Frequency analysis of a sound

Here we will analysis the sound produced by a frog. The file is called Allobates_femoralis_c1.wav. The question here is let suppose we capture some [musical] sound , what kind of information we can extract by signal processing.

Install and load libraries
Define your local path

```
install.packages( c('tuneR', 'seewave') )
library(tuneR)
library(seewave)
```

tuneR , seewave and chorrrds are dedicated to sound and music processing.
Package tabr is also dedicated to music analysis and generation

Here are the weblink describing functions:
tuneR : https://cran.r-project.org/web/packages/tuneR/tuneR.pdf
seewave : https://cran.r-project.org/web/packages/seewave/seewave.pdf

Download and save all your music data files in a path, for example:
path = "D:/DataProcessingIII/MusicAnalysis/"

1 start

install the required packages

```
> install.packages( c('tuneR', 'seewave') )
```

load the required packages

```
> library(tuneR)
> library(seewave)
```

You may define a path variable to tell R where is your .wav file is located.

```
> path=" D:/DataProcessingIII/MusicAnalysis/"
```

For Windows OS, you need to change \ to / in your path string.

import into R the audio file Allobates_femoralis.wav into an object named s
(Hint: use readWave function, refer to tuneR.pdf)

```
> s <- readWave( paste0(path, "Allobates_femoralis.wav") )
```

determine the class of s

> class(s)

determine the duration of s
(use duration function, refer to seewave.pdf)

> duration(s)

2 oscillogram
visualizes as an oscillogram
You can use oscillo function to visualize it as an oscillogram:
(refer to seewave.pdf)

> oscillo(s)

listen to s
You can also start a media player to listen to it.
(Hint: use listen function, refer to seewave.pdf)

> listen(s)

determine graphically the start and end time positions of the first call based on the oscillogram you obtained in step 2, find out the first call happens at which duration
(from and to time of the first duration)

create a new object named c1 containing only the first call
(hint: use cutw and fir functions, refer to seewave.pdf)

save the new object (c1 object) into a new audio file named Allobates_femoralis_c1.wav
(refer to seewave.pdf)

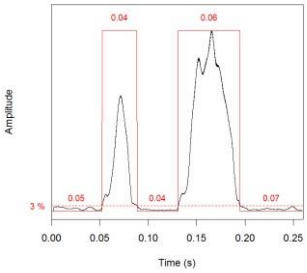
> savewav(c1, filename=paste0(path, "Allobates_femoralis_c1.wav"))

The paste0() function concatenate elements without a separator; it also takes one optional argument, collapse. We use paste0() instead of paste() because it is faster and doesn't need a sep argument.

3 Temporal Analysis
take automatic measurements on c1

(hint: use timer function, refer to seewave.pdf)

> timer(c1, ssmooth=200, threshold=3)



take manual time measurements on the amplitude envelope c1
(hint: use oscillo function, refer to seewave.pdf)

> oscillo(env(c1,plot=FALSE), f=c1@samp.rate, identify=TRUE)

Here, env() function returns the absolute or Hilbert amplitude envelope of a time wave.

4 Dominant frequency

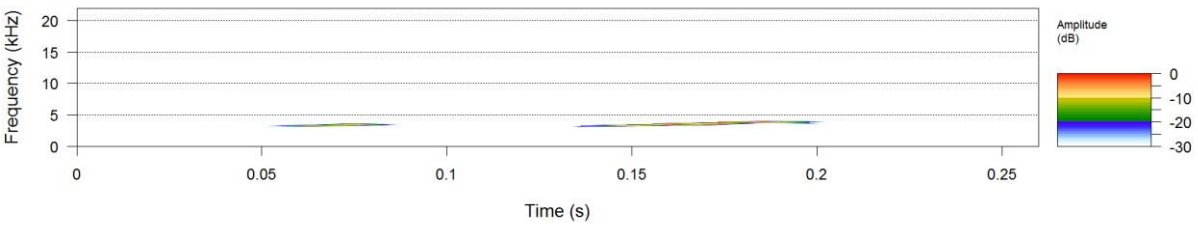
visualize c1 as a spectrogram (default parameters)
(hint: use spectro function, refer to seewave.pdf)

> spectro(c1)

(tip: if you have an error such as Error in plot.new() : figure margins too large, enlarge the window of plot and rerun the function)

repeat the following spectrogram ofc1obtained with a Fourier window made of 1024 samples and an overlap between successive windows of 87.5%
(hint: use spectro function with different parameters, refer to seewave.pdf)

> spectro(c1, osc=TRUE, collevels=seq(-60,0,2), l=1024, ovlp=87.5, colwave="blue")



track the dominant frequency ofc1
(hint: use dfreq function, refer to seewave.pdf)

df <- dfreq(c1, threshold=5, wl=1024, ovlp=87.5)

track the fundamental frequency of c1
(hint: use fund function, refer to seewave.pdf)

> f0 <- fund(c1, threshold=5, wl=1024, ovlp=87.5, fmax=2000)

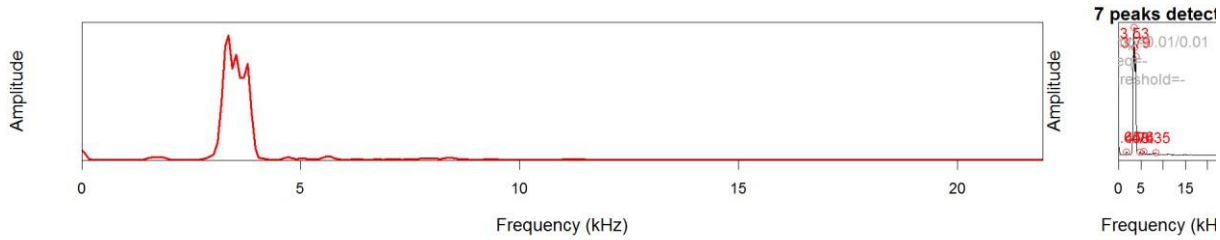
(tip: if you have an error such as Error in plot.new() : figure margins too large, enlarge the window of plot and rerun the function)

5 Spectral analysis
compute the mean frequency spectrum of the first note of c1
(tip : use *timer* and *meanspec,* refer to seewave.pdf)

find the main frequency peaks of the mean frequency spectrum
(tip: use fpeaks)

get the main features (properties) of the mean frequency spectrum
(tip: use specprop function, refer to seewave.pdf)

> specprop(sp)

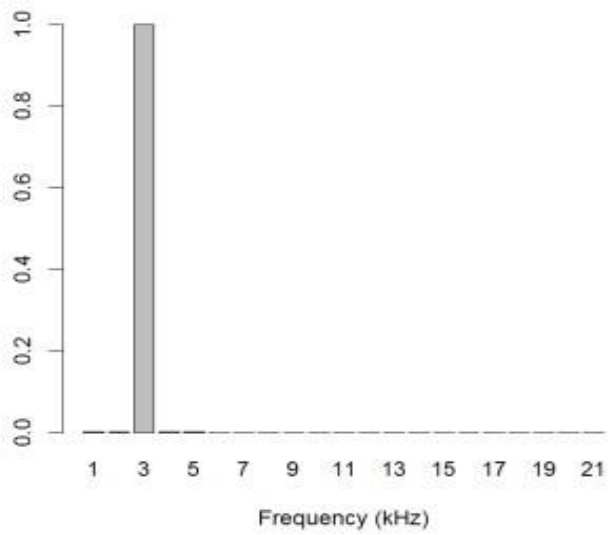


Using *meanspec* and *which* find the dominant frequency

Plot with *meanspec*

Using *box* function and *soundscapespec* plot the spectrogram of frequencies
(refer to seewave.pdf)

> box()
> soundscapespec(c1)


0.0
0.2
0.4
0.6
0.8
1.0
1
3
5
7
9
11
13
15
17
19
21
Frequency (kHz)