Vinicius Tonetti

Mestrando no PPG em Ciências Biológicas (Zoologia), Instituto de Biociências da UNESP Rio Claro.

Áreas de interesse: conservação, ecologia e taxonomia de aves neotropicais.

Exercícios

Currículo Lattes

Trabalho Final

Proposta A

Criação de uma função que organiza tabelas com histórico de detecção em modelos de ocupação.

O protocolo mais básico usado para a estimação da ocupação de espécies consiste em visitas a locais previamente determinados para constatar a presença ou ausência de individuo(s), ou indícios de indivíduos, da espécie de interesse. Atualmente existe uma série de modelos (e.g. Mackenzie et al. 2002) que estimam a ocupação de um táxon levando em conta a detecção imperfeita, principalmente por meio de visitas repetidas aos mesmos locais amostrados, onde são gerados históricos de detecção da espécie. Cada histórico de detecção é formado por zeros (indicando ausência de detecção da espécie no local visitado) e 1 (indicando detecção). Quando o estudo envolve comunidades são geradas para cada local amostrado tabelas com o histórico de detecção para as diversas espécies. O número de locais amostrados, assim como o número de visitas a cada local e o número de espécies detectadas em cada amostra, pode variar grandemente, resultando em tabelas que podem conter uma enorme quantidade de dados.

Um tipo de tabela padrão gerada em cada visita a determinado local em um estudo de ocupação de comunidade é uma tabela com uma coluna indicando a espécie, uma coluna com o histórico de detecção (0 e 1), sendo que em alguns casos pode ser uma série de 0 e 1 (ver Farnsworth et al. 2002), outra(s) coluna(s) com variáveis climáticas coletadas no momento da amostragem e uma série de colunas com variáveis ambientais do local amostrado.No final do estudo seriam geradas diversas tabelas para cada visita a cada local amostrado com todas essas informações.

A proposta seria criar uma função que busque o nome de cada uma das espécies registradas em cada visita nas tabelas resultantes e guardasse essas informações em tabelas para cada local e gerasse também tabelas para cada uma das espécies em todos os locais.Essas tabelas seriam guardadas em um array. Assim as informações do histórico de detecção, variáveis ambientais no momento da amostragem e variáveis ambientais dos locais amostrados seriam acessadas de maneira mais fácil, auxiliando na modelagem da ocupação e probabilidade de detecção para cada espécie. Poderiam ser gerados também histogramas para cada local amostrado, indicando a frequência de detecção de cada espécie no local e histogramas para cada espécie indicando a frequência de presença e

ausência no histórico de detecção para cada local. A função possuirá argumentos para a escolha do tipo de tabelas geradas (se apenas tabelas por localidade, ou espécie, ou data), para a truncagem de certos históricos de detecção e para observações faltantes.

Referências:

Farnsworth et al. 2002 - A removal model for estimating detection probabilities from point-count surveys.

Mackenzie et al. 2002 - Estimating site occupancy rates when detection probabilities are less than one.

Proposta B

Criação de uma função que testa a colinearidade, ou multicolinearidade, entre variáveis.

Em estudos ecológicos que envolvem testes de regressão é comum a coleta de muitos dados simultâneos. No entanto, na maior parte das vezes, esse excesso de dados pode prejudicar os modelos.

A função proposta analisaria uma tabela com diversos dados coletados como objeto de entrada e executaria testes de correlação entre cada duas variáveis. Como objeto de saída a função retornaria uma tabela mostrando os resultados desses testes de correlação entre cada duas variáveis e geraria gráficos para que esses efeitos possam ser visualizados. A seleção de quais variáveis usar no modelo ficaria por conta do usuário. Para poupar esforço amostral e não coletar uma quantidade grande de dados que não seriam utilizados nas análises,o usuário poderia medir um número grande de variáveis para poucas amostras e então, com o auxílio da função, selecionar quais variáveis seriam coletadas no restante do estudo.

> A proposta A me parece bem mais interessante que a B,especialmente a possibilidade se utilizar isso para comparar espécies e habitats. Acho que vc deveria seguir com ela —- *Vitor Rios*

Código da função org.table (Proposta A)

```
#Função que organiza tabelas de ocupação. Argumentos x = entrada de dados
(tabela); tab.local = Lógico, se TRUE exibe um
#array com todas as espécies registradas em cada localidade; tab.data =
Lógico, se TRUE retorna um array com as espécies registradas por data;
#tab.data = Lógico, se true exibe dois tipos de gráficos de barras: um para
cada local, mostrando o número de detecções de cada espécie naquele local,
#o outro gráfico é um gráfico de barras mostrando o número de detecções e
não-detecções (com as barras pareadas) de todas as espécies amostradas, em
```

```
todos os locais e todas datas.

org.table <- function(x,tab.local=FALSE,tab.data=FALSE,gráficos=FALSE)
        {
        x[x==""] <- NA                        ## Como o argumento
"blank.lines.skip" da função read.table não estava ocultando as linhas em
branco, usei essas três primeiras linhas da
        z <- x[rowSums(is.na(x))!=ncol(x),]  ## função para excluir as
linhas em branco da tabela de entrada.
        y<- data.frame(z)
    if(colnames(y)[1]!="especie")            ## Para que a função execute de
maneira correta é necessário que a primeira coluna se chame "especie", com o
primeiro caracter em caixa baixa e sem acento na palavra especie.
        {                                    ## Caso essa premissa não seja
cumprida a função retorna um warning com essa mensagem.
        warning(" Atenção! Sua tabela deve possuir a primeira coluna chamada
'especie'. Caso essa coluna exista, confirme que seu nome não possui acento
e que o primeiro caracter está em caixa baixa. ")
        }
    if(colnames(y)[2]!="data")               ## Para que a função execute de
maneira correta é necessário que a segunda coluna se chame "data", com o
primeiro caracter em caixa baixa.
        {                                    ## Caso essa premissa não seja
cumprida a função retorna um warning com essa mensagem.
        warning(" Atenção! Sua tabela deve possuir a segunda coluna chamada
'data'. Caso essa coluna exista, confirme que o primeiro caracter está em
caixa baixa. ")
        }
    if(colnames(y)[3]!="local")              ## Para que a função execute de
maneira correta é necessário que a terceira coluna se chame "local", com o
primeiro caracter em caixa baixa.
        {                                    ## Caso essa premissa não seja
cumprida a função retorna um warning com essa mensagem.
        warning(" Atenção! Sua tabela deve possuir a terceira coluna chamada
'local'. Caso essa coluna exista, confirme que o primeiro caracter está em
caixa baixa. ")
        }
        data <- unique(y$data)                          ## Essa série de
procedimentos é para criar uma tabela para cada data diferente. Primeiro
atribuí-se cada valor de data indivídual a um objeto chamado "data",
        n.data <- sum(table(data))                      ## Depois
atribuí-se a um objeto chamado "n.data" o número de valores diferentes do
objeto data
        tabelas_datas <- list(rep(NA,n.data))           ## Cria-se listas
vazias, preenchidas com NA, com o  número de objetos vazios igual ao valor
do objeto n.data
        for(i in 1:n.data)                              ## Aqui são
feitos os ciclos para preencher os objetos da lista vazia "tabelas_datas", o
número de obejtos dentro da lista vazia é igual ao número de datas com
valores diferentes no objeto de entrada, na coluna data.
        {
```

```
        tabelas_datas[i] <- list(y[y$data==data[i],])      ## A lista vazia
criada na linha anterior é preenchida com os valores do objeto de entrada
para cada valor de data diferente.
        }
        spp <- unique(y$especie)                           ## Essa série de
procedimentos é para criar uma tabela para cada espécie diferente. Primeiro
atribuí-se cada valor de espécie indivídual a um objeto chamado "spp",
        n.spp <- sum(table(spp))                           ## Depois
atribuí-se a um objeto chamado "n.spp" o número de valores diferentes do
objeto spp
        tabelas_especies <- list(rep(NA,n.spp))           ## Cria-se listas
vazias, preenchidas com NA, com o  número de objetos vazios igual ao valor
do objeto n.spp
        for(i in 1:n.spp)                                  ## Aqui são
feitos os ciclos para preencher os objetos da lista vazia
"tabelas_especies", o número de obejtos dentro da lista vazia é igual ao
número de espécies com valores diferentes no objeto de entrada, na coluna
especie.
        {
        tabelas_especies[i] <- list(y[y$especie==spp[i],]) ## A lista vazia
criada na linha anterior é preenchida com os valores do objeto de entrada
para cada valor da coluna especie diferente.
        }
        locais <- unique(y$local)                          ## Essa série de
procedimentos é para criar uma tabela para cada local diferente. Primeiro
atribuí-se cada valor de local indivídual a um objeto chamado "locais",
        n.locais <- sum(table(locais))                     ## Depois
atribuí-se a um objeto chamado "n.locais" o número de valores diferentes do
objeto locais
        tabelas_locais <- list(rep(NA,n.locais))           ## Cria-se listas
vazias, preenchidas com NA, com o  número de objetos vazios igual ao valor
do objeto n.locais
        for(j in 1:n.locais)                               ## Aqui são
feitos os ciclos para preencher os objetos da lista vazia
"tabelas_especies", o número de obejtos dentro da lista vazia é igual ao
número de espécies com valores diferentes no objeto de entrada, na coluna
especie.
        {
        tabelas_locais[j] <- list(y[y$local==locais[j],])  ## A lista vazia
criada na linha anterior é preenchida com os valores do objeto de entrada
para cada valor da coluna local diferente.
        }
    if(gráficos==TRUE)       ## Caso o valor do argumento "gráficos" seja
igual a TRUE, segue-se com os procedimentos abaixo. Primeiramente cria-se o
gráfico com os valores de detecção e não-detecção de cada espécie em todos
os locais e todas as datas
      {
          nomes.spp<- as.character(spp)  ## Primeiro os nomes das espécies
são salvos como caracteres, para serem exibidos nas tabelas
        detec <- rep(NA,n.spp)         ## Cria-se listas vazias para serem
```

```
preenchidas com as detecções
        n.detec <- rep(NA,n.spp)       ## Cria-se listas vazias para serem
preenchidas com as não-detecções
        for(i in 1:n.spp)              ## Cria-se ciclos para preencher as
listas vazias com o número de detecções (detec) e não-detecções (n.detec)
        {
        detec[i] <- na.omit(sum(tabelas_especies[[i]][,grep(pattern =
"historico",colnames(tabelas_especies[[i]]))])) ## Soma-se o número de
detecções de cada uma das espécies amostradas
        n.detec[i] <-
na.omit(sum(table(which(tabelas_especies[[i]][,grep(pattern =
"historico",colnames(tabelas_especies[[i]]))] == 0))))## Soma-se o número de
não-detecções de cada uma das espécies amostradas
        }
        mx <-
matrix(c(detec,n.detec),nrow=n.spp,ncol=2,dimnames=list(c(nomes.spp),c("dete
c","n.detec"))) ## Cria-se uma matriz, com as linhas para cada espécie e
duas colunas, detecções e não-detecções de cada uma delas
        x11() ## Abre-se uma janela gráfica
        par(cex=0.8,mar=(c(13,5,5,5)+ 0)) ## Alguns parâmetros da janela são
ajustados
barplot(t(mx),beside=T,las=3,ylim=c(0,max(c(n.detec,detec))),legend.text=c("
detecção","não-detecção"),main="Número de detecções e não-detecções das
espécies amostradas") ## Um gráfico de barras é plotado com duas barras para
cada espécie: detecções e não detecções. O valor limite do eixo y é
atribuído como sendo o valor máximo das detecções e não-detecções
        }
      if (gráficos== TRUE &
sum(table(colnames(tabelas_locais[[1]])[grep(pattern =
"historico",colnames(tabelas_especies[[1]]))])) == 1) ## Depois do gráfico
com o número de detecção e não-detecção total de todas as espécies em todos
os locais são criados os gráficos de barras com o valor de detecção de cada
espécie em cada local,
       {
## para que a função execute de maneira correta, foi necessário estabelecer
dois procedimentos. O primeiro deles, estabelecido nesse "if", é aplicado
para tabelas em que o número de colunas com históricos de detecção é igual a
1.
        Local <- paste("Número de detecções das espécies presentes no
local",1:n.locais)                                              ## Primeiro
é criado um ciclo para os nomes contidos na legenda de cada gráfico para
cada local com a função paste.
        for(i in 1:n.locais)    ## O número de gráficos gerados nessa etapa
é igual ao número de locais diferente na tabela de entrada (n.locais)
        {
        x11()  ## No início do ciclo é aberta uma janela gráfica para que os
gráficos gerados sejam plotados
        par(cex=0.8,mar=(c(13,5,5,5)+ 0)) ## Alguns dos parâmetros dessa
janela gráfica são ajustados.
        barplot(tabelas_locais[[i]][,grep(pattern =
"historico",colnames(tabelas_especies[[i]]))],names.arg=tabelas_locais[[i]][
```

```
,1],width=0.1,main=Local[i],las=3) ## Cada gráfico de cada local é plotado
seguindo os parâmetros estabelecidos nessa linha.
        }
         }
    if (gráficos==TRUE &
sum(table(colnames(tabelas_locais[[1]])[grep(pattern =
"historico",colnames(tabelas_especies[[1]]))])) > 1) ## para que a função
execute de maneira correta, foi necessário estabelecer dois procedimentos. O
segundo deles, estabelecido nesse "if", é aplicado para tabelas em que o
número de colunas com históricos de detecção é maior que 1.
        {
       Local <- paste("Número de detecções das espécies presentes no
local",1:n.locais)  ## Primeiro é criado um ciclo para os nomes contidos na
legenda de cada gráfico para cada local com a função paste.
       for(i in 1:n.locais)  ## O número de gráficos gerados nessa etapa é
igual ao número de locais diferente na tabela de entrada (n.locais)
       {
       x11()   ## No início do ciclo é aberta uma janela gráfica para que os
gráficos gerados sejam plotados
       par(cex=0.8,mar=(c(13,5,5,5)+ 0)) ## Alguns dos parâmetros dessa
janela gráfica são ajustados.
       barplot(apply(tabelas_locais[[i]][,grep(pattern =
"historico",colnames(tabelas_especies[[i]]))],1,sum),names.arg=tabelas_locai
s[[i]][,1],width=0.1,main=Local[i],las=3) ## Cada gráfico de cada local é
plotado seguindo os parâmetros estabelecidos nessa linha.
      }
    }
    if(tab.local==TRUE & tab.data==TRUE)  ## A partir desse momento são
estabelecidas as condições em que cada uma das tabelas, ou gráficos, devem
ser mostradas na tela.
        {
return(c(array(tabelas_especies),array(tabelas_locais),array(tabelas_datas))
) ## A primeira condição é que quando os argumentos tab.local e tab.data
sejam igual a TRUE, as tabelas com cada local diferente e com cada data
diferente sejam mostradas na tela. A tabela para cada espécie diferente é
sempre mostrada na tela
        }
    if(tab.local==TRUE & tab.data==FALSE) ## A segunda condição é que quando
os argumentos tab.local é igual a TRUE e tab.data igual a FALSE as tabelas
com cada local diferente  seja mostrada na tela e com cada data diferente
não. A tabela para cada espécie diferente é sempre mostrada na tela
        {
        return(c(array(tabelas_especies),array(tabelas_locais)))
        }
    if(tab.local==FALSE & tab.data==TRUE) ## A terceira condição é para
mostrar a tabela com as datas na tela e não mostrar as tabelas com os locais
diferentes. A tabela para cada espécie diferente é sempre mostrada na tela
        {
        return(c(array(tabelas_especies),array(tabelas_locais)))
        }
```

```
    if(tab.local==FALSE & tab.data==FALSE) ## A quarta condição é para não
mostrar na tela nem a tabela para cada local diferente nem a tabela para
cada data diferente.
        {
        return(array(tabelas_especies))
        }
      }
```

Help da função org.table

```
org.table                    package: desconhecido                  R
Documentation


Organizadora de tabelas


    Description:

A função org.table pode ser utilizada para organizar, facilitar o acesso, e
facilitar a visualização de dados contidos em tabelas de ocupação. A partir
de dados brutos de tabelas de ocupação, a função busca o nome de cada
espécie presente ao longo de toda a tabela e guarda as linhas com as
informações relativas a essa espécie (como data, local, histórico de
detecção, entre outras possíveis variáveis ambientais coletadas no momento
da amostragem) em tabelas separadas para cada espécie. A função possui
argumentos para executar o mesmo procedimentos com cada local e data.


    Usage:

org.table (x,tab.local=FALSE,tab.data=FALSE,gráficos=FALSE)

    Arguments:

x              tabela. A tabela deve apresentar a primeira coluna com o nome
"especie", segunda coluna chamada "data" e a terceira coluna chamada
"local". Deve-se tomar o cuidado para que o nome das colunas sejam escritos
exatamente dessa maneira, sem acentos e em caixa baixa. É preferível que o
arquivo esteja em formato .csv

tab.local      Lógico. Se local for TRUE é gerado um array para todas as
espécies em todas as datas para cada local.

tab.data       Lógico. Se tab.data for TRUE é gerado um array para todas as
espécies em todos os locais na mesma data.

gráficos       Lógico. Se gráficos for TRUE são gerados dois tipos de
gráficos de barras: um para cada local, indicando o número de detecção de
```

```
cada espécie e outro para todas espécies em todos os locais, indicando a
frequência de detecção e não-detecção de cada uma delas.

    Details:

Função que organiza, e facilita a visualização, de dados contidos em tabelas
de ocupação. Nessas tabelas são registrados os históricos de detecção
(indicado por um) e não-detecção (indicado por zero) das espécies alvo nos
locais de amostragens. Esses históricos de detecção/não-detecção obtidos por
meio de visitas repetidas aos mesmos locais podem ser utilizados para se
estimar a probabilidade de detecção de um táxon (Mackenzie et al. 2006). Há
situações em que, para um mesmo local e data, é possível obter mais de um
histórico de detecção (e.g. Alldredge et al. 2007). Essa função pode ser
utilizada para facilitar o acesso a esses dados de históricos de ocupação
para cada espécie, assim como facilitar sua visualização, para a modelagem
da probabilidade de detecção.A geração dos gráficos permite, para uma
análise exploratória inicial, visualizar o número de detecções de cada
espécie em cada local, indicando quais espécies mais frequentes em cada
local, assim como visualizar os números de detecções e não-detecções de cada
espécie em todos os locais, indicando quais espécies são mais e menos
detectáveis. Todas essas informações podem ser utilizadas para traçar co-
relações entre cada táxon e o ambiente amostrado. A função requer que a
tabela de entrada possua a primeira coluna chamada "especie", a segunda
chamada "data" e a terceira "local", os nomes devem ser escritos em caixa
baixa e sem acento.


    Value:

A função retorna arrays com todas as váriáves para cada uma das espécies,
para cada um dos locais e para cada data. Pode retornar também gráficos, um
para cada local indicando o número de detecções de cada espécie naquele
local e um para cada espécie em todos os locais, indicando o número de
detecções e não-detecções para cada espécie.


    Author(s):

Vinicius Rodrigues Tonetti
contato: vrtonetti@gmail.com


Examples:

x <- read.table("dados_brutos.csv",header=TRUE,sep=";")
org.table(x, tab.local=TRUE, tab.data=TRUE, gráficos=TRUE)


References:
```

Alldredge, Mathew W.; Pollock, Kenneth H.; Simons, Theodore R.; Colazzo, Jaime A.; Shriner, Susan A. & Johnson, D. H. 2007. Time-of-detection Method for Estimating Abundance from Point-Count Surveys. The American Ornithologist's Union, 124:2-11.

Arquivos

função org.table:

funcao_org_table.r

help da função:

help_-_org_table.r

arquivo utilizado no exemplo:

dados_brutos.csv