

Mathias M. Pires



Doutorado em Ecologia na USP. Título da tese: “Redes tróficas do Pleistoceno: estrutura e fragilidade”. Orientado por Paulo Roberto Guimarães Jr.

Meus Exercícios

[mathias_01_f](#)

[mathias_02_f](#)

[mathias_03_f](#)

[mathias_04_f](#)

[mathias_05_f](#)

[mathias_06_f](#)

[mathias_07_f](#)

[mathias_08_f](#)

[mathias_09_f](#)

Proposta de Trabalho Final

Principal

O padrão encaixado ou aninhado (“nestedness”) está entre os padrões mais reportados para a distribuição de espécies em metacomunidades e para a redes de interações entre espécies. Entretanto as métricas que quantificam o grau de aninhamento de matrizes de ocorrência ou interações foram pensadas para matrizes binárias. O objetivo é gerar uma função capaz de calcular o grau de aninhamento em matrizes quantitativas. Essa função lê matrizes quantitativas em um diretório, calcula o índice de interesse e calcula sua significância comparando o valor empírico com uma distribuição do mesmo índice gerada a partir de matrizes simuladas por modelos teóricos.

Comentário

O plano principal é interessante e desafiador. A princípio não disponibilize nenhuma alternativa para o usuário. Isso facilita seu trabalho, já que a função é ambiciosa. Seria interessante em um outro momento incluir opções em argumentos para, por exemplo, diferentes cenários nulos (diferentes simulações para as matrizes). Boa sorte!

— [Alexandre Adalardo de Oliveira](#) 2010/03/31 11:51

Plano B

As interações entre os elementos de um sistema, como entre as espécies de uma comunidade, pode se representada por uma matriz de adjacência A onde os elementos são representados por linhas e colunas. Cada célula a_{ij} dessa matriz é preenchida por 1 se o elemento i interage com j e por 0 em caso contrário. Existe uma grande diversidade de medidas e índices que permitem descrever a estrutura dessas matrizes e, portanto, a estrutura do sistema. No entanto, é necessário avaliar se essa estrutura é de fato gerada pelos processos de interesse ou se poderia surgir devido somente ao acaso. Nesse sentido meu objetivo é definir uma função capaz de gerar matrizes teóricas a partir dos parametros (i.e, número de elementos e de interações) de uma matriz empírica. A função gera o número de matrizes determinada pelo usuário e de acordo com o modelo selecionado. São quatro modelos distintos: 1)interações equiprováveis; 2)interações equiprováveis nas linhas; 3)interações equiprováveis nas colunas; 4)probabilidade de interações dependentes o número de interações original. Com um conjunto de matrizes teóricas é possível então gerar uma distribuição de valores de qualquer estatística de interesse e comparar com o valor empírico.

Página de Ajuda

ANODF

package:none

R Documentation

ANODF: Aninhamento pra matrizes quantitativas

Description:

Calcula o grau de aninhamento de uma matriz quantitativa (quadrada ou não) e retorna o valor da estatística ANODF (Abundance Nestedness metric based on overlap and decreasing fill), os valores médios de ANODF calculados a partir de uma coleção de matrizes teóricas e o intervalo de confiança ao nível de significância de 5%.

Usage:

```
ANODF(mat=NULL, nam="matrix.txt", read=TRUE, sort=TRUE, nsim=1000,
nullmodel=TRUE, write.r=TRUE)
```

Arguments:

mat	objeto que armazena matriz já definida a ser analisada (deve ser usada com read=FALSE; ler sobre argumento read abaixo)
nam	nome de um arquivo txt contendo a matriz de interesse
read	se verdadeiro le matriz indicada por argumento nam; se falso,

função considera a matriz definida no argumento `mat`.

`sort` se verdadeiro ordena matriz de entrada para maximizar aninhamento. Só usar `sort=FALSE` caso haja uma razão para usar a matriz no formato original

`nsim` número de matrizes teóricas geradas

`nullmodel` gera matrizes teóricas para teste de significância de ANODF. (detalhes dos modelos Details)

`write.r` se verdadeiro gera arquivo txt contendo resultados. Cada linha é o resultado de uma análise

Details:

A métrica ANODF permite avaliar o grau de aninhamento em uma matriz quantitativa. Para isso são verificadas todos os pares de linhas e colunas na matriz. Para cada par de linhas li e lj onde $j > i$ é calculada a porcentagem de células em lj cujo valor é menor que o das células em li na mesmas posições de colunas. Somente os valores maiores que zero são considerados. A somatória dessas porcentagens multiplicada por 100 é o grau de aninhamento das linhas. O procedimento é o mesmo para colunas. O aninhamento total é dado então por:

$$ANODF = 2 * (ANODF(linhas) + ANODF(colunas)) / (m * (m - 1) + n * (n - 1)),$$

onde m é número de linhas e n o número de colunas na matriz. ANODF vai de 0, não aninhada, até 100, completamente aninhada.

O primeiro modelo teórico incluído na função corresponde a uma aleatorização de cada registro presente na matriz, mantendo portanto o número de registros.

O segundo modelo corresponde a uma aleatorização da posição dos valores entre as células da matriz portanto mantém as relações de grandeza entre as observações.

Value:

Se `nullmodel=FALSE`, retorna somente o valor empírico

Se `nullmodel=TRUE`, retorna uma lista de valores

`comp1` : valor empírico

`comp2` : intervalo de confiança (5%) para modelo 1

`comp3` : média para modelo 1

`comp4` : média para modelo 2

`comp5` : intervalo de confiança (5%) para modelo 2

Warning:

Se não é definida uma matriz usando o argumento `mat` e o argumento `read` é determinado como falso função não retorna nenhum valor é é retornada mensagem de erro.

Note:

Se não for especificado o nome da matriz procurada no diretório é `matriz.txt`. Caso deseje analisar matrizes com outros nomes use o argumento `nam` ou siga as orientações no parágrafo abaixo.

A função pode rodar com `dataframe` (desde que esteja no formato de uma matriz) ou matrizes pré-definidas como um objeto que não estão no diretório. Nesse caso o nome do objeto deve ser especificado no argumento `mat` e o argumento `read` como `FALSE`. Caso `read` não seja definido a matriz usada será lida do diretório.

Se argumento `write.r=TRUE`, também gera um arquivo `txt` aonde são impressos os resultados de cada análise em sequência no seguinte formato:
nome da matriz: valor empírico; média para modelo 1, intervalo de confiança (5%) para modelo 1; média para modelo 2, intervalo de confiança (5%) para modelo 2;

Author(s):

Mathias Mistretta Pires - Programa de pós-graduação em ecologia, Universidade de São Paulo.

References:

Almeida-Neto M., Guimaraes, P., Guimaraes, P.R., Loyola, R.D. and Ulrich, W. (2008). A consistent metric for nestedness analysis in ecological systems: reconciling concept and measurement. *Oikos*,117,1227-1239.

Examples:

```
vec.1=c(5:1)
vec.2=c(4:0)
vec.3=c(3:0,0)
vec.4=c(2:0,rep(0,2))
vec.5=c(1,rep(0,4))
mat=data.frame(vec.1,vec.2,vec.3,vec.4,vec.5)
ANODF(mat=mat,read=false)
#Essa é uma matriz completamente aninhada e, portanto ANODF deve ser igual a 100.
```

```
vec.1=rep(5,5)
vec.2=c(rep(4,4),0)
vec.3=c(rep(3,3),0,0)
```

```
vec.4=c(rep(2,2),0,0,0)
vec.5=c(1,rep(0,4))
nested.c=data.frame(vec.1,vec.2,vec.3,vec.4,vec.5)
ANODF(mat=nested.c,read=false)
#Essa é uma matriz com colunas completamente aninhadas, sem aninhamento nas
linhas e, portanto ANODF deve ser igual a 50.
```

Código da Função

```
ANODF=function(mat=NULL, nam="matrix.txt", read=TRUE, sort=TRUE, nsim=1000,
nullmodel=TRUE, write.r=TRUE)
{
  #puxa matriz
  if (read==TRUE)
  {
    mat=read.table(nam,sep="\t",header=FALSE)  #Lê matrix no diretório
  }
  mat=as.matrix(mat)
  dim=dim(mat)
  m=dim[1]      #numero de linhas
  n=dim[2]      #número de colunas

  #ordenação da matriz a partir dos totais marginais (default)
  if (sort==TRUE)
  {
    rmarg=apply(mat,1,sum)
    cmarg=apply(mat,2,sum)
    rmarg[m+1]=NA
    mat=rbind(mat,cmarg)
    mat=cbind(mat,rmarg)
    mat=mat[,order(mat[m+1,],decreasing=TRUE)]
    mat=mat[order(mat[,n+1],decreasing=TRUE),]
    mat=mat[-(m+1),]
    mat=mat[,-(n+1)]
  }
  #Cálculo do ANODF
  Nc= (n*(n-1))/2      #numero de comparações par a par entre colunas
  Nr= (m*(m-1))/2      #número de comparações par a par entre linhas
  kc=matrix(rep(NA,Nc),nrow=m,ncol=Nc)      #matriz que armazena resultado
de teste lógico das colunas
  kr=matrix(rep(NA,Nr),nrow=Nr,ncol=n)      #matriz que armazena resultado
de teste lógico das linhas
  mat[which(mat==0)]=NA      #substitui 0 por NA para facilitar cálculos

  #Por colunas
  #Qual o número de células com valores menores em colunas à esquerda do que
à direita?
  u=1
```

```
for (i in 1:(n-1))
{
  for (l in (i+1):n)
  {
    kc[,u]=mat[,l]<mat[,i]
    u=u+1

  }
}
#número de células comparadas
nij=matrix(NA,nrow=m,ncol=Nc)
u=1
for (i in 1:(n-1))
{
  for (l in (i+1):n)
  {
    nij[,u]=mat[,i]+mat[,l]
    u=u+1

  }
}
n.c=apply(nij>0,2,sum,na.rm=TRUE)      #número de células comparadas em cada
par de linhas
soma.c=apply(kc,2,sum,na.rm=TRUE)      #número de células na coluna a
direita com valores menores que os da coluna a esquerda
prop.c=soma.c/n.c                      #porcentagem de valores menores dado
o total de comparações
anodfc=100*sum(prop.c,na.rm=TRUE)      #aninhamento nas colunas

#Por linhas
#Qual o número de células com valores menores nas linhas abaixo do que nas
linhas acima?
u=1
for (i in 1:(m-1))
{
  for (l in (i+1):m)
  {
    kr[u,]=mat[l,]<mat[i,]
    u=u+1
  }
}

#número de células comparadas
nji=matrix(NA,nrow=Nr,ncol=n)
u=1
for (i in 1:(m-1))
{
  for (l in (i+1):m)
  {
    nji[u,]=mat[i,]+mat[l,]
```

```

    u=u+1
  }
}

n.r=apply(nji>0,1,sum, na.rm=TRUE)      #numero de células comparadas entre
pares de linhas (cada posição um par)
soma.r=apply(kr,1,sum,na.rm=TRUE)
prop.r=soma.r/n.r                      #proporção dos valores comparados
que são menores nas linhas mais abaixo
anodfr=100*sum(prop.r,na.rm=TRUE)      #aninhamento nas linhas

anodft=(2*(anodfc+anodfr))/((m*(m-1))+(n*(n-1)))  #calcula métrica ANODF

if (nullmodel==FALSE & write.r==TRUE)      #quando usuário não deseja
rodar modelos teóricos retorna resultado
{
  write.table (paste(nam,":","ANODF =",anodft),"result.txt",
append=TRUE, col.names=FALSE,row.names=FALSE)
  return (anodft)
}
if (nullmodel==FALSE & write.r==FALSE)
{
  return (anodft)
}
if (nullmodel==TRUE)
{
  anodft.emp=anodft                      #armazena o valor empírico
  #=====
  #Modelos teóricos
  #=====
  #nullmodel1: aleatorização dos registros
  #=====
  mat.bin=mat
  mat.bin[which(mat>0)]=1
  C=sum(mat.bin,na.rm=TRUE)/(m*n)
  N=sum(mat, na.rm=TRUE)
  mod.l=rep(NA,nsim)
  for(l in 1:nsim)
  {
    y=1
    mat.temp=matrix(0,nrow=m,ncol=n)
    while (y<=N)
    {
      i=round(runif(1,1,m))
      j=round(runif(1,1,n))
      p=runif(1,0,1)
      if (p<C)
      {
        mat.temp[i,j]= mat.temp[i,j]+1
        y=y+1
      }
    }
  }
}

```

```
    }
  }
  #calcula ANODF para matrizes teoricas
  mod.1[l]=ANODF (mat=mat.temp, read=FALSE,nsim=1, nullmodel=FALSE,
write.r=FALSE)
}
mean.1=mean(mod.1)
quant.1=quantile(mod.1,c(0.025,0.975))
#=====
#nullmodel2: aleatorização dos valores
#=====
mat2=as.numeric(mat)
mat2=mat2[is.na(mat2)==FALSE]
lg=length(mat2)
mod.2=rep(NA,nsim)
for(u in 1:nsim)
{
  mat3=mat2
  mat.temp2=matrix(NA,nrow=m,ncol=n)
  y=0
  while (y<lg)
  {
    i=round(runif(1,1,m))
    j=round(runif(1,1,n))
    if (is.na(mat.temp2[i,j])==TRUE)
    {
      v=round(runif(1,1,(lg-y)))      #sorteia um valor da matriz
original
      p=runif(1,0,1)
      if (p<C)
      {
        mat.temp2[i,j]= mat3[v]
        mat3=mat3[-v]                #Exclui valor já atribuido
        y=y+1
      }
    }
  }
  #calcula ANODF para matrizes teoricas
  mod.2[u]=ANODF
(mat=mat.temp2,read=FALSE,nsim=1,nullmodel=FALSE,write.r=FALSE)
}
mean.2=mean(mod.2)
quant.2=quantile(mod.2,c(0.025,0.975))

results=list(anodft.emp,mean.1,quant.1, mean.2, quant.2)
names(results)=c("ANODF","Avg null 1", "IC null 1 (5%)","Avg null 2",
"IC null 2 (5%)")
if (write.r==FALSE)
{
  return(results)
}
```

```
}  
if (write.r==TRUE)  
{  
  write.table(paste(nam,":","ANODF =",round(anodft.emp,2),";","Avg_null  
1_ANODF =", round(mean.1,2),",", "IC =", round(quant.1[1],2),"--",  
round(quant.1[2],2),";","Avg_null 2_ANODF =", round(mean.2,2),",", "IC =",  
round(quant.2[1],2),"--", round(quant.2[2],2)), "result.txt",  
sep="\t",col.names=FALSE,row.names=FALSE, append=TRUE)  
  return(results)  
}  
}
```

Arquivo da Função

ANODF

From:

<http://ecor.ib.usp.br/> - **ecoR**

Permanent link:

http://ecor.ib.usp.br/doku.php?id=05_curso_antigo:r2010:alunos:trabalho_final:mathiasmpires:start



Last update: **2020/08/12 06:04**