

soundmeteR: pacote em R para calcular níveis de pressão e intensidade sonora

Roteiro para a oficina do I Simpósio de Física aplicada à Ecologia e Conservação

Cássio Rachid Simões (cassiorachid@conservasom.com.br)

2025-11-05

Contents

| | | |
|----------|---|----------|
| 1 | Introdução | 1 |
| 2 | Preparação do ambiente | 1 |
| 2.1 | Programas necessários | 2 |
| 2.2 | Instalação do <i>devtools</i> | 2 |
| 2.3 | Instalação do <i>soundmeteR</i> | 2 |
| 3 | Visão geral do <i>soundmeteR</i> | 2 |
| 4 | Exemplos de uso | 3 |
| 4.1 | Carregando os pacotes e o arquivos de exemplo | 3 |
| 4.2 | Aplicações do <i>soundmeteR</i> em gravações não calibradas | 3 |
| 4.3 | Aplicação do <i>soundmeteR</i> em gravações com calibração | 4 |

1 Introdução

Este é um roteiro apresentando o fluxo de análises utilizando o pacote *soundmeteR*. O presente documento foi idealizado para a oficina realizada durante o I Simpósio de Física aplicada à Ecologia e Conservação, realizado em novembro de 2025 em Foz do Iguaçu-PR.

Todos os códigos apresentados neste roteiro devem ser executados no R. Você pode usar RStudio, VS Code (com suporte a R) ou outra interface gráfica de sua preferência. Recomenda-se especialmente o uso do RStudio, por ser uma interface intuitiva e amplamente adotada pela comunidade R.

2 Preparação do ambiente

O fluxo de trabalho apresentado nesse documento foi pensado de maneira que os participantes possam tanto acompanhar as etapas de análise utilizando seus próprios computadores durante a oficina quanto executar de forma independente após participarem da oficina. Para isso, é necessário que os participantes tenham o *R*, versão do *Rtools* equivalente e o *Rstudio* instalados e atualizados em suas máquinas.

Atualmente, o pacote *soundmeteR* está disponível apenas no repositório oficial do GitHub (<https://github.com/ConservaSom/soundmeteR>) e para instalá-lo será necessário também a instalação do pacote *devtools*, que é utilizado para instalar pacotes diretamente do GitHub.

Nos tópicos a seguir estão descritas as etapas para a preparação do ambiente de análise.

2.1 Programas necessários

Os programas necessários para a instalação e utilização do *soundmeteR* são:

- *R*: <https://cran.r-project.org/>
- *Rstudio*: <https://posit.co/download/rstudio-desktop/>
- *Rtools* (apenas para usuários Windows): <https://cran.r-project.org/bin/windows/Rtools/>

2.2 Instalação do *devtools*

O *devtools* é um pacote essencial para usuários avançados de R, pois permite a instalação de pacotes que ainda não foram publicados no CRAN (o repositório oficial do R). Para instalá-lo, basta executar o comando abaixo dentro do R ou Rstudio:

```
install.packages("devtools")
```

2.3 Instalação do *soundmeteR*

É possível encontrar as instruções de instalação do *soundmeteR* no repositório oficial do GitHub (<https://github.com/ConservaSom/soundmeteR>), mas para facilitar o processo, basta executar o código abaixo para instalar o *soundmeteR* e suas dependências:

```
devtools::install_github("ConservaSom/soundmeteR", dependencies = TRUE)
```

Para verificar se o *soundmeteR* foi instalado corretamente, basta carregar o pacote utilizando o comando abaixo (nenhuma mensagem de erro deve ser exibida):

```
library("soundmeteR")
```

3 Visão geral do *soundmeteR*

O pacote *soundmeteR* é uma ferramenta desenvolvida em R para extração de valores de energia sonora (pressão e intensidade) a partir de gravações em formato Wave (.wav). Ele oferece uma série de funções relacionadas ao fluxo de trabalho para obtenção destas medidas, incluindo a possibilidade de calibração para o cálculo de níveis absolutos de pressão e intensidade e aplicação de curvas de ponderação de frequências (curvas A, B C, Z). O pacote vem sendo aplicado em fluxos de trabalhos e nas análises alguns artigos científicos a alguns anos e atualmente está em fase final de desenvolvimento e preparação do manuscrito. Abaixo estão as funções principais e uma breve descrição de suas funcionalidades. O arquivo PDF do manual do pacote disponibilizado no GitHub (<https://github.com/ConservaSom/soundmeteR>) contém uma descrição detalhada de todas as funções disponíveis.

| Função | Descrição |
|--------------|--|
| leqbands | Calcula o Nível Sonoro Equivalente (Leq) para arquivos para para um arquivos Wave em oitavas ou terças de oitavas |
| leqbands_cal | O mesmo que o *leqbands*, porém permite a calibração a partir de um sinal de referência dentro do mesmo arquivo Wave |
| soundmeter | Computa medidas similares às de um decibelímetro a partir de arquivos Wave, permitindo a escolha da janela de amostragem (fast ou slow) e apresenta medidas comumente encontradas nestes tipos de equipamentos como Leq, L10, L50, L90, Lmax, entre outras |

4 Exemplos de uso

4.1 Carregando os pacotes e o arquivos de exemplo

Carregando os pacotes necessários:

```
# carregar o pacote soundmeteR
library(soundmeteR)
# Carregar o monitoraSom
library(monitoraSom)
# carregando seewave para visualização do sonogramas
library(seewave)
# carregando tuneR para manipulação de arquivos Wave
library(tuneR)
# Carregando dplyr para manipulação dos datasets
library(dplyr)
# Carregando ggplot2 para manipulação de gráficos
library(ggplot2)
```

Carregando o exemplo interno do pacote e visualizando o espectrograma. O exemplo é um canto de um *Thamnophilus stictocephalus*.

```
# Carregando arquivo de exemplo do pacote soundmeteR (Thamnophilus stictocephalus)
data(tham)

# Corte do trecho do arquivo em que o canto da espécie aparece
som <- cutw(
  tham,
  from = 3.5, # zoom no tempo para o momento que o canto da espécie aparece
  to = 7.2,
  output = "Wave"
)

# verificando o arquivo
som

# Espectrograma do arquivo de exemplo
spectro(
  som,
  flim = c(0, 8), # zoom na frequência
  scale = FALSE
)
```

Escutando o arquivo. Algumas máquinas e sistemas operacionais podem necessitar que você defina o reprodutor de áudio com a função `setWavPlayer()` do pacote *tuneR*.

```
# Caso queira escutar o arquivo
listen(
  som
)
```

4.2 Aplicações do *soundmeteR* em gravações não calibradas

A função `leqbands()` calcula o nível sonoro equivalente (Leq) em bandas de oitavas ou terças de oitavas para arquivos Wave. Quando utilizado em gravações não calibradas, a função retorna os valores em dBFS (decibéis em full scale - relativo ao ponto de distorção em 0 dB).

```

# Leq por oitavas
leqbands(
  som,
  band = "octaves", # bandas de frequência utilizadas
  weighting = "A" # curva de ponderação de frequências
)

# Leq por terças de oitavas
leqbands(
  som,
  band = "thirds", # bandas de frequência utilizadas
  weighting = "A" # curva de ponderação de frequências
)

```

A função `soundmeter()` calcula diversas medidas de níveis sonoros similares às medidas obtidas por um decibelímetro. Quando utilizado em gravações não calibradas, a função retorna os valores em dBFS (decibéis em full scale - relativo ao ponto de distorção em 0 dB).

```

# soundmeter por oitavas
soundmeter(
  som,
  tw = "fast", # ponderação temporal "fast" ou "slow"
  fw = "A", # curva de ponderação de frequências
  bands = "octaves" # bandas de frequência utilizadas
)

# soundmeter por terças de oitavas
soundmeter(
  som,
  tw = "fast", # ponderação temporal "fast" ou "slow"
  fw = "A", # curva de ponderação de frequências
  bands = "thirds" # bandas de frequência utilizadas
)

```

4.3 Aplicação do *soundmeteR* em gravações com calibração

Para aplicar a calibração é necessário extrair o valor a partir de uma gravação com sinal de referência ou das especificações do aparelho e microfone utilizados. Aqui, realizaremos o a partir de uma gravação de um sinal de referência com pressão sonora conhecida.

```

# definindo diretório de trabalho
setwd(dirname(rstudioapi::getActiveDocumentContext())$path))

# lendo o arquivo de calibração
calib <- readWave("./calibration/calibration.wav")

# plotando no espectro
spectro(
  calib,
  scale = F
)

# extraindo valor de calibração
calib_value <- calibration(
  calib,

```

```

from = 2, # tempo de início da leitura
to = 8, # tempo de final da leitura
channel = "left", # canal da gravação
leqsignal = 94 # valor de referência do sinal
) %>%
  print()

```

Nesse momento utilizaremos o resultado obtido ao final da prática realizada com o pacote *monitoraSom* no turno anterior para obter valores de pressão sonora para as detecções de *Basileuterus culicivorus*.

Precisamos abrir o app de validação do *monitoraSom* para selecionar as detecções de cantos que não tenham sobreposição com ruído de fundo. Esta ação é necessária pois o ruído pode interferir nas medidas de pressão sonora, gerando valores incoerentes com a realidade.

```

launch_validation_app(
  project_path = ".", validation_user = "User",
  templates_path = "./templates/", soundscapes_path = "./soundscapes/",
  input_path = "./detections/df_detecs_soundmeter.csv",
  output_path = "./detections/df_detecs_soundmeter.csv",
  dyn_range_tmpl = c(-36, 0),
  dyn_range_detec = c(-96, -36),
  wl = 512,
  ovlp = 70,
  time_guide_interval = 0,
  freq_guide_interval = 0,
  overwrite = TRUE,
)

```

Agora, com o resultado da filtragem acima, vamos ler a planilha com as detecções de interesse para, em seguida, extrair os valores de pressão sonora. Aqui, já filtramos apenas as gravações que identificamos como úteis para a extração da pressão sonora.

```

detections <- read.csv(
  "./detections/df_detecs_soundmeter.csv" # planilha com as detecções
) %>%
  filter(
    validation == "TP" # filtro dos arquivos úteis
  ) %>%
  print()

```

Calculando as medidas de pressão sonora de cada detecção com a função *soundmeter*.

```

spl <- soundmeter(
  files = detections$soundscape_path, # lista dos arquivos com o paths
  channel = "left", # canal da gravação a ser utilizado
  from = detections$detection_start, # tempo de início da detecção
  to = detections$detection_end, # tempo final da detecção
  CalibValue = calib_value, # fator de calibração obtido anteriormente
  tw = "fast", # janela de ponderação do tempo
  fw = "A", # curva de ponderação de frequência
  bandpass = c(2500,9000) # filtro de passagem de banda
) %>%
  print()

```

Colocando os valores de L_{Amax} calculados dentro da matriz de detecções úteis e ordenando em ordem crescente de pressão sonora.

```

detections <- detections %>%
  mutate(
    LAmaz = spl$LAmaz
  ) %>%
  arrange(LAmaz) %>% # colocando em ordem crescente de LAmaz
  print()

```

Gerando uma prancha para visualizar os cantos detectados e seus valores de pressão sonora.

```

# gerando o gráficos para visualizar os resultados
detection_plots1 <- detections %>%
  split(., seq(nrow(.))) %>%
  purrr::map(., ~ {
    wav <- tuneR::readWave(
      filename = .x$soundscape_path,
      from = .x$detection_start,
      to = .x$detection_end,
      units = "seconds"
    )
    res <- fast_spectro(
      rec = wav, f = wav@samp.rate, wl = 512, ovlp = 70,
      dyn_range = c(-102, -42), color_scale = "greyscale 1",
      freq_guide_interval = 0, time_guide_interval = 0,
      zoom_freq = c(
        .x$template_min_freq,
        .x$template_max_freq
      )
    ) +
    theme_bw() +
    theme(legend.position = "none") +
    ggtitle( paste0(.x$LAmaz, " dB"))
    return(res)
  }, .progress = TRUE)

# Número de gráficos gerados
length(detection_plots1)

cowplot::plot_grid(
  detection_plots1[[1]], detection_plots1[[2]],
  detection_plots1[[3]], detection_plots1[[4]],
  detection_plots1[[5]], detection_plots1[[6]],
  detection_plots1[[6]], detection_plots1[[7]],
  detection_plots1[[8]], detection_plots1[[9]],
  nrow = 2
)

```