Ingrid Pinheiro Paschoaletto

Mestranda em Zoologia, Instituto de Biociências, USP.

O título de minha tese é: “Análise da organização do cuidado parental em uma ave (*Prunella modularis*, Passeriformes, Prunellidae) com sistema variável de acasalamento”, orientado pelo Professor Dr. Eduardo S. A. Santos.

Meus exercícios

No link: Meus Exercícios

Proposta de trabalho final

Olá Ingrid,

Sou Gustavo A. Ballen, monitor encargado de dar um retorno sobre suas propostas. Vou comentar em geral aqui sobre as duas propostas.

A proposta A é basicamente uma expressão aritmética simples calculando cada componente dela em uma linha para finalmente integrar todas aqueles pequenos cálculos numa saída. Não condiz com o objetivo do trabalho final de ser uma função desafiante que incorpore os elementos de programação apresentados na disciplina (e.g., estruturas de controle de fluxo, estrutura de objetos). Já a proposta B é um script para gerar um boxplot a partir de um data frame; é simples demais e não demanda esforço algum para desenvolver.

Por gentileza apresentar reformule fortemente a primeira proposta, ou melhor ainda, prepare uma terceira, dessa vez levando em consideração que ela deve incorporar pelos menos estruturas de controle de fluxo e representar um desafio real que acrescente seu processo de aprendizado de programação no R.

Gustavo A. Ballen

— *Alexandre Adalardo de Oliveira* *2019/06/18 16:13* Olá Ingrid,

Concordo com a avaliação do Gustavo acima. Nenhuma das propostas faz uma tarefa que seja algo que mereça uma função específica. Note que não há controle de fluxo em nenhuma delas. Aguardamos a reformulação ou a nova proposta.

Olá. Discordo em alguns pontos com a avaliação, porém poupando tempo e energia, fiz uma terceira proposta (ver a PROPOSTA C ao final da página) que imagino atender aos requisitos exigidos. Aguardo retorno em breve para poder executar a proposta, obrigada.

- Ingrid

— *Alexandre Adalardo de Oliveira 2019/06/24 16:13* Olá Ingrid,

A proposta C é bem interessante. Não entendi alguns passos, que poderiam estar melhor explicados no pseudo-código, mas o contexto geral representa um bom desafio, concentre-se agora na execução.

Quando a sua discordância, estamos abertos a avaliar qualquer que seja desde que haja alguma proposição. Quando discordar, apresente a argumentação, a discordância sem ela parece apenas uma reclamação sem sentido. Como não houve, e ambos avaliadores tiveram a mesma percepção quanto a falta de controle de fluxo nas sua propostas, continuamos com nossa posição.

Olá professor.

Organizei melhor o pseudo-código para que ficasse mais claro e espero ter esclarecido as dúvidas. O código está pronto e o help também, ambos no final da página.

- Ingrid

Plano A: Sirena

Contextualização A função irá calcular o preço de venda de produtos de confeitaria (podendo abranger outros gêneros alimentícios também) produzidos artesanalmente, de acordo com os preços encontrados no mercado para a matéria prima de cada receita. Muitas pessoas trabalham no ramo da produção artesanal de doces, seja como uma renda principal, adicional ou um hobby. Quando se trabalha desta forma, uma das maiores dúvidas é: “Quanto devo cobrar pelo meu produto?”, pois o valor deve cobrir o custo, a embalagem, a mão de obra e o lucro desejado, e ainda assim ser atrativo ao cliente. Este cálculo pode ser feito de uma forma bem detalhada e complexa, levando em consideração n fatores envolvidos na precificação, mas muitas confeiteiras utilizam uma fórmula simplificada (abaixo) para facilitar esta etapa.

PVU = {[2*(C+I*C) + L*(C+I*C)] /R} + CE

Onde PVU é o 'preço de venda unitário', C é o custo da receita, I representa os 'custos incalculáveis' (ex água, energia. Normalmente varia entre 0,1 e 0,25), L é a porcentagem de lucro desejada (mínimo de 0,5), R corresponde ao rendimento da receita e CE ao custo da embalagem por unidade. Se esta for a renda principal do usuário, recomendamos que busque na internet ou em cursos especializados uma forma mais detalhada de determinar o valor do seu produto, uma vez que a fórmula mencionada não incluí despesas como por exemplo o aluguel de um ateliê/loja. Porém, no caso de confeiteiros iniciantes ou que fazem disto uma renda extra, esta função é de tremenda ajuda, uma vez que diminui muito o tempo gasto na etapa de precificação do produto.

Entrada O usuário entrará os seguintes argumentos: `venda.sirena(receita, rendimento, I, embalagem, mercado, quantidade, lucro)`

- `receita`: Objeto da classe data.frame, contendo 3 colunas (sendo estas 1- Ingrediente, 2- Unidade de medida e 3- Quantidade utilizada na receita) e um ingrediente por linha, sendo o limite de linhas o tanto de ingredientes utilizados na receita. Desejável a possibilidade de adicionar novas receitas a função.
- `rendimento`: Objeto da classe numérica, indicando quantas unidades (ou gramas) rende a receita colocada no primeiro argumento. Deve estar na mesma unidade do argumento 'quantidade'.
- `I`: Vetor numérico definindo os gastos incalculáveis, com o valor entre 0,1 e 0,25.
- `embalagem`: Objeto da classe data.frame com duas colunas, sendo a primeira os componentes da embalagem (ex: caixa, cm de fita, etiqueta, sousplat etc.) e a segunda a quantidade de cada um a ser utilizada em uma unidade do produto.
- `mercado`: Objeto da classe data.frame, contendo 5 colunas (sendo estas 1- Ingrediente, 2- Unidade de medida, 3- Quantidade comprada, 4- Valor pago, 5- valor/quantidade) e um item por linha, incluindo todos os utilizados na receita e na embalagem. Desejável a possibilidade de adicionar novos itens a esta lista.
- `quantidade`: vetor numérico indicando a quantidade final desejada do produto. Deve estar na mesma unidade do argumento 'rendimento'.
- `lucro`: Vetor numérico definindo a margem de lucro desejada, com o valor mínimo de 0,5.

Pseudocódigo

- 1. Para cada `receita$ingrediente`, a função irá multiplicar a quantidade utilizada pelo valor encontrado na última coluna do data.frame `mercado`, onde consta o `preço/quantidade` da compra daquele `ingrediente`. (Ainda preciso aprender como fazer essa parte).
- Irá guardar o resultado de cada `ingrediente` na nova coluna `receita$custo`.
- Soma todos os valores de `receita$custo` e guarda no objeto `custo.receita`.
- Multiplica o valor de `custo.receita` pelo valor definido no argumento `I` e guarda no objeto `IC`.
- Adiciona a `IC` o valor de `custo.receita` e guarda no objeto `CIC`.
- Multiplica `CIC` por dois e guarda no objeto `MO`.
- Multiplica `CIC` pelo valor definido no argumento lucro e guarda no objeto `L`.
- Soma os valores dos objetos `MO` e `L` e guarda em `MOL`.
- Divide `MOL` pelo valor definido no argumento rendimento e guarda no objeto `CR`.

- Para cada item do data.frame `embalagem`, a função irá multiplicar a quantidade utilizada pelo valor encontrado na última coluna do data.frame `mercado`, onde consta o `preço/quantidade` da compra daquele material. (Mesma observação do item 1 acima).
- Irá guardar o resultado de cada material na nova coluna `embalagem$custo`.
- Soma todos os valores de `embalagem$custo` e guarda no objeto `CE`.
- Soma os valores de `CR` e `CE` e guarda no objeto `PVU`.
- Retorna `cat("Custo=", custo.receita, " | CE=", CE, " | Lucro=", L, \n"O preço de venda sugerido é", PVU)`.

Saída A função deve retornar os valores de C (custo), CE (custo da embalagem) e o valor sugerido de venda. Deve estar da seguinte forma: *Custo = XXX | CE = XXX | Lucro = XXX "O preço de venda sugerido é XXXXX"*

Plano B: Cuidado Parental

Contextualização Na ecologia comportamental, um importante foco de estudo é o cuidado parental observado em diversas espécies e frequentemente estudado em aves. Este comportamento é muito interessante pois abarca uma série de conflitos, tanto entre os pais e a prole, quanto entre os próprios pais, especialmente em espécies com cuidado biparental. No último caso, o conflito se dá na decisão de cada parental acerca do quanto investir naquela prole de acordo com o investimento do parceiro, pois é mais vantajoso para cada pai que o seu parceiro faça todo o trabalho e ele apenas colha os benefícios do sucesso reprodutivo. Com isto em mente, seria interessante que houvesse uma forma prática de realizar uma análise exploratória do comportamento de cuidado parental de determinada espécie, baseado em observações realizadas ao longo de um período. A partir destes resultados primários, seria possível traçar um delineamento mais precisos para estudos futuros. A função proposta irá calcular a média de tempo que aves de ambos os sexos investem em cuidado parental ao longo de um período de observação. Ao final da análise a função deve plotar automaticamente um gráfico contendo as informações separadas por sexo.

Entrada `parentais(obs, tempo)`

- `obs`:objeto da classe data.frame contendo as observações do tempo despendido de cuidado parental de machos e fêmeas em uma população.
- `tempo`: objeto de classe numérica especificando qual o tempo total de observação em dias.

Pseudocódigo:

- Lê o data.frame `obs`.
- Chama o pacote `Dplyr` na biblioteca para facilitar a manipulação do data.frame.
- Através da função `group_by()` agrupa os dados das aves de acordo com o sexo.
- Cria uma nova tabela, apenas com as informações necessárias para o gráfico final, através da função `summary()`.
- Renomeia as colunas da nova tabela.
- Calcula a média, mediana, desvio padrão e quantis.
- Chama o pacote `ggplot2` para elaborar o gráfico `boxplot`.
- Plota-se o gráfico final.

Saída Ao final a função deverá retornar um gráfico contendo as informações adquiridas com a análise dos dados (média, desvio padrão, etc.).

Plano C: Gazelas

Contextualização

Dentro da disciplina de comportamento animal existem diversos modelos para tentar prever/explicar o que ocorre em determinadas situações observadas na natureza. Por vezes, estes modelos demoram a serem compreendidos pelos alunos, e como uma forma de facilitar o entendimento, muitos professores optam por utilizar jogos e simulações didáticas. Entre estes modelos há o de sinalização honesta de qualidade, onde essa sinalização serve para demonstrar para os seus predadores (ou parceiros sexuais) que são indivíduos de alta qualidade e assim alcançar o seu objetivo (não ser predado ou copular, respectivamente).

Um bom exemplo sobre sinalização honesta na interação presa/predador são as gazelas, que com frequência, ao encontrar um predador, saltam. A principal hipótese é de que este salto funciona como um sinal de força daquela potencial presa, indicando que ela será mais difícil de ser capturada, de modo que o predador desista de predá-la e opte pelas gazelas que não saltam, que seriam mais fracas.

Na disciplina, os professores dividiram a turma em grupos de 3 e cada grupo realizou uma dinâmica com o auxílio de um tabuleiro e dados para testar na prática esta situação das gazelas. A função aqui proposta tem como objetivo realizar este jogo e retornar um data.frame com o resultado, com a opção de simular várias vezes esta dinâmica e apresentar um gráfico com a frequência que as gazelas conseguiram escapar ou viraram comida de leão.

As regras do jogo são:

* O leão não sabe quem é a gazela forte e quem é a fraca.

* As gazelas dão um salto para demonstrar vigor ao leão, e este salto gasta pontos energéticos.

* As gazelas saem 5 posições a frente do leão.

* O leão escolhe a sua presa e a partir daí a gazela não escolhida já escapou e não gasta mais energia.

* A cada rodada é sorteado o número de ‘posições’ que cada jogador andará. Se o leão alcançar a gazela, esta é predada, porém se ela chegar ao fim das rodadas antes que o leão a alcance, ela escapa e o leão fica com fome.

Entrada

O usuário entrará o seguinte argumento na função:

```
gazela.saltitante(simular=TRUE, repetir=100, rodadas=15)
```

Se o usuário quiser ter acesso ao gráfico com a frequência de predação e fuga das gazelas, deverá colocar `simular=TRUE`, podendo alterar também o número de vezes que o jogo ocorrerá no argumento `repetir` (default=100). Caso queira ter acesso apenas ao data.frame de uma simulação, então deverá colocar `simular=FALSE`. Para ambos os casos o usuário irá poder alterar também o número de rodadas que o jogo terá no argumento `rodadas` (default=15).

Pseudocódigo

1. `if` simular=TRUE, a função entrará em um `for` para repetir o jogo tantas vezes quanto definido em `repetir` e os resultados (escapou ou morreu) separados pela classe da gazela (forte ou fraca) serão guardados em uma matriz para a confecção do gráfico (barplot) após o loop.
2. `if` simular=FALSE, a função pula o `for` e cria um data.frame `g` com uma linha nomeada para cada indivíduo (2 gazelas e 1 leão), onde:

- Na primeira coluna (ind) estará o número do indivíduo (1 ou 2 para as gazelas e NA para o leão);
- Na segunda o vigor da gazela (determinado aleatoriamente com o `sample` de 1:10 sem repetição, leão=NA);
- Na terceira a classe da gazela (forte ou fraca, dependendo do vigor, leão=NA);
- Na quarta o salto que servirá de 'pista' para o leão, sorteado em um `sample` cujo o range varia de acordo com a classe da gazela, leão=0)
- As colunas seguintes serão as rodadas do jogo, onde na linha da gazela escolhida pelo leão estarão os resultados dos samples de fuga e na linha do leão, os de perseguição. A linha da gazela não escolhida estará preenchida com o valor 5, que corresponde a sua posição inicial. O jogo acaba quando são realizadas todas as rodadas definidas no argumento da função. Se a gazela for capturada antes do jogo acabar o data.frame só mostrará as colunas das rodadas que aconteceram.
- A última coluna será preenchida com o gasto energético de todos, somando-se o salto inicial a última posição (menos 5) das gazelas e a última posição do leão, já que ele só gastou energia para perseguir a presa.

1. Os resultados dos saltos serão disponibilizados para o leão e este poderá escolher se irá perseguir a gazela 1 ou a 2 através do `readline ("Leão, escolha sua presa:")`.
2. A partir daí começa a perseguição. Com o auxílio das funções `if` e `else`, se a gazela escolhida for forte, será realizado um `sample(5:10,1)` a cada rodada para determinar sua fuga, se for fraca o `sample(3:8,1)`. O leão avança com um `sample(5:10,1)`.
3. A cada rodada serão verificadas as posições do leão e da gazela, caso leão>=gazela, o usuário receberá a mensagem "Gazela, você foi predada". Caso contrário o jogo continua até que se acabem as rodadas.

Saída

* Se simular=TRUE, a função retornará um gráfico semelhante a este:

* Se simular=FALSE, a função retorna um data.frame e uma frase informando o status da gazela. X corresponde aos resultados obtidos nas etapas da função:

CÓDIGO gazela.saltitante

No link: CÓDIGO

HELP gazela.saltitante

No link: HELP