Lucas Paoliello de Medeiros

Sou aluno de graduação em Ciências Biológicas do IB-USP. Estou desenvolvendo um projeto de iniciação científica sob a orientação da pós-doutoranda Esther Sebastián González no Departamento de Ecologia do IB. O objetivo é comparar a estrutura modular de diversas redes de mutualismos de plantas e animais, procurando entender como a força do mutualismo pode afetar essa estrutura. O conhecimento do R será muito útil para a análise e interpretação dos dados.

Meus Exercícios

exec

Minha Proposta

Plano A

A modularidade é um padrão que aparece em diversas redes de interações mutualísticas entre plantas e animais (Olesen et al., 2007, Donatti et al., 2011). Módulos são agrupamentos dentro da rede, em que espécies interagem mais entre si do que com espécies de outros módulos. Uma rede de interações é representada por meio de uma matriz de adjacência, cujos elementos podem ser 0 ou 1, sendo que as colunas são espécies de plantas e as linhas, de animais. O grau de modularidade da rede, assim como o número de módulos e suas composições, podem ser obtidos por meio de um algoritmo chamado simulated annealing (SA). O programa que usei para fazer essa análise foi o programa MODULAR (Marquitti et al., 2012), que me forneceu, para diversas redes, o número de módulos, o grau de modularidade e duas tabelas, uma contendo uma lista das interações da rede e outra que mostra a qual módulo cada espécie pertence. Podemos atribuir um papel ecológico para cada espécie da rede com base em dois parâmetros: Z, que define a sua posição em relação à outras espécies do módulo e C, que mede quão bem ela se conecta com espécies de outros módulos (Olesen et al., 2007). Dessa forma as espécies podem ser classificadas em periféricas (baixo Z e C), conectoras entre módulos (baixo Z e alto C), conectoras dentro de módulos (“module hub”) (alto Z e baixo C) e super-generalistas (“network hub”) (alto Z e C).

Zi = Kis – Ks / SDks , onde i é uma espécie, s é o número do módulo, Kis é o número de interações no mesmo módulo, Ks é a média do número de interações das espécies dentro do módulo e SDks é o desvio padrão.

Ci = 1 – Σ (Kit / Ki)^2 , onde a somatória percorre todos os módulos (de t a n), Kit é o número de interações da espécie i para o módulo t, incluindo o módulo de i e Ki é o número de interações da espécie i.

Pretendo criar uma função no R que me forneça os valores de Z e C para cada espécie da rede,

utilizando as tabelas de resultado geradas pelo programa MODULAR. A função pode também plotar em um gráfico esses valores para se ter ideia da proporção dos diferentes papéis ecológicos dentro da rede.

Referências:

Donatti, C I; Guimarães, P R; Galetti, M; Pizzo M A; Marquitti, F M D & Dirzo, R. 2011. Analysis of a hyper-diverse seed dispersal network: modularity and underlying mechanisms. *Ecology Letters*, 14, 773-781

Marquitti, F M D; Guimarães, P R; Pires, M M and Bittencourt, L F. 2012. MODULAR: Software for Autonomous Computation of Modularity for Large Network Sets. *PLOS Computational Biology*

Oleson, J M; Bascompte, J; Dupont, Y L; Jordano, P. 2007. The modularity of pollination networks. *PNAS*, 104, 50, 19891-19896

Plano B

Minha segunda opção é elaborar uma função no R que calcule o número de pacotes de figurinhas que devemos comprar para se ter uma alta probabilidade (95% de chance) de completar o álbum. Para isto, criarei um modelo simplificado, cujos parâmetros são: número total de espaços no álbum (ex. 600), número de figurinhas por pacote (ex. 3). Para simplificar, farei com que a chance de obter qualquer figurinha é a mesma e não é possível trocar as repetidas com coleguinhas. Poderei criar simulações de álbuns sendo preenchidos para obter uma curva de probabilidades ou trabalhar com equações de análise combinatória.

O modelo pode até ter alguma utilidade ecológica. Podemos imaginar que os espaços do álbum são manchas de habitat e as figurinhas dos pacotes são novas espécies que chegam por migração, por exemplo, como em um cenário de metapopulação, mas com espécies ao invés de populações. Completar o álbum nesse caso, pode significar atingir o máximo de riqueza de espécies nessa área.

Comentários

Oi Lucas, o plano A parece mais promissor e acredito que também irá ajudá-lo na sua pesquisa. Acho que do jeito que está proposto a construção do plano A parece simples. No entanto, dá para complicar um pouco pensando que vc pode utilizar várias redes e, assim, terá várias matrizes. Então (depois de conversar com os outros monitores) uma sugestão seria vc construir sua função permitindo que a análise seja feita com várias redes e não apenas com uma. Vc pegaria todos os outputs do Modular de todas as redes ao mesmo tempo para usar como input na sua função. E o output da função poderia ser um data frame com as métricas que vc propõe calculadas para cada uma das redes. Hamanda

Função zc.modules

Help da função

```
zc.modules                                          package: nenhum
R Documentation


Calcula os valores de z e c de espécies de uma rede.


Description:

        A função zc.modules utiliza duas tabelas, uma com as interações de
uma rede e outra com o módulo de cada espécie, para calcular os valores de z
(within-module degree) e c (among-module connectivity) para cada espécie. A
função retorna uma tabela que contém  os módulos da cada par de espécies que
interagem e outra com os valores de z e c de cada espécie.


Usage:

        zc.modules(interactions, members)


Arguments:

        interactions         data frame.    Um data frame de duas colunas,
que contenha todos os pares de interação de uma rede. A primeira coluna deve
conter termos Ri (onde i = 1, 2, 3, …, m) representando as espécies das
linhas de uma matriz de adjacência com m linhas. A segunda coluna deve
conter termos Cj (onde j = 1, 2, 3, …, n)  representando as espécies das
colunas de uma matriz de adjacência com n colunas.

        members             data.frame.    Um data frame de duas colunas,
que contenha todas as espécies Ri e Cj na primeira coluna e o módulo de cada
uma delas na segunda coluna.


Details:

        A função zc.modules foi criada para calcular os valores de z e c das
espécies de uma rede à partir dos dados de saída do programa MODULAR, o qual
gera as tabelas interactions e members necessárias como argumentos da
função.
        É necessário que os argumentos da função sejam data frames com os
termos Ri e Cj, pois a função opera utilizando esses termos como fatores.
        A fórmula para o cálculo do z é Zi = Kis − Ks / SDks , onde i é uma
espécie, s é o número do seu módulo, Kis é o número de interações intra-
módulo dessa espécie, Ks é a média do número de interações intra-módulo das
espécies no módulo e SDks é o desvio padrão.
        A fórmula para o cáculo do c é Ci = 1 − Σ(Kit / Ki)^2 , onde a
```

```
somatória percorre todos os módulos (de t a n), Kit é o número de interações
da espécie i para o módulo t, incluindo o módulo de i e Ki é o número de
interações da espécie i.
        No cálculo do z, o denominador da fração (SDks) pode ser zero, nesse
caso, não há valor de z para essa espécie e há um NaN na tabela de
resultados.


Value:

        A função retorna uma lista com duas posições.

    comp1: Um data frame contendo quatro colunas. As duas primeiras com as
interações entre as espécies Ri (primeira coluna) e Cj (segunda coluna) como
na tabela interactions. As outras duas contém o módulo das espécies Ri
(terceira coluna) e das espécies Cj (quarta coluna).
    comp2: Um data frame de quatro colunas. As duas primeiras são idênticas
à tabela members e as outras duas contém os valores de z (terceira coluna) e
c (quarta coluna) para cada espécie.


Author:
        Lucas Paoliello de Medeiros
        lucaspdmedeiros@gmail.com


References:

        Olesen, J M; Bascompte, J; Dupont, Y L; Jordano, P. 2007. The
modularity of pollination networks. PNAS, 104, 50, 19891-19896


Examples:

        Baixe os arquivos "med2.txt" e "members_med2.txt.txt" e salve-os no
seu diretório de trabalho. Crie dois data frames no R com estes arquivos da
seguinte forma:

        interactions = read.table("med2.txt", sep=" ")

        members = read.table("members_med2.txt.txt", sep="\t")

        Utilize a função com esses dois data frames como argumentos.
```

Código da função

```
zc.modules = function(interactions, members)
{
```

```
#########################################################################
#############
  ##### Criando um data.frame que contem uma lista das interacoes das
especies da rede ####
  ##### e o modulo em que se encontram.
#################################################
#########################################################################
#############
  colnames(interactions)=c("spp.row","spp.col")
  # Ordenando a tabela, isso sera necessario quando fizermos o ciclo
  interactions = interactions[order(interactions$spp.col),]
  colnames(members)=c("spp", "module")
  # Ordenamos a outra tabela
  members = members[order(members$spp),]
  n.col = length(interactions$spp.col)  # Comprimento da tabela
  results.col = rep(NA, n.col) # Vetor de NAs para colocarmos os resultados
do ciclo
  for (i in 1 : n.col)
  {
    # O modulo da especie i da tabela members. Este i eh o primeiro level do
fator spp
    module.col = members$module[members$spp[i]]
    # Primeiro fazemos um teste logico com a tabela interactions, que coloca
TRUE
    # nas posicoes que contem o level i do fator spp.col. Depois pegamos
somente as posicoes
    # em que temos TRUE (com which) e colocamos o modulo dessa especie
nessas posicoes.
results.col[(which((interactions$spp.col)==levels(interactions$spp.col)[i]))
] = module.col
  }
  # Agora fazemos a mesma coisa com o fator spp.row, que contem os levels Ri
  n.row = length(interactions$spp.row)
  results.row = rep(NA, n.row)
  for (i in 1: n.row)
  {
    # Para fazer essa indexacao somamos o comprimento dos levels da tabela
interactions
    # a i. Como a tabela members esta ordenada, comecamos a atribuir o
modulo a module.row
    # a partir do primeiro Ri.
    module.row = members$module[members$spp[i +
length(levels(interactions$spp.col))]]
    # Aqui criamos o vetor module.row da mesma forma que criamos results.col
results.row[(which((interactions$spp.row)==levels(interactions$spp.row)[i]))
] = module.row
  }
  # Criamos um data.frame com os resultados
  results = data.frame(interactions, results.row, results.col)
#########################################################################
##########
```

```
  ################################## Calculando z e c
##################################
##########################################################################
##########
  ################################### Calculo do z
#####################################
  #### Calculando kis, numero de interacoes de cada especie i dentro do seu
modulo ####
  levels.col = length(levels(interactions$spp.col)) # Numero de especies em
interacions$spp.col
  kis.col = rep(NA, levels.col)
  modules.col = rep(NA, levels.col)
  for (i in 1 : levels.col)
  {
    modules.col[i] = members$module[members$spp[i]] # Modulo da primeira
especie (Ci)
    # Soma de quantas interacoes a especie i possuem dentro do seu modulo
    kis.col[i] = sum((results$spp.col == levels(results$spp.col)[i]) &
(results$results.row == modules.col[i] & results$results.col ==
modules.col[i]))
  }
  kis.col # Vetor com os valores de kis para cada Ci
  modules.col # Vetor com os modulos de cada um dos Ci do vetor kis.col
  ###### Calculando kis para os Ri ######
  levels.row = length(levels(interactions$spp.row))
  kis.row = rep(NA, levels.row)
  modules.row = rep(NA, levels.row)
  for (i in 1 : levels.row)
  {
    # Modulo da primeira especie Ri
    modules.row[i] = members$module[members$spp[i +
length(levels(interactions$spp.col))]]
    # Criamos o vetor kis.row da mesma forma que criamos kis.col
    kis.row[i] = sum((results$spp.row == levels(results$spp.row)[i]) &
(results$results.row == modules.row[i] & results$results.col ==
modules.row[i]))
  }
  kis.row # Vetor com os valores de kis para cada Ri
  modules.row # Vetor com os modulos de cada um dos Ri do vetor kis.r
  #### Fazendo um ciclo que percorra todos os modulos e para cada um
calcular ks e SDks ####
  n.modules = rep(NA, length(unique(members$module)))
  # Criamos um vetor que tenha em cada posicao o numero de especies do
modulo
  # que so tem interacoes dentro do modulo
  for (i in min(members$module) : max(members$module))
  {
    n.modules[i + 1] = length(c(kis.col[modules.col == i],
kis.row[modules.row == i]))
  }
```

```
  ##### Calculando o z de cada especie e colocando os valores em um vetor
#####
  z.list = list()
  for (i in min(members$module) : max(members$module))
  {
    ks.mean = mean(c(kis.col[modules.col == i], kis.row[modules.row == i]))
    ks.sd = sd(c(kis.col[modules.col == i], kis.row[modules.row == i]))
    z.list[[i + 1]] = (c(kis.col[modules.col == i], kis.row[modules.row ==
i]) - ks.mean) / ks.sd
  }
  # Cada posicao da lista kis.list contem os valores de z das especies de um
modulo,
  # concatenamos esses valores em um unico vetor
  z.vetor = c()
  for (i in 1 : length(z.list))
  {
    z.vetor = c(z.vetor, z.list[[i]])
  }
  ###### Criando uma tabela final que contem as especies e os respectivos
valores de z ######
  # Precisamos criar uma lista que contenha, em cada posicao, vetores com as
especies
  # (Ci e Ri) que possuem interacoes dentro de um modulo. A lista tera o
mesmo numero
  # de posicoes que o numero de modulos. Precisamos fazer isso para parear
essas especies
  # com os valores de z calculados.
  spp.list = list()
  for (i in min(members$module) : max(members$module))
  {
    spp.list[[i + 1]] = members$spp[members$module == i, drop = T]
  }
  spp.vetor = c()
  for (i in 1 : length(spp.list))
  {
    spp.vetor = c(spp.vetor, levels(spp.list[[i]]))
  }
  z.vetor = round(z.vetor, 3)
  # Guardamos os resultados de z pareados com as especies correspondentes em
um data.frame
  results.z = data.frame(spp.vetor, z.vetor)
  colnames(results.z) = c("spp","z")
  # Ordenamos as especies
  results.z[order(results.z$spp),]
  ################################### Calculo do c
#####################################
  ##### Criando um vetor que contem o numero de interacoes total (k) de cada
especie da rede #####
  k.spp = rep(NA, length(members$spp))
  # Primeiro colocamos no vetor os valores para as especies Ci
  for (i in 1 : length(levels(results$spp.col)))
```

```
  {
    k.spp[i] = sum(results$spp.col == levels(results$spp.col)[i])
  }
  # Agora preenchemos o resto com os valores para as especies Ri
  for (i in 1 : length(levels(results$spp.row)))
  {
    k.spp[i + length(levels(results$spp.col))] = sum(results$spp.row ==
levels(results$spp.row)[i])
  }
  ##### Fazer uma lista (kit.list) que contenha os valores de Kit de cada
especie em cada posicao #####
  # Esse vetor tera em cada posicao o numero de interacoes que a especie
realiza com o
  # modulo (uma posicao para cada modulo)
  sum.kit = rep(NA, length(unique((members$module))))
  kit.list = list()
  # Os valores de m percorrem todas as especies Ci. Para cada uma delas,
criamos o vetor
  # sum.kit que contem os valores de Kit daquela especie para cada modulo.
Colocamos
  # o vetor de cada especie em uma posicao da lista kit.list
  for (m in 1 : length(levels(results$spp.col)))
  {
    for (i in min(members$module) : max(members$module))
    {
      sum.kit[i + 1] = sum(results$spp.col == levels(results$spp.col)[m] &
results$results.row == i)
    }
    kit.list[[m]] = sum.kit
  }
  # Agora colocamos na lista kit.list os vetores sum.kit das especies Ri
  for (m in 1 : length(levels(results$spp.row)))
  {
    for (i in min(members$module) : max(members$module))
    {
      sum.kit[i + 1] = sum(results$spp.row == levels(results$spp.row)[m] &
results$results.col == i)
    }
    kit.list[[m +  length(levels(results$spp.col))]] = sum.kit
  }
  ####### Usando o vetor k.spp e a lista kit.list, calculamos o c de cada
especie ########
  results.c = rep(NA, length(members$spp))
  for (i in 1 : length(members$spp))
  {
    results.c[i] = 1 - sum((kit.list[[i]]/k.spp[i])^2) # Formula para
calcular o c
  }
  results.c = round(results.c, 3)
  results.c = data.frame(members$spp, results.c)
```

```
  colnames(results.c) = c("spp","c")
  ################ Colocando os valores de z e c em uma mesma tabela
################
  members.z.c = data.frame(members, results.z$z, results.c$c)
  colnames(members.z.c) = c("spp", "module", "z", "c")
  colnames(results) = c("spp.row", "spp.col", "module.row", "module.col")
  results.final = list(results, members.z.c)
}
```

Arquivos para baixar

med2.txt

members_med2.txt.txt

funcao_zc_modules.r

funcao_zc_modules.txt

help_zc_modules.txt