Disciplina IBE 875 - Modelagem de Distribuição de Espécies

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

Roteiro – Script 4 – Geração dos modelos usando os algoritmos de envelope (SRE) e de regressão (GLM e GAM)

Nesse exercício, iremos rodar nossos primeiros modelos considerando os algoritmos de envelope e os de regressão. Para tal, precisaremos ter realizado as etapas anteriores, uma vez que usaremos o conjunto de variáveis preditoras (`biostack2`), os registros de ocorrência filtrados (`procnias_thin`) e as pseudo-ausências geradas (`procnias-bm100disk`). 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.

Principais 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: SRE (*Surface Range Envelope*), GLM (*Generalized Linear Model*), GAM (*Generalized Additive Model*).

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.

Apesar do pacote em si já estabelecer uma série de parâmetros padrão (*default*) é interessante conhecer quais parâmetros são considerados. Antes, para organizar alguns dos resultados que serão gerados, vamos criar uma pasta e nomeá-la como ‘outputs’.

- Criando um diretório 'Outputs' para guardar tabelas com resultados das modelagens

```
dir.create("Outputs")
```

Nesse diretório, serão armazenadas as tabelas referentes aos testes de importância das variáveis e de avaliação do desempenho dos modelos.

Agora vamos checar as opções padrões, já embutidas no pacote, em relação aos algoritmos. Preste atenção, especificamente, nas especificações dos algoritmos que vamos usar nesse exercício: SRE, GLM, GAM

- 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

```
GLM = list( type = 'quadratic',
            interaction.level = 0,
            myFormula = NULL,
            test = 'AIC',
            family = binomial(link = 'logit'),
            mustart = 0.5,
            control = glm.control(epsilon = 1e-08, maxit = 50
, trace = FALSE) )
```

No GLM, podemos observar que o padrão já é permitir a modelagem de relações não-lineares com a inclusão do tipo ‘quadrático’ e que usa a família de distribuição de erros ‘binomial’ (presença/ausência). Isso já inclui todas as boas práticas para esse algoritmo no que diz respeito a nossa modelagem em si. Então, não precisaremos mexer em nada aqui.

```
GAM = list( algo = 'GAM_mgcv',
            type = 's_smoother',
            k = -1,
            interaction.level = 0,
            myFormula = NULL,
            family = binomial(link = 'logit'),
            method = 'GCV.Cp',
```

```
                    optimizer = c('outer','newton'),
                    select = FALSE,
                    knots = NULL,
                    paraPen = NULL,
                    control = list(nthreads = 1, irls.reg = 0, epsilon =
1e-07
      , maxit = 200, trace = FALSE, mgcv.tol = 1e-07, mgcv.half = 15
      , rank.tol = 1.49011611938477e-08
      , nlm = list(ndigit=7, gradtol=1e-06, stepmax=2, steptol=1e-04,
iterlim=200, check.analyticals=0)
      , optim = list(factr=1e+07)
      , newton = list(conv.tol=1e-06, maxNstep=5, maxSstep=2, max-
Half=30, use.svd=0)
      , outerPIsteps = 0, idLinksBases = TRUE, scalePenalty = TRUE
      , efs.lspmax = 15, efs.tol = 0.1, keepData = FALSE, scale.est =
fletcher
      , edge.correct = FALSE) )
```

No GAM, apesar das definições serem bem mais extensas do que no GLM, vamos prestar atenção em alguns parâmetros: `algo, type, k, Family`. Podemos observar que o pacote usado para modelar o GAM é um dos mais usados '`mgcv`', que o tipo de ajuste a ser realizado é o '`s`' (o único disponível nesse pacote), que envolve como a função de *Thin Plate Splines* vai penalizar o coeficiente base para controlar e evitar possíveis sobreajustes e que gera os melhores valores de performance de Erro Quadrático Médio, que o número de nós (k) é igual a -1 e a família tbm é binomial. Nesse caso, o número de nós usado como *default* é -1, o que implica dizer que o método de GCV (*Generalized Cross Validation*) irá escolher automaticamente para você. Em geral, esse método pode levar a relações pouco realistas e ter algum efeito de sobreajuste. Para evitar isso, vamos colocar `k = 4`, considerado pela literatura adequado para evitar sobreajustes.

```
SRE = list( quant = 0.025)
```

Nesse caso, o *default* do SRE indica que os valores mais extremos das variáveis preditoras dentro do quantil de 5% seja descartado para produzir o envelope climático. Como estamos considerando ambos os extremos (valores mais altos e menores) da sua distribuição, então totalizará 10% dos valores que não serão considerados, o que está adequado.

- Modificando as opções do Algoritmo GAM quanto ao numero de nós (k = 4) para evitar sobre ajuste

Observe que abaixo, criamos um novo objeto (`modelo_op`), indicando as 'novas' opções dos modelos, além do *default*. Só estamos modificando o número de nós (k) de -1 (*default*) para 4 para o algoritmo GAM. O resto permanece igual.

```
modelo_op <- BIOMOD_ModelingOptions(GAM = list( algo = 'GAM_mgcv', type = 's_smoother', k = 4, interaction.level = 0, myFormula = NULL, family = binomial(link = 'logit'), method = 'GCV.Cp', optimizer = c('outer','newton'), select = FALSE, knots = NULL,  paraPen = NULL, control = list(nthreads = 1, irls.reg = 0, epsilon = 1e-07, maxit = 200, trace = FALSE, mgcv.tol = 1e-07, mgcv.half = 15, rank.tol = 1.49011611938477e-08, nlm = list(ndigit=7, gradtol=1e-06, stepmax=2, steptol=1e-04, iterlim=200, check.analyticals=0), optim = list(factr=1e+07), newton = list(conv.tol=1e-06, maxNstep=5, maxSstep=2, maxHalf=30, use.svd=0), outerPIsteps = 0, idLinksBases = TRUE, scalePenalty = TRUE

, efs.lspmax = 15, efs.tol = 0.1, keepData = FALSE , scale.est = "fletcher", edge.correct = FALSE) ))
```

Agora que já ajustamos algumas opções dos algoritmos 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 numero 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 comentado o que ele faz. Então, podemos ver que estamos modelando a distribuição da Araponga, usando o conjunto de pseudo-ausências espacialmente estruturadas (bm.procnias100disk) a partir de 3 algoritmos (`GLM`, `GAM`, `SRE`). Vamos fazer 3 rodadas para cada algoritmo e conjunto de pseudo-ausência, 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 aula prática de avaliação).

```
procnias1model = BIOMOD_Modeling(
  bm.procnias100disk, #objeto das pseudo-ausências
```

```
        models = c("GLM", "GAM","SRE"), #algoritmos de escolha

        models.options = modelo_op, #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 de-
sempenho dos modelos

        SaveObj = TRUE, #se os modelos serão salvos ou não

        rescal.all.models = F,

        do.full.models = FALSE,

        modeling.id = "procnias_envelope_regres")
```

Quando você rodar esse código, você perceberá que na tela de console, uma série de informações começarão a ser geradas. 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-ausência (`PA1`) está sendo usada para a primeira rodada da modelagem (`RUN1`) para cada algoritmo (`GLM`, `GAM`, `SRE`), 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=GLM ( quadratic with no interaction )

            Stepwise procedure using AIC criteria

            selected formula : procnias ~ I(WC_bio3_lonlat^2) +
WC_bio3_lonlat + I(WC_bio7_lonlat^2) +
```

```
    WC_bio15_lonlat + I(WC_bio15_lonlat^2) + I(WC_alt_lonlat^2)
<environment: 0x0000019dc26103a8>

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

Model=GAM
       GAM_mgcv algorithm chosen
      Automatic formula generation...
      > GAM (mgcv) modelling...
      Evaluating Model stuff...
      Evaluating Predictor Contributions...

Model=Surface Range Envelop
      Evaluating Model stuff...
      Evaluating Predictor Contributions...


-=-=-=-=-=-=-= procnias_PA1_RUN2
Model=GLM ( quadratic with no interaction )
      Stepwise procedure using AIC criteria
      selected formula : procnias ~ I(WC_bio3_lonlat^2) +
WC_bio3_lonlat + I(WC_bio7_lonlat^2) +
    WC_bio7_lonlat
<environment: 0x0000019dcce79d10>

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

Model=GAM
       GAM_mgcv algorithm chosen
      Automatic formula generation...
      > GAM (mgcv) modelling...
      Evaluating Model stuff...
      Evaluating Predictor Contributions...
```

```
Model=Surface Range Envelop

      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-ausência (`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 pseudo-ausência + rodada + algoritmo) e se algum falhou

```
procnias1model
```

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 você tem duas pastas a mais: '.BIOMOD_DATA' e 'models'. Em 'models', você verá o arquivo de cada um dos 36 modelos gerados.

Agora que temos os nossos modelos gerados, 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_procnias1=get_variables_importance(procnias1model)
```

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_procnias1
```

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-ausência (PA).

# salvando um objeto com os valores médios de importância de cada variável para cada algoritmo

```
var_import_procnias1=apply(var_import_procnias1, c(1,2), mean)
```

# obtendo os valores médios de importância de cada variável para cada algoritmo

```
var_import_procnias1
```

```
# Salvando os valores de importancia das variaveis em um arquivo
.csv

write.csv(var_import_procnias1,paste0("./Outputs/", "_", "var_im-
port_procnias1.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. Ou seja, será que a Precipitação Anual tem uma relação positiva com a Araponga? Isto é, áreas com maiores precipitações são as áreas com maior adequabilidade para a Araponga? Só com o teste de importância das variáveis não conseguimos obter essa informação, que é crucial para interpretarmos nossos resultados. Para isso, precisamos gerar as curvas respostas – a relação entre a adequabilidade e cada variável preditora.

- Carregando os modelos individuais que foram gerados acima

```
procnias1_glm=BIOMOD_LoadModels(procnias1model, models = 'GLM')

procnias1_gam=BIOMOD_LoadModels(procnias1model, models = 'GAM')

procnias1_sre=BIOMOD_LoadModels(procnias1model, models = 'SRE')
```

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.

```
glm_responsecurve= biomod2::response.plot2(models = procnias1_glm, Data = get_formal_data(procnias1model, 'expl.var'),show.variables = get_formal_data(procnias1model, 'expl.var.names'), do.bivariate = F, save.file = 'jpeg', ImageSize = 720, name= "GLM_curva_resposta", fixed.var.metric = 'median', legend = F, display_title = T, data_species = get_formal_data(procnias1model, 'resp.var') , plot = T)
```

```
gam_responsecurve= biomod2::response.plot2(models = procnias1_gam, Data = get_formal_data(procnias1model, 'expl.var'),show.variables = get_formal_data(procnias1model, 'expl.var.names'), do.bivariate = F, save.file = 'jpeg', ImageSize = 720, name= "GAM_curva_resposta", fixed.var.metric = 'median', legend = F, display_title = T, data_species = get_formal_data(procnias1model, 'resp.var') )
```

```
sre_responsecurve= biomod2::response.plot2(models = procnias1_sre, Data = get_formal_data(procnias1model, 'expl.var'),show.variables = get_formal_data(procnias1model, 'expl.var.names'), do.bivariate = F, save.file = 'jpeg', ImageSize = 720, name= "SRE_curva_resposta", fixed.var.metric = 'median', legend = F, display_title = T, data_species = get_formal_data(procnias1model, 'resp.var') )
```

Os comandos acima, salvaram na sua pasta de diretório do R os arquivos jpeg com as curvas de resposta para cada um dos algoritmos acima. Se quiser visualizar essas curvas no R, rode os comandos abaixo.

```
biomod2::response.plot2(models = procnias1_glm, Data = get_formal_data(procnias1model, 'expl.var'),show.variables = get_formal_data(procnias1model, 'expl.var.names'), do.bivariate = F, name= "GLM_curva_resposta", fixed.var.metric = 'median', legend = F, display_title = T, data_species = get_formal_data(procnias1model, 'resp.var') )

biomod2::response.plot2(models = procnias1_gam, Data = get_formal_data(procnias1model, 'expl.var'),show.variables = get_formal_data(procnias1model, 'expl.var.names'), do.bivariate = F, name= "GAM_curva_resposta", fixed.var.metric = 'median', legend = F, display_title = T, data_species = get_formal_data(procnias1model, 'resp.var') )

biomod2::response.plot2(models = procnias1_sre, Data = get_formal_data(procnias1model, 'expl.var'),show.variables = get_formal_data(procnias1model, 'expl.var.names'), do.bivariate = F, name= "SRE_curva_resposta", fixed.var.metric = 'median', legend = F, display_title = T, data_species = get_formal_data(procnias1model, '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 4
```