Heteronychus arator

Nico

2/18/2022

## Front Matter

The following script was developed cooperatively by the SHSU SDM working group, including Laura Bianchi, Austin Brenek, Jesus Castillo, Nick Galle, Kayla Hankins, Kenneth Nobleza, Chris Randle, Nico Reger, Alyssa Russell, Ava Stendahl based on [tutorials](https://rspatial.org/) provided by Robert Hijmans and Jane Elith. Chris Randle composed the following script from many scripts developed by the SHSU SDM working group.

*This works best if your environment is empty at the start.*

I have tried to set this up to eliminate required changes to the code. When you see text in **BOLD** below, that will be an indication that you need to make a decision.

# Libraries

library(dismo)

## Loading required package: raster

## Loading required package: sp

library(sp)  
library(raster)  
library(stats)  
library(dplyr)

##   
## 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

library(knitr)  
library(rgeos)

## rgeos version: 0.5-8, (SVN revision 679)  
## GEOS runtime version: 3.9.1-CAPI-1.14.2   
## Please note that rgeos will be retired by the end of 2023,  
## plan transition to sf functions using GEOS at your earliest convenience.  
## GEOS using OverlayNG  
## Linking to sp version: 1.4-5   
## Polygon checking: TRUE

library(maptools)

## Checking rgeos availability: TRUE  
## Please note that 'maptools' will be retired by the end of 2023,  
## plan transition at your earliest convenience;  
## some functionality will be moved to 'sp'.

library(rgdal)

## 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: 1.5-27, (SVN revision 1148)  
## Geospatial Data Abstraction Library extensions to R successfully loaded  
## Loaded GDAL runtime: GDAL 3.2.1, released 2020/12/29  
## Path to GDAL shared files: C:/Users/User/Documents/R/win-library/4.1/rgdal/gdal  
## GDAL binary built with GEOS: TRUE   
## Loaded PROJ runtime: Rel. 7.2.1, January 1st, 2021, [PJ\_VERSION: 721]  
## Path to PROJ shared files: C:/Users/User/Documents/R/win-library/4.1/rgdal/proj  
## PROJ CDN enabled: FALSE  
## Linking to sp version:1.4-5  
## 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 C:/Users/User/Documents/R/win-library/4.1/rgdal/proj

library(ecospat)

## Loading required package: ade4

## Loading required package: ape

##   
## Attaching package: 'ape'

## The following objects are masked from 'package:raster':  
##   
## rotate, zoom

## Loading required package: gbm

## Loaded gbm 2.1.8

## Registered S3 methods overwritten by 'adehabitatMA':  
## method from  
## print.SpatialPixelsDataFrame sp   
## print.SpatialPixels sp

library(usdm)  
library(mgcv)

## Loading required package: nlme

##   
## Attaching package: 'nlme'

## The following object is masked from 'package:usdm':  
##   
## Variogram

## The following object is masked from 'package:dplyr':  
##   
## collapse

## The following object is masked from 'package:raster':  
##   
## getData

## This is mgcv 1.8-38. For overview type 'help("mgcv-package")'.

# Genus and Species strings

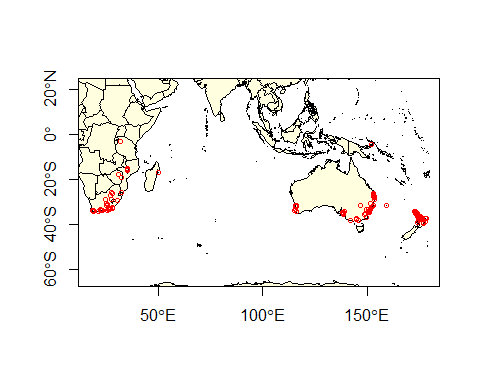
Their are many places in this code where you will need to save files with filenames including the genus and species. We’ll save these as strings to automate the creation of file names. Enter your genus name and specific epithet in the quotes below.

genus<-"Heteronychus"  
species<-"arator"

## Occurrence Data

Import occurrence data from csv file already generated (2020-2021), or using the script “Occurrence\_Data.rmd” and visualize it.

sdmdata <- read.csv(file = "Heteronychus arator.CLEAN.csv")  
sdmdata <- subset(sdmdata, !is.na(lon) & !is.na(lat))  
dups <-duplicated(sdmdata[ c('lon', 'lat')])  
sdmdata<-sdmdata[!dups,]  
#and visualize the data  
#first lets get the extent of the data (the coordinates of the smallest box needed to encapsulate the data) To do this I first need to convert sdmdata into a spatial points dataframe with the same crs as "wrldsmpl", a giant spatial polygons data frame available from maptools  
sdmdataframe<-data.frame(sdmdata)  
data(wrld\_simpl)  
coordinates(sdmdataframe) <- ~lon+lat  
crs(sdmdataframe) <- crs(wrld\_simpl)  
#And then extract the extent  
e<-extent(sdmdataframe)  
xmin<-xmin(e)  
xmax<-xmax(e)  
ymin<-ymin(e)  
ymax<-ymax(e)  
# and then plot a map and add the points from sdmdata  
plot(wrld\_simpl, xlim=c(xmin,xmax), ylim=c(ymin,ymax), axes=TRUE, col="light yellow")  
box()  
points(sdmdata$lon, sdmdata$lat, col='red', cex=0.75)



Let’s divide the data into training and testing data sets. The following code divides the data set into 80% training and 20% testing.

#let's make sdmdata into a dataframe  
data(wrld\_simpl)  
coordinates(sdmdata) <- ~lon+lat  
crs(sdmdata) <- crs(wrld\_simpl)  
  
#let's extract just the coordinates  
presence <- coordinates(sdmdata)  
#First we'll make a random list of integers from 1-5 as long as our presence data. Setting the seed results in a repeatable random process  
set.seed(0)  
#now make a list as long as the number of rows in presence consisting of a random series of integers from 1-5  
group <- kfold(presence, 5)  
#Then we want to use this to retrieve the number of the rows in the presence data that are associated with the number 1 in our group index.  
test\_indices <- as.integer(row.names(presence[group == 1, ]))  
#and create a new list of coordinates including only those rows that are NOT in test indices. These are all the row numbers NOT corresponding with the test\_indices (which is ~80% of the data).  
pres\_train <- presence[-test\_indices,]  
#and those that do correspond with test indices (20%) of the data  
pres\_test <- presence[group ==1,]

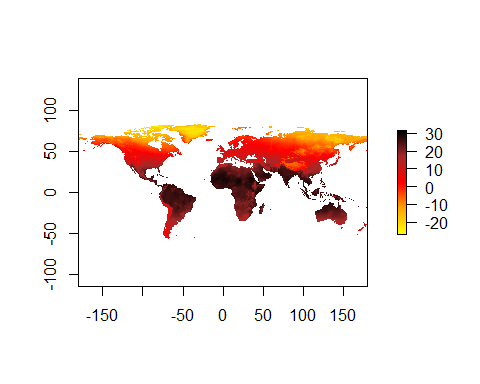
Save pres\_data and test\_data as csv files just in case.

#first presdata\_train  
outdata<-data.frame(pres\_train)  
colnames(outdata)<-c("lon","lat")  
write.csv(outdata, file=paste0(genus,"\_",species,"\_train.csv"), row.names=FALSE)  
  
#and then presdata\_test  
outdata<-data.frame(pres\_test)  
colnames(outdata)<-c("lon","lat")  
write.csv(outdata, file=paste0(genus,"\_",species,"\_test.csv"), row.names=FALSE)

## Predictor data

Let’s get the giant predictor file, name the bands, and generate our raster color schemes. This predictor set consists of all 35 Climond layers and elevation. Get it from Randle and keep it in your directory.

predictors<-stack('Climond\_elev\_HI.tif')  
bands<-c("Ann\_mean\_temp", "Mean\_durnal\_temp\_range", "Isothermality", "Temp\_Seasonality", "MaxTemp\_WarmestWeek", "MinTemp\_ColdestWeek", "Temp\_Annual\_Range", "Mean\_temp\_wettest\_q", "Mean\_temp\_driest\_q", "Mean\_temp\_warmiest\_q", "Mean\_temp\_coldest\_q", "Ann\_ precip", "Precip\_driest\_week", "Precip\_wettest\_week", "Precip\_Seasonality", "Precip\_wettest\_q", "Precip\_driest\_q", "Precip\_warmest\_q", "Precip\_coldest\_q", "Ann\_mean\_rad", "Hghest\_weekly\_rad", "Lowest\_weekly\_rad", "Rad\_seasonality", "Rad\_wettest\_q", "Rad\_driest\_q", "Rad\_warmest\_q", "Rad\_coldest\_q", "Ann\_mean\_moisture", "Highsets\_weekly\_moisture", "Lowest\_weekly\_moisture", "Moisture\_seasonality", "Mean\_moisture\_wettest\_q", "Mean\_moisture\_driest\_q", "Mean\_moisture\_warmest\_q", "Mean\_moisture\_coldest\_q", "Elev", 'human\_impact')  
names(predictors)<-bands  
cool<-colorRampPalette(c('gray','green','dark green',"blue"))  
warm<-colorRampPalette(c('yellow', 'orange', 'red', 'brown', 'black'))  
plot(predictors[["Ann\_mean\_temp"]], col=warm(100))

 And now we will use the VIFstep function to identify layers contributing most to collinearity (variance inflation factor). Rather than do this from a raster, I think it makes much more sense to do this from a dataframe in which we have sampled all the layers at the presence points only. This is because the larger a species distribution is, the lower the probability of collinearity across the range, even if layers are collinier where the species actually exists in the range.

#extract environmental data using the points in sdmdata  
env\_data<-extract(predictors,sdmdata)  
#give names to the columns  
colnames(env\_data)<-bands  
#run the vif  
vif<-vifstep(env\_data)  
#and let's find the layers that were excluded and drop them  
excluded<-vif@excluded  
predictors<-dropLayer(predictors,excluded)  
#and let's just go ahead and see which layers were dropped.  
NClayers<-names(predictors)  
NClayers

## [1] "Mean\_durnal\_temp\_range" "Isothermality" "Mean\_temp\_warmiest\_q"   
## [4] "Ann\_.precip" "Precip\_Seasonality" "Precip\_warmest\_q"   
## [7] "Precip\_coldest\_q" "Hghest\_weekly\_rad" "Rad\_wettest\_q"   
## [10] "Rad\_driest\_q" "Elev" "human\_impact"

## General additive model

# Data preparation

Generally speaking, we want to sample absence data from the region in which the presence data occur. There are two ways to do that, and one of them is better than the other. The first is to sample randomly. That may seem like a good idea, but its counter-intuitively not. The reason is that the presence data likely includes sampling bias. This will be inherent in p(hypothesis). If we include the same sampling bias in p(data), they cancel out in Bayes Theorem. The way that we’ll do that is to create circles around our data and sample absence points from within those. The circles will have a diameter equal to the average distance between points.

#convert presence training data into a spatial points dataframe.  
pres\_train\_SPDF<-SpatialPoints(pres\_train)  
crs(pres\_train\_SPDF) <- crs(wrld\_simpl)  
#Let's get the average distance between points (great circle distance--takes into account the curvature of the earth). spDists creates a matrix of distances between points. This includes zeros.   
#I have to use a smaller distance in this case becacuse of the raster geometry  
dist<-110000  
x <- circles(pres\_train\_SPDF, d=110000, lonlat=TRUE)  
#and convert those into polygons  
pol <- polygons(x)  
#and draw a number of samples from that approximately three times the number of presence points. We'll chop that down at the end.  
samp1 <- spsample(pol, nrow(pres\_train)\*3, type='random', iter=25)

## Warning in proj4string(obj): CRS object has comment, which is lost in output

#and get the cell numbers from the raster stack (right to left, up to down)  
cells <- cellFromXY(predictors, samp1)  
#and transform each of those to the center of its cell.  
abs\_train <- xyFromCell(predictors, cells)  
#You'll get a warning saying that your CRS object has lost a comment. This is unimportant and can be ignored.

And let’s go ahead and extract the presence data, remove rows with NA values, and add a column of 1s.

pres\_train\_data<-extract(predictors,pres\_train)  
complete<-complete.cases(pres\_train\_data)  
pres\_train\_data<-pres\_train\_data[complete,]  
pres\_train\_data<-cbind(pres\_train\_data,1)

Now we want to extract predictors for the absence data, remove rows with NA values and chop it down to the size of our presence training data, and combine these into one data frame with column names (pa is the last column of 0,1 which indicates presence or absence)

abs\_train\_data<-extract(predictors,abs\_train)  
#remove rows with NA values  
complete<-complete.cases(abs\_train\_data)  
abs\_train\_data<-abs\_train\_data[complete,]  
#and select a number of rows equal to the presence training data  
abs\_train\_data<-abs\_train\_data[1:nrow(pres\_train\_data),]  
#and add a column of zeros to the end.  
abs\_train\_data<-cbind(abs\_train\_data,0)  
#put the two matrices together and name the colmns  
train\_data<-rbind(pres\_train\_data,abs\_train\_data)  
colnames(train\_data)<-c(names(predictors),"pa")  
train\_data<-as.data.frame(train\_data)  
colnames(train\_data)

## [1] "Mean\_durnal\_temp\_range" "Isothermality" "Mean\_temp\_warmiest\_q"   
## [4] "Ann\_.precip" "Precip\_Seasonality" "Precip\_warmest\_q"   
## [7] "Precip\_coldest\_q" "Hghest\_weekly\_rad" "Rad\_wettest\_q"   
## [10] "Rad\_driest\_q" "Elev" "human\_impact"   
## [13] "pa"

# Training the GAM and making predictions.

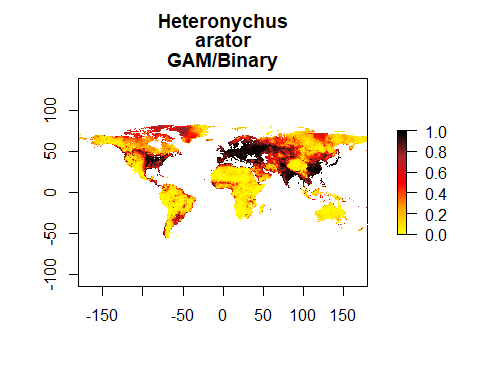
**This is a pain in the neck because all of the layers have to be specified. I recommend printing the column names in the console colnames(train\_data) and then copying them and formatting them**

gam <- gam(pa ~ Mean\_durnal\_temp\_range + Isothermality + Mean\_temp\_warmiest\_q + Ann\_.precip + Precip\_Seasonality + Precip\_warmest\_q + Precip\_coldest\_q + Hghest\_weekly\_rad +   
Rad\_wettest\_q + Rad\_driest\_q + Elev + human\_impact,  
 family = binomial(link = "logit"), data=train\_data)  
summary(gam)

##   
## Family: binomial   
## Link function: logit   
##   
## Formula:  
## pa ~ Mean\_durnal\_temp\_range + Isothermality + Mean\_temp\_warmiest\_q +   
## Ann\_.precip + Precip\_Seasonality + Precip\_warmest\_q + Precip\_coldest\_q +   
## Hghest\_weekly\_rad + Rad\_wettest\_q + Rad\_driest\_q + Elev +   
## human\_impact  
##   
## Parametric coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.9950114 3.0005540 -0.332 0.74018   
## Mean\_durnal\_temp\_range -0.4301029 0.1507211 -2.854 0.00432 \*\*   
## Isothermality -7.7203239 3.2520465 -2.374 0.01760 \*   
## Mean\_temp\_warmiest\_q 0.1654402 0.1330186 1.244 0.21360   
## Ann\_.precip 0.0024733 0.0018643 1.327 0.18463   
## Precip\_Seasonality -0.2816647 1.1120377 -0.253 0.80005   
## Precip\_warmest\_q -0.0049270 0.0039715 -1.241 0.21475   
## Precip\_coldest\_q -0.0020935 0.0034663 -0.604 0.54586   
## Hghest\_weekly\_rad 0.0258742 0.0135693 1.907 0.05654 .   
## Rad\_wettest\_q -0.0047030 0.0103737 -0.453 0.65029   
## Rad\_driest\_q -0.0151267 0.0091479 -1.654 0.09821 .   
## Elev 0.0006875 0.0006437 1.068 0.28554   
## human\_impact 0.1427848 0.0192568 7.415 1.22e-13 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
##   
## R-sq.(adj) = 0.5 Deviance explained = 45.3%  
## UBRE = -0.16723 Scale est. = 1 n = 346

Let’s make some predictions and export them to a file

GAMpreds <- predict(predictors, gam, type = 'response')  
writeRaster(GAMpreds, filename = paste0(genus,"\_",species,"\_GAM.tif"), overwrite=TRUE)  
plot(GAMpreds, main=c(genus,species,'GAM/Binary'),col=warm(100), zlim=c(0,1))  
points(pres\_test, col='white', cex =.4, pch=3)



## MaxEnt

We need many more background points for MaxEnt and BRT than we needed for GAM. Let’s go ahead and generate those.

samp1 <- spsample(pol, 10000, type='random', iter=25)

## Warning in proj4string(obj): CRS object has comment, which is lost in output

#and get the cell numbers from the raster stack (right to left, up to down)  
cells <- cellFromXY(predictors, samp1)  
#and transform each of those to the center of its cell.  
background\_train <- xyFromCell(predictors, cells)  
#You'll get a warning saying that your CRS object has lost a comment. This is unimportant and can be ignored.

Let’s go ahead and set a locations for java **This will obviously be specialized for your computer. Try to find the ‘home’ folder in java and specify the path below**

Sys.setenv(JAVA\_HOME='/Library/Java/JavaVirtualMachines/jdk-11.0.1.jdk/Contents/Home')

First we let the program know to start up maxent using the command maxent. After that, all we need to do is to make a model oject (me\_model), from the raster data and the presence training data.

maxent()

## Loading required namespace: rJava

## java.home option:

## JAVA\_HOME environment variable: /Library/Java/JavaVirtualMachines/jdk-11.0.1.jdk/Contents/Home

## Warning in fun(libname, pkgname): Java home setting is INVALID, it will be ignored.  
## Please do NOT set it unless you want to override system settings.

## This is MaxEnt version 3.4.3

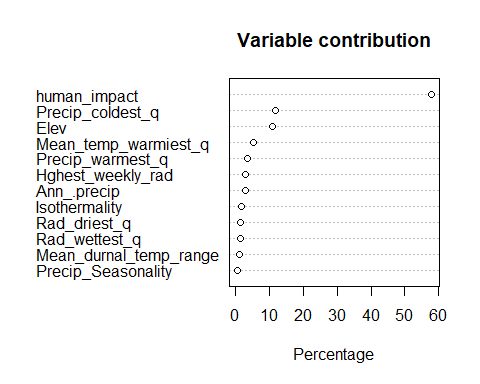
me\_model <- maxent(predictors, pres\_train, a=background\_train)

## Warning in .local(x, p, ...): 19 (14.62%) of the presence points have NA  
## predictor values

## Warning in .local(x, p, ...): 3606 (36.06%) of the presence points have NA  
## predictor values

## This is MaxEnt version 3.4.3

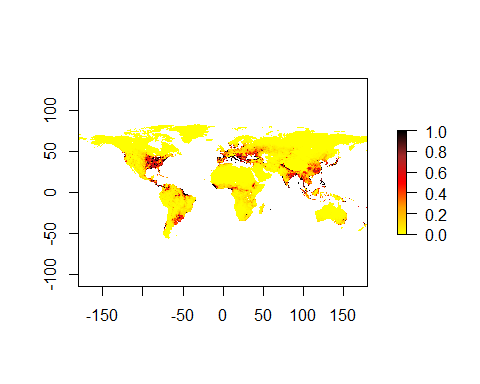
#and plot the models most important layers  
par(mfrow=c(1,1))  
plot(me\_model)

 Let’s go ahead and make some predictions

MEpreds<-predict(predictors, me\_model, type='response')

## This is MaxEnt version 3.4.3

writeRaster(MEpreds, filename=paste0(genus,"\_",species,"\_ME.tif"),overwrite=TRUE)  
#and plot  
plot(MEpreds, col=warm(100), zlim=c(0,1))



## Boosted regression trees

We need to prepare data for BRT in much the same way that we did for GAM, with the exception that we will need a lot more background data. We can use the 10,000 points that we already generated for ME

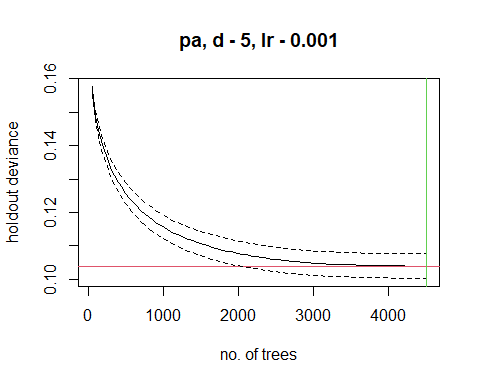
#let's get the data from our predictors  
bg\_train\_data<-extract(predictors,background\_train)  
#and bind a column of 0 to the end of it  
bg\_train\_data<-cbind(bg\_train\_data,0)  
#and convert it to a data frame  
bg\_train\_data<-as.data.frame(bg\_train\_data)  
#and then combine it withe the presence training data  
pres\_train\_data<-as.data.frame(pres\_train\_data)  
BRT\_data<-rbind(pres\_train\_data, bg\_train\_data)  
colnames(BRT\_data)<-c(names(predictors),"pa")

And now we can train a model using the first X columns of train\_data and the 6th as the response. Let’s start with tree complexity of 5 and learning rate of 0.001

sdm.tc5.lr001 <- gbm.step(data=BRT\_data, gbm.x = 1:nlayers(predictors), gbm.y = ncol(BRT\_data), family = "bernoulli", tree.complexity = 5, learning.rate = 0.001, bag.fraction = 0.5)

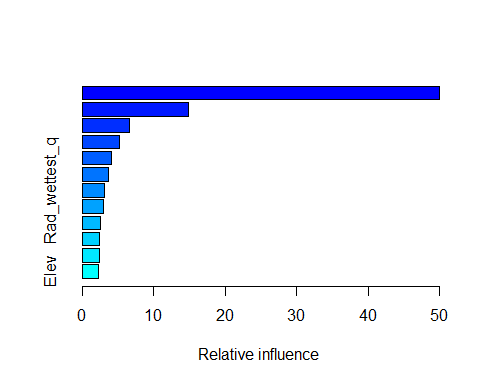
##   
##   
## GBM STEP - version 2.9   
##   
## Performing cross-validation optimisation of a boosted regression tree model   
## for pa and using a family of bernoulli   
## Using 10173 observations and 12 predictors   
## creating 10 initial models of 50 trees   
##   
## folds are stratified by prevalence   
## total mean deviance = 0.1723   
## tolerance is fixed at 2e-04   
## ntrees resid. dev.   
## 50 0.1565   
## now adding trees...   
## 100 0.1487   
## 150 0.1434   
## 200 0.1393   
## 250 0.1361   
## 300 0.1334   
## 350 0.1311   
## 400 0.1291   
## 450 0.1273   
## 500 0.1257   
## 550 0.1243   
## 600 0.123   
## 650 0.1218   
## 700 0.1207   
## 750 0.1197   
## 800 0.1187   
## 850 0.1179   
## 900 0.1171   
## 950 0.1164   
## 1000 0.1157   
## 1050 0.115   
## 1100 0.1144   
## 1150 0.1138   
## 1200 0.1133   
## 1250 0.1128   
## 1300 0.1123   
## 1350 0.1119   
## 1400 0.1115   
## 1450 0.1111   
## 1500 0.1107   
## 1550 0.1103   
## 1600 0.1099   
## 1650 0.1096   
## 1700 0.1093   
## 1750 0.109   
## 1800 0.1087   
## 1850 0.1084   
## 1900 0.1081   
## 1950 0.1079   
## 2000 0.1077   
## 2050 0.1075   
## 2100 0.1073   
## 2150 0.1071   
## 2200 0.1069   
## 2250 0.1067   
## 2300 0.1065   
## 2350 0.1064   
## 2400 0.1062   
## 2450 0.106   
## 2500 0.1059   
## 2550 0.1057   
## 2600 0.1056   
## 2650 0.1055   
## 2700 0.1054   
## 2750 0.1053   
## 2800 0.1051   
## 2850 0.105   
## 2900 0.105   
## 2950 0.1049   
## 3000 0.1048   
## 3050 0.1047   
## 3100 0.1047   
## 3150 0.1046   
## 3200 0.1045   
## 3250 0.1045   
## 3300 0.1045   
## 3350 0.1044   
## 3400 0.1044   
## 3450 0.1043   
## 3500 0.1043   
## 3550 0.1042   
## 3600 0.1042   
## 3650 0.1042   
## 3700 0.1042   
## 3750 0.1041   
## 3800 0.1041   
## 3850 0.1041   
## 3900 0.1041   
## 3950 0.104   
## 4000 0.104   
## 4050 0.104   
## 4100 0.104   
## 4150 0.104   
## 4200 0.104   
## 4250 0.1039   
## 4300 0.1039   
## 4350 0.1039   
## 4400 0.1039   
## 4450 0.1039   
## 4500 0.1039

## fitting final gbm model with a fixed number of 4500 trees for pa



##   
## mean total deviance = 0.172   
## mean residual deviance = 0.082   
##   
## estimated cv deviance = 0.104 ; se = 0.004   
##   
## training data correlation = 0.632   
## cv correlation = 0.516 ; se = 0.028   
##   
## training data AUC score = 0.961   
## cv AUC score = 0.929 ; se = 0.011   
##   
## elapsed time - 0.15 minutes

summary(sdm.tc5.lr001)



## var rel.inf  
## human\_impact human\_impact 50.032049  
## Isothermality Isothermality 14.824760  
## Precip\_coldest\_q Precip\_coldest\_q 6.664412  
## Mean\_durnal\_temp\_range Mean\_durnal\_temp\_range 5.218680  
## Ann\_.precip Ann\_.precip 4.032572  
## Precip\_Seasonality Precip\_Seasonality 3.598973  
## Rad\_wettest\_q Rad\_wettest\_q 3.053134  
## Rad\_driest\_q Rad\_driest\_q 3.030228  
## Mean\_temp\_warmiest\_q Mean\_temp\_warmiest\_q 2.522830  
## Precip\_warmest\_q Precip\_warmest\_q 2.452162  
## Hghest\_weekly\_rad Hghest\_weekly\_rad 2.363357  
## Elev Elev 2.206843

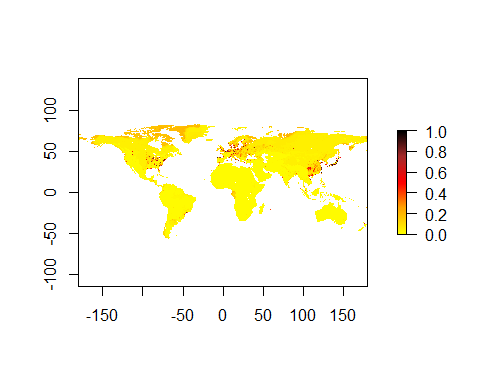
**Note: you may want to try different combinations! If your trees are converging too slowly, raise the tree complexity by 1 or two, and back the learning rate down. On the other hand of your holdout deviance drops very quickly and slowly starts to rise, you are overfitting. Drop the tree complexity and raise the learning rate.**

Let’s make predictions and save them

BRTpreds<-predict(predictors, sdm.tc5.lr001, type='response')

## Using 4500 trees...  
##   
## Using 4500 trees...  
##   
## Using 4500 trees...  
##   
## Using 4500 trees...

writeRaster(BRTpreds, filename=paste0(genus,"\_", species,"\_BRT.tif"), overwrite=TRUE)  
#and plot  
plot(BRTpreds, col=warm(100), zlim=c(0,1))



## Evaluation

We want to generate the following metrics for each of the three models: AUC, COR, maximum Kappa, TRS, and it wouldn’t kill us to have a Boyce graph either.

#Absence Testing Data First we’ll use the pres\_test data to generate absence test data. This time we want about the same number of points for both. To do that, we’ll generate 4x the number of absence points as presence points and chop it to size.

pres\_test\_SPDF<-SpatialPoints(pres\_test)  
data("wrld\_simpl")  
crs(pres\_test\_SPDF) <- crs(wrld\_simpl)  
#now we are going to make circles of about a degree (110000 meters at the equator). I'm working in a relatively small area, but if your data are widespread, you can increase this by changing d.  
x <- circles(pres\_test\_SPDF, d=dist, lonlat=TRUE)  
#and convert those into polygons  
pol <- polygons(x)  
#and draw a number of samples from that...because   
samp1 <- spsample(pol, 4\*length(pres\_test), type='random', iter=25)

## Warning in proj4string(obj): CRS object has comment, which is lost in output

#and get the cell numbers from the raster stack (right to left, up to down)  
cells <- cellFromXY(predictors, samp1)  
#and transform each of those to the center of its cell.  
abs\_test <- xyFromCell(predictors, cells)  
#You'll get a warning saying that your CRS object has lost a comment. This is unimportant and can be ignored.

GAM evaluation

p<-extract(GAMpreds,pres\_test)  
a<-extract(GAMpreds,abs\_test)  
#And let's get rid of nasty NA values and shrink a to the size of p  
p<-p[!is.na(p)]  
a<-a[!is.na(a)]  
a<-a[1:length(p)]  
#Let's look at the shape of these data  
#lets weld all the data together  
all\_vals<-c(p,a)  
e<-evaluate(p=p,a=a)  
AUC\_GAM<-e@auc  
COR\_GAM<-e@cor  
pa<-c(replicate(length(p),1),replicate(length(a),0))  
kappaGAM<-ecospat.max.kappa(all\_vals,pa)  
TSS\_GAM<-ecospat.max.tss(all\_vals,pa)  
print(paste('Max kappa: ', kappaGAM[2] ))

## [1] "Max kappa: 0.541666666666667"

print(paste('TSS:', TSS\_GAM[[2]]))

## [1] "TSS: 0.541666666666667"

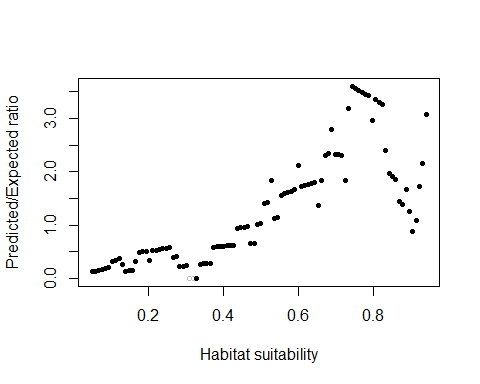
e

## class : ModelEvaluation   
## n presences : 48   
## n absences : 48   
## AUC : 0.8261719   
## cor : 0.5645992   
## max TPR+TNR at : 0.6396539

And let’s go ahead and estimate the Boyce Index

ecospat.boyce(fit=GAMpreds,pres\_test,nclass=0,PEplot = TRUE)

## Warning in if (class(obs) == "data.frame" | class(obs) == "matrix") {: the  
## condition has length > 1 and only the first element will be used



## $F.ratio  
## [1] 0.1353592 0.1373206 0.1490358 0.1643517 0.1809988 0.1966661 0.3191136  
## [8] 0.3430200 0.3670164 0.2593745 0.1373206 0.1447757 0.1516714 0.3143531  
## [15] 0.4864661 0.4959910 0.5035416 0.3411525 0.5207928 0.5306337 0.5428593  
## [22] 0.5548549 0.5683864 0.5831079 0.3999587 0.4134794 0.2145937 0.2239394  
## [29] 0.2328193 0.0000000 0.0000000 0.0000000 0.2649388 0.2710831 0.2783005  
## [36] 0.2846211 0.5806130 0.5912711 0.5988363 0.6041208 0.6092889 0.6136910  
## [43] 0.6175746 0.9390563 0.9469403 0.9516293 0.9655453 0.6526732 0.6588610  
## [50] 1.0038358 1.0240972 1.4006983 1.4305803 1.8266824 1.1176597 1.1426443  
## [57] 1.5549280 1.5843068 1.6071784 1.6370642 1.6652362 2.1167423 1.7135111  
## [64] 1.7396369 1.7604017 1.7793182 1.7947046 1.3629338 1.8279637 2.3075779  
## [71] 2.3326450 2.7986047 2.3249830 2.3145647 2.3003563 1.8301518 3.1755238  
## [78] 3.5984204 3.5502353 3.5181028 3.4866792 3.4411633 3.4210155 2.9637675  
## [85] 3.3445578 3.3042009 3.2535744 2.3989992 1.9569282 1.9132406 1.8628267  
## [92] 1.4453667 1.3897908 1.6613277 1.2571885 0.8851495 1.0898721 1.7153054  
## [99] 2.1582970 3.0642526  
##   
## $Spearman.cor  
## [1] 0.875  
##   
## $HS  
## [1] 0.05008429 0.05908344 0.06808259 0.07708175 0.08608090 0.09508006  
## [7] 0.10407921 0.11307837 0.12207752 0.13107667 0.14007583 0.14907498  
## [13] 0.15807414 0.16707329 0.17607244 0.18507160 0.19407075 0.20306991  
## [19] 0.21206906 0.22106822 0.23006737 0.23906652 0.24806568 0.25706483  
## [25] 0.26606399 0.27506314 0.28406230 0.29306145 0.30206060 0.31105976  
## [31] 0.32005891 0.32905807 0.33805722 0.34705637 0.35605553 0.36505468  
## [37] 0.37405384 0.38305299 0.39205215 0.40105130 0.41005045 0.41904961  
## [43] 0.42804876 0.43704792 0.44604707 0.45504623 0.46404538 0.47304453  
## [49] 0.48204369 0.49104284 0.50004200 0.50904115 0.51804030 0.52703946  
## [55] 0.53603861 0.54503777 0.55403692 0.56303608 0.57203523 0.58103438  
## [61] 0.59003354 0.59903269 0.60803185 0.61703100 0.62603016 0.63502931  
## [67] 0.64402846 0.65302762 0.66202677 0.67102593 0.68002508 0.68902424  
## [73] 0.69802339 0.70702254 0.71602170 0.72502085 0.73402001 0.74301916  
## [79] 0.75201831 0.76101747 0.77001662 0.77901578 0.78801493 0.79701409  
## [85] 0.80601324 0.81501239 0.82401155 0.83301070 0.84200986 0.85100901  
## [91] 0.86000817 0.86900732 0.87800647 0.88700563 0.89600478 0.90500394  
## [97] 0.91400309 0.92300224 0.93200140 0.94100055

ME Evaluation

p<-extract(MEpreds,pres\_test)  
a<-extract(MEpreds,abs\_test)  
#And let's get rid of nasty NA values and shrink a to the size of p  
p<-p[!is.na(p)]  
a<-a[!is.na(a)]  
a<-a[1:length(p)]  
#Let's look at the shape of these data  
#lets weld all the data together  
all\_vals<-c(p,a)  
e<-evaluate(p=p,a=a)  
AUC\_ME<-e@auc  
COR\_ME<-e@cor  
pa<-c(replicate(length(p),1),replicate(length(a),0))  
kappaME<-ecospat.max.kappa(all\_vals,pa)  
TSS\_ME<-ecospat.max.tss(all\_vals,pa)  
print(paste('Max kappa: ', kappaME[2] ))

## [1] "Max kappa: 0.625"

print(paste('TSS:', TSS\_ME[[2]]))

## [1] "TSS: 0.625"

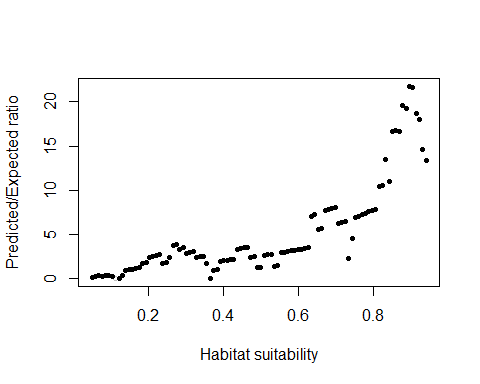
e

## class : ModelEvaluation   
## n presences : 48   
## n absences : 48   
## AUC : 0.8621962   
## cor : 0.6460174   
## max TPR+TNR at : 0.4213143

And let’s go ahead and estimate the Boyce Index

ecospat.boyce(fit=MEpreds,pres\_test,nclass=0,PEplot = TRUE)

## Warning in if (class(obs) == "data.frame" | class(obs) == "matrix") {: the  
## condition has length > 1 and only the first element will be used



## $F.ratio  
## [1] 0.07624213 0.25433396 0.32361664 0.25940676 0.30229131 0.34786932  
## [7] 0.19792213 0.00000000 0.00000000 0.27449649 0.90332569 0.98216842  
## [13] 1.06177502 1.14260634 1.22521811 1.72433328 1.81214750 2.38514359  
## [19] 2.50050511 2.61357896 2.71314034 1.69118770 1.75503541 2.41711980  
## [25] 3.75141210 3.85201825 3.32352221 3.44866656 2.86276778 2.94002085  
## [31] 3.05090231 2.36072255 2.44373439 2.52239690 1.74148627 0.00000000  
## [37] 0.93052149 0.95836325 1.96467340 2.00661761 2.06534876 2.11311459  
## [43] 2.16538879 3.32275176 3.39254921 3.45594223 3.50417123 2.37555756  
## [49] 2.42761010 1.24207283 1.27707053 2.60787271 2.67291756 2.71854346  
## [55] 1.38823442 1.42127124 2.92371655 2.96530032 3.04806388 3.10795943  
## [61] 3.17157138 3.23692769 3.31842464 3.37706268 3.46324866 7.06846437  
## [67] 7.19712307 5.48826563 5.64589676 7.67652976 7.79525243 7.93688619  
## [73] 8.07522277 6.19158549 6.32710115 6.49555306 2.20600545 4.49150659  
## [79] 6.89940814 7.00477479 7.19975910 7.37257801 7.54561285 7.68381706  
## [85] 7.77762808 10.43149086 10.57824384 13.45554603 10.96253951 16.68304994  
## [91] 16.79709423 16.65074975 19.60139543 19.29048649 21.75866476 21.61001899  
## [97] 18.67125245 17.98514772 14.56411896 13.38799338  
##   
## $Spearman.cor  
## [1] 0.869  
##   
## $HS  
## [1] 0.05000033 0.05900033 0.06800033 0.07700032 0.08600032 0.09500032  
## [7] 0.10400031 0.11300031 0.12200031 0.13100030 0.14000030 0.14900030  
## [13] 0.15800030 0.16700029 0.17600029 0.18500029 0.19400028 0.20300028  
## [19] 0.21200028 0.22100027 0.23000027 0.23900027 0.24800026 0.25700026  
## [25] 0.26600026 0.27500025 0.28400025 0.29300025 0.30200024 0.31100024  
## [31] 0.32000024 0.32900024 0.33800023 0.34700023 0.35600023 0.36500022  
## [37] 0.37400022 0.38300022 0.39200021 0.40100021 0.41000021 0.41900020  
## [43] 0.42800020 0.43700020 0.44600019 0.45500019 0.46400019 0.47300018  
## [49] 0.48200018 0.49100018 0.50000018 0.50900017 0.51800017 0.52700017  
## [55] 0.53600016 0.54500016 0.55400016 0.56300015 0.57200015 0.58100015  
## [61] 0.59000014 0.59900014 0.60800014 0.61700013 0.62600013 0.63500013  
## [67] 0.64400012 0.65300012 0.66200012 0.67100012 0.68000011 0.68900011  
## [73] 0.69800011 0.70700010 0.71600010 0.72500010 0.73400009 0.74300009  
## [79] 0.75200009 0.76100008 0.77000008 0.77900008 0.78800007 0.79700007  
## [85] 0.80600007 0.81500006 0.82400006 0.83300006 0.84200006 0.85100005  
## [91] 0.86000005 0.86900005 0.87800004 0.88700004 0.89600004 0.90500003  
## [97] 0.91400003 0.92300003 0.93200002 0.94100002

BRT Evaluation

p<-extract(BRTpreds,pres\_test)  
a<-extract(BRTpreds,abs\_test)  
#And let's get rid of nasty NA values and shrink a to the size of p  
p<-p[!is.na(p)]  
a<-a[!is.na(a)]  
a<-a[1:length(p)]  
#Let's look at the shape of these data  
#lets weld all the data together  
all\_vals<-c(p,a)  
e<-evaluate(p=p,a=a)  
AUC\_BRT<-e@auc  
COR\_BRT<-e@cor  
pa<-c(replicate(length(p),1),replicate(length(a),0))  
kappaBRT<-ecospat.max.kappa(all\_vals,pa)  
TSS\_BRT<-ecospat.max.tss(all\_vals,pa)  
print(paste('Max kappa: ', kappaBRT[2] ))

## [1] "Max kappa: 0.625"

print(paste('TSS:', TSS\_BRT[[2]]))

## [1] "TSS: 0.625"

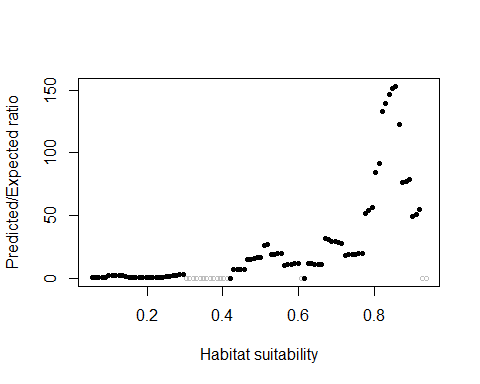
e

## class : ModelEvaluation   
## n presences : 48   
## n absences : 48   
## AUC : 0.874783   
## cor : 0.5009211   
## max TPR+TNR at : 0.03988713

And let’s go ahead and estimate the Boyce Index

ecospat.boyce(fit=BRTpreds,pres\_test,nclass=0,PEplot = TRUE)

## Warning in if (class(obs) == "data.frame" | class(obs) == "matrix") {: the  
## condition has length > 1 and only the first element will be used

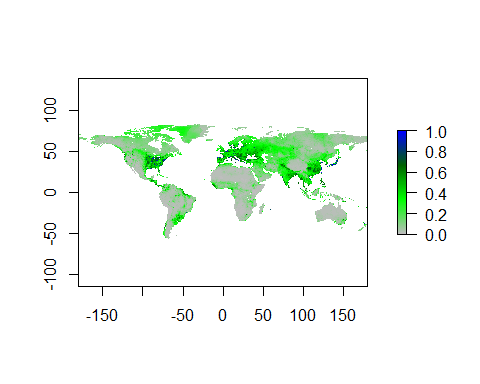


## $F.ratio  
## [1] 0.6277887 0.7434087 0.7566515 0.7890294 1.0606129 2.1735293  
## [7] 2.5845372 2.2342451 2.0152184 2.5222305 1.8315404 0.8731965  
## [13] 0.8870096 0.6913412 0.4749639 0.4841894 0.4961305 0.7606343  
## [19] 0.7339702 0.9644084 0.5774116 0.8969662 1.3696627 1.8414672  
## [25] 2.1005342 2.3701563 2.7301944 3.0521204 0.0000000 0.0000000  
## [31] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000  
## [37] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000  
## [43] 6.7810859 6.9327789 7.1178252 7.3223604 14.8919693 15.2281757  
## [49] 15.6223656 16.3811662 16.8382031 26.1203105 27.3671034 18.7060625  
## [55] 19.1752781 19.6181631 20.2236620 10.5782438 11.0683555 11.2640632  
## [61] 11.6178484 11.8704103 0.0000000 0.0000000 11.8704103 11.6532686  
## [67] 11.4211318 11.1328314 10.7973788 31.4446517 30.7971790 29.7067780  
## [73] 29.1282379 28.2433900 27.6976240 18.6300834 18.7826639 18.6908172  
## [79] 19.0478677 19.4848196 20.1703014 51.7455610 54.0377772 56.7102687  
## [85] 84.4034851 91.3170811 132.8715681 139.4989821 146.6344799 151.4770981  
## [91] 153.0950110 122.8039232 76.3436506 77.4784887 79.1907205 49.5683703  
## [97] 50.6634006 54.6038873 0.0000000 0.0000000  
##   
## $Spearman.cor  
## [1] 0.884  
##   
## $HS  
## [1] 0.05479037 0.06370357 0.07261677 0.08152998 0.09044318 0.09935639  
## [7] 0.10826959 0.11718279 0.12609600 0.13500920 0.14392241 0.15283561  
## [13] 0.16174881 0.17066202 0.17957522 0.18848842 0.19740163 0.20631483  
## [19] 0.21522804 0.22414124 0.23305444 0.24196765 0.25088085 0.25979406  
## [25] 0.26870726 0.27762046 0.28653367 0.29544687 0.30436008 0.31327328  
## [31] 0.32218648 0.33109969 0.34001289 0.34892610 0.35783930 0.36675250  
## [37] 0.37566571 0.38457891 0.39349211 0.40240532 0.41131852 0.42023173  
## [43] 0.42914493 0.43805813 0.44697134 0.45588454 0.46479775 0.47371095  
## [49] 0.48262415 0.49153736 0.50045056 0.50936377 0.51827697 0.52719017  
## [55] 0.53610338 0.54501658 0.55392978 0.56284299 0.57175619 0.58066940  
## [61] 0.58958260 0.59849580 0.60740901 0.61632221 0.62523542 0.63414862  
## [67] 0.64306182 0.65197503 0.66088823 0.66980144 0.67871464 0.68762784  
## [73] 0.69654105 0.70545425 0.71436745 0.72328066 0.73219386 0.74110707  
## [79] 0.75002027 0.75893347 0.76784668 0.77675988 0.78567309 0.79458629  
## [85] 0.80349949 0.81241270 0.82132590 0.83023911 0.83915231 0.84806551  
## [91] 0.85697872 0.86589192 0.87480512 0.88371833 0.89263153 0.90154474  
## [97] 0.91045794 0.91937114 0.92828435 0.93719755

# Making the ensemble and evaluation

The ensemble is simply the average of GAM, ME, and BRT predictions weighted by AUC.

ENSpreds<-(GAMpreds\*AUC\_GAM+MEpreds\*AUC\_ME+BRTpreds\*AUC\_BRT)/(AUC\_GAM+AUC\_ME+AUC\_BRT)  
writeRaster(ENSpreds, filename=paste0(genus,"\_",species,"\_ENS.tif"), overwrite=TRUE)  
plot(ENSpreds, col=cool(100), zlim=c(0,1))

 And let’s evaluate

p<-extract(ENSpreds,pres\_test)  
a<-extract(ENSpreds,abs\_test)  
#And let's get rid of nasty NA values and shrink a to the size of p  
p<-p[!is.na(p)]  
a<-a[!is.na(a)]  
a<-a[1:length(p)]  
#Let's look at the shape of these data  
#lets weld all the data together  
all\_vals<-c(p,a)  
e<-evaluate(p=p,a=a)  
AUC\_ENS<-e@auc  
COR\_ENS<-e@cor  
pa<-c(replicate(length(p),1),replicate(length(a),0))  
kappaENS<-ecospat.max.kappa(all\_vals,pa)  
TSS\_ENS<-ecospat.max.tss(all\_vals,pa)  
print(paste('Max kappa: ', kappaENS[2] ))

## [1] "Max kappa: 0.604166666666667"

print(paste('TSS:', TSS\_ENS[[2]]))

## [1] "TSS: 0.604166666666667"

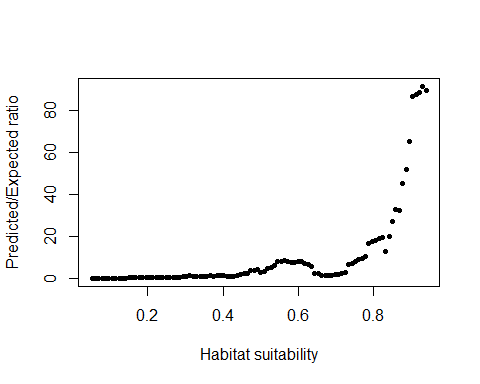
e

## class : ModelEvaluation   
## n presences : 48   
## n absences : 48   
## AUC : 0.8526476   
## cor : 0.626013   
## max TPR+TNR at : 0.4311356

and the Boyce ploy for the ensemble

ecospat.boyce(fit=ENSpreds,pres\_test,nclass=0,PEplot = TRUE)

## Warning in if (class(obs) == "data.frame" | class(obs) == "matrix") {: the  
## condition has length > 1 and only the first element will be used



## $F.ratio  
## [1] 0.1251535 0.1381389 0.2037359 0.2259939 0.2485007 0.1349879  
## [7] 0.2175124 0.2320372 0.2458580 0.1741923 0.2809255 0.4052953  
## [13] 0.4328734 0.3443040 0.3628318 0.3801239 0.3989475 0.4176434  
## [19] 0.4348323 0.6006320 0.7689244 0.6270303 0.4759266 0.4814416  
## [25] 0.4865005 0.6532961 0.6650321 1.0263276 1.0621520 1.2879515  
## [31] 1.1519036 1.0126118 1.0724268 1.1360257 1.2188109 1.3139170  
## [37] 1.1367634 1.5324642 1.6331704 1.3854668 1.0997234 1.1603740  
## [43] 1.2316225 1.3068590 1.8500833 2.4454716 2.5795951 3.8284706  
## [49] 4.0394401 4.2377760 3.1721856 3.3287321 4.9033424 5.1282721  
## [55] 6.1254361 8.0036409 8.3699389 8.7426169 8.2075192 7.6115608  
## [61] 7.8870717 8.0908917 8.3546931 7.4321958 6.5612148 5.7334082  
## [67] 2.4138125 2.5647095 1.3665614 1.4670952 1.5727357 1.7371332  
## [73] 1.9478200 2.1946060 2.5357842 2.9485257 6.6243884 7.4678061  
## [79] 8.2852719 9.0646769 9.7465502 10.5831254 16.7561856 17.7139284  
## [85] 18.2495751 18.9325531 19.4682790 13.1575632 20.1762164 27.1083128  
## [91] 33.1028185 32.6503882 45.6066558 51.9742383 65.3007763 86.9711857  
## [97] 87.8424350 88.6785320 91.6781133 89.9358143  
##   
## $Spearman.cor  
## [1] 0.947  
##   
## $HS  
## [1] 0.05174174 0.06071073 0.06967972 0.07864871 0.08761771 0.09658670  
## [7] 0.10555569 0.11452468 0.12349368 0.13246267 0.14143166 0.15040065  
## [13] 0.15936965 0.16833864 0.17730763 0.18627662 0.19524561 0.20421461  
## [19] 0.21318360 0.22215259 0.23112158 0.24009058 0.24905957 0.25802856  
## [25] 0.26699755 0.27596655 0.28493554 0.29390453 0.30287352 0.31184252  
## [31] 0.32081151 0.32978050 0.33874949 0.34771849 0.35668748 0.36565647  
## [37] 0.37462546 0.38359446 0.39256345 0.40153244 0.41050143 0.41947042  
## [43] 0.42843942 0.43740841 0.44637740 0.45534639 0.46431539 0.47328438  
## [49] 0.48225337 0.49122236 0.50019136 0.50916035 0.51812934 0.52709833  
## [55] 0.53606733 0.54503632 0.55400531 0.56297430 0.57194330 0.58091229  
## [61] 0.58988128 0.59885027 0.60781927 0.61678826 0.62575725 0.63472624  
## [67] 0.64369523 0.65266423 0.66163322 0.67060221 0.67957120 0.68854020  
## [73] 0.69750919 0.70647818 0.71544717 0.72441617 0.73338516 0.74235415  
## [79] 0.75132314 0.76029214 0.76926113 0.77823012 0.78719911 0.79616811  
## [85] 0.80513710 0.81410609 0.82307508 0.83204408 0.84101307 0.84998206  
## [91] 0.85895105 0.86792004 0.87688904 0.88585803 0.89482702 0.90379601  
## [97] 0.91276501 0.92173400 0.93070299 0.93967198

and finally let’s make a table of evaluation metrics

#Let's go in this order of columns, left to right: AUC, COR, Kappa, TSS  
eGAM<-c(AUC\_GAM,COR\_GAM,kappaGAM[2], TSS\_GAM[[2]])  
eME<-c(AUC\_ME, COR\_ME, kappaME[2],TSS\_ME[[2]])  
eBRT<-c(AUC\_BRT, COR\_BRT, kappaBRT[2],TSS\_BRT[[2]])  
eENS<-c(AUC\_ENS, COR\_ENS, kappaENS[2], TSS\_ENS[[2]])  
all\_evals<-rbind(eGAM,eME,eBRT,eENS)  
colnames(all\_evals)<-c("AUC", "COR","MaxKappa","TSS")  
rownames(all\_evals)<-c("GAM","MaxEnt", "BRT", "Ensemble")  
write.csv(all\_evals, file=paste0(genus,"\_",species, '\_eval.csv'))