**Predictive invasion dynamics in American populations of the Eurasian Collared Dove *Streptopelia decaocto***

# --------------------Fitting Minimum Volume Ellipsoids as Niche Models----------------------

# Original code by Jorge Soberón, August 2015

## Minimum Volume Ellipsoids (MVE) can be used as niche models, mostly when one is interested in fitting a niche not too constrained by the details of the observed data. To do this, we must (1) calculate ellipsoids, and (2) calculate, for all pixels in a region of interest, the environmental distance of each pixel to a centroid of the ellipse.

## Ellipsoids can be calculated in many dimensions, and are characterized by a centroid and by a matrix (symmetric) that describes the directions of the axes and their lengths.

# Define the Mahalanobis function that calculates the distance from a point (‘p’) to an ellipse of centroid (‘m’) and matrix (‘s’). The parameters then are: *p*, the test point, *m*, the centroid of the ellipse (of a distribution), and *s,* which is the INVERSE of the covariance matrix of the ellipse.

maja=function(p,m,s)((p-m)%\*%s%\*%t(p-m))^0.5

# Load required libraries

library(dismo)

library(ellipsoid)

library(foreign)

library(MASS)

library(raster)

# --------------- DATA PREPARATION ---------------

# Set working directory

setwd("<path to chosen working directory>");

# load environmental rasters (ASCII format)

rList = list.files(path = " xxxxxx ", pattern = "\*.asc$", full.names = T)

rList

>[1] " xxxxxx.asc"

>[2] " xxxxxx.asc"

>[3] " xxxxxx.asc"

>[4] " xxxxxx.asc"

>[5] " xxxxxx.asc"

# Rasterize and name each environmental variable to be used in analyses

comp1 = raster(rList [1])

comp2 = raster(rList [2])

comp3 = raster(rList [3])

comp4 = raster(rList [4])

comp5 = raster(rList [5])

# Stack the environmental layers

layers = stack(comp1, comp2, comp3, comp4, comp5)

# Read in the .csv file containing the 'training points" (species occurrence data to be used in model calibration) and check formatting. The .csv should contain 3 columns: species ID, longitude, and latitude. Multiple data subsets can be merged into a single file using ‘rbind’; if this is done, be sure to add a column to flag/identify the original subset number.

refined2 = read.csv(“xxxxxx.csv”)

head(refined)

> Species Set Longitude Latitude

>1 Dove 1 37.97220 37.0341

>2 Dove 1 19.96090 50.0297

>3 Dove 1 35.33580 36.9941

>4 Dove 1 -1.79000 38.9800

>5 Dove 1 -0.23527 49.2645

>6 Dove 1 27.17720 38.4553

# If “refined2” is a "list", convert to matrix

refined = as.matrix(refined2[ , 2:4])

# Extract the values of the principal components to the occurrence data

vars = extract(layers, refined[ , 2:3]) # “2:3” denotes the columns with the longitude-latitude data

crds\_vrs = cbind(refined, vars)

# check that the new matrix contains SpeciesID, longitude, latitude, and extracted environmental data for each point

head(crds\_vrs)

> Set Longitude Latitude Comp1 Comp2 Comp3 Comp4

>[1,] 1 37.97220 37.0341 2.0284936 -1.9795928 -0.07688452 0.4622065

>[2,] 1 19.96090 50.0297 0.4175917 1.0267162 -0.76874685 -0.3727416

>[3,] 1 35.33580 36.9941 2.4763570 -0.5523745 -0.01509122 -0.1737162

>[4,] 1 -1.79000 38.9800 1.4031270 -0.5879212 -1.33913255 -0.3164902

# Index by species ID. This is *only* necessary if there are point observation data for multiple species in the .csv.

i1=which(refined[ , 1] == 1)

i2=which(refined[ , 1] == 2)

i3=which(refined[ , 1 ]== 3)

i4=which(refined[ , 1] == 4)

i5=which(refined[ , 1] == 5)

#------------- GENERATE MVEs -------------

# Define the function to calculate the number of points to be included in MVE calculation. NOTE: “nD” designates the species (i.e., ‘i1’) and ‘level’ designates the model threshold.

NDquantil <- function(nD, level) return(round(nD \* level/1))

# Specify the species, assign a threshold, and calculate the number of points to include in analyses. In the code below, the threshold is 0.95, or E = 5%. What you’re doing here calculating the number of occurrence points for species ‘il’ excluding the most extreme 5%, which will then be used to generate the minimum volume ellipsoids to be used in model calibration.

n1 <- NDquantil(length(i1), 0.95)

n2 <- NDquantil(length(i2), 0.95)

n3 <- NDquantil(length(i3), 0.95)

n4 <- NDquantil(length(i4), 0.95)

n5 <- NDquantil(length(i5), 0.95)

# Generate ellipsoids. Ellipsoids are represented by a (1) centroid and (2) matrix of covariance. NOTE: The values of the highlighted column range below will depend on the number of environmental variables to be extracted. The range below (4:7) indicate that ellipsoids are being generated based on 4 variables.

# Subset 1

mve1 = cov.mve(crds\_vrs[i1, 4:7], quantile.used = n1)

mu1 = matrix(mve1$center, nrow = 1) # Create a matrix of the covariances

s1 = mve1$cov

# Subset 2

mve2 = cov.mve(crds\_vrs[i2, 4:7], quantile.used = n2)

mu2 = matrix(mve2$center, nrow=1) # Create a matrix of the covariances

s2 = mve2$cov

# Subset 3

mve3 = cov.mve(crds\_vrs[i3, 4:7], quantile.used = n3)

mu3 = matrix(mve3$center, nrow=1) # Create a matrix of the covariances

s3 = mve3$cov

# Subset 4

mve4 = cov.mve(crds\_vrs[i4, 4:7], quantile.used = n4)

mu4 = matrix(mve4$center, nrow=1) # Create a matrix of the covariances

s4 = mve4$cov

# Subset 5

mve5 = cov.mve(crds\_vrs[i5, 4:7], quantile.used = n5)

mu5 = matrix(mve5$center, nrow=1) # Create a matrix of the covariances

s5 = mve5$cov

# Take the inverse of the covariances

invs1 = solve(s1)

invs2 = solve(s2)

invs3 = solve(s3)

invs4 = solve(s4)

invs5 = solve(s5)

# --------------- MODEL CALIBRATION ---------------

# To proceed with model calibration, you must first generate a regular grid (a.k.a., “fishnet”) of the training/calibration region. The grid must be set to match the spatial resolution of the environmental data; be sure to add XY coordinates to labels. The resulting .dbf will be used to then apply the defined ellipsoids to every point in raster.

# Read in regular grid .dbf file for the calibration region

randT=read.dbf("xxxxxx.dbf")

# check that the grid read in properly (e.g., the longitude and latitude are there)

head(randT)

dim(randT)

>[1] 1774698 3

# Extract environmental data from the raster stack to the calibration region grid. NOTE: there will be A LOT will be NAs.

vrsT = extract(layers, randT[ , 2:3])

# check that everything read in and extracted properly

head(vrsT)

> Comp1 Comp2 Comp3 Comp4

>[1,] NA NA NA NA

>[2,] NA NA NA NA

>[3,] NA NA NA NA

>[4,] NA NA NA NA

>[5,] NA NA NA NA

>[6,] NA NA NA NA

# Just out of curiosity, let us see how many are NAs

vrsTsna = na.omit(vrsT)

dim(vrsTsna)

>[1] 904600 4

dim(vrsTsna) / TotalNumberPixelsInGrid

>[1] 0.5097205 # About 50% of pixels are NAs

# Create the matrices that will contain the distance of environment to centroid. The matrix size will be [Total number of pixels in grid x 1].

dT1 = matrix(0, ncol = 1, nrow = 1774698)

dT2 = matrix(0, ncol = 1, nrow = 1774698)

dT3 = matrix(0, ncol = 1, nrow = 1774698)

dT4 = matrix(0, ncol = 1, nrow = 1774698)

dT5 = matrix(0, ncol = 1, nrow = 1774698)

# Calculate environmental distance of each ellipsoid from the centroid

for(i in 1:1774698)dT1[i, 1] = maja(vrsT[i, ], mu1, invs1)

for(i in 1:1774698)dT2[i, 1] = maja(vrsT[i, ], mu2, invs2)

for(i in 1:1774698)dT3[i, 1] = maja(vrsT[i, ], mu3, invs3)

for(i in 1:1774698)dT4[i, 1] = maja(vrsT[i, ], mu4, invs4)

for(i in 1:1774698)dT5[i, 1] = maja(vrsT[i, ], mu5, invs5)

# The resulting table should have the following: longitude, latitude, one column for each environmental variable, and a column for dT1.

Mcalib1 = cbind(randT, vrsT, dT1)

Mcalib2 = cbind(randT, vrsT, dT2)

Mcalib3 = cbind(randT, vrsT, dT3)

Mcalib4 = cbind(randT, vrsT, dT4)

Mcalib5 = cbind(randT, vrsT, dT5)

# Check that it worked.

head(Mcalib1);

> ID XCOORD YCOORD Comp1 Comp2 Comp3 Comp4 dTraw1 dTraw2 dTraw3 dTraw4 dTraw5

>1 0 -32.04333 72.87667 NA NA NA NA NA NA NA NA NA

>2 1 -31.96003 72.87667 NA NA NA NA NA NA NA NA NA

# You’re more than halfway through your application of MVEs to ENM approaches! Save model calibration in .csv to the path or your choosing then continue on to the final step of the process—model projection.

write.csv(Mcalib1, "<YourAwesomeMVEmodelCalibrationFilepathHere.csv>") write.csv(Mcalib2, "<YourAwesomeMVEmodelCalibrationFilepathHere.csv>")

write.csv(Mcalib3, "<YourAwesomeMVEmodelCalibrationFilepathHere.csv>")

write.csv(Mcalib4, "<YourAwesomeMVEmodelCalibrationFilepathHere.csv>")

write.csv(Mcalib5, "<YourAwesomeMVEmodelCalibrationFilepathHere.csv>")

# --------------- MODEL PROJE­CTION ---------------

# Read in environmental rasters for projection region

c1W = raster("xxxxxx.asc")

c2W = raster("xxxxxx.asc")

c3W = raster("xxxxxx.asc")

c4W = raster("xxxxxx.asc")

# Stack the environmental layers

world = stack(c1W, c2W, c3­­­­­­W, c4W)

# Convert from raster to points

pntsw = rasterToPoints(world)

dim(pntsw)

>[1] 1737302 6

head(pntsw)

> x y Comp1 Comp2 Comp3 Comp4

>[1,] -112.1250 78.62501 -4.348080 -0.3390261 0.8083666 -1.995361

>[2,] -112.0417 78.62501 -4.348080 -0.3390261 0.8083666 -1.995361

# Convert to matrix

NAOWvrs = as.matrix(pntsw[ , 3:6])

# Create the matrix that will contain the distance of environment to centroid. The matrix size will be equivalent to the dimension of ‘randT\_fullproj’ (e.g., TotalNumberPixelsInGrid x 1).

dNAOWP1 = matrix(0, ncol = 1, nrow = 1737302)

dNAOWP2 = matrix(0, ncol = 1, nrow = 1737302)

dNAOWP3 = matrix(0, ncol = 1, nrow = 1737302)

dNAOWP4 = matrix(0, ncol = 1, nrow = 1737302)

dNAOWP5 = matrix(0, ncol = 1, nrow = 1737302)

# Calculate environmental distance of each ellipsoid from the centroid. Remember, ‘*mu#*’ are the matrices of covariances and ‘*invs#*’ are the inverses of the covariances calculated in model calibration.

for(i in 1:1737302)dNAOWP1[i,1] = maja(NAOWvrs[i, ], mu1, invs1)

for(i in 1:1737302)dNAOWP2[i,1] = maja(NAOWvrs[i, ], mu2, invs2)

for(i in 1:1737302)dNAOWP3[i,1] = maja(NAOWvrs[i, ], mu3, invs3)

for(i in 1:1737302)dNAOWP4[i,1] = maja(NAOWvrs[i, ], mu4, invs4)

for(i in 1:1737302)dNAOWP5[i,1] = maja(NAOWvrs[i, ], mu5, invs5)

# Combine all model data into a single file

NAOW = cbind(pntsw, dNAOWP1, dNAOWP2, dNAOWP3, dNAOWP4, dNAOWP5)

colnames(NAOW) = c("long", "lat", "Comp1", "Comp2", "Comp3", "Comp4", "Model1", "Model2", "Model3", "Model4", "Model5")

# Write out as a .csv

write.csv(NAOW," xxxxxx.csv")