ATENÇÃO: ESTA PÁGINA É UMA VERSÃO ANTIGA DO ROTEIRO E ESTÁ DESATIVADA, PARA ACESSAR O ROTEIRO ATUAL ACESSE ESTE LINK

# Teoria neutra da biodiversidade - Roteiro em R

A Teoria Neutra é um modelo de processos estocásticos de nascimentos, mortes, especiações e migrações. As probabilidades de cada um destes eventos ocorrerem definem uma dinâmica surpreendente. A melhor maneira de entender isto é simular este processo, como faremos nos exercícios a seguir.

## Preparação: entendendo caminhadas aleatórias

A Teoria Neutra usa uma classe de modelos de dinâmica estocástica, a caminhada aleatória de soma zero. Por isso precisamos entender algumas propriedades importantes dessas dinâmicas.

Faça os tutoriais de caminhadas aleatórias. Quando estiver certo(a) de que compreendeu esses dois modelos, volte para cá.

## A Teoria Neutra: simulação passo a passo


Agora que entendemos os modelos de caminhada aleatória e soma zero vamos construir o modelo estocástico da Teoria Neutra, passo a passo, com funções em R.

### Preparação: ambiente R

Este exercício é feito no ambiente de programação e análise de dados R. Você não precisa conhecer a linguagem R para fazê-lo, porque damos os comandos já prontos para executar. Eles estão reproduzidos nesta página, e também em um arquivo, abaixo. A única coisa que você precisa saber é como enviar os comandos escritos neste arquivo para o R. Para isso você pode copiar os comandos desta página e colar na linha de comando do R. Mas é bem mais prático usar o arquivo de comandos, ou *script*. Para isso, siga os seguintes passos:

1. Instale em seu computador o ambiente R, instruções aqui.
2. Crie um diretório em seu computador para os exercícios.

3. Copie para este diretório o arquivo abaixo:
    1. Códigos das funções: funcoes_neutr.r
    2. Todos os comandos deste exercício: ex_neutra.r
4. Abra o R a partir do arquivo de comandos `ex_neutra.r`. Certifique-se de que você está no diretório onde estão os arquivos.
5. Carregue no R as funções que vamos usar neste exercício digitando `source("funcoes_neutr.r")` na linha de comando.
6. Os comandos no arquivo "ex_neutra.r" estão na mesma ordem deste exercício. Siga o roteiro, enviando os comandos indicados a cada seção.
7. Se você não sabe como enviar os comandos do arquivo veja aqui.

## Dinâmica Local sem Migração

Vamos começar com um modelo para a comunidade em um dado local, usando um jogo de soma zero, similar ao jogo de apostas da seção anterior. As regras são:

1. A comunidade tem um total fixo de indivíduos, $J$
2. A cada intervalo de tempo, um dos indivíduos é sorteado para morrer
3. Em seguida, os indivíduos remanescentes são sorteados, para definir quem produzirá o filhote que ocupará o lugar do morto.

Para simular este processo, usamos a função em R `sim.hub1`. Se você não carregou o arquivo funcoes_neutr.r com todas as funções como descrito na seção anterior, copie e cole o código abaixo na linha de comando

```
rich <- function(x)length(unique(x)) ## funcao auxiliar

sim.hub1=function(S= 100, j=10, D=1, ciclo=2e4, step=1000){
  ## Tamanho da comunidade
  J <- S*j
  ##Matrizes para guardar os resultados
  ## matriz da especie de cada individuo por ciclo
  ind.mat=matrix(nrow=J,ncol=1+ciclo/step)
  ##CONDICOES INICIAIS##
  ##Deduzidas de acordo com o modelo de Hubbell:
  ## Todas as especies comecam com o mesmo numero de individuos (j=J/S)
  ind.mat[,1] <- rep(1:S,each=j)
  cod.sp <- ind.mat[,1]
  ##Aqui comecam as simulacoes
  for(i in 2:(1+ciclo/step)){
    for(j in 1:step){
      ##Indice dos individuos que morrem
      morte <- sample(1:J,D)
      ##Indice dos individuos que produzem filhotes para substituir os
mortos
      novos <- sample(1:J,D,replace=T)
      ##Substituindo
      cod.sp[morte]<-cod.sp[novos]
    }
    ## A cada step ciclos os resultados sao gravados
```

```
    ind.mat[,i] <- cod.sp
  }
  tempo <- seq(0,ciclo,by=step)
  colnames(ind.mat) <- tempo
  invisible(ind.mat)
  plot(tempo,apply(ind.mat,2,rich), xlab="Tempo (ciclos)", ylab="N de
espécies", type="l",
       main=paste("Dinâmica Neutra sem Colonização", "\n S=",S," J=",J)
       ,ylim=c(0,S))
}
```

Os argumentos desta função são:

- S: número inicial de espécies
- `j`: número inicial de indivíduos por espécies. Começamos com o mesmo número de indivíduos por espécie, portanto o tamanho da comunidade será $J=Sj$
- D : número de mortes por ciclo, que manteremos sempre em uma.
- `ciclo`: número de intervalos a simular
- `step`: intervalo de registro dos dados, como nas funções anteriores.

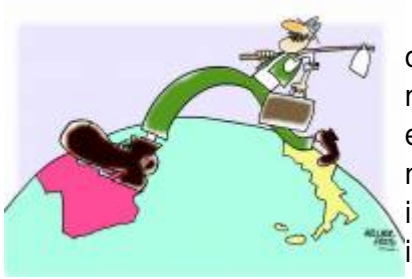
Simule uma comunidades com 100 espécies e 2 indivíduos por espécie:

```
sim.hub1(S=100, j=2)
```

O que acontece com o número de espécies com o passar do tempo? Verifique se isto muda aumentando o tamanho da comunidade, que é o produto $Sj$. Portanto basta manter o mesmo número de espécies e aumentar o número de indivíduos por espécie:

```
par(mfrow=c(2,2))## para 4 gráficos na mesma janela
sim.hub1(S=100, j=2)
sim.hub1(S=100, j=4)
sim.hub1(S=100, j=8)
sim.hub1(S=100, j=12)
par(mfrow=c(1,1)) ## Volta a um grafico por janela
```

## Incluindo Migrações



Sabemos que as comunidades não são sistemas fechados. Então a chegada de migrantes pode compensar a perda de espécies que vimos na simulação anterior. Vamos supor, então, que há um reservatório externo de migrantes, que chamamos **metacomunidade**. Uma maneira bem simples de se fazer isto é supor uma metacomunidade infinita, com todas as espécies do início da simulação, nas proporções iniciais. Precisamos definir também a taxa de migração: ela será a probabilidade de um indivíduo morto na comunidade ser substituído por um propágulo vindo de fora, da metacomunidade.

Abaixo está a função de R que inclui esta modificação. Ela tem mais um argumento, `m`, que é taxa de migração.

```
sim.hub2=function(S= 100, j=10, D=1, ciclo=2e4, step=1000, m=0.05){
  ## Tamanho da comunidade
  J <- S*j
  ##Matrizes para guardar os resultados
  ## matriz da especie de cada individuo por ciclo
  ind.mat=matrix(nrow=J,ncol=1+ciclo/step)
  ##CONDICOES INICIAIS##
  ## Todas as especies comecam com o mesmo numero de individuos (j=J/S)
  ## Rotulo de especies para cada um dos inividuos
  ind.mat[,1] <- rep(1:S,each=j)
  ## Repetindo este rotulo no vetor que sofrera modificacoes
  cod.sp <- ind.mat[,1]
  ##Aqui comecam as simulacoes
  for(i in 2:(1+ciclo/step)){
    for(j in 1:step){
      ##Indice dos individuos que morrem
      morte <- sample(1:J,D)
      ## Indice dos individuos mortos que serao repostos por migrantes
      defora <- sample(c(TRUE,FALSE),size=D,replace=T,prob=c(m,1-m))
      ##Indice dos individuos que produzem filhotes para substituir os
mortos
      novosd <- sample(1:J,D-sum(defora),replace=T)
      novosf <- sample(1:J,sum(defora),replace=T)
      ##Substituindo
      ## Mortos por propagulos de dentro
      if(length(novosd)>0){
        cod.sp[morte[!defora]]<-cod.sp[novosd]
      }
      ## Mortos por propagulos de fora
      if(length(novosf)>0){
        cod.sp[morte[defora]]<-ind.mat[,1][novosf]
      }
    }
    ## A cada step ciclos os resultados sao gravados
    ind.mat[,i] <- cod.sp
  }
  tempo <- seq(0,ciclo,by=step)
  colnames(ind.mat) <- tempo
  invisible(ind.mat)
  plot(tempo,apply(ind.mat,2,rich), xlab="Tempo (ciclos)", ylab="N de
espécies", type="l",
       main=paste("Dinâmica Neutra com Colonização da Comunidade Original",
"\n S=",S," J=",J," m=",m),ylim=c(0,S))
  }
```

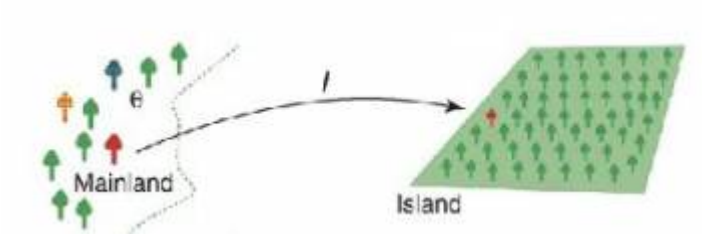
Compare a dinâmica de número de espécies ao longo do tempo em comunidades sem migração, e com valores crescentes de taxa de migração com os comando abaixo. Em todos começamos com uma comunidade com 100 espécies, com dois indivíduos por espécies.

```
par(mfrow=c(2,2)) ## abre espaço para 4 graficos na mesma janela
sim.hub2(S=100, j=2,m=0)
```

```
sim.hub2(S=100, j=2,m=0.1)
sim.hub2(S=100, j=2,m=0.2)
sim.hub2(S=100, j=2,m=0.4)
par(mfrow=c(1,1))
```

O que acontece se aumentamos o tamanho da comunidade? Experimente começar com 10 indivíduos por espécie.

## Uma Metacomunidade mais Realista




Um reservatório infinito de espécies não parece ser uma premissa muito realista. Que tal substituí-lo por um conjunto de populações com a mesma dinâmica que usamos para a comunidade? Teríamos, então, dois sistemas acoplados, cada um com sua dinâmica estocástica de nascimentos e mortes.

Mas se a metacomunidade também segue a dinâmica estocástica de soma zero, também perderá espécies com o tempo. Como resolver? Começamos por admitir que a metacomunidade é muito maior que a comunidade, pois representa o *pool* regional de colonizadores. Ou seja, é um sistema bem maior, pois tem mais espécies e indivíduos. Vamos supor, muito modestamente, que nela há o dobro de espécies da comunidade, cada uma com dez vezes mais indivíduos.

Apenas para lembrar o efeito do tamanho da comunidade sobre a erosão de espécies, use novamente a função de simulação sem migração para comparar sistemas que diferem nesta ordem de grandeza:

```
par(mfrow=c(2,1))
sim.hub1(S=100, j=2, ciclo=2e4,step=500)
sim.hub1(S=200, j=20, ciclo=2e4,step=500)
par(mfrow=c(1,1))
```

Já vemos que para tamanhos razoáveis (ou mesmo pequenos) de metacomunidades a erosão de espécies é bem lenta. Portanto, uma entrada de espécies também a uma taxa muito lenta já seria suficiente para compensar as extinções. Se for tão lenta quanto o tempo necessário para a evolução de uma nova espécie no sistema já temos a solução: na metacomunidade, as espécies perdidas são repostas por novas que surgem, no tempo evolutivo!

Assim, definimos uma taxa de especiação, $\nu$[282)], que expressa a probabilidade de um indivíduo morto na metacomunidade ser reposto por um indivíduo de uma nova espécie. Esta taxa é extremamente baixa, mas pode ser suficiente para manter, ou mesmo elevar, o número de espécies na metacomunidade.

Aqui vai a função para simular estes dois sistemas acoplados, que é o cenário imaginado por Hubbell:

```
sim.hub3=function(Sm=200, jm=20, S=100, j=2, m=0.01, nu=0.0001, D=1,
ciclo=1e4, step=100){
```

```
  ## Tamanho da metacomunidade
  Jm <- Sm*jm
  ## Tamanho da comunidade
  J <- S*j
  ##Matrizes para guardar os resultados
  ## matriz da especie de cada individuo por ciclo
  ## Na metacomunidade
  meta.mat=matrix(nrow=Jm,ncol=1+ciclo/step)
  ## Na comunidade
  ind.mat=matrix(nrow=J,ncol=1+ciclo/step)
  ##CONDICOES INICIAIS##
  ## Todas as especies comecam com o mesmo numero de individuos (j=J/S)
  ## METACOMUNIDADE
  meta.mat[,1] <- rep(1:Sm,each=jm)
  ## Repetindo este rotulo no vetor que sofrera modificacoes
  meta.sp <- meta.mat[,1]
  ##COMUNIDADE
  ## Rotulo de especies para cada um dos individuos
  ind.mat[,1] <- rep(1:S,each=j)
  ## Repetindo este rotulo no vetor que sofrera modificacoes
  cod.sp <- ind.mat[,1]
  ##Aqui comecam as simulacoes
  for(i in 2:(1+ciclo/step)){
    for(j in 1:step){
      ##Indice dos individuos que morrem
      ## Na comunidade
      morte <- sample(1:J,D)
      ## Na metacomunidade
      meta.morte <- sample(1:Jm,D)
      ## Indice dos individuos mortos da comunidade que serao repostos por
migrantes
      defora <- sample(c(TRUE,FALSE),size=D,replace=T,prob=c(m,1-m))
      ## Indice dos individuos mortos da metacomunidade que serao repostos
por novas especies
      meta.defora <- sample(c(TRUE,FALSE),size=D,replace=T,prob=c(nu,1-nu))
      ##Indice dos individuos que produzem filhotes para substituir os
mortos da comunidade
      novosd <- sample(1:J,D-sum(defora),replace=T)
      novosf <- sample(1:Jm,sum(defora),replace=T)
      ##Indice dos individuos que produzem filhotes para substituir os
mortos da metacomunidade
      meta.novosd <- sample(1:Jm,D-sum(meta.defora),replace=T)
      meta.novosf <- sample(1:Jm,sum(meta.defora),replace=T)
      ##Substituindo
      ## N metacomunidade ##
      ## Mortos por propagulos de dentro
      if(length(meta.novosd)>0){
        meta.sp[meta.morte[!meta.defora]]<-meta.sp[meta.novosd]
      }
      ## Mortos por novas especies
      if(length(meta.novosf)>0){
```

```
        meta.sp[meta.morte[meta.defora]]<-max(meta.sp)+1
      }
      ## Na comunidade ##
      ## Mortos por propagulos de dentro
      if(length(novosd)>0){
        cod.sp[morte[!defora]]<-cod.sp[novosd]
      }
      ## Mortos por propagulos de fora
      if(length(novosf)>0){
        cod.sp[morte[defora]]<-meta.sp[novosf]
      }
    }
    ## A cada step ciclos os resultados sao gravados
    ind.mat[,i] <- cod.sp
    meta.mat[,i] <- meta.sp
  }
  tempo <- seq(0,ciclo,by=step)
  colnames(ind.mat) <- tempo
  colnames(meta.mat) <- tempo
  resultados <- list(metacomunidade=meta.mat,comunidade=ind.mat)
  invisible(resultados)
  ## Graficos
  plot(tempo,apply(meta.mat,2,rich), xlab="Tempo (ciclos)", ylab="N de
espécies", type="l",
       main=paste("Dinâmica Neutra com Colonizacao da Metacomunidade", "\n
Jm=",Jm," nu=",nu," Theta=",2*Jm*nu,
         "S=",S," J=",J," m=",m), ylim=c(0,max(apply(meta.mat,2,rich))))
  lines(tempo,apply(ind.mat,2,rich),col="red")
  }
```

Agora temos argumentos também para os parâmetros da metacomunidade:

- `Sm`: número de espécies
- `jm`: número de indivíduos por espécie
- `nu`: taxa de especiação

Usando os tamanhos de comunidades e metacomunidades que já definimos, avalie o efeito de aumentar a taxa de migração, mantendo os outros parâmetros constantes:

```
par(mfrow=c(2,2))
sim.hub3(S=100, j=2,Sm=200,jm=20,nu=1e-9, m=0)
sim.hub3(S=100, j=2,Sm=200,jm=20,nu=1e-9, m=0.1)
sim.hub3(S=100, j=2,Sm=200,jm=20,nu=1e-9, m=0.2)
sim.hub3(S=100, j=2,Sm=200,jm=20,nu=1e-9, m=0.4)
```

Experimente também variar os tamanhos da comunidade e da metacomunidade, as taxas de migração e de especiação. Outra boa idéia é aumentar o tempo das simulações, para avaliar a dinâmica a longo prazo. Para isto, aumente o valor do argumento `ciclo`, ou a simulação pode ficar muito lenta.

### Perguntas

- Em escala de tempo ecológico a metacomunidade desta simulação tem efeito muito diferente da metacomunidade fixa e infinita da simulação anterior?
- Qual o efeito de uma maior taxa de especiação na metacomunidade sobre a dinâmica da metacomunidade?
- O que acontece se a metacomunidade é muito pequena?

## Para saber mais

### Introduções

- Cassemiro, F.A.S. & Padial, A.A. 2008. Teoria Neutra da Biodiversidade: aspectos teóricos, impacto na literatura e perspectivas. Oecologia Brasiliensis, 12 (4): 706-719.
- Alonso, D., R. S. Etienne, and A. J. Mckane 2006. The merits of neutral theory. Trends in Ecology & Evolution 21: 451-457.
- Um pacote em R para simulação e ajuste dos modelos de distribuição de espécies previstos pela teoria. **A introdução é uma excelente explicação da teoria**:
    - Hankin, R. 2007. Introducing untb, an R Package For Simulating Ecological Drift Under the Unified Neutral Theory of Biodiversity. Journal of Statistical Software 22: 12 http://www.jstatsoft.org/v22/i12/.

### Mais avançadas

- O livro (referência básica, mas nem sempre didática quanto ao modelo):
    - Hubbell, S.P. (2001). The Unified Neutral Theory of Biodiversity and Biogeography. Princeton University Press.
- Rosindell, J., Hubbell, S. P. & Etienne, R. S. 2011. The Unified Neutral Theory of Biodiversity and Biogeography at Age Ten. Trends in Ecology & Evolution 26:340-348. Ótima revisão sobre o tema e seu impacto.
- Renshaw, E. 1991. Modelling biological populations in space and time Cambridge University Press. Excelente apresentação de dinâmicas estocásticas.
- Uma boa revisão da evidência empírica até a época, com comparações com outros modelos neutros: Brian J. McGill, Brian A. Maurer, Michael D. Weiser (2006) EMPIRICAL EVALUATION OF NEUTRAL THEORY. Ecology: Vol. 87, No. 6, pp. 1411-1423.

R, comunidades, teoria neutra

282)

letra grega “nu”, correpondente ao nosso “n”