DANIEL ANDRÉS CHIRIVI JOYA

Doutorando em zoologia, LAL, Instituto de Biociências, USP.

Título da minha tese: “Taxonomia e filogenia do gênero *Phrynus* Lamarck (Amblypygi, Phrynidae”, Orientado pelo Dr. Ricardo Pinto da Rocha.

Interese em sistematica e ecologia de aracnídeos, principalmente aranhas e amblypígidos.

Meus trabalhos

Linque para a página com os meus exercícios resolvidos: Exercícios Daniel Chirivi.

Proposta de trabalho final

Pincipal

Em filogenia e analises taxonômicas, são usados caracteres como evidencia das relações entre clados e a definição dos mesmos, um dos princípios que se assomem no processo é a independência de caracteres (sereno, 2007), porém, já é conhecido que os caracteres não são totalmente independentes e que um es estado de um caráter pode se relacionar com o estado de outro, já seja por uma proximidade espacial, um relacionamento funcional, ou por desequilibro de ligação, entre outras (Maglia et. Al., 2014).

O conhecimento da relação entre caracteres pode ser uma informação valiosa para os sistematas e taxonomistas, esta pode ser levada em conta na hora de analisar coerentemente uma filogenia, estabelecer homologias primarias, e um bom numero de de outras utilidades dependendo da área na qual seja a área de estudo.

Proposta:

Neste trabalho se propõe implementar analises exploratórias sobre os conjuntos de caracteres que se presentam normalmente numa analise filogenética, tentando descobrir patrões na presencia dos estados de caráter, mediante a procura de caracteres relacionados. Isto mediante uma função implementável na linguagem R, que seja facilmente utilizável seguindo alguns parâmetros simples.

Utilidade e aplicação:

O principal objetivo da função é permitir ao pesquisador reconhecer conjuntos de carácteres cujos estados se encontrem relacionados, e dizer que mostrem um padrão de correlação entre eles, de modo que se no caráter A se apresenta o estado de caráter A1, exista uma tendência a que no caráter B se apresente o estado B1.

E importante ressaltar que esta analise não é igual a uma analise filogenética nem pretende atribuir

relações de causa-efeito entre os carácteres. Trata-se simplesmente de uma análise exploratória sobre as frequências dos estados de um caráter em relação a outros.

Porém, o pesquisador poderá ter um melhor conhecimento dos caracteres que esteja utilizando, poderá descrever tendências de aparição dos estados de caráter, e estabelecer conjuntos de estados que caracterizam táxons. Além de descrever a variação dos caracteres e ressaltar aqueles que sejam independentes, e finalmente dar a esta variação uma interpretação à luz da filogenia, interações ecologias, distribuições geográficas, entre outras.

Estrutura da função:

A função deverá constar de um parâmetro que receba uma matriz onde as colunas sejam os caracteres, e as filas sejam as espécies, as colunas deverão conter o es estados de caráter para cada um dos caracteres por espécie.

A função deverá pegar a informação e fazer testes pareados avaliando a influencia entre os caráteres, e retornar como resultado aqueles que sejam significantes. Para isto se tomarão os caracteres como variáveis (conjuntos de dados). Estes conjuntos dados acostumam ser categóricos não ordinais, o que impede fazer testes de correlação tradicionais, ou os testes que oferece a função “lm()” do R. Dada esta condição serão usados Testes Exactos de Fisher para variáveis qualitativas. Este teste deverá ser realizado para cada uma das variáveis da matrix em contraste com as outras, gerando um data-frame com os valores de significância do teste, e retornando este ao final do procedimento.

O pacote “tree” do R tem uma opção gráfica para visualizar grupos de variáveis não paramétricas, é dizer categóricas, chamada “árvores de regressão ”, esta apresenta um dendrograma agrupando as variáveis em função de uma. Além dos testes de Fisher , a função proposta terá um argumento para solicitar fazer este procedimento, em função a uma variável desejada, é dizer o caráter que seja o alvo de estudo do pesquisador.

LITERATURA CITADA

Maglia , A. Leopold, J. & Ghatti, V. (2014). Identifying Character Non-Independence in Phylogenetic Data Using Data Mining Techniques, in: APBC '04 Proceedings of the second conference on Asia-Pacific bioinformatics. Vol. 29. pp. 181-189.

Sereno, P. 2007. (2007) . Logical basis for morphological characters in phylogenetics. Cladistics, 23. 565-587.

Plano B

Durante os trabalhos e modelação de analise filogenética os pesquisadores acostumam usar matrizes de caracteres, as vezes se faz necessário criar uma matriz somente para testar métodos filogenéticos, analises sobre filogenias, ou simplesmente para funções pedagógicas.

Proposta:

Se propõe criar uma função que, construa matrizes aleatórias de acordo aos requerimentos do usuário. A função deverá construir matizes onde o usuário possa escolher o numero de espécies, o numero de caracteres e seus respectivos números de estados de caráter. Também deverá definir se um caráter deve estar relacionado com outro.

Utilidade e aplicação:

A função deverá proporcionar uma matriz que seja utilizável em programas de analise filogenética como TNT, ou POY, que contenha os requerimentos do usuário. A função deve ser rápida de utilizar, e poupar tempo ao usuário.

Estrutura da função:

Deverá conter, ao menos três argumentos básicos, o numero de espécies, o numero de caracteres e o numero de estados de caráter. Estes deverão ser digitados manualmente pelo usuário. Deverão ter implementados argumentos que determinem o número estados de caráter desejados e para que quantidade de caracter, de modo que o usuário possa especificar que deseja uma matriz com um um número “X” de caracteres, dos quais uma quantidade “Y” devera conter um numero “Z” de estados de caráter, em quanto outra porção da matriz contenha um numero “W” de estados.

A complexidade da função esta em a versatilidade que possa ter, e a liberdade que da ao usuário de escolher os requerimentos da matriz.

Um argumento final pode pedir que a matriz seja guardado como um arquivo em formato csv.

Página de Ajuda

```
function "charac.ind" to do independence analyses of a matrix of
phylogenetic characters

Description:

 This function take a phylogenetic matrix of characters, and makes a fisher
test  for categorical variables
  contrasting all the caracters and return  the p  value for each contrast,
the function  present a subset with the significant contrasts.
  It is possible ask for to do a regression tree, for to find groups of
related characters

Usage:

  charac.ind (charac, reg.tree= F, m.var= 1)

Arguments:

   charac           a matrix of characters and their states for each
species, placing species by rows, and characters by columns
   reg.tree         if reg.tree =  TRUE, the function makes a regression
tree using the function "tree", from the package "tree", contrasting all
characters of the imput matrix
   m.var             a number indicating the position of the starting
caracter in the imput matrix, for the regression  tree
Details:
  Matrix must be bidimentional, with categorical caracter states, the
species must be placed by rows
```

```
  and characters by columns, without names for rows and columns.
  For the option of regression tree, is necesary specify the character from
which the contrasts will be started,
  this character must be indicated in the argument "m.var" giving the
position of the character in the imput matrix.
  For the correct usage of option reg.tree it is necesary install de package
"tree".

Value:

  A list with  the following components:
  out.matrix:    matrix with all p value, for each contrast
  sing:          text: "Fisher`s Exact Test for Count"
  resume:        sub set of "out.matrix" with the significant p values
  Test:          text: "Fisher`s Exact Test for Count"
  Hypothesis:    text: "Alternative hypothesis: two.sided"
  Significan:    text: "Significancie level: * < 0.05, ** < 0.01, *** <
0.001"

Warning:
  This function is restricted by the working conditions of fucntions
"fisher.test" and "tree".

Author(s):

  Daniel Chirivi Joya
  Zoology department
  Universidade de Sao Paulo

References:

  Chap, T. (2003) Introductory bioestatistics. John Wiley & Sons, Inc.
Hoboken. New Jersey. Pages 229-230.
  Van-Emden, H. (2008) Statistics for Terrified Biologists. Blackwell
Publishing Ltd. Australia. Pages 291-297.
  Wei-Yin, L. (2011) Classification and regression trees. WIREs Data Mining
and Knowledge Discovery 1, 14-23.
See Also:

  help(fisher.test)
   help(tree)

Examples:

 ## charac<- rcharac.mx(70,60,50,10)
 ## charac.ind(charac, reg.tree = F, m.var= 2)
```

Código da Função

Pra testar esta função, pode criar uma matriz aleatoria usando a função "rcharac.matrix" ubicada na secção "plano B" desta pagina.

```
charac.ind <- function(charac, reg.tree= F, m.var= 1) {  #  inicia a função com três argumentos, a    matriz, opção da arbore de regressão e o caracter de inicio da arvore de regressão.

n.char <- length(charac[1,]) # calcula o número de caracteres com a longitude das filas.
n.row <- sum(seq(1,n.char-1)) # calcula o número de caracteres que vai ter a matriz interna da função
out.matrix <- matrix(data = NA, nrow =n.row, ncol = 2 ) # cria uma matriz para ubicar os dados finais.
p.matrix <- matrix(data = NA, nrow =n.char, ncol = n.char )  # cria uma matriz que será usada para colocar  os dados produzidos pela função.
name.matrix <- matrix(data = NA, nrow =n.char, ncol = n.char ) # cria uma matriz que vai conter os nomes   de cada comparação

dimnames(out.matrix) <- list(NULL, c("char_a vs char_b","p-value")) #  cria uma lista com os nomes da matriz final.

 for(b in seq(1, n.char-1)){ # inicia o ciclo de comparações desde 1 até o número de caracteres menos um
# este vai pegar cada carater da matriz de do usuario
for(i in seq(b+1, n.char)) { # inicia um segundo ciclo que vai pegar o caracter seguinte ao  seleccionado    no ciclo anterior.
 fisher.int <- fisher.test(table(charac[,b], charac[,i]), simulate.p.value=TRUE,B=1e3) # faz um teste de   fisher de um caracter com outro.
p.matrix[b,i] <- round(fisher.int$p.value, 4) # e vai colocando o p redondeado numa matriz.
  } # fecha o primiero ciclo.
 } # fecha o segundo ciclo.

p.matrix2 <- na.exclude( as.vector(t(p.matrix))) # transforma a matriz com os valores p num vector    excluido os NA.

p.matrix3 <- rep(" ", length(p.matrix2) ) # cria um vector vacio com a mesma longitude do p.matrix2.

for(k in 1: length(p.matrix2)){ # come??a um ciclo para colocar  os valores de significancia
 if (p.matrix2[k]< 0.05) { p.matrix3[k]<- paste(p.matrix2[k],"*")} # Coloca aos valores menores a 0.05 um  "*".
 if (p.matrix2[k]< 0.01) { p.matrix3[k]<- paste(p.matrix2[k],"**")}# Coloca aos valores menores a 0.01  dois "*".
 if (p.matrix2[k]< 0.001) { p.matrix3[k]<- paste(p.matrix2[k],"***")}#
```

```
Coloca aos valores menores a 0.001  tres "*".
 if (p.matrix2[k] >= 0.05) {p.matrix3[k]<- p.matrix2[k]} # coloca no vector
os valores maiores o iguais a 0.05, sem modificações.
 } # fecha o ciclo.

 out.matrix[,2] <- p.matrix3 # coloca na matriz de saida, na segunda coluna,
os valores de p, ja com o  indicativo de significância.



for(a in seq(1, n.char-1)){ # cria um ciclo  da mesma longitude  que o
numero de caracteres menos um
 for(j in seq(a+1, n.char)) { #  cria um segundo ciclo que vai ser sempre
uma posição adiante do ciclo  anterior
name.matrix[a,j] <-  paste("char", a, "Vs", "char", j, sep = " " ) # cria
uma matriz con os nomes  de   cada comparação anteriormente feita, exemplo :
"character 1 vs 2".
 } # fecha o primeiro ciclo.
  } # fecha o segundo ciclo.



 name.matrix2 <- na.exclude( as.vector(t(name.matrix))) # transforma a
matriz anterior en un vector excluindo os NA.

out.matrix[,1] <- name.matrix2 # coloca os nomes na matriz de saída.
 as.data.frame(out.matrix)  # transfoma em um data.frame a matriz de saida,
para ele ter um aspecto mais   organizado.
  ##########################################

resume <- subset(out.matrix, out.matrix[,2]< 0.05) #  subset com os valores
significantes.
sign <- "Significant contrast" # obejeto com o nome que vai ter esta parte
do teste no objeto de saída.
Test <- "Fisher`s Exact Test for Count" # obejeto com o titulo do test.
Hypothesis <- "Alternative hypothesis: two.sided" # objeto com uma mensajem
que indica que o teste é a   duas vias.
Significan<-  "Significancie level: * < 0.05, ** < 0.01, *** < 0.001"  #
objeto com as explicações dos    indicadores de significância.

results <- list(out.matrix,sign, resume, Test, Hypothesis, Significan) #
cria um objeto tipo listado com  os resultados finais.

##################################################
if (reg.tree== TRUE)  { # condicional, se o argumento de  solicitude da
arvore de regressão e verdadeiro

library(tree) # chama o pacote "tree".

charac <- charac # cria o objeto "charac" dentro da função.
contrastes <- c(paste("charac[",m.var,"]~") , paste( "charac[",
seq(1,n.char-1),"] +"),    paste("charac[",n.char,"]"))  # cria um objeto
```

```
para que escreva os contrastes a serem feitos pela função  "tree".
# normalmente o usuario deve colocar  estes contrastes manualmente na função
mas ele calcula quantas    variáveis tem a matriz e indica que faza um
contraste com todos
nm.var <- paste("charac[,",m.var,"] +")  # como a fun??ao tem uma variavel
central, sobre a qual iniciam  os contrastes, a posição de esta variavel o
carater, dentro da matriz do usuario debe ser indicada
# esta linha simula o texto que vai ter esa variavel produto da linha 71
deste script, para logo poder   tirar ese texto, do contrario ficaria
repetida.
contrastes2 <- subset(contrastes, contrastes != nm.var ) # faz um subset do
objeto contraste excluindo o  texto repetido para que a variavel central
solo seja nomeada uma vez.

tree1 <- tree(contrastes2) # faz uma arvore de regressão com o objeto
contrastes2, nesta   linha se simula o texto que normalmente seria
introduzido pelo usuario manualmente

plot(tree1) # plota a arvore.
text(tree1) # coloca os nomes na arvore.
 } # fecha o ciclo


return(results)  # retorna o objeto de saída.
} # fecha a função.
```

Página de Ajuda plano B

```
function "rcharac.mx" to do aleatory  character matrix for phylogenetic
analyses

Description:

This function build a matrix of characters, acording the requeriments of the
usuary using aleatory character
   states, allowing select the number of species, number of characters, and
number of character states for each character

Usage:
  charac.mx(n.sp, n.char, cbin= n.char, ctrin=0, ctet=0, cpen=0, csex=0,
cop1= c(0,0), cop2= c(0,0))
  charac.ind (charac, reg.tree= F, m.var= 1)

Arguments:

   n.sp             number of species (rows).
   n.char            number of characters (columns).
   cbin              number of characters with two character states, by
default is equal to number of characters.
```

```
   ctrin             number of characters with tree character states.
   ctet              number of characters with four character states.
   cpen              number of characters with five character states.
   csex              number of characters with six character states.
   cop1              character optional 1:  allow a completly free selection
of number of character states for a
                     specific number of caracters (columns).  The first
argument is te number of characters and the second is the number of
character states.
   cop2              character optional 2:  allow a completly free selection
of number of character states for a
                     specific number of caracters (columns).  The first
argument is te number of characters and the second is the number of
character states.
Details:

  The function fills the matrix with binary characters by default.
  If number of characters is bigger than sum of characters of all arguments
specified by the usuary, the matrix is completed by NA.
Value:

  A bidimentional matrix with species placed by rows and characters by
columns.

Warning:

  The sum of characters of all arguments specified by the usuary can´t be
superior to number of characters specified in the argument n.char.

Author(s):

  Daniel Chirivi Joya
  Zoology department
  Universidade de Sao Paulo

References:

  Scotland, R. & Pennington, T. (2000) Homology and Systematics: Coding
Characters for Phylogenetic   Analysis. Taylord and Francis. London. 219Pp.

Examples:

  ##rcharac.mx(n.sp=100, n.char=70, cbin= 10, ctrin=10, ctet=10, cpen=10,
csex=10, cop1= c(10,8), cop2= c(0,0))
  ## rcharac.mx(70,200,100,50,50)
```

Código da Função plano B

```
rcharac.mx <- function(n.sp, n.char, cbin= n.char, ctrin=0, ctet=0, cpen=0,
csex=0, cop1= c(0,0), cop2=    c(0,0) ){ # cria os parametros da função,
número de sp, número de caracteres, número de caracteres binarios, número de
caracteres com tres estados, cuatro estados, cinco estados, seis estados,
caracter opcional 1 y 2.

a0 <- c(0,1) # cria um objeto com estados, 0 e 1
a1 <- c(0,1,2) # cria um objeto com  três estados, 0,  1 e 2
a2 <- c(0,1, 2,3) # cria um objeto com  quatro estados de caracter.
a3 <- c(0,1, 2,3,4) # cria um objeto com  cinco estados de caracter
a4 <- c(0,1, 2,3,4,5) # cria um objeto com  seis estados
ax1 <- seq(from=0, to= (cop1[2])-1) # cria um objeto com  uma sequência
desde a posição 2 do argumento   cop1 (número de estados.
ax2 <- seq(from=0, to= (cop2[2])-1) # cria um objeto com  uma sequência
desde a posição 2 do argumento  cop2 (número de estados).
example <- matrix(data = NA, nrow = n.sp , ncol = n.char) #  cria uma matriz
com o numero de especies e número de caracteres

for(i in 1:cbin) {      # inicia o ciclo  desde 1 até o número de caracteres
binarios
  example[,i] <- sample(a0, n.sp, replace = T)} # coloca na matriz na
valores aleatorios de  0 e 1

if(cbin==n.char | (ctrin==0) & (ctet==0) & (cpen==0) & (csex==0) &
(cop1[1]==0) & (cop2[1]==0)) { # se o número de caracteres binarios e igual
ao número
  # total de caracteres ou se não tem  valores nos outros argumentos
  return(example)} # retorna a matrix

else {  # se não, continua a função

  for(a in (cbin+1):(cbin+ctrin)) { #  com um  ciclo desde onde terminaram
os caracteres binarios, até o numero de caracteres com três estados.
    example[,a] <- sample(a1, n.sp, replace = T)}} # continua dando valores
na matriz, desta vez com caracteres com três estados.
if((ctet==0) & (cpen==0) & (csex==0) & (cop1[1]==0) & (cop2[1]==0)){ # se os
seguintes argumentos não têm valores
    return(example)}  # retorna a matriz.
else { # se não
  for(b in (cbin+ctrin+1):(cbin+ctrin+ctet)) { # começa um ciclo desde a
ultima posição com valores até o número de catacteres de quatro estados
    example[,b] <- sample(a2, n.sp, replace = T)}} # continua enchendo a
matriz com caracteres de quatro estados
if((cpen==0) & (csex==0) & (cop1[1]==0) & (cop2[1]==0)){ # se os seguintes
argumentos estão vazios
  return(example)}  # retorna a matrix
else { # se não
  for(c in (cbin+ctrin+ctet+1):(cbin+ctrin+ctet+cpen)) { # começa um ciclo
```

```
desde a ultima posição com valores até o número de catacteres de cinco
estados
    example[,c] <- sample(a3, n.sp, replace = T)}} # continua enchendo a
matriz com caracteres de cinco estados.
if((csex==0) & (cop1[1]==0) & (cop2[1]==0)){return(example)}# se os
seguintes argumentos estao vazios retorna a matriz
else{ # se não
  for(d in (cbin+ctrin+ctet+cpen+1):(cbin+ctrin+ctet+cpen+csex)) {  #começa
um ciclo desde a última posição com valores até o número de catacteres de
seis estados
    example[,d] <- sample(a4, n.sp, replace = T)}}# continua enchendo a
matriz com caracteres de seis estados.

if((cop1[1]==0) & (cop2[1]==0)){return(example)} # se os seguintes
argumentos estão vazios retorna a matriz
   else{ # se não
     for(e in
(cbin+ctrin+ctet+cpen+csex+1):(cbin+ctrin+ctet+cpen+csex+cop1[1])) { #
começa um ciclo desde a ultima posição com valores até o número de
catacteres no  no argumento cop1 (posição 1)
       example[,e] <- sample(ax1, n.sp, replace = T)}}# continua enchendo a
matriz com caracteres com o número de estados requerido pelo usuario.
if( cop2[1]==0){return(example)} # se os seguintes argumentos estao vazios
retorna a matriz
        else{# se não
          for(f in
(cbin+ctrin+ctet+cpen+csex+cop1[1]+1):(cbin+ctrin+ctet+cpen+csex+cop1[1]+cop
2[1])) { #começa um ciclo desde a ultima posição com valores até o número de
catacteres no  no argumento cop2 (posição 1)
            example[,f] <- sample(ax2, n.sp, replace = T)}}  # continua
enchendo a matriz com caracteres com o número de estados requerido pelo
usuario
   return(example)} # fecha o último ciclo e retorna a matriz
```