RAFAEL CHAVES

doutorando em Ecologia, Instituto de Biociências - USP

Estou cursando doutorado no Programa de Pós-Graduação em Ecologia - IB, e trabalho na Secretaria do Meio Ambiente do estado. Meu interesse em pesquisa gira em torno da interface entre ciência ecológica e políticas públicas ambientais, com foco em restauração de ecossistemas e ecologia da paisagem. Tenho interesse em entender os drivers locais e de paisagem para o sucesso da restauração ecológica, bem como em compreender como seus benefícios podem ser otimizados na escala da paisagem.

Meus exercícios

Para acessar meus exercícios resolvidos, clique em: Exercicios

---

Proposta de Trabalho Final

PLANO A - Função: vizinhos

Contexto e usos potenciais

Calcular o número de vizinhos em um raio determinado a partir de múltiplos pontos permite inúmeras aplicações nas ciências biológicas.
No campo da restauração ecológica, no qual atuo, um sítio degradado (local) tem probabilidades mais altas de expressão de processos de regeneração natural quanto maiores forem as fontes de propágulos disponíveis (por exemplo, árvores nativas em uma paisagem dominadas por pasto). Assim, contabilizar o número de árvores em torno de um sítio em restauração, e a influência que este número pode ter na retomada de processos ecológicos importantes para a auto-sustentabilidade do ecossistema, pode ser uma ferramenta importante de análise e testes de hipóteses.
No entanto, diversos outros usos são possíveis. Para encontrar pontos com maiores chances de avistamento de uma espécie de ave que consome os frutos de determinada espécie de árvore, poderíamos selecionar locais candidatos a realização de ponto focal, e aplicando esta função poderíamos encontrar qual deles possui mais indivíduos da espécie de árvore em determinado raio de alcance.
Mesmo fora das ciências biológicas, podemos imaginar inúmeros outros usos. Por exemplo, alguém que queira se mudar e busque morar em um bairro arborizado, com uma base de dados das árvores

urbanas e suas coordenadas poderia aplicar esta função a uma lista de casas potenciais, verificando qual delas tem mais árvores por perto. O mesmo poderia ser feito trocando-se as árvores por serviços úteis, na análise de proximidade.

Visão geral da tarefa

A função irá ranquear locais (pontos) em ordem decrescente com relação ao número de vizinhos localizados a determinado raio de distância de cada local. O croqui com a distribuição espacial dos locais ranqueados e do conjunto de vizinhos será o produto colateral da função.

Passo a passo da função

Entrada

```
vizinhos(locais, vizi, raio, min.vizi, croqui)
```

- locais = Data frame com coordenadas x e y dos locais testados
- vizi = Data frame com coordenadas x e y dos vizinhos encontrados
- raio = Raio de abrangência no qual será computado o número de vizinhos
- min.vizi = Número mínimo de vizinhos a ser considerado para ranquear os locais e apresentá-los no croqui. Locais com número de vizinhos menor que este serão excluídos da análise
- croqui = Opção de gerar visualização gráfica da distribuição espacial dos locais e dos vizinhos

Configurações padrão para os argumentos de entrada

- O default para <min.vizi> será 0, de modo que todos os locais aparecerão no ranking final
- O default para <croqui> será TRUE, gerando portanto a visualização gráfica da distribuição espacial dos locais e dos vizinhos
- Os demais argumentos, <locais> , <vizi> e <raio>, não terão default, e deverão ser informados pelo usuário da função

Testes de premissa

- O objeto <locais> deve possuir exatamente duas colunas (para coordenadas x e y), se não for, a função para e retorna a mensagem “número de colunas para o objeto <locais> é diferente de 2”
- As duas colunas do objeto <locais> devem ser da classe <numeric>, se não for, a função para e retorna a mensagem “ao menos uma das colunas do objeto <locais> não é numerica”
- O objeto <vizi> deve possuir exatamente duas colunas (para coordenadas x e y), se não for, a função para e retorna a mensagem “número de colunas para o objeto <vizi> é diferente de 2”
- As duas colunas do objeto <vizi> devem ser da classe <numeric>, se não for, a função para e retorna a mensagem “ao menos uma das colunas do objeto <vizi> não é numerica”
- O objeto <raio> deve ser um número maior que zero. Se o objeto for menor ou igual a zero, para a função e retorna a mensagem “o 'raio' informado deve ser maior que zero”
- O objeto <min.vizi>, se informado, deve ser maior que zero. Se não for, a função para e retorna

a mensagem “ o argumento 'min.vizi' deve ser maior que zero.” (testar não no início, mas quando entrar no código)
- O objeto <croqui> deve ser da classe <logical> (testar não no início, mas quando entrar no código)

Correções dos argumentos

- Se o objeto <locais> contiver NAs, as respectivas linhas serão excluídas do objeto, e será apresentada a mensagem “as linhas de <locais> contendo NA(s) foram excluídas”
- Se o objeto <vizi> contiver NAs, as respectivas linhas serão excluídas do objeto, e será apresentada a mensagem “as linhas de <vizi> contendo NA(s) foram excluídas”
- O objeto <min.vizi>, se informado, deve ser da classe <integer>. Se não for, a função o converterá para <integer> (testar não no início, mas quando entrar no código)

Pseudo-código

1. Criar matriz com distâncias <dists> entre cada local (linhas da matriz) e cada um dos vizinhos (colunas da matriz)
2. Criar novo data frame <rank> a partir do objeto <locais> incluindo uma terceira coluna <n.vizi>
3. Atribuir à nova coluna <n.vizi> valores inteiros resultantes da soma de testes lógicos realizados entre os valores encontrados em <dists> e o <raio> informado, identificando quantas <dists> são menores ou iguais a <raio>. A soma dos resultados dos testes lógicos descrita acima retornará quantos vizinhos (colunas de <dists>) estão dentro do raio, para cada local (linhas de <dists>)
4. Criar critério de ordenamento para <rank>, em ordem decrescente para a coluna <n.vizi>
5. Reordenar <rank> segundo o critério estabelecido, sobrescrevendo o objeto anterior
6. Caso o usuário tenha informado um valor para o argumento <min.vizi>, os locais com número de vizinhos inferior a <min.vizi> serão excluídos do objeto <rank>.
7. Nesse caso, o novo <rank>, deverá sobrescrever o objeto <locais> anterior.
8. Caso o usuário não tenha informado um valor para o argumento <min.vizi>, o item anterior não é executado
9. Caso o usuário não tenha alterado o default de <croqui>, que indica geração de visualização gráfica, a função deverá:
    1. plotar a totalidade dos vizinhos no gráfico
    2. plotar, em cor ou símbolo diferente, os locais listados em <rank>
    3. plotar as circunferências dos raios de inclusão em torno de cada local
10. Caso a opção de <croqui> seja pela não geração de gráfico, o item anterior não é executado

Saída

- Tabela com locais ranqueados conforme o número de vizinhos, em ordem decrescente
- Se croqui=TRUE, é gerado também o gráfico de distribuição espacial dos locais ranqueados e da totalidade dos vizinhos, apresentando o raio de inclusão em torno de cada local

Desafios previamente identificados para a execução do plano A

- Aprender a utilizar a função order(), ou outra similar, para ranquear a tabela que será o produto principal da função
- Gerar coluna com número de vizinhos abrangido pelo <raio>
- Aprender a diferenciar graficamente pontos plotados nos argumentos <locais> e <vizi>
- Aprender a plotar o raio no entorno de cada um dos locais

PLANO B - Função: monitoramento

Contexto

O Brasil possui uma importante legislação que prevê a proteção e restauração da vegetação nativa, a Lei 12.651/2012, também conhecida como Novo Código Florestal. Esta lei prevê que determinadas áreas de preservação nos imóveis rurais, marcadamente as APPs (Áreas de Presenvação Permanente) e RLs (Reservas Legais) devem ser mantidas com vegetação nativa e, caso esta vegetaçnao esteja ausente, a vegetação deve ser restaurada. No entando, a lei federal não detalha os parâmetros que devem ser observados para considerar a vegetação restaurada, ficando tal regulamentação a cargo dos estados. O primeiro estado a regulamentar estes parâmetros foi São Paulo, que estipulou três indicadores ecológicos, que nada mais são que variáveis resposta que devem ser medidas em campo, por meio de metodologia pré-estabelecida, para verificar se os projetos de restauração atendem aos resultados ecológicos mínimos esperados, indicados por valores de referência (Chaves et al. 2015). Atualmente, outros estados como Rio de Janeiro, Mato Grosso e Distrito Federal já adotam sistemas semelhantes de monitoramento, e outros estão a caminho de adotar. Em todos estes casos, são os executores de projetos que medem as variáveis em campo, informando os resultados ao órgão ambiental. No entanto, o órgão precisa realizar auditorias em uma parte dos projetos.
A função descrita nesta proposta poderia auxiliar a tomada de decisão para a seleção de projetos a serem auditados, auxiliando a implementação das políticas ambientais em SP e outros estados brasileiros.
Apesar de ser esta a inspiração para a função, ela poderia ser utilizada por quaisquer interessados em comparar valores observados com valores esperados, para uma quantidade y de variáveis resposta em uma quantidade x de observações.

Visão geral da tarefa

A função irá confrontar: a) valores aferidos em campo para um conjunto de variáveis resposta; com b) valores de referência esperados. Serão aferidos: c) o número de variáveis com valores abaixo do esperado (nível crítico), e d) o índice de criticidade para identificar as observações que ficaram mais aquém dos valores esperados. Os produtos da função serão o histograma com a distribuição dos valores encontrados para o índice de criticidade e a tabela ranqueando as observações segundo o índice.

Passo a passo da função

Entrada

```
monitora(dados, referencia, salva.hist, salva.tabela)
```

- dados = Data frame composto por um número y de variáveis resposta (representados por colunas do data frame) e um número x de observações (linhas do data frame). Os nomes de cada variável resposta deverão constar como títulos de cada coluna (head=TRUE).
- referencia = Os valores de referência esperados estarão disponíveis neste argumento que deve ser um data frame de referência, com número de linhas e colunas iguais aos do argumento 1 <dados>
- salva.hist = permite salvar o histograma gerado, caso o usuário necessite
- salva.tabela = permite salvar a tabela de ranqueamento, caso o usuário necessite

Padrões para entrada

- salva.hist = NULL, indicando que não será salvo o histograma, apenas visualizado. O usuário pode alterar.
- salva.tabela = NULL, indicando que não será salva a tabela, apenas visualizada. O usuário pode alterar.
- <dados> e <referência> não terão padrão de entrada, devendo ser informados pelo usuário.

Testes de premissa

- Tanto para <dados> quanto para <referencia>, os NAs devem ser transformados para zero nos data frames antes de serem carregados como argumentos da função
- Todas as células devem ter valores numéricos
- O números de linhas e colunas de <dados> e <referencia> devem ser iguais entre si

Pseudo-código

1. Criar data frame de resultado, chamado <resultado>, que inicialmente será criado como uma cópia do objeto <dados>.
2. Criar nova coluna do data frame <resultado> chamada <n.criticos>, na qual constará, para cada observação, quantas das variáveis resposta tiveram valor observado inferior ao esperado.
3. Criar nova coluna do data frame <resultado> chamada <criticidade>, na qual será calculado, para cada observação, o “índice de criticidade” (ver abaixo) que permitirá identificar as observações que ficaram mais aquém dos valores esperados.
4. Estabelecer critério de reordenamento do data frame <resultado>:
    1. critério 1: ordem crescente dos índices de criticidade,
    2. critério 2: ordem decrescente de n.criticos.
5. Reordenar o data frame <resultado>, sobrescrevendo no mesmo objeto
6. Caso o usuário tenha informado que quer salvar o histograma com o comando <salva.hist>, deverá ser aberta função para salvar a imagem do histograma
7. Caso o usuário tenha informado que quer salvar o histograma com o comando <salva.tabela>, deverá ser aberta função para salvar a tabela retornada pela função

O índice de criticidade possui a seguinte fórmula preliminar:

```
Índice de criticidade = -(100^nc) + ∑ (vo-vr)
nc = número de variáveis resposta com nível crítico (abaixo do esperado)
vo = valores observados para cada variável resposta com nível crítico
vr = valores de referência para a respectiva variável resposta com nível
crítico
```

obs: Os elementos que sucedem o símbolo de somatório serão tantos quantas forem as colunas do argumento <dados>

Saída

- Gerar histograma (produto colateral da função) com a ocorrência dos valores do índice de criticidade encontrados para cada observação
- Apresentar tabela final, já ordenada em ordem crescente dos índices de criticidade

Observações

- A concepção do "índice de criticidade" foi produzido no âmbito desta proposta de trabalho final, por isso não está indicada referência bibliográfica.
- Após um tempo desenvolvendo a ideia deste plano B, percebi que aspectos que não estão claros para mim na tarefa em si podem dificultar a execução do código e help da função.
- O fato do "índice de criticidade" ainda não ter sido testado até o momento dificulta a minha concepção, e acaba me trazendo desafios menos relacionados ao R e mais ao funcionamento do índice.

Referências

- Chaves RB, Durigan G, Brancalion PHS, Aronson J (2015) On the need of legal frameworks for assessing restoration projects success: new perspectives from São Paulo state (Brazil). Restoration Ecology, 23(6): 754-759

—-

> Olá Rafael. As suas duas propostas são factiveis e com um nível de dificuldade equilibrado. Eu particularmente acho a proposta A mais interessante e útil para outros usuários. Na proposta A, eu tomaria cuidado com o o nome do parâmetro ("plot") que é o nome de uma função, isso vai confundir as pessoas e talvez até o R. Mas isso é um pequeno detalhe. Sobre os seus testes de premissa, talvez você possa deixar sua função mais robusta se você tentasse corrigir alguns erros. Por exemplo, se existe NA no data frame do plot ou do vizi, você remove os NAs e ao invés de dar erro dá um warning dizendo que rodou, mas removeu os NAs. O mesmo para numeric, vc a tenta força as informações para ser numeric e faz um teste lógico para saber se conseguiu transformar em número ou não. Se sim, dá um warning dizendo que teve que transformar, se não aí dá erro.

Acho que você não terá dificuldade em fazer essa função, mas vai te exigir um certo conhecimento sobre controle de fluxo. Seria legal se até a versão final da proposta você já conseguisse detalhar o que vai ter dentro dos for e if que você vai precisar. Mas claro, pensar nesses detalhes também faz parte do desenvolvimento da função em si. Portanto, fique a vontade para usar o tempo de fazer a função para pensar nesses detalhes.

Boa Sorte e taca-le pau!

— *Bruno Travassos* *2019/06/14 12:13*

Oi Bruno, Conforme havia indicado por email, concordei com sua avaliação de que a proposta A é mais interessante, e a selecionei para desenvolver o trabalho final. Sendo assim, na versão final da proposta, incorporei suas sugestões:

1. Alterar o nome do primeiro argumento de “plot” para “locais”
2. Acrescentar algumas correções de erros dos argumentos:
    1. Remoção de linhas com NAs em `locais` e `vizi`
    2. Transformar `min.vizi` na classe integer

Também incluí mensagens de aviso e refinei as premissas.
`locais` e `vizi` serem numéricos eu preferi manter como premissa, para que o usuário esteja consciente do que está fazendo, já que a conversão de arquivos de coordenadas que não estejam em escala absoluta (p.ex. geográficas), poderia retornar resultados errados.
No final achei que a função ficou menos engessada sem especificar um par(), de modo que o usuário pode inserir seus parâmetros gráficos favoritos antes de rodar a função.
Agradeço pelas sugestões!
Abraços,
Rafael

Trabalho Final

Para a entrega do trabalho final, decidi desenvolver a proposta A, cujo objetivo é ranquear locais quanto ao seu número de vizinhos a determinado raio, e secundariamente gerar um croqui para visualização gráfica. Dessa forma, desenvolvi o código da função `vizinhos`, disponível no link do item ***1***, bem como a respectiva página de ajuda, disponível no link do item ***2***:

1. Código da função: Função Vizinhos.
2. Página de ajuda da função: Help_Vizinhos.