

Rodrigo Silva do Carmo



Mestrando em Ecologia, Instituto de Biociências USP.

[Rodrigo Silva do Carmo](#)

PROPOSTA A: Cardápio aleatório condicionado às necessidades do usuário

Contextualização

A biodisponibilidade de um nutriente é a proporção em que ele é absorvido pela alimentação e usado para o funcionamento do organismo (Agget, 2010). No entanto, os nutrientes interagem uns com os outros, o que forma uma rede complexa de relações que pode mudar a biodisponibilidade de um nutriente e resultar em uma absorção reduzida ou até mesmo na inibição da absorção, ou pode ocorrer a maximização da absorção do nutriente (HKH van Het, 2010).

Nos tempos atuais, as pessoas se mostram interessadas e preocupadas com sua dieta alimentar, com intuito de evitar problemas de saúde associados a uma dieta desequilibrada, perda, ganho ou manutenção do peso e também para auxílio na definição da forma física atrelado à atividade física. Como cada organismo possui suas particularidades, a quantidade de caloria que uma pessoa precisa diariamente varia de acordo com a idade, sexo, altura, peso e nível de sedentarismo, dessa forma varia também a quantidade de caloria necessária para manter, ganhar ou perder peso (USDA, 2016).

Sabendo disso, e tendo como premissa que em geral as pessoas preferem variar os itens alimentares de uma refeição para a outra, a criação de uma ferramenta que permite o usuário criar um cardápio adequado para a maximização de um determinado nutriente, considerando uma quantidade x de quilocaloria que não deve ser ultrapassada por refeição e tendo maior diversidade de itens alimentares, é algo que pode ser muito prático e atrativo. Por tanto, a função proposta toma como parâmetros um nutriente desejado; um outro nutriente que aumenta a biodisponibilidade do nutriente desejado; os tipos de alimentos a serem incluídos no cardápio; um valor referência de kcal por refeição que não deve ser ultrapassado; número de refeições e os dados de onde todas as informações são tiradas.

Planejamento da função

Entrada: Nutricard (Nutri, Nutripar, alimento1, alimento2, alimento3, alimento4, alimento5, Size1, Size2, Size3, Size4, Size5, Kcal, N)

Nutri= vetor com os valores do nutriente desejado (classe: numeric, Nutri > 0).

Nutripar= vetor com os valores do nutriente que forma par com o nutriente desejado e aumenta sua biodisponibilidade (class:numeric, Nutripar> 0).

alimento1= dataframe contendo um subset de alimentos (tirado do banco de dados completo Dados) que pertencem a um grupo específico (ex:legume,fruta...) contendo Nutri, Nutripar, as quilocalorias de cada alimento e o restante de nutrientes de cada alimento com seus valores.

alimento2= dataframe contendo um subset de alimentos (tirado do banco de dados completo

Dados) que pertencem a um grupo específico (ex:legume,fruta...) contendo Nutri, Nutripar, as quilocalorias de cada alimento e o restante de nutrientes de cada alimento com seus valores.

alimento3= dataframe contendo um subset de alimentos (tirado do banco de dados completo Dados) que pertencem a um grupo específico (ex:legume,fruta...) contendo Nutri, Nutripar, as quilocalorias de cada alimento e o restante de nutrientes de cada alimento com seus valores.

alimento4= dataframe contendo um subset de alimentos (tirado do banco de dados completo Dados) que pertencem a um grupo específico (ex:legume,fruta...) contendo Nutri, Nutripar, as quilocalorias de cada alimento e o restante de nutrientes de cada alimento com seus valores.

alimento5= dataframe contendo um subset de alimentos (tirado do banco de dados completo Dados) que pertencem a um grupo específico (ex:legume,fruta...) contendo Nutri, Nutripar, as quilocalorias de cada alimento e o restante de nutrientes de cada alimento com seus valores.

Size1= Numero de itens para os alimentos em alimento1.

Size2= Numero de itens para os alimentos em alimento2.

Size3= Numero de itens para os alimentos em alimento3.

Size4= Numero de itens para os alimentos em alimento4.

Size5= Numero de itens para os alimentos em alimento5.

Kcal= Valor referência de caloria que não deve ser ultrapassado por refeição.

N= numero de refeições.

Exemplo (fictício) de dados de entrada (Alimento1 e Alimento2)

Alimento1:

Alimento	Tipo Alimento	Kcal	Ferro mg(Nutri)	VitaminaC mg(Nutripar)	... Demais nutrientes
Banana	Fruta	10	30	3.5	...
Mamão	Fruta	20	9	5	...
Uva	Fruta	15	4.5	6.9	...

Alimento2:

Alimento	Tipo Alimento	Kcal	Ferro mg(Nutri)	VitaminaC mg(Nutripar)	... Demais nutrientes
Abóbora	Legume	40	24	2.9	...
Pepino	Legume	27	18	9	...
Brócolis	Legume	31	19.9	3.3	...

Verificando os parâmetros:

Nutri é um vetor numérico e > 0 , se não escreve: "Nutri precisa ser da classe numeric e > 0 ".

Nutripar é um vetor numérico e > 0 , se não escreve: "Nutripar precisa ser da classe numeric e > 0 ".

”.

alimento1 é um dataframe, se não escreve: “alimento1 precisa ser um dataframe”.

alimento2 é um dataframe, se não escreve: “alimento2 precisa ser um dataframe”.

alimento3 é um dataframe, se não escreve: “alimento3 precisa ser um dataframe”.

alimento4 é um dataframe, se não escreve: “alimento4 precisa ser um dataframe”.

alimento5 é um dataframe, se não escreve: “alimento5 precisa ser um dataframe”.

Size1 é um valor inteiro e > 0 , se não escreve: “Size1 precisa ser valor inteiro e > 0 ”.

Size2 é um valor inteiro e > 0 , se não escreve: “Size2 precisa ser valor inteiro e > 0 ”.

Size3 é um valor inteiro e > 0 , se não escreve: “Size3 precisa ser valor inteiro e > 0 ”.

Size4 é um valor inteiro e > 0 , se não escreve: “Size4 precisa ser valor inteiro e > 0 ”.

Size5 é um valor inteiro e > 0 , se não escreve: “Size5 precisa ser valor inteiro e > 0 ”.

Kcal é um vetor numérico e > 0 , se não escreve: “Kcal precisa ser da classe numeric e > 0 ”.

N é um numero inteiro e > 0 , se não escreve: “N precisa ser numero inteiro e > 0 ”.

Pseudo-código:

1. Cria objeto medianutri com a média geral de Nutri obtida através de Dados.
2. Cria objeto medianutripa com a média geral de Nutripa obtida através de Dados.
3. Cria objeto richnutri1 com itens em Alimento1; richnutri2 com itens em Alimento2; richnutri3 com itens em Alimento3; richnutri4 com itens em Alimento4; richnutri5 com itens em Alimento5 que contenham valores de Nutri e Nutripa acima de medianutri e medianutripa, respectivamente, para excluir alimentos pobres no teor de nutriente desejado e nutriente par e obter apenas alimentos que possuem valores destes nutrientes acima das suas respectivas médias.
4. Entra em ciclo while contendo if (4.4.), em que o resultado das condições em if deve ser associado aos argumentos “TRUE” contidos dentro de while, com numero de replicas= N.
 - 4.1. A cada ciclo é amostrado aleatoriamente itens de richnutri1,richnutri2,richnutri3,richnutri4 e richnutri5 (cada um separadamente).
 - 4.2. A quantidade de amostras escolhidas de cada tipo de alimento é determinada por Size1, Size2,Size3,Size4 e Size5.
 - 4.3. Criar objeto alimentos contendo os alimentos selecionados aleatoriamente pelo ciclo.
 - 4.4. Utilizar função “if” para que A kcal por ciclo deve ser $\leq$ ou = Kcal, e Nutri e Nutripa devem ser maiores que medianutri e medianutripa a cada ciclo.
5. Criar lista cardprep contendo: cada refeição junto com seus respectivos alimentos + o

Nutritotal (soma de Nutri de cada alimento da refeição) + o Nutripartotal (soma de Nutripar de cada alimento da refeição) + a Kcaltotal (soma da kcal dos alimentos da refeição).

Saída:

Cardápio de refeições contendo o nome dos itens alimentares e o tipo de alimento a qual pertencem mais as calorias, valor do nutriente desejado e valor do nutriente por do nutriente desejado obtidos em cada refeição.

PROPOSTA B: Espécies e risco de extinção

Contextualização

Para a conservação da biodiversidade, se deve priorizar algumas espécies em relação a outras, para maximizar o esforço empregado, tendo maiores benefícios com menor custo. Para isso, uma das formas de decidir quais espécies possuem maior risco de extinção é através do conhecimento da distribuição geográfica das espécies, que pode ser descrito pela extensão de ocorrência e pela área de ocupação (Gaston & Fuller, 2009).

A extensão de ocorrência é descrita como a área contida dentro do menor limite imaginário (representado por uma linha) que engloba todos os pontos em que a espécie está presente, por outro lado, a área de ocupação considera a presença e ausência das espécies dentro do limite da extensão de ocorrência da espécie (IUCN, 2001).

Assim, para ter uma base fundamentada para poder avaliar quais espécies merecem maior atenção e esforço para conservação, além de saber a distribuição geográfica das espécies, é também imprescindível saber quais espécies são dominantes e quais são raras no sistema de estudo, e se a abundância das espécies muda de acordo com o tipo de ambiente analisado ou até mesmo com o tempo. Dessa forma, a função aqui proposta determinaria em quantos locais do sistema de estudo cada espécie está presente, com isso pode-se descobrir quais espécies possuem grande e pequena área de ocupação, resultado este que pode ser complementado com a comparação entre a abundância de cada espécie com a abundância média (soma da abundância de todas as espécies dividido pelo número de espécies presentes) do ponto ou região analisada, fornecendo um bom indicativo de dominância e raridade das espécies. Ao final, a função geraria uma lista contendo as espécies que tem abundância acima e abaixo da abundância média de cada ponto de acordo com o habitat (caso haja mais de um tipo de habitat) e replica (caso o usuário queira identificar mudanças temporais na abundância das espécies), e um dataframe contendo o total de pontos do sistema de estudo e o total de pontos em que cada espécie ocorre (para obter a área de ocupação).

Planejamento da função

Entrada: riscoextSP (area,pon,distr,abun,hab,rep,dados)

area= Vetor de classe "character" com os nomes das áreas que englobam os pontos.

pon = Vetor de classe "character" com os nomes dos pontos.

distr = TRUE or FALSE (verificar ou não a área de ocupação das espécies).

abun = TRUE or FALSE (verificar ou não a abundância de cada espécie em relação a abundância

média do ponto ou área (soma das abundancias das espécies dividido pela riqueza de espécies).

hab = Vetor com os tipos de habitat.

rep = TRUE or FALSE (analisar os dados temporalmente devido às réplicas por ponto ou desconsiderar a escala temporal, respectivamente).

dados = objeto dataframe (linha são observações contendo as abundancias das espécies e colunas são espécies).

Exemplo de dados a ser utilizado:

area	ponto	habitat	replica	sp1	sp2	sp3
floresta1	p1	F	1	5	0	6
floresta2	p1	F	1	2	7	5
floresta1	p2	F	2	0	11	1
floresta2	p1	M	1	1	2	0
floresta2	p2	M	2	2	3	0
floresta1	p3	F	3	9	1	2

Verificando os parâmetros: area é um vetor da classe “character”, se não escreve: “area precisa ser da classe character”.

pon é um vetor da classe “character”, se não escreve: “pon precisa ser da class character”.

distr é um argumento lógico TRUE or FALSE, se não escreve: “distr precisa ser TRUE or FALSE”.

abun é um argumento lógico TRUE or FALSE, se não escreve: “abun precisa ser TRUE or FALSE”.

hab é um vetor da classe character, se não escreve: “hab precisa ser da classe character”.

rep é um argumento lógico TRUE or FALSE, se não escreve: “rep precisa ser TRUE or FALSE”.

dados é um dataframe em que linhas são observações com a abundancia das espécies e colunas são as espécies, se não escreve: “dados precisa ser dataframe com observações nas linhas e espécies nas colunas”.

Pseudo-código:

1. If rep=FALSE soma-se a abundancia total (todas as réplicas) de cada espécie em dados e guarda no objeto dadossemrep de classe dataframe ELSE utiliza-se dados diretamente para considerar a variação temporal e guarda em dadoscomrep.

2. Criar objeto abuntotal contendo a abundancia total (todas as espécies) de cada ponto (através de dadossemrep) se rep=FALSE, caso rep=TRUE, calcula-se a abundancia total (todas as espécies) de cada ponto por réplica de cada ponto (através de dadoscomrep) e guarda no objeto abuntotalrep.

2.2. Criar objeto zeroum que é uma função que transforma dados de abundancia em presença/ausência.

2.2.1. Aplicar o objeto zeroum em dadoscomrep se rep= TRUE ou no objeto dadossemrep se rep= FALSE.

2.2.2. Criar objeto `riq`(através de `dadossemrep`) contendo a riqueza de espécies para cada ponto se `rep=FALSE`, caso `rep=TRUE` obtém-se a riqueza de cada réplica por ponto (através de `dadoscomrep`) e guarda no objeto `riqrep`.

2.3. Criar objeto `abunmedia` contendo a abundância média para cada ponto se `rep=FALSE`, através da divisão de `abuntotal` por `riq`. Caso `rep=TRUE`, criar objeto `abunmediarep` contendo a abundância média para cada ponto por réplica através da divisão de `abuntotalrep` por `riqrep`.

3. Criar objeto `spdomi` com nomes das espécies que possuem abundância $>$ `abunmedia` em cada ponto caso `rep=FALSE`. Se `rep=TRUE`, criar objeto `spdomirep` com nomes das espécies que possuem abundância $>$ `abunmediarep`.

3.1. Criar objeto `sprara` com nomes das espécies que possuem abundância $<$ `abunmedia` em cada ponto se `rep=FALSE`. Caso `rep=TRUE`, criar objeto `sprararep` com nome das espécies que possuem abundância $<$ `abunmediarep`.

4. Criar data frame `areaocup` com a relação entre o total de ocorrências de cada espécie e o total de pontos presentes no sistema de estudo.

5. Criar lista `spINFO` contendo: `spdomi`, `sprara` e `areaocup` se `rep=FALSE`, caso `rep=TRUE`, criar lista com o nome `spINFOrep` contendo: `spdomirep`, `sprararep` e `areaocup`.

Saída:

Lista contendo 3 objetos: as espécies por ponto ou ponto e réplica que possuem abundância acima da abundância média do ponto ou do ponto e réplica, as espécies por ponto ou ponto e réplica que possuem abundância abaixo da abundância média do ponto ou ponto e réplica, e um dataframe com o número total de pontos no sistema de estudo e o número total de ocorrências de cada espécie (área de ocupação).

Referências:

Aggett PJ. (2010). Population reference intakes and micronutrient bioavailability: a European perspective. *American Journal of Clinical Nutrition* 91(suppl):1433S-1437S.

Gaston, K. J., FULLER, R. A. (2009). The sizes of species' geographic ranges. *Journal of Applied Ecology*, 46: 1-9, doi: 10.1111/j.1365-2664.2008.01596.x.

IUCN. (2001). IUCN Red List categories and criteria: Version 3.1. Prepared by IUCN Species Survival Commission. World Conservation Union, Gland, Switzerland and Cambridge, United Kingdom. li + 30 pp.

USDA. (2016). Dietary Guidelines For Americans 2015-2020: 8th Edition.

van Het Hof KH, K. H., West, C. E., Weststrate, J. A., Joseph Hautvast, G.A.J. (2000). Dietary factors that affect the bioavailability of carotenoids. *Journal of Nutrition* 130(3):503-506.

Vitor Rios

Rodrigo, suas propostas precisam de um pouco mais de detalhamento pra poder decidir com qual seguir

A sua proposta A é interessante, mas fiquei com algumas dúvidas. A

entrada me parece confusa. Pra que serve o objeto dados? Porquê colocar os nutrientes como número e não pelo nome? Cada alimento só possui um nutriente? Aliás, Nutri e Nutripar precisam estar separados dentro do alimento? Seria interessante se você fizesse um exemplo dos dados de entrada pra podermos entender melhor como sua função vai funcionar e manipular esses dados. Porque fazer a média de Nutri e Nutripar?

A proposta B também precisa detalhar um pouco mais. area é um vetor de que tipo? você vai trabalhar com coordenadas ou nomes de locais? O mesmo vale para pontos. O que vc chama de “indícios sobre a área de ocupação das espécies”? Em rep, onde ficam as informações temporais? Em dados, que informação está dentro das observação? coordenadas? datas? nome do local? Tente esquematizar os objetos de entrada na forma de tabelas, ajuda a interpretar o que vc quer dizer. Como você vai realizar os cálculos, em que objeto e quando? se vc usar a função zerou antes de calcular as abundâncias vai dar erro

Quando vc tiver feito as modificações, me envie um email para eu olhar novamente [Vitor Rios](#)

Rodrigo Silva do Carmo

Escolhi a proposta A (**Nutricard**) como sugerido pelo monitor. Para que a função fosse mais generalizada, foram criados mais 3 argumentos ausentes na proposta inicial, são eles: o Nutriposi, Nutriparposi e Kcalposi que são referentes à posição da coluna de Nutri, Nutripar e Kcal, respectivamente. Foi criado um ciclo “for” dentro de um ciclo “while” e os seguintes objetos colocados dentro de alimentos: s1,s2,s3,s4,s5 contendo os itens selecionados nos objetos richnutri.

Após o término do ciclo “for”, foram criados os objetos:colKcal, colNutri e colNutripar contendo as colunas de Nutri,Nutripar e Kcal presente em todas as dataframes da lista cardprep. Logo cada coluna foi somada e os resultados guardados em: somacol1,somacol2, somacol3, posteriormente todos os elementos dos objetos foram testados logicamente se eram menores que Kcal e maiores que medianutri e medianutripar, respectivamente, e os resultados guardados em check.Kcal,check.Nutri e check.Nutripar.

Por fim, utilizou-se a função “if” para testar se os objetos recém criados satisfaziam as condições presente dentro do ciclo “while”.

Função Nutricard

[nutricard..r](#)

Help da Função Nutricard

[nutricard..help.txt](#)

From:

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

Permanent link:

http://ecor.ib.usp.br/doku.php?id=05_curso_antigo:r2019:alunos:trabalho_final:rodrigo.silva_19:start

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

