Flávia Maria Darcie Marquitti

Mestranda em Ecologia, UNICAMP. Trabalho com redes complexas de interações mutualísticas

Meus Exercícios

Link: exercicios_flavia

Proposta de Trabalho Final

Principal

Trabalho com redes de interação mutualística entre espécies. Redes de interação podem ser representadas por matrizes binárias (A), em que as espécies se distribuem nas linhas e nas colunas e preencho o elemento aij dessa matriz com 1 caso a espécie da linha i interaja com a espécie da coluna j. Preencho aij com 0 caso não ocorra interação entre as espécies da linha i e coluna j.

Em redes de interação entre espécies ou mesmo em redes sociais, existe um centro e uma periferia. O centro é composto por espécies (ou pessoas) que estabelecem muitas interações com outras espécies. Existem algumas métricas em redes complexas que estabelecem valores contínuos de centralidade. Em meu trabalho de mestrado utilizo uma métrica de centralidade para redes bipartidas do programa Ucinet 6.0 que define que nós pertencem ao centro e que nós pertencem à periferia. No entanto, esse programa não esclarece como funciona o algoritmo dessa métrica, calculando quem é centro quem é periferia por métodos de otimização sem oferecer um modelo nulo

A minha idéia é passar para a plataforma R um programa que pensei para resolver este meu problema:

O meu programa ordena a matriz para o aninhamento máximo. O programa faz uma divisão entre as linhas e uma divisão entre as colunas, nessa ordem, dividindo a matriz em quatro sub-matrizes. A primeira sub-matriz acima à esquerda é composta pelas espécies centrais. A última sub-matriz abaixo à direita é composta pelas espécies periféricas. O programa fará todas as combinações possíveis de cortes nas linhas e nas colunas. A divisão selecionada será aquela cuja diferença entre as conectâncias da primeira e da última sub-matrizes for máxima. O programa dará como saída a lista de espécies centrais e periféricas.

Ainda há um problema a ser resolvido (e estou pensando nele com carinho) que é como decidir qual corte escolher (entre cortes de diferentes combinações) quando houver empate das diferenças de conectâncias.

Comentários PI

Ótima! Vc tem clareza do que quer e como fazer. Manda ver!

Plano B

Eu moro em uma república com 10 meninas ao todo. Quando a conta de telefone chega é a maior trabalheira pra ver quem ligou pra quem. E a nossa prestadora de serviço telefônico oferece a conta 5 dias antes do vencimento num formato bem amigável!

A minha idéia é ter uma lista com os principais números que cada pessoa liga associados aos nomes das moradoras. Com isso, no dia que a conta estiver disposta on line, entro com ela nesse programa e ela faz automaticamente as contas para cada moradora. Ainda, soma as chamadas perdidas (sem dona) e divide por 10. A saída será em forma de uma tabela em txt que fornece quanto cada uma gastou com suas chamadas, quanto cada uma gastou com perdidas e o total por pessoa. Abaixo dessa tabela estarão quais são os números das chamadas perdidas.

PS: se não pudermos fazer programas que não tenham a ver com Biologia, por favor me avisem que eu penso em um outro plano B

Comentários PI

Pode fazer sobre assuntos sim, mas prefira o biológico.

Ajuda da função

```
core.peri                package:none                R Documentation

             Centro e periferia de redes binárias aninhadas bipartidas



Description:

Retorna quais os rótulos das linhas e das colunas de redes binárias
aninhadas bipartidas compõem o centro e a periferia da rede de entrada.

Usage:

core.peri(name.file, names.spp, read = F, sep.table, write.it = F)

Arguments:

name.file   nome da entrada
names.spp   lógico. Se TRUE, a entrada possui rótulos nas linhas e colunas.
Se FALSE, o programa nomeia linhas e colunas conforme delas na entrada
read        lógico. Se TRUE, lê um arquivo de entrada (usar aspas). Se FALSE
(default), lê uma variável (sem aspas)
```

```
sep.table   o caracter separador de campo do arquivo de entrada. Usar aspas.
Não usar se read = FALSE
write.it    gera um arquivo txt para a saída



Details:

A matriz de entrada é ordenada para a forma aninhada e então é feito um
corte entre as linhas e um corte entre as colunas, dividindo a matriz de
entrada em quatro submatrizes.

São feitos todas as combinações de cortes de linhas e colunas possíveis
(i.e., (i-1)*(j-1) combinações, onde i é o número de linhas da matriz de
entrada e j é o número de colunas da matriz de entrada).

A primeira submatriz é aquela à esquerda na parte superior e a quarta é
aquela à direita no canto inferior.

A função calcula a conectância (C) nessas duas submatrizes através da
formula:

    C = E/(m*n),    onde E é o número de 1's, m é o número de linhas e n é o
número de colunas (Jordano 1987).

A combinação de cortes de linha e coluna é escolhido conforme:
    1) maior diferença de conectâncias entre a primeira e quarta submatrizes
    2) dentre as combinações com a maior diferença em (1), é escolhida a
combinação que possui maior proporção de linhas e colunas na primeira
submatriz.



Value:

É impresso na tela uma matriz em que na primeira coluna estão os rótulos das
linhas e colunas que pertencem ao centro. Na segunda coluna, estão os
rótulos das linhas e colunas que pertencem à periferia.

Se names.spp = FALSE, os rótulos das linhas começam com a letra R e das
colunas com a letra C. O número em seguida à letra se refe à ordem que se
encontrava na matriz de entrada.

Se write.it = TRUE e read = TRUE, é gerado um arquivo de saída  com o nome
"out_name.file.txt". Se write.it = TRUE e read = FALSE, é gerado um arquivo
com o nome "out.txt"



Warning:



Note:
```

```
A função não é executada para matrizes de entrada com valores faltantes ou
não binários.


Author(s):

Flávia Maria Darcie Marquitti - Programa de pós-graduação em ecologia,
Universidade Estadual de Campinas

References:

Jordano, P. 1987. Patterns of mutualistic interactions in pollination and
seed dispersal: connectance, dependence asymmetries, and coevolution.
American Naturalist 129: 657-677.

Examples:

test1 =
matrix(c(1,0,1,0,1,0,0,0,0,1,1,1,1,1,1,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,1,0,0
,0),7,5)
core.peri(test1, names.spp = F, read = F, write.it = F)



test2 =
matrix(c(1,0,1,0,1,0,0,0,0,1,1,1,1,1,1,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,1,0,0
,0),7,5)
rownames(test2) = (c("a","b","c","d","e","f","g"))
colnames(test2) = c("A","B","C","D", "E")
core.peri(test2, names.spp = T, read = F, write.it = F)

test =
matrix(c(1,0,1,0,1,0,0,0,0,1,1,1,1,1,1,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,1,0,0
,0),7,5)
write.table(test, sep = "\t", row.names = F, col.names =F,
file=paste(getwd(),"/test3.txt", sep=""))
core.peri("test3.txt", names.spp = F, read = T, sep = "\t", write.it = T)

test =
matrix(c(1,0,1,0,1,0,0,0,0,1,1,1,1,1,1,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,1,0,0
,0),7,5)
rownames(test) = (c("a","b","c","d","e","f","g"))
colnames(test) = c("A","B","C","D", "E")
write.table(test, sep = "\t", row.names = T, col.names = T,
file=paste(getwd(),"/test4.txt", sep=""))
core.peri("test4.txt", names.spp = T, read = T, sep = "\t", write.it = T)
```

Código da função

```
nest = function(net){ ## Ordena a matriz para a forma aninhada
```

```
  row.degree = apply(net,1,sum) # faz a soma dos totais marginais das linhas
(grau das linhas)
  col.degree = apply(net,2,sum) # faz a soma dos totais marginais das
colunas (grau das colunas)
  net.nest = net[order(row.degree, decreasing=TRUE), order(col.degree,
decreasing=TRUE)] # ordena a matriz, pelos totais marginais das linhas e
colunas de forma decrescente
}


conect.prop = function(net.nest, name.file){ ## Faz cortes na rede, calcula
a conectancia da 1a. e 4a. submatrizes, calcula a proporção de spp na 1a.
submatriz
  m = 1
  Ci = matrix(rep(NA),nrow((net.nest)-1)*(ncol(net.nest)-1),6) # cria uma
matriz que guardará os cortes da linhas e colunas, a conectância na primeira
e
                                                                  # quarta
subamtrizes, proporção de linhas e colunas na primeira subamtriz e a
diferença entre as conectâncias
  for(j in 1:(ncol(net.nest)-1)){
    for(i in 1:(nrow(net.nest)-1)){
      sub.1 = net.nest[1:i,1:j] # define a primeira submatriz da rede de
entrada já aninhada
      sub.4 = net.nest[(i+1):nrow(net.nest),(j+1):ncol(net.nest)] # define a
quarta submatriz da rede de entrada já aninhada
      Ci[m,1] = i # guarda o corte da linha
      Ci[m,2] = j # guarda o corte da coluna
      Ci[m,3] = sum(apply(as.matrix(sub.1),1,sum))/(i*j) # guarda a
conectância da primeira submatriz
      Ci[m,4] = sum(apply(as.matrix(sub.4),1,sum))/((nrow(net.nest)-
i)*(ncol(net.nest)-j)) # guarda a conectância da quarta submatriz
      Ci[m,5] = (i*j)/(ncol(net.nest)*nrow(net.nest)) # guarda a proporção
de linhas e colunas na primeira submatriz
      m = m+1
    }
  }
  Ci[,6]= Ci[,3] - Ci[,4] # Calcula diferença de conectâncias da primeira e
quarta submatrizes
  colnames(Ci) = c("corte.linha", "corte.coluna", "Conect1a. (C)",
"Conect4a. (c)", "prop.sub1", "C-c")
  Ci.res = Ci[Ci[,6]==max(Ci[,6],na.rm=T), ] # Separa os resultados dos
cortes com maiores diferenças de conectâncias e define o corte com maior
proporção de linhas e colunas dentre os que
                                                 # possuem maior diferença de
conectâncias
  Ci.res = Ci.res[sort.list(Ci.res[,5], decreasing=TRUE), ]
  row.cut = Ci.res [1,1]
  col.cut = Ci.res [1,2]
  core = c(colnames(net.nest[,1:col.cut]), rownames(net.nest[1:row.cut,])) #
define os rótulos da entrada que pertencem ao centro da rede
```

```
  periphery = c(colnames(net.nest[,(col.cut+1):ncol(net.nest)]),
rownames(net.nest[(row.cut+1):nrow(net.nest),])) # define os rótulos da
entrada que pertencem à periferia da rede
  out.put = matrix ( , max(length(core),length(periphery)),2) # faz uma
matriz de saída em que os rótulos das linahs e colunas centrais ficam na
primeira coluna e os rótulos das linahs e
                                                                # colunas
periféricas ficam na segunda coluna
  out.put[1:length(core),1] = core
  out.put[1:length(periphery),2] = periphery
  colnames(out.put) = c("core","periphery")
  return(out.put)
}



core.peri = function(name.file, names.spp, read = F, sep.table, write.it =
F){ ## Define as linhas e colunas centrais e periféricas de redes aninhadas
  if(names.spp == TRUE){
    if(read == TRUE){
      net.in = read.table(name.file, sep = sep.table, header = T, row.names
= 1) # leitura da rede de entrada que já possui rótulos nas linhas e colunas
    }
    else{
      net.in = name.file # leitura da rede de entrada que já possui rótulos
nas linhas e colunas
    }
    for(i in 1:nrow(net.in)){
      for(j in 1:ncol(net.in)){
        if(is.na(net.in[i,j])){ # verifica se a rede possui valores
faltantes
          print("The input file has missing values")
          return()
        }
        if(net.in[i,j] != 1 & net.in[i,j] != 0){ # verifica se a rede é
binária
        print("The input file is not a binary matrix")
        return()
        }
      }
    }
    net.nest = nest(net.in) # ordena a rede de entrada para forma aninhada,
chamando a função "nest"
    out = conect.prop(net.nest, name.file)   # faz os cortes da rede para
definir centro e periferia e gerar o arquivo de saída, chamando a função
"conect.prop"
  }

  if(names.spp == FALSE){
    if(read == TRUE){
      net.in = read.table(name.file, sep = sep.table, header = F) # leitura
```

```
da rede sem rótulos
    }
    else {
      net.in = name.file # leitura da rede sem rótulos
    }
    colnames(net.in) = paste("C",1:ncol(net.in), sep="") # nomeação dos
rótulos das colunas conforme a ordem da entrada
    rownames(net.in) = paste("R", 1:nrow(net.in), sep="") #  nomeação dos
rótulos das linhas conforme a ordem da entrada
    for(i in 1:nrow(net.in)){
      for(j in 1:ncol(net.in)){
        if(is.na(net.in[i,j])){ # verifica se a rede possui valores
faltantes
          print("The input file has missing values")
          return()
        }
        if(net.in[i,j] != 1 & net.in[i,j] != 0){ # verifica se a rede é
binária
        print("The input file is not a binary matrix")
        return()
        }
      }
    }
    net.nest = nest(net.in) # ordena a rede de entrada para forma aninhada,
chamando a função "nest"
    out = conect.prop (net.nest, name.file)  # faz os cortes da rede para
definir centro e periferia e gerar o arquivo de saída, chamando a função
"conect.prop"
  }
  if (read == T & write.it == T){
    write.table(out, row.names = F, sep = "\t", file=paste(getwd(),"/out_",
name.file, sep="")) # imprime a saída
  }
  if (read == F & write.it == T){
    write.table(out, row.names = F, sep = "\t",
file=paste(getwd(),"/out.txt", sep="")) # imprime a saída
  }
  return(out)
}
```

Arquivo da função

- Trabalho final - Flávia Marquitti