Practical 4 Species Distribution Modelling I

MD

20/02/2020

```r
setwd("P:/GEOG70922/Week5")



install.packages(c("dismo","maptools","glmnet","maxnet","raster","sp"))



library(dismo)

files <- list.files(path=paste(system.file(package="dismo"),
                               '/ex', sep=''), pattern='grd', full.names=TRUE )

files




env<-stack(files[1:8])


#use the names() function to label each covariate layer

names(env) <- c("MeanTemp.","AnnualPrecip.","Precip.WettestQuarter","PrecipDriestQuart
er","Max.Temp.","Min.Temp.","Temp.Range","MeanTemp.WettestQuarter")



plot(env)

library(maptools)
 data(wrld_simpl)

#And now plot:
 # first layer of the RasterStack
 plot(env, 1)
 # note the "add=TRUE" argument with plot
 plot(wrld_simpl,add=TRUE)




bradypus <- gbif("Bradypus", "variegatus*",sp=TRUE)



wgs84<-crs.latlong<-crs("+init=epsg:4326")


crs(bradypus)
```

```
plot(env, 1)

plot(wrld_simpl,add=TRUE)

plot(bradypus,add=TRUE,col='red')



View(wrld_simpl@data)



SA<-subset(wrld_simpl,wrld_simpl$SUBREGION==5|wrld_simpl$SUBREGION==13)#subset the wor
ld map to South and Central America

plot(SA)


#Set the bradypus crs to that of the SA object created above.
crs(bradypus)<-crs(SA)

#subset our points to only those inside SA
bradypus<-bradypus[SA, ]

plot(SA)

plot(bradypus, add=TRUE)





set.seed(11)

#create random points on cells of the 'env' object within the extent of bradypus and a
voiding cells containing points from bradypus

back.xy <- randomPoints(env, n=1000,p= bradypus,ext = extent(bradypus))


#Create Spatial Points layer from back.xy

back<-SpatialPoints(back.xy,crs(bradypus))

#make sure crs matches other data

crs(back)<-crs(SA)


#subset to SA object

back<-back[SA,]

plot(SA)
plot(bradypus,add=TRUE)

plot(back,add=TRUE, col='red')




#absence values

eA<-extract(env,back)

#presence values
eP<-extract(env,bradypus)

#create data frames from values
Pres.cov<-data.frame(eP,Pres=1)


Back.cov<-data.frame(eA,Pres=0)


#combine both data sets using the rbind() function (binds datsets with same columns by
stacking one on top of the other - i.e. joining rows together)
all.cov<-rbind(Pres.cov,Back.cov)



head(all.cov)
tail(all.cov)
summary(all.cov)




bc_bradypus <- bioclim(Pres.cov[,c('MeanTemp.', 'Temp.Range', 'AnnualPrecip.')])


bc_bradypus






bioclim.map <- predict(env, bc_bradypus)
plot(bioclim.map, axes = F, box = F, main = "bioclim: MeanTemp.,Temp.Range,AnnualPreci
p.")




bc_bradypus2<-bioclim(Pres.cov[,1:8])
```

```
bioclim.map2 <- predict(env, bc_bradypus2)
plot(bioclim.map2, axes = F, box = F, main = "bioclimAll")


response(bc_bradypus2)


#Specify model
glm.bradypus<-glm(Pres~.,binomial(link='logit'), data=all.cov)


#get summary of outputs - regression coefficients are log(odds) where positive values
 denote a positive relationship between the predictor variable and the liklihood of sp
ecies presence (i.e. the liklihood that Pres =1).

summary(glm.bradypus)






#Use the predict function to estimate probabilities for our study area based on the ra
ster stack object 'env'. Note the "response" argument tells the predict function to re
turn values as probabilities, i.e. from 0 to 1).
glm.map <- predict(env, glm.bradypus, type = "response")

#plot the map
plot(glm.map,main="glmBradypus")




 library(maxnet)
 library(glmnet)
 maxnet.ppm <- maxnet(all.cov$Pres, all.cov[,1:8],maxnet.formula(all.cov$Pres,all.cov
[,1:8],classes="lq")) # runs a presence-only Maxent model selecting feature "linear" a
nd "quadratic" using the "lq" argument.





maxnet.cloglog.map <- predict(env, maxnet.ppm, clamp=F, type="cloglog") # make predict
ions to map and transform values with the clog-log argument (similar to logistic trans
form) to get range 0 to 1.

plot(maxnet.cloglog.map, axes=F, box=F, main="Maxnet-clog")



#For evaluating the first bioclim model we only povide the MeanTemp.,Temp.Range and An
nualPrecip. predictors which are columns 1,2 and 7 in the Pres.cov data frame
```

```
############### model validation

eBC <- evaluate(Pres.cov[,c(1,2,7)],Back.cov,bc_bradypus)

#For model two validation, provide all predictors
eBC2 <- evaluate(Pres.cov,Back.cov,bc_bradypus2)

#inspect both evaluation reults

eBC

eBC2




 plot(eBC,'ROC') #plot the receiver operatoring characteristic curves

 plot(eBC2,'ROC')



#GLM evaluation

#Model with all predictors
eglmAll <- evaluate(all.cov[all.cov$Pres==1,],all.cov[all.cov$Pres==0,], glm.bradypus)


#inspect
eglmAll
plot(eglmAll,'ROC')



#Repeat for Maxnet

 maxeval<-evaluate(p=bradypus, a=back, maxnet.ppm, env)

 maxeval

 plot(maxeval,'ROC')



#Kfold validation approach

set.seed(5) #ensure random selection of cases in each fold is replicable

#set number of folds to use
folds=5

#partiction presence and absence data according to folds using the kfold() function.
```

```r
kfold_pres <- kfold(Pres.cov, folds)
kfold_back <- kfold(Back.cov, folds)


#create an empty list to hold our results (remember there will be five sets)
bc_K<-list()
par(mfrow=c(2,3))
# for loop to iterate over folds
for (i in 1:folds) {
  train <- Pres.cov[kfold_pres!= i,]
  test <- Pres.cov[kfold_pres == i,]
  backTrain<-Back.cov[kfold_back!=i,]
  backTest<-Back.cov[kfold_back==i,]
  dataTrain<-rbind(train,backTrain)
  dataTest<-rbind(test,backTest)
  bc_bradypus_K<-bioclim(train[,1:8])#for bioclimm we only need presence data to train
the model!
  bc_K[[i]] <- evaluate(p=test[,1:8],a=backTest[,1:8], bc_bradypus_K)#use testing data
(kfold==i) for model evaluation

  #check the AUC by plotting ROC values

  plot(bc_K[[i]],'ROC')

}


#inspect

bc_K


#get the AUC from the evaluation reults

aucBIO <- sapply( bc_K, function(x){slot(x, 'auc')} )


#calculate the mean AUC value for comparison with other models


mean(aucBIO)

Opt_BIO<-sapply(bc_K, function(x){ x@t[which.max(x@TPR + x@TNR)] } )

Opt_BIO





#take the mean of the OptBIO list for application in our predictions

Mean_OptBIO<- mean(Opt_BIO)
```

```
Mean_OptBIO



prBIO <- predict(env, bc_bradypus_K,type = "response")
par(mfrow=c(1,2))
plot(prBIO, main='BIO_K')
plot(wrld_simpl, add=TRUE, border='dark grey')
plot(prBIO > Mean_OptBIO, main='presence/absence')
plot(wrld_simpl,add=TRUE, border ='dark grey')





#create an empty list to hold our results (remember there will be five sets)
eGLM<-list()
par(mfrow=c(2,3))

for (i in 1:folds) {
  train <- Pres.cov[kfold_pres!= i,]
  test <- Pres.cov[kfold_pres == i,]
  backTrain<-Back.cov[kfold_back!=i,]
  backTest<-Back.cov[kfold_back==i,]
  dataTrain<-rbind(train,backTrain)
  dataTest<-rbind(test,backTest)
  glm_eval <- glm(Pres~.,binomial(link = "logit"), data=dataTrain)#this is our glm mod
el trained on presence and absence points
  eGLM[[i]] <- evaluate(p=dataTest[ which(dataTest$Pres==1),],a=dataTest[which(dataTes
t$Pres==0),], glm_eval)#use testing data (kfold==i) for model evaluation

  #check the AUC by plotting ROC values

  plot(eGLM[[i]],'ROC')

}


#inspect

eGLM


aucGLM <- sapply( eGLM, function(x){slot(x, 'auc')} )

#calculate the mean values for comparison with other models


mean(aucGLM)
```

```
Opt_GLM<-sapply( eGLM, function(x){ x@t[which.max(x@TPR + x@TNR)] } )

Opt_GLM



#take the mean to be applied to our predictions

Mean_OptGLM<- mean(Opt_GLM)

trGLM<-plogis(Mean_OptGLM)

trGLM


prGLM <- predict(env, glm_eval,type = "response")
par(mfrow=c(1,2))
plot(prGLM, main='GLM, regression')
plot(wrld_simpl, add=TRUE, border='dark grey')
plot(prGLM > trGLM, main='presence/absence')
plot(wrld_simpl,add=TRUE, border ='dark grey')



###################################IPP data prep



# select a large number of background points using the sampleRegular() function (note
 that with this function we have the option of returning an spatial points option usin
g 'sp=TRUE')

BG<-sampleRegular(env,20000,xy=FALSE,sp=TRUE)



BG<-spTransform(BG,crs(bradypus)) #align CRS


backPPM<-BG[SA,] #subset to study area



#create Pres variable and assign '0'
backPPM$Pres=0



back.covPPM<-data.frame(backPPM)


back.covPPM<-back.covPPM[,1:9] # remove unwanted columns
```

```
all.covPPM<-rbind(Pres.cov,back.covPPM) # rbind for new data frame

head(all.cov)


# specify the gim model assigned very large weights to background points
glm.ppm <- glm(Pres ~. , family = binomial(link=logit), weights = 10^(1-Pres),
               data = all.covPPM)

summary(glm.ppm) # summary of regression






eGLMppm<-list()
par(mfrow=c(2,3))

for (i in 1:folds) {
  train <- Pres.cov[kfold_pres!= i,]
  test <- Pres.cov[kfold_pres == i,]
  backTrain<-back.covPPM[kfold_back!=i,]
  backTest<-back.covPPM[kfold_back==i,]
  dataTrain<-rbind(train,backTrain)
  dataTest<-rbind(test,backTest)
  glm_evalPPM <- glm(Pres ~. , family = binomial(link=logit), weights = 10^(1-Pres), d
ata=dataTrain) # weight background points to approximate a point proces model
  eGLMppm[[i]] <- evaluate(p=dataTest[ which(dataTest$Pres==1),],a=dataTest[which(data
Test$Pres==0),], glm_evalPPM)
  plot(eGLMppm[[i]],'ROC')

}

aucGLMppm <- sapply( eGLMppm, function(x){slot(x, 'auc')} )
aucGLMppm
mean(aucGLMppm)

TNR_GLMppm<-sapply( eGLMppm, function(x){ x@t[which.max(x@TPR + x@TNR)] } )

TNR_GLMppm

Mean_TNR_GLMppm<- mean(TNR_GLMppm)

Mean_TNR_GLMppm



#the glmPPM model seems to work well only inside the extent of of the bradypus point d
ata set so clip the output to this extent:
bradExtent<-extent(bradypus)
```

```
prPPM <- predict(env, glm_evalPPM,type = "link",ext=bradExtent)
par(mfrow=c(1,2))
plot(prPPM, main='GLMppm, regression')
plot(wrld_simpl, add=TRUE, border='dark grey')
tr <- Mean_TNR_GLMppm
plot(prPPM > tr, main='presence/absence')
plot(wrld_simpl, add=TRUE, border='dark grey')
```

```
#We can perform the same evaluation process on our maxent model also (we will leave k-
fold validation of the #lesser-performing bioclim model).
```

```
par(mfrow=c(2,3))

eMAX<-list()

folds=5

kfold_pres <- kfold(Pres.cov, folds)
kfold_back <- kfold(Back.cov, folds)


for (i in 1:folds) {
  train <- Pres.cov[kfold_pres!= i,]
  test <- Pres.cov[kfold_pres == i,]
  backTrain<-Back.cov[kfold_back!=i,]
  backTest<-Back.cov[kfold_back==i,]
  dataTrain<-rbind(train,backTrain)
  dataTest<-rbind(test,backTest)
  maxnet_eval <- maxnet(dataTrain$Pres, dataTrain[,1:8],regmult=3)
  eMAX[[i]] <- evaluate(p=dataTest[which(dataTest$Pres==1),],a=dataTest[which(dataTest
$Pres==0),], maxnet_eval)
  plot(eMAX[[i]],'ROC')

}
```

```
#Again, we can access our AUC and MaxTPR+TNR statistics for this latest model (for ref
erence, the equivalent AUC #evaluation for the same analysis using the stand-alone Max
ent package was 0.901).
```

```
aucMAX <- sapply( eMAX, function(x){slot(x, 'auc')} )

#calculate the mean values for coparison with other models

aucMAX

mean(aucMAX)


#Get maxTPR+TNR for the maxnet model

Opt_MAX<-sapply( eMAX, function(x){ x@t[which.max(x@TPR + x@TNR)] } )

Opt_MAX

Mean_OptMAX<-mean(Opt_MAX)

Mean_OptMAX

# The threshold values for the maxnet model relate to the raw outputs (prior to transf
ormation). We can acheive our #presence/absence map simply by applying our threshold t
o these raw values. To do so, we map the raw results of the #model (i.e, this time we
 do not opt for a 'cloglog' or other transform) and then remove all values below our #
threshold.




prMAX <- predict(env, maxnet_eval)
par(mfrow=c(1,2))
plot(prMAX, main='Maxent Prediction')
plot(wrld_simpl, add=TRUE, border='dark grey')
plot(prMAX > Mean_OptMAX, main='presence/absence')
plot(wrld_simpl,add=TRUE, border ='dark grey')
```