378137,298.257223563,
##             LENGTHUNIT["metre",1]],
##         ID["EPSG",6326]],
##     PRIMEM["Greenwich",0,
##         ANGLEUNIT["degree",0.0174532925199433],
##         ID["EPSG",8901]],
##     CS[ellipsoidal,2],
##         AXIS["longitude",east,
##             ORDER[1],
##             ANGLEUNIT["degree",0.0174532925199433,
##                 ID["EPSG",9122]]],
##         AXIS["latitude",north,
##             ORDER[2],
##             ANGLEUNIT["degree",0.0174532925199433,
##                 ID["EPSG",9122]]]]
```

```
##   X     Long       Lat                  spp pres
## 1 1 145.1798 -15.43206 Limnodynastes_peroni    0
## 2 2 145.1821 -15.43206 Limnodynastes_peroni    0
## 3 3 145.1821 -15.43676 Limnodynastes_peroni    0
## 4 4 145.1821 -15.43911 Limnodynastes_peroni    0
## 5 5 145.2503 -15.46730 Limnodynastes_peroni    0
## 6 6 145.2550 -15.47200 Limnodynastes_peroni    0
```

```
##       Long       Lat pres
## 1 145.1798 -15.43206    0
## 2 145.1821 -15.43206    0
## 3 145.1821 -15.43676    0
## 4 145.1821 -15.43911    0
## 5 145.2503 -15.46730    0
## 6 145.2550 -15.47200    0
```

```
##       Long       Lat pres
## 1 145.3323 -16.26103    1
## 2 145.3880 -16.56270    1
## 3 145.2750 -16.59310    1
## 4 145.2732 -16.60149    1
## 5 145.3270 -16.63571    1
## 6 145.3270 -16.65170    1
```

```
## Coordinate Reference System:
## Deprecated Proj.4 representation: +proj=longlat +datum=WGS84 +no_defs
## WKT2 2019 representation:
## GEOGCRS["unknown",
##     DATUM["World Geodetic System 1984",
##         ELLIPSOID["WGS 84",6378137,298.257223563,
##             LENGTHUNIT["metre",1]],
##         ID["EPSG",6326]],
```

```
##     PRIMEM["Greenwich",0,
##         ANGLEUNIT["degree",0.0174532925199433],
##         ID["EPSG",8901]],
##     CS[ellipsoidal,2],
##         AXIS["longitude",east,
##             ORDER[1],
##             ANGLEUNIT["degree",0.0174532925199433,
##                 ID["EPSG",9122]]],
##         AXIS["latitude",north,
##             ORDER[2],
##             ANGLEUNIT["degree",0.0174532925199433,
##                 ID["EPSG",9122]]]]
```

```
##       Long            Lat             pres             AWAP        
##  Min.   :144.6   Min.   :-28.92   Min.   :0.0000   Min.   :0.02754  
##  1st Qu.:148.2   1st Qu.:-27.52   1st Qu.:0.0000   1st Qu.:0.12221  
##  Median :152.8   Median :-26.67   Median :1.0000   Median :0.15185  
##  Mean   :150.9   Mean   :-24.44   Mean   :0.6013   Mean   :0.16231  
##  3rd Qu.:153.0   3rd Qu.:-20.34   3rd Qu.:1.0000   3rd Qu.:0.19023  
##  Max.   :153.5   Max.   :-15.43   Max.   :1.0000   Max.   :0.38251  
##    Bioclim05       Bioclim06        Bioclim12           EVI          
##  Min.   :23.82   Min.   : 1.600   Min.   : 561.9   Min.   :-0.0411502  
##  1st Qu.:29.17   1st Qu.: 8.000   1st Qu.: 999.5   1st Qu.:-0.0043831  
##  Median :29.76   Median : 8.992   Median :1208.5   Median :-0.0019141  
##  Mean   :30.01   Mean   : 9.760   Mean   :1310.1   Mean   :-0.0019677  
##  3rd Qu.:31.09   3rd Qu.:11.299   3rd Qu.:1552.8   3rd Qu.: 0.0007291  
##  Max.   :35.04   Max.   :18.400   Max.   :3866.7   Max.   : 0.0235833  
##  wetland_connectivity
##  Min.   :0.02754     
##  1st Qu.:0.12221     
##  Median :0.15185     
##  Mean   :0.16231     
##  3rd Qu.:0.19023     
##  Max.   :0.38251
```

```
## [1] "Long"                "Lat"                 "pres"
## [4] "AWAP"                "Bioclim05"           "Bioclim06"
## [7] "Bioclim12"           "EVI"                 "wetland_connectivity"
```

There are many ways to evaluate your model. With statistical models you can evaluate residual plots to check that your model meets assumptions.

It is often a good idea to break your data into randomly selected training and testing data. Often you would randomly remove 20% or more of

your data and withold that from the model building process. Once your model was finsihed using your training data, you would then test your

model with this training data. If you have very little data, bootstrapping can be used to see if removal of a small percentage of data repeatedly

changes results. This gives you a good understanding of the confidence intervals around your results and can reduce the impact of outliers on your

inal result. Cross-validation requires more data. I deally for cross-validation you break your data into 10 folds (subsets) and compare results between

folds, or summarise variability between folds. If you can afford to set aside 20% + of your data, withholding a test data set

is good practice. Ideally, you will test your model with completely independent data.

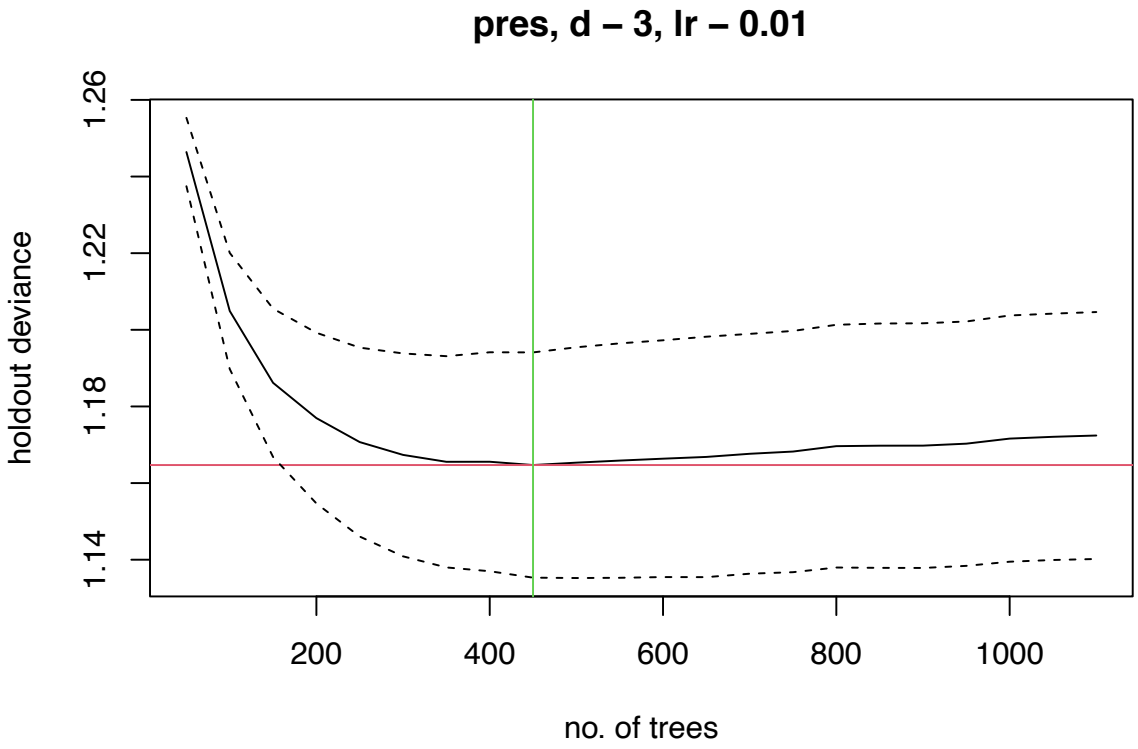
here we evaluate the BRT model with the training data used to construct the model

First we run the BRT model again without additional variables identified in our simplify function

```
LiPe_mod_new <- gbm.step(data = LiPe_preds3, gbm.x = c(4:7), gbm.y = 3, family = "bernoulli", tree.compl
```

```
##
##
##  GBM STEP - version 2.9
##
## Performing cross-validation optimisation of a boosted regression tree model
## for pres and using a family of bernoulli
## Using 1580 observations and 4 predictors
## creating 10 initial models of 50 trees
##
##  folds are stratified by prevalence
## total mean deviance =  1.345
## tolerance is fixed at  0.0013
## ntrees resid. dev.
## 50    1.2464
## now adding trees...
## 100   1.2049
## 150   1.1862
## 200   1.177
## 250   1.1707
## 300   1.1674
## 350   1.1656
## 400   1.1656
## 450   1.1647
## 500   1.1653
## 550   1.1659
## 600   1.1664
## 650   1.1668
## 700   1.1676
## 750   1.1682
## 800   1.1696
## 850   1.1698
## 900   1.1698
## 950   1.1703
## 1000   1.1716
## 1050   1.1721
## 1100   1.1724

## fitting final gbm model with a fixed number of 450 trees for pres
```




```
##
## mean total deviance = 1.345
## mean residual deviance = 1.086
##
## estimated cv deviance = 1.165 ; se = 0.029
##
## training data correlation = 0.502
## cv correlation =  0.422 ; se = 0.032
##
## training data AUC score = 0.789
## cv AUC score = 0.72 ; se = 0.023
##
## elapsed time -  0.07 minutes
```

```
LiPe_1s <- LiPe_preds3[LiPe_preds3$pres==1,]
x_1s<- LiPe_1s$Long
y_1s<-LiPe_1s$Lat
xy_1s<-cbind(x_1s,y_1s)
xy.sp_1s<-SpatialPoints(xy_1s)
crs(xy.sp_1s) <- crs(LiPe_predictors)
crs(xy.sp_1s)
```

```
## Coordinate Reference System:
## Deprecated Proj.4 representation: +proj=longlat +datum=WGS84 +no_defs
## WKT2 2019 representation:
```

```
## GEOGCRS["unknown",
##     DATUM["World Geodetic System 1984",
##         ELLIPSOID["WGS 84",6378137,298.257223563,
##             LENGTHUNIT["metre",1]],
##         ID["EPSG",6326]],
##     PRIMEM["Greenwich",0,
##         ANGLEUNIT["degree",0.0174532925199433],
##         ID["EPSG",8901]],
##     CS[ellipsoidal,2],
##         AXIS["longitude",east,
##             ORDER[1],
##             ANGLEUNIT["degree",0.0174532925199433,
##                 ID["EPSG",9122]]],
##         AXIS["latitude",north,
##             ORDER[2],
##             ANGLEUNIT["degree",0.0174532925199433,
##                 ID["EPSG",9122]]]]
```

```
LiPe_0s <- LiPe_preds3[LiPe_preds3$pres==0,]
x_0s<- LiPe_0s$Long
y_0s<-LiPe_0s$Lat
xy_0s<-cbind(x_0s,y_0s)
xy.sp_0s<-SpatialPoints(xy_0s)
crs(xy.sp_0s) <- crs(LiPe_predictors)
crs(xy.sp_0s)
```

```
## Coordinate Reference System:
## Deprecated Proj.4 representation: +proj=longlat +datum=WGS84 +no_defs
## WKT2 2019 representation:
## GEOGCRS["unknown",
##     DATUM["World Geodetic System 1984",
##         ELLIPSOID["WGS 84",6378137,298.257223563,
##             LENGTHUNIT["metre",1]],
##         ID["EPSG",6326]],
##     PRIMEM["Greenwich",0,
##         ANGLEUNIT["degree",0.0174532925199433],
##         ID["EPSG",8901]],
##     CS[ellipsoidal,2],
##         AXIS["longitude",east,
##             ORDER[1],
##             ANGLEUNIT["degree",0.0174532925199433,
##                 ID["EPSG",9122]]],
##         AXIS["latitude",north,
##             ORDER[2],
##             ANGLEUNIT["degree",0.0174532925199433,
##                 ID["EPSG",9122]]]]
```

look at which predictors are in our predictors stack

```
names(LiPe_predictors)
```

```
## [1] "AWAP"                 "Bioclim05"            "Bioclim06"
## [4] "Bioclim12"            "EVI"                  "wetland_connectivity"
```

Here we drop the variables that were dropped from the LiPe_mod_new BRT model

This is so we can predict all locations using the subset of variables used in the model

In any model, the variables you use, need to match the variables you use to predict

column names, or raster stack names need to match exactly

```
LiPe_preds_brt_new <- LiPe_predictors[[which(c(TRUE,TRUE,TRUE,TRUE,FALSE,FALSE))]]

brt_pred <- predict(LiPe_preds_brt_new,LiPe_mod_new,type="link") # note we usually use type link, but t
```

```
## Using 450 trees...
##
## Using 450 trees...
##
## Using 450 trees...
##
## Using 450 trees...
##
## Using 450 trees...
##
## Using 450 trees...
##
## Using 450 trees...
##
## Using 450 trees...
##
## Using 450 trees...
```

```
brt_eval_training <- dismo::evaluate(xy.sp_1s, xy.sp_0s, LiPe_mod_new, LiPe_preds_brt_new)
```

```
## Using 450 trees...
##
## Using 450 trees...
```

#There are a variety of ways to calculate a threshold, maximising "kappa" is one well supported method, but there are many others
in Maxent we saw a table with a variety of ways to calculate thresholds

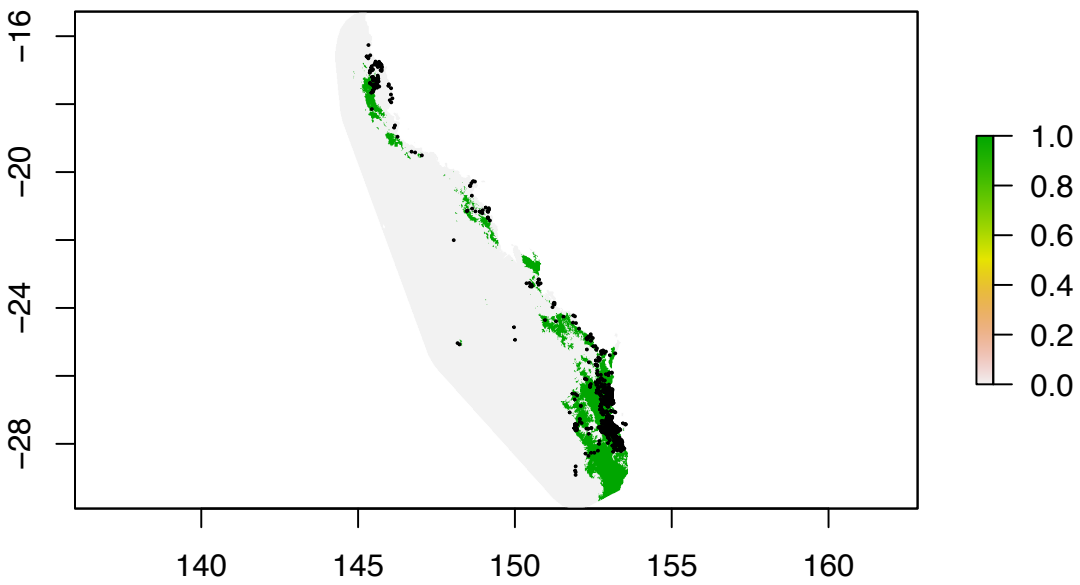
```
thresh_brt <- threshold(brt_eval_training,stat = "kappa")
```

another approach to thresholding - there are many choose the one that best fits in your model

thresh_brt2 <- threshold(brt_eval_training,stat = "spec_sens")

```
m <- c(-5, thresh_brt, 0,  thresh_brt, 2.1, 1)
reclass <- matrix(m, ncol= 3, byrow= TRUE)
rc_brt <- reclassify(brt_pred, reclass)

plot(rc_brt)
points(xy.sp_LiPe,pch=20,cex=0.2)
```



look at training evaluation

```
brt_eval_training
```

```
## class          : ModelEvaluation
## n presences    : 950
## n absences     : 630
## AUC            : 0.7890284
## cor            : 0.5045333
## max TPR+TNR at : 0.6269524
```

```
x_s<- LiPe_preds3$Long
y_s<-LiPe_preds3$Lat
xy_s<-cbind(x_s,y_s)
xy.sp_s<-SpatialPoints(xy_s)
crs(xy.sp_s) <- crs(LiPe_predictors)

predicted <- raster::extract(rc_brt,xy.sp_s)

test1 <- as.data.frame(cbind(predicted = predicted, reference = LiPe_preds3$pres))

test1$reference <- factor(test1$reference, levels = c("1","0"))
test1$predicted <- factor(test1$predicted, levels = c("1","0"))

conf_matrix <- confusionMatrix(test1$predicted,test1$reference)
```

this gives us the confusion matrix

```
conf_matrix$table
```

```
##            Reference
## Prediction   1   0
##          1 792 255
##          0 158 375
```

```
TP <- conf_matrix$table[1,1]
FP <- conf_matrix$table[1,2]
FN <- conf_matrix$table[2,1]
TN <- conf_matrix$table[2,2]

FPR <- FP/(FP + TN)
FNR <- FN/(FN + TP)
TPR <- TP/(TP + FN)
TNR <- TN/(TN + FP)

TSS <- TPR + TNR - 1  # if we were predicting perfectly our TSS would = 1
TSS # we are far from perfect in this model, and a much lower number that AUC
```

```
## [1] 0.4289223
```

a perfect AUC value would also be 1

TSS often gives a better indication of prediction because it takes into

account anbalanced errors between positives and negatives

```
Precision <- TP / (TP + FP)
Recall <- TP / (TP + FN)
F1 <- (2*Precision*Recall)/(Precision+Recall)#  F1 is another statistic, now thought to be
```

better than TSS

```
F1 <- TP /(TP + 0.5*(FP + FN)) # this is another way to calculate the same value
```

but the literature refers to Precision and Recall above (same thing though)

again perfect score would be 1

```
F1 # as you can see in our case F1 falls between AUC and TSS scores
```

```
## [1] 0.7931898
```

now we are going to test our results from ALA records that were not from the FrogID project

for the most part we are just repeating code from step1

```
galah_config(email = "r.clemens@griffith.edu.au")
```

```
## These configuration options will only be saved for this session.
##     Set 'preserve = TRUE' to preserve them for future sessions.
```

```
LiPe_ALA<-galah_call() %>%
  galah_identify("Limnodynastes peroni")%>%
  galah_filter(datasetName != "FrogID")%>%
  galah_filter(coordinateUncertaintyInMeters < 200)%>%
```

```
  galah_filter(year>1999)%>%
  galah_filter(stateProvince == "Queensland")%>%
  galah_select("datasetName","year")%>%
  atlas_occurrences()
```

```
## This query will return 851 records
```

```
##   |                                                                      |
```

```
LiPe_ALA<-na.omit(LiPe_ALA)

LiPe_pts_ALA <- SpatialPoints(coords = cbind(LiPe_ALA$decimalLongitude, LiPe_ALA$decimalLatitude),CRS(as

LiPe_pres_ALA<- rep(1,length(LiPe_ALA$decimalLatitude))
```

Then create a raster with a value of 1 for each gridcell where a LiPe was recorded

```
LiPe_pres_raster_ALA<-raster::rasterize(LiPe_pts_ALA,base,LiPe_pres_ALA, fun = min, background=0)
LiPe_pres_raster2_ALA<-raster::projectRaster(LiPe_pres_raster_ALA,base)
```

```
## Warning in raster::projectRaster(LiPe_pres_raster_ALA, base): input and ouput
## crs are the same
```

```
LiPe_pres_raster2_ALA<-raster::mask(LiPe_pres_raster2_ALA,base)
large_base <- aggregate(base, fact=4)
cell_no_ALA<- raster::extract(large_base,LiPe_pts_ALA,cellnumbers=TRUE)
LiPe_cells_ALA<- cbind(LiPe_ALA,cell_no_ALA)
```

```
require(dplyr)
```

```
## Loading required package: dplyr
```

```
## Warning: package 'dplyr' was built under R version 4.1.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:raster':
##
##     intersect, select, union
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
LiPe_thinned_ALA <- LiPe_cells_ALA %>%
  group_by(cells) %>%
  slice_sample(n = 1)
```

```
frogs_ALA<-galah_call() %>%
  galah_identify("Amphibia")%>%
  galah_filter(datasetName != "FrogID")%>%
  galah_filter(coordinateUncertaintyInMeters < 100)%>%
  galah_filter(year>1999)%>%
  galah_filter(stateProvince == "Queensland")%>%
  galah_select("datasetName","year")%>%
  atlas_occurrences()
```

```
## This query will return 15019 records
```

```
##   |                                                                      |
```

```
frogs_ALA$unique_visit<- paste0(frogs_ALA$decimalLatitude,frogs_ALA$decimalLongitude,frogs_ALA$eventDat
frogs_ALA$visitID <- as.numeric(as.factor(frogs_ALA$unique_visit))
```

```
frogs_ALA2 <- frogs_ALA %>%
  group_by(decimalLatitude, decimalLongitude, visitID) %>%
  summarise(no_spp = length(unique(scientificName)))
```

```
## `summarise()` has grouped output by 'decimalLatitude', 'decimalLongitude'. You
## can override using the `.groups` argument.
```

```
frogs_ALA3 <- frogs_ALA2 %>%
  group_by(decimalLatitude, decimalLongitude) %>%
  summarise(no_visits = length(visitID))
```

```
## `summarise()` has grouped output by 'decimalLatitude'. You can override using
## the `.groups` argument.
```

```
frogs_ALA3 <- na.omit(frogs_ALA3)

visits_pts_ALA<-SpatialPoints(coords = cbind(frogs_ALA3$decimalLongitude, frogs_ALA3$decimalLatitude),C
b1_ALA <- rasterize(visits_pts_ALA, base, frogs_ALA3$no_visits, fun=sum, background=0)

bb_ALA <- bbox(b1_ALA)
visit_locations_ALA<- raster::extract(b1_ALA, visits_pts_ALA, cellnumbers=TRUE)
visit_locations2_ALA <- as.data.frame(na.omit(visit_locations_ALA))
cellID_ALA <- unique(visit_locations2_ALA$cells)
```

```
xy_visits_ALA <- raster::xyFromCell(b1_ALA,cell = cellID_ALA)

cellStats(b1_ALA ,"max")
```

```
## [1] 261
```

```
m <- c(0, 2.9, 0,  2.9, 880, 1)
reclass <- matrix(m, ncol= 3, byrow= TRUE)
rc_ALA <- reclassify(b1_ALA, reclass)

visits_3or_more_ALA <- mask(rc_ALA,base)

Zero_LiPe_ALA <- visits_3or_more_ALA - LiPe_pres_raster2_ALA

freq(Zero_LiPe_ALA)
```

```
##      value    count
## [1,]    -1      539
## [2,]     0  8787725
## [3,]     1      358
## [4,]    NA 18402826
```

```
m <- c(-2, 0.1, 0,  0.1, 2, 1)
reclass2 <- matrix(m, ncol= 3, byrow= TRUE)
rc2_ALA <- reclassify(Zero_LiPe_ALA, reclass2)
Zero_LiPe2_ALA<-mask(rc2_ALA,base)
freq(Zero_LiPe2_ALA)
```

```
##      value    count
## [1,]     0  8788264
## [2,]     1      358
## [3,]    NA 18402826
```

extract the cell numbers from the 0 grid where the value ==1

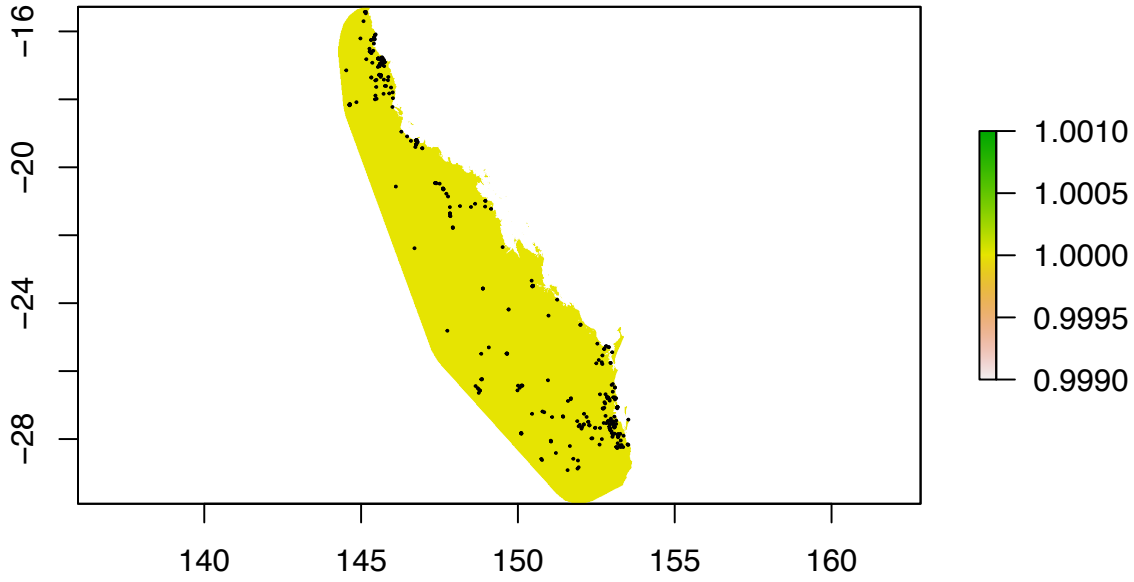
```
cell_vals_0_ALA<-Which(Zero_LiPe2_ALA ==1,cells=TRUE)
```

these are the lat / longs for locations where at least three surveys were done, but zero LiPe were detected - these are our pseudo absences

```
xy_zero_LiPe_ALA <- xyFromCell(Zero_LiPe2_ALA,cell = cell_vals_0_ALA)
```

#plot the absence locations

```
plot(base)
points(xy_zero_LiPe_ALA, pch=20,cex=0.2)
```




```
xy_zero_LiPe_ALA <- as.data.frame(xy_zero_LiPe_ALA)
```

Now lets turn our presence and absence points into one dataset

```
head(LiPe_thinned_ALA)
```

```
## # A tibble: 6 x 11
## # Groups:   cells [6]
##   datasetName     year decimalLatitude decimalLongitude eventDate scientificName
##   <chr>          <int>           <dbl>            <dbl> <chr>     <chr>
## 1 ""              2011           -16.2             145. "2011-09~ Limnodynastes~
## 2 ""              2012           -16.2             145. "2012-05~ Limnodynastes~
## 3 "iNaturalist ~  2014           -16.3             145. ""        Limnodynastes~
## 4 ""              2007           -16.7             145. "2007-03~ Limnodynastes~
## 5 "iNaturalist ~  2019           -16.8             146. "2019-09~ Limnodynastes~
## 6 ""              2012           -16.8             146. "2012-04~ Limnodynastes~
## # ... with 5 more variables: taxonConceptID <chr>, recordID <chr>,
## #   dataResourceName <chr>, cells <dbl>, base_LiPe <dbl>
```

```
ALA_preds1 <- as.data.frame(cbind(Long = LiPe_thinned_ALA$decimalLongitude, Lat = LiPe_thinned_ALA$deci
ALA_preds2 <- as.data.frame(cbind(Long = xy_zero_LiPe_ALA$x, Lat = xy_zero_LiPe_ALA$y,pres = 0))
ALA_preds <- rbind(ALA_preds1,ALA_preds2)

x_a<- ALA_preds$Long
y_a<-ALA_preds$Lat
xy_a<-cbind(x_a,y_a)
xy.sp_a<-SpatialPoints(xy_a)
crs(xy.sp_a) <- crs(LiPe_predictors)
```

now we just extract the 1's & 0's using our lat longs from the new ALA data

extracted from the same thresholded brt predictions from above

```
predicted_a <- raster::extract(rc_brt,xy.sp_a)

test2 <- as.data.frame(cbind(predicted = predicted_a, reference = ALA_preds$pres))

test2$reference <- factor(test2$reference, levels = c("1","0"))
test2$predicted <- factor(test2$predicted, levels = c("1","0"))

conf_matrix2 <- confusionMatrix(test2$predicted,test2$reference)
conf_matrix2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   0
##          1 405 113
##          0 126 222
##
##                Accuracy : 0.724
##                  95% CI : (0.6929, 0.7536)
##     No Information Rate : 0.6132
##     P-Value [Acc > NIR] : 4.718e-12
##
##                   Kappa : 0.4224
##
##  Mcnemar's Test P-Value : 0.4376
##
##             Sensitivity : 0.7627
##             Specificity : 0.6627
##          Pos Pred Value : 0.7819
##          Neg Pred Value : 0.6379
##              Prevalence : 0.6132
##          Detection Rate : 0.4677
##    Detection Prevalence : 0.5982
##       Balanced Accuracy : 0.7127
##
##        'Positive' Class : 1
##
```

```r
TP2 <- conf_matrix2$table[1,1]
FP2 <- conf_matrix2$table[1,2]
FN2 <- conf_matrix2$table[2,1]
TN2 <- conf_matrix2$table[2,2]

FPR2 <- FP2/(FP2 + TN2)
FNR2 <- FN2/(FN2 + TP2)
TPR2 <- TP2/(TP2 + FN2)
TNR2 <- TN2/(TN2 + FP2)

TSS2 <- TPR2 + TNR2 - 1
TSS2 # notice this is a big drop from our original TSS statistic
```

```
## [1] 0.4253984
```

```r
TSS
```

```
## [1] 0.4289223
```

```r
Precision2 <- TP2 / (TP2 + FP2)
Recall2 <- TP2 / (TP2 + FN2)
F1_b <- 2*((Precision2*Recall2)/(Precision2+Recall2))
F1_b # notice this is a very slight drop from our original F1 statsitic
```

```
## [1] 0.772164
```

```r
F1
```

```
## [1] 0.7931898
```

When we used data to test the model that was not associated with the FrogID project, TSS, accuracy, & precision scores all fell markedly

Encouragingly the F1 score did not fall that much, and as a user you will need to decidd on what levels of accuracy in your confusion matrix are good enough for the intended use of the model

If I was looking to predict well in order to select what reserves to make for this frog, I would want better performance.

The first step I would take to improve this model, would be to select pseudo_absence locations at only those locations where similarly distributed frogs were seen but our target species was not seen on three visits. Our psuedo absence points are still including places in the arid interior where this species would never occur.

Second, as mentioned before, I would also look for better freshwater wetland layers, perhaps look at how often an area is wet over the last decoade.

OTHER places to learn how to generate SDMs in R

http://www.earthskysea.org/best-practices-in-species-distribution-modeling-a-workshop-in-r/

a series of lectures on SDMs https://www.youtube.com/watch?v=obuMW5NAtJE