Natália Targhetta

Mestre em Ecologia pelo Instituto Nacional de Pesquisas da Amazônia (INPA), atualmente trabalha como técnica de laboratório no Departamento de Ecologia do IB-USP.

Exercício 1

Propostas Trabalho Final

Proposta A: Trabalhando com banco de dados.

Contextualização:

Atualmente, muitos pesquisadores na área da Ecologia Vegetal têm trabalhado com banco de dados, compilando informações sobre espécies ou grupos de espécies de plantas. O trabalho com banco de dados tem suas vantagens: na maioria das vezes, não é necessário saídas a campo e a obtenção da informação é, em um primeiro momento, rápida. No entanto, trabalhar as informações obtidas de bancos de dados pode demandar bastante tempo, uma vez que os diferentes bancos retornam informações de diferentes maneiras e é preciso organizá-las para que se consiga implementar as análises necessárias e extrair a informação desejada.

Um dos muitos usos que se pode fazer das informações fornecidas pelos bancos de dados é a modelagem da distribuição geográfica das espécies, que utiliza a informação fornecida pelos bancos de dados (ocorrência das espécies, coordenadas geográficas), atrelada a dados locais de altitude e variáveis climáticas, tais como as fornecidas pelo GoogleEarth e WorldClim/BioClim, principais fontes utilizadas para a obtenção das variáveis abióticas.

A função tem como objetivo organizar a informação dos bancos de dados mais utilizados para obter informação de espécies de plantas, especialmente as tropicais/neotropicais (GBIF, SpeciesLink, NYH). A função irá organizar as informações de modo que, ao final, estejam alocadas na mesma coluna, em uma única tabela. A função pode remover informação duplicada (com base em alguns critérios) e poderá obter as altitudes e variáveis climáticas dos pontos de ocorrência da espécie.

Função:

Entrada: plantbase (dados=c(“GBIF”, “SpLink”, “NYH”), rep, alt, bioclim)

Argumentos:

- dados = arquivos dos bancos de dados (class: data frame)
- GBIF = arquivo do GBIF
- SpLink = arquivo do SpeciesLink

- NYH = arquivo do New York Botanical Garden Herbarium
- rep = retira repetições (class logical: TRUE/FALSE)
- alt = busca altitude no Google Earth (class logical: TRUE/FALSE)
- bioclim = busca variáveis climáticas do bioclim (class logical: TRUE/FALSE)

Verificação dos Parâmetros:

- dados é um data frame? Se não for retorna “não é um data frame”.
- dados = GBIF e/ou SpLink e/ou NYH? Se não for retorna “dados precisam ser GBIF e/ou SpLink e/ou NYH”.
- rep não é lógico (TRUE ou FALSE)? Se não for retorna “necessita argumento lógico: TRUE ou FALSE”.
- alt não é lógico (TRUE ou FALSE)? Se não for retorna “necessita argumento lógico: TRUE ou FALSE”.
- bioclim não é lógico (TRUE ou FALSE)? Se não for retorna “necessita argumento lógico: TRUE ou FALSE”.

Pseudo-Código:

1. Preenche células vazias com NA.

2. Reorganiza/redefine as colunas de cada objeto (GBIF, SpLink, NYH).

3. Cria um novo objeto da classe data frame resultante da junção dos objetos anteriores.

4. Se rep = TRUE, verifica linhas com informação repetida.

```
A. Cria um ciclo para verificar repetição nas linhas referentes às seguintes informações das colunas: número de tombo no herbário, nome do coletor, número de coleta, data da coleta, localidade da coleta.
    A.1. Se informações das cinco colunas são iguais: deixa a linha do primeiro objeto inserido no argumento da função e exclui a(s) outra(s) linha(s).
    A.2. Se informações são iguais em quatro das cinco colunas: deixa a linha do primeiro objeto inserido no argumento da função e exclui a(s) outra(s) linha(s).
    A.3. Se informações são iguais em três das cinco colunas: retorna mensagem de aviso “checar dados das linhas”.
    A.4. Se informações são iguais em duas das cinco colunas: retorna mensagem de aviso “checar dados das linhas”.
```

5. Se alt = TRUE, abre o aplicativo Google Earth.

```
A. Busca altitude para cada linha se coordenadas geográficas são fornecidas.
B. Cria nova coluna no objeto e coloca o valor da altitude na linha correspondente.
```

6. Se bioclim = TRUE, instala e abre o pacote dismo.

```
A. Chama a função “biovars” do pacote dismo.
B. Aplica a função para cada linha se coordenadas geográficas são
fornecidas.
C. Cria colunas no objeto e coloca os valores das variáveis ambientais.
```

7. Salva o objeto no diretório de trabalho.

Saída:

Um data frame com colunas reorganizadas e valores de altitude e variáveis climáticas.

Proposta B: Uma ajuda para quem necessita de uma dieta restritiva

Contextualização:

Muitas pessoas necessitam de dietas restritivas devido ao surgimento de doenças ao longo de suas vidas, tais como o diabetes, que atualmente atinge cerca de 425 milhões de pessoas em todo o mundo (1). Uma das formas de controlar os níveis sanguíneos de glicemia é através da ingestão controlada de carboidratos (2). Sabendo a quantidade de carboidratos ingeridos, torna-se mais fácil ajustar a dose da medicação, e até mesmo interromper ou retardar o seu uso em alguns casos.

A quantidade de carboidratos que podem ser ingeridos diariamente varia de pessoa para pessoa, e até mesmo em uma mesma pessoa ao longo do tempo. No que diz respeito à quantidade de carboidratos ingeridos, as dietas podem ser classificadas da seguinte maneira (2):

1. Dieta cetogênica de muito baixo carboidrato: 20-50 g/dia ou < 20% de uma dieta de 2000 kcal/dia;

2. Dieta de baixo carboidrato: < 130 g/dia ou < 26% de uma dieta de 2000 kcal/dia;

3. Dieta de ingestão moderada de carboidratos: de 26% a 40% do total de uma dieta de 2000 kcal/dia;

4. Dieta de alta ingestão de carboidratos: > 45% do total de uma dieta de 2000 kcal/dia.

A função tem como objetivo calcular as quantidades/porções de alimentos de acordo com a quantidade de carboidratos escolhida, podendo ser separado por classes de alimentos, que podem ser escolhidos pelo usuário ou por sorteio pela função.

Função:

Entrada: carb.control (alimentos = c(vetor com alimentos escolhidos), g, ref = c(“cf”, ”al”, ”jn”, ”todos”), sort, info=c(vetor com alimentos escolhidos))

Argumentos:

- alimentos = alimentos escolhidos de acordo com a tabela da função (ver help) (class: vector).

Classificados nos seguintes grupos: proteínas, legumes, verduras, frutas, cereais, massas, laticínios, oleaginosas, doces.
- g = quantidade de carboidratos em gramas (class: integer).
- ref = tipo de refeição, podendo escolher entre um, dois ou todos os tipos (cf, al, jn, todos).
- cf = café, al = almoço, jn = jantar
- sort = sorteia alimentos e calcula respectivas quantidades (class logical: TRUE ou FALSE).
- info = informação sobre a quantidade de carboidratos no alimento contida na tabela de alimentos da função (class: vector).

Verificação dos Parâmetros:

- alimentos ≠ dos listados na tabela da função? Se sim, retorna “alimento não contido na tabela!”.
- ref = (“todos”) ou (“cf”, “al”, “jn”) e g < 20? Se sim, retorna “quantidade mínima de carboidratos ingeridos deve ser de 20 g por dia!”.
- sort é TRUE ou FALSE? Se não for retorna “sort deve ser TRUE ou FALSE!”.
- sort = TRUE e alimentos = (vetor com alimentos escolhidos)? Se sim, retorna “se deseja escolher seus alimentos, sort = FALSE”).

Pseudo-Código:

1. Se alimentos = (vetor com alimentos escolhidos), faz um sorteio e gera um data frame com as quantidades dos alimentos escolhidos.

```
A. Quantidades de alimentos dos grupos “proteínas”, “verduras”, “legumes”, “frutas” e “cereais” devem ser maiores do que quantidades de alimentos dos outros grupos.
```

```
B. Quantidade de carboidratos não deve ter diferença maior do que 20% entre grupos.
```

2. Se sort = TRUE, faz um sorteio dos alimentos e suas respectivas quantidades.

```
A. Se ref = “cf”, sorteia e calcula quantidades de alimentos das classes “laticínios”, “pães”, “frutas”, “oleaginosas”.
```

```
B. Se ref = “al” e/ou “jn”, sorteia e calcula quantidades de alimentos das classes “proteínas”, “verduras”, “legumes”, “cereais” ou “massa”.
```

```
C. Se ref = “todos”, pode sortear e calcular quantidades de todas as classes de alimentos, exceto “doces”.
```

3. se info = (vetor com nomes dos alimentos), retorna a informação da quantidade de carboidratos em uma porção determinada do alimento.

Saída:

Um data frame com o tipo de refeição e seus alimentos com quantidades/porções que somam a quantidade de carboidrato escolhida.

Referências:

1. IDF Diabetes Atlas 2017.

2. Feinman et al. 2015. Dietary carbohydrate restriction as the first approach in diabetes management: Critical review and evidence base. Nutrition, 31:1-13.

Comentários Julia

PROPOSTA A

Oi Natália,

Achei a proposta A bem clara e factível. Dois comentários:

1) Como as coordenadas geográficas são fornecidas nos bancos de dados? Todas no mesmo formato e DATUM? Sugiro checar isso e se for o caso incluir comandos para corrigir caso as coordenadas possam ser imcompatíveis entre sí e com o google earth.

2) Você já tem familiaridade em extrair os dados do google earth ? Sei que o pacote geonames têm uma função para elevação, caso ajude. E também a função elevation https://www.rdocumentation.org/packages/rgbif/versions/0.9.9/topics/elevation

PROPOSTA B Proposta B também é factível. Achei interessante a ideia, me parece mais simples que a proposta A mas pode se tornar mais complexa e um ótimo caminho e criativo.

1)Fiquei com uma dúvida, você diz que “podendo ser separado por classes de alimentos, que podem ser escolhidos pelo usuário ou por sorteio pela função.” Não sei se entendi bem, a função irá utilizar ou as informaçoes em “info” ou irá sortear alimentos dentro das classes contidas em “alimentos” , é isso?

2)Se estiver inspirada, sua função poderia também interagir com páginas na internet sobre alimentos. Olha só isso: Using R to Analyze Food Blogs - DataScience.com- https://www.datascience.com/blog/analyzing-food-blogs-with-r Achei curioso. Caso sinta-se segura, isso pode dar ideias de como tornar a função mais complexa.

Recomendo buscar desafios para obter uma ótima função (e avaliação) mas vamos conversando para manter dentro do plano factível para você. Então considere esse tutorial como uma inspiração.

Considerações finais

Acho as duas propostas factíveis e interessantes. Leve em consideração os comentários e sugiro escolher uma e seguir.

Ao fazer alterações na sua proposta, inclua aqui a nova proposta atualizada. (Não apague a antiga).

Estou à disposição caso fique com dúvidas :)

Beijos,

Julia

Coment. Natália

Oi Júlia! Muito obrigada pelos comentários!

Ainda não bati o martelo, mas estou considerando um pouco mais a proposta B. Em relação aos seus comentários sobre ela, 1) a função utilizará somente a informação (da quantidade de carboidratos) contida na tabela que estará dentro da própria função. O usuário poderá descrever os itens que deseja consumir ou pedir pra função sorteá-los. Portanto, os alimentos escolhidos ou sorteados terão que estar descritos na tabela/inseridos na função. Caso contrário a função retornaria um aviso.

O argumento info somente seria utilizado caso o usuário queira obter a informação de determinado alimento (p. ex. info=c("farelo de aveia","arroz branco"). A função retorna na tabela/data frame farelo de aveia = 1 colher de sopa = 4 g de carboidratos, arroz branco = 1 colher de sopa = 5 g de carboidratos. Apenas como informação para o usuário.

2) Achei bem bacana a ideia da função interagir com outras páginas da internet também, mas confesso que fico na dúvida por causa da complexidade... Também não sei como inserir a interação na minha função, talvez interagindo com páginas específicas de grupos específicos sobre diabetes e/ou com receitas para esse fim. Você pensou algo assim também?

Vou pensar melhor e tentar decidir o quanto antes.

Comentários Julia 16/05

Oi Natalia, Como você disse que está tendendo para a Proposta B fiquei pensando em como te ajudar para tornar mais complexa, mas sem ficar impossível.

De acordo com sua criatividade você pode utilizar outras ideias para tornar a função mais flexível e incluir ciclos nela.

Se optar por incluir a parte da receita da internet, escrevi um pseudo-tutorial que pode te ajudar. Não é um código, afinal você que terá que produzi-lo, mas te mostra um caminho a se seguir, acho que pode dar uma luz. Veja se ajuda e conforme for produzindo explore os blogs e também nos escreva se precisar!

É importante que você tome uma decisão final e comunique o quanto antes junto de sua proposta final. Chame ela de proposta C e poste aqui, por favor. OK?!

DICAS#####

rm(list=ls()) ####1 LER página
thepage1=readLines("http://www.diabetesevoce.com.br/blog/category/receitas/")

##AQUI VOCÊ TERÁ QUE ESCREVER O CÓDIGO PARA LER TODAS AS PÁGINAS DE RECEITAS DO SITE, SÃO 13 PÁGINAS. PODE FAZER ISSO DE MODO AUTOMÁTICO OU MANUAL.

####2 Procurar por uma receita com o alimento escolhido. Exemplo: chocolate.

#NO CASO DE SUA FUNÇÃO VOCÊ TERÁ QUE UTILIZAR O ARGUMENTO FORNECIDO PELO USUÁRIO. pesquisa=grep("Chocolate", thepage1) pesquisa thepage1[707] thepage1[520]

#Neste site o alimento poderá aparecer como parte do título (707) da receita ou como item nos ingredientes (520). Acho que para tornar viável, sugiro utilizar somente receitas em que o alimento esteja no título, ou seja que ele seja o protagonista.

3 Sendo assim, aqui você teria que incluir uma parte de código para considerar somente o item que contém o alimento no título. No caso, seria thepage1[707] . #Uma dica seria você checar se nesse string que a pesquisa retorna existe "http" no meio. Se houver, será o título, pois o título é um link. Se não houver será parte do texto de ingredientes. # A mesma estrutura de código você pode usar para outras páginas de receitas. E dependendo da página você pode facilitar seu código. Uma sugestão é pesquisar as páginas na internet e ver uma que possua um formato mais simples, mas com variedade de receitas. Eu coloquei essa nesse exemplo pois achei mais fácil de lidar. Mas utilize essa ou outra que você se sentir mais a vontade.

4 Agora você pode retirar o texto que será o endereço para a página da receita desejada. #No nosso caso: thepage1[707] #Queremos retirar todo o conteúdo que começa com http e que vai até /\" #Isto é, precisamos retirar a parte : "http://www.diabetesevoce.com.br/blog/sorvete-chocolate/\"

#Se você prestar atenção, esse é um padrão de critério que vale para todos os links de receitas. #Exemplo 2: pesquisa2=grep("Morango", thepage1) pesquisa2 thepage1[672] #Precisamos retirar : "http://www.diabetesevoce.com.br/blog/pave-de-morango/\"

#Se tiver dificuldade aqui, nos escreva. Mas existem muitos tutoriais nos StackOverflow e em outros blogs ensinando como cortar string de acordo com um padrão no início e um padrão no final.

5 Uma vez feito isso, você pode salvar esse pedaço do texto, em um objeto (character). E então pode utiliza-lo para a função abrir a página dessa receita para o usuário: #No caso do exemplo ficaria assim: browseURL(seu objeto , encodeIfNeeded = FALSE) #Que na prática significa: browseURL("http://www.diabetesevoce.com.br/blog/sorvete-chocolate", encodeIfNeeded = FALSE)

##

Comentário Natália 19/05

Oi Júlia! Vamos lá, vou fazer a proposta B!

Achei um bom complicador a questão de buscar a receita no site (minha primeira impressão, confesso), mas vou tentar incluir isso na minha função.

Não tinha entendido o por quê de [707] e [520] e isso tinha me confundido, mas pelo o que entendi depois de rodar o código essas são as linhas/posição em que aparece o nome do alimento na página, no caso "chocolate", é isso mesmo? No meu caso, quando busco pelo nome do objeto (pesquisa) aparecem os números 707, 715, 723, 731, mas não 520 como no seu exemplo...

Um complicador seria quando aparecer o mesmo alimento em dois títulos na mesma página, como "sorvete", por exemplo, que aparece em duas receitas. Daí talvez utilizar "http" também ajudaria, só preciso ver como agora!

Volto a escrever no caso de dúvidas (que com certeza terei)!

Segue abaixo a proposta.

Obrigada de novo!

##

Proposta C - Uma ajuda para quem necessita de uma dieta restritiva

Contextualização:

Muitas pessoas necessitam de dietas restritivas devido ao surgimento de doenças ao longo de suas vidas, tais como o diabetes, que atualmente atinge cerca de 425 milhões de pessoas em todo o mundo (1). Uma das formas de controlar os níveis sanguíneos de glicemia é através da ingestão controlada de carboidratos (2). Sabendo a quantidade de carboidratos ingeridos, torna-se mais fácil ajustar a dose da medicação, e até mesmo interromper ou retardar o seu uso em alguns casos.

A quantidade de carboidratos que podem ser ingeridos diariamente varia de pessoa para pessoa, e até mesmo em uma mesma pessoa ao longo do tempo. No que diz respeito à quantidade de carboidratos ingeridos, as dietas podem ser classificadas da seguinte maneira (2):

1. Dieta cetogênica de muito baixo carboidrato: 20-50 g/dia ou < 20% de uma dieta de 2000 kcal/dia;

2. Dieta de baixo carboidrato: < 130 g/dia ou < 26% de uma dieta de 2000 kcal/dia;

3. Dieta de ingestão moderada de carboidratos: de 26% a 40% do total de uma dieta de 2000 kcal/dia;

4. Dieta de alta ingestão de carboidratos: > 45% do total de uma dieta de 2000 kcal/dia.

A função tem como objetivo calcular as quantidades/porções de alimentos de acordo com a quantidade de carboidratos escolhida, podendo ser separado por classes de alimentos, que podem

ser escolhidos pelo usuário ou por sorteio pela função.

Função:

Entrada: caRb (alimentos = c(vetor com alimentos escolhidos), g, ref = c(“cf”, ”al”, ”jn”, ”todos”), sort, info=c(vetor com alimentos escolhidos),receita)

Argumentos:

- alimentos = alimentos escolhidos de acordo com a tabela da função (ver help) (class: vector). Classificados nos seguintes grupos: proteínas, legumes, verduras, frutas, cereais, massas, laticínios, oleaginosas, doces.
- g = quantidade de carboidratos em gramas (class: integer).
- ref = tipo de refeição, podendo escolher entre um, dois ou todos os tipos (cf, al, jn, todos).
- cf = café, al = almoço, jn = jantar
- sort = sorteia alimentos e calcula respectivas quantidades (class logical: TRUE ou FALSE).
- info = informação sobre a quantidade de carboidratos no alimento contida na tabela de alimentos da função (class: vector).
- receita = busca na página da internet receita(s) que contenha o alimento em seu título (class: character).

Verificação dos Parâmetros:

- alimentos ≠ dos listados na tabela da função? Se sim, retorna “alimento não contido na tabela!”.
- ref = (“todos”) ou (“cf”, “al”, “jn”) e g < 20? Se sim, retorna “quantidade mínima de carboidratos ingeridos deve ser de 20 g por dia!”.
- sort é TRUE ou FALSE? Se não for retorna “sort deve ser TRUE ou FALSE!”.
- sort = TRUE e alimentos = (vetor com alimentos escolhidos)? Se sim, retorna “se deseja escolher seus alimentos, sort = FALSE”).
- receita contém alimento que não está listado nos títulos da página? Retorna “alimento não encontrado!”

Pseudo-Código:

1. Se alimentos = (vetor com alimentos escolhidos), faz um sorteio e gera um data frame com as quantidades dos alimentos escolhidos.

- A. Quantidades de alimentos dos grupos “proteínas”, “verduras”, “legumes”, “frutas” e “cereais” devem ser maiores do que quantidades de alimentos dos outros grupos.
- B. Quantidade de carboidratos não deve ter diferença maior do que 20% entre grupos.

2. Se sort = TRUE, faz um sorteio dos alimentos e suas respectivas quantidades.

- A. Se ref = “cf”, sorteia e calcula quantidades de alimentos das classes “laticínios”, “pães”, “frutas”, “oleaginosas”.
- B. Se ref = “al” e/ou “jn”, sorteia e calcula quantidades de alimentos das classes “proteínas”,

"verduras", "legumes", "cereais" ou "massa".
- C. Se ref = "todos", pode sortear e calcular quantidades de todas as classes de alimentos, exceto "doces".

3. se info = (vetor com nomes dos alimentos), retorna a informação da quantidade de carboidratos em uma porção determinada do alimento.

4. se receita = (character), retorna a(s) receita(s) cujo alimento escolhido encontra-se no título.

Saída:

Um data frame com o tipo de refeição e seus alimentos com quantidades/porções que somam a quantidade de carboidrato escolhida, podendo retornar receitas com o alimento escolhido.

Referências:

1. IDF Diabetes Atlas 2017.

2. Feinman et al. 2015. Dietary carbohydrate restriction as the first approach in diabetes management: Critical review and evidence base. Nutrition, 31:1-13.

Comentários Julia

Oi Natália,

Se você conseguir incluir a busca na internet ficará legal. Ou se pensar em outra ideia, na verdade a ideia da internet surgiu mais como uma opção para tornar a função mais complexa. Mas se não, está bem. A proposta C ficou bem clara e vejo que você já esta com o código caminhando. A ideia é que a função exija um pouco de complexidade e esforço, mas não que se torne algo fora dos limites do que você julgar viável.

1. Sim, o [707] era a posição da palavra referente ao alimento. Achei curioso aparecer em outra posição para você, eu rodei em dois navegadores diferentes. Mas siga conforme rodar no seu computador. Se quiser compartilhe seu código comigo. Deixei meu e-mail no final da msg. 2. Sobre a leitura de dados no site, no caso de ter duas receitas, acho que você poderia fazer o código para utilizar somente a primeira opção que aparecer. Já seria ótimo.

Qualquer coisa meu e-mail é juliambmolina@gmail.com

Bjs!

Comentários sobre a função

Por diversos motivos (que fugiram um pouco do meu alcance) não consegui colocar a interação com paginas da internet, como havia conversado com a Julia. Acredito que se houvesse um pouco mais de tempo (que foi o meu principal empecilho) eu conseguiria incluir esse argumento na função. De resto, tentei seguir ao máximo o que propus desde o inicio. O argumento sort foi excluido, uma vez que percebi que se a pessoa escolher o alimento que deseja não faria sentido a função sortear os alimentos também.

Código da função

```
##Função caRb

caRb<-function(alimento=TRUE, g, ref=FALSE) #definir argumentos da função
{
  if(missing(g)) #conferir se argumento g (quantidade de carboidrato) foi
colocado na função
  {stop("definir quantidade de carboidrato!")} #se não foi, para e retorna
um aviso
  if(class(g) != "numeric") #conferir se g é da classe numérica
  {stop("g deve ser numérico ou inteiro!")} #se não for, para e retorna um
aviso
  if(g > 225) # se o valor de g > 225
    warning("Cuidado! Quantidade de carboidratos muito alta!") #retorna um
aviso, mas não para a função
  if(g < 10) #se valor de g < 10
    warning("Atenção! Quantidade de carboidrato baixa!") #retorna um aviso,
mas não para a função
  if(class(alimento) == "character" & ref == "café") #se usuário escolher
alimentos e tipo de refeição
  {stop("Escolha o alimento ou o tipo de refeição!")} #para e retorna um
aviso
  if (class(alimento) == "character" & ref == "almoço") #se usuário escolher
alimentos e tipo de refeição
  {stop("Escolha o alimento ou o tipo de refeição!")} #para e retorna um
aviso
  if(class(alimento) == "character" & ref == "almoço veg") #se usuário
escolher alimentos e tipo de refeição
  {stop("Escolha o alimento ou o tipo  de refeição!")} #para e retorna um
aviso
  if(class(alimento) == "character" & ref == "janta") #se usuário escolher
alimentos e tipo de refeição
  {stop("Escolha o alimento ou o tipo de refeição!")} #para e retorna um
aviso
  if(class(alimento) == "character" & ref == "janta veg") #se usuário
escolher alimentos e tipo de refeição
  {stop("Escolha o alimento ou o tipo de refeição!")} #para e retorna um
aviso
  #inserir vetores com os nomes dos alimentos, unidades de medida, peso,
quantidade de carboidrato e classe dos alimentos,
  #que em seguida serão transformados em um data frame.
```

```
  {
    #vetor com nomes dos alimentos
    Alimento=c("abacate","abacaxi","abacaxi em
calda","abóbora","abobrinha","açaí com guaraná","acarajé","acerola","açúcar
refinado","água de coco","alcachofra","alfajor","almôndega","ameixa seca",
            "ameixa vermelha","amendoim caramelizado","amendoim torrado com
sal","amora","arroz branco","arroz-doce","arroz integral","aveia em
flocos","banana-maçã","banana-ouro",
            "banana-prata","banana à milanesa","banana-passa","batata
cozida","batata assada","batata frita","batata-doce assada","batata-doce
cozida","batata-doce frita","beijinho","beterraba cozida","bife à
milanesa","biscoito água e sal",
            "biscoito aveia e mel","biscoito champanhe","biscoito de
coco","biscoito cream cracker","biscoito de polvilho","biscoito
maizena","Passatempo recheado","Passatempo sem recheio",
            "biscoito recheado","rosquinha de coco","biscoito
wafer","bolinho de arroz frito","bolinha de queijo","bolinho de
bacalhau","bobó de camarão","bolo com glacê","bolo de banana","bolo de
cenoura",
            "bolo de fubá","bolo de milho","bolo de tapioca","bolo de
chocolate","brigadeiro","broa de fubá","broa de milho","cacau em pó","café
sem açúcar","caju","cajuzinho","caldo-de-cana","canjica",
            "caqui","carambola(s)","castanha de caju","castanha da
amazônia","castanha portuguesa","ketchup","cenoura
cozida","granola","cerveja","chá sem
açúcar","champanhe","chantili","chocolate em pó",
            "chocolate Alpino","chocolate ao leite","chocolate ao leite
diet","chocolate Batom","chocolate Bis","chocolate Charge","chocolate
Chokito","chocolate Confete","chocolate Crunch","chocolate Diamante Negro",
            "chocolate Galak","chocolate Kinder Ovo","chocolate meio
amargo","chocolate Milkbar","chocolate Nescau","chocolate
Prestígio","chocolate Sensação","chocolate Serenata de Amor","chocolate
Sonho de Valsa",
            "chocolate Stickadinho","chocolate Suflair","chocolate
Talento","chocolate Talento diet","chocolate Twix","chocotone","chuchu
cozido","coalhada","cocada","coco ralado","couve-flor à milanesa",
            "couve-flor cozida","couve refogada","coxinha","creme de
espinafre","creme de leite","creme de
milho","croissant","croquete","curau","cuscuz paulista","damasco seco","doce
de abóbora com coco",
            "doce de batata-doce","doce de coco","doce de goiaba","doce de
leite","doce de mamão","empada","empadão","enrolado de salsicha","ervilha
enlata(s)da","ervilha torta cozida","esfiha de carne",
            "esfiha de queijo","farelo de aveia","farelo de trigo","farinha
de arroz","farinha láctea","farinha de mandioca","farinha de milho","farinha
de rosca","farinha de trigo","fécula de batata",
            "feijão branco cozido","feijão cozido","figo","figo
cristalizado","figo em calda","figo seco","filé à milanesa","pinha","frutas
cristalizadas","fubá","Gatorade","gelatina diet","gelatina",
            "geleia de amora","geleia de damasco","geleia de
```

```
framboesa","gemada","goiaba","goiabada","goiabada light","grão-de-bico
cozido","homus","iogurte com frutas","iogurte com frutas light",
            "iogurte com mel","Danette","Danoninho","iogurte natural
desnatado","iogurte natural integral","jabuticaba","jaca","jiló
cozido","Karo","kibe assado","kibe cru","kibe frito","kiwi","laranja",
            "laranja-lima","lasanha à bolonhesa","leite de cabra","leite de
coco","leite condensado","leite condensado light","leite de soja
integral","leite de soja light","leite de vaca desnatado",
            "leite de vaca integral","leite de vaca semidesnatado","lentilha
cozida","limão","maçã","macarrão cozido","maisena","mamão formosa","mamão
papaia","mandioca cozida","mandioca frita","inhame cozido",
            "mandioquinha","manga","manjar","maracujá","maria-
mole","marmelada","marshmallow","massa de
pastel","mel","melancia","melão","merengue","milho cozido","milho verde
enlata(s)do","milk-shake de chocolate",
            "mini pizza","miojo","misto-quente","morango","musse de
chocolate","musse de maracujá","Mucilon de arroz","Mucilon de milho","nabo
cozido","nectarina","achocolata(s)do em pó light","achocolata(s)do em pó",
            "nêspera","Neston aveia","Neston
vitamina","nhoque","nozes","nuggets de frango","nuggets de peixe","nuggets
de legumes","Nutella","olho-de-sogra","ovinhos de
amendoim","paçoca","palmito em conserva",
            "pamonha","panetone","panqueca de carne","panqueca de
frango","pão baguete","pão ciabatta","pão de batata","pão de centeio","pão
de forma","pão de forma light","pão de hamburguer",
            "pão de cachorro quente","pão de leite","pão de mel","pão de
milho","pão de queijo","pão doce recheado","pão doce simples","pão
francês","pão italiano","pão sírio","pão sovado","pastel assado",
            "pastel de feira","pastel português","pavê de chocolate","pavê
de nozes","pé-de-moleque","pepino","pêra","pêssego","pêssego em
calda","pimentão cozido","pinhão cozido","pipoca","pirão de farinha de
mandioca",
            "pirulito","pitanga","pizza","polenta","polvilho","pudim de
leite condensado","pudim de pão com passas","purê de batata","queijadinha de
coco","quiabo cozido","quiche de queijo","quindim","rabanada",
            "rabanete
cru","rapadura","ravioli","refrigerante","risole","risoto de frango","risoto
milanês","romã","sagu em vinho","salada de frutas","salpicão de
frango","salsichão","sanduíche natural","sequilho",
            "shoyu","soja cozida","sopa creme de cebola(s)","sopa creme de
cogumelo","sopa creme de espinafre","sopa creme de palmito","sopa de
ervilha","sopa de feijão","sopa de frango","sopa de legumes com carne",
            "sopa de lentilha","sopa de macarrão","sorvete de
massa","sorvete de massa light","picolé de brigadeiro","picolé de
chocolate","picolé de coco","picolé de frutas","frozen yogurt","frozen
yogurt diet",
            "suco de abacaxi sem açúcar","suco de acerola sem açúcar","suco
de caju sem açúcar","suco de laranja sem açúcar","suco de maçã sem
açúcar","suco de melancia sem açúcar","suco de morango sem açúcar",
            "suco de pêssego sem açúcar","suco de tomate","suco de
uva","suflê de espinafre","suflê de legumes","suflê de
```

```
queijo","sushi","suspiro","tabule","tangerina","tapioca","tomate","torrada",
            "trigo cozido","torta de liquidificador","torta de
morango","tutu de feijão","uva","uva itália","uva passa","vagem
cozida","vatapá","Yakult")
    #vetor com os números das unidades de medida.
    numero_medida<-
c(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1/2,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1/2,1,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1)
    #vetor com as unidades de medida.
    medida<-c("colher(es) de sopa","fatia(s) média(s)","fatia(s)
média(s)","colher(es) de sopa","colher(es) de sopa","porção(es)","unidade(s)
média(s)","unidade(s) média(s)","colher(es) de
sopa","copo(s)","unidade(s)","unidade(s) média(s)","unidade(s)
média(s)",
              "unidade(s) média(s)","pacote(s)","colher(es) de
sopa","unidade(s) média(s)","colher(es) de sopa","colher(es) de
sopa","colher(es) de sopa","colher(es) de sopa","unidade(s)
média(s)","unidade(s) média(s)","unidade(s) média(s)","unidade(s) média(s)",
              "unidade(s)","colher(es) de sopa","colher(es) de
sopa","colher(es) de sopa","colher(es) de sopa","colher(es) de
sopa","fatia(s) pequena(s)","unidade(s)","colher(es) de sopa","unidade(s)
média(s)","unidade(s)","unidade(s)","unidade(s)","unidade(s)","unidade(s)",
"unidade(s)","unidade(s)","unidade(s)","unidade(s)","unidade(s)","unidade(s)
","unidade(s)","unidade(s) média(s)","unidade(s) pequena(s)","unidade(s)
grande(s)","colher(es) de sopa","fatia(s) média(s)","fatia(s)
média(s)","fatia(s) média(s)","fatia(s) média(s)","fatia(s) média(s)",
              "fatia(s) média(s)","fatia(s) média(s)","unidade(s)
média(s)","fatia(s) média(s)","fatia(s) média(s)","colher(es) de
sopa","copo(s)","unidade(s) média(s)","unidade(s)
média(s)","copo(s)","colher(es) de sopa","unidade(s) pequena(s)","unidade(s)
média(s)","unidade(s)","unidade(s)",
              "unidade(s)","colher(es) de sopa","colher(es) de
sopa","xícara(s) de chá","lata(s)","xícara(s) de chá","taça(s)","colher(es)
de sopa","colher(es) de
sopa","unidade(s)","unidade(s)","unidade(s)","unidade(s)","unidade(s)","unid
ade(s)","unidade(s)","porção(es)",
"unidade(s)","unidade(s)","unidade(s)","unidade(s)","unidade(s)","unidade(s)
","unidade(s)","unidade(s)","unidade(s)","unidade(s)","unidade(s)","unidade(
```

```
s)","unidade(s)","unidade(s)","unidade(s)","unidade(s)","fatia(s)
média(s)","colher(es) de sopa",
               "colher(es) de sopa","unidade(s) média(s)","colher(es) de
sopa","ramo(s) médio(s)","ramo(s) médio(s)","colher(es) de sopa","unidade(s)
média(s)","colher(es) de sopa","colher(es) de sopa","colher(es) de
sopa","unidade(s) média(s)","unidade(s) grande(s)","porção(es) pequena(s)",
               "fatia(s) média(s)","unidade(s)","colher(es) de
sopa","colher(es) de sopa","colher(es) de sopa","colher(es) de
sopa","colher(es) de sopa","colher(es) de sopa","unidade(s)
média(s)","fatia(s) média(s)","unidade(s) média(s)","colher(es) de
sopa","colher(es) de sopa",
               "unidade(s) média(s)","unidade(s) média(s)","colher(es) de
sopa","colher(es) de sopa","colher(es) de sopa","colher(es) de
sopa","colher(es) de sopa","colher(es) de sopa","colher(es) de
sopa","colher(es) de sopa","colher(es) de sopa","colher(es) de
sopa","colher(es) de sopa",
               "unidade(s) grande(s)","unidade(s) média(s)","unidade(s)
média(s)","unidade(s) média(s)","filé médio","unidade(s)
média(s)","colher(es) de sopa","colher(es) de
sopa","copo(s)","porção(es)","porção(es)","colher(es) de sopa","colher(es)
de sopa","colher(es) de sopa",
               "colher(es) de sopa","unidade(s) média(s)","fatia(s)
pequena(s)","fatia(s) pequena(s)","colher(es) de sopa","colher(es) de
sopa","unidade(s)","unidade(s)","copo(s)","potinho","unidade(s)","copo(s)","
copo(s)","unidade(s)","bago(s)","colher(es) de sopa","colher(es) de sopa",
               "porção(es)","porção(es)","unidade(s) média(s)","unidade(s)
média(s)","unidade(s) média(s)","unidade(s) média(s)","pedaço(s)
médio(s)","copo(s)","copo(s)","colher(es) de sopa","colher(es) de
sopa","copo(s)","copo(s)","copo(s)","copo(s)","copo(s)","colher(es) de
sopa","colher(es) de sopa","unidade(s) média(s)",
               "pegador(es)","colher(es) de sopa","fatia(s)
média(s)","unidade(s) média(s)","colher(es) de sopa","pedaço(s)
médio(s)","colher(es) de sopa","colher(es) de sopa","unidade(s)
média(s)","porção(es)","unidade(s) média(s)","porção(es)","fatia(s)
pequena(s)","colher(es) de sopa",
               "unidade(s) média(s)","colher(es) de sopa","fatia(s)
média(s)","fatia(s) média(s)","colher(es) de sopa","espiga(s)
grande(s)","colher(es) de sopa","copo(s) grande(s)","unidade(s)
média(s)","pacote(s)","unidade(s)","unidade(s) média(s)","colher(es) de
sopa","colher(es) de sopa","colher(es) de sopa",
               "colher(es) de sopa","colher(es) de sopa","unidade(s)
média(s)","colher(es) de sopa","colher(es) de sopa","unidade(s)
grande(s)","colher(es) de sopa","colher(es) de
sopa","escumadeira(s)","unidade(s)
média(s)","unidade(s)","unidade(s)","unidade(s)","colher(es) de sopa",
               "unidade(s) média(s)","porção(es)","unidade(s)","colher(es)
de sopa","unidade(s)","fatia(s)
pequena(s)","unidade(s)","unidade(s)","unidade(s) média(s)","unidade(s)
média(s)","unidade(s)
média(s)","fatia(s)","fatia(s)","fatia(s)","unidade(s)","unidade(s)","unidad
e(s)","unidade(s) pequena(s)","fatia(s)",
```

```
                "unidade(s)
média(s)","unidade(s)","unidade(s)","unidade(s)","fatia(s)
média(s)","unidade(s) média(s)","fatia(s)","unidade(s)
média(s)","unidade(s)","unidade(s) média(s)","colher(es) de
sopa","colher(es) de sopa","unidade(s) média(s)","colher(es) de sopa",
                "unidade(s) média(s)","unidade(s) média(s)","colher(es) de
sopa","colher(es) de sopa","unidade(s)","saco médio","colher(es) de
sopa","unidade(s)","unidade(s)","fatia(s) média(s)","colher(es) de
sopa","colher(es) de sopa","fatia(s) média(s)","fatia(s)
média(s)","colher(es) de sopa",
                "unidade(s) média(s)","colher(es) de sopa","unidade(s)
média(s)","unidade(s) média(s)","unidade(s) média(s)","colher(es) de
sopa","pedaço(s) médio(s)","escumadeira(s)","copo(s)","unidade(s)
média(s)","colher(es) de sopa","colher(es) de sopa","unidade(s)
média(s)","colher(es) de sopa",
                "colher(es) de sopa","colher(es) de sopa","unidade(s)
média(s)","unidade(s) média(s)","unidade(s) pequena(s)","colher(es) de
sopa","colher(es) de sopa","concha(s) média(s)","concha(s)
média(s)","concha(s) média(s)","concha(s) média(s)","concha(s)
média(s)","concha(s) média(s)","concha(s) média(s)",
                "concha(s) média(s)","concha(s) média(s)","concha(s)
média(s)","bola(s)","bola(s)","unidade(s)","unidade(s)","unidade(s)","unidad
e(s)","porção(es)","porção(es)","copo(s)","copo(s)","copo(s)","copo(s)","cop
o(s)","copo(s)","copo(s)","copo(s)","copo(s)","copo(s)","colher(es) de
sopa",
                "colher(es) de sopa","pedaço(s) médio(s)","unidade(s)
média(s)","unidade(s) média(s)","colher(es) de sopa","unidade(s)
média(s)","colher(es) de sopa","fatia(s) média(s)","unidade(s)","colher(es)
de sopa","fatia(s) pequena(s)","fatia(s) pequena(s)","colher(es) de
sopa","gomo(s)","gomo(s)",
                "colher(es) de sopa","colher(es) de
sopa","unidade(s)","unidade(s)")
    #vetor com os pesos equivalentes às unidades de medida.
    unidade<-
c(45,75,75,36,30,200,100,12,15,200,100,50,30,5,16,20,17,8,20,40,20,15,65,40,
40,45,17,30,30,30,42,42,30,15,20,80,8,8,8,8,7,3,5,15,6,13,10,10,40,10,60,28,
60,70,60,60,60,80,60,15,60,60,16,50,50,15,200,25,85,60,2.5,6,10,15,25,40,350
,
200,100,15,15,13,30,30,16,7,40,32,30,24,30,30,20,50,28,40,33,38,20,21,12,50,
25,25,16,40,20,20,70,9,90,60,20,50,35,15,35,40,55,100,100,7,40,40,50,50,40,5
0,55,110,27,30,25,60,60,9,9,17,20,16,15,15,20,15,17,17,70,55,50,28,120,60,
20,20,200,25,25,20,17,17,15,170,40,40,22,30,200,200,185,110,45,185,185,5,12,
60,15,50,50,50,76,180,90,190,200,240,15,15,200,200,240,240,240,18,15,100,110
,20,170,155,30,120,30,15,140,90,45,44,40,40,17,15,200,100,25,100,24,300,
100,90,85,12,30,30,9,9,35,100,16,25,40,20,20,100,5,23,23,23,25,20,30,30,15,1
00,40,60,60,100,50,50,25,25,25,70,58,50,15,50,20,70,50,50,50,60,40,25,100,35
,37,37,20,18,110,60,30,13,10,20,30,5,15,120,15,16,80,80,30,35,40,138,35,60,
35,55,50,200,35,25,25,50,20,38,25,100,120,3,12,17,130,130,130,130,130,130,13
0,130,130,130,100,100,65,65,65,65,200,200,200,200,200,200,200,200,200,200,20
0,200,55,55,90,22,10,40,135,35,15,8,25,30,60,35,8,8,18,20,100,80)
```

```
    #vetor com as quantidades de carboidrato equivalentes às unidades de
medida.
    carboidrato<-
c(3,10,22,2,1,35,23,1,15,10,11,33,1,2,2,15,3,1,5,13,3,9,17,9,9,11,14,6,6,6,1
0,10,18,8,2,6,5,5,6,6,5,2,4,10,4,8,6,7,15,3,14,6,37,33,38,25,33,48,30,9,30,3
0,3,0,5,7,40,5,17,5,1,1,5,5,3,32,13,0,12,2,7,8,17,16,9,5,24,25,24,14,19,15,1
1,28,19,
23,22,7,12,13,8,30,13,12,9,23,2,1,37,1,11,3,2,18,4,0.5,26,19,21,16,23,4,18,2
4,29,21,22,28,18,37,3,3,4,23,15,4,5,14,15,14,12,11,15,12,4,3,11,41,20,18,8,1
5,16,15,12,0,3,11,11,12,5,19,26,5,5,14,31,11,31,25,9,12,12,0.6,2,5,11,8,9,11
,
11,20,9,30,10,7,8,9,7,1,12,12,12,3,1,15,24,16,14,13,9,29,9,3,24,21,10,26,30,
30,5,12,12,7,23,28,5,60,27,59,29,1,9,10,8,8,2,12,13,21,4,15,18,21,1,3,4,7,3,
10,13,20,0.5,32,22,9,9,57,24,29,15,14,11,40,31,30,10,29,7,30,28,28,28,34,23,
10,
30,10,7,14,14,1,16,7,6,0.5,0.6,11,9,5,1,24,8,14,21,22,5,17,3,37,14,21,1,48,2
0,22,10,6,7,8,18,5,2,3,29,2,1,2,6,12,5,11,20,18,3,8,16,20,23,22,19,19,16,15,
61,24,10,5,3,22,19,12,8,8,8,30,1,5,5,15,9,7,15,29,1,5,6,10,22,6,1,1,13,2,9,1
3)
    #vetor com as classes dos respectivos alimentos
    Classe<-
c("fruta","fruta","doce","vegetal","vegetal","doce","outro","fruta","doce","
bebida","vegetal","doce","proteina","seca","fruta","doce",
"outro","fruta","cereal","doce","cereal","cereal.matinal","fruta","fruta","f
ruta","outro","doce","vegetal","vegetal","outro",
             "vegetal","vegetal","outro","doce",
"vegetal","proteina","biscoito","biscoito","biscoito","biscoito","biscoito",
"biscoito",
"biscoito","biscoito","biscoito","biscoito","biscoito","biscoito","salgado",
"salgado","salgado","prato","doce","doce","doce",
"doce","doce","doce","doce","doce","doce","doce","outro","bebida","fruta","d
oce","bebida","doce","fruta","fruta","castanha",
"castanha","castanha","tempero","vegetal","cereal.matinal","bebida.alcool","
bebida","bebida.alcool","doce","doce","doce","doce",
"doce","doce","doce","doce","doce","doce","doce","doce","doce","doce","doce"
,"doce","doce","doce","doce","doce","doce","doce",
"doce","doce","doce","doce","doce","vegetal","laticinio","doce","fruta","veg
etal","vegetal","vegetal","salgado","prato","laticinio",
"prato","pao","salgado","doce","prato","seca","doce","doce","doce","doce","d
oce","doce","salgado","salgado","salgado","leguminosa",
"vegetal","salgado","salgado","cereal.matinal","farinha","farinha","farinha"
,"farinha","farinha","farinha","farinha","farinha",
"leguminosa","leguminosa","fruta","doce","doce","seca","proteina","fruta","s
eca","farinha","bebida","doce","doce","doce","doce",
"doce","bebida","fruta","doce","doce","leguminosa","prato","laticinio","lati
cinio","laticinio","laticinio","laticinio","laticinio",
"laticinio","fruta","fruta","vegetal","adocante","prato","prato","salgado","
fruta","fruta","fruta","prato","laticinio","outro",
"doce","doce","bebida","bebida","laticinio","laticinio","laticinio","legumin
osa","fruta","fruta","prato","farinha","fruta","fruta",
"vegetal","outro","vegetal","vegetal","fruta","doce","fruta","doce","doce","
```

```
doce","salgado","adocante","fruta","fruta","doce",
"vegetal","vegetal","doce","salgado","outro","salgado","fruta","doce","doce"
,"farinha","farinha","vegetal","fruta","outro","outro",
"fruta","farinha","farinha","prato","castanha","outro","outro","outro","doce
","doce","salgado","doce","vegetal","doce","doce","prato",
"prato","pao","pao","pao","pao","pao","pao","pao","pao","pao","doce","pao","
salgado","doce","doce","pao","pao","pao","pao","salgado",
"salgado","doce","doce","doce","doce","vegetal","fruta","fruta","doce","vege
tal","castanha","outro","prato","doce","fruta","salgado",
"prato","farinha","doce","doce","prato","doce","vegetal","salgado","doce","d
oce","vegetal","doce","prato","bebida","prato","prato",
"prato","fruta","doce","fruta","prato","salgado","salgado","doce","tempero",
"leguminosa","prato","prato","prato","prato","prato",
"prato","prato","prato","prato","prato","doce","doce","doce","doce","doce","
doce","doce","doce","bebida","bebida","bebida","bebida",
"bebida","bebida","bebida","bebida","bebida","bebida","prato","prato","prato
","prato","doce","prato","fruta","farinha","vegetal",
"pao","cereal","prato","doce","prato","fruta","fruta","seca","vegetal","prat
o","bebida")
    tabela<-
data.frame(Alimento,Classe,numero_medida,medida,unidade,carboidrato) #criar
data frame com informações dos alimentos
    }
  if (ref == FALSE){ #condição para a função retornar o cálculo dos
alimentos escolhidos no argumento "alimento"
    al<-tabela[tabela$Alimento %in% alimento, ] #ler os caracteres listados
no argumento "alimento" e que estão contidos na tabela da função
    carb<-g/length(alimento) #definir a quantidade de carboidrato em cada
alimento escolhido, em proporção(es) igual para cada alimento (quantidade
(g)/numero de alimentos)
    numero<-vector() #criar vetor vazio para colocar o resultado do for para
o cálculo da nova medida do alimento
    peso<-vector() #criar vetor vazio para colocar resultado do for para o
cálculo do novo peso do alimento
    for(i in 1:nrow(al)){ #calcular valores para todas as linhas do data
frame criado no passo anterior
      numero[i]<-((carb*al[i,"numero_medida"])/al[i,"carboidrato"])
#calcular novo valor de medida do alimento
      peso[i]<-((carb*al[i,"unidade"])/al[i,"carboidrato"]) #calcular novo
peso do alimento
    }
    result<-al[["carboidrato"]]<-round(carb) #substituir a coluna
"carboidrato" pelo novo cálculo de carboidrato arredondado
    colnames(al)[colnames(al)=="carboidrato"]<-"carboidrato(g)" #dar nome à
coluna
    result2<-al[["numero_medida"]]<-round(numero,1) #substituir a coluna
"numero_medida" pelo novo valor, arredondando para uma casa decimal
    colnames(al)[colnames(al)=="numero_medida"]<-" " #dar nome à coluna
    result3<-al[["unidade"]]<-round(peso) #substituir a coluna "unidade"
pelo novo valor do peso arredondado
```

```
    colnames(al)[colnames(al)=="unidade"]<-"peso(g)/volume(ml)" #dar nome à
coluna
  }
  else{ #se condição acima for falsa (ref=TRUE)
    if(ref == "café")  { #condição se ref = café
      itens=4 #estabelecer o número de itens da refeição
      carb<-g/itens #estabelecer a quantidade de carboidrato em cada item
(dividir g igualmente entre alimentos)
      al1<-sample((tabela[Classe == "fruta", "Alimento"]),1) #fazer o
sorteio do alimento da classe "fruta"
      al1.1<-tabela[tabela$Alimento %in% al1, ] #separar a linha com o
alimento sorteado
      al2<-sample((tabela[Classe == "pao", "Alimento"]),1) #fazer sorteio do
alimento da classe "pão"
      al2.1<-tabela[tabela$Alimento %in% al2, ] #separar a linha com o
alimento sorteado
      al3<-sample((tabela[Classe == "laticinio", "Alimento"]),1) #fazer o
sorteio do alimento da classe "laticinio"
      al3.1<-tabela[tabela$Alimento %in% al3, ] #separar a linha com o
alimento sorteado
      al4<-sample((tabela[Classe == "cereal.matinal", "Alimento"]),1) #fazer
o sorteio do alimento da classe "cereal.matinal"
      al4.1<-tabela[tabela$Alimento %in% al4, ] #separar a linha com o
alimento sorteado
      al<-rbind(al1.1, al2.1, al3.1, al4.1) #juntar linhas dos alimentos
sorteados em um novo data frame
      numero<-vector() #criar vetor vazio para colocar o resultado do for
para o cálculo da nova medida do alimento
      peso<-vector() #criar vetor vazio para colocar resultado do for para o
cálculo do novo peso do alimento
      for(i in 1:nrow(al)){ #calcular valores para todas as linhas do data
frame criado no passo anterior
        numero[i]<-((carb*al[i,"numero_medida"])/al[i,"carboidrato"])
#calcular novo valor de medida do alimento
        peso[i]<-((carb*al[i,"unidade"])/al[i,"carboidrato"]) #calcular novo
peso do alimento
      }
      result<-al[["carboidrato"]]<-round(carb) #substituir a coluna
"carboidrato" pelo novo cálculo de carboidrato arredondado
      colnames(al)[colnames(al)=="carboidrato"]<-"carboidrato(g)" #dar nome
à coluna
      result2<-al[["numero_medida"]]<-round(numero,1) #substituir a coluna
"numero_medida" pelo novo valor, arredondando para uma casa decimal
      colnames(al)[colnames(al)=="numero_medida"]<-" " #dar nome à coluna
      result3<-al[["unidade"]]<-round(peso) #substituir a coluna "unidade"
pelo novo valor do peso arredondado
      colnames(al)[colnames(al)=="unidade"]<-"peso(g)/volume(ml)" #dar nome
à coluna
                          }
    if(ref == "almoço" | ref == "janta"){ #condição se ref = "almoço" ou ref
= "janta"
```

```
      itens=4 #estabelecer o número de itens da refeição
      carb<-g/itens #estabelecer a quantidade de carboidrato em cada item
(dividir g igualmente entre alimentos)
      al1<-sample((tabela[Classe == "cereal", "Alimento"]),1) #fazer o
sorteio do alimento da classe "cereal"
      al1.1<-tabela[tabela$Alimento %in% al1, ] #separar a linha com o
alimento sorteado
      al2<-sample((tabela[Classe == "leguminosa", "Alimento"]),1) #fazer o
sorteio do alimento da classe "leguminosa"
      al2.1<-tabela[tabela$Alimento %in% al2, ] #separar a linha com o
alimento sorteado
      al3<-sample((tabela[Classe == "vegetal", "Alimento"]),1) #fazer o
sorteio do alimento da classe "vegetal"
      al3.1<-tabela[tabela$Alimento %in% al3, ] #separar a linha com o
alimento sorteado
      al4<-sample((tabela[Classe == "proteina", "Alimento"]),1) #fazer o
sorteio do alimento da classe "proteina"
      al4.1<-tabela[tabela$Alimento %in% al4, ] #separar a linha com o
alimento sorteado
      al<-rbind(al1.1, al2.1, al3.1, al4.1) #juntar linhas dos alimentos
sorteados em um novo data frame
      numero<-vector() #criar vetor vazio para colocar o resultado do for
para o cálculo da nova medida do alimento
      peso<-vector() #criar vetor vazio para colocar resultado do for para o
cálculo do novo peso do alimento
      for(i in 1:nrow(al)){ #calcular valores para todas as linhas do data
frame criado no passo anterior
        numero[i]<-((carb*al[i,"numero_medida"])/al[i,"carboidrato"])
#calcular novo valor de medida do alimento
        peso[i]<-((carb*al[i,"unidade"])/al[i,"carboidrato"]) #calcular novo
peso do alimento
      }
      result<-al[["carboidrato"]]<-round(carb) #substituir a coluna
"carboidrato" pelo novo cálculo de carboidrato arredondado
      colnames(al)[colnames(al)=="carboidrato"]<-"carboidrato(g)" #dar nome
à coluna
      result2<-al[["numero_medida"]]<-round(numero,1) #substituir a coluna
"numero_medida" pelo novo valor, arredondando para uma casa decimal
      colnames(al)[colnames(al)=="numero_medida"]<-" " #dar nome à coluna
      result3<-al[["unidade"]]<-round(peso) #substituir a coluna "unidade"
pelo novo valor do peso arredondado
      colnames(al)[colnames(al)=="unidade"]<-"peso(g)/volume(ml)" #dar nome
à coluna
    }
    if(ref == "almoço veg" | ref == "janta veg"){ #condição se ref = "almoço
veg" ou ref = "janta veg"
      itens=4 #estabelecer o número de itens da refeição
      carb<-g/itens #estabelecer a quantidade de carboidrato em cada item
(dividir g igualmente entre alimentos)
      al1<-sample((tabela[Classe == "cereal", "Alimento"]),1) #fazer o
```

```
sorteio do alimento da classe "cereal"
      al1.1<-tabela[tabela$Alimento %in% al1, ] #separar a linha com o
alimento sorteado
      al2<-sample((tabela[Classe == "leguminosa", "Alimento"]),1) #fazer o
sorteio do alimento da classe "leguminosa"
      al2.1<-tabela[tabela$Alimento %in% al2, ] #separar a linha com o
alimento sorteado
      al3<-sample((tabela[Classe == "vegetal", "Alimento"]),1) #fazer o
sorteio do alimento da classe "vegetal"
      al3.1<-tabela[tabela$Alimento %in% al3, ] #separar a linha com o
alimento sorteado
      al4<-sample((tabela[Classe == "vegetal", "Alimento"]),1) #fazer o
sorteio do alimento da classe "vegetal"
      al4.1<-tabela[tabela$Alimento %in% al4, ] #separar a linha com o
alimento sorteado
      al<-rbind(al1.1, al2.1, al3.1, al4.1) #juntar linhas dos alimentos
sorteados em um novo data frame
      numero<-vector() #criar vetor vazio para colocar o resultado do for
para o cálculo da nova medida do alimento
      peso<-vector() #criar vetor vazio para colocar resultado do for para o
cálculo do novo peso do alimento
      for(i in 1:nrow(al)){ #calcular valores para todas as linhas do data
frame criado no passo anterior
        numero[i]<-((carb*al[i,"numero_medida"])/al[i,"carboidrato"])
#calcular novo valor de medida do alimento
        peso[i]<-((carb*al[i,"unidade"])/al[i,"carboidrato"]) #calcular novo
peso do alimento
                            }
      result<-al[["carboidrato"]]<-round(carb) #substituir a coluna
"carboidrato" pelo novo cálculo de carboidrato arredondado
      colnames(al)[colnames(al)=="carboidrato"]<-"carboidrato(g)" #dar nome
à coluna
      result2<-al[["numero_medida"]]<-round(numero,1) #substituir a coluna
"numero_medida" pelo novo valor, arredondando para uma casa decimal
      colnames(al)[colnames(al)=="numero_medida"]<-" " #dar nome à coluna
      result3<-al[["unidade"]]<-round(peso) #substituir a coluna "unidade"
pelo novo valor do peso arredondado
      colnames(al)[colnames(al)=="unidade"]<-"peso(g)/volume(ml)" #dar nome
à coluna
                                                }
  }
  return(al) #retorna o data frame al
}
```

Help da Função

```
caRb                    package:unknown                R documentation
```

```
Description:

caRb calcula as porções de alimentos conforme a quantidade de carboidrato
escolhida. Alimentos são
escolhidos pelo usuário ou sorteados pela função de acordo com o tipo de
refeição escolhida.
Produz um data frame com o nome do alimento, sua categoria, porção, peso e
quantidade de carboidratos.


Usage:

caRb (alimento = TRUE, g, ref = FALSE)


Arguments:

alimento:  vetor com um ou mais nomes de alimentos escolhidos pelo usuário.
Alimentos devem estar contidos
na tabela da função.

g: número positivo que indica a quantidade de carboidrato (em gramas) que o
usuário deseja consumir.
Argumento da classe numérica ou inteira.

ref: refeição escolhida pelo usuário. Pode ser “café”, “almoço”, “almoço
veg”, “janta”. Usuário deve
escolher apenas um tipo de refeição.


Details:

Se alimento = TRUE, retorna um data frame com as porções dos alimentos
escolhidos e seus respectivos pesos
e quantidades de carboidrato. A função também retorna a classificação do
alimento (“bebida”,  “biscoito”,
“castanha”, “cereal”, “cereal matinal”, “doce”, “farinha”, “fruta”,
“laticínio”, “leguminosa”, “prato”
(pratos elaborados, ex: panqueca de carne, lasanha), “proteína” (proteína
animal), “salgado”, ”seca”
(frutas secas e/ou passas), “tempero”, “vegetal”, “outro” (alimentos que não
se enquadram nas classes
descritas acima). Nomes dos alimentos devem estar contidos na tabela da
função, que contém 356 itens.

Valor de g é dividido igualmente pelo número de itens escolhidos quando
alimento = TRUE ou pelo número  de
itens de cada refeição (n = 4) quando ref = TRUE.

Se ref = “café”, a função irá sortear e calcular porções dos alimentos das
```

```
classes “fruta”, “pão”, “bebida”
e “cereal matinal”. Se ref = “almoço” ou ref = “janta”, a função irá sortear
e calcular porções dos
alimentos das classes “cereal”, “leguminosa”, “vegetal” e “proteína”. Se ref
= “almoço veg” ou ref = “janta
veg”, a função irá excluir o alimento da classe animal e substituir por um
alimento da classe “vegetal”.

Porções e medidas dos alimentos são arredondadas para uma casa decimal. Peso
do alimento a ser consumido e
quantidade de carboidrato são arredondados para um número inteiro. Devido ao
arredondamento, pode haver
pequena variação em relação ao valor de “g” fornecido pelo usuário.

Alguns alimentos podem retornar medida = Inf, o que significa que seu valor
é muito pequeno. Tente
recalcular inserindo um valor de g maior.


Warnings:

g precisa ser inserido na função. Se ausente, a função para e retorna um
aviso ao usuário.

Se g > 225, a função retornará um aviso (“Cuidado!  Quantidade de
carboidrato muito alta!”). Se g < 10 a
função retornará um aviso (“Atenção! Quantidade de  carboidrato baixa!”).


Author:

Natália Targhetta
natalia.targhetta@usp.br

References:

1. Comida que Cuida 2. O prazer na mesa e na vida de quem tem diabetes.
2015. Sanofi-Aventis Brasil.

2. Feinman et al. 2015. Dietary carbohydrate restriction as the first
approach in diabetes management: Critical review and evidence base.
Nutrition, 31:1-13.

Examples:

#definir previamente o vetor alimento
alimento<-c("arroz integral","panqueca de carne","tomate")
#inserir o objeto na função
x<-caRb(alimento,g=30)
x
```

```
#definir alimentos diretamente na função
y<-caRb(alimento=c("bolo de chocolate","suco de morango sem
açúcar","castanha de caju"), g=27)
y

#deixar a função sortear os alimentos
#escolher o tipo de refeição
cafe<-caRb(g=25,ref="café")
cafe

almoço<-caRb(g=35,ref="almoço")
almoço

veg<-caRb(g=24,ref="almoço veg")
veg

Tabela com os alimentos que podem ser escolhidos na função:

abacate
abacaxi
abacaxi em calda
abóbora
abobrinha
açaí com guaraná
acarajé
acerola
açúcar refinado
água de coco
alcachofra
alfajor
almôndega
ameixa seca
ameixa vermelha
amendoim caramelizado
amendoim torrado com sal
amora
arroz branco
arroz-doce
arroz integral
aveia em flocos
banana-maçã
banana-ouro
banana-prata
banana à milanesa
banana-passa
batata cozida
batata assada
batata frita
batata-doce assada
batata-doce cozida
```

```
batata-doce frita
beijinho
beterraba cozida
bife à milanesa
biscoito água e sal
biscoito aveia e mel
biscoito champanhe
biscoito de coco
biscoito cream cracker   biscoito de polvilho
biscoito maizena   Passatempo recheado   Passatempo sem recheio biscoito
recheado
rosquinha de coco
biscoito wafer
bolinho de arroz frito   bolinha de queijo
bolinho de bacalhau
bobó de camarão
bolo com glacê
bolo de banana
bolo de cenoura
bolo de fubá
bolo de milho
bolo de tapioca
bolo de chocolate
brigadeiro
broa de fubá
broa de milho
cacau em pó
café sem açúcar
caju
cajuzinho
caldo-de-cana
canjica
caqui
carambola
castanha de caju
castanha da amazônia
castanha portuguesa
ketchup
cenoura cozida
granola
cerveja
chá sem açúcar
champanhe
chantili
chocolate em pó
chocolate Alpino
chocolate ao leite
chocolate ao leite diet
chocolate Batom
chocolate Bis
chocolate Charge
```

```
chocolate Chokito
chocolate Confete
chocolate Crunch
chocolate Diamante Negro
chocolate Galak
chocolate Kinder Ovo
chocolate meio amargo
chocolate Milkbar
chocolate Nescau
chocolate Prestígio
chocolate Sensação
chocolate Serenata de Amor
chocolate Sonho de Valsa
chocolate Stickadinho
chocolate Suflair
chocolate Talento
chocolate Talento diet
chocolate Twix
chocotone
chuchu cozido
coalhada
cocada
coco ralado
 couve-flor à milanesa
couve-flor cozida
couve refogada
coxinha
creme de espinafre
creme de leite
creme de milho
croissant
croquete
curau
cuscuz paulista
damasco seco
doce de abóbora com coco
doce de batata-doce
doce de coco
doce de goiaba
doce de leite
doce de mamão
empada
empadão
enrolado de salsicha
ervilha enlatada
ervilha torta cozida
esfiha de carne
esfiha de queijo
farelo de aveia
farelo de trigo
```

```
farinha de arroz
farinha láctea
farinha de mandioca
farinha de milho
farinha de rosca
farinha de trigo
fécula de batata
feijão branco cozido
feijão cozido
figo
figo cristalizado
figo em calda
figo seco
filé à milanesa
pinha
frutas cristalizadas
fubá
Gatorade
gelatina diet
gelatina
geleia de amora
geleia de damasco
geleia de framboesa
gemada
goiaba
goiabada
goiabada light
grão-de-bico cozido
homus
iogurte com frutas
 iogurte com frutas light
iogurte com mel
Danette
Danoninho
iogurte natural desnatado
iogurte natural integral
jabuticaba
 jaca
jiló cozido
Karo
kibe assado
kibe cru
kibe frito
kiwi
laranja
laranja-lima
lasanha à bolonhesa
leite de cabra
leite de coco
leite condensado
leite condensado light
```

```
leite de soja integral
leite de soja light
leite de vaca desnatado
leite de vaca integral
leite de vaca semidesnatado
lentilha cozida
limão
maçã
macarrão cozido
maisena
mamão formosa
mamão papaia
mandioca cozida
mandioca frita
inhame cozido
mandioquinha
manga
manjar
maracujá
maria-mole
marmelada
marshmallow
massa de pastel
mel
melancia
melão
merengue
milho cozido
milho verde enlatado
milk-shake de chocolate
mini pizza
miojo
misto-quente
morango
musse de chocolate
musse de maracujá
Mucilon de arroz
Mucilon de milho
nabo cozido
nectarina
achocolatado em pó light
achocolatado em pó
nêspera
Neston aveia
Neston vitamina
nhoque
nozes
nuggets de frango
nuggets de peixe
nuggets de legumes
```

```
Nutella
olho-de-sogra
ovinhos de amendoim
paçoca
palmito em conserva
pamonha
panetone
panqueca de carne
panqueca de frango
pão baguete
pão ciabatta
pão de batata
pão de centeio
pão de forma
pão de forma light
pão de hamburguer
pão de cachorro quente
pão de leite
pão de mel
pão de milho
pão de queijo
pão doce recheado
pão doce simples
pão francês
pão italiano
pão sírio
pão sovado
pastel assado
pastel de feira
pastel português
pavê de chocolate
pavê de nozes
pé-de-moleque
pepino
pêra
pêssego
pêssego em calda
pimentão cozido
pinhão cozido
pipoca
pirão de farinha de mandioca
pirulito
pitanga
pizza
polenta
polvilho
pudim de leite condensado
pudim de pão com passas
purê de batata
queijadinha de coco
quiabo cozido
```

```
quiche de queijo
quindim
rabanada
rabanete cru
rapadura
ravioli
refrigerante
risole
risoto de frango
risoto milanês
romã
sagu em vinho
salada de frutas
salpicão de frango
salsichão
sanduíche natural
sequilho
shoyu
soja cozida
sopa creme de cebola
sopa creme de cogumelo
sopa creme de espinafre
sopa creme de palmito
sopa de ervilha
sopa de feijão
sopa de frango
sopa de legumes com carne
sopa de lentilha
sopa de macarrão
sorvete de massa
sorvete de massa light
picolé de brigadeiro
picolé de chocolate
picolé de coco
picolé de frutas
frozen yogurt
frozen yogurt diet
suco de abacaxi sem açúcar
suco de acerola sem açúcar
suco de caju sem açúcar
suco de laranja sem açúcar
suco de maçã sem açúcar
suco de melancia sem açúcar
suco de morango sem açúcar
suco de pêssego sem açúcar
suco de tomate
suco de uva
suflê de espinafre
suflê de legumes
suflê de queijo
```

```
sushi
suspiro
tabule
tangerina
tapioca
tomate
torrada
trigo cozido
torta de liquidificador
torta de morango
tutu de feijão
uva
uva itália
uva passa
vagem cozida
vatapá
Yakult
```

Arquivos função

script_funcao.r

help_da_funca_o_nt.txt

help_da_funca_o.docx