# Filipe Serrano

Estudante de Doutorado em Ecologia no IB-USP.

O meu projeto aborda biogeografia e macroecologia da maior família de serpentes neotropical e propõe elucidar como distribuição, riqueza e diversidade filogenética atual refletem processos históricos e ecológicos, gerando também informação para ações de conservação.

Orientado por Cristiano Nogueira

# Exercícios

Link para exercícios resolvidos: Exercícios

# Trabalho final

### Plano A: Modelo nulo realista de Diversidade Filogenética

A unidade da biodiversidade é geralmente o número de espécies num local, a Riqueza (SR). No entanto as espécies são todas iguais “mas algumas são mais iguais do que outras”, pois refletem processos evolutivos diferentes. A Diversidade Filogenética (PD)[1)] visa medir quanta história evolutiva está contida num só lugar, somando o comprimento do ramo entre cada espécie presente na filogenia.

PD é altamente correlacionada com SR e por isso, modelos nulos são necessários para entender se a PD de um local é superior ou inferior ao esperado com as espécies aí presentes. Até à data os modelos nulos disponíveis randomizam espécies da filogenia mantendo constante a SR do local. No entanto, estes modelos não representam um verdadeiro modelo nulo da PD do local pois realisticamente muitas espécies presentes na filogenia não ocorreriam no local por restrições abióticas (e.g. temperatura, humidade, etc), bióticas (tipo de presas) ou biogeográficas (barreiras ou contingência histórica). Desta forma é necessário especificar uma escala regional da qual randomizar as espécies potencialmente presentes à escala local.

A minha proposta é criar uma função que calcule um modelo nulo de PD de um local com composição e SR conhecidas e cuja escala regional possa ser determinada (pela distância ao local focal) para que apenas as espécies nela presente possam ser randomizadas.

#### Planejamento da função

#### Input:

1. Filogenia em formato .tre
2. Dataframe de cada local (1 local/linha) com colunas de:

- `local` = nome do local (class: string)
- x = longitude (classe: numeric com 2 casas decimais).
- y = latitude (classe: numeric om 2 casas decimais).
- `A:k` = presença/ausência de cada espécie A,B,k...(classe: binary)

**Pseudo-código:**

1. `require(caper)`, calcula PD de uma filogenia
2. Cria coluna SR no dataframe, somando todas as presenças de cada local
3. Cria coluna PD no dataframe, usando `caper`
4. Plota todos os locais num só mapa, com valores de SR como label
5. Calcula `distancia` euclideana entre os locais utilizando coordenadas x e y
6. Pergunta qual o local `focal` e que `raio` de distância o utilizador quer selecionar para escala regional e seleciona todos `locais.reg` cuja `distancia` ao local `focal` seja igual ou menor que o `raio`
7. Lista todas as espécies `species.reg` presentes em `locais.reg` e calcula `SR.reg`, a soma de todas as espécies únicas
8. Cria objeto `simulacao` com N NAs.
9. Atribui ao primeiro valor de simulacao o PD de `focal`
10. Entra em um ciclo `for` com contador `i` de 2 até N.
    1. Amostra número de espécies igual aSRdas `SR.reg` sem reposição e calcula PD para esse subset.
    2. Repete N-1 vezes e coloca valores em `simulacao`.
11. Cria histograma de `simulacao` com linha vertical para PD observado em `focal`
12. Calcula o p-valor para verificar se PD observado é diferente do esperado pelo modelo nulo realista

**Verificando os parâmetros:**

- N é um número inteiro e maior que 0? Se não, retorna: “N precisa ser um numero inteiro e > 0.”
- `focal` é um string existent na coluna `local`? Se não, retorna: “focal precisa ser um local.”
- Informação nas colunas `A:k` é binária (1 ou 0)? Se não, retorna: “Para cada espécie A:k, colocar 1(presença) ou 0(ausência).”

**Output:**

- Histograma com a distribuição dos valores de PD do modelo nulo.
- Valor de PD observado plotado no histograma.
- Significância (p-value): probabilidade de valor de PD observado ser diferente ou superior/inferior (dependendo se teste bilateral ou unilateral for selecionado, respetivamente).

Comentários Felipe Ernesto

Fala Filipe!

Achei a proposta bem interessante. Me parece viável, mas sem ser simples demais, e imagino que

poderia ser do interesse de outras pessoas. Vou fazer alguns comentários que são mais dicas do que problemas.

Vamos lá.

1. Sua função usa um pacote do R, o caper. Eu dei uma olhada no manual dele e parece ser simples de usar e me parece tb que ele disponibiliza tudo que vc precisa. Só queria comentar que pacotes com documentações inintendíveis e que não fazem tudo o que gostaríamos não são raros. Por isso é bom sempre se certificar que vc vai conseguir usar o pacote sem maiores dificuldades, senão vc pode ter problemas. Não sei se vc já conhecia ele, mas tb não estou muito preocupado com isso pq me parece que o pacote é tranquilo de usar.
2. Na linha 5 do pseudo-código, vc não especificou ao certo como vc vai armazenar as distâncias dos locais. Não sei ao certo se vc só não detalhou isso ou se vc de fato não pensou no assunto. De toda forma, existe uma forma bem fácil de armazenar essas distâncias. Fiz esse comentário tb pra falar que se vc mudar a ordem da linha 5 com a linha 6 vc pode reduzir o número de cálculos feitos e mudar a forma como serão armazenadas as distâncias. Aí é contigo, faça como preferir.
3. Talvez esse seja o comentário mais importante. Sobre o passo 6 do pseudo-código, pelo que eu entendi vc vai pedir `focal` e `raio` de forma interativa, certo? Você chegou a pensar em colocar essas variáveis como argumentos da função? Acho que isso facilitaria a automatização dela. Por exemplo, se alguém quisesse colocá-la dentro de um loop pra rodar com vários parâmetros diferentes. (Pensei numa coisa que pode tornar o código complicado demais, mas caso vc termine tudo rápido e queira adicionar mais coisas, as variáveis `focal` e `raio` poderiam ser vetores e vc mesmo faz essa automatização)
4. Não sei se entendi direito o que é `SR.reg`. Se for só a riqueza regional, não entendi pra quê fazer esse cálculo, não entendi onde isso seria usado. Tb pq na linha 10a acho que vc quis dizer `especies.reg` em vez de `SR.reg`.
5. Imagino que o `N` seja um input da função, porém ele não foi detalhado no começo. Acho legal o `N` ter um valor default que gere p-valores confiáveis, pra o usuário não ter que precisar se preocupar com esse detalhe. Porém, se ele quiser mais o menos é bom ele ter a liberdade de mudar.
6. Sobre o input no formato .tre. Pelo que entendi vc vai usar o pacote caper pra ler esse formato pra vc e te devolver um objeto. Nesse caso estava pensando que talvez o ideial fosse exigir o usuário carregar o pacote e ler o .tre fora da função, aí chamar a função, que agora recebe o objeto e não mais o arquivo .tre. Aí vc poderia verificar se o input é o objeto mesmo e tb soltar um warning caso o pacote não esteja carregado. Mas isso já o detalhe do detalhe, fica a seu critério. Inclusive pq não sei se o Alê vai achar tudo bem sua função depender de um pacote.
7. Sobre o input `x` e `y`, é meio esquisto definir os locais só com a latitude e longitude. Isso porque essas cordenadas te dão apenas um ponto e não dá pra as espécies viverem umas em cima das outras rs. Porém imagino que qualquer tentativa de melhorar isso (definindo os locais como regiões 2D) só vai te atrapalhar, pq tudo pode ficar muito complexo e vai desviar o foco do trablaho. Comentei isso mais pra vc dar uma justificada nesse aspecto e falar que o ponto de entrada é o centro de massa ou qualquer coisa do tipo.
8. Por fim, veja direitinho o que sua função vai retornar. O histograma é uma imagem e não pode ser retornada junto com o objeto. Sua função pode desenhar ele e depois retornar o resto.

É isso... Ufa! haha

Qualquer coisa me procura ou escreve algo aqui mesmo que eu olho.

Abs

Hey Felipe. Muito obrigado pelos comentários! Então, por ordem:

1. É uma função simples, usaria só para carregar a filogenia e carregar PD.
2. Pensei em armazenar as distâncias como uma matriz, seria isso? Quanto à troca de passos: então primeiro dar distância entre locais, aí já escolher `focal` e `raio`, o que devolveria `locais.reg`. Era isso? Não sei se entendi direito.
3. Claro, armazenar como argumento tornaria tudo mais fácil, legal! A parte dos vetores é que não entendi tão bem.
4. `SR.reg` seria a riqueza regional, sim. Seria só outro parâmetro mas que na verdade não é necessário para o cálculo. Seria sim mais legal sair no summary final.
5. Então já deixar `N` como argumento mas deixar um DEFAULT=5000, por exemplo?
6. Legal, o warning! Fazer o `require` vem antes da função em si. Inicialmente ia fazer a fórmula de cálculo de PD mas fica bastante complexo com mais do que 2 espécies. Assim decidi fazer a versão mais direta e mais realista, visto que os usuários vão mais frequentemente importar filogenias do que matrizes com distâncias de PD.
7. Na verdade até dá, vai que os bichos/plantas querem mesmo ahah. Mas na verdade a ideia original veio de um ponto centróide de uma quadrícula. Mas claro, conseguir distâncias 2D seria uma avanço muito bacano!
8. Concordo totalmente, acho que sair um summary no final seria muito valioso, tendo até o histograma a acompanhar.

Valeu, Felipe, aguardo os teus comentários e já vou começando! Abraço

Comentários Felipe Ernesto

Fala Filipe!

Vou responder só os itens necessários:

2. Do jeito que está acho que uma matriz é o melhor mesmo. Note que ela vai ficar simétrica. Sobre a troca de linhas, a ideia é que se vc receber `focal` e `raio` primeiro, vc só precisa calcular a distância dos outros locais até `focal`, não precisando calcular a distância entre os outros locais. Acho que o jeito da matriz é um pouco mais geral, permitindo que vc não mude muito o código caso encontre algum uso pra essas distâncias no futuro, mas do jeito que está agora vc não precisa delas.

3. Deixa pra lá o lance dos vetores haha

5. Isso aí!

É isso,

Abs

## Plano B: Cálculo de dominância de habitat

**Input:**

1. Dataframe de coordenadas de ocorrências de espécies provindas de grid

- `especie` = nome do especie (class: string)
- x = longitude (classe: numeric com 2 casas decimais).
- y = latitude (classe: numeric om 2 casas decimais).

2. Dataframe de coordenadas de habitat provindas de grid

- `habitat` = nome de classe geográfica (e.g. ecorregião, bioma, fitofisionomia) (class: string)
- x = longitude (classe: numeric com 2 casas decimais).
- y = latitude (classe: numeric om 2 casas decimais).

**Pseudo-código:**

1. Fazer merge dos dois dataframes num só
2. Cria matrix de contagem para número de pontos de cada `espécie` "única" em cada `habitat`
3. Calcula percentagem de ocorrência de cada `espécie` em cada `habitat`
4. Calcula o `habitat` com maior média, `dom.habitat`
5. Pergunta se utilizador quer fazer merge de um ou mais `habitat`, para coalescer categorias
6. Se sim, soma colunas de `habitat` indicadas
7. Calcula novamente o `habitat` com maior média, `dom.habitat`
8. Cria objeto `simulacao` com N+r NAs, com N igual ao número de `espécie` e `r` igual ao número de simulações
9. Aloca primeiros N valores em `simulacao` para `dom.habitat` de cada `especie`
10. Entra num ciclo `for` com contador `i` de N+1 até `r`+N.
    1. Simula, com reposição, cada possível par de coordenadas x e y encontrado segundo uma distribuição uniforme
    2. Calcula a percentagem de ocorrência dos pontos simulados para `dom.habitat`
    3. Guarda valores simulados em `simulacao`, começando em N+1
11. Cria histograma de `simulacao` com linha vertical para cada `especie`, identificada com label
12. Calcula o p-valor

> Estou com alguma dificuldade em decidir entre fazer o "habitat" dominante (usando a média) entre vários possíveis habitats ou em só permitir classes complementares de habitat (e.g. área aberta vs. floresta, planaltos vs. áreas baixas). Caso fizesse o último, poderia simular mais facilmente as percentagens e seria mais intuitivo determinar se uma espécie é ou não especialista.

Comentários Felipe Ernesto

Fala Filipe!

Então, dessa vez ficou complicado de entender pq não teve contextualização. Tentei pegar a ideia pelo pseudo-código, mas não sei se entendi direito. De toda forma, computacionalmente acho que atende aos critérios exigidos, mas acho que a falta de contextualização poderia afetar a nota. Vou fazer alguns comentários, mas não sei o quanto eles vão fazer sentido:

1. O que exatamente vc quis dizer na linha 3 do pseudo-código? É o número de vezes em que a espécie S ocorre no habitat X dividido pelo número de ocorrências da espécie S em todos os habitats?
2. Na linha 4, a média seria pra um dado habitat vc pega a proporção de ocorrência de cada espécie naquele habitat e tira a média. Tentei pensar aqui, mas pra mim não foi trivial o que essa média significa. Uma espécie que ocorra bastante em um dado habitat pode puxar a média bem pra cima, mesmo se as outras espécies ocorrem pouco ou nada.
3. Na linha 5, esse merge me parece que teria que ser feito antes, pq vc basicamente vai ter que refazer tudo.
4. Ok, na linha 9 parece ficar mais claro o que é o `dom.habitat`, é o habitat preferido de cada espécie. Mais aí vc não precisa tirar nenhuma média, é só comparar a proporção de ocorrência da espécie em todos os habitats.
5. Não entendi o que rola dentro do `for`. Os x e y são do habitat ou da espécie?

Enfim, parece que vc tem alguma ideia por trás e imagino que seja razoável, mas não consegui pegar direito. Acho que vc pode ficar com o plano A, mas se por acaso vc acabar precisando do B vc vai ter que explicar melhor pra eu conseguir te ajudar.

Qualquer coisa posta aqui ou me procura!

Abs

# Função final

```
regionull.PD=function(tre=NULL,com=NULL,focal=NULL,radius=NULL,nsim=2000,tes
t=NULL,hist=FALSE,col="darkgreen"){
  com=as.data.frame(com)#força o objeto de composição de comunidades em
dataframe
  com[4:length(com)]=(com[4:length(com)]>0)+0 #converte todas as entradas
com números >1 de cada espécie para 1, de forma a que até um objeto com
abundâncias seja utilizável
  com$SR=apply(com[4:length(com)],1,sum) #calcula species richness (SR),
somando todas as presenças em cada local (desde a 4ª coluna até à penúltima)
  xy=data.frame(com[2:3]) #isola coordenadas noutro dataframe
  rownames(xy)=com$local #nomeia cada par de coordenadas com o local
correspondente
  dist.com=as.matrix(dist(xy),upper=F) #calcula distância euclideana entre
locais e guarda objeto numa matriz simétrica de distâncias
  SR.focal=com[com$local==focal,"SR"] #calcula SR de local focal
  if(SR.focal<2)  #se a riqueza do local focal for inferior a 2
    {stop ("local focal deve ter mais do que uma espécie presente!") #pára a
função, avisando o utilizador que é impossível calcular o PD nulo da região
se o local focal não tiver mais do que uma espécie
  }
  else { #se a riqueza do local for superior a 1
  dist.focal=dist.com[focal,] #retorna apenas distâncias entre focal e
outros locais
  locais.reg=names(which(dist.focal<radius)) #retorna quais locais estão a
```

```
menos de "radius" de distância
  if (length(locais.reg)<2) { #se o número de locais regionais for inferior
a 2 (não abrangendo nenhum outro local além do focal)
    stop(paste("Nenhum local a menos de", radius, "de distância")) #pára a
função e retorna a mensagem
  }
  com.reg=com[com$local %in% locais.reg, ]#retorna apenas informação dos
locais a menos de "radius" de distância
  focal.range=4:(length(com.reg)-1) #seleciona só desde a 4ª coluna até à
penúltima, onde estão as presenças/ausências
  species.focal=names(which(apply(com.reg[com.reg$local ==
focal,focal.range], 2, sum)>=1)) #retorna quais espécies existem no local
focal
  species.reg=names(which(apply(com.reg[focal.range], 2, sum)>=1)) #retorna
quais espécies únicas existem a "radius" de distância. Selecionam-se apenas
espécies com 1(presença) em pelo menos um local
  if(!require(caper)){install.package("caper");library(caper)}#caso pacote
"caper" não esteja carregado, instala e carrega pacote "caper".
  clmat=clade.matrix(tre) #próprio do pacote "caper", manipula a árvore
filogenética para análise
  pd.focal=pd.calc(clmat,tip.subset=species.focal) #calcula a distância
filogenética (PD) do local focal, o nosso valor observado
  result=rep(NA,nsim) #cria objeto para armazenar nsim simulações
  result[1]=pd.focal #guarda o valor observado na primeira posição
    for(i in 2:nsim) {#loop de nsim vezes e armazena a partir da 2ª posição
    pd.reg=pd.calc(clmat,tip.subset=sample(species.reg,SR.focal)) #cálculo
de PD com espécies aleatoriamente escolhidas das presentes na região (a
menos de radius de distância) mas respeitando SR do local focal
    result[i]=pd.reg #armazena no objeto result
    res=mean(result) - result #calcula resíduos de PD
    res.maior=(sum(res > res[1])/nsim) #calcula probabilidade do resultado
ser maior do que o esperado
    res.menor=(sum(res < res[1])/nsim) #calcula probabilidade do resultado
ser menor do que o esperado
    res.dif=sum((abs(res) > abs(res[1])))/nsim #calcula probabilidade do
resultado ser diferente do esperado
    }
  if(hist){ #se hist=TRUE
    hist(res,main="Calculando Diversidade filogenética nula
realista",xlab="Resíduos de Diversidade Filogenética",ylab="Frequência",xlim
= c(min(res),max(res))) #cria histograma de valores obtidos/esperados
    abline(v = res[1], col=col, lwd=2)#Assinala valor observado no
histograma
  }
  if(test=="higher") #se for selecionado o parâmetro "higher" no argumento
test
    {
    if (res.maior>=0.05){ #se p-value for igual ou maior que 0.05, não
significante
      res.maior.i="Local não apresenta divergência filogenética" #cria
objeto com interpretação de não aceitar H0
```

```
    }
    else{ #se p-value for menor que 0.05, significante
      res.maior.i="Local apresenta divergência filogenética" #cria objeto
com interpretação de não aceitar H0
    }
    list.maior=list("p-value" = res.maior, "PD do local" = pd.focal[1],
"interpretação" = res.maior.i) #cria uma lista com os objetos que o
utilizador receberá
    return(list.maior)} #devolve lista
  if(test=="lower") #se for selecionado o parâmetro "lower" no argumento
test
  {
    if (res.menor>=0.05){ #se p-value for igual ou maior que 0.05, não
significante
      res.menor.i="Local não apresenta convergência filogenética" #cria
objeto com interpretação de aceitar H0
    }
    else{ #se p-value for menor que 0.05, significante
      res.menor.i="Local apresenta convergência filogenética" #cria objeto
com interpretação de não aceitar H0
    }
    list.menor=list("p-value" = res.menor, "PD do local" = pd.focal[1],
"interpretação" = res.menor.i) #cria uma lista com os objetos que o
utilizador receberá
    return(list.menor)} #devolve lista
  if(test=="different") #se for selecionado o parâmetro "different" no
argumento test
  {
    if (res.dif>=0.05){ #se p-value for igual ou maior que 0.05, não
significante
      res.dif.i="Local não apresenta padrão filogenético" #cria objeto com
interpretação de aceitar H0
    }
    else{ #se p-value for menor que 0.05, significante
      res.dif.i="Local apresenta padrão filogenético" #cria objeto com
interpretação de não aceitar H0
    }
    list.dif=list("p-value" = res.dif, "PD do local" =
pd.focal[1],"interpretação" = res.dif.i) #cria uma lista com os objetos que
o utilizador receberá
    return(list.dif)} #devolve lista
    }
}
```

Ficheiro script da função:regionull.pd.r

## 2. Help da função

```
regionull.PD             package:unknown                R Documentation

~~ Modelo nulo realista de Diversidade Filogenética ~~

Description:

     ~~ Esta função permite testar se a Diversidade Filogenética (PD, Faith
1992) de um local é maior ou menor do que a esperada ao acaso, usando um
modelo nulo geograficamente realista para randomizar a presença no local de
outras espécies presentes a certa distância.~

Usage:

     ~~ regionull.PD=function(tre,com,focal,radius,nsim,test,hist,col)~~

Arguments:

 tre
    árvore filogenética a ser escolhida, no formato .tre

 com
    dataframe de locais, suas coordenadas e presença/ausência de espécies

 focal
    comunidade local focal

radius
    distância ao local focal

 nsim
    número de simulações, default=2000

 test
    "different", "lower" ou "higher" conforme padrão,convergência ou
divergência filogenética for esperada, respetivamente, default="different"
 hist
    TRUE gera histograma, default=FALSE

 col
    cor de valor de PD assinalada no histograma, default="darkgreen"


Details:

     ~ A ordem das colunas do dataframe 'com' deverá ser: local, longitude,
latitude, seguido de todas as espécies com presença/ausência~

Value:

     ~ A função devolve:
    - o p-value do teste escolhido ("different", "lower" ou "higher" ),
```

```
    - PD do local 'focal',
    - a possível interpretação do resultado e
    - histograma dos valores de PD simulados para o local, com o PD
observado assinalado com a cor definida pelo argumento 'col'


Warning:

    ~ A função retorna erro se:
    - Nome das espécies no dataframe 'com' não corresponde às espécies na
'tre' carregada
    - Número de espécies do 'focal' é menor do que 2 (não é possível
calcular distância)
    - Número de locais a menos de 'radius' é inferior a 2 (só é selecionado
o 'focal' e portanto não é possível simular PD com espécies que ocorram
noutros locais)

Note:

     ~~Necessário o package "caper"~~
    ~Função também aceita matrizes com valores de abundância, mas converte
em presença/ausência~
    ~Decidi usar este pacote para ler a árvore filogenética ao invés de
calcular PD por uma matriz de distâncias pois o utilizador comum geralmente
usa uma árvore já completa pois a matriz de distâncias é um jeito menos
prático de visualizar relações filogenéticas~



Author:

     Filipe Serrano, IB-USP filipe.serrano@usp.br
    http://cnbiogeo.wixsite.com/cristiano-nogueira/people


References:

     ~Faith, D. P. (1992). Conservation evaluation and phylogenetic
diversity. Biological conservation, 61(1), 1-10. ~

See Also:

     ~~ Funções: pd.calc(), dist() ~~~

Examples:

     ## Exemplo simples
    tre=(rtree(n=20)) #cria árvore filogenética randómica, com 20 tips
("espécies)
    tips.tre=tre$tip.label #cria objeto com nome das espécies
```

```
    com=data.frame(matrix(ncol=3+length(tips.tre),nrow=5)) #cria data.frame
vazio

    colnames(com)=c("local","x","y",tips.tre) #atribui nomes às colunas do
data.frame
    local=c("jaboticabal","matao","bebedouro","dobrada","ribeirao") # cria
objeto com nomes dos locais
    com$local=local #atribui locais à coluna 'local' do data.frame

    x=runif(5,min=0, max=40) #simula valores de longitude de 0 a 40
    y=runif(5,min=0, max=40) #simula valores de latitude de 0 a 40
    com$x=x #atribui valores de longitude à coluna 'x' de com
    com$y=y #atribui valores de latitude à coluna 'x' de com

    for (i in 4:length(com)) #simula presença ou ausência para as espécies e
armazena-as nas colunas de espécie do data.frame
        {
        com[i]=rbinom(n=5,size=1,prob=0.5)
        }
     ## Correndo a função regionull.PD para verificar convergência
filogenética de Jaboticabal, simulando 1500 vezes o valor de PD ao incluir
espécies presentes a menos de 80 de distância
    ##
regionull.PD(tre,com,focal="jaboticabal",radius=80,test="lower",hist=T,nsim=
1500,col="darkblue") # executa a função



    ## Exemplo mais complexo, com lagartos Anolis
    tre2=read.tree("Anolis.tre") #ficheiro deve estar na pasta do working
directory
    tips.tre2=tre2[4]$tip.label
    df=data.frame(matrix(ncol=3+length(tips.tre2),nrow=5))
    colnames(df)=c("local","x","y",tips.tre2)
    local=c("jaboticabal","matao","bebedouro","dobrada","ribeirao")
    df$local=local
    x=runif(5,min=0, max=40)
    y=runif(5,min=0, max=40)
    df$x=x
    df$y=y

    for (i in 4:lenght(com))
        {
        df[i]=rbinom(n=5,size=1,prob=0.5)
        }

## Correndo a função regionull.PD para verificar divergência filogenética de
Ribeirao simulando 2500 vezes o valor de PD ao incluir espécies presentes a
menos de 45 de distância
    ##
regionull.PD(tre=tre2,com=df,focal="ribeirao",radius=45,test="higher",hist=T
,nsim=2500,col="firebrick") # executa a função
```

Example A - random tree Example B - Anolis Impossível carregar ficheiro .tre, fazer botão direito e “salvar como” anolis.tre Ficheiro propriedade de Latin American Macroevolution Workshop, co-taught by Ricardo Betancur, Luke Harmon, & Liam Revell, Página do Workshop

1)

Faith, D. P. (1992). Conservation evaluation and phylogenetic diversity. Biological conservation, 61(1), 1-10.