Disciplina IBE 875 - Modelagem de Distribuição de Espécies

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

Roteiro – Script 5 – Geração dos modelos usando os algoritmos de classificação e máxima entropia

Esse exercício será muito similar ao anterior, porém, nesse iremos rodar os modelos considerando os algoritmos de aprendizado de máquina do tipo de classificação/regressão (Random Forest e Generalized Boosted Method, especificamente a Boosted Regression Tree aqui) e de máxima entropia (MaxEnt). Para tal, precisaremos ter realizado as etapas anteriores, uma vez que usaremos o conjunto de variáveis preditoras (`biostack1`), os registros de ocorrência filtrados (`procnias_f`) e as pseudo-ausências geradas (`procniasbm100disk`). Nesse momento, ainda não projetaremos as predições dos modelos no espaço geográfico, nem avaliaremos o desempenho dos modelos. Isso será realizado na nossa aula 6.

Para esse exercício, entretanto, como vamos usar o MaxEnt, você precisa ter instalado o JAVA no seu PC e baixar o Maxent do drive da disciplina no seu PC.

- Primeiro, salve o arquivo maxent.rar na pasta que você está usando o R para a disciplina
- Extraia o arquivo maxent.rar. Você verá que 3 arquivos estarão presentes agora na sua pasta principal: maxent (em lotes), maxent.jar, maxent.sh.

A localização do arquivo maxent.jar é importante para que o MaxEnt funcione dentro da lógica do BIOMOD 2. Vamos ter que indicar isso mais tarde no script.

Funções usadas

- `BIOMOD_Modeling`
- `get_variables_importance`
- `response.plot2`

Uma das vantagens de se usar o BIOMOD 2 é conseguir usar diferentes algoritmos, partições para calibração do modelo em treino/teste e usar diferentes métricas de avaliação de desempenho dos modelos em poucas funções. O BIOMOD 2 permite usar até 10 algoritmos, mas vamos focar em apenas seis dos mais usados pela literatura. Nesse exercício vamos usar 3 dos 6: RF (*Random Forest*), GBM (*Generalized Boosting Method*), MaxEnt (*Maximum Entropy model*).

#Nota: Apesar do termo GBM ser usado aqui e no BIOMOD 2, o teste que de fato é realizado é o *Boosted Regression Tree* (BRT).

Para fazer a modelagem da distribuição da Araponga no BIOMOD 2, a função principal que vamos usar é a `BIOMOD_Modeling` onde vamos indicar todos os parâmetros usados.

Assim como no exercício anterior, vamos conhecer os parâmetros *default* desses algoritmos. Preste atenção, especificamente, nas especificações dos algoritmos que vamos usar nesse exercício: `RF, GBM, MAXENT.Phillips`

- Checando as opções 'default' de cada algoritmo

```
BIOMOD_ModelingOptions()
```

Quando você rodar essa linha de comando, você perceberá que várias informações serão geradas na tela de console. Suba até chegar no início dos resultados referentes a essa função. O início se parece com o formato abaixo:

```
-=-=-=-=-=-=-=-=-=-=-=-=  'BIOMOD.Model.Options'  -=-=-=-=-=-=-=-
```

Agora vamos observar as definições dos algoritmos que usaremos nesse exercício

```
GBM = list( distribution = 'bernoulli',
            n.trees = 2500,
            interaction.depth = 7,
            n.minobsinnode = 5,
            shrinkage = 0.001,
            bag.fraction = 0.5,
            train.fraction = 1,
            cv.folds = 3,
            keep.data = FALSE,
            verbose = FALSE,
            perf.method = 'cv',
            n.cores = 1)
```

No caso do GBM (*Boosted Regression Tree*), as especificações também estão de acordo com a literatura, indicando um numero grande de árvores a serem realizadas (`n.trees`) e um baixo nível de `shrinkage`, indicando boa taxa de aprendizado. A fração aleatória dos dados usada para ajustar cada árvore (`bag.fraction`) esta no limite inferior do adequado (indicado de 50 a 75%) e por isso vamos alterar essa configuração a seguir. O uso de 3 partições para fazer um teste de validação cruzada para estimar os erros (`cv.folds`) está adequado.

```
RF = list( do.classif = TRUE,
           ntree = 500,
           mtry = 'default',
           nodesize = 5,
           maxnodes = NULL),
```

No caso das *Random Forest*, temos as configurações básicas e adequadas, indicando que arvores de classificação estará sendo computada (presença/ausência), o número de arvores máximas que serão geradas (`ntree`), que o numero de variáveis que poderão ser usadas a cada divisão (`mtry`) será a base 'default' do pacote randomForest (√(n variáveis explanatórias)), que o tamanho máximo dos nós (`nodesize`) permitirá árvores grandes a serem criadas e sem limitação do número máximo de árvores (`maxnodes`) .

No caso do MaxEnt, as especificações estão adequadas, indicando a criação de 10.000 pontos de background (`maximumbackground = 'default'`), indicando so diferentes *features* permitindo relações lineares, não-lineares (quadráticas), a interação entre variáveis e as abordagens de binarização das covariáveis a partir de valor de corte. Para evitar potenciais modelos sobreajustados, vamos retirar um dos features "`Hinge`". Manteremos os fatores de multiplicação, como está seguindo sugestões em Radosavjlevic & Anderson (2013)

```
MAXENT.Phillips = list( path_to_maxent.jar = 'C:/Users/rhtar/OneDrive/R/ENM_PPGE',
                        memory_allocated = 512,
                        background_data_dir = 'default',
                        maximumbackground = 'default',
                        maximumiterations = 200,
                        visible = FALSE,
                        linear = TRUE,
                        quadratic = TRUE,
                        product = TRUE,
                        threshold = TRUE,
                        hinge = TRUE,
                        lq2lqptthreshold = 80,
                        l2lqthreshold = 10,
                        hingethreshold = 15,
                        beta_threshold = -1,
                        beta_categorical = -1,
                        beta_lqp = -1,
```

```
                beta_hinge = -1,
                betamultiplier = 1,
                defaultprevalence = 0.5)
```

- Modificando as opções do Algoritmo GBM em relação a fração aleatória dos dados usados para ajustar cada árvore

Observe que abaixo, criamos um novo objeto (`modelo_op2`), indicando as 'novas' opções dos modelos, além do *default*. Estamos modificando 1) no GBM, a fração dos dados para ajustar cada árvore (`bag.fraction` de 0.5 (*default*) para 0.7) para ficar mais similar ao processo de calibração e teste dos outros algoritmos; 2) no MaxEnt, a remoção do *feature* '`Hinge`'para tentar evitar potenciais problemas de sobreajuste dado a complexidade muito grande dos modelos. O resto permanece igual.

```
      modelo_op2=BIOMOD_ModelingOptions(GBM = list( distribution = 'bernoulli',
                                                    n.trees = 2500,
                                                    interaction.depth = 7,
                                                    n.minobsinnode = 5,
                                                    shrinkage = 0.001,
                                                    bag.fraction = 0.7,
                                                    train.fraction = 1,
                                                    cv.folds = 3,
                                                    keep.data = FALSE,
                                                    verbose = FALSE,
                                                    perf.method = 'cv',
                                                    n.cores = 1),  ,
MAXENT.Phillips = list( path_to_maxent.jar = 'C:/Users/rhtar/OneDrive/R/ENM_PPGE',  memory_allocated = 512,
background_data_dir = 'default',
maximumbackground = 'default',
maximumiterations = 200, visible = FALSE,
linear = TRUE,
quadratic = TRUE,
                    product = TRUE,
                    threshold = TRUE,
                    hinge = FALSE,
                    lq2lqptthreshold = 80,
                    l2lqthreshold = 10,
```

```
                    hingethreshold = 15,
                    beta_threshold = -1,
                    beta_categorical = -1,
                    beta_lqp = -1,
                    beta_hinge = -1,
                    betamultiplier = 1,
                    defaultprevalence = 0.5))
```

Antes de começarmos a modelar a distribuição usando os 3 algoritmos, precisamos preparar o BIOMOD para se conectar com as especificações do MaxEnt.

- Formatando os dados, camadas e fazendo conexão para o formato do Maxent

```
    # Para trabalhar com o MAXENT nós temos que criar um caminho para um diretório contendo nossas variaveis explanatórias em ascii


    maxent.background.dat.dir <- "maxent_bg"
    dir.create(maxent.background.dat.dir, showWarnings = FALSE, recursive = TRUE)
```

Ao fazer isso, vocês vão perceber que na pasta que estão trabalhando, vai surgir uma outra pasta chamada ‘maxent_bg’. Lá vamos salvar as variáveis preditoras no formato .ascii

```
    ## salvando as variáveis preditoras em .ascii
    for(var_ in names(biostack2)){
      cat("\n> saving", paste0(var_, ".asc"))
      writeRaster(subset(biostack1, var_),
                  filename = file.path(maxent.background.dat.dir, paste0(var_, ".asc")),
                  overwrite = TRUE)
    }
```

Agora um passo importante é definir o caminho para o arquivo maxent.jar. A partir dessa definição, a função BIOMOD_Modeling vai reconhecer que o maxent está disponível e vai permitir a modelagem com esse algoritmo.

```
    path.to.maxent.jar <- file.path(getwd(), "maxent.jar")
```

Agora que já ajustamos algumas opções dos algoritmos e formatos que usaremos, vamos começar a modelar! Usaremos a função `BIOMOD_Modeling` e indicaremos todos os parâmetros que queremos usar.

- Definindo os parâmetros dos modelos, incluindo a escolha dos algoritmos, o número de rodadas e a divisão do conjunto de dados entre treino e teste.

Na função `BIOMOD_Modeling` alguns argumentos são indispensáveis para funcionar corretamente. Na função abaixo, para cada um dos argumentos, está explicado o que faz. Então, podemos ver que estamos modelando a distribuição da Araponga, usando o conjunto de pseudo-ausencias espacialmente estruturadas (bm.procnias100disk) a partir de 3 algoritmos (`GBM, RF, MAXENT.Phillips`). Vamos fazer 3 rodadas para cada algoritmo e conjunto de pseudo-ausencia, particionando nossos dados em 70% para calibrar os modelos (treino) e 30% para teste (validação) (`DataSplit`). Nesse caso, escolhemos 3 rodadas apenas para facilitar o processo de computação, uma vez que quanto mais rodadas, maior o tempo necessário para rodar todos os modelos. De forma geral, com base na literatura, por volta de 10 rodadas é um número adequado.

Como uma etapa importante da modelagem é investigar quais variáveis mais influenciam na distribuição da espécie, vamos usar 3 permutações para gerar o teste de importância das variáveis que vem com o pacote (`VarImport` – explicações mais detalhadas sobre esse teste logo depois ainda nesse mesmo exercício) . Além disso, vamos usar os algoritmos `TSS` e `ROC` para avaliar o desempenho dos modelos (que abordaremos na nossa sexta aula).

```
procnias2model = BIOMOD_Modeling(
  bm.procnias100disk, #objeto das pseudo-ausências
  models = c("GBM", "RF","MAXENT.Phillips"), #algoritmos de escolha
  models.options = modelo_op2, #opções personalizadas ou padrões dos algoritmos
  NbRunEval = 3, #Numero de rodadas
  DataSplit = 70, #Divisão treino/teste
  Prevalence = 0.5,
  VarImport = 3, #permutações para gerar o valor de importância das variáveis
  models.eval.meth = c("TSS", "ROC"), #método de avaliação do desempenho dos modelos
  SaveObj = TRUE, #se os modelos serão salvos ou não
  rescal.all.models = F,
  do.full.models = FALSE,
  modeling.id = "procnias_classif_maxent")
```

Assim como no exercício anterior, quando você rodar esse código, você perceberá que na tela de console, uma série de informações começarão a ser geradas. Dessa vez, o tempo de processamento talvez seja um pouco mais demorado devido ao MaxEnt. O que basicamente está sendo mostrado ali é um 'log' do que está sendo rodado. Aqui é importante perceber a lógica de como os modelos estão sendo construídos. Cada modelo aqui corresponde a: Algoritmo + rodada da modelagem (`RUN` - até 3) + conjunto de pseudo-ausencia (`PA` – até 4). Logo no início dessas informações, você perceberá que o primeiro conjunto de pseudo-ausencia (`PA1`) está sendo usada para a primeira rodada da modelagem (`RUN1`) para cada algoritmo (`GBM, RF, MAXENT.Phillips`), seguindo a mesma lógica sucessivamente. Veja uma parte dessas informações que vão aparecer na sua tela de console abaixo:

```
-=-=- Run :  procnias_PA1



-=-=-=-=-=-=-=- procnias_PA1_RUN1


Model=Generalised Boosting Regression
      2500 maximum different trees and 3 Fold Cross-
ValidationCV: 1
CV: 2
CV: 3


      Evaluating Model stuff...
      Evaluating Predictor Contributions...


Model=Breiman and Cutler's random forests for classification and
regression
      Evaluating Model stuff...
      Evaluating Predictor Contributions...


Model=MAXENT.Phillips
      Creating Maxent Temp Proj Data..
 Running Maxent...
 Getting predictions...
      Removing Maxent Temp Data..
      Evaluating Model stuff...
      Evaluating Predictor Contributions...
```

```
-=-=-=-=-=-=-=- procnias_PA1_RUN2


      Model=Generalised Boosting Regression
            2500  maximum  different  trees  and   3   Fold  Cross-
ValidationCV: 1
      CV: 2
      CV: 3


            Evaluating Model stuff...
            Evaluating Predictor Contributions...


      Model=Breiman and Cutler's random forests for classification and
regression
            Evaluating Model stuff...
            Evaluating Predictor Contributions...


      Model=MAXENT.Phillips
            Creating Maxent Temp Proj Data..
       Running Maxent...
       Getting predictions...
            Removing Maxent Temp Data..
            Evaluating Model stuff...
            Evaluating Predictor Contributions...
```

Ao fim da rodada desse código, você terá realizado a sua primeira modelagem de distribuição da Araponga usando 3 algoritmos diferentes. A partir da lógica acima (Algoritmo + Rodada de modelagem (`RUN`) + conjunto de de pseudo-ausencia (`PA`)), você consegue saber quantos modelos você rodou ao total. Logo, 3 algoritmos x 3 rodadas x 4 conjuntos de PA dá um total de 36 modelos diferentes rodados. Você consegue saber exatamente esses modelos rodando o nome do objeto que você acabou de criar onde contém os parâmetros da modelagem.

- Sumário do objeto com os modelos criados, onde é possível ver quais modelos foram gerados (conjunto de pseudoausencia + rodada + algoritmo) e se algum falhou

```
Procnias2model
```

Ótimo! Conseguimos ver cada modelo gerado e que nenhum deles falhou. Além disso, dê uma olhada na sua pasta onde você está salvando o seu workspace do R e olhe

na pasta ‘Procnias’. Você verá que agora em ‘models’, você terá uma outra pasta com o nome do `modeling.id = "procnias_classif_maxent"` com os 36 modelos gerados.

Mais uma vez, vamos tentar entender qual das variáveis preditoras que inserimos no modelo foram as mais importantes. Para isso, vamos usar a função `get_variables_importance`. O princípio desse teste é embaralhar uma única variável preditora do conjunto de dados, por vez, modelar esse conjunto embaralhado e depois calcular uma correlação simples de Pearson entre as predições de referência (modelos originais) e as do conjunto de dados ‘embaralhado’. O valor resultante será dado como 1-valor de correlação. Quanto maior o valor, maior a influência da variável no modelo.

- Obtendo a importância de cada variável usada nos modelos

```
var_import_procnias2=get_variables_importance(procnias2model)
```

Com esse código acima, criamos um objeto onde o teste de importância da variável foi realizado.

```
# obtendo os valores de importância de cada variável para cada modelo
var_import_procnias2
```

Ao rodar o nome do objeto criado, você perceberá na sua tela de console, que a importância de cada variável foi realizada para cada modelo, ou seja, algoritmo x rodada x conjunto de PA. É interessante notar que para um mesmo algoritmo, os valores referentes à importância das variáveis mudam. Para facilitar nosso entendimento, vamos obter os valores médios para cada algoritmo, unindo todas as rodadas e os conjuntos de pseudo-ausencia (PA).

```
# salvando um objeto com os valores médios de importância de cada variável para cada algoritmo
var_import_procnias2=apply(var_import_procnias2, c(1,2), mean)
# obtendo os valores médios de importância de cada variável para cada algoritmo
var_import_procnias2

# Salvando os valores de importancia das variáveis em um arquivo .csv
write.csv(var_import_procnias2,paste0("./Outputs/", "_", "var_import_procnias2.csv"))
```

Agora conseguimos entender qual variável foi a mais importante para influenciar a distribuição da Araponga dependendo de cada algoritmo. Além disso, salvamos um

arquivo .csv na nossa pasta ‘Outputs’ com esses valores. Esse arquivo pode ser aberto em qualquer Excel para facilitar a visualização e depois a inclusão em um artigo, relatório, etc.

Qual variável foi a mais importante para cada algoritmo? Houve diferenças entre algoritmos?

Apesar de agora sabermos quais variáveis são as mais importantes para modelar a distribuição da Araponga, ainda não sabemos a relação entre a adequabilidade de habitat da Araponga e cada variável preditora. Então vamos refazer o mesmo procedimento do exercício anterior para gerar as curvas respostas, só que dessa vez para os algoritmos GBM, RF, MaxEnt.

- Carregando os modelos individuais que foram gerados acima

```
procnias2_gbm=BIOMOD_LoadModels(procnias2model, models = 'GBM')
procnias2_rf=BIOMOD_LoadModels(procnias2model, models = 'RF')
procnias2_MaxEnt=BIOMOD_LoadModels(procnias2model, models = 'MAXENT.Phillips')
```

Agora que carregamos as predições de cada modelo (algoritmo x rodada x conjunto de PA) para dentro do nosso workspace, podemos construir as curvas respostas. Para isso usaremos a função `response.plot2`. Essa função é uma adaptação ao método proposto por Elith et al (2005) e permite plotar as curvas respostas de um modelo independente do algoritmo usado, permitindo uma comparação direta das respostas das diferentes abordagens estatísticas. Nessa função, temos que indicar qual objeto onde rodamos os modelos (`procnias1_glm`) e temos que indicar as nossas variáveis preditoras (ou explanatórias) (`expl.var`) e a nossa variável resposta `(resp.var)`. O restante dos argumentos da função, indica que vamos salvar o plot em uma figura, do tipo ‘jpeg’, com qualidade de 720p. Além disso, para gerarmos as curvas respostas, n-1 variáveis são mantidas constante a um valor fixo (media, mediana, mínima ou máximo) e apenas a variável resposta varia ao longo de todas as variáveis. No caso abaixo, indicamos que as curvas respostas serão geradas entre os valores preditos de adequabilidade para cada algoritmo e a mediana dos valores de cada variável preditora.

- Plotando as curvas resposta de cada modelo para cada variável, incluindo todas as rodadas.

```
biomod2::response.plot2(models = procnias2_gbm, Data = get_formal_data(procnias2model, 'expl.var'),show.variables = get_formal_data(procnias2model, 'expl.var.names'), do.bivariate = F, save.file = 'jpeg', ImageSize = 720, name= "gbm_curva_resposta", fixed.var.metric = 'median', legend = F, display_title = T, data_species = get_formal_data(procnias2model, 'resp.var') , plot = T)
```

```
      biomod2::response.plot2(models = procnias2_rf, Data =
get_formal_data(procnias2model, 'expl.var'),show.variables =
get_formal_data(procnias2model, 'expl.var.names'), do.bivariate = F,
save.file = 'jpeg', ImageSize = 720, name= "rf_curva_resposta",
fixed.var.metric = 'median', legend = F, display_title = T, data_species
= get_formal_data(procnias2model, 'resp.var') )


      biomod2::response.plot2(models = procnias2_MaxEnt, Data =
get_formal_data(procnias2model, 'expl.var'),show.variables =
get_formal_data(procnias2model, 'expl.var.names'), do.bivariate = F,
save.file = 'jpeg', ImageSize = 720, name= "MaxEnt_curva_resposta",
fixed.var.metric = 'median', legend = F, display_title = T, data_species
= get_formal_data(procnias2model, 'resp.var'))
```

Unindo o valor de importância de cada variável e a direção da relação entre adequabilidade e cada variável (curvas resposta), podemos interpretar ecologicamente, de forma mais adequada, como as variáveis mais importantes influenciam na distribuição da Araponga.

```
      #Fim do Script 5
```