Gabriella Leal

Doutoranda em Ecologia Aplicada pela ESALQ-USP, Laboratório de Ecologia de Vertebrados, IB/USP.

No momento, tenho especial interesse por ecologia de comunidades e ecologia aplicada. No doutorado, investigo como taxocenoses de serpentes respondem a substituição do seu habitat natural por plantações de eucalipto e como sistemas agroflorestais podem ser manejados para conservar a maior diversidade possível de serpentes, considerando as dimensões taxonômica, funcional e filogenética da diversidade.

I. Meus exercícios

exericios_1_ao_3.r

5_1.r

5_2.r

5_3.r

exercicios_7b.r

exercicios_aula_8.r

9_2.r

II. Trabalho Final

II.I. Proposta A: Calculando sinal filogenético para múltiplos atributos: estatística – K

As similaridades ecológicas entre as espécies podem estar relacionadas à sua história evolutiva compartilhada e, neste caso, são detectadas pelo sinal filogenético entre a característica ou atributo de interesse e a distância filogenética entre as espécies. O sinal filogenético pode ser definido como a covariação entre diferenças na medida de um atributo entre espécies e a distância filogenética que as separa (Blomberg et al. 2003). Um sinal filogenético forte (i.e. próximo de 1 ou > 1, no caso da estatística-K) significa que o atributo de interesse é compartilhado por espécies evolutivamente próximas, enquanto que espécies filogeneticamente distantes são menos similares em seus atributos, logo o sinal filogenético é fraco (i.e. próximo de 0) (Pagel, 1999; Blomberg et al., 2003). Tal informação pode ser importante na tomada de decisão sobre quais atributos escolher para calcular diversidade funcional, além de ser útil também em análises de diversidade filogenética baseadas em atributos conservados evolutivamente (por exemplo, ao verificar padrões de agregação ou sobre-dispersão filogenética para inferir processos de montagem de comunidades ecológicas (Webb, 2000)).

Embora haja um grande número de estudos que utilizam muitos atributos das espécies, o cálculo do sinal filogenético dos mesmos é feito para cada atributo individualmente. As funções que calculam sinal filogenético, como phylosignal (pacote picante), fitContinuous e fitDiscrete (pacote geiger), aceitam como entrada somente um único atributo (em geral, um vetor contendo o nome das espécies

e as medidas para um atributo). A função phylosignal estima o sinal filogenético com base na estatística-K e retorna o valor estimado e sua significância comparada a um cenário nulo. Porém, é necessário utilizá-la repetidas vezes para diferentes atributos. Logo, a primeira proposta objetiva formular uma função que aceite um data frame com muitos atributos, sendo um por coluna, e retorne o sinal filogenético e os parâmetros estimados pela função phylosignal para todos os atributos do data frame. A segunda proposta pretende incorporar a primeira e incluir outra possibilidade de análise estatística para cálculo do sinal filogenético, o λ de Pagel (Pagel, 1999). A função fitContinuous permite o cálculo dessa estatística, porém não verifica se o valor estimado difere do esperado em um cenário nulo. Desse modo, na segunda proposta pretendo possibilitar o cálculo do sinal filogenético pelas estatísticas K ou λ para diferentes atributos ao mesmo tempo, retornando o sinal filogenético, os demais parâmetros estimados por essas funções e o histograma para verificar se a estatística λ observada diferencia do cenário nulo. Neste caso, optei por gerar um modelo nulo que aleatorize os terminais da filogenia, embaralhando o parentesco entre as espécies, mas mantendo a distância entre os ramos da árvore filogenética, utilizando a função tipShuffle (pacote picante).

Entrada

x = data frame contendo as medidas dos atributos para cada espécie (classe numeric). Data frame contendo nome das espécies na primeira coluna e medidas de um único atributo em cada coluna subsequente.

tree = árvore filogenética do grupo de interesse (classe phylo)

model = índice para calcular sinal filogenético (classe character). Aceita os índices “lambda”, “BM” e “K.statistic” fornecidas pelas funções fitContinuous do pacote geiger e phylosignal do pacote picante, respectivamente.

Verificando os parâmetros

x é um dado contínuo > 0? Se não, avisar: “x precisa ser da classe numeric ”

x contém NA? Se sim, avisar: “análise não pode ser feita com NA”

tree é da classe phylo? Se não, avisar: “objeto deve ser da classe phylo.”

Nome das espécies em x difere do nome das espécies em tree? Se sim, avisar: “impossível fazer a análise. Nome das espécies deve ser idêntico em x e tree. Você pode utilizar a função tiplabels, do pacote ape, para corrigir essa condição.”

Pseudo-código

1. Cria objeto dados para guardar a leitura da planilha com os atributos.
2. Cria objeto phy.sign para guardar o retorno da função phylosignal {ape}
3. Cria objeto result com valor NULL
4. Inicia um ciclo FOR com contador i que lê dados da segunda coluna até o número de colunas de dados

- Cria objeto sp.trait com duas colunas, a primeira do objeto dados e a do contador i
- Verifica se modelo solicitado em x é “K.statistic”
- Se for: calcula sinal filogenético pela função phylosignal e atribui resultado ao objeto phy.signi
- Em result, guarda o resultado dos ciclos do FOR usando a função append. Result será uma lista contendo phy.sign

Saída

Sinal filogenético estimado para cada atributo (estatística K e os parâmetros estimados)

II.II. Proposta B: Calculando sinal filogenético para múltiplos atributos: estatísticas – K e λ

Entrada

Idem à proposta A.

Verificando os parâmetros

Idem à proposta A.

Pseudo-código

1. Cria objeto dados para guardar a leitura da planilha com os atributos
2. Cria objeto phy.sign para guardar o retorno da função phylosignal {ape}
3. Cria objeto continuous para guardar o retorno da função fitContinuous {geiger}
4. Cria objeto result com valor NULL
5. Inicia um ciclo FOR com contador i que lê dados da segunda coluna até o número de colunas de dados

- Cria objeto sp.trait com duas colunas, a primeira do objeto dados e a do contador i
- Verifica se modelo solicitado em x é “K.statistic”
- Se for: cria objeto sp.trait.k para guardar os dados usados nessa análise, calcula sinal filogenético pela função phylosignal e atribui resultado ao objeto phy.signi
- Verifica se modelo solicitado em x é “lambda”
- Se for: cria objeto sp.trait.c para guardar os dados usados nessa análise, calcula sinal filogenético pela função fitContinuous e atribui resultado ao objeto continuousi
- Em result, guarda o resultado dos ciclos do FOR usando a função append. Result será uma lista contendo os seguintes objetos de classe lista: sp.trait.k, phy.sign, sp.trait.c e continuous.

1. 6. Cria objeto null.distrib com valor NULL
2. 7. Inicia um ciclo FOR com contador m que vai de 1 até 999

- Inicia um ciclo FOR com contador l que lê sp.trait.c do primeiro até o número de objetos desta lista
- Calcula sinal filogenético pela função fitContinuous de uma árvore filogenética com os terminais da filogenia embaralhados pela função tipShuffle{Picante}, usando dado = l e atribua resultado ao objeto null.phylosig[l]
- Cria objeto times[m] para guardar os 999 valores de null.phylosig[l]
- Encerra o ciclo guardando todos os resultados de times em null.distrib, usando a função append. Null.distrib será uma lista de tamanho l com vetores dos valores nulos de sinal filogenético para cada atributo de l

1. 8. Cria objeto Hist com valor NULL
2. 9. Inicia ciclo FOR com contador j que lê null.distrib do primeiro ao último objeto

- Plota histograma de j, usando a função hist e atribua resultado ao objeto freq.null
- Sobrescreva Hist guardando os resultados de freq.null

1. 10. Cria objeto line com valor NULL
2. 11. Entra num ciclo FOR com contador a que lê continuous do primeiro ao último objeto,

pegando de cada objeto dessa lista somente o primeiro valor

- Plota uma linha de a, usando a função abline e atribua ao objeto estim.phylosig
- Sobrescreva line com o resultado de estim.phylosig

1. 12. Cria objeto limit com valor NULL
2. 13. Inicia um ciclo FOR com contador b que lê null.distrib do primeiro ao último objeto

- Calcula o percentil de 95% da distribuição nula e atribua ao objeto q[b]
- Sobrescreva limit com os resultados de q

Saída

Lista contendo:

- Sinal filogenético estimado para cada atributo (estatística K ou lambda e os parâmetros estimados. Para mais detalhes, ver retorno das funções phylosignal e fitContinuous)
- Histograma com a distribuição nula e a estimativa do sinal filogenético para verificar se o valor estimado encontra-se fora do percentil de 95% da distribuição nula (assumindo teste unicaudal e alpha = 0.05)

Referências bibliográficas

Blomberg, S. P., Garland, T. & Ives, A. R. 2003. Testing for phylogenetic signal in comparative data: Behavioral traits are more labile. *Evolution* 57:717-745.

Pagel, M. 1999. Inferring the historical patterns of biological evolution. *Nature* 401:877-884.

Webb, C. O. 2000. Exploring the phylogenetic structure of ecological communities: An example for rain forest trees. *American Naturalist* 156:145-155.

> *— Alexandre Adalardo de Oliveira 2019/06/12 11:03*
>
> Olá Gabriella,
>
> A proposta está bem conduzida e clara. Note que o valor de K, quando é 1, significa que os taxa estão relacionados, como esperado por um processo evolutivo nulo baseado em movimento browniano. Somente se o valor for maior que 1 é que temos conservantismo (“sinal filogenético forte”), ou seja, os atributos de espécies aparentadas são mais parecidos do que esperado pelo grau de parentesco. Acho que o problema da primeira proposta é que ela já está implementada no prórprio pacote picante com a função `multiPhylosignal`, se estivesse usando outro pacote, como o `ape`, poderia até considerar a proposta, mas, como está usando o pacote, não há muito sentido.
>
> Minha sugestão é que vá pela sua segunda propostas, apesar de estar implementada em outros pacotes, isso não importa, pode desenvolver a sua própria. Deixe a opção para que o usuário escolha entre K, lambda ou ambos (essa opção não está lá proposta), e faça o teste de signficância como sugere para cada um dos atributos do data frame. Bom trabalho.

> — *Gabriella Leal 2019/07/01 17:23*
>
> Oi Alexandre,
>
> Fiz algumas modificações na proposta inicial da segunda função. Porque notei que a função fitContinuous calcula o lambda por log-verossimilhança, optei por fazer o cálculo de uma taxa de verossimilhança ao invés de plotar um histograma desses valores. Considerei distribuição qui-quadrado com grau de liberdade = 1 para testar se o valor do sinal filogenético observado é diferente do cenário nulo. Reduzi o cenário nulo para 99 vezes para rodar mais rápido.
>
> Para testar cada linha do script usei como base o conjunto de dados caudata do pacote geiger e simulei dados para mais atributos, criando um dataframe. Esse conjunto de dados está explícito no exemplo do help e consegui obter em cada linha do script exatamente o que queria. Porém, ao testá-lo na minha função, retorna valor nulo. Passei dois dias tentando descobrir porquê e não consegui. Gostaria de entender o que está errado e se puder me dar retorno, te agradeço.

II.III Código da função

```
# criar a função phyloSignal.dataframe que contém na entrada: x, tree, model
e hist

phyloSignal.dataframe <- function(x, tree, model = c("lambda",
"K.statistic"))


{

# TESTE DE PREMISSAS #

# primeira: checar se cada coluna de x é numérica e se tem valor 0
{
  if (class(x[,c(ncol(x))])  != "numeric") {
   stop("x deve ser numeric") }
  if (any(x  == 0)) {
    stop("x deve ser > 0") }

# segunda: checar se tem NA

if (any(is.na(x[,c(ncol(x))]) == TRUE))
{ stop("remova NA") }

# terceira: checar se objeto phylo é da classe phylo

 if (class(tree) != "phylo")
{ stop("tree deve ser da classe phylo") }
```

```
# Para guardar resultado do ciclo FOR: cria uma lista de NA do tamanho do
número de colunas de x e atribui ao
objeto sp.trait

sp.trait <- list(rep(NA, ncol(x)))

# inicia ciclo FOR com contador i que lê de 1 até o número de colunas de x.
# cria trait que é um vetor guarda os valores de cada coluna de x
# atribui cada valor dentro de trait ao nome da espécie correspondente,
usando as funções names e row.names
# sobrescreve sp.trait criando uma lista que contém trait

   for(i in 1:ncol(x)) {
    trait <- x[ ,i]
      names(trait) <- row.names(x)  # porque as funções usadas para calcular
sinal filogenético pedem vetores
      nomeados
    sp.trait[[i]] <- trait }

# retira o primeiro dataframe da lista porque possui duas colunas com nome
das espécies e não é de interesse
sp.trait[[1]] <- NULL



# se o modelo solicitado for k.statistic, cria objeto phy.sign: uma lista de
NA do tamanho de sp.trait
# inicia ciclo FOR com contador j que lê de 1 até o tamanho da lista
sp.trait
# calcula sinal filogenético com a função phylosignal, usando cada vetor de
sp.trait como entrada no argumento x
  da função e tree no argumento phylo. Atribui ao objeto phylosign
# depois sobrescreve phy.sign com phylosign.
if(model == "K.statistic") {
  phy.sign <- list(rep(NA, length(sp.trait)))
   for (j in 1:length(sp.trait))  {
    phylosign <- phylosignal(x = sp.trait[[j]], phy = tree, checkdata = T,
reps = 999)
      phy.sign [[j]] <- phylosign  }
}

# se o modelo solicitado for lambda, cria objeto continuous como uma lista
de NA do tamanho de sp.trait
# inicia ciclo FOR com contador k que lê de 1 até o tamanho da lista
sp.trait
# calcula sinal filogenético com a função fitContinuous, usando cada vetor
de sp.trait como entrada no argumento
  dat da função e tree no argumento phy. Atribui ao objeto fitcont
# sobrescreve continuous com os valores do sinal filogenético de cada
atributo calculado pela função
fitContinuous (primeiro objeto da lista que retorna), armazenados
```

```
temporariamente no objeto fitcont

if(model == "lambda") {
  continuous <- list(rep(NA, length(sp.trait)))
     for (k in 1:length(sp.trait)) {
      fitcont <- fitContinuous(dat = sp.trait[[k]], phy = tree, model =
"lambda")
        continuous[[k]] <- fitcont
}

# Simulando uma distribuição de valores nulos para o sinal filogenético #

null.cont <- list(rep(NA, length(sp.trait)))    # cria objeto null.cont como
lista de NA de tamanho de sp.trait
null.phylo <- list(null.cont)                   # atribui uma lista de
null.cont a null.phylo
null.distrib <- list(rep(NA, length(sp.trait))) # cria objeto null.distrib
como uma lista de NA do tamanho de
sp.trait


# inicia ciclo FOR com contador m que lê de 1 a 99 vezes (OBS: reduzi o
tamanho do cenário nulo para rodar
  mais rápido)
# inicia outro ciclo FOR com contador l que lê de 1 até o tamanho de
sp.trait
# embaralha os terminais da filogenia com a função tipShuffle, calcula o
sinal filogenético de cada atributo,
  atribui a null.cont e este a null.phylo;
# sobrescreve null.distrib com null.phylo. Esta será uma lista do tamanho de
sp.trait que contém 99 listas com
  valores nulos para cada atributo

for (m in 1:6) {
for (l in 1:length(sp.trait)) {
  null.phylo[[m]] <- null.cont [[l]] <- fitContinuous(dat = sp.trait[[l]],
phy = tipShuffle(tree), model =
"lambda")
  null.distrib [[l]] <- null.phylo } }


# pega em null.distrib o valor do sinal filogenético de cada atributo para
cada uma das vezes que fitContinuous
rodou (= 99 vezes)

lambda <- list(rep(NA, length(sp.trait)))  # cria lista de NA do tamanho de
X e atribui ao objeto lambda
l.sig <- as.vector(lambda)  # cada elemento de lambda será um vetor
numérico. Atribui ao objeto l.sig

# inicia ciclo FOR com contador g que lê de 1 até 99 vezes
```

```
# inicia ciclo FOR com contador h que lê de 1 até o tamanho de x
# para cada atributo pega todos os valores de lambda gerados e guarda como
vetor em l.sig
# sobrescreve lambda com l.sig
# lambda é uma lista que contém vetores dos valores nulos gerados para cada
atributo

for (g in 1:6) {
  for (h in 1:length(sp.trait)) {
    l.sig[g] <- null.distrib[[h]][[g]]$opt$lnL
    lambda [[h]] <- l.sig } }

# calcula a média dos lambdas obtidos na distribuição nula

null.lambda <- rep(NA, length(sp.trait))  # cria vetor de NA e atribui a
null.lambda
n.lambda <- as.vector(null.lambda)  # cria n.lambda como vetor de
null.lambda

# inicia ciclo FOR com contador f que lê de 1 até o tamanho de x
# sobrescreve n.lambda com os valores das médias de cada elemento de lambda
# sobrescreve null.lambda com n.lambda

for (f in 1:length(sp.trait)) {
  n.lambda[f] <- mean(lambda[[f]])
  null.lambda <- n.lambda }

# teste de verossimilhança para verificar se o sinal filogenético observado
(guardado em continuous)
# pode ser gerado ao acaso (valores nulos em null.lambda). Assume
distribuição qui-quadrado com grau de liberdade
= 1


test <- rep(NA,length(sp.trait))      # cria test e atribui NA
test.1 <- as.vector(test)             # cria test.1 como vetor de test
p.value <- rep(NA,length(sp.trait))   # cria p.value e atribui NA
p.value.1 <- as.vector(p.value)       # cria p.value.1 como vetor de p.value

# inicia ciclo FOR com contador e que lê de 1 até o tamanho de x
# sobrescreve test.1 com cálculo da taxa de verossimilhança para cada
atributo: diferença entre lambda observado
  e nulo, depois multiplica por 2
# sobrescreve test com test.1
# calcula distribuição qui-quadrado para cada atributo e atribui a p.value.1
# sobrescreve p.value com p.value.1


for (e in 1:length(sp.trait)) {
  test.1[e] <- 2*(continuous[[e]]$opt$lnL - null.lambda[e])
```

```
  test <- test.1
  p.value.1[e] <- pchisq(test[e],df = 1, lower.tail = F)
  p.value <- p.value.1 }

## Retorno de acordo ao modelo

if(model == "lambda") {
  return(list(continuous, test, p.value)) }

if(model == "k.statistic") {
  return(list(phy.sign)) }

} }

}
```

II.IV Documentação

```
~~Calcula sinal filogenético a partir de dados contínuos em dataframe ou
matriz~~

Description:

   ~~Calcula estatística K e/ou lambda para cada atributo, assim como o
valor de P, baseado em um cenário nulo
construído por aleatorização dos terminais da filogenia.~~
Usage:

   ~~phyloSignal.dataframe(x, tree, model = c("lambda", "K.statistic")~~

Arguments:

 ~~ tree = árvore filogenética da classe phylo
    x = dataframe ou matriz com nome dos táxons na primeira coluna. Estes
devem ser compatíveis com nomes em
        tree.
    model = modelo para cálculo do sinal filogenético. Calcula lambda e/ou
estatística K. ~~

Details:

   ~~Árvore filogenética deve conter nomes idênticos e mesmo número de
táxons do dataframe.
     Pacotes geiger e picante são requeridos.~~

Value:

   ~~Retorna lista contendo:

      continuous : Lista contendo lista de retorno de fitContinuous para
cada atributo de x. Valores de lambda
```

```
                    são estimados por log-verossimilhança pela função
fitContinuous e armazenados em 'opt$lnL' na
                    lista de retorno dessa função  (Veja detalhes em
fitContinuous).

      test: Vetor contendo os valores da taxa de verossimilhança de cada
atributo.
      p.value: Vetor contendo os valores de P considerando distribuição qui-
quadrado com 1 grau de liberdade.

      phy.sign: Lista contendo dataframe de retorno de phylosignal para cada
atributo (veja detalhes em
                phylosignal).~~


Author:

   ~~Gabriella Leal (gabriellaleal7@gmail.com)~~

References:

   ~Blomberg, S. P., Garland, T. & Ives, A. R. 2003. Testing for
phylogenetic signal in comparative data:
    Behavioral traits are more labile. Evolution 57:717-745.

    Pagel, M. 1999. Inferring the historical patterns of biological
evolution. Nature 401:877-884.~

See Also:

   ~~fitContinuous {geiger}
     phylosignal {picante} ~~~

Examples:

   ## Exemplo executável para o código da função: ----
   ## ==> library(geiger)
   ##       data(caudata)
   ##        t1 <- caudata$dat
   ##        t2 <- rnorm(1000, 4, 1.5)
   ##        t2 <- sample(t2, 197)
   ##        t3 <- rnorm(1000, 4, 0.3)
   ##        t3 <- sample(t3, 197)
   ##        t4 <- rnorm(1000, 5, 2)
   ##        t4 <- sample(t4, 197)
   ##        n.data <- data.frame(caudata$phy["tip.label"],t1, t2, t3, t4)
   ##        phyloSignal.dataframe(x = n.data, tree = caudata$phy, model =
"BM")
```