Juliana Rodrigues Moron

Mestre em Ecologia Aplicada ao Manejo e Conservação de Recursos Naturais. Pesquisadora colaboradora do Laboratório de Bioacústica e Ecologia Comportamental da Universidade Federal de Juiz de Fora - UFJF/LABEC.

Lattes: http://lattes.cnpq.br/9310301739879224

Meus exercícios

Execícios preparatórios: notaR + exercicio1.r

Exercícios da Aula 2: notaR

Exercícios da Aula 3: notaR

Exercícios da Aula 4: notaR + ex4.r

Exercícios da Aula 5: ex5.r 

Exercícios da Aula 6: notaR

Exercícios da Aula 7a: notaR + 107.2.r

Exercícios da Aula 7b: 7b.r

Exercícios da Aula 8: ex8.r

Exercícios da Aula 9: notaR + 9.2.r

Proposta de Trabalho Final

Proposta A

Trabalhar com bioacústica envolve analisar grande volume de dados; buscando, reconhecendo e extraindo parâmetros de frequência, duração e amplitude de sinais acústico. Arquivos de som de qualidade normalmente são pesados e levam tempo para serem processados, carregando a memória do computador durante a execução. Mas esses arquivos muitas vezes contêm longos períodos de silêncio entre um e outro sinal acústico. No pacote “seewave” para análise acústica, existem as funções: cutw (corta uma seção da time wave), pastew (cola uma time wave na outra) e zapsilw (deleta períodos de silêncio de uma time wave). Dessa forma, minha proposta é a partir das funções que já existem no “seewave” elaborar uma função que atue dentro de um diretório com vários

arquivos de som, reconheça seções silenciosas nesses arquivos, as remova e junte somente as seções com sinal acústico em poucos passos; limpando o arquivo de som e o deixando mais leve e somente com sinais de interesse.

> Oi Juliana, Sua proposta parece interessante. Porém me preocupou você descrever como: “elaborar um código”. Um código para uma análise ou preparação de dados é diferente de uma função. Vejo que sua proposta pode ser executada como uma função. Atente-se a deixar a entrada geral e a execução das tarefas dentro da função o mais geral possível. Deve funcionar não só pro seu diretório, mas para o diretório de qualquer pessoa que tenha muitos arquivos de som. Quais são os argumentos da função? Diretório e arquivos de interesse? Pense em como a função executará as tarefas que você descreve. — *Sara*

Proposta B

Outra proposta, ainda atuando nesse diretório com vários arquivos de interesse, é criar uma função que separe as gravações muito longas ou pesadas (em 4 ou 5 arquivos diferentes), ou junte gravações menores (em só um arquivo). Pois esse trabalho também depende da capacidade da máquina e demanda tempo.

> Acho uma ótima proposta. É bom especificar na função que pode demorar. Pode até sugerir quantos cafés a pessoa pode tomar enquanto espera.
>
> As duas propostas são boas e viáveis. Escolha a que mais te motive (atentando para os pontos que coloquei na 1) e siga em frente!
>
> — *Sara 2016/04/29 11:49*

> Obrigada pelas dicas e motivação Sara!! Creio que vou ficar com a proposta B, vamos ver quantos cafezinhos vou ter que tomar enquanto elaboro... — *Juliana*

Ajuda para função gather - Proposta B

```
gather                package:seewave                R Documentation


Description:
  Atua dentro de um diretório com vários arquivos de audio, juntando
gravações menores em só um arquivo de som.

Usage:
  for(i in 2:length(x))
  { gather = pastew(gather, x[i], f, at="start", join=TRUE, tjunction=0,
choose=FALSE, marks=TRUE, output="Wave") }

Arguments:
  x Diretorio com os arquivos de audio em formato WAV.
  gather Objeto final criado a partir do primeiro elemento da lista de
arquivos de audio
  f Frequência de amostragem da onda sonora (em Hz). Não precisa ser
especificado se embutidos na onda.
  at Posicao na segunda onda sonora onde a primeira sera colada.
Especificada como "start", "middle" ou "end".
  join Se TRUE as duas ondas serão coladas e juntadas, removendo o último
ponto da onda 2.
  tjunction Vetor numerico para remover cliques na juncao das ondas sonoras.
O valor especifica a duracao em segundos onde o valor real sera substituido
por uma interpolacao linear. Essa duracao deve ser de alguns milisegundos.
  choose logical, se TRUE o ponto onde wave1 será colado onda 2 (= at) pode
ser graficamente escolhido com um cursor.
  marks logical, se TRUE mostra onde onda 1 foi colada (por padrão TRUE).
  output character string, a classe do objeto a ser retornado, "matrix",
"Wave", "Sample", "audioSample" ou "ts".

Details:
  A partir de um diretorio com arquivos de audio, juntam-se rapidamente
gravacoes menores em um unico arquivo de audio.

Value:
  Um arquivo WAVE é um arquivo de tipo RIFF (Resource Interchange File
Format) utilizado para o armazenamento de dados de áudio descompactados. É
identificado pela extensão .wav.

Author(s):
  Juliana Rodrigues Moron
  julianamoron@hotmail.com

References:
Crawley, M. J. The R Book. Wiley, New York, 2007.

Sueur J., T. Aubin, C. Simonis. 2007. Equipment review: seewave, a free
```

```
modular tool for sound analysis and synthesis. Bioacoustics 18(2) 212-226.

See also:
Sueur, Jerome. (2014) Package ‘seewave’.

Ligges, Uwe. (2015) Package ‘tuneR’.

Examples:
#No exemplo apresentado a seguir sao plotados os arquivos de audio de baleia
bowhead fornecidos.

setwd("dir") #coloca no diretório de interesse
read.wav = lapply(list.files(pattern="*.wav"), readWave) #aplica a funcao
readWave na lista de arquivos de interesse
read.wav

gather = read.wav[1] #cria um objeto final a partir do primeiro elemento da
lista de arquivos
for(i in 2:length(read.wav))     #cria um for para juntar todos os elementos
{ gather = pastew(gather, read.wav[i], f=4000, at="start", join=TRUE,
choose=FALSE, marks=TRUE, output="Wave") }
```

Ajuda para função divide - Proposta B

```
divide               package:seewave               R Documentation


Description:
  Atua dentro de um diretório com um arquivo de audio longo, separando-o em
arquivos mais curtos e leves.

Usage:
  for(i in 1:length(duration$s.start))
  {
    divide = cutw(file, from = file$s.start[i], to = file$s.end[i],
choose=FALSE, output ="Wave")
    savewav(divide, filename=paste0("divide-", i, ".wav"))
  }

Arguments:

duration objeto com a duracao dos periodos de sinal de interesse e pausa do
arquivo de audio
divide Objeto final da funcao cutw de acordo com os periodos encontrados no
duration
file Arquivo de som
```

```
f Frequência de amostragem da onda sonora (em Hz). Não precisa ser
especificado se embutidos na onda
from Posicao inicial da onda sonora que sera criada
to Posicao final da onda sonora que sera criada
choose logical, se TRUE os pontos from e to podem ser escolhidos
graficamente com um cursor no oscilograma
output character string, a classe do objeto a ser retornado, "matrix",
"Wave", "Sample", "audioSample" ou "ts"
savewav Salva os arquivos cortados no diretorio de acordo com a ordem que
foi cortado

Details:
  A partir de um diretorio com um arquivo de audio, gerar rapidamente
gravacoes menores cortadas desse arquivo maior.

Value:
  Um arquivo WAVE é um arquivo de tipo RIFF (Resource Interchange File
Format) utilizado para o armazenamento de dados de áudio descompactados. É
identificado pela extensão .wav.
References

Author(s):
  Juliana Rodrigues Moron
  julianamoron@hotmail.com

References:
  Crawley, M. J. The R Book. Wiley, New York, 2007.

  Sueur J., T. Aubin, C. Simonis. 2007. Equipment review: seewave, a free
modular tool for sound analysis and synthesis. Bioacoustics 18(2) 212-226.

See also:
  Sueur, Jerome. (2014) Package 'seewave'.

  Ligges, Uwe. (2015) Package 'tuneR'.
  Bagwell, Chris. (2013) SoX – Sound eXchange, the Swiss Army knife of audio
manipulation.

Examples:
  #No exemplo apresentado a seguir e plotado o arquivo de audio de baleia
azul fornecido.
  duration = timer(file, f=500, threshold=5, msmooth=c(50,0)) #marca os
periodos com sinal de interesse e pausa, importante saber o threshold

for(i in 1:length(duration$s.start)) #corta o arquivo de acordo com os
sinais de interesse e pausa identificados e os salva no diretorio
{
  divide = cutw(file, from = duration$s.start[i], to = duration$s.end[i],
choose=FALSE, output ="Wave")
  savewav(divide, filename=paste0("divide-", i, ".wav"))
```

```
}
```

Código para função gather - Proposta B

```
################################################
## PARA OS ARQUIVOS bowheadwhalesong_example ##
################################################

##baixar os arquivos num diretorio no seu computador

##instalar os pacotes necessários
require(tuneR) #para abrir e salvar os arquivos de audio
require(seewave) #para juntar os arquivos

setwd("dir") #coloca no diretório de interesse

list = list.files(pattern="*.wav") #lista os arquivos do diretório
list

read.wav = lapply(list, readWave) #aplica a funcao readWave na lista de
arquivos de interesse, com o mesmo tamanho
read.wav

gather = read.wav[1] #cria um objeto final a partir do primeiro elemento da
lista de arquivos
for(i in 2:length(read.wav))     #cria um for para juntar todos os elementos
{ gather = pastew(gather, read.wav[i], f=4000, at="start", join=TRUE,
tjunction=0, choose=FALSE, marks=TRUE, output="Wave") }

savewav(gather, f = 4000, filename = "gather.wav") #salva o arquivo de audio
com todos os elementos
```

Código para função divide - Proposta B

```
#####################################################
## PARA O ARQUIVO bluewhale_example (menos pesado) ##
#####################################################

#instala os pacotes necessários
require(seewave)
require(tuneR)

file=readWave("diretorio\\arquivo.wav") #le o arquivo
```

```
duration = timer(file, f=500, threshold=5, msmooth=c(50,0)) #marca os
periodos com sinal de interesse e pausa, importante saber o threshold

for(i in 1:length(duration$s.start)) #corta o arquivo de acordo com os
sinais de interesse e pausa identificados e os salva no diretorio
{
  divide = cutw(file, from = duration$s.start[i], to = duration$s.end[i],
choose=FALSE, output ="Wave")
  savewav(divide, filename=paste0("divide-", i, ".wav"))
}

########################################
## EXTRA: PARA ARQUIVOS MAIS PESADOS ##
########################################

##recomenda-se o download do sox (https://sourceforge.net/projects/sox/)
require(seewave)

sox("diretorio\\arquivo.wav arquivo_clip.wav trim 0.25 0.75") #o comando
funciona com: input output effect
```

Arquivos para a função gather - Proposta B

gather_help.r gather.r

Arquivos para a função divide - Proposta B

divide_help.r divide.r
