Disciplina IBE 875 - Modelagem de Distribuição de Espécies

Professores: Rodrigo Tardin, Maria Lucia Lorini, Mariana Mira Vasconcellos

Roteiro – Script 3 – Registros de ocorrência, pseudo-ausências e background.

Nesse exercício, iremos importar os registros de ocorrência da espécie e fazer diferentes etapas de filtragem desses dados para evitar potenciais vieses. Além disso, criaremos as pseudo-ausências considerando três diferentes estratégias, mais comumente discutidas na literatura científica.

Funções usadas

- `clean_coordinates`
- `filter`
- `select`
- `thin`
- `BIOMOD_FormatingData`

Agora que você já tem as suas variáveis preditoras organizadas, vamos passar a outra etapa importante dentro do protocolo ODMAP: os registros de ocorrência. Mas, primeiro, vamos definir seu diretório de trabalho que deve ser a pasta inicial da disciplina onde você colocou os arquivos de dados para as aulas práticas. E vamos carregar os pacotes necessários e os objetos salvos do script 2.

```
setwd("C:/Users/rhtar/OneDrive/R/SDM_PPGE_PPGBio/") #Mude para o
endereço da pasta da disciplina no seu computador

getwd()

## Carregando pacotes

library(tidyr)

library(maptools)

library(spThin)

library(raster)

library(biomod2)
```

- Carregando os objetos do script 2

```
load("script2.RData")
```

- Carregando os registros de ocorrência

Assim como as camadas, os registros de ocorrência da Araponga estão disponíveis na pasta do Google Drive da disciplina. Como o arquivo está no formato .csv, vamos usar uma função específica par isso.

```
procnias=read.csv("./Procnias/procnias_nudicollis.csv", header=T, sep=";")

View(procnias)
```

Os dados que você acabou de carregar são registros obtidos no GBIF (*Global Biodiversity Information Facility*). Nessas planilhas, vocês terão acesso a diversas informações sobre os registros dessas espécies, incluindo classificação taxonômica, coordenadas geográficas, informações temporais, etc. Nesse sentido, cada linha representa um registro de ocorrência. Uma vez que os registros são alimentados por diversas fontes, uma etapa importante é fazer uma filtragem, para checar inconsistências, remover potenciais vieses de amostragem, etc.

- Checando inconsistências relacionadas aos registros de ocorrências - problemas, filtrando e fazendo limpeza

Abaixo iremos remover linhas (registros de ocorrência) nas colunas das coordenadas que tenham dados faltantes (NA).

```
procnias_na <- procnias %>% tidyr::drop_na(decimalLongitude, decimalLatitude)

View(procnias_na)
```

Quantos registros foram removidos por não conter coordenadas?

Além de limpar os registros que não possuam coordenadas geográficas, muitas vezes existem outros problemas que precisam ser filtrados, como registros duplicados, que estejam fora da área de estudo, etc. Além disso, podemos tentar diminuir um potencial viés de incerteza de onde foi obtido, removendo registros que estejam ao redor das capitais, em áreas urbanas, etc.

- Limpando registros com outros problemas referente as coordenadas e marcando os registros potencialmente problemáticos

```
flags_spatial <- CoordinateCleaner::clean_coordinates(
  x = procnias_na,
  species = "species",
```

```
  lon = "decimalLongitude",
  lat = "decimalLatitude",
  tests = c("capitals", #raio ao redor de capitais
                "centroids", # raio ao redor de centroides de paises e
provincias
            "duplicates", # duplicatas
            "equal", # coordenadas iguais
            "gbif", # raio ao redor da sede da GBIF
                  "institutions", # raio ao redor de instituicoes de
pesquisa em biodiversidade
            "seas", # pontos no mar
            "urban", # pontos dentro de areas urbanas
            "validity", # ponto de fora do sistema de coordenadas
            "zeros" # zeros e pontos onde lat = lon ))
```

O que esse teste faz, basicamente, é indicar (*flag*) os registros que podem ser problemáticos baseado nos critérios que você definiu acima (remover duplicatas, pontos ao redor de capitais, coordenadas iguais, pontos no mar, etc)

```
# Resultado dos registros que foram problematicos
# TRUE = coordenadas 'limpas'
# FALSE = coordenadas potencialmente problematicas
head(flags_spatial)
summary(flags_spatial)
```

A partir de agora você já sabe que existem vários registros que podem ser problemáticos e que precisam ser removidos

- Excluindo os pontos considerados como problemáticos na analise anterior

```
procnias_f <- procnias_na %>%
  dplyr::filter(flags_spatial$.summary == TRUE)
```

Para facilitar o uso do arquivo que contém os registros, vamos dar uma limpada e manter apenas as colunas que são importantes para as etapas seguintes da modelagem.

- Limpando e selecionando as colunas de interesse

```
procnias_f = procnias_f %>%
  dplyr::select(species, decimalLongitude, decimalLatitude)
```

- Checando a extensão do arquivo limpo

```
nrow(procnias_f)

colnames(procnias_f)
```

Quantos registros ficaram retidos após a remoção dos problemáticos?

Agora, vamos visualizar os registros de ocorrência depois de terem passado por essa primeira etapa de filtragem, no polígono de distribuição da Araponga

Plotando os registros de ocorrência filtrados para visualização

```
alt=raster("./Camadas/Presente/WC_alt_lonlat.tif")

plot(alt, xlim=c(-80,-30),ylim=c(-40,10), main="Pontos de ocorrencia por altitude")

data(wrld_simpl)

plot(wrld_simpl, add=TRUE, border='darkgray', lwd=1)

points(procnias_f$decimalLongitude, procnias_f$decimalLatitude, col='blue', cex=0.3)
```

Além dos problemas de inconsistências relacionados aos arquivos baixados no GBIF, alguns problemas podem estar associados ao viés amostral, refletindo uma amostragem desigual e tendenciosa que pode trazer sérias consequências a modelagem de distribuição de uma espécie. Nesse caso, precisamos filtrar os registros baseado na sua distribuição espacial, para evitar problemas de autocorrelação espacial. Iremos usar a função 'thin'. Essa função usa um algoritmo de aleatorização que cria um conjunto de dados no qual todos os registros de ocorrência estarão separados a uma distância 'x'.

- Filtrando os registros de ocorrência baseado em uma distância 'x' para evitar potenciais problemas de autocorrelação espacial dos dados no modelo

```
procnias_thin <-
  thin( loc.data = procnias_f, #objeto contendo os registros
                lat.col = "decimalLatitude", long.col = "decimalLongitude",
        spec.col = "species",
           thin.par = 9.2 #distancia em km que os registros vão ficar separados#
        ,reps = 5, #numero de repetições
        locs.thinned.list.return = TRUE,
        write.files = TRUE,
```

```
                max.files = 2, #numero máximo de arquivos contendo os
registros filtrados criados com o algoritmo

               out.dir = "C:/Users/rhtar/OneDrive/R/ENM_PPGE/Procnias",
out.base = "procnias_thinned" #pasta onde os arquivos com os registros
e log foram salvos,

              write.log.file = TRUE,

              log.file = "procnias_thinned_log.txt")
```

Uma vez que se trata de um algoritmo de aleatorização, cada arquivo gerado terá registros levemente diferentes entre si. O objetivo da função é chegar em um conjunto de dados que apresente o número máximo de registros obedecendo-se os parâmetros acima (principalmente o argumento referente as repetições). Podemos checar se o número de registros chegou ao máximo a partir das repetições usadas. No console você terá mais detalhes sobre o resultado desse teste e o número máximo de registros que foi retido após o processo de filtragem espacial.

- Explorando visualmente o efeito do processo de filtragem espacial dos dados

```
plotThin(procnias_thin)
```

A partir dos resultados desse teste (gráfico na aba 'Plot'), é possível ver que os valores chegaram no valor máximo com as repetições usadas. Dê uma olhada no arquivo 'log' com todos os detalhes, que foi gravado na sua pasta (indicado no argumento 'out.dir' da função acima).

Agora que a filtragem espacial foi realizada e gerou novos arquivos, precisamos carregar um desses arquivo para o R.

- Carregando o novo arquivo com o processo de filtragem espacial realizado.

```
      procnias_f=read.csv("./Procnias/procnias_thinned_thin1.csv",
sep=",")

      View(procnias_f)
```

- Visualizando os registros de ocorrência filtrados em relação aos antigos

```
plot(alt, xlim=c(-80,-30),ylim=c(-40,10), main="Pontos de ocorrencia
por atitude")

plot(wrld_simpl, add=TRUE, border='darkgray', lwd=1)

points(procnias_f$decimalLongitude, procnias_f$decimalLatitude,
col='blue', cex=0.3)

points(procnias_thin$decimalLongitude, procnias_thin$decimalLatitude,
col='red', cex=0.2)
```

Uma vez que para as etapas de modelagem seguintes é necessário ter no seu arquivo de registros (procnias_thin) uma coluna com '1' preenchido para cada linha, temos que fazer isso no R.

- Inserindo uma coluna contendo todos os registros igual a 1

```
resp.occ=as.numeric("resp.occ")
procnias_thin[,"resp.occ"] = 1
```

- Checando se houve adição da coluna

```
colnames(procnias_thin)
View(procnias_thin)
```

Agora que todos os registros foram filtrados quanto a inconsistências e para evitar potenciais problemas de autocorrelação espacial, estamos prontos para gerar as pseudo-ausencias. Para isso, primeiro temos que formatar os registros para o framework do BIOMOD 2.

- Formatando os registros de ocorrência para a modelagem no ambiente BIOMOD

```
#Salvando um objeto apenas com o nome da espécie
resp.name <- 'procnias'

# dados de presença da nossa espécie
is.numeric(procnias_thin$resp.occ)
myResp <- as.numeric(procnias_thin[,"resp.occ"])

#Criando um objeto apenas com as coordenadas da Araponga e concatenando o valor 1 para todas as coordenadas
myRespXY <- procnias_thin[which(myResp==1),c("decimalLongitude", "decimalLatitude")]

# Criando uma camada raster baseada nas camadas ambientais do script 2 para nossa resposta variável (registros)
myResp <- reclassify(subset(biostack2,1,drop=TRUE), c(-Inf,Inf,0))
myResp[cellFromXY(myResp,myRespXY)] <- 1
```

Os passos acima são cruciais para gerarmos as pseudo-ausências referentes aos algoritmos que vamos usar nas etapas seguintes. Como não temos ausências verdadeiras, temos que criar pseudo-ausências. A criação da pseudo-ausência pode ser feita seguindo três protocolos distintos: pseudo-ausências aleatórias (`strategy = "random"`), espacialmente estruturadas (`strategy = "disk"`) e climaticamente estruturada (`strategy = "sre"`). Vamos fazer isso usando uma das funções mais importantes do BIOMOD 2, a função `BIOMOD_FormatingData()`.

- Criação das pseudo-ausências

```
#Estratégia aleatória

#100 pseudo ausências e 4 sets

bm.procnias100 = BIOMOD_FormatingData(resp.var = resp.occ,
#variavel resposta (registros)
                                      expl.var = biostack2,
#variaveis preditoras
                                        resp.xy = myRespXY,
#objeto com as coordenadas e o valor de ocorrência 1
                                    resp.name = resp.name,
                                     PA.nb.rep = 4, #numero de
conjuntos de pseudo-ausências
                                         PA.nb.absences = 100,
#numero de pseudo-ausencias
                                    PA.strategy = "random"
#tipo de estrategia usada para gerar as pseudo-ausencias)
```

Quando você roda o nome do objeto com as pseudo-ausências que você criou, você terá um sumário do número das pseudo-ausencias criadas e o número de conjuntos (*sets*). É sempre importante checar esses dois parâmetros para saber se o que você indicou na função `BIOMOD_FormatingData`, de fato foi realizado.

```
#sumário do novo objeto criado

bm.procnias100


#Visualização da configuração espacial das pseudo-ausências

plot(bm.procnias100)
```

Certo! Acabamos de criar 100 pseudo-ausências para cada um dos 4 conjuntos aleatoriamente distribuídos. Agora vamos usar a estratégia espacialmente estruturada, onde indicamos uma distância mínima dos registros de ocorrência onde as pseudo-

ausências serão geradas. Perceba que nesse caso, além do argumento `strategy` mudar, também temos um outro argumento onde indicamos a distância mínima (`PA.dist.min`). Nesse caso vamos indicar que as pseudo-ausências só serão geradas a partir de uma distância de 100 km dos registros (100.000 metros).

```
      bm.procnias100disk = BIOMOD_FormatingData(resp.var = resp.occ,
#variavel resposta (registros)
                                                expl.var = biostack1,
#variaveis preditoras
                                                  resp.xy = myRespXY,
#objeto com as coordenadas e o valor de ocorrência 1
                                              resp.name = resp.name,
                                                PA.nb.rep = 4, #numero de
conjuntos de pseudo-ausências
                                                PA.nb.absences = 100,
#numero de pseudo-ausencias
                                                PA.strategy = "disk",
#tipo de estrategia usada para gerar as pseudo-ausencias)
                                                PA.dist.min = 100000)
#distancia em metros


      #sumário do novo objeto criado
      bm.procnias100disk
```

Maravilha! Acabamos de criar 100 pseudo-ausências para cada um dos 4 conjuntos. Dessa vez, diferentemente da estratégia aleatória, todas as pseudo-ausências só foram geradas a partir de 100km de cada registro. Vamos observar isso em um mapa

```
      #Visualização da configuração espacial das Pseudo Ausências
espacialmente estruturadas
      plot(bm.procnias100disk)
```

A terceira maneira de criarmos pseudo-ausências é a estratégia climaticamente estruturada. Nesse caso, primeiro vamos criar uma modelagem de envelope climático usando o algoritmo 'Species Range Envelope' (SRE), que é uma versão equivalente ao BIOCLIM. A geração das pseudo-ausências vai ocorrer fora desse envelope climático. Nesse caso, estamos indicando que dentro do envelope climático criado usando o SRE a espécie estaria presente e que apenas fora desse envelope a espécie não ocorre.

```
      bm.procnias100sre = BIOMOD_FormatingData(resp.var = resp.occ,
```

```
                                                    expl.var = biostack2,
                                                    resp.xy = myRespXY,
                                                    resp.name = resp.name,
                                                    PA.nb.rep = 4,
                                                      PA.nb.absences = 100,
PA.strategy = "sre", PA.sre.quant = 0.025, na.rm = T)

      #sumário do novo objeto criado
      bm.procnias100sre

      #Visualização da configuração espacial das Pseudo Ausências
      plot(bm.procnias100sre)
```

Vamos agora salvar todas esses objetos como um RData que podemos carregar durante os próximos passos. Dessa forma, não precisamos gerar todos esses objetos novamente. Especialmente o dados de pseudo-ausência que iremos usar nos próximos passos.

```
      save.image(file="script3.RData")
```

Quais foram as principais diferenças na geração das pseudo ausências com as diferentes estratégias?

Explore mais opções, alterando o numero de repetições, numero de pseudo-ausencia, distancia minima e quantil do SRE

```
      #Fim do Script 3
```