

# Danilo Muniz



Biólogo, Doutorando em Ecologia na USP, na modalidade Comportamental e Evolutiva. Minha tese de doutorado é sobre seleção sexual e sistemas de acasalamento. Envolve opiliões, sítios de oviposição e um mega experimento em campo. O orientador dessa maluquice toda é o Glauco Machado.

exec

## Trabalho Final - Propostas

### Plano A - montagem de uma matriz de relacionamentos a partir de dados de encontro

Esta função receberia dados de “encontro”, em cada linha representa um evento em que dois indivíduos foram avistados próximos entre si, as duas colunas seguintes trazem a identificação dos indivíduos e uma quarta coluna pode trazer uma medida de distância entre os indivíduos.

Caso uma distância seja fornecida, a função deve receber um parâmetro proximidade mínima para que o evento seja considerado um “encontro” (o que esse encontro representa é problema de quem chamar a função). Além do conjunto de dados e da distância limite, a função recebe como parâmetros dois valores lógicos indicando se a matriz resultante deve ser binária (apenas zeros e uns) ou com pesos (valores entre zero e um), e se os indivíduos dividem-se em duas classes (constituindo uma rede bipartida) ou não.

A função devolve a matriz de relacionamentos e a conectância da rede.

### Plano B - Simulação de monogamia com cópulas extra-par

A idéia é simular um sistema de acasalamento monogâmico e espacialmente explícito, imitando um ninhal de aves com reprodução colonial, com diferentes níveis de cópulas extra-par e parâmetros de escolha deste parceiro adicional. O espaço seria representado de forma matricial, com cada célula da matriz representando o espaço para um “ninho”. Cada ninho possui um macho e uma fêmea, e a fêmea pode, ou não, copular com um macho adicional.

Os machos teriam um atributo de qualidade (provavelmente seguindo uma distribuição normal) que poderia ser usado como parâmetro de escolha da fêmea. As fêmeas por sua vez teriam um raio de procura na matriz (único para todas), cada fêmea escolhe seu parceiro dentro de seu raio de procura de forma aleatória ou baseada na qualidade dos machos.

Os parâmetros iniciais da função seriam: dois inteiros representando o tamanho da matriz, um inteiro representando o raio de procura da fêmea, um número de 0 a 1 representando a proporção de fêmeas que vão realizar cópulas extra-par, e um valor lógico se as fêmeas escolhem o macho adicional ou copulam aleatoriamente.

A função devolveria um vetor com a distribuição de número de cópulas por macho e algumas estatísticas de média e variância de número de fêmeas por macho e o potencial para a seleção sexual

(aqui computada como  $\text{média}^2/\text{variância}$  das fêmeas por macho).

## Comentários

Os planos parecem bons e interessantes. No plano A parece que o desafio está mais na construção da matriz de interação e no seu limiar de distância.. Lendo a descrição dos dados de entrada do Plano A fiquei imaginando se a forma mais simples de representar esses dados não seria uma matriz com colunas e linhas representadas por indivíduos e o valor na matriz representada pelo número de interações... depois percebi que no final essa é um dos resultados, portanto parto da premissa que os dados sejam coletados da forma como descreveu e um dos desafios é construir a matriz, além de calcular a conectância. Nesse sentido, fiquei imaginando se não seria útil uma função que fizesse isso para limiares diferentes de distâncias. Ou seja, dado uma matriz de avistamento qual seria a conectância dependendo da distância crítica definida. Uma possibilidade de resultado é um gráfico da conectância em função da distância crítica, o que mostraria como essa decisão arbitrária muda a interpretação dos mesmos dados!

O plano B parece interessante tb, me parece que faltou um contexto teórico <sup>1)</sup>, de qq forma a proposta ficaria muito mais sedutora independente da ignorância do leitor... tipo umas quatro linhas!

— [Alexandre Adalardo de Oliveira](#) 2012/04/03 21:46

### Respondendo ao comentário

Decidi seguir pelo Plano A, a idéia é realmente pegar dados que são coletados no campo em forma de “encontros” e transformar em uma matriz de relacionamentos. Acho que pode ser útil para qualquer um que trabalhe com redes sociais de interações.

Gostei da ideia de calcular a conectância para diferentes distâncias críticas, e estou trabalhando nisso. Acho que também vou dar a opção da função gerar um gráfico.

(E realmente ficou faltando dar um contexto teórico melhor pro plano B...)

— [Danilo](#) 2012/04/11 20:11

## Trabalho Final

### Página de ajuda

monta.rede  
Documentation

package: none

R

Monta redes de relacionamento a partir de observações de encontro ou proximidade entre indivíduos.

## Description

A função gera uma matriz de relacionamento a partir de dados de encontros entre indivíduos (um encontro é um evento no qual os indivíduos foram observados próximos entre si).  
Nesses dados podem ou não constar valores da distância entre os indivíduos quando observados, e nesse caso o usuário deve fornecer pelo menos um valor distância crítica, a distância máxima entre dois indivíduos para que um encontro seja considerado válido.

## Usage

```
Function monta.rede (x, y, d=NA, dist.critica=0, bip=FALSE,  
transforma="brut", graf=FALSE)
```

## Arguments

x, y vetores de character com as identidades dos indivíduos que foram observados juntos ou próximos. A função considera que o individuo em x[1] esteve próximo do indivíduo y[1] e assim por diante.

d vetor de distâncias entre os indivíduos x e y quando foram vistos juntos, se não for informado todos os encontros são considerados válidos.

dist.critica  
valor único ou vetor numérico com as distancias críticas, valores de distância máxima para que um encontro seja considerado válido. Caso um vetor seja fornecido, uma matriz de relacionamento é construída para cada distância crítica.

bip parâmetro lógico, define se a rede é bipartida ou não. Numa rede bipartida há duas classes de indivíduos e só pode haver conexões entre os indivíduos de classes diferentes.

transforma  
parâmetro do tipo character que pode receber as opções "brut", "bin" ou "cont". Caso receba "brut" os dados não são transformados e uma matriz com o número de encontros entre cada par de indivíduos é devolvida. "bin" significa uma transformação para dados binários, e "cont" representa a transformação em um valor contínuo entre zero e um.

graf parâmetro lógico que define se um gráfico de conectância em função de distância

crítica deve ser desenhado, caso dados de distância não tenham sido fornecidos, ou haja apenas uma distância crítica o gráfico nunca é desenhado, mesmo que graf receba TRUE.

## Values

Retorna uma lista contendo:

matriz1 ... matrizn: uma matriz de relacionamento para cada distância crítica. A matriz pode conter a contagens de encontros entre os indivíduos, valores binários ou entre zero e um de acordo com a transformação escolhida (ver parâmetro transforma)

conectancias: um vetor com o valor de conectância de cada matriz.

## Details

O parâmetro "bip" só deve ser usado como TRUE caso a rede seja perfeitamente bipartida.

Caso esse parâmetro receba TRUE e haja indivíduos com a mesma identidade nos vetores x e y a função considera que eles são indivíduos diferentes. A função não está preparada para lidar com matrizes com mais que duas classes de indivíduos.

Os vetores x, y e d devem possuir os mesmos comprimentos. Caso o usuário não queira utilizar um vetor de distâncias este deve receber NA (valor default).

## References

Costa, L. da F., Rodrigues, F.A., Travieso, G. & Boas, P.R. 2007. Characterization of complex networks: A survey of measurements. *Advances in Physics*, 56 (1):167-242.

Krause, J., Croft, D.P. & James, R. 2007. Social network theory in the behavioral sciences: potential applications. *Behavioral Ecology and Sociobiology*, 62 (1):15-27.

## Examples

```
#exemplo com rede bipartida
```

```
#40 encontros aleatorios entre 8 mongois e 11 franceses
```

```
mongois =sample(paste("M", 1:8, sep=""), 40, replace=TRUE)  
franceses =sample(paste("F", 1:11, sep=""), 40, replace=TRUE)  
distancias = (runif(40, min=0, max=100))
```

```
d.criticas = c(20,40,60,80,100)

monta.rede(mongois, franceses, distancias, d.criticas, bip=TRUE,
transforma="brut", graf=TRUE)

#exemplo com rede unipartida
#mais de 100 encontros aleatorios entre dez pessoas

pessoas = paste("p", 1:10, sep="")
p.linhas = sample(pessoas, 130, replace=TRUE)
p.colunas = sample(pessoas, 130, replace=TRUE)

#exclusao de encontros de um individuo com ele mesmo
encontros.consigo.mesmo = p.linhas==p.colunas
p.linhas = p.linhas[!encontros.consigo.mesmo]
p.colunas = p.colunas[!encontros.consigo.mesmo]

p.distancias = runif(length(p.linhas), min=0, max=200)

p.d.crit = c(12,25,50,100)
monta.rede(p.linhas, p.colunas, p.distancias, p.d.crit, bip=FALSE,
transforma="bin", graf=TRUE)
```

### Código da função

```
#PROJETO FINAL DO R
#=====
#FUNCAO CONECTANCIA, SO Eh VALIDA PRA REDES NAO DIRECIONADAS, COMO EH
#O CASO AQUI
conectancia.rede = function(matriz, bip=FALSE)
{
  if (bip == TRUE)
  {
    maximo.conexoes = dim(matriz)[1] * dim(matriz)[2]
    resultado = (sum(matriz>0))/maximo.conexoes
  }
  else
  {
    maximo.conexoes = ( dim(matriz)[1] * (dim(matriz)[1]-1)) / 2
    resultado = (sum(matriz[upper.tri(matriz)]>0))/maximo.conexoes
  }
  return(resultado)
}
#=====
#FUNCAO DE TRANSFORMACAO DA MATRIZ
transforma.rede = function(matriz, transforma="bin")
{
  if (transforma == "bin")
    matriz[matriz!=0] = 1
  else if (transforma == "cont")
```

```
    matriz = matriz / max(matriz) #calcula uma forza de interacao
    return(matriz)
}
#FUNCAO MONTA.REDE=====
#funcao principal que monta a rede
monta.rede = function(x, y, d=NA, dist.critica=0, bip=FALSE,
transforma="brut", graf=FALSE)
{

    #se o usuario pedir uma transformacao invalida
    #a funcao avisa e nao transforma os dados
    if (transforma!="bin" & transforma!="cont" & transforma!="brut")
    {
        cat("\tTransformacao invalida de dados! Dados brutos serao
devolvidos.\n")
        transforma="brut"
    }

    if (is.na(d[1]))
        n.matrizes = 1
    else
        n.matrizes = length(dist.critica)

    #ETAPA 1 - preparacao dos vetores de individuos e da matriz vazia
    if (bip) #prepara dois vetores de individuos
    {
        indL = as.character(unique(x)) #individuos das linhas
        indC = as.character(unique(y)) #individuos das colunas
    }
    else #se a rede nao eh bipartida, cada individuo tem uma linha e uma
coluna
        indL = indC = as.character(unique(c(x,y)))
    n.linhas = length(indL)
    n.colunas = length(indC)
    n.encontros = length(x)

    #matriz de valores zero e lista vazia
    #as identificacoes dos individuos sao nomes de linhas e colunas
    matriz = matrix(0, nrow=n.linhas, ncol=n.colunas, dimnames=list(indL,
indC))
    lista = list(NA)
    #ETAPA 2 - contagem de encontros entre os individuos
    if (!is.na(d[1])) #se houver um vetor de distancias
    for (h in 1:n.matrizes) #para cada distancia critica
    {
        for(i in 1:n.encontros) #percorre os encontros
        {
            if (d[i]<=dist.critica[h]) #caso o encontro seja valido soma 1 na
matriz
                matriz[x[i], y[i]] = matriz[x[i], y[i]] +1
            }
        }
    }
}
```

```
}
if (!bip)
  matriz = matriz+t(matriz)
#se a rede nao eh bipartida tem que somar encontros entre i e j com os
entre j e i

#ETAPA 3 - TRANSFORMACAO DOS DADOS
#transformacao dos dados
if (transforma != "brut")
  matriz = transforma.rede(matriz, transforma)

#ETAPA 4 - GUARDANDO NUMA LISTINHA (E NOMEANDO POR NUMERO)
lista[[h]] = matriz
names(lista)[h] = paste("matriz", h, sep="")
matriz[1:n.linhas, 1:n.colunas] = 0 #zera a matriz pra proxima rodada

} #fecha o for de distancias criticas
else #se nao ter vetor de distancias ele cai aqui
{ #aqui tem as etapas 2 a 4 de novo, mas para os dados sem distancia
  for(i in 1:n.encontros)
    matriz[x[i], y[i]] = matriz[x[i], y[i]] +1 #para cada encontro soma
1 na matriz

  if (!bip)
    matriz = matriz+t(matriz)

  if (transforma != "brut")
    matriz = transforma.rede(matriz, transforma)

  lista[[1]] = matriz
  names(lista)[1] = "matriz"
}

#ETAPA 5 - CALCULO DAS CONECTANCIAS
conectancias = NA

for (k in 1:n.matrizes)
  conectancias[k] = conectancia.rede(lista[[k]], bip)

lista[[n.matrizes+1]] = conectancias
if (n.matrizes > 1)
  names(lista)[n.matrizes+1] = "conectancias"
else
  names(lista)[n.matrizes+1] = "conectancia"

#ETAPA BONUS - GRAFICO!
if (graf & n.matrizes > 1)#se so tem uma matriz não tem poque fazer
grafico!!
  plot(conectancias~dist.critica, bty="l", xlab="Distancias criticas",
ylab="Conectancia", col="red", pch=17, cex=2)
```

```
return(lista)  
}#end
```

## Arquivo do Código

[monta.r](#)

1)

falha minha!

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

Permanent link:  
[http://ecor.ib.usp.br/doku.php?id=05\\_curso\\_antigo:r2013:alunos:trabalho\\_final:daniomuniz:start](http://ecor.ib.usp.br/doku.php?id=05_curso_antigo:r2013:alunos:trabalho_final:daniomuniz:start)

Last update: **2020/08/12 09:04**

