

Simulação da estrutura espacial de uma população vegetal

Dois processos principais atuam determinando a estrutura espacial de populações vegetais: os processos ligados ao nicho (ou seja, em que a estrutura ambiental seria o principal fator para o estabelecimento de um indivíduo em um determinado local) e os processos neutros (em que o principal fator seria a limitação de dispersão). Segundo Janzen-Connell, a probabilidade de um propágulo alcançar e sobreviver um determinado local segue a distribuição de Poisson.

Ao longo de muitos anos na literatura se debateu qual destes dois processos seria mais importante e atualmente se considera que ambos podem ocorrer simultaneamente.

Proponho na minha função simular a dinâmica de uma população vegetal segundo os modelos de nicho, neutro e ambos simultaneamente. Neste caso estamos interessados principalmente no padrão espacial final, mas seria possível também acessar as simulações intermediárias. Para isto, pensei em fazer a função de duas maneiras diferentes:

Primeira maneira

Utilizando espaço contínuo

Prós: seria mais fácil visualizar padrões de agregação em qualquer x_{max} e y_{max}
Contras: Não tenho certeza de como transformaria as informações da matriz ambiente para espaço contínuo. Essa ideia está menos estruturada na minha cabeça.

Entrada:

```
nichoneutro(N, modelo, nsim, xmax, ymax, plot = TRUE, i.neutro, i.nicho)
```

- n = número de indivíduos (*classe: integer, $N > 0$*).
- $nsim$ = Numero de iterações da simulação (*classe: integer, $N > 0$*).
- $modelo$ = Modelo a ser escolhido (*classe character: "nicho", "neutro", "nichoneutro"*).
- x_{max} = x máximo (*classe: integer, $N > 0$*).
- y_{max} = y máxima (*classe: integer, $N > 0$*).
- $plot$ = lógico. Se verdadeiro retorna gráfico final da distribuição espacial.
- $i.nicho$ = influência proporcional dos processos relacionados a nicho (*classe numeric $0 < i.nicho < 1$*).
- $i.neutro$ = influência proporcional dos processos neutros (*classe numeric $0 < i.neutro < 1$*).

Verificação de parâmetros:

- n é um número inteiro e maior que 0? Se não, escreve: " N precisa ser um número inteiro e > 0 ."
- $nsim$ é um número inteiro e maior que 0? Se não, escreve: " $Nsim$ precisa ser um número

inteiro e > 0."

- modelo é um character contendo "nicho", "neutro" ou "nichoneutro"? Se não, escreve *"modelo precisa ser um character contendo "nicho", "neutro" ou "nichoneutro"*.
- xmax é um número inteiro e maior que 0? Se não, escreve: *"xmax precisa ser um número inteiro e > 0."*
- ymax é um número inteiro e maior que 0? Se não, escreve: *"ymax precisa ser um número inteiro e > 0."*
- n é menor que nrow*ncol? Se não, retorna *"não há espaço para tantas arvores. Diminua N ou aumente nrow e/ou ncol."*
- plot é lógico? Se não, escreve *"plot precisa ser lógico."*
- $0 < i.neutro < 1$? Se não, escreve *"i.neutro deve ser entre 0 e 1."*
- $0 < i.nicho < 1$? Se não, escreve *"i.neutro deve ser entre 0 e 1."*
- Os argumentos *i.nicho* e *i.neutro* estão sendo utilizados quando *modelo = "nicho"* ou *"neutro"*? Se sim, escreve *"Os argumentos i.nicho e i.neutro não estão sendo utilizados"*.
- $i.nicho + i.neutro = 1$? Se não, escreve *"a soma de i.nicho e i.neutro deve ser 1."*

Pseudo-Código:

- Cria objeto espaço, um dataframe com n linhas e 2 colunas (x e y) contendo apenas 0.
- Sorteia valores aleatórios entre 0 e xmax para x, e 0 e ymax para y e coloca em espaço
- Cria dataframe simulação com nsim linhas e quatro colunas: x_mort, y_mort, x_col, y_col, contendo NAs.

Se modelo = "nicho"

- Cria o objeto ambiente, uma matriz com ymax linhas e xmax colunas. Geração de gradiente ambiental adaptando da função [cria_gradiente.r](#). A matriz terá valores entre 0 e 1 que estarão associadas a probabilidade de mortalidade naquela posição.
- Ciclo for de 1 a nsim
 - Sorteio de uma árvore para morrer de acordo com as probabilidades da matriz ambiente¹⁾
 - Guarda x_mort[i] e y_mort[i] em simulação
 - Sorteio aleatório para uma posição de uma nova árvore, sobrescreve a árvore que morreu
 - Guarda x_col[i] e y_col[i] em simulação

Se modelo = "neutro"

- Ciclo for de 1 a nsim.
 - Sorteio aleatório de uma árvore para morrer.
 - Guarda x_mort[i] e y_mort[i] em simulação.
 - Sorteio de uma árvore para se reproduzir.
 - Sorteio de uma distância em relação a árvore escolhida seguindo a probabilidade de Poisson.
 - Sorteio de um radiano.
 - Guarda posição da nova árvore em espaço, sobrescrevendo a que morreu
 - Guarda x_col[i] e y_col[i] em simulação.

Se modelo = “nichoneutro”

- Cria o objeto ambiente, uma matriz com ymax linhas e xmax colunas. Geração de gradiente ambiental adaptando da função `cria_gradiente.r`. A matriz terá valores entre 0 e 1 que estarão associadas a probabilidade de mortalidade naquela posição.
- Ciclo for de 1 a nsim
 - Sorteio de uma árvore para morrer de acordo com as probabilidades da matriz `ambiente*i.nicho`
 - Guarda `x_mort[i]` e `y_mort[i]` em simulação
 - Sorteio de uma distância em relação a árvore escolhida seguindo a probabilidade de Poisson, com média proporcional a `i.neutro`.
 - Sorteio de um radiano.
 - Guarda posição da nova árvore em espaço, sobrescrevendo a que morreu
 - Guarda `x_col[i]` e `y_col[i]` em simulação.

Saída:

- Se `plot = TRUE`, Gera gráfico em que cada cédula corresponde a uma posição da matriz espaço ao final de todas iterações. Coloca um ponto em cada cédula que apresenta 1. Se `modelo = “nicho”` ou `“nichoneutro”` também coloca em escalas de cores a probabilidade de mortalidade associada aquela posição.
- Retorna dataframe simulação

Segunda maneira

Utilizando matrizes para indicar a localização dos pontos.

Prós: talvez seja mais simples fazer um gradiente ambiental associado a posição de cada ponto, uma vez que cada ponto de espaço terá sua correspondência na matriz ambiente

Contras: seria necessário ncol e nrow muito grandes para verificar alguma estrutura espacial

Entrada:

```
nichoneutro(N, modelo, nsim, ncol, nrow, plot = TRUE, i.neutro, i.nicho)
```

- `n` = número de indivíduos (*classe: integer, $N > 0$*).
- `nsim` = Numero de iterações da simulação (*classe: integer, $N > 0$*).
- `modelo` = Modelo a ser escolhido (*classe character: “nicho”, “neutro”, “nichoneutro”*).
- `ncol` = número de colunas de habitat da paisagem (*classe: integer, $N > 0$*).
- `nrow` = número de linhas de habitat da paisagem (*classe: integer, $N > 0$*).
- `plot` = lógico. Se verdadeiro retorna gráfico final da distribuição espacial.
- `i.nicho` = influência proporcional dos processos relacionados a nicho (*classe numeric $0 < i.nicho < 1$*).
- `i.neutro` = influência proporcional dos processos neutros (*classe numeric $0 < i.neutro < 1$*).

Verificação de parâmetros:

- n é um número inteiro e maior que 0? Se não, escreve: “ N precisa ser um número inteiro e > 0 .”
- $nsim$ é um número inteiro e maior que 0? Se não, escreve: “ $Nsim$ precisa ser um número inteiro e > 0 .”
- `modelo` é um character contendo “`nicho`”, “`neutro`” ou “`nichoeneutro`”? Se não, escreve “`modelo` precisa ser um character contendo “`nicho`”, “`neutro`” ou “`nichoeneutro`”.
- $ncol$ é um número inteiro e maior que 0? Se não, escreve: “ $ncol$ precisa ser um número inteiro e > 0 .”
- $nrow$ é um número inteiro e maior que 0? Se não, escreve: “ $nrow$ precisa ser um número inteiro e > 0 .”
- `plot` é lógico? Se não, escreve “`plot` precisa ser lógico.”
- $0 < i.neutro < 1$? Se não, escreve “ $i.neutro$ deve ser entre 0 e 1.”
- $0 < i.nicho < 1$? Se não, escreve “ $i.neutro$ deve ser entre 0 e 1.”
- Os argumentos `i.nicho` e `i.neutro` estão sendo utilizados quando `modelo = “nicho”` ou “`neutro`”? Se sim, escreve “Os argumentos `i.nicho` e `i.neutro` não estão sendo utilizados”.
- $i.nicho + i.neutro = 1$? Se não, escreve “a soma de `i.nicho` e `i.neutro` deve ser 1.”

Pseudo-Código:

- Cria objeto espaço, uma matriz com $nrow$ linhas e $ncol$ colunas contendo apenas 0.
- Sorteio aleatório de n posições de espaço e atribuição de 1 para as posições sorteadas.
- Cria dataframe `simulação` com $nsim$ linhas e quatro colunas: `x_mort`, `y_mort`, `x_col`, `y_col`, contendo NAs.

Se `modelo = “nicho”`

- Cria o objeto ambiente, uma matriz com $nrow$ linhas e $ncol$ colunas. Geração de gradiente ambiental adaptando da função `cria_gradiente.r`. A matriz terá valores entre 0 e 1 que estarão associadas a probabilidade de mortalidade naquela posição.
- Ciclo for de 1 a $nsim$
 - Sorteio de uma árvore para morrer de acordo com as probabilidades da matriz ambiente
 - Guarda `x_mort[i]` e `y_mort[i]` em `simulação`
 - Atribui 0 a posição de espaço sorteada
 - Sorteio aleatório da posição de uma nova árvore
 - Guarda `x_col[i]` e `y_col[i]` em `simulação`
 - Atribui 1 a posição de espaço sorteada

Se `modelo = “neutro”`

- Ciclo for de 1 a $nsim$
 - Sorteio aleatório de uma árvore para morrer
 - Guarda `x_mort[i]` e `y_mort[i]` em `simulação`
 - Atribui 0 a posição de espaço sorteada
 - Sorteio de uma árvore para se reproduzir

- Sorteio de uma distância em relação a árvore escolhida seguindo a probabilidade de Poisson.
- Sorteio de uma direção em relação a árvore original (N, S, L, O, NE, NO, SE, SO)
- Atribui 1 a posição de espaço sorteada
- Guarda `x_col[i]` e `y_col[i]` em simulação

Se modelo = “nichoneutro”

- Cria o objeto ambiente, uma matriz com `nrow` linhas e `ncol` colunas. Geração de gradiente ambiental adaptando da função `cria_gradiente.r`. A matriz terá valores entre 0 e 1 que estarão associadas a probabilidade de mortalidade naquela posição.
- Ciclo for de 1 a `nsim`
 - Sorteio de uma árvore para morrer de acordo com as probabilidades da matriz ambiente
 - Guarda `x_mort[i]` e `y_mort[i]` em simulação
 - Atribui 0 a posição de espaço sorteada
 - Sorteio de uma distância em relação a árvore escolhida seguindo a probabilidade de Poisson.
 - Sorteio de uma direção em relação a árvore original (N, S, L, O, NE, NO, SE, SO)
 - Atribui 1 a posição de espaço sorteada
 - Guarda `x_col[i]` e `y_col[i]` em simulação

Saída:

- Se `plot = TRUE`, gera gráfico em que cada cédula corresponde a uma posição da matriz espaço ao final de todas iterações. Coloca um ponto em cada cédula que apresenta 1. Se `modelo = “nicho”` ou “nichoneutro” também coloca em escalas de cores a probabilidade de mortalidade associada aquela posição.
- Retorna dataframe simulação

1)

Tenho dúvidas de como faria esta parte

From:

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

Permanent link:

http://ecor.ib.usp.br/doku.php?id=05_curso_antigo:r2018:alunos:trabalho_final:jennifer.auler:fun1



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