Thais Sasso Lopes

Mestranda em Ecologia pelo Labvert - IB- USP, sob a orientação do Prof. Dr. Márcio Martins. Trabalharei com detecção de anfíbios em riachos da Mata Atlântica por meio de DNA ambiental.

Currículo Lattes

Exercícios

Linque para a página com os meus exercícios resolvidos exec

Trabalho Final

Plano A: Gráfico exploratório pós-teste Kruskal-Wallis: comparações múltiplas

Contextualização:

O teste de Kruskal-Wallis (KW) é um teste não-paramétrico, análogo aos testes de “soma-de-postos” para comparações de médias de populações de *n* amostras independentes (Verzani, 2005). Esse teste nos informa se há uma diferença entre as medidas de três ou mais tratamentos de interesse, embora não reporte especificamente quais tratamentos diferem uns dos outros. Para tanto, após o teste de KW pode ser feito uma comparação múltipla *post hoc*, que apontará quais tratamentos são diferentes e quais não o são, fornecendo os *p*-valores para cada comparação par a par.

Ao realizar comparações múltiplas, é comum a necessidade de expressar graficamente a semelhança/diferença entre as medidas obtidas em cada tratamento. Não há, entretanto, uma função em R que processe os resultados de tal teste e os represente em um gráfico. Proponho desenvolver uma função que realize um teste de Kruskal-Wallis seguido de uma comparação múltipla, e em seguida produza um gráfico que demonstre as relações de semelhança/diferença entre os diversos tratamento. algo que poderia ajudar o pesquisador a entender melhor seus dados e publicá-los.

A função em si:

Com a intenção de visualizar essa comparação múltipla, a minha função se encarregaria de expressar graficamente o resultado das comparações par a par indicando quais grupos são diferentes ou não:

Os passos da função serão:

1. Realizar o teste de KW com dados na forma de vetores, dataframe ou lista;

2. Verificar o formato de entrada dos dados. Caso os dados sejam fornecidos em forma de vetores, a

função deverá organizá-los em forma de lista antes de realizar o teste KW fazendo a devida identificação de qual valor corresponde a qual tratamento;

3. Há ao menos três funções de pacotes diferentes que realizam estes testes e que poderiam estar na minha função: kruskal.test(), kruskal() e kruskalmc, que retornam o *p*-valor do teste KW ou das comparações múltiplas;

4. Plotar os dados de cada tratamento em um mesmo gráfico, que pode ser semelhante a um boxplot. Os tratamentos poderão ser dispostos no gráfico em ordem crescentede médias caso o pesquisador assim o queira (algo também controlado por um argumento da função);

5. Identificar os tratamentos em que não houve diferença significativa com letras (a, b, c, etc). ou seja, tratamentos sob a mesma letra informarão ao pesquisador que se tratam de tratamentos em que não houve diferença significativa entre eles;

6. A função pode ainda imprimir no terminal, além dos p-valores dos testes, uma lista informando os postos dos dados e o ranque médio de cada tratamento (valores que nem sempre são informados quando se realiza o teste de KW no R) usados no cálculo do teste de KW.;

7. Passos extras...Embelezamento do gráfico ao gosto do pesquisador: argumentos para alterar alguns aspectos do gráfico.

Verzani, John. 2005. Using R for introductory statistics. Chapman & Hall. p391.

Plano B: Acrescentando diversas informações a uma filogenia

Contextualização:

Essa ideia veio da ressuscitação dos meus dados de IC, no qual tive que plotar dados de dieta, tamanho corpóreo e tamanho de ninhada de serpentes na filogenia do grupo que estava trabalhando. Para tanto, tive que fazer uma média ou proporção desses dados e plotar apenas um valor para cada espécie para fazer uma reconstrução do carater ancestral. Pensei então que poderia mostrar esses dados completos, junto à filogenia, sem perder tanta informação.

A função em si:

A função plotará uma filogenia e ao lado dela o pesquisador poderá acrescentar alguma informação de interesse, como dieta, composto químico produzido, tipo de asa etc. Além disso o pesquisador poderá selecionar um grupo de interesse na filogenia e nomeá-lo..

Os passos da função serão:

1. Ler e imprimir uma filogenia disponibilizada pelo usuário, tanto a partir de um arquivo de texto quanto em formatação parentética;

2. Ler e organizar os dados do usuário. Será preciso fornecer um data.frame com as informações para

cada espécie a serem plotadas na filogenia;

3. Se as informações forem qualitativas, um quadro de n cores será criado, sendo que cada cor corresponderá a uma das categorias do objeto. Cada espécie terá n quadrados, e somente aqueles que corresponderem à informação fornecida será preenchido com a cor equivalente à informação;

Por exemplo, as minhas informações são sobre presas presentes na dieta de um grupo de serpentes, sendo as categorias da dieta: "anfíbios"(cor azul), "aves" (cor amarela), "pequenos mamíferos"(cor verde) e "lesmas"(cor vermelha). Uma das espécies na minha filogenia come apenas lesmas, então ela só terá o quadrado vermelho pintado. Já uma outra espécie come pequenos mamíferos, sapos e aves, ela terá então 3 quadrados pintados (verde, azul e amarelo). Isso se repetirá ao longo de toda a filogenia. Quando finalizado, poderá ser visto um padrão em alguns grupos das serpentes, com algumas que aparentemente só se alimentam de lesmas, outros apenas de anfíbios, etc. E nesse momento eu posso querer selecionar n espécies que se alimentam de sapos e que correspondem à tribo Xenodontini e dar um destaque a elas fazendo uma chave envolvendo todas as espécies e colocando o nome da tribo ao lado.

Para a proposta B acho que essas filogenias de alguns trabalhos já publicados podem ilustrar:

Filogenia de microorganismos com informação do local do corpo onde cada espécie ocorre indicado por uma cor diferente. Perceba que cada espécie na filogenia possui a informação de localidade, o que minha função faria seria receber todas essas informações e plotar na filogenia para visualizar padrões ao longo da filogenia. exemplo_1

Filogenia retirada do trabalho Conserved class of queen pheromones stops social insect workers from reproducing, Oystaeyen et al, 2014, Science....mostrando qual composto químico foi encontrado em cada espécie representando cada um por uma cor ao lado da filogenia: exemplo_3

Dica geral: sempre que sua proposta envolver uma saída gráfica, faça um esquema da proposta, nem que seja um esboço no Paint, pra gente poder entender melhor o que vc está propondo. A descrição verbal do gráfico às vezes confunde mais do que ajuda. A proposta A me parece um pouco confusa, pois a maioria dos testes post-hoc já informa entre quais grupos houve diferença significativa. Sua função, pelo que entendi é um "wrapper" para funções já existentes de testes KW, e o que ela faria é apenas ordenar e plotar os tratamentos (boxplot?) por valor da média, indicando quais grupos são diferentes. Além disso, vc não dá (ou não diz se dá) a opção de escolher qual o teste post-hoc a ser usado Se você for implementar as comparações múltiplas, pode ser válido, mas só para escolher qual função aplicar, não me parece uma boa ideia. Na função B eu não sei se consegui visualizar direito o que vc quer fazer. Me parece que vc quer plotar ao lado da filogenia as características das espécies, ou vc quer plotar tipo uma tabela espécie x característica, preenchendo as características com uma cor de acordo com o valor delas? uma visualização gráfica aqui ajudaria. tente dar uma clareada nas suas propostas pra gente poder te

ajudar melhor —- *Vitor Rios*

Plano final:

Plano B com as seguintes modificações: Ao lado da filogenia será plotado colunas das variáveis presentes no data.frame representadas por um gradiente de cor, de acordo com a proporção de cada variável em relação ao conjunto de dados de cada espécie da filogenia. Variáveis com proporções maiores serão representados por quadrantes de cores mais escuras, já aquelas com proporções menores por quadrantes de cores mais claras, variando então do branco ao tom escuro (degradê).

Além disso, quando alguma das espécies possuir valor alto par uma das variáveis, e esta possuir proporção maior do que um indíce (escolhido pelo usuário), uma classificação será plotada para esta espécie (de acordo com a escolha do tema ecológico pelo usuário). Sendo assim, a função não somente plotará as variáveis presentes para cada espécie, a proporção de cada uma, como também ressaltará as espécies em que o usuário possa ter interesse.

Página de ajuda (Help):

```
ecophy              package: unknown             R Documentation


Filogenia com informações ecológicas

Description

A função plota informações ecológicas em um gradiente de cores junto a uma
filogenia fornecida. Os dados ecológicos devem ser categóricos e refletirem
abundâncias (ex. Tema ecológico: Dieta, Categorias: Alimento A, Alimento B,
etc, Abundâncias: espécie 1 consumiu três, cinco, etc de cada alimento,
etc.)


Use

ecophy (phy, data, eco="dieta", index=0.7, type="phylogram", n=10, color=
"grape", font=1, place.legend="topleft", cex=3, pch=22,  edge.width=1,
edge.lty=1, edge.color="black", show.tip.label=TRUE, contorno="black",
col.text="grey30")


Arguments

phy     filogenia. Objeto da classe “phylo”.

data        data.frame com informações de abundâncias (class “numeric” ou
“integer”). Cada coluna do data.frame representa um categoria do tema
```

```
ecológico e cada linha (rowname) representa uma espécie.  A ordem das
espécies no data.frame deve ser a mesma ordem destas na filogenia.

type        especifica o formato em que a filogenia será desenhada. Opções:
"phylogram" (default) ou "cladogram".

eco     especifica o tema ecológico que será analisado e usado para
classificar as espécies. Opções: “dieta”, “atividade”, “hábito” ou “outro”.
Cada opção possuirá diferentes classificações das espécies: dieta
(especialista ou generalista), atividade (diurna, crepuscular ou noturna),
hábito (terrestre, aquático, fossorial ou arborícola), outro (qualquer outro
tema, nesse caso as espécies que tiverem valor acima do index escolhido será
indicada por um “*”).

index       “numeric” com valor entre 0 e 1 (i.e. index=0 significa 0%,
index=1 significa 100%), especifica a porcentagem mínima qua a espécie tem
que ter de uma das variáveis para ser classificada em alguma das categorias
do tema.

n       “numeric”, especifica o número amostral mínimo de dados que cada
espécie deve ter para esta ser incluida nas análises e classificações. Evita
que análises e classificações sejam feitas para as espécies com poucos
dados, ou seja, dados não representativos.

color       “character”, especifica o gradiente de cor desejada na plotagem
dos dados ao lado da filogenia. Opções:"cinza”,"coral","grape" (Default),
"grass" ou "sky".

font        “numeric”, especifica o tipo de fonte usada nos terminais.
Opções: 1 (texto simples, Default), 2 (negrito), 3 (itálico) ou 4 (negrito
itálico).

place.legend    “character”, especifica a posição da legenda no dispositivo
gráfico. Opções: “topleft” (Default), “bottomleft”.

cex     “numeric”, especifica o tamanho do símbolo plotado ao lado da
filogenia. Default=3

pch     “numeric”, especifica o tipo de símbolo plotado ao lado da
filogenia. Opções: 21, 22 (Default), 23, 24, 25.

edge.width  “numeric”, especifica a espessura dos ramos na filogenia.

edge.lty    “numeric”, especifica o tipo de desenho da linha. Opções: 1
(plain, Default), 2 (dashed), 3(dotted), 4(dotdash), 5(longdash),
6(twodash).

edge.color  especifica a cor dos ramos na filogenia.

show.tip.label  argumento lógico, especifica se os nomes nas extremidades da
filogenia devem ser mostrados ou não.
```

contorno especifica a cor do contorno do símbolo plotado ao lado da filogenia. Opções: veja as funções palette() ou colors(); NULL fará com que o símbolo não tenha linha de contorno. Default: “black”.

col.text especifica a cor do texto da classificação que aparecerá ao lado da filogenia

Details

Data deve possuir apenas valores do modo numeric. Rownames em data devem ser iguais e na mesma ordem que os nomes das extremidades de phy.

Value

ecophy retorna ao usuário uma figura no dispositivo gráfico, contendo a filogenia ao lado uma tabela de dados ecológicos. Os dados variam em um gradiente de cor de acordo com a sua proporção no conjunto de dados de cada espécie.

Cada coluna na tabela será nomeada conforme as colunas do data.frame.

A espécie que possuir um dos dados com proporção acima do valor determinado em index será classificada em uma das categorias do tema ecológico.

A espécie que possuir conjunto de dados inferior ao determinado em “n” não não possuirá dados plotados na tabela ao lado do seu nome na filogenia.

Nos temas eco==“atividade” ou “habito”, os nomes das classificações que aparecerão ao lado da filogenia corresponderão aos nomes das colunas em data.

Warning

Esta função necessita dos pacotes “plotrix” e “grDevices” instalados na área de trabalho.
Apenas os pch de números 22 a 25 podem ser escolhidos nesta função pois somente estes podem ser preenchidos com cores diferentes, caso contrário a função não funcionará como desejado.

Author(s)

Thais S. Lopes (thais.lopes@usp.br)

See Also

```
Função read.tree do pacote “ape” para criação de objetos da classe “phylo”.


Example

require(ape)
phy.teste<- read.tree(text = "((Puma_concolor,
Felis_silvestris),((((Lachesis_muta,Boa_constrictor),Vipera_berus),Naja_naja
),(((Canis_lupus,Canis_latrans),Canis_adustus),
((Camelus_bactrianus,Camelus_dromedarius),
(Cebus_olivaceus,Saimiri_ustus)))));")

phy.names<-phy.teste$tip.label

#Exemplo_1_Tema:Dieta

anfibio<-c(0, 0, 0, 0, 0, 0, 10, 9,89,100, 1, 2, 10)
serpente<-c(0,0,16,26,98,21, rep(0,7))
lagarto<-c(4,5,32,123,67,43,12,10,8,rep(0,4))
ave<-c( 49,123, rep(0,11))
dieta<-data.frame(phy.names, anfibio, serpente, lagarto, ave, row.names=1)
dieta[dieta==0]<-NA
ecophy(phy.teste, dieta)
ecophy(phy.teste, dieta, n=20, index=0.9) #determinando que apenas as
espécies com n amostral maior que 20 sejam consideradas na tabela, e que
para ser especialista elas devem possuir algum dos dados com proporção maior
ou igual a 90%.


#Exemplo_2_Tema:Atividade

noturno<-sample(30, 13, replace=TRUE)
diurno<-sample(c(0:9), 13, replace=TRUE)
crepuscular<-sample(c(0:5, 20:30), 13, replace=TRUE)
activity<-data.frame(phy.names, diurno, noturno, crepuscular, row.names=1)
ecophy(phy.teste, activity, eco="atividade", type="cladogram", color=grass)

#Exemplo_3_Tema:Outro

amazonia<-round(seq(0,200,length.out=13))
caatinga<-sample(30, 13, replace=TRUE)
cerrado<-sample(13, 13)
campos<-sample(90, 13)
mata_atl<-round(seq(200, 13, length.out=13))
ocorrencia<-data.frame(phy.names, amazonia, caatinga, cerrado, campos,
mata_atl, row.names=1)
ecophy(phy.teste, ocorrencia, eco="outro", pch=22, edge.lty=3,
contorno=FALSE, edge.color="darksalmon")
```

Código da função:

```
####################   ecophy    ########################

#################### a função em si ########################

ecophy<-function(phy , data, eco="dieta", index=0.7, type="phylogram", n=10
, color= "grape", font=1, place.legend="topleft", cex=3, pch=22,
edge.width=1, edge.lty=1, edge.color="black", show.tip.label=TRUE,
contorno="black", col.text="grey30")
{
  ## CRIANDO LISTA DE OPÇÕES DOS ARGUMENTOS:
  type<-match.arg(type, c("phylogram", "cladogram")) #estabelecendo as
opções de desenho da filogenia dentro do argumento type
  # CRIANDO GRADIENTES DE CORES:
  cor<-list() #criando uma lista
  cor[["cinza"]]<-colorRampPalette(c("gray88", "gray3"))(10) #criando
gradiente de cor cinza e guardando em uma das posições da lista
  cor[["coral"]]<- colorRampPalette(c("antiquewhite", "coral"))(10) #criando
gradiente de cor coral e guardando em uma das posições da lista
  cor[["grape"]]<- colorRampPalette(c("antiquewhite",
"maroon4"))(10)#criando gradiente de cor grape e guardando em uma das
posições da lista
  cor[["grass"]]<-colorRampPalette(c("antiquewhite", "darkgreen"))(10)
#criando gradiente de cor grass e guardando em uma das posições da lista
  cor[["sky"]]<- colorRampPalette(c("antiquewhite", "darkblue"))(10)#criando
gradiente de cor sky e guardando em uma das posições da lista
  color<-cor[[color]] #criando objeto color com a cor de gradiente da lista
cor, definido pelo argumento color da função
  ## CRIANDO RANGE DE CORES:
  col.range<-data.frame(ncolor=c(1:10), range.i=c(NA), range.f=c(NA))#cria
um data.frame com 3 colunas
  for (i in 1:10)
  {
    col.range[i, 2]<-round((i-1)/10, digits=4)#preenchendo a coluna 2
    col.range[i,3]<-round((i-0.1)/10, digits=4)#preenchendo a coluna 3
    col.range[10,3]<-1
  }
  ##VERIFICANDO SE O PACOTE plotrix E grDevices ESTÃO CARREGADOS. EM CASO
NEGATIVO, UMA MENSAGEM DE ERRO APARECERÁ:
  pac<-require("plotrix")
  if(pac==FALSE) {cat("\n", "#Erro# O pacote plotrix não está carregado!
\n")}
  pac2<-require("grDevices")
  if(pac2==FALSE){cat("\n", "#Erro# O pacote grDevices não está carregado!
\n")}
  ## CRIANDO DIVERSAS VERIFICAÇÕES DOS OBJETOS PEDIDOS NA FUNÇÃO PARA QUE
ESTA RODE NORMALMENTE:
  if(missing(phy) | missing(data)) #caso o usuário não forneça os objetos
```

```
phy ou data
  {stop("#ERRO# Nada foi feito. Objeto phy ou data não fornecidos.")} #para
a função e imprime mensagem de erro
  classe<-c(NA) #objeto que terá tamanho igual ao número de colunas do
data.frame
  for(i in 1:ncol(data))
  {
    classe[i]<-is.numeric(data[,i]) #vetor TRUE ou FALSE se os dados nas
colunas do data.frame for ou não numeric
  }
  if(sum(classe)!=ncol(data)) #verificando se o vetor classe possui só TRUE
ou algum FALSE (para rodar a função o número de TRUE somados tem que ser
igual ao número de colunas do data.frame)
  {
    stop("\n #ERRO# Nada foi feito: seus dados não são da classe correta
\n") ##para a função e imprime mensagem de erro
  }
  else #sum(classe)==ncol(data), então a função pode seguir
    if(class(phy)!= "phylo") #verificando se o input phy está no formato
desejado para a função rodar
    {
      stop("\n #ERRO# Nada foi feito: seus dados não são da classe correta
\n") ##para a função e imprime mensagem de erro
    }
  else #(class(phy)=="phylo"), então a função pode seguir
    if (sum(rownames(data)==phy$tip.label)==dim(data)[1]) #verificando se os
inputs têm os mesmos nomes, ou seja, se para cada espécie na filogenia, há
uma linha de informação no data.frame
    {
      ## AQUI AS VERIFICAÇÕES TERMINAM E A FUNÇÃO PODE RODAR:
      data[data==0]<-NA  #inserindo NA onde for zero
      data.prorp<-data.frame((phy$tip.label), row.names=1) #criando um
data.frame para preencher com dados de proporção de cada item do data
original
      for (i in 1:dim(data)[1])
      {
        soma<-sum(data[i,], na.rm=TRUE) #somando a quantidade de dado
presente para cada espécie
        if (soma >= n) #determinando se a análise deve ser feita ou não para
aquela espécie, dependendo se a quantidade de dados presentes for maior ou
igual ao limite estabelecido
        {for (j in 1:dim(data)[2])
        {data.prorp[i,j]<-round(data[i,j]/soma, digits=3)} #calculando a
proporção de cada valor do data.frame em cada linha
        }
      }
      colnames(data.prorp)=colnames(data)#renomeando as colunas de
data.prorp
      #PLOTANDO A FILOGENIA
      plot.phylo(phy, x.lim=
2*length(phy$tip.label)+((max(nchar(phy$tip.label)))-6)/2,
```

```
edge.width=edge.width, edge.lty=edge.lty, edge.color=edge.color,
show.tip.label=show.tip.label, type=type) #plotando a filogenia e colocando-
a na metade do dispositivo gráfico por meio de x.lim=2* o comprimento de phy
(o x.lim padrão é 1* comprimento de phy)
      ##CRIANDO AS CLASSIFICAÇÕES DOS TEMAS DE ECOLOGIA:
      if (eco=="dieta")
      {
        for(i in 1: nrow(data.prorp))
        {if(any(data.prorp[i,]>=index, na.rm =TRUE)) #criando uma condição
se algum valor de determinada linha for maior ou igual ao index
{text(x=length(phy$tip.label)+ncol(data)+1+((max(nchar(phy$tip.label)))-6)/2
, y=i, labels="Especialista",pos=4, col=col.text)} #colocando a
classificação de especialista se na coluna especialista houver valor maior
ou igual ao index
        }
      }
      if (eco=="atividade"| eco=="habito")
      {
        for(i in 1: nrow(data.prorp))
        {if(any(data.prorp[i,]>=index , na.rm =TRUE)) #criando uma condição
se algum valor de determinada linha for maior ou igual ao index
{text(x=length(phy$tip.label)+ncol(data)+1+((max(nchar(phy$tip.label)))-6)/2
, y=i, labels=colnames(data.prorp[which(data.prorp[i,]>=index)]), pos=4,
col=col.text)} #colocando a classificação se houver valor maior ou igual ao
index
        }
      }
      if (eco=="outro")
      {
        for(i in 1: nrow(data.prorp))
        {if(any(data.prorp[i,]>=index, na.rm =TRUE)) #criando uma condição
se algum valor de determinada linha for maior ou igual ao index
{text(x=length(phy$tip.label)+ncol(data)+1+((max(nchar(phy$tip.label)))-6)/2
, y=i, labels="*", pos=4, col=col.text)} #colocando a classificação se
houver valor maior ou igual ao index
        }
      }
      for(u in 1:10)   #trocando as proporções do data.prorp por números
inteiros correspondentes ao número da cor no gradiente de cores escolhida
      {
        data.prorp[data.prorp==1]<-10  #se a proporção for igual a 1, ou
seja, 100%, ele recebe a cor de valor 10, que é equivalente à cor mais
escura
      }
      for(u in 1:10)
      {
        data.prorp[data.prorp<=(u-0.1)/10]<-u #repetindo a ação do for
acima, agora com números diferentes de 1
      }
      for(z in 1:length(phy$tip.label))
```

```
      {
        for(i in 1:ncol(data.prorp))
        {
          if(is.na(data.prorp[z,i]))
          {
points((length(phy$tip.label)+((max(nchar(phy$tip.label)))-6)/2)+i, z,
pch=pch, bg="white", cex=cex, col=contorno)
          }
          else
points((length(phy$tip.label)+((max(nchar(phy$tip.label)))-6)/2)+i, z,
pch=pch, bg=color[data.prorp[z, i]], cex=cex, col=contorno)
        }
      }
      ##CRIANDO AS POSIÇÕES DAS LEGENDAS:
      leg<-list() #criando uma lista
      leg[["topleft"]]<-
list(xl=0,yb=length(phy$tip.label)-2,xr=0.5,yt=length(phy$tip.label),align="
lt",gradient="y")##criando a legenda topleft e guardando-o na lista leg
      leg[["bottomleft"]]<-
list(xl=0,yb=1,xr=3,yt=1.2,align="rb",gradient="x")##criando a legenda
bottomleft e guardando-o na lista leg
      place<-leg[[place.legend]] #criando objeto place a partir da opção no
argumento "place.legend" e da lista leg
      legend<-color.legend(place$xl,place$yb,place$xr, place$yt,
place$align, place$gradient, legend=c("0%","100%"),rect.col=color,cex=1 )
#inserindo legenda
      ##CRIANDO NOMES NO EIXO x
      pontos<-rep(NA, times=ncol(data)) #criando vetor pntos apenas com NA
      for(i in 1:ncol(data))#fazendo um loop que irá de 1 a número de
colunas no data.frame
      {
        pontos[i]<-
(length(phy$tip.label)+((max(nchar(phy$tip.label)))-6)/2)+i #criando a
posição e cada legenda do eixo x
        axis(1, at=c(pontos), labels=colnames(data), col="white", font=font,
las=3) #plotando o eixo x, com as posições do vetor pontos, com os nomes das
colunas do data.frame e deitado
      }
    }
  else
  {
    stop("\n #ERRO# Nada foi feito: as espécies no data.frame não
correspondem às espécies na filogenia \n") #parando e criando mensagem de
erro que aparecerá no console caso a igualdade de nomes no data.frame e
filogenia não seja cumprida
  }
```

Help Código_da_função