Leonardo Chaves

Doutorando no Programa de Etnobiologia e Conservação da Natureza - UFRPE

Laboratório de Ecologia e Evolução de Sistemas Socioecológicos LEA

Desenvolve trabalhos na área de Etnobiozoologia, com particular interesse em compreender os fatores que norteiam a atividade de caça de subsistência em comunidades tradicionais e os efeitos em cascata dessa atividade.

Exercício

PROPOSTA “A”

Na pesquisa etnobiológica é muito comum o uso do Índice de Saliência de Smith (ver Sutrop, 2001) como descritor quantitativo da importância cultural de um determinado recurso natural (planta medicinal, animal caçado, etc) em relação a outros recursos de um mesmo domínio cultural. O índice de Saliência é calculado em função da frequência de citações que um dado recurso recebe e da ordem em que ele é citado em listas livres elaboradas a partir de entrevistas realizadas com pessoas que compõem uma comunidade tradicional estudada. O Índice de Saliência de Smith foi ainda corrigido por Smith Borgatti em 1997 e uma nova adaptação ao seu cálculo foi proposta por Sutrop em 2001. O objetivo da função, portanto, será calcular o valor do índice de Saliência para cada item de um conjunto de listas livres.

Como entrada, o usuário deverá fornecer um data frame com a identificação dos informantes entrevistados (nome, por exemplo) e os itens citados por cada um deles (na ordem citada). E deverá indicar em um dos argumentos da função por meio de qual fórmula o índice deverá ser calculado (Smith, 1993, Smith e Borgatti, 1997 ou Sutrop, 2001).

Como resultado, a função retornará uma lista com todas as plantas citadas acompanhadas do valor do Índice de Saliência e a probabilidade do valor calculado para cada item ser explicado pelo acaso.

> Leonardo, Essa proposta parece muito interessante, mas estou tendo dificuldade de julgar o grau de dificuldade dela por falta de informações sobre o índice. Não consegui achar as referências citadas; por favor inclua as referências bibliográficas se citar outros textos. Se puder incluir as fórmulas que serão usadas no próprio texto, melhor ainda.
>
> Estou em dúvida sobre o tipo de entrada que sua função requer. Poderia dar um exemplo? Um exemplo da vida real pode servir para pensar que tipo de estrutura de dados se adequa melhor a esses dados. Você disse que receberia um data.frame, mas um data.frame corresponde a dados tabelados, enquanto a entrada que você descreve parece envolver uma lista ordenada de nomes de planta para cada pessoa, sendo que não necessariamente

as listas têm o mesmo tamanho. Talvez processar esses dados seja uma parte interessante do desenvolvimento da função.

Por enquanto são esses os comentários que tenho a fazer. Espero respostas para detalhar melhor essa proposta!

—-*Mali*

PROPOSTA “B”

Um grande número de etnobiólogos, ao investigar um determinado domínio cultural por meio de entrevistas, assume como premissa que a ordem de citação dos itens de uma lista livre reflete sua importância cultural. Alguns trabalhos, entretanto, têm demonstrado que na elaboração de tais listas existe uma forte influência de vieses cognitivos. Tais trabalhos têm levantado a possibilidade de que possíveis falhas durante a entrevista podem levar o informante a elaborar sua lista guiado por pistas semânticas. Nesse cenário, podemos citar como exemplo um informante que, ao ser questionado a respeito de plantas medicinais, no lugar de citar as espécies que conhece na ordem de importância cultural (conforme se pressupõe) as agrupa, por exemplo, por seu alvo de tratamento ou cita em sequência plantas cuja parte utilizada seja a mesma.

O objetivo da função, portanto é verificar se os itens citados em uma lista livre se apresentam agrupados em função de algum fator indicado pelo pesquisador.

O usuário deverá fornecer um data frame com a identificação dos informantes entrevistados, os itens da lista livre na ordem citada e fatores correspondentes aos possíveis vieses cognitivos cuja influência se pretende avaliar.

A função realizará um teste de permutação e retornará a probabilidade de existir agrupamento dos itens da lista em função de cada um dos fatores presentes no data frame.

Leonardo, Que proposta bacana! Parece tecnicamente complicada, mas tem uma motivação bem clara, e pode ser usada muitas vezes em um estudo sobre esse assunto.

A dúvida mais importante sobre esta proposta, que não está esclarecida no texto, é: qual é a análise estatística que você vai fazer para determinar se os nomes estão agrupados? Se você já sabe qual é o modelo estatístico que vai usar, descreva ele na proposta. Senão, comece a pesquisar o mais rápido possível.

Ainda estou com a mesma dúvida sobre o objeto de entrada da proposta A. Aqui mais ainda me parece que oque voê quer não é um data.frame, mas uma lista de listas. Além disso, nesta proposta além dos nomes das plantas você precisa de uma referência de qual classificação cada uma dela tem. Há várias formas de incluir esse dado: pode haver uma tabela separada ligando cada planta a sua classificação; ou as listas de nomes podem ser

substituídas por listas de classificação (ex: em vez de “cidreira”,“camomila”,“boldo”, podemos receber só “chá”,“chá”,“chá”); ou as listas podem incluir ambas as informações; etc etc. Na prática, qual desses modelos vai ser melhor depende das suas decisões e de qual modelo estatístico vai utilizar.

Aguardo essas informações faltantes. :)

—-*Mali*

FUNÇÃO SALIENCE

```
salience<- function (ll, indice="Smith93")
    {
        colnames(ll)<-c("nomes", "item")        #Renomeei as colunas pra
facilitar meu trabalho
        llord<-ll[order(ll$nomes),]            #Algumas funções que usei
retornam os resultados em ordem alfabética, por isso, achei melhor já partir
da mesma forma
        ninf<-length(levels(ll$nomes))        #Número de listas (informantes
entrevistados)
        nintens<-length(levels(ll$item))        #Número total de itens
        tamlista<-as.vector(table(ll$nomes))    #Cria um vetor com o tamanho
de cada lista

        #____________________Preparação para o Monte Carlo_____________

        listsimul<-as.character(seq(1, length(levels(ll$item))))    #Cria
uma lista de itens com o mesmo número que os itens totais
        size.list<-seq(max(tamlista), min(tamlista))            #Cria um
vetor com opções de tamanhos de lista (entre a maior e a menor lista do
dataframe de entrada)

        matrlista<-(matrix(data=NA, nrow=nintens, ncol=ninf))        #Cria
matrizes para armazenar os itens sorteados,
        matrank<-(matrix(data=NA, nrow=nintens, ncol=ninf))        #o
ranking do item,
        resultMC<-(matrix(data=NA, nrow=nintens, ncol=300))        #os
valores de saliência simulados.


if(indice=="Smith93")#_____________________________Smith93_____________________
_______
            {
                resultLRL<-(matrix(data=NA, nrow=nintens, ncol=ninf))
#Cria uma matriz para armazenar os rankings/cálculo parcial do índice de
saliência
```

```
                for(i in 1:ninf)
#Preenche a matriz com os valores de ranking/cálculo parcial do índice de
saliência
                    {
                        resultLRL[,i]<-c((tamlista[i]-seq(0,
(tamlista[i]-1)))/
                        tamlista[i], rep(NA,times=(nintens-tamlista[i])))
#Cálculo parcial Smith93
                    }
                llord$LRL<-100*(as.numeric(na.omit(as.vector(resultLRL))))
#Transforma a matriz em vetor (retirando os NAs)
                Smith1993<-as.numeric(tapply(llord$LRL, llord$item,
sum)/ninf)    #Somas os valores, finalizando o cálculo
                dfSmith1993<-data.frame(levels(llord$item), Smith1993)
#Monta o DF de saída
                Exit<-dfSmith1993[order(dfSmith1993$Smith1993,
decreasing=T),]      #Organizar o DF em função dos valores de saliência
calculados
                colnames(Exit)<-c("Itens", "Smith93")

                #___________Monte Carlo Smith93_____________________________

                for(j in 1:300)                                      #Decidi
por poucas simulações, pois em cada cíclo, varios valores são calculados
                    {
                        for(k in 1:ninf)
                            {
                                num.item<-sample(size.list, 1)
#Escolhe aleatoriamente quantos itens distintos serão citados na lista
                                listaparc<-sample(listsimul, size=num.item)
#Cria uma lista simulada com tamanho sorteado na linha anterior
                                matrlista[,k]<-c(listaparc,
rep(NA,times=(nintens-length(listaparc))))       #Preenche a matriz com os
itens sorteados
                                matrank[,k]<-c((length(listaparc)-seq(0,
(length(listaparc)-1)))/           #Preenche a matriz com os
rankings/valores parciais do cálculo
                                length(listaparc),    rep(NA,times=(nintens-
length(listaparc))))
                            }
                        resA<-as.character(na.omit(as.vector(matrlista)))
#Transforma a matriz em vetor
                        resB<- 100*(as.numeric(na.omit(as.vector(matrank))))
#Transforma a matriz em vetor
                        resAB<-data.frame(resA, resB)
#Monta o DF
                        resABC<-as.numeric(tapply(resAB$resB, resAB$resA,
sum)/ninf)        #Finaliza o cálculo
                        resultMC[,j]<-c(resABC, rep(NA,times=(nintens-
length(resABC))))        #Preenche a matriz com os resultados
```

```
                    }

            }

########_____A seguir (até o Cálculo do valor de p), o código segue a mesma
sequência, com pequenas alterações, ajustando-se aos
índices.#######################

if(indice=="Smith95")#_____________________________Smith95_____________________
_________________
            {
                resultLRL1<-(matrix(data=NA, nrow=nintens, ncol=ninf))
                for(i in 1:ninf)
                    {
                        resultLRL1[,i]<-c((tamlista[i]-seq(1,
tamlista[i]))/(tamlista[i]-1), rep(NA,times=(nintens-tamlista[i])))
                    }
                llord$LRL1<-(as.numeric(na.omit(as.vector(resultLRL1))))
                Smith1995<-as.numeric(tapply(llord$LRL1, llord$item,
sum)/ninf)
                dfSmith1995<-data.frame(levels(llord$item), Smith1995)
                Exit<-dfSmith1995[order(dfSmith1995$Smith1995,
decreasing=T),]  ##Organizar
                colnames(Exit)<-c("Itens", "Smith95")

                #___________Monte Carlo Smith95_____________________________

                for(j in 1:300)
                    {
                        for(k in 1:ninf)
                            {
                                num.item<-sample(size.list, 1)
#Escolhe aleatoriamente quantas plantas o informante vai citar
                                listaparc<-sample(listsimul, size=num.item)
#Cria uma lista simulada com tamanho sorteado na linha anterior
                                matrlista[,k]<-c(listaparc,
rep(NA,times=(nintens-length(listaparc))))
                                matrank[,k]<-c((length(listaparc)-seq(1,
length(listaparc)))/(length(listaparc)-1), rep(NA,times=(nintens-
length(listaparc))))
                            }
                        resA<-as.character(na.omit(as.vector(matrlista)))
                        resB<- as.numeric(na.omit(as.vector(matrank)))
                        resAB<-data.frame(resA, resB)
                        resABC<-as.numeric(tapply(resAB$resB, resAB$resA,
sum)/ninf)
                        resultMC[,j]<-c(resABC, rep(NA,times=(nintens-
length(resABC))))
                    }

            }
```

```
if(indice=="Smith97")#_____________________________Smith97_____________________
_____________________________
            {
                resultLRL.1<-(matrix(data=NA, nrow=nintens, ncol=ninf))
                for(i in 1:ninf)
                    {
                        resultLRL.1[,i]<-c((tamlista[i]-seq(1,
tamlista[i])+1)/tamlista[i], rep(NA,times=(nintens-tamlista[i])))
                    }
                llord$LRL.1<-(as.numeric(na.omit(as.vector(resultLRL.1))))
                Smith1997<-as.numeric(tapply(llord$LRL.1, llord$item,
sum)/ninf)
                dfSmith1997<-data.frame(levels(llord$item), Smith1997)
                Exit<-dfSmith1997[order(dfSmith1997$Smith1997,
decreasing=T),]  ##Organizar
                colnames(Exit)<-c("Itens", "Smith97")


                #___________Monte Carlo Smith97_____________________________

                for(j in 1:300)
                    {
                        for(k in 1:ninf)
                            {
                                num.item<-sample(size.list, 1)
#Escolhe aleatoriamente quantas plantas o informante vai citar
                                listaparc<-sample(listsimul, size=num.item)
#Cria uma lista simulada com tamanho sorteado na linha anterior
                                matrlista[,k]<-c(listaparc,
rep(NA,times=(nintens-length(listaparc))))
                                matrank[,k]<-c((length(listaparc)-seq(1,
length(listaparc))+1)/length(listaparc), rep(NA,times=(nintens-
length(listaparc))))
                            }
                        resA<-as.character(na.omit(as.vector(matrlista)))
                        resB<- as.numeric(na.omit(as.vector(matrank)))
                        resAB<-data.frame(resA, resB)
                        resABC<-as.numeric(tapply(resAB$resB, resAB$resA,
sum)/ninf)
                        resultMC[,j]<-c(resABC, rep(NA,times=(nintens-
length(resABC))))
                    }
            }
if(indice=="Sutrop2014")#_____________________________Sutrop2014_____________
_____________________________
            {
                tt<-table(ll)
                freq<-apply(tt, 2, sum)
                freq1<-as.data.frame(freq)
                freq2<-freq1$freq^2
```

```
                resultsalcog<-(matrix(data=NA, nrow=nintens, ncol=ninf))
                for(i in 1:ninf)
                    {
                        resultsalcog[,i]<-c(seq(1,
tamlista[i]),rep(NA,times=(nintens-tamlista[i])) )
                    }
                llord$salcog<-(as.numeric(na.omit(as.vector(resultsalcog))))
                Sutrop2014<-as.numeric(tapply(llord$salcog, llord$item,
sum))
                Sutrop2014.<-freq2/(Sutrop2014*ninf)
                dfSutrop2014<-data.frame(levels(llord$item), Sutrop2014.)
                Exit<-dfSutrop2014[order(dfSutrop2014$Sutrop2014.,
decreasing=T),]  ##Organizar
                colnames(Exit)<-c("Itens", "Sutrop2014")

                #___________Monte Carlo
Sutrop2014_______________________________

                for(j in 1:300)
                    {
                        for(k in 1:ninf)
                            {
                                num.item<-sample(size.list, 1)
#Escolhe aleatoriamente quantas plantas o informante vai citar
                                listaparc<-sample(listsimul, size=num.item)
#Cria uma lista simulada com tamanho sorteado na linha anterior
                                matrlista[,k]<-c(listaparc,
rep(NA,times=(nintens-length(listaparc))))
                                matrank[,k]<-c(seq(1, length(listaparc)),
rep(NA,times=(nintens-length(listaparc))))
                            }
                        resA<-as.character(na.omit(as.vector(matrlista)))
                        resB<- as.numeric(na.omit(as.vector(matrank)))
                        resAB<-data.frame(resA, resB)
                        ttMC<-table(resA)
                        freqMC<-as.numeric(ttMC)
                        freqMC2<-freqMC^2
                        resAB.<-(tapply(resAB$resB, resAB$resA, sum))
                        resAB..<-as.numeric(resAB.)*ninf
                        resABC<-freqMC2/resAB..
                        resultMC[,j]<-c(resABC, rep(NA,times=(nintens-
length(resABC))))
                    }
            }

#############_____________________________________Cálculo do valor de
p_____________________________

        resultMC2<-(as.numeric(na.omit(as.vector(resultMC))))
#Transforma a matriz com os resultados das simulações em vetor
        Sal<-Exit[,2]                                          #Cria um objeto
```

```
com os valores de saliência para cada item
        Exit$p.valor<-rep(NA, times=length(Sal))
#Cria uma coluna no DF de saída para a inclusão do valor de p
            for (i in 1:length(Sal))                                #Compara
cada valor de saliência com o DF de saída
                {
                    if(Sal[i]>mean(resultMC2))                    #Verifica
se o valor de saliência é maior que a média
                        {
                            caudadir<-sum(resultMC2>=Sal[i])        #Se o
valor for maior que a média, é calculada a probalidade
                            p.D<-caudadir/length(resultMC2)        #dele ser
significativamente maior que os valores calculados
                            Exit$p.valor[i]<-p.D
                        }
                    else
                        {
                            caudaesq<-sum(resultMC2<=Sal[i])        #Se o
valor for menor que a média, é calculada a probalidade
                            p.E<-caudaesq/length(resultMC2)        #dele ser
significativamente menor que os valores calculados
                            Exit$p.valor[i]<-p.E
                        }
                }
        return(Exit)                                            #A função
retorna um DF com os itens, os valor se saliência
    }
```

HELP

```
salience                                    R Documentation
            Índice de Saliencia para Etnobiólogos e Antropólogos

Descrição

A função calcula Índices de Saliência uteis ou populares entre Etnobiólogos
e Antropólogos. A função retorna também a probabilidade de cada índice ter
sido gerado ao acaso, apresentando um p-valor.

Uso

salience(ll, indice="Smith93")

Argumentos

ll
    Um data.frame com duas variáveis categóricas: a identificação dos
```

```
entrevistados/informantes e os itens citados para quais pretende-se calcular
o índice.

indice
    O Índice de Saliência que deve ser calculado. Pode ser escolhido entre
"Smith93", "Smith95", "Smith97" e "Sutrop2014".


Detalhes

As formulas utilizadas para os cálculos de cada os índices são:
Smith (1993): (Σ(((Li − Rj)/Li)100))/N
Smith (1995): (Σ((Li − Rj)/(Li − 1)))/N
Smith (1997): (Σ((Li − Rj + 1)/Li))/N
Sutrop (2014): = F^2/(N Σ Rj)

Onde, para todas as fórmulas,
Li = Tamanho da lista em que o termo é citado,
Rj = Posição do item em uma dada lista Li,
N = Número total de listas (ou entrevistados/informantes),
F = Frequência de um item.
Obs.: No índice de Smith (1993) o primeiro item de uma lista tem Rj = 0.
Para todos os demais índices, o primeiro item possui Rj = 1.


Autor

Leonardo da Silva Chaves.

References

Smith, J. J. 1993. Using ANTHROPAC 3.5 and a spreadsheet to compute a free-
list salience
index. Cultural Anthropology Methods 5 (3): 1–3.

Smith, J. J., and S. P. Borgatti. 1997. Salience counts—and so does
accuracy: Correcting and
updating a measure for free-list-item salience. Journal of Linguistic
Anthropology 7 (2):
208–9.

Smith, J. J., L. Furbee, K. Maynard, S. Quick, and L. Ross. 1995. Salience
counts: A domain
analysis of English color terms. Journal of Linguistic Anthropology 5 (2):
203–16.

Sutrop, U. (2001). List task and a cognitive salience index. Field methods,
13(3), 263-276.

Exemplo
```

```
Entrevistados<-c(rep("Socorro", times=10), rep("Dan", times=15), rep("Timy",
times=12), rep("Guara",14))
Itens<-letters[1:20]
Lista<-c(sample(hh, 10), sample(hh, 15), sample(hh, 12), sample(hh, 14))

dfteste<-data.frame(Entrevistados, Itens)

salience(dfteste, indice="Sutrop2014")
```

ARQUIVOS DA FUNÇÃO

salience.r

help_salience.txt

Comentários sobre a função finalizada:

A função é bastante complexa e funciona corretamente, mas o objeto de entrada é pouco intuitivo, o help da função não está suficientemente claro, e o exemplo do help não gera o objeto de entrada corretamente, o que é um problema grave.

Além disso, no cálculo dos p-valores, não estão inclusos casos em que o mesmo ítem é citado mais de uma vez pela mesma pessoa (sample com replace), muito embora a função calcule os índices sem problemas para esses casos, e o índice de Sutrop2014 leve essa freqüência em consideração.

—-*Mali*