Lygia Aguiar Del Matto

Mestranda pelo Programa de Pós-graduação em Zoologia do IB-USP. Estou interessada em entender como fatores sociais e a disponibilidade de alimento influenciam o investimento nas competições pré- e pós-copulatória em machos de *Nephila clavipes*.

exec

Propostas

Proposta A

Motivação e proposta em linhas gerais

Algumas pessoas necessitam tomar diversos remédios diariamente, seja por doenças crônicas, por tratamentos pós-operatórios ou combinações de complicações e tratamentos. Cada remédio deve ser tomado em intervalos de tempo diferentes e possui algumas restrições (por exemplo: não pode ser tomado logo após as refeições, não pode ser tomado junto com outros remédios, etc.). Um paciente poderá encontrar dificuldades em administrar seus remédios se forem muitos remédios prescritos e também se cada remédio tiver muitas restrições. Pensando em facilitar a vida do paciente, minha proposta é criar uma função no R que retorne uma tabela com os horários em que devem ser tomados os diferentes remédios e em quais dias. Minha função calcularia os melhores horários para tomar cada um dos remédios com base em suas restrições. A tabela a ser desenvolvida a partir desses cálculos poderia ser imprimida pelo usuário para que ele pudesse tê-la em todos os lugares, ou até mesmo pendurá-la na porta da geladeira para seus avós.

Objeto de entrada:

O objeto de entrada da minha função será um *dataframe* com seis colunas. São elas: “nome do remédio”, “tipo de remédio” (comprimido, gotas, colírio, pomada, etc.), “intervalo entre doses”, “horário da primeira dose”, “duração do tratamento” e “restrições” (alimentar ou combinação com outros medicamentos). Minha ideia é que o *dataframe* seja preenchido pelo usuário através de perguntas geradas pela função. A função perguntaria, por exemplo, “Qual o nome do remédio?” e salvaria a resposta na primeira linha e coluna de um *dataframe*, em seguida perguntaria sobre o intervalo entre as doses, salvando a resposta na primeira linha e na segunda coluna do *dataframe* e assim por diante. Caso o usuário deseje, ele poderá fornecer um arquivo com todos esses dados já tabelados com o argumento *file*.

Objeto de saída:

Meu objeto de saída será uma tabela indicando os horários em que devem ser tomados todos os remédios levando em consideração restrições e etc. Apesar de ser uma tabela, minha ideia é customizá-la (alterando fonte, cores, etc.) utilizando funções relacionadas a gráficos. Dessa forma eu “desenharia” a tabela, como se fosse um gráfico, para que o usuário possa salvá-la em uma imagem

a ser impressa. Se houver algum pacote que permita customizar tabelas no R, sem que seja necessário fazer a customização da maneira como descrevi acima, ficarei feliz em saber.

Incorporando sugestões

A “Motivação e proposta em linhas gerais” e o “Objeto de entrada” continuam os mesmos, porém o “Objeto de saída” será modificado após sugestão da monitora.

Novo objeto de saída: Meu objeto de saída será uma tabela, salva em formato Microsoft Word ou PDF, indicando os horários em que devem ser tomados todos os remédios levando em consideração restrições e etc. A tabela será feita utilizando o pacote RMarkdown que utiliza a linguagem de formatação de texto Markdown e permite criar arquivos externos.

Proposta B

Motivação e proposta em linhas gerais

Para comprar uma casa, um carro, ou até mesmo uma bicicleta é necessário planejar. Minha proposta é uma função do R que calcule o quanto será necessário economizar mensalmente para comprar algum item caro, acima de sua renda mensal.

Objeto de entrada:

O objeto de entrada dessa função é um *dataframe* com cinco colunas. São elas: “renda mensal”, “poupança” (indica a quantidade de dinheiro que a pessoa possui no presente), “tempo a comprar” (indica em quanto tempo o usuário gostaria de comprar um item), “custo do item” e “gasto mensal” (quanto o usuário gasta mensalmente, em média). Caso haja mais de um item que o usuário deseja comprar, a função levará em consideração os dados do primeiro item para fazer os cálculos do segundo. O usuário responderá perguntas retornadas pelo R para que o *dataframe* seja preenchido.

Objeto de saída:

Há dois objetos de saída para essa função. O primeiro é um *dataframe* que indica os valores a serem economizados mensalmente pelo usuário para que ele consiga comprar os itens que deseja em determinado intervalo de tempo. O segundo objeto de saída é um gráfico de barras (para cada item) que compara a quantidade de dinheiro na poupança, o gasto mensal, o custo do item e a quantidade a ser economizada mensalmente (calculada pela função).

Oi Lygia, a pergunta que não quer calar é cadê a proposta do email? Fiquei #chateada sem ela. Mas tudo bem, acho que seu tempo em criar propostas inovadoras está bem utilizado.

Sobre a proposta 1, existem alguns pacotes do R para gerar *reports*, dê uma olhada no RMarkdown. Com ele você pode criar arquivos em Latex ou Word, acho uma solução bem mais elegante e desafiadora do que fazer estilo gráfico. A documentação é vasta e bem explicativa.

Sobre a proposta 2, eu consideraria deixar o tempo a

comprar como um argumento que pode ou não ser preenchido (às vezes esta é a pergunta). De modo que um dos valores que a função retornaria é o próprio tempo que a pessoa deve economizar. Seria interessante também ter uma porcentagem máxima que a pessoa possa economizar por mês. Imagine que eu quero comprar uma bicleta de 5.000 reais em 5 meses, mas eu recebo 1.500/mês. A função deveria me dizer: ops, você não pode guardar mais da metade do seu salário ao mês. Para comprar a bicicleta você deve economizar por XX meses.

—-*Sara Mortara*

Resposta

Oi Sara, muito obrigada pela ajuda e pelas dicas!

Acabei deixando a ideia do e-mail de lado. Achei que ela podia ser uma opção bem simples e que não exploraria muito o que aprendi sobre o R. Penso que optando por ela eu passaria muito mais tempo pesquisando do que fazendo a função. Mas ainda me interesso por essa ideia, vamos ver se o tempo vai me ajudar a fazê-la algum dia.

E decidi que vou fazer a função da primeira proposta! Vou procurar sobre esses pacotes que você mencionou, obrigada!

Lygia

Fantástico Lygia. Encare o desafio e seja feliz! — *Sara Mortara*

Função medica

Atenção! Antes de utilizar o código abaixo é necessário que você faça três coisas:

Primeiro, instale o pacote “rmarkdown” no seu R!

Segundo, instale o pacote “knitr” no seu R!

Terceiro, baixe o arquivo tabelaremedios.txt e mude a extensão para .Rmd (peço perdão por isso, eu tentei subir o arquivo já em .Rmd, mas apareceu a seguinte mensagem de erro: “O envio foi bloqueado. Essa extensão de arquivo é proibida!”). Salve o arquivo tabelaremedios.Rmd em seu diretório de trabalho.

O código de tabelaremedios.Rmd está abaixo:

```
---
title: "Tabela de Remédios" #dá o título do arquivo
output: word_document #especifica qual o formato do output
---

```{r file, echo=FALSE}
## o código acima mostra que suas próximas linhas são códigos do R.
echo=FALSE indica que eles não devem aparecer no arquivo Word gerado pela
função
tabela.remedios <- read.table("tabela.txt", header=TRUE, sep=" ", dec=",")
#salva um data.frame com a tabela gerada pela função medica no R

```


```{r tabela, echo=FALSE, response="asis"}
## o código acima mostra que suas próximas linhas são códigos do R.
echo=FALSE indica que eles não devem aparecer no arquivo Word gerado pela
função e response="asis" indica que apenas a resposta das funções deverão
aparecer no arquivo Word exatamente como foram geradas
library(knitr) #abre o pacote knitr, onde está a função kable, usada abaixo
kable(tabela.remedios, row.names=FALSE, align="c") #função kable transforma
o data.frame em uma tabela, removendo a primeira coluna com os números das
linhas (row.names=FALSE), e centralizando as colunas com (align="c")

```


Obrigada por usar a função medica! Melhoras!
```

Apenas seguindo os três passos acima você poderá gerar um arquivo de Word com a sua tabela de remédios.

Abaixo, o help da função medica:

```
medica{}                package: -                R Documentation

Tabela de medicamentos

Description:

A função medica é uma função interativa que retorna uma tabela em documento
Word com as datas e os horários em que devem ser tomadas doses de diferentes
remédios.

Usage:

medica(arquivo)

Arguments:

arquivo                 O usuário deve especificar um arquivo em formato CSV
```

em seu diretório de trabalho para que ele seja lido pela função. Caso o usuário não especifique um arquivo, a função irá rodar uma série de perguntas que deverão ser respondidas pelo usuário.

Details:

O arquivo em formato CSV a ser colocado no argumento arquivo deve ter 7 colunas com os seguintes nomes e na seguinte ordem: “Nome”, “Tipo”, “Doses”, “Intervalo”, “Hora.1a” , “Duration” , “Restrictions”. Dados faltantes devem ser preenchidos com NA.

As perguntas e as colunas referentes ao nome, tipo, doses e restrições do remédio podem ser respondidas como o usuário preferir. Caso não se aplique a pergunta, a resposta pode ser preenchida com NA.

A pergunta da quantidade de remédios deve ser respondida com um número.

A pergunta e a coluna de intervalo de tempo entre doses deve ser respondida apenas com o número de horas. Caso sejam horas cortadas, o usuário pode responder com uma proporção da hora. Por exemplo: 30 minutos = 0.5 horas

A pergunta da primeira hora a ser tomada o remédio deve ser respondida obrigatoriamente no formato “horas:minutos”

A pergunta referente a duração do tratamento deverá ser respondida em número de dias.

Value:

A função retorna uma tabela salva em um documento Word (.docx). Para que isso seja possível, antes de rodar a função é necessário salvar o documento “tabelaremedios.Rmd” no diretório de trabalho e instalar os pacotes “knitr” e “rmarkdown”. O código do documento “tabelaremedios.Rmd” está no endereço: http://ecologia.ib.usp.br/bie5782/doku.php?id=bie5782:01_curso_atual:alunos:trabalho_final:lygia.matto:start

Warning:

É necessário responder todas as perguntas para que a função funcione corretamente.

Perguntas em que respostas diferentes de números impossibilitariam o funcionamento da função:

```
            Olá! Quantos remédios você precisa tomar?
            De quantas em quantas horas você deve tomar esse remédio?
            Que horas você tomará esse remédio pela primeira  vez?
(Responder no seguinte formato: "horas:minutos")
            Por quantos dias você deve tomar esse remédio?
```

É necessário salvar o documento “tabelaremedios.Rmd” no diretório de trabalho para que a tabela em documento Word seja produzida pela função.

```
Além disso é necessário ter instalado os pacotes “knitr” e “rmarkdown”.

Author(s):

Lygia Aguiar Del Matto (2016)
lygia.delmatto@gmail.com

Examples:

#para usar o prompt com perguntas direto
medica()

#para usar a função com o arquivo fornecido de exemplo
medica(“remedios1.csv”)
```

Abaixo, o código da função medica:

```
################################## Função Medica
###################################
################################################## por Lygia Aguiar Del Matto
- 2016

###Não esqueça de instalar os pacotes "knitr" e "rmarkdown" antes de
começar!###

medica <- function(arquivo) #cria a função medica() que possui apenas
"arquivo" como argumento.
  ###Abaixo há duas opções de input: ou usuário carrega um arquivo .csv com
os nomes das colunas pré-estabelecidos (ver help) ou o usuário responde
algumas perguntas que preencherão um data.frame com os mesmos nomes de
colunas pré-estabelecidos
{
  if(missing(arquivo)) #se o usuário não fornecer um arquivo .csv de input,
ele irá responder algumas perguntas para formar um dataframe
  {
    cat("Bem-vindo a função medica!") #cria uma mensagem recepicionando o
usuário
    quantidade.c <- readline(prompt="Olá! Quantos remédios você precisa
tomar? ") #pergunta quantos remédios o usuário precisa tomar e salva a
quantidade fornecida por ele em um objeto de classe character
    quantidade <- as.numeric(quantidade.c) #transforma o objeto criado pela
pergunta acima em um objeto de classe númerica, para que seja possível
realizar operações matemáticas com ele
    matriz.remedios <- matrix(rep(NA, (quantidade*7)), nrow=quantidade,
ncol=7) #cria uma matriz de repetição de NAs com número de linhas igual a
quantidade respondida pelo usuário da primeira pergunta. A matriz possui 7
colunas que correspondem ao nome do remédio (Nome), tipo de remédio (Tipo),
quantidade de doses (Doses), intervalo entre as doses (Intervalo), primeira
hora a ser tomada a primeira dose do remédio (Hora.1a), a duração do
```

```
tratamento (Duration) e restrições (Restrictions).
    cat("A partir de agora, você irá responder perguntas sobre cada remédio, um por vez.\n ") #retorna uma mensagem ao usuário informando-o de que as perguntas devem ser respondidas para cada remédio individualmente
    for(j in 1:quantidade) #Inicía um ciclo de perguntas sobre os remédios. O ciclo vai de 1 até a quantidade de remédios informada na primeira pergunta.
    {
      cat ("As próximas perguntas serão sobre o ", j, sep="","o remédio de seu tratamento") #Retorna uma mensagem informando ao usuário qual remédio ele irá responder as próximas perguntas
      nome <- readline(prompt="Qual o nome do remédio? ") #Pergunta ao usuário qual o nome do remédio e salva a resposta da pergunta em um objeto chamado nome
      matriz.remedios[j,1]=nome #Coloca o objeto "nome" na posição indicada(j,1) na matriz de NAs criadas fora da função for()
      tipo <- readline(prompt="Qual o tipo de remédio? ")#Pergunta ao usuário qua o tipo de remédio e salva a resposta da pergunta em um objeto chamado tipo
      matriz.remedios[j,2]=tipo #Coloca o objeto "tipo" na posição indicada (j,2) na matriz de NAs criadas fora da função for()
      doses <- readline(prompt="Quantas doses desse remédio você precisa tomar em cada horário? ") #Pergunta ao usuário quantas doses serão tomadas por vez e salva a resposta da pergunta em um objeto chamado doses
      matriz.remedios[j,3]=doses #Coloca o objeto "doses" na posição indicada (j,3) na matriz de NAs criadas fora da função for()
      intervalo1 <- readline(prompt="De quantas em quantas horas você deve tomar esse remédio? ") #Pergunta ao usuário o intervalo de tempo entre as doses e salva a resposta da pergunta em um objeto chamado intervalo1
      matriz.remedios[j,4]=intervalo1 #Coloca o objeto "intervalo1" na posição indicada (j,4) na matriz de NAs criadas fora da função for()
      hora.1a <- readline(prompt='Que horas você tomará esse remédio pela primeira  vez? \n (Responder no seguinte formato: "horas:minutos") ') #Pergunta ao usuário o primeiro horário em que ele tomará seu remédio e salva a resposta da pergunta em um objeto chamado hora.1a
      matriz.remedios[j,5]=hora.1a #Coloca o objeto "hora.1a" na posição indicada (j,5) na matriz de NAs criadas fora da função for()
      duracao <- readline(prompt="Por quantos dias você deve tomar esse remédio?  ") #Pergunta ao usuário a duração do tratamento e salva a resposta da pergunta em um objeto chamado duracao
      matriz.remedios[j,6]=duracao #Coloca o objeto "duracao" na posição indicada (j,6) na matriz de NAs criadas fora da função for()
      restricao <- readline(prompt="Restrições para o uso do medicamento? ") #Pergunta ao usuário se há restrições para o tratamento e salva a resposta da pergunta em um objeto chamado restricao
      matriz.remedios[j,7]=restricao #Coloca o objeto "restricao" na posição indicada (j,7) na matriz de NAs criadas fora da função for()
    }
    matriz.remedios #retorna a matriz de remédios após todos os ciclos de for() estarem completos - a ideia é que a matriz esteja preenchida com os dados fornecidos pelo usuário
```

```
    remedios <- data.frame(matriz.remedios) #transforma a matriz de remédios
em um data.frame
    colnames(remedios) <- c("Nome", "Tipo", "Doses", "Intervalo", "Hora.1a",
"Duration", "Restrictions") #nomea as colunas do dataframe remedios
    remedios$Intervalo <- as.numeric(as.character(remedios$Intervalo))
#convertendo a classe do vetor de Intervalo dentro do dataframe remedios.
Converter a classe desse vetor para "numeric" permite que algumas funções
sejam feitas com ele no restante do código da função medica(). Para que seja
possível converter o vetor para "numeric" é necessário convertê-lo para
"character" primeiro.
    remedios$Duration <- as.numeric(as.character(remedios$Duration))
#convertendo a classe do vetor de Duration dentro do dataframe remedios.
Converter a classe desse vetor para "numeric" permite que algumas funções
sejam feitas com ele no restante do código da função medica(). Para que seja
possível converter o vetor para "numeric" é necessário convertê-lo para
"character" primeiro.
  } else { #Caso o arquivo seja fornecido pelo usuário:
    remedios<-read.table(arquivo, header=TRUE, sep=";", dec=".", as.is=TRUE)
#Cria um data frame a partir de um arquivo com todos os dados a serem usados
na função. O arquivo deve estar em formato .csv para funcionar
  }
  ####A seguir serão criados os objetos para compor a tabela com os horários
dos remédios ordenados pela data e pela hora
  for(i in 1:length(remedios$Nome)) #Criando vetores com as datas, os
horários, o nome, o tipo e as doses para cada remédio a ser tomados
  {
    data.h <-Sys.Date() #Cria objeto com a data do dia em que a função for
usada
    hora1 <- remedios$Hora.1a[i] #Cria um objeto com a primeira hora em que
será tomado o remédio correspondente a linha i
    data.h.e.hora1 <- paste(data.h,hora1) #Une os objetos com a data atual e
com a primeira hora em que será tomado o remédio e salva o resultado em um
objeto
    data.h.e.hora1a <- strptime(data.h.e.hora1, "%Y-%m-%d %H:%M")
#Transforma o objeto com a data atual e a primeira hora em um objeto de
classe POSIXt (relacionado ao tempo). Na função strptime() está especificado
o formato da data e da hora que estão no objeto data.h.e.hora1a
    intervalo <- as.difftime(remedios$Intervalo[i], units="hours")
#Transforma o intervalo de tempo entre cada dose do remédio correspondente a
linha i em um objeto de classe difftime (em horas - especificado pelo
argumento units) e salva o resultado em um objeto
    data.final <- strptime((Sys.Date()+remedios$Duration[i]), "%Y-%m-%d")
#Objeto com a data final em que será tomado a última dose de um remédio
correspondente a linha i, em classe POSIXt. Essa data foi obtida a partir da
soma da data atual e da duração do tratamento.
    datas.e.horas <- seq.POSIXt(from=data.h.e.hora1a, to=data.final,
by=intervalo) #Gera uma sequência de datas e horários de classe POSIXt, da
data e hora atual (argumento from) até o último dia de tratamento (argumento
to), somando os invervalos entre doses dos remédios (argumento by).
    seq.datas <- format(datas.e.horas, "%d/%m") #Salva um vetor apenas com a
```

```
sequência das datas em formato dia/mês
    seq.horas <- format(datas.e.horas, "%H:%M") #faz um vetor apenas com a
sequência das horas em formato horas:minutos
    seq.nome <- rep(remedios$Nome[i], times=length(seq.datas)) #cria um
vetor com os nomes dos remédios - uma repetição do nome do remédio da linha
i vezes o comprimento do vetor de datas para aquele remédio
    seq.tipo <- rep(remedios$Tipo[i], times=length(seq.datas)) #cria um
vetor com os tipos de remédio - uma repetição do tipo do remédio da linha i
vezes o comprimento do vetor de datas para aquele remédio
    seq.doses<- rep(remedios$Doses[i], times=length(seq.datas)) #cria um
vetor com as doses de remédio - uma repetição das doses do remédio da linha
i vezes o comprimento do vetor de datas para aquele remédio
    seq.restricoes<- rep(remedios$Restrictions[i], times=length(seq.datas))
#cria um vetor com as doses de remédio - uma repetição das doses do remédio
da linha i vezes o comprimento do vetor de datas para aquele remédio
    #Agora a tabela dos remédios será feita em duas etapas. Primeiro será
gerada uma tabela para o primeiro remédio e a tabela dos próximos remédios
serão incorporadas a essa tabela inicial
    if(i==1) #Para criar uma tabela inicial apenas com o primeiro remédio
    {
      tabela <- data.frame(seq.nome, seq.tipo, seq.doses, seq.restricoes,
seq.datas, seq.horas) #Cria um dataframe com a sequência de nome, a
sequência de tipo, a sequência do número de doses, a sequência de datas e a
sequência de horas do primeiro remédio
      colnames(tabela) <- c("Remédio", "Tipo", "Doses", "Restrições",
"Data", "Hora") #nomea as colunas do dataframe gerado acima
    }
    tabela #Retorna a tabela de remédios apenas para o primeiro remédio
gerada em if(i==1)
    if(i>1) #Quando há mais de um remédio, a tabela gerada acima é editada
adicionando-se mais linhas a ela correspondentes aos outros remédios
    {
      tabela2 <- data.frame(seq.nome, seq.tipo, seq.doses, seq.restricoes,
seq.datas, seq.horas) #Cria um dataframe com a sequência de nome, a
sequência de tipo, a sequência do número de doses, a sequência de datas e a
sequência de horas do segundo, terceiro, ... i remédio
      colnames(tabela2) <- c("Remédio", "Tipo", "Doses", "Restrições",
"Data", "Hora") #nomea as colunas do dataframe gerado acima
      tabela <- rbind(tabela, tabela2) #Une a tabela gerada quando i==1 com
a tabela gerada quando i>1 em um objeto chamado tabela (substituindo a
tabela gerada quando i==1)
    }
  }
  ##### Ordenando as linhas da tabela final:
  tabela$Data <- strptime(as.character(tabela$Data), "%d/%m") #Converte a
classe do vetor Data no dataframe tabela para a classe POSIXt. Com isso é
possível ordenar as linhas pelas datas. Caso contrário, o R ordenaria as
linhas com base nessa coluna utilizando apenas os números dos dias,
ignorando os meses. Para fazer isso é necessário converter o vetor para
"character" primeiro e determinar o formato da data na função strptime()
  tabela$Hora <- strptime(as.character(tabela$Hora), "%H:%M") #Converte a
```

```
classe do vetor Hora no dataframe tabela para a classe POSIXt. Com isso é
possível ordenar as linhas pelas horas. Para fazer isso é necessário
converter o vetor para "character" primeiro e determinar o formato das horas
na função strptime()
  tabela <- tabela[order(tabela$Data, tabela$Hora),] #Salva a tabela com as
linhas ordenadas primeiro pelas datas e em seguida pelos horários dos
remédios
  tabela$Hora <- format(tabela$Hora, "%H:%M") #Formata o vetor de Hora no
dataframe tabela para que a hora seja exibida no formato horas:minutos
  tabela$Data <- format(tabela$Data, "%d/%m") #Formata o vetor de Data no
dataframe tabela para que a data seja exibida no formato dia/mês
  write.table(tabela, file="tabela.txt") #Cria um arquivo .txt com a tabela
pronta. Esse arquivo foi criado para ser usado pelo Rmarkdown para criar um
arquivo de word.
  library(knitr) #abre o pacote knitr, necessário para gerar a tabela em um
documento word.
  library(rmarkdown) #abre o pacote rmarkdown, necessário para gerar um
documento word
  render("tabelaremedios.Rmd") #"roda" o documento .Rmd já salvo no
diretório de trabalho. Esse documento irá abrir a tabela salva em um arquivo
.txt e irá embeleza-la levemente criando um arquivo word.
  return(tabela) #retorna o dataframe com a tabela de remédios pronta
}

############################## Fim da função medica
################################
```

Arquivos da função

Arquivo que você deve baixar antes de tudo, converter a extensão para .Rmd e salvar no seu diretório de trabalho –>tabelaremedios.txt

Help da função –> helpmedica.txt

Código da função –> medica.r

Exemplo de tabela que pode ser usada para testar a função –> remedios1.csv