Diabrotica speciosa

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

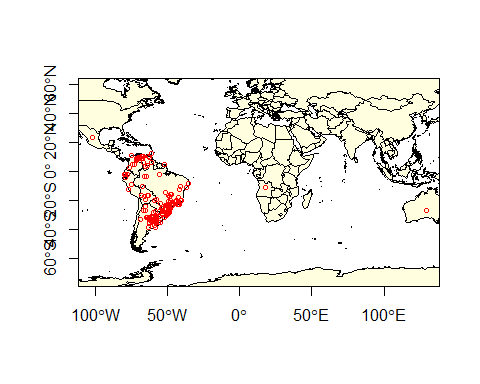
Their are many places in this code where you will need to save files with filenames including the genus and species. We’ll save these as strings to automate the creation of file names. Enter your genus name and specific epithet in the quotes below.

genus<-"Diabrotica"  
species<-"speciosa"

## 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 = "Diabrotica speciosa.CLEAN.csv")  
##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,]

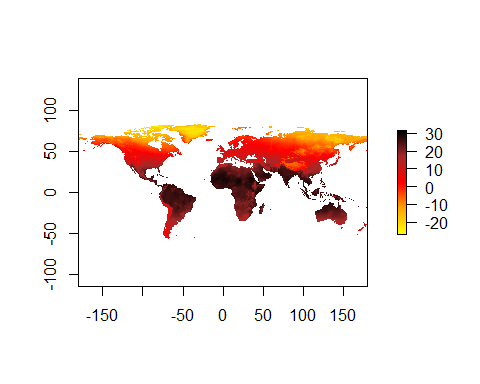
Save pres\_data and test\_data as csv files just in case.

#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"   
## [3] "Mean\_temp\_wettest\_q" "Ann\_.precip"   
## [5] "Precip\_wettest\_week" "Precip\_warmest\_q"   
## [7] "Precip\_coldest\_q" "Ann\_mean\_rad"   
## [9] "Lowest\_weekly\_rad" "Rad\_driest\_q"   
## [11] "Highsets\_weekly\_moisture" "Moisture\_seasonality"   
## [13] "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.   
dist<-spDists(pres\_train\_SPDF,longlat = TRUE)  
#replace the zeros with NA  
dist[dist == 0]<-NA  
#and calculate the mean--this is the average distance between points...the result will be in kilometers, but we need to convert it to meters so we multiply by 1000  
dist<-1000\*mean(dist, na.rm=TRUE)  
#now we are going to make circles using the average distance between points as the diameter.   
x <- circles(pres\_train\_SPDF, d=dist, 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"   
## [3] "Mean\_temp\_wettest\_q" "Ann\_.precip"   
## [5] "Precip\_wettest\_week" "Precip\_warmest\_q"   
## [7] "Precip\_coldest\_q" "Ann\_mean\_rad"   
## [9] "Lowest\_weekly\_rad" "Rad\_driest\_q"   
## [11] "Highsets\_weekly\_moisture" "Moisture\_seasonality"   
## [13] "Elev" "human\_impact"   
## [15] "pa"

# Training the GAM and making predictions.

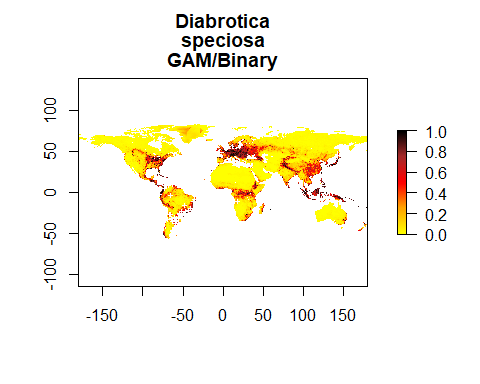
**This is a pain in the neck because all of the layers have to be specified. I recommend printing the column names in the console colnames(train\_data) and then copying them and formatting them**

gam <- gam(pa ~ Mean\_durnal\_temp\_range + Isothermality + Mean\_temp\_wettest\_q + Ann\_.precip + Precip\_wettest\_week + Precip\_warmest\_q + Precip\_coldest\_q +   
Ann\_mean\_rad + Lowest\_weekly\_rad + Rad\_driest\_q + Highsets\_weekly\_moisture + Moisture\_seasonality + 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\_wettest\_q +   
## Ann\_.precip + Precip\_wettest\_week + Precip\_warmest\_q + Precip\_coldest\_q +   
## Ann\_mean\_rad + Lowest\_weekly\_rad + Rad\_driest\_q + Highsets\_weekly\_moisture +   
## Moisture\_seasonality + Elev + human\_impact  
##   
## Parametric coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -5.9755977 3.3713373 -1.772 0.07632 .   
## Mean\_durnal\_temp\_range -0.3149500 0.1315331 -2.394 0.01665 \*   
## Isothermality 5.9914754 2.4313293 2.464 0.01373 \*   
## Mean\_temp\_wettest\_q -0.0159243 0.0590422 -0.270 0.78738   
## Ann\_.precip -0.0003385 0.0011725 -0.289 0.77283   
## Precip\_wettest\_week 0.0359196 0.0398717 0.901 0.36765   
## Precip\_warmest\_q 0.0006724 0.0016751 0.401 0.68812   
## Precip\_coldest\_q -0.0012464 0.0016891 -0.738 0.46056   
## Ann\_mean\_rad 0.0409777 0.0162052 2.529 0.01145 \*   
## Lowest\_weekly\_rad -0.0124083 0.0159462 -0.778 0.43649   
## Rad\_driest\_q -0.0194760 0.0074079 -2.629 0.00856 \*\*   
## Highsets\_weekly\_moisture 1.1481433 1.7951791 0.640 0.52245   
## Moisture\_seasonality -3.4281723 2.1100420 -1.625 0.10423   
## Elev 0.0007824 0.0003703 2.113 0.03458 \*   
## human\_impact 0.1781300 0.0206971 8.607 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
##   
## R-sq.(adj) = 0.609 Deviance explained = 53.6%  
## UBRE = -0.2958 Scale est. = 1 n = 492

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)  
#The backround data has too many NA values so to fix this:  
#First I get the predictor data associated with my points  
background\_train\_data<-extract(predictors,background\_train)  
#Then I remove all the points that don't have data  
complete<-complete.cases(background\_train\_data)  
background\_train<-background\_train[complete,]

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.

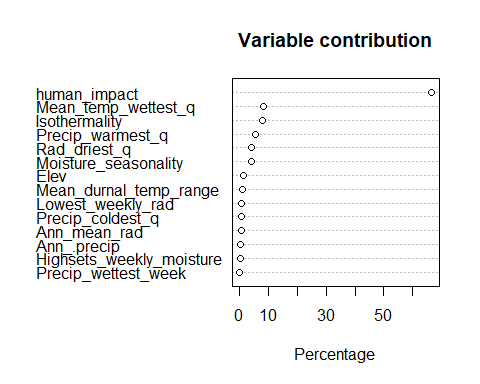
## This is MaxEnt version 3.4.3

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

## Warning in .local(x, p, ...): 5 (3.25%) of the presence points have NA predictor  
## values

## This is MaxEnt version 3.4.3

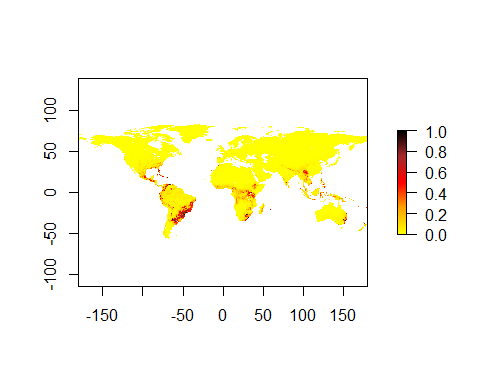
#and plot the models most important layers  
par(mfrow=c(1,1))  
plot(me\_model)

 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

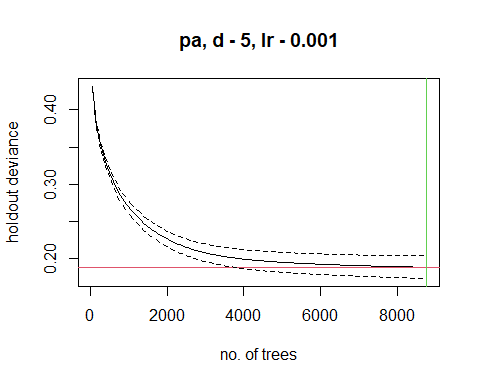
#let's get the data from our predictors  
bg\_train\_data<-extract(predictors,background\_train)  
#and bind a column of 0 to the end of it  
bg\_train\_data<-cbind(bg\_train\_data,0)  
#and convert it to a data frame  
bg\_train\_data<-as.data.frame(bg\_train\_data)  
#and then combine it withe the 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)

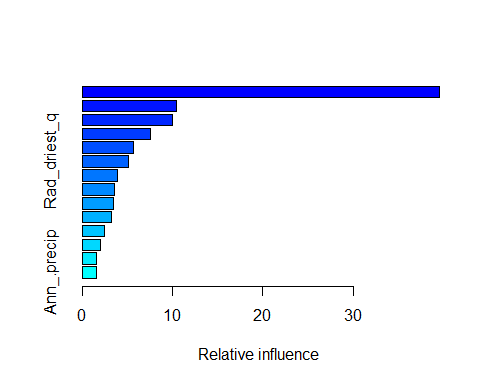
##   
##   
## GBM STEP - version 2.9   
##   
## Performing cross-validation optimisation of a boosted regression tree model   
## for pa and using a family of bernoulli   
## Using 3982 observations and 14 predictors   
## creating 10 initial models of 50 trees   
##   
## folds are stratified by prevalence   
## total mean deviance = 0.4637   
## tolerance is fixed at 5e-04   
## ntrees resid. dev.   
## 50 0.4301   
## now adding trees...   
## 100 0.4063   
## 150 0.3878   
## 200 0.3727   
## 250 0.36   
## 300 0.349   
## 350 0.3394   
## 400 0.3308   
## 450 0.3231   
## 500 0.3161   
## 550 0.3097   
## 600 0.3037   
## 650 0.2983   
## 700 0.2932   
## 750 0.2885   
## 800 0.2843   
## 850 0.2801   
## 900 0.2763   
## 950 0.2727   
## 1000 0.2693   
## 1050 0.266   
## 1100 0.2631   
## 1150 0.2602   
## 1200 0.2574   
## 1250 0.2548   
## 1300 0.2523   
## 1350 0.2499   
## 1400 0.2476   
## 1450 0.2455   
## 1500 0.2434   
## 1550 0.2413   
## 1600 0.2394   
## 1650 0.2375   
## 1700 0.2357   
## 1750 0.2339   
## 1800 0.2323   
## 1850 0.2307   
## 1900 0.2291   
## 1950 0.2277   
## 2000 0.2263   
## 2050 0.225   
## 2100 0.2237   
## 2150 0.2226   
## 2200 0.2214   
## 2250 0.2203   
## 2300 0.2192   
## 2350 0.2181   
## 2400 0.2171   
## 2450 0.2161   
## 2500 0.2152   
## 2550 0.2142   
## 2600 0.2134   
## 2650 0.2126   
## 2700 0.2119   
## 2750 0.211   
## 2800 0.2103   
## 2850 0.2096   
## 2900 0.2089   
## 2950 0.2082   
## 3000 0.2076   
## 3050 0.2069   
## 3100 0.2064   
## 3150 0.2058   
## 3200 0.2053   
## 3250 0.2048   
## 3300 0.2043   
## 3350 0.2039   
## 3400 0.2035   
## 3450 0.203   
## 3500 0.2026   
## 3550 0.2022   
## 3600 0.2018   
## 3650 0.2014   
## 3700 0.2011   
## 3750 0.2007   
## 3800 0.2004   
## 3850 0.2   
## 3900 0.1997   
## 3950 0.1994   
## 4000 0.1991   
## 4050 0.1988   
## 4100 0.1985   
## 4150 0.1982   
## 4200 0.198   
## 4250 0.1977   
## 4300 0.1975   
## 4350 0.1972   
## 4400 0.197   
## 4450 0.1968   
## 4500 0.1966   
## 4550 0.1964   
## 4600 0.1962   
## 4650 0.196   
## 4700 0.1959   
## 4750 0.1956   
## 4800 0.1954   
## 4850 0.1953   
## 4900 0.1951   
## 4950 0.1949   
## 5000 0.1947   
## 5050 0.1946   
## 5100 0.1944   
## 5150 0.1943   
## 5200 0.1942   
## 5250 0.194   
## 5300 0.1938   
## 5350 0.1936   
## 5400 0.1935   
## 5450 0.1934   
## 5500 0.1932   
## 5550 0.1931   
## 5600 0.193   
## 5650 0.1929   
## 5700 0.1928   
## 5750 0.1927   
## 5800 0.1926   
## 5850 0.1925   
## 5900 0.1924   
## 5950 0.1923   
## 6000 0.1922   
## 6050 0.1921   
## 6100 0.192   
## 6150 0.1919   
## 6200 0.1918   
## 6250 0.1917   
## 6300 0.1916   
## 6350 0.1915   
## 6400 0.1914   
## 6450 0.1913   
## 6500 0.1912   
## 6550 0.1911   
## 6600 0.191   
## 6650 0.1909   
## 6700 0.1908   
## 6750 0.1908   
## 6800 0.1907   
## 6850 0.1906   
## 6900 0.1905   
## 6950 0.1904   
## 7000 0.1903   
## 7050 0.1902   
## 7100 0.1901   
## 7150 0.19   
## 7200 0.1899   
## 7250 0.1899   
## 7300 0.1898   
## 7350 0.1897   
## 7400 0.1897   
## 7450 0.1896   
## 7500 0.1895   
## 7550 0.1895   
## 7600 0.1894   
## 7650 0.1893   
## 7700 0.1893   
## 7750 0.1892   
## 7800 0.1891   
## 7850 0.189   
## 7900 0.189   
## 7950 0.1889   
## 8000 0.1889   
## 8050 0.1888   
## 8100 0.1887   
## 8150 0.1887   
## 8200 0.1886   
## 8250 0.1886   
## 8300 0.1885   
## 8350 0.1885   
## 8400 0.1885   
## 8450 0.1884   
## 8500 0.1884   
## 8550 0.1884   
## 8600 0.1883   
## 8650 0.1882   
## 8700 0.1882   
## 8750 0.1882

## fitting final gbm model with a fixed number of 8750 trees for pa



##   
## mean total deviance = 0.464   
## mean residual deviance = 0.111   
##   
## estimated cv deviance = 0.188 ; se = 0.015   
##   
## training data correlation = 0.882   
## cv correlation = 0.766 ; se = 0.024   
##   
## training data AUC score = 0.987   
## cv AUC score = 0.95 ; se = 0.009   
##   
## elapsed time - 0.13 minutes

summary(sdm.tc5.lr001)



## var rel.inf  
## human\_impact human\_impact 39.583149  
## Mean\_temp\_wettest\_q Mean\_temp\_wettest\_q 10.394609  
## Moisture\_seasonality Moisture\_seasonality 10.022322  
## Isothermality Isothermality 7.518284  
## Elev Elev 5.699630  
## Rad\_driest\_q Rad\_driest\_q 5.078158  
## Mean\_durnal\_temp\_range Mean\_durnal\_temp\_range 3.865033  
## Lowest\_weekly\_rad Lowest\_weekly\_rad 3.516831  
## Ann\_mean\_rad Ann\_mean\_rad 3.433216  
## Highsets\_weekly\_moisture Highsets\_weekly\_moisture 3.196070  
## Precip\_warmest\_q Precip\_warmest\_q 2.455274  
## Precip\_wettest\_week Precip\_wettest\_week 2.025339  
## Precip\_coldest\_q Precip\_coldest\_q 1.619871  
## Ann\_.precip Ann\_.precip 1.592214

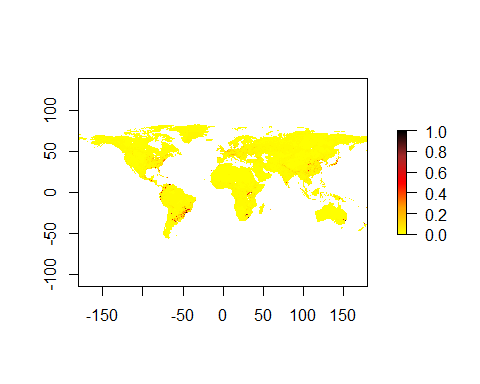
**Note: you may want to try different combinations! If your trees are converging too slowly, raise the tree complexity by 1 or two, and back the learning rate down. On the other hand of your holdout deviance drops very quickly and slowly starts to rise, you are overfitting. Drop the tree complexity and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

Let’s make predictions and save them

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

## Using 8750 trees...  
##   
## Using 8750 trees...  
##   
## Using 8750 trees...  
##   
## Using 8750 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: 0.721311475409836"

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

## [1] "TSS: 0.721311475409836"

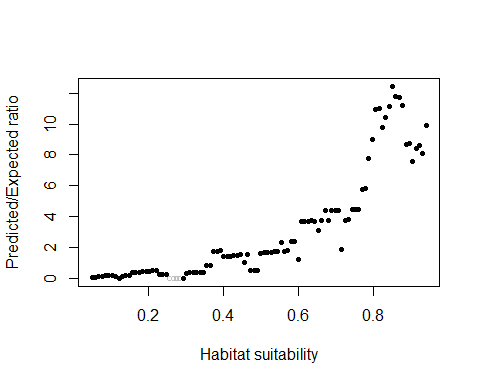
e

## class : ModelEvaluation   
## n presences : 61   
## n absences : 61   
## AUC : 0.9111798   
## cor : 0.7379597   
## max TPR+TNR at : 0.5075893

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] 0.04701390 0.07653791 0.11036458 0.13856736 0.16334306 0.18762957  
## [7] 0.21215986 0.11949966 0.00000000 0.14576260 0.15899023 0.17196406  
## [13] 0.36793434 0.38780313 0.41009348 0.43150973 0.45053146 0.46794133  
## [19] 0.48424055 0.50257785 0.26074491 0.27033136 0.28025542 0.00000000  
## [25] 0.00000000 0.00000000 0.00000000 0.00000000 0.34520609 0.35820642  
## [31] 0.36878439 0.37961176 0.38934238 0.39985795 0.82076205 0.84294587  
## [37] 1.72905588 1.76930295 1.80783140 1.38634823 1.41093072 1.43993460  
## [43] 1.46484114 1.49290578 1.51698213 1.02717693 1.56381925 0.52849032  
## [49] 0.53414950 0.54069602 1.64313705 1.66695214 1.68648548 1.70405304  
## [55] 1.71403382 1.73606511 2.32717860 1.76835261 1.77838243 2.40495309  
## [61] 2.41870136 1.21697827 3.66414076 3.70173489 3.70929424 3.73162989  
## [67] 3.71373995 3.11476664 3.73586436 4.38400405 3.74862564 4.37869389  
## [73] 4.40695366 4.39843750 1.88598953 3.76901027 3.78689342 4.43846373  
## [79] 4.44617051 4.46912725 5.76741104 5.81366575 7.75041382 9.02820167  
## [85] 10.93070025 10.98945448 9.77140452 10.39583333 11.11135953 12.43709691  
## [91] 11.78778625 11.74922398 11.18044657 8.65675487 8.76512221 7.55000342  
## [97] 8.38384331 8.58329599 8.09489470 9.87289414  
##   
## $Spearman.cor  
## [1] 0.962  
##   
## $HS  
## [1] 0.05007343 0.05906909 0.06806476 0.07706042 0.08605609 0.09505175  
## [7] 0.10404742 0.11304308 0.12203875 0.13103441 0.14003007 0.14902574  
## [13] 0.15802140 0.16701707 0.17601273 0.18500840 0.19400406 0.20299973  
## [19] 0.21199539 0.22099106 0.22998672 0.23898239 0.24797805 0.25697372  
## [25] 0.26596938 0.27496505 0.28396071 0.29295638 0.30195204 0.31094771  
## [31] 0.31994337 0.32893904 0.33793470 0.34693036 0.35592603 0.36492169  
## [37] 0.37391736 0.38291302 0.39190869 0.40090435 0.40990002 0.41889568  
## [43] 0.42789135 0.43688701 0.44588268 0.45487834 0.46387401 0.47286967  
## [49] 0.48186534 0.49086100 0.49985667 0.50885233 0.51784800 0.52684366  
## [55] 0.53583932 0.54483499 0.55383065 0.56282632 0.57182198 0.58081765  
## [61] 0.58981331 0.59880898 0.60780464 0.61680031 0.62579597 0.63479164  
## [67] 0.64378730 0.65278297 0.66177863 0.67077430 0.67976996 0.68876563  
## [73] 0.69776129 0.70675696 0.71575262 0.72474829 0.73374395 0.74273961  
## [79] 0.75173528 0.76073094 0.76972661 0.77872227 0.78771794 0.79671360  
## [85] 0.80570927 0.81470493 0.82370060 0.83269626 0.84169193 0.85068759  
## [91] 0.85968326 0.86867892 0.87767459 0.88667025 0.89566592 0.90466158  
## [97] 0.91365725 0.92265291 0.93164857 0.94064424

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: 0.819672131147541"

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

## [1] "TSS: 0.819672131147541"

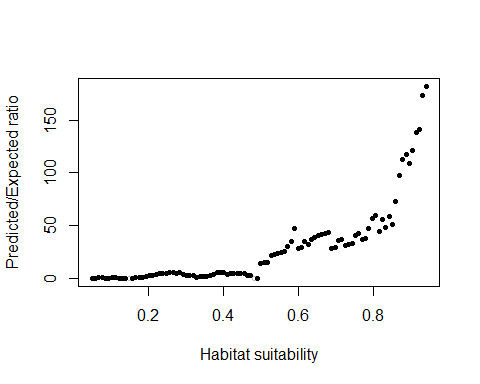
e

## class : ModelEvaluation   
## n presences : 61   
## n absences : 61   
## AUC : 0.9380543   
## cor : 0.7754276   
## max TPR+TNR at : 0.2359768

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] 0.06860555 0.30788831 0.40461713 0.37983622 0.31181408  
## [6] 0.37682553 0.43910677 0.50170188 0.28297442 0.31482756  
## [11] 0.34577713 0.00000000 0.00000000 0.44849457 0.48400029  
## [16] 0.51963784 1.67756967 2.40808647 3.21962786 3.45341879  
## [21] 4.41243822 4.68394874 4.98586725 5.33208700 5.64281601  
## [26] 4.93051850 5.17585264 3.28402042 2.28716458 2.40002221  
## [31] 2.52097093 1.32039122 1.38737061 1.45352339 1.51890679  
## [36] 3.17744526 3.35856207 5.32077652 5.61215507 5.89741239  
## [41] 4.14605962 4.29094880 4.49758261 4.67729422 4.80135699  
## [46] 4.95584603 2.55583151 2.64914944 0.00000000 0.00000000  
## [51] 14.50162083 14.88518354 15.27362778 21.99533599 22.56056122  
## [56] 23.28721817 24.34824435 25.40511317 29.79815443 34.70071640  
## [61] 47.75674297 28.42505493 29.50310550 35.24046663 31.77609553  
## [66] 37.29912374 39.17136085 40.81013945 41.73158051 42.93047066  
## [71] 43.67801617 28.44874028 29.22530376 36.02988201 36.91358619  
## [76] 31.35457589 32.56409959 33.15438539 40.73912413 42.27583266  
## [81] 37.01214692 38.00554654 46.98990856 56.69277560 59.54938832  
## [86] 44.51968180 55.50652002 48.45077953 58.98733203 50.68869082  
## [91] 73.16067708 97.42728884 112.69937420 117.05708333 108.68436210  
## [96] 121.42850968 138.75300826 140.77564687 172.92523674 182.02656499  
##   
## $Spearman.cor  
## [1] 0.939  
##   
## $HS  
## [1] 0.05000024 0.05900011 0.06799998 0.07699985 0.08599972 0.09499959  
## [7] 0.10399946 0.11299933 0.12199920 0.13099907 0.13999894 0.14899881  
## [13] 0.15799868 0.16699856 0.17599843 0.18499830 0.19399817 0.20299804  
## [19] 0.21199791 0.22099778 0.22999765 0.23899752 0.24799739 0.25699726  
## [25] 0.26599713 0.27499700 0.28399687 0.29299675 0.30199662 0.31099649  
## [31] 0.31999636 0.32899623 0.33799610 0.34699597 0.35599584 0.36499571  
## [37] 0.37399558 0.38299545 0.39199532 0.40099519 0.40999506 0.41899494  
## [43] 0.42799481 0.43699468 0.44599455 0.45499442 0.46399429 0.47299416  
## [49] 0.48199403 0.49099390 0.49999377 0.50899364 0.51799351 0.52699338  
## [55] 0.53599325 0.54499313 0.55399300 0.56299287 0.57199274 0.58099261  
## [61] 0.58999248 0.59899235 0.60799222 0.61699209 0.62599196 0.63499183  
## [67] 0.64399170 0.65299157 0.66199144 0.67099132 0.67999119 0.68899106  
## [73] 0.69799093 0.70699080 0.71599067 0.72499054 0.73399041 0.74299028  
## [79] 0.75199015 0.76099002 0.76998989 0.77898976 0.78798963 0.79698951  
## [85] 0.80598938 0.81498925 0.82398912 0.83298899 0.84198886 0.85098873  
## [91] 0.85998860 0.86898847 0.87798834 0.88698821 0.89598808 0.90498795  
## [97] 0.91398782 0.92298770 0.93198757 0.94098744

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: 0.80327868852459"

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

## [1] "TSS: 0.80327868852459"

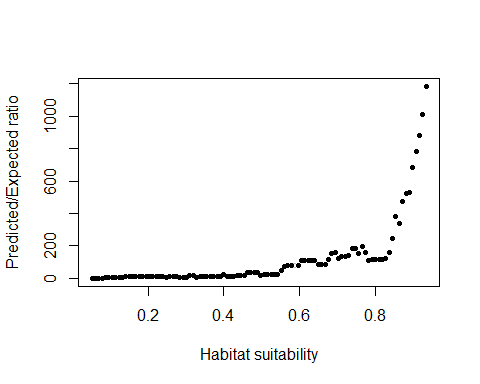
e

## class : ModelEvaluation   
## n presences : 61   
## n absences : 61   
## AUC : 0.9297232   
## cor : 0.6390722   
## max TPR+TNR at : 0.03929156

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

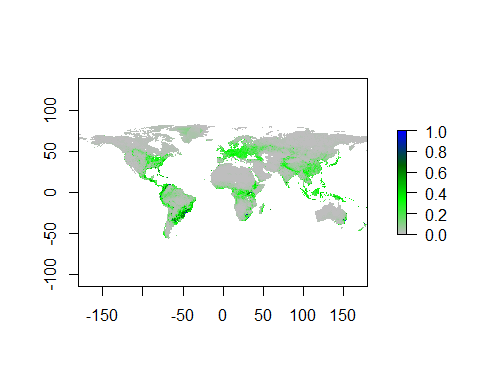


## $F.ratio  
## [1] 0.239059 0.960399 1.591050 2.363834 3.775629 4.706242  
## [7] 4.897634 6.752774 7.459737 6.956641 9.399659 12.184045  
## [13] 10.791039 12.174122 11.651335 13.136082 12.251300 13.683418  
## [19] 12.009961 10.102740 14.724161 12.126079 8.845623 9.579139  
## [25] 10.445308 11.394265 6.165226 6.805644 7.261606 15.747590  
## [31] 16.611696 8.598708 9.399659 9.831222 10.365149 11.084951  
## [37] 11.768474 12.227411 12.632059 26.523508 13.653626 14.046850  
## [43] 14.880138 15.375274 16.533486 17.453839 35.760820 36.887736  
## [49] 37.841730 38.932511 20.754802 21.570716 22.744252 23.663831  
## [55] 24.185348 24.252158 48.370696 76.563499 78.153839 78.620429  
## [61] 81.289641 81.289641 111.837978 109.059394 112.554888 110.085031  
## [67] 109.741016 84.687600 85.791022 88.086434 119.446003 155.110976  
## [73] 157.900742 125.418304 133.019413 133.525190 139.908865 182.901693  
## [79] 185.216904 153.349891 197.731560 158.185248 112.075931 115.516859  
## [85] 116.539132 119.717472 116.539132 120.264127 156.772879 248.470224  
## [91] 381.707880 340.942961 473.392616 522.011318 530.615900 683.417103  
## [97] 779.226146 877.928125 1012.993990 1183.023715  
##   
## $Spearman.cor  
## [1] 0.956  
##   
## $HS  
## [1] 0.05146714 0.06039216 0.06931718 0.07824220 0.08716722 0.09609224  
## [7] 0.10501726 0.11394228 0.12286730 0.13179232 0.14071734 0.14964236  
## [13] 0.15856738 0.16749240 0.17641742 0.18534244 0.19426745 0.20319247  
## [19] 0.21211749 0.22104251 0.22996753 0.23889255 0.24781757 0.25674259  
## [25] 0.26566761 0.27459263 0.28351765 0.29244267 0.30136769 0.31029271  
## [31] 0.31921773 0.32814275 0.33706777 0.34599279 0.35491781 0.36384283  
## [37] 0.37276785 0.38169287 0.39061789 0.39954291 0.40846793 0.41739295  
## [43] 0.42631797 0.43524299 0.44416801 0.45309303 0.46201805 0.47094307  
## [49] 0.47986809 0.48879311 0.49771813 0.50664314 0.51556816 0.52449318  
## [55] 0.53341820 0.54234322 0.55126824 0.56019326 0.56911828 0.57804330  
## [61] 0.58696832 0.59589334 0.60481836 0.61374338 0.62266840 0.63159342  
## [67] 0.64051844 0.64944346 0.65836848 0.66729350 0.67621852 0.68514354  
## [73] 0.69406856 0.70299358 0.71191860 0.72084362 0.72976864 0.73869366  
## [79] 0.74761868 0.75654370 0.76546872 0.77439374 0.78331876 0.79224378  
## [85] 0.80116880 0.81009382 0.81901883 0.82794385 0.83686887 0.84579389  
## [91] 0.85471891 0.86364393 0.87256895 0.88149397 0.89041899 0.89934401  
## [97] 0.90826903 0.91719405 0.92611907 0.93504409

# 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: 0.770491803278689"

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

## [1] "TSS: 0.770491803278689"

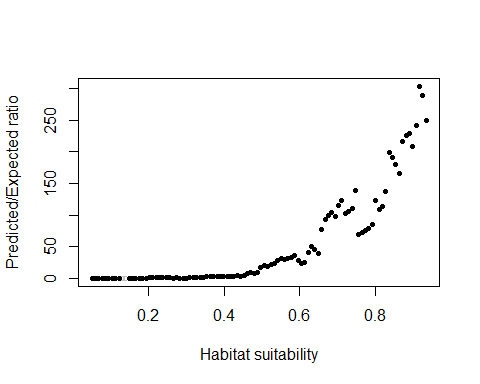
e

## class : ModelEvaluation   
## n presences : 61   
## n absences : 61   
## AUC : 0.9321419   
## cor : 0.7605258   
## max TPR+TNR at : 0.3494005

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] 0.07987936 0.17145862 0.23236831 0.28704063 0.25733618  
## [6] 0.29475842 0.32796857 0.24079544 0.26182971 0.00000000  
## [11] 0.00000000 0.00000000 0.17130973 0.36183824 0.37800182  
## [16] 0.59082605 0.61395006 0.84823973 0.87191193 0.89513714  
## [21] 0.91838289 0.93913634 0.95318183 0.72768536 0.49153358  
## [26] 0.75522864 0.51903877 0.53824298 0.56438438 0.89508390  
## [31] 1.27185270 1.36313660 1.47247788 2.01470563 2.23130210  
## [36] 2.51252969 2.26986782 3.21020961 3.63050254 3.24018500  
## [41] 2.69468424 3.01555344 3.39667833 3.79671958 2.79817729  
## [46] 4.63939471 8.50375944 9.27846253 8.06919233 8.66661525  
## [51] 16.57808707 20.58448124 19.51570935 21.22063838 22.87122023  
## [56] 28.52731519 30.99481465 29.40429127 31.48307825 33.60031096  
## [61] 36.55857748 28.10269286 24.06931117 25.65166180 40.61348304  
## [66] 49.76110830 45.21518240 39.83339950 77.54026619 92.90244709  
## [71] 98.75456974 103.52926002 97.66814740 115.06266383 122.61565992  
## [76] 102.53175182 105.93401207 110.25784929 139.57521860 69.67683532  
## [81] 72.31697900 75.68345905 79.23539034 84.90600822 122.90993750  
## [86] 108.60966495 113.77038337 137.17626953 199.52911932 190.85394022  
## [91] 180.08782051 166.09451014 217.07014080 226.39980301 229.02472826  
## [96] 209.03050595 242.37279525 303.68582940 288.79214638 249.05762411  
##   
## $Spearman.cor  
## [1] 0.985  
##   
## $HS  
## [1] 0.05076355 0.05970021 0.06863686 0.07757352 0.08651018 0.09544683  
## [7] 0.10438349 0.11332015 0.12225681 0.13119346 0.14013012 0.14906678  
## [13] 0.15800344 0.16694009 0.17587675 0.18481341 0.19375007 0.20268672  
## [19] 0.21162338 0.22056004 0.22949669 0.23843335 0.24737001 0.25630667  
## [25] 0.26524332 0.27417998 0.28311664 0.29205330 0.30098995 0.30992661  
## [31] 0.31886327 0.32779993 0.33673658 0.34567324 0.35460990 0.36354655  
## [37] 0.37248321 0.38141987 0.39035653 0.39929318 0.40822984 0.41716650  
## [43] 0.42610316 0.43503981 0.44397647 0.45291313 0.46184979 0.47078644  
## [49] 0.47972310 0.48865976 0.49759641 0.50653307 0.51546973 0.52440639  
## [55] 0.53334304 0.54227970 0.55121636 0.56015302 0.56908967 0.57802633  
## [61] 0.58696299 0.59589964 0.60483630 0.61377296 0.62270962 0.63164627  
## [67] 0.64058293 0.64951959 0.65845625 0.66739290 0.67632956 0.68526622  
## [73] 0.69420288 0.70313953 0.71207619 0.72101285 0.72994950 0.73888616  
## [79] 0.74782282 0.75675948 0.76569613 0.77463279 0.78356945 0.79250611  
## [85] 0.80144276 0.81037942 0.81931608 0.82825274 0.83718939 0.84612605  
## [91] 0.85506271 0.86399936 0.87293602 0.88187268 0.89080934 0.89974599  
## [97] 0.90868265 0.91761931 0.92655597 0.93549262

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'))