Acoustic Data Analysis

Walter MX Zimmer

2024-10-26

Contents

1 Introduction

The following text is a basic introduction to use of acoustics in marine ecology and cetacean research. It touches briefly on the underlying physics and technology providing the required notion that is necessary to understand underwater acoustics.

The text tries then to demonstrate how one would approach acoustic data in an ecology context (what type of sound is out there). Also, as sound is fundamental for cetacean life, the text addresses how the analysis of cetacean sound can help to learn about their life.

For this, we address two use cases, one where we analyse some acoustic recordings and try to understand what has been recorded, that is general acoustic data analysis, and another one where we try to learn more about the sound animals make and use, labelled below bio-acoustics. We approach each use case step-by-step and the user is invited to change the processing and visualization parameters, were possible. This may require for R library functions to use the help to learn about the different parameters.

The data processing is using two data sets, one being a recording of striped dolphins for the first use case and the other one the recordings of a echolocating sperm whale for the bio-acoustic analysis.

The processing is carried out by means a a series of R-scripts, which require the following libraries that may needed to be installed:

- `tuneR` to read wav files,
- `gsignal` for some signal processing, and
- `fields` for specialized graphics.

2 Some physics and technology

The first part of this text addresses some basic physics of sound: what is sound, how it is described, how it is measured and how sound propagates.

2.1 Sound description

Sound is the rapid variation of the pressure (deviation from mean pressure) and can be observed using ears and technical sensors (microphone, hydrophone, etc.). Sound only exists in compressible media and does therefore not exist in vacuum.

A propagating sound wave consists of temporal and spacial alternating compressions and rarefactions of the air, water or any compressible medium, the effect of which propagates through the compressible medium. For a fixed location sound pressure varies in time and for a fixed time snap-shot sound pressure varies in space.

2.1.1 Sound frequency

As sound is alternating compression and rarefaction, it is possible to measure how often this alternation occurs in a given time frame. The number of alternations per second is then called the frequency of the sound wave. Humans can hear sound frequency from 30 Hz to about 17 kHz (nominally 20Hz to 20 kHz). Sound with frequencies below 20 Hz is also called infrasound, and ultrasound above 20 kHz. Cetaceans use the whole frequency spectrum from as low as 20 Hz to well over 100 kHz, that is, a large amount of cetacean sound is not audible to humans and require technical instrumentation to be observed.

2.1.2 Sound speed

Sound propagates in air with about 330 m/s and in water about 1500 m/s. The sound speed is not the speed of the particles that are moving around when air or water is compressed, but is the speed of the maximum compression moving through the medium. In other words, if the sound is a short pulse (e.g. sperm whale click), it is the location of the pulse that moves with 1500 m/s through the water. The sound speed depends on the physical and chemical properties of the water (temperature, pressure, and salinity) and increases with temperature, pressure and salinity.

Online resources to estimate the sound speed on ocean waters may be found at npl.co.uk

Here are some typical values for the Mediterranean Sea (salinity 35 ppm, latitude 40°N)

Depth [m]	Temperature [°C]	Sound speed [m/s]
0	20.0	1521
100	13.5	1504
500	13.5	1510
1000	13.5	1518
2000	13.5	1535

2.1.3 Sound pressure

In air the mean atmospheric pressure is about 100 kPa (Pa == Pascal) at the earth surface. Humans can hear sound (variation from mean pressure) as small as 20 μPa. Sound that exceeds 20 Pa is considered by humans as painful loud. In water the mean hydrostatic pressure increases with depth about 100 kPa every 10 m. In an extreme quiet Ocean (no wind, no ships, no earthquakes), ambient noise decreases with frequency and is about 1000 μPa at 10 Hz 10 μPa at 10 kHz. A sperm whale echolocation click can be as loud as 300 kPa.

2.1.4 Sound intensity

Sound intensity is the mean sound energy in a 1 second interval in the direction of sound propagation and is measured in W/m^2 (Watt per square meter). The sound intensity is effectively the Mean-Squared pressure divided by the product of density of the propagation medium and sound speed in it.

$$I = \frac{< P^2 >}{\rho c}$$

with sound pressure P , density of the mediumρ, and sound speed in the medium c.

Water is much less compressible than air (higher density and higher sound speed) and consequently the same sound pressure corresponds to less sound energy in water than in air, or equivalently the same sound energy corresponds to a higher sound pressure in water than in air.

Considering a sound intensity of 1 W/m^2 then the sound pressure in air corresponds to 1 Pa, but to 1.26 kPa in water.

In other words, as energy corresponds to capability to carry out work, the same sound pressure exposure is less damaging in water than in air.

2.1.5 Decibel

As typical sound pressure may vary over 6 decades in air (20 μPa to 20 Pa) and may vary over 10 decades in underwater scenarios (from about 10 μPa to over 300kPa of sperm whale echolocation clicks measured at close ranges), the sound pressure is mostly expressed as a logarithmic quantity, the decibel, or dB. Any logarithmic scale is a relative scale and requires a reference pressure. This reference pressure is (for historical reasons) 20 μPa for sound in air and 1 μPa for underwater sound.

In practical terms:

$$L = 10 \log_{10}(P^2/P_{ref}^2) = 20 \log_{10}(P/P_{ref})$$

where P_{ref} is the reference pressure (20 μPa in air and 1 μPa in water).

Having different reference pressure values makes it important to always indicate its value. So, the maximal sound pressure of sperm whale clicks is given as $L = 230$ dB rel. 1μPa (or in short: $L = 230$ dB \\ 1μPa).

2.2 Sound propagation

Sound is propagating through air and water in form of pressure waves. It is therefore important to know how fast sound propagates and how much sound pressure may be observed at a certain distance from the sound source. The first effect is geometrical spreading of the sound energy and the second effect is the absorption of sound energy.

2.2.1 Sound spreading

While sound is propagating through the ocean it has to obey the law of physics, one of which is that the energy and therefore also the overall sound energy, if not absorbed, must remain constant. The consequence of this is that sound pressure, which propagates in all directions from the source (e.g. a sperm whale) must decrease with the distance from the source (or radius of increasing sphere around source). This decrease is called spherical spreading loss. In case that sound propagation is limited by surface and bottom, the sound pressure decreases with the square root of the distance and then called cylindrical spreading loss. This decrease in sound pressure is also called transmission loss and is calculated in dB as

Spherical spreading:

$$TL(R) = 20 * \log_{10}(R/1m)$$

Cylindrical spreading:

$$TL(R) = 10 * \log_{10}(R/1m)$$

We note that using the logarithmic scale the transmission loss is presented in dB and the range R is reference to 1 m. This is important if one wanted to estimate the sound of, say a sperm whale click at 1 km. Assuming spherical spreading, then the geometrical transmission loss is 60 dB ($20 * \log_{10}(1000m/1m)$). The sound pressure of a sperm whale click is at 1 km 1000 times weaker than at 1 m and the intensity decreases by a factor of 1 million (pressure squared).

2.2.2 Sound absorption

Sound that propagates through water experiences not only geometrical spreading but is also absorbed due to physical and chemical properties of the medium. The absorption is highly frequency dependent and the following table gives some characteristic absorption values in sea waters.

Online resources to estimate the sound absorption in sea water may be found at npl.co.uk

Here are some typical values (temperature 13.5°C, 100 m depth)

Frequency [kHz]	Attenuation [dB/km]
0.1	0.001
0.2	0.004
1.0	0.060
2.0	0.130
10.0	0.900
20.0	3.100
100.0	36.600

2.2.3 Propagation modelling

While sound propagation can be approximated with simple formulas, realistic estimation of sound transmission loss requires the use of computer models that follow the sound through the ocean and take into account varying sound speeds and changing bottom profiles. A selection of such models that address different physical approximations can be found on the Web in the Acoustic Toolbox

2.3 Acoustic recording systems

Underwater sound is sensed with hydrophones that convert sound pressure into electric voltages. This electric voltage is then converted by an analog-digital-converter (ADC) into digital numbers to be processed and stored by a computer. The acoustic data are stored on the computer in general as binary data, that is not directly readable directly by humans. A variety of computer programs exist that allow the processing and recording of acoustic data. The most widely used software is PAMGuard that is frequently used for real-time mitigation purposes.

As the sound pressure is typically measured in Pa and the the computer stores the data as digital numbers relative to some ADC reference voltage, a conversion is required before we can interpret the measurements in terms of acoustics . This conversion consist of two steps, the first one converts from stored digital numbers into Volts ('un-doing' the ADC) and the second one converts from Volt to Pascal ('un-doing' the hydrophone).

The conversion factor from Pa to Volt is frequently in the order of one milli Volt per Pascal (1 mV/Pa), which is equivalent to -180 dB // 1V/μPa. In other words, one needs a sound pressure of 1000 Pa to generate 1 Volt with a hydrophone. This value includes some basic electric amplification in the acquisition system. The conversion from Volt to binary depends on analog-digital-converter, which typical converts 1 to 5 Volt into 16 or 24 bit integers.

While there exist a myriad of software for the analysis of acoustic data, in the following we will use basic R language functions to learn about acoustic data and how to carry out basic data processing. This will allow us to handle the most frequent types of sound files (using the wav format) and visualize the important sound features without the use of dedicated software. At the same time we have the flexibility to visualize the data in such ways that convey best the information of interest.

---

3 Acoustic data analysis

3.1 Step 0

When we are confronted with acoustic data, the first step is always to listen to the data. The human brain is very good in signal processing. so why not use it. The first dataset is labelled `stenella.wav` and should be found in the subdirectory `./data` of this pdf and all the R-scripts.

3.2 Step 1

After listen to the data, the first processing step is to load some meta data from the wav file using `script1.R`

```
# script1.R

library('tuneR')

root <- './data'
filename <- file.path(root,'Stenella.wav')
#
head <- readWave(filename,header = TRUE)

fsamp <- head$sample.rate
nchan <- head$channels
nbits <- head$bits
nsamp <- head$samples

print(paste('fsamp=',fsamp,'; nchan=',nchan,'; nsamp=',nsamp))
```

```
## [1] "fsamp= 44100 ; nchan= 1 ; nsamp= 3161851"
```

```
print(paste('nbits=',nbits))
```

```
## [1] "nbits= 16"
```

```
print(paste('duration=',nsamp/fsamp))
```

```
## [1] "duration= 71.6973015873016"
```

The script reads from the wav file only the header and prints some important parameters.

We note from the print-out that the data are sampled with 44.1 kHz, only mono (nchan=1) and over 70 seconds long. We note further that the data are stored as 16 bit integers, that is the values of the data range from 2^{-15} to $2^{15} - 1$.

3.3 Step 2

The next step is then to load the data into memory to be visualized. As 70 seconds may be too long to be plotted quickly (R plot seems to be very slow), we only load the first 6 seconds.

```
#script2.R

to <- 0
te <- 6
data <- readWave(filename,from=to, to= te, units='seconds')

# recall sample rate
fsamp <- data@samp.rate
```

```r
# extract data
xx <- data@left

# generate time axis to time series
N <- length(xx)
tt <- to+seq(0,N-1,1)/fsamp
```

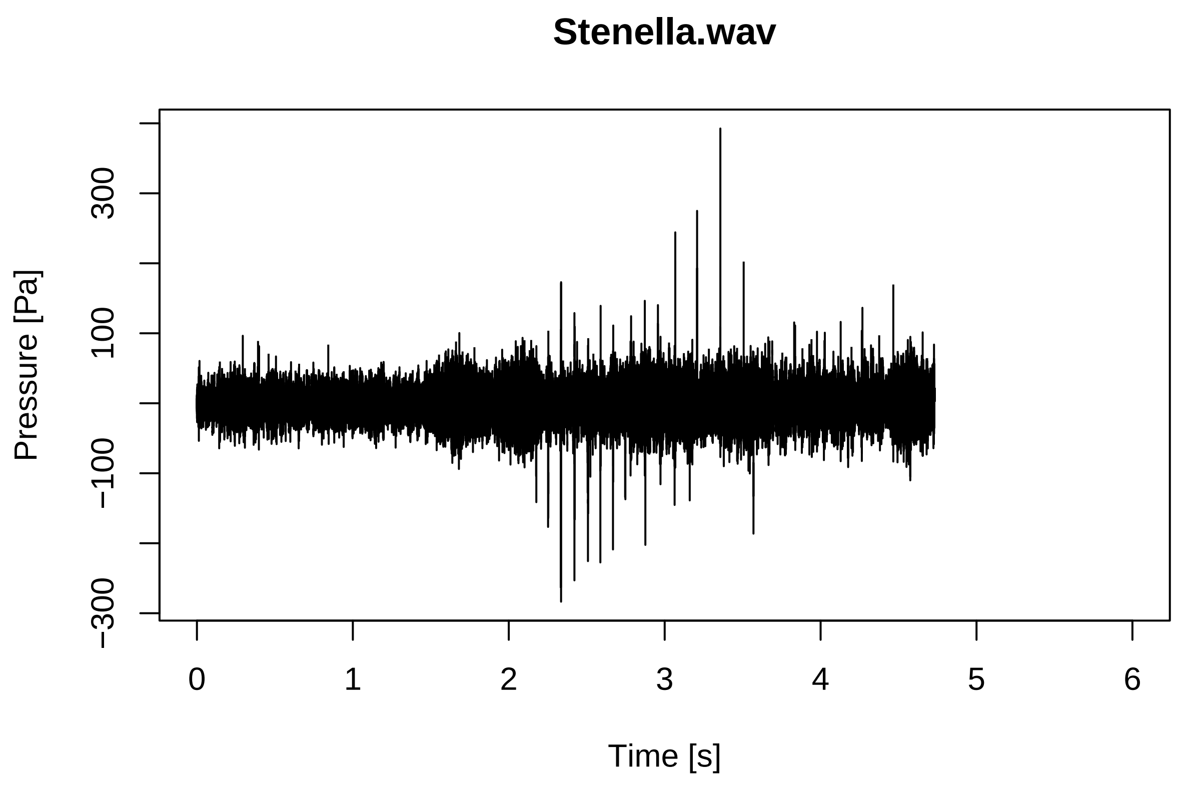
The content of the 6 second wav file is loaded into the `data` object, containing again the sampling frequency and the left/right data of a potential stereo audio file. As we noted earlier, there is only one channel present and we access is as the first or left channel.

As the data are in computer internal format, they should now be calibrated, or transformed into sound pressure. As the correct hydrophone sensitivity and the gain settings of the acquisition system is unknown, we simply assume a reasonable value for `cal_hyd` which is the hydrophone sensitivity including all amplification gains before the analog digital conversion.

```r
# calibrate
cal_hyd <- 1e-3 # V/Pa # assumed effective sensitivity of -180 dB//1V/Pa
cal <- (1/cal_hyd)*(1/2^(nbits-1))

# apply calibration to data
xx <- xx*cal
```

At that point we can plot the data. As straight method is

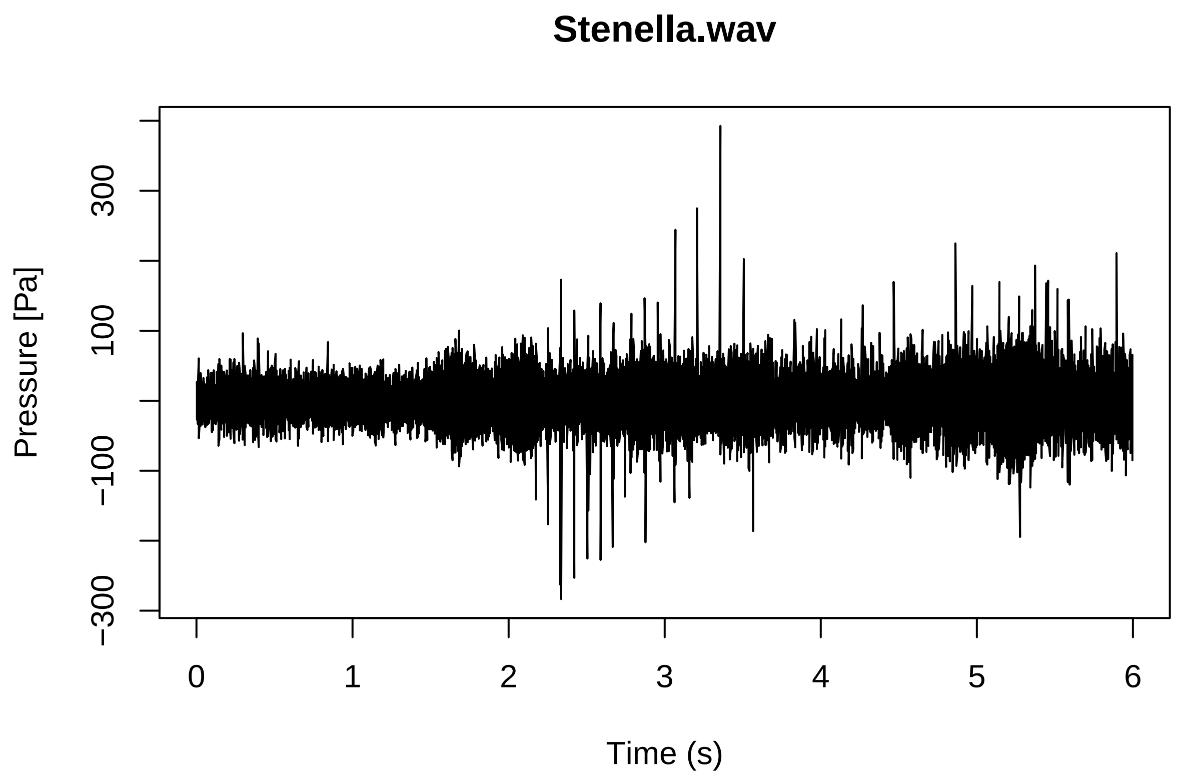
```r
plot(tt,xx,type='l',
      xlab = 'Time [s]', ylab = 'Pressure [Pa]', main = basename(filename))
```



but this is for large time series far too slow.

However, if we subsample the time series then the plotting operation is much faster. For this we create first a function that reduces the amount of data significantly and plot the subsampled data.

```r
#--------------------------------------------------------------------------
psamp <- function(t,x,np)
{
  # generate indices
  i <- seq(1,length(x))

  # cut indices in equal chunks
  i1 <- cut(i,seq(1,length(i),length.out=np+1))

  # split data
  y <- split(x,i1)

  # get average time of chunks
  s <- lapply(split(t,i1),mean)

  # get min,max value of chunks
  z <- sapply(y,fivenum)[c(1,5),]

  # return new time and data vector
  return(list(t=rep(s,each=2), x=(as.vector(apply(z, 2, sample)))))
}

# sub-sample time series plot
d=psamp(tt,xx,1000)

plot(d$t,d$x, type='l',
      xlab = 'Time (s)', ylab = 'Pressure [Pa]', main = basename(filename))
```



We note that the time series shows two type of 'features', some distinct series spikes or transients with pretty regular spacings between 2 and 4 s. Theses are indicative to odontocete clicks. There are not perfectly regular as that would indicate mechanical, i.e. human origin. We have about 10 clicks per second and this is typical to dolphin click trains.

We note further, that there is overall more or less constant 'noise', where no features can be identified, with some shore periods of increased 'noise', e.g. around 2 and 5.4 s. To understand, what these 'features' may be, we plot the data as spectrogram in step 3

3.4 Step 3

We generate a spectrogram of the data using `script3`. After first loading some libraries,

```
#script3.R

library(gsignal)
library(fields)
```

we generate a spectrogram using the spectrogram function of the `gsignal` package.

```
# number of points to use for the fft
nfft=1024

# window size (in points)
win=hanning(256)

# overlap (in points)
overlap=128
```

```r
# create spectrogram
spec <- specgram(x = xx,
                 n = nfft,
                 fs = fsamp,
                 window = win,
                 overlap = overlap)

# discard phase information
P <- abs(spec$S)

# extract time and frequency axis
t <- spec$t
f <- spec$f
```

To obtain the spectrogram, the time series is divided into chunk of short data snippets and then a Fourier transform generates the spectral representation for each of this small data chunks. When stacked in time one obtains the spectrogram.

As the spectrogram transforms a time series into a time-frequency image, we obtain from the spectrogram not only the matrix of spectral values but also the associated time and frequency vectors.

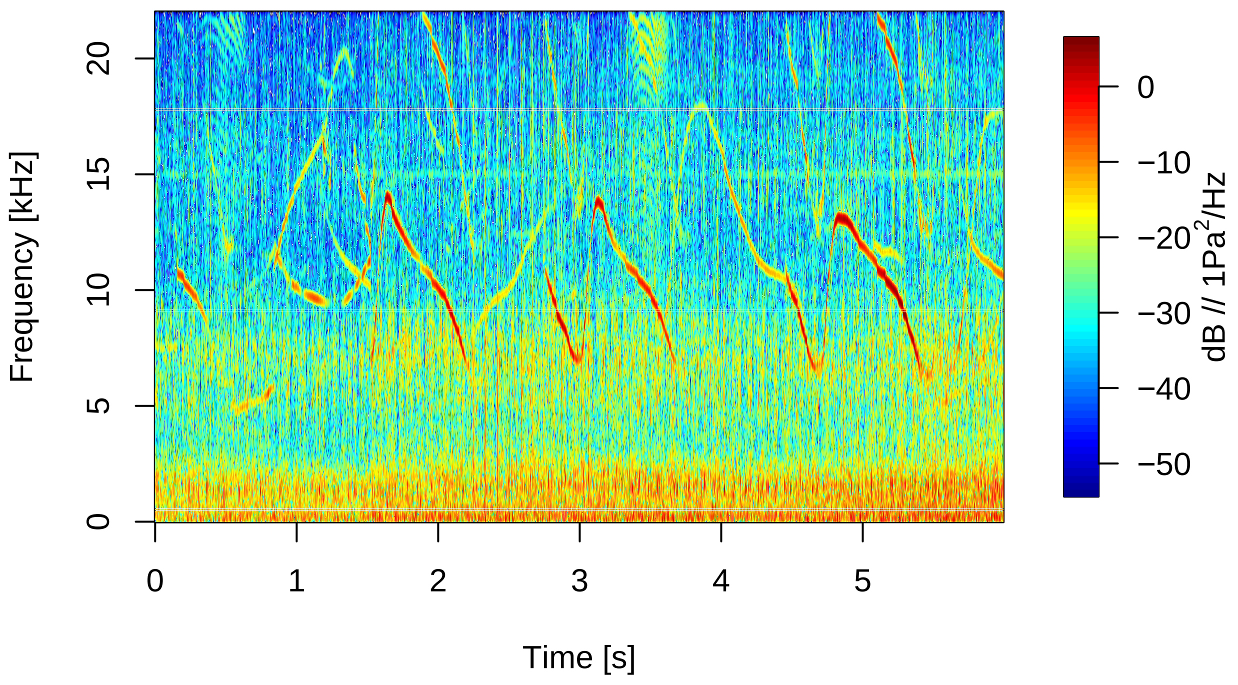
IWhen calling the spectrogram function, we provide in addition to the time series, also parameters that control the processing: in particular, the number of frequencies `n` to be estimated, the sampling frequency `fs`, a temporal window function `window` that weights the data ensuring that the beginning and the end of the data chunk have no or little impact on the spectrum, and how much the different data chunks overlap in time. The length of the window function should always be smaller or equal to the number of frequencies `n`.

For the actual application we are interested in the power spectral density (PSD) and therefore we scale the spectrogram accordingly. This scaling takes into account the sampling frequency and the shape of the weighting window. We correct also for the fact that the `gsignal::spectrogram` reports only half of the spectral values that are expected by the PSD.

```r
# scale to power spectral density (PSD)
scale <- fsamp * as.vector(win %*% win)
P <- P^2/scale

# correct for negative frequencies
nf=dim(P)[1]
n2 <- nf - nf %% 2
P[2:n2,] <- 2*P[2:n2,]
```

At this point we can visualize the spectrogram. We do so in the dB scale. As we have already estimated a power quantity, we obtain the dB values by taking the `10log10` of the PSD.

```r
# plot spectrogram
# convert to dB
L <- 10*log10(P)
Lmax=max(L)

db  <- expression(paste("dB // 1",Pa^2,"/Hz"))
imagePlot(x = t,
      y = f/1000,
      z = t(L),
      zlim=c(Lmax-60,Lmax),
      ylab = 'Frequency [kHz]',
      xlab = 'Time [s]',
      legend.lab=db,
      legend.line=2.5
)
```



The spectrogram of these 6 seconds reveals what the aural inspection of `step 0` indicates, that the data contain a sequence of dolphin whistles. These whistles are not easily identified in the time series plot, but easily detectable in the spectrogram. Contrary, the clicks that are obvious in the time series are difficult to see in the spectrogram.

At about 3.5 s there is a feature at about 20 kHz that is hardly visible in the time series. This is a fast sequence of high-frequency clicks, called buzz, which is typical for inter-dolphin communication. It seems that this feature extends to higher frequency meaning that to cover better the different dolphin sounds, a higher sampling frequency, e.g. 96 kHz, would be more appropriate.

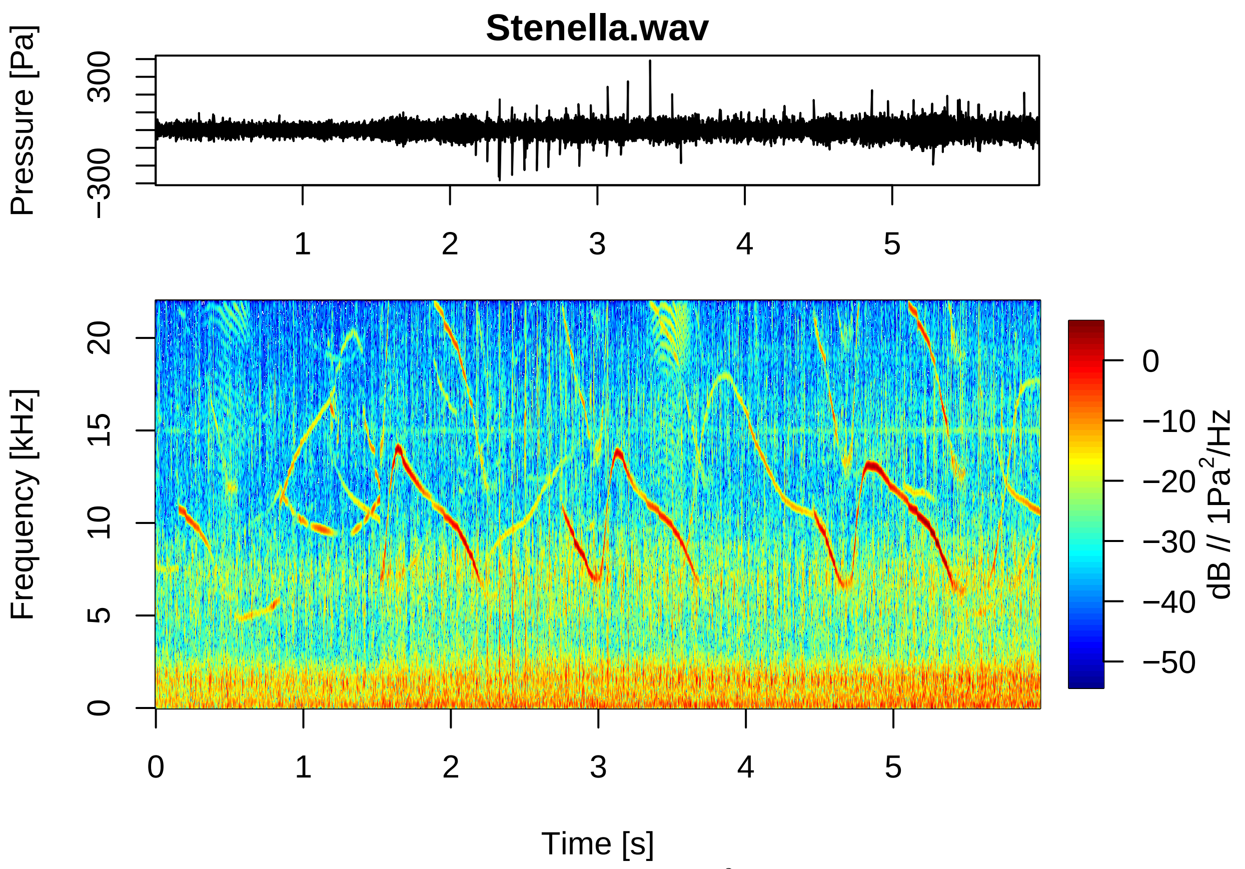
It is therefore useful to combine temporal and spectral information when visualizing bio-acoustic data. We do this with step 4.

3.5 Step 4

In `script4`, we first define an utility function that scales both the time-series plot and the spectrogram in an opportune way to allow direct comparison of temporal and spectral information and then visualize the acoustic data.

```
#script4.R

#dv <- dev.off()
#dev.new(width = 15, height = 10)
```

```r
plot_acoustic <- function(tt,xx,t,f,L,title)
{
  split.screen(rbind(c(0,1, 0.65, 1), c(0, 1, 0, 0.65)))
  par(xaxs="i")

  screen(1)
  # plot time series
  par(mar = c(3, 4, 1.5, 5.5))

  plot(tt,xx, type='l',
        xlab = '', ylab = 'Pressure [Pa]', main = title)

  screen(2)
  # plot spectrogram
  par(mar = c(4, 4, 0, 1))

  db  <- expression(paste("dB // 1",Pa^2,"/Hz"))
  Lmax=max(L)

  imagePlot(x = t,
      y = f/1000,
      z = t(L),
      zlim=c(Lmax-60,Lmax),
      ylab = 'Frequency [kHz]',
      xlab = 'Time [s]',
      legend.lab=db,
      legend.line=2.5
  )
}

d <- psamp(tt,xx,1000)        # sub-sample data

plot_acoustic(d$t,d$x,t,f,L,basename(filename))
```



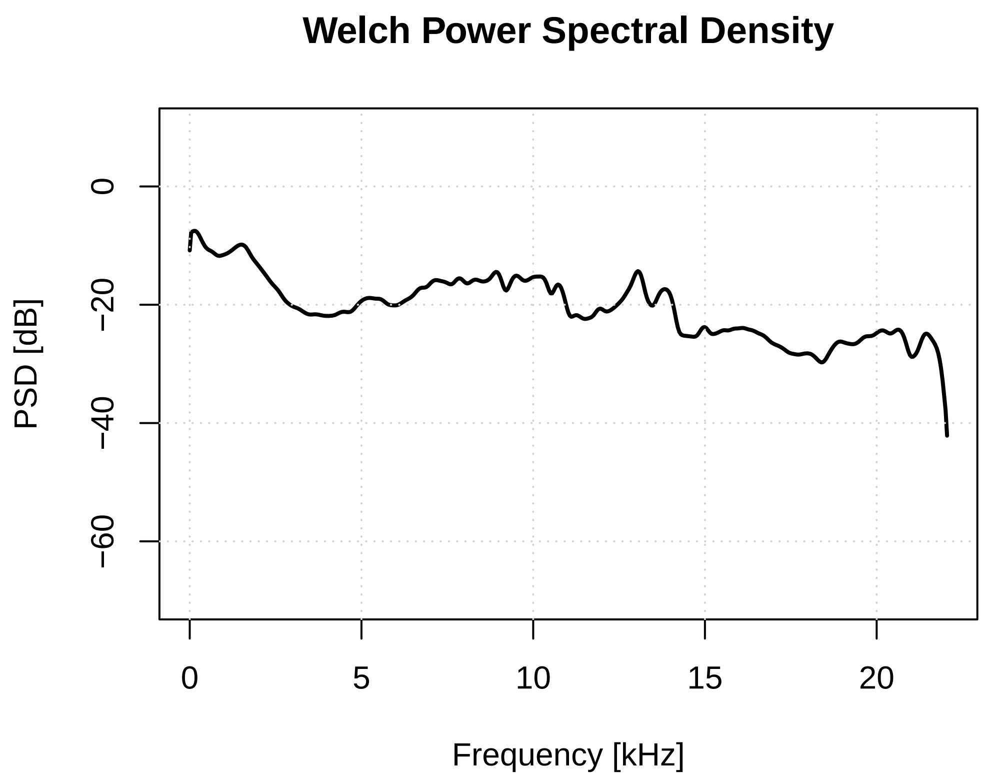
One may note that the PSD spectrogram is plotted as `dB//1Pa`^{2}`/Hz`, having 1 Pa as reference value and not 1 μPa as suggested in the physics part in this text. If one wanted to change this, one only needs to multiply the PSD by 10^6, or add 120 to the dB value. In other words 1`dB//1Pa`^{2}`/Hz` is equal to 120`dB//1`μ`Pa`^{2}`/Hz`.

3.6 Step 5

Very often we are not interested in the detailed spectrogram, but wanted to obtain an averaged spectrum, also called periodogram. This is not simply a Fourier transform of the whole time series, but effectively a temporal average of a spectrogram. In general, one uses a method that is suggested by Welch and is implemented as `pwelch`in the `gsignal` package.

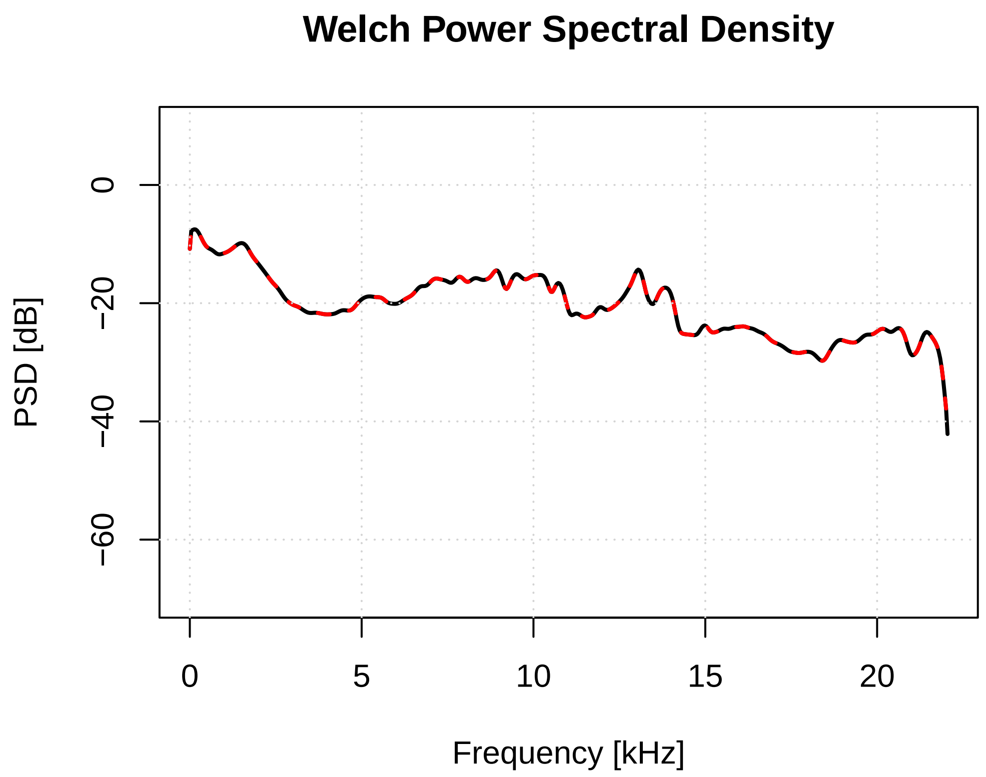
```
# script5

library('gsignal')

# use xx from script2.R
pxx <- pwelch(xx,fs=fsamp,window=hanning(256),nfft=1024)

#dv <- dev.off()
#dev.new()

par(mar=c(5.1, 4.1, 4.1, 7.1),xpd=TRUE)
plot(pxx$freq/1000,10*log10(pxx$spec),type='l', ylim=c(-70,10),lwd=2,
      main="Welch Power Spectral Density",
      xlab="Frequency [kHz]",ylab="PSD [dB]")

par(xpd = FALSE)
```

```r
grid()
```



When calling the `pwelch` function we specify the same window function and fft length as we used for the spectrogram. This will allow easier comparisons next.

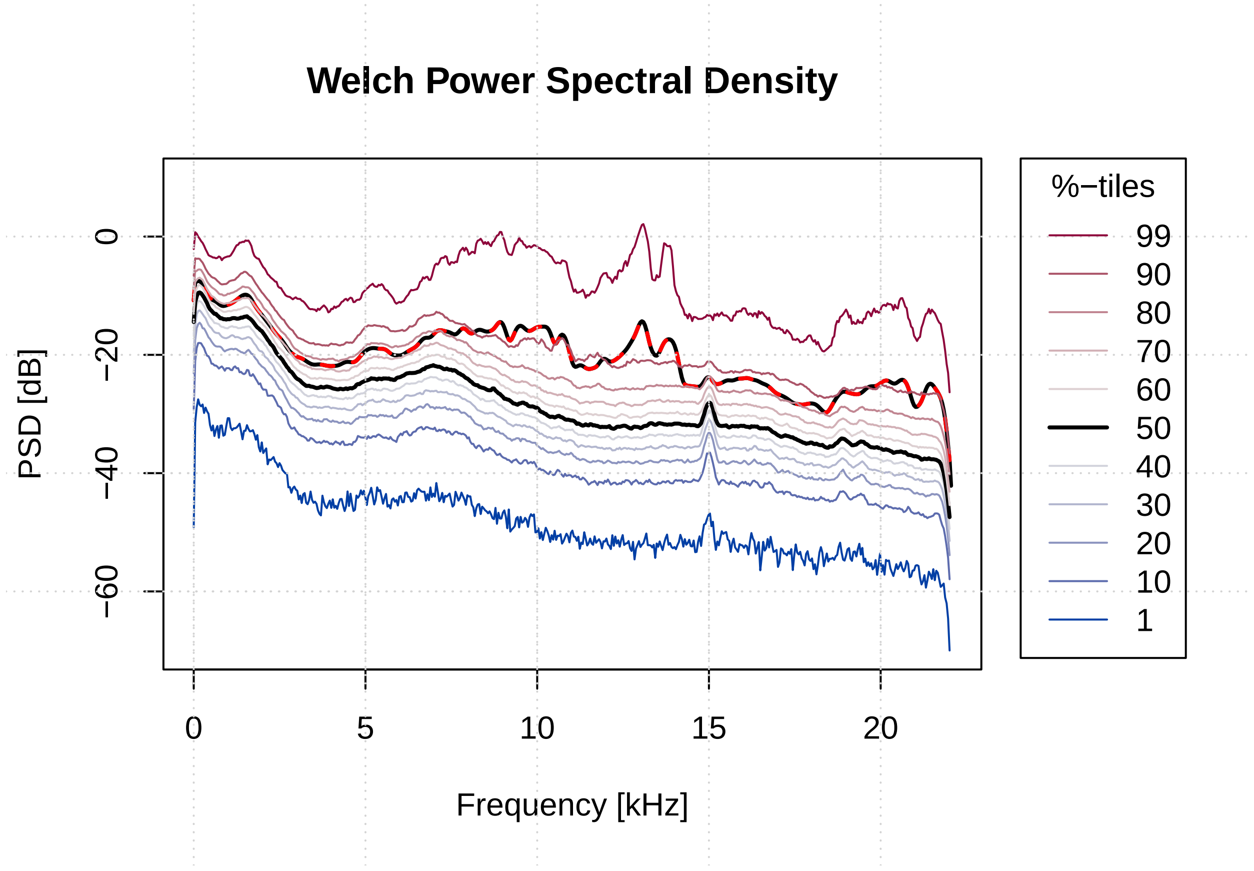
```r
#-------------------------------------------------------------
#To show that the periodogram is the temporal average of the spectrogram,
#we simply average the spectrogram (PSD values in linear scale and not log (dB) domain)

# use P from script3.R
V <- apply(P,1,mean)
V <- 10*log10(V)

# replot previous data
par(mar=c(5.1, 4.1, 4.1, 7.1),xpd=TRUE)
plot(pxx$freq/1000,10*log10(pxx$spec),type='l', ylim=c(-70,10),lwd=2,
      main="Welch Power Spectral Density",
      xlab="Frequency [kHz]",ylab="PSD [dB]")

lines(f/1000,V,lty=2,lwd=2,col='Red')
par(xpd = FALSE)
grid()
```



Obviously, the temporal mean of the spectrogram (red-dashed line) corresponds exactly to the periodogram estimated with `pwelch`.

```
#--------------------------------------------------------------
#Now, what about the temporal statistics of the different frequency bins?
#This is important if we wanted to assess the acoustic activity present in the data.
#For this we simply run for all frequencies a quantile analysis and add them to the plot.

quants=c(0.01,seq(1,9)/10,0.99)

U <- apply(P,1,quantile,quants)
U <- 10*log10(U)

cols <- hcl.colors(nrow(U),"Blue-Red")
lw <- rep(1,nrow(U))

# emphasize median (50 percentile)
lw[6] <-2
cols[6] <-'#000000'

# replot previous data
par(mar=c(5.1, 4.1, 4.1, 7.1),xpd=TRUE)
plot(pxx$freq/1000,10*log10(pxx$spec),type='l', ylim=c(-70,10),lwd=2,
      main="Welch Power Spectral Density",
      xlab="Frequency [kHz]",ylab="PSD [dB]")

lines(f/1000,V,lty=2,lwd=2,col='Red')
```

```
# plot all percentiles
ret <- sapply(seq(1,11),function(i){lines(x=f/1000,y=U[i,],col=cols[i],lwd=lw[i])})
grid(nx = NULL, ny = NULL, col = "lightgray", lty = "dotted")
# add legend
legend("topright", inset=c(-0.25,0),
       legend=rev(quants)*100, col=rev(cols),lwd=rev(lw), title="%-tiles")
par(xpd = FALSE)
grid()
```



This plot shows the periodogram and different percentiles plots together. One can deduce that most percentiles (10% to 80%) are pretty parallel indicating that they contain common features, like the tonal at 15 kHz and increased background noise between 5 and 10 kHz. The 99%-tile, which should be close to the maximal value, however, shows different features, for example in the 10 to 15 kHz region. It is therefore fair to say that the difference between the 99%-tile and the median (50%-time) is an indication of temporary acoustic activity.

---

4 Bio acoustics

Underwater sound may not only be useful to describe the underwater soundscape or to improve abundance estimates, but may be also very important to learn about the underwater behaviour of cetaceans. In the following we will use data recorded from a sperm whale to learn more about their diving behaviour. For this a couple of wav files recorded in the Mediterranean Sea will be analysed. The selected files show the first echolocation clicks of the sperm whale after initiating a deep foraging dive.

4.1 Step 0

Again the first step is to listen to one of the data files: e.g. `sw01_275_08_37.wav`. What we hear is sequence of clicks separated by about 1 s that are characteristic for sperm whales. We do not hear any whistling or other tonal sound. The interval between the clicks is about 1 s, which is typical for sperm whale echolocation and much longer than the click interval we have seen for the striped dolphins.

4.2 Step 1

Next, we load the header of the wav file and inspect especially the sampling rate, which is a little bit lower than the one of the recordings analysed before.

```r
#script11

library('tuneR')

root <- './data'
filename <- file.path(root,'sw01_275_08_37.wav')
#
head <- readWave(filename,header = TRUE)
fsamp <- head$sample.rate
nchan <- head$channels
nbits <- head$bits
nsamp <- head$samples
print(paste('fsamp=',fsamp,'; nchan=',nchan,'; nsamp=',nsamp))
```

```
## [1] "fsamp= 31250 ; nchan= 1 ; nsamp= 1875000"
```

```r
print(paste('nbits=',nbits))
```

```
## [1] "nbits= 16"
```

```r
print(paste('duration=',nsamp/fsamp, 's'))
```

```
## [1] "duration= 60 s"
```

4.3 Step 2

As the data is 1 minute long we load the first 10 seconds and calibrate the data.

```r
# script12

#load data
#-----------------------------------------------------
to <- 0
te <- 10
data <- readWave(filename,from=to, to= te, units='seconds')
#
# recall sample rate
```

```
fsamp <- data@samp.rate
#extract data
xx <- data@left
#
N <- length(xx)
tt <- to+seq(0,N-1,1)/fsamp
#
# calibration value
cal_hyd <- 0.1 # V/Pa # reported effective sensitivity of -140 dB//1V/uPa
cal <- (1/cal_hyd)*(1/2^(nbits-1))

xx <- xx*cal
```

Next, we plot the time series and spectrogram of the data to verify the presence of echolocation clicks and the absence of whistles. For this, we use the functions defined in an earlier script adding a function that sub-samples also the spectrogram for faster plotting.

```
# process data
#-------------------------------------------------------
library(gsignal)

#load local functions
source("./bioacustico/psamp.R")
source("./bioacustico/plot_acoustics.R")

# define a new function to sub-sample the spectrogram
msamp <- function(t,u,na)
{
  # get dimension of matrix
  nu=dim(u)
  nf=nu[1]
  nd=nu[2]

  # generate 3d matrix
  v=array(u,dim=c(nf,na,nd/na))

  # get maximum value of 2nd dimension
  w=apply(v,c(1,3),max)

  # generate new time vector
  s=t[seq(1,length(t),na)]

  # construnct return list and return
  ret=list()
  ret$t <- s
  ret$m <- w
  return(ret)
}
```

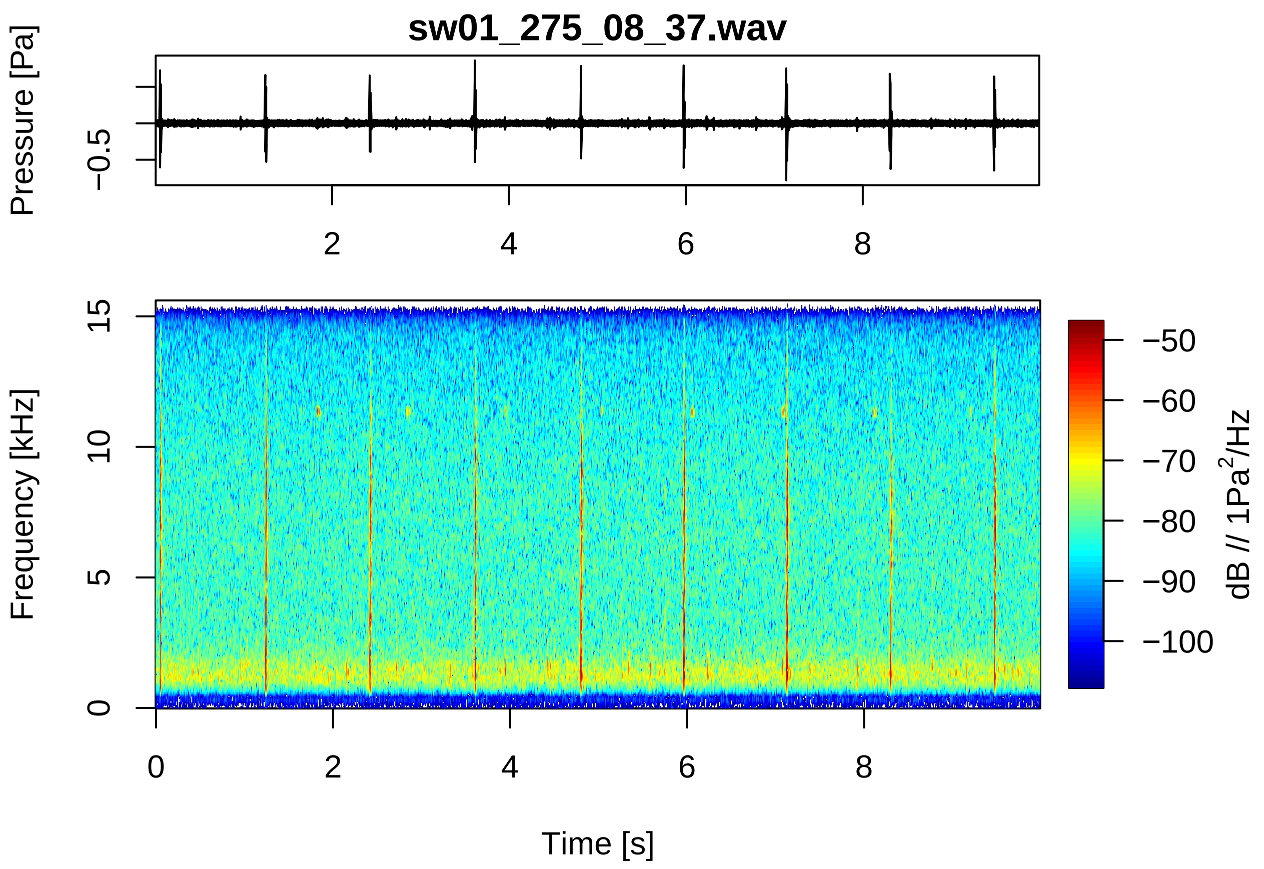
After these definition we calculate the spectrogram, similar to what we have done for the dolphin data.

```
#-------------------------------------------------------

# number of points to use for the fft
nfft=1024
```

```
# window size (in points)
win=hanning(256)

# overlap (in points)
overlap=128

# create spectrogram
spec <- specgram(x = xx,
                 n = nfft,
                 fs = fsamp,
                 window = win,
                 overlap = overlap)
# discard phase information
P <- abs(spec$S)

# config time axis
t <- spec$t
f <- spec$f

# scale to power spectral density (PSD)
scale <- fsamp * as.vector(win %*% win)
P <- P^2/scale

# sub-sample timeaxis
M=msamp(t,P,2)

# convert to dB
Q <- 10*log10(M$m)
T <- M$t
```

Finally we visualize both the time series and the spectrogram.

```
# visualize data
#---------------------------------------------
library(fields)

d=psamp(tt,xx,1000)          # sub-sample data

#dev.new(width = 15, height = 10)

plot_acoustic(d$t,d$x,T,f,Q,basename(filename))
```



The time series shows clearly 9 short transients that are more powerful than the background noise and could be easily heard by listening to the wav file. These transients are sperm whale echolocation clicks and also clearly visible in the spectrogram where furthermore no tonal sound or whistles may be seen.

4.4 Step 3

What we have now is a sequence of clear sperm whale echolocation clicks and it may be interesting to have a closer look to these echolocation clicks. For this we must first extract each of these clicks. This is done by obtaining the exact time when a click occurs, that is by detecting the click. What is easy for the human brain needs some small algorithm, a click detector, that is implemented in the beginning of `script13`.

```
# script 13

detector <- function(yy,TH,win1,win2)
{
  # click threshold detection
  #-----------------------------------------
  idet <- (yy>TH)
  #
  # prune detections
  #-----------------------------------------
  jj=which(idet)
  nj=length(jj)
  kk=jj[2:nj]-jj[1:nj-1]

  mm=which(kk>win1) # win1 separation between detections
  jdet <- c(jj[1],jj[mm+1],jj[nj]) # vector of detection boundery indices
  nj <- length(jdet)
  dd <- 0*jdet[1:nj-1]
  for (ii in seq(1,nj-1))
  { j1<- jdet[ii]
    j2<- j1+win2
    nn <- j1
    ymax <- 0
    for(nn in seq(j1,j2))
    { if(yy[nn]>ymax)
      { ymax <- yy[nn]
        jj <- nn
      }
    }
    dd[ii]=jj
  }

  # generate return list and return
  ret=list()
  ret$det=dd
  ret$idet=idet
  return(ret)
}
```

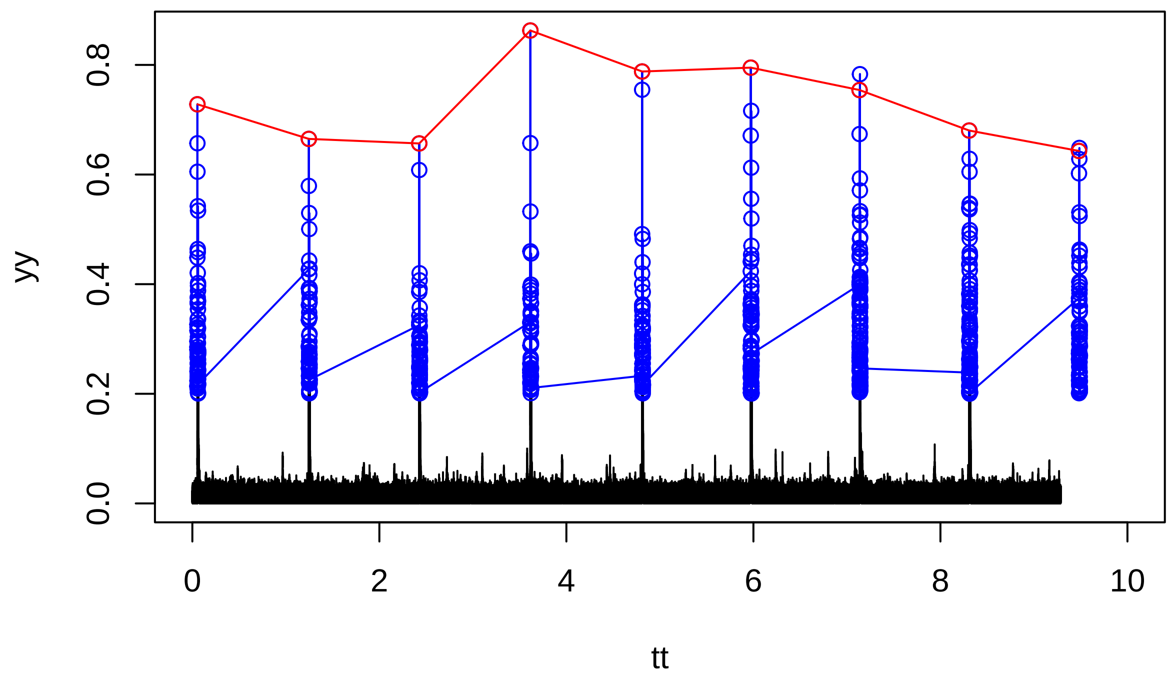
The detector first considers as interesting data all values that exceed a given threshold. This threshold is chosen such that it separates noise from interesting data. Next, the threshold crossings are split into different detections using a minimal separation (again chosen a-priori) between consecutive detections. Finally, the first local maxima found after the onset of the detection is used for the location of the click.

This is a very simple algorithm with three parameters and obviously, in the literature a variety of different detection methods can be found. Especially, for more complicated cases, multiple animals, disturbances and low pressure clicks, more sophisticated algorithms are suggested.

As said, when performing statistical analysis of spectrograms, the difference between the 99%-tile and the median would be a good indicator for the presence of short acoustic activities, indicating that a more sophisticated spectral based method could be an interesting extension of detection algorithm. However, such approaches are beyond the scope of this basic text.

In the following there is a visualisation of the time series and the different steps of the detection process.

```
yy <- abs(xx)

plot(tt,yy,type='l')

D=detector(yy,0.2,0.5*fsamp,10)

idet=D$idet
lines(tt[idet],yy[idet],type='o',col='blue')

icl <- D$det
tcl <- tt[icl]
ycl <- yy[icl]

# plot click detections
#-------------------------------------------
lines(tcl,ycl,type='o',col='red')
```



The plot shows in black the absolute pressure of the click, in blue all threshold crossings and in red the resulting pressure values of the detected clicks. As the maximum pressure value of a click may not be close to the beginning, the click but some time later.

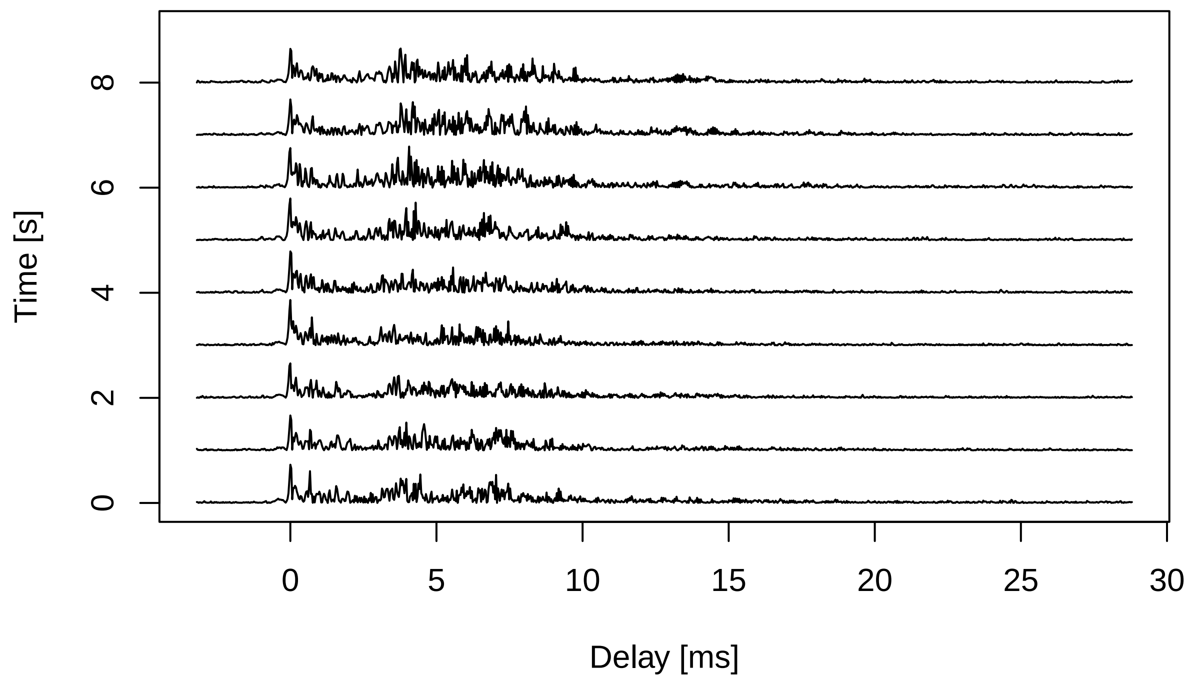
4.5 Step 4

Now we are ready to look into more details of the detected clicks. For this we stack somewhat more than 30 ms (approximately from -3 ms to 27 ms) as done in `script14.R`.

```
# script14

#dv<-dev.off()
#dev.new()
i1 <- icl[1]-100
i2 <- i1+1000
plot((tt[i1:i2]-tcl[1])*1000,yy[i1:i2],type='l',ylim=c(0,9),
     xlab='Delay [ms]', ylab='Time [s]')


ndet <- length(icl)
print(paste('ndet=',ndet))
```

```
## [1] "ndet= 9"
```

```
for (ii in seq(2,ndet))
{ i1 <- icl[ii]-100
  i2 <- i1+1000
  lines((tt[i1:i2]-tcl[ii])*1000,yy[i1:i2]+(ii-1),type='l')
}
```



We note first that all clicks are perfectly lined up and that after an initial pulse more signal comes between 3 ms and 10 ms, which look similar but not exactly equal. Something is going on here. It turns out that the 9 clicks shown here are not sufficient to learn more from the data, so in the next step we process much more data.

4.6 Step 5

At this point we load this and 4 other wav files and repeat the same detection and stacking procedure. Again, we load the header from the first file assuming that sampling frequency, numbers of bits and channels are the same for all files. `script15` has the option to load only the first 5 files or all files in the data directory,

```
# script15

library('tuneR')

source('./bioacustico/detector.R')

#==========================================================================
root <- './data'
filename <- file.path(root,'sw01_275_08_37.wav')
#
head <- readWave(filename,header = TRUE)
fsamp <- head$sample.rate
nchan <- head$channels
```

```
nbits <- head$bits
nsamp <- head$samples
print(paste('fsamp=',fsamp,'; nchan=',nchan,'; nsamp=',nsamp,'\n'))
```

```
## [1] "fsamp= 31250 ; nchan= 1 ; nsamp= 1875000 \n"
```

```
print(paste('nbits=',nbits))
```

```
## [1] "nbits= 16"
```

```
print(paste('duration=',nsamp/fsamp, 's\n'))
```

```
## [1] "duration= 60 s\n"
```

As the size of the 5 wav files is only a couple of MBytes, we load next all data into memory and calibrate them.

```
# calibration value
cal_hyd <- 0.1 # V/Pa # assumed effective sensitivity of -140 dB//1V/uPa
cal <- (1/cal_hyd)*(1/2^(nbits-1))

#list of data
data_root='./data'
fnames=list.files(path=data_root,pattern='sw01_275')

if(1) # change to 'if(0)' to use all available files in data directory
{
  nfiles=5
  tdel=30
} else
{
  nfiles=length(fnames)
  tdel=120
}

#---------------------------------------------------
# load data into single vector
xx=c()
for(fname in fnames[1:nfiles])
{ print(fname)
  data <- readWave(file.path(data_root,fname))
  #extract data
  xx <- c(xx,data@left)
}
```

```
## [1] "sw01_275_08_37.wav"
## [1] "sw01_275_08_38.wav"
## [1] "sw01_275_08_39.wav"
## [1] "sw01_275_08_40.wav"
## [1] "sw01_275_08_41.wav"
```

```
#
# recall sample rate
fsamp <- data@samp.rate
#
N <- length(xx)
tt <- seq(0,N-1,1)/fsamp
```

```
yy <- xx*cal
```

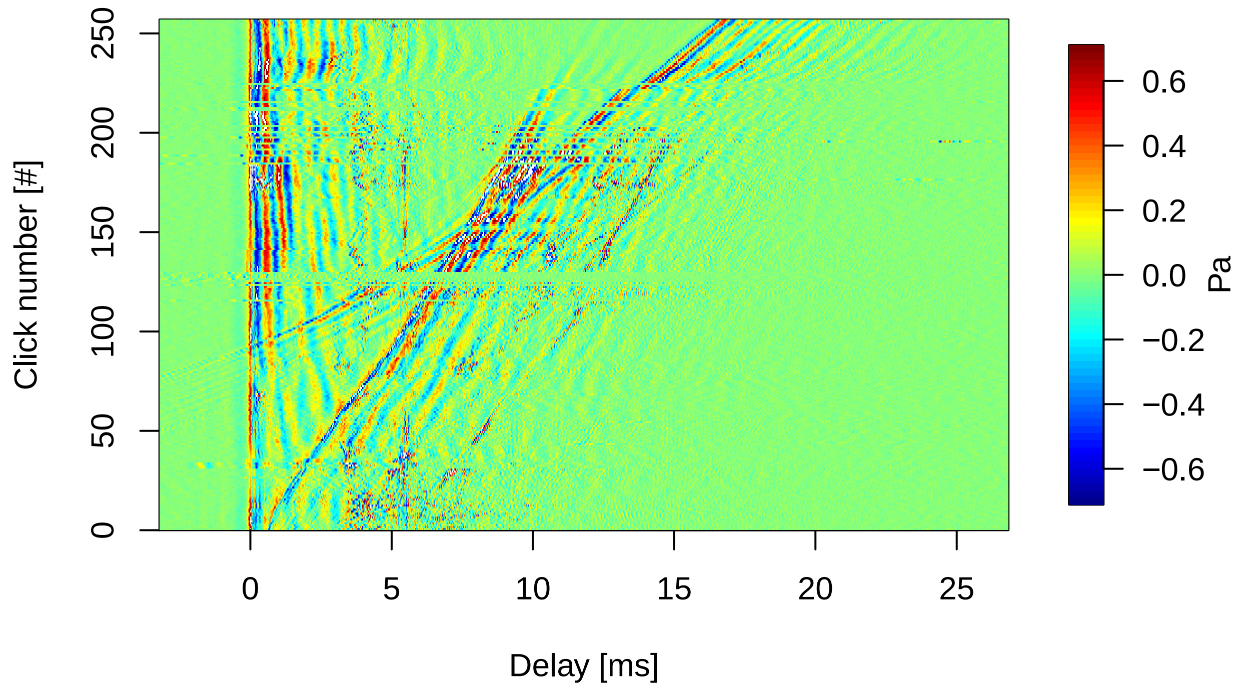
At this point we run the detector.

```
D=detector(yy,0.2,0.5*fsamp,10)

icl <- D$det
ndet <- length(icl)
tcl <- tt[icl]
ycl <- yy[icl]
ici <- c(0,tcl[2:ndet]-tcl[1:ndet-1])

print(paste('ndet=',ndet))
```

```
## [1] "ndet= 257"
```

Finally we plot the result. There are two options for the visualization depending on the number of detected clicks to be stacked. If this number is smaller than, say, 150 clicks, then the same type of plot shown before is done. Otherwise the clicks are shown a colour image. Different to the previous step (`script14.R`) the time series is plotted as positive/negative and not as absolute values. In cases where there are more than 1000 detections the absolute value of the time series is again plotted to improve visualization.

```
#
#==============================================================
library(fields)

nsdel <- round(tdel*fsamp/1000)
if (ndet<150)
{
  # stack detections
  #dev.new(width = 10, height = 7)
  ymax <- max(yy)

  i1 <- icl[1]-100
  i2 <- i1+nsdel
  plot((tt[i1:i2]-tcl[1])*1000,yy[i1:i2]/ymax+tcl[1],type='l',col='blue',
                         ylim=c(0,tcl[ndet]+1),
                         xlab ="Delay [ms]", ylab = "Time [s]")

  for (ii in seq(2,ndet))
  { i1 <- icl[ii]-100
    i2 <- i1+1000
    lines((tt[i1:i2]-tcl[ii])*1000,yy[i1:i2]/ymax+tcl[ii],type='l',col='blue')
  }
} else
{ # use imageplot
  M <-  array(dim=c(nsdel+1,ndet))

  idel=0
  to <- idel*nsdel/fsamp*1000
  #
  for (ii in seq(1,ndet))
  {  i1 <- icl[ii]+idel*nsdel-100
     i2 <- i1+nsdel
     M[,ii]=yy[i1:i2]
  }

  y <- seq(0,ndet)
  x <- seq(-100,nsdel-100)/fsamp*1000

  if(ndet>1000)  M <- abs(M)

  clim=c(max(-0.7,min(M)),min(0.7,max(M)))

  #dev.new(width = 15, height = 7)
  imagePlot( M,
      x=x,
      y=y,
      ylab = 'Click number [#]',
      xlab = 'Delay [ms]',
      legend.lab='Pa',
      legend.line=2.5,
      zlim=clim
  )
}
```



As we have more than 250 detections, a colour plot shows the result (yellow-red are positive and cyan-blue are negative values), Again, all detections are lined up precisely.

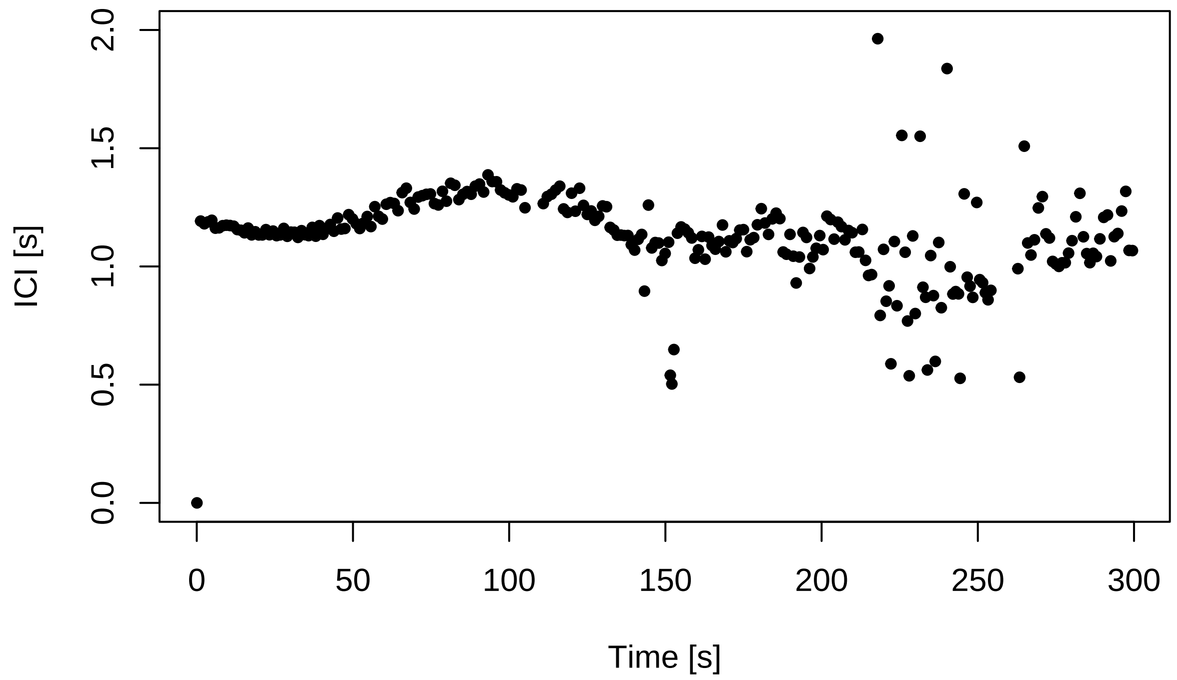
We note a series of interesting features:

- there are low frequency oscillations (red/blue sequences) following the initial pulse that go up to 10 ms
- the frequency of these oscillations increase with click numbers

- there are two additional sequences that sloping off the detected click.
  - one starting immediately with click one
  - the other one starting with click 100
- This multipath related repetition of clicks allow under certain circumstances estimation of whale range and depth.
- In addition to the precise line-up of the initial click, there is a parallel line about 5.1 ms. This line is especially visible at detections ranging from #120 to #200. This is the internal reflection of the sperm whale click by the frontal sac at the scull and also called Inter-Pulse-Interval (IPI).

Having estimated the time when the sperm whale emitted the echolocation clicks, we typically plot the time between consecutive clicks, the inter-click-interval, or ICI.

```
#dev.new()
plot(tcl,ici,pch=20,ylim=c(0,2),xlab='Time [s]', ylab='ICI [s]')
```



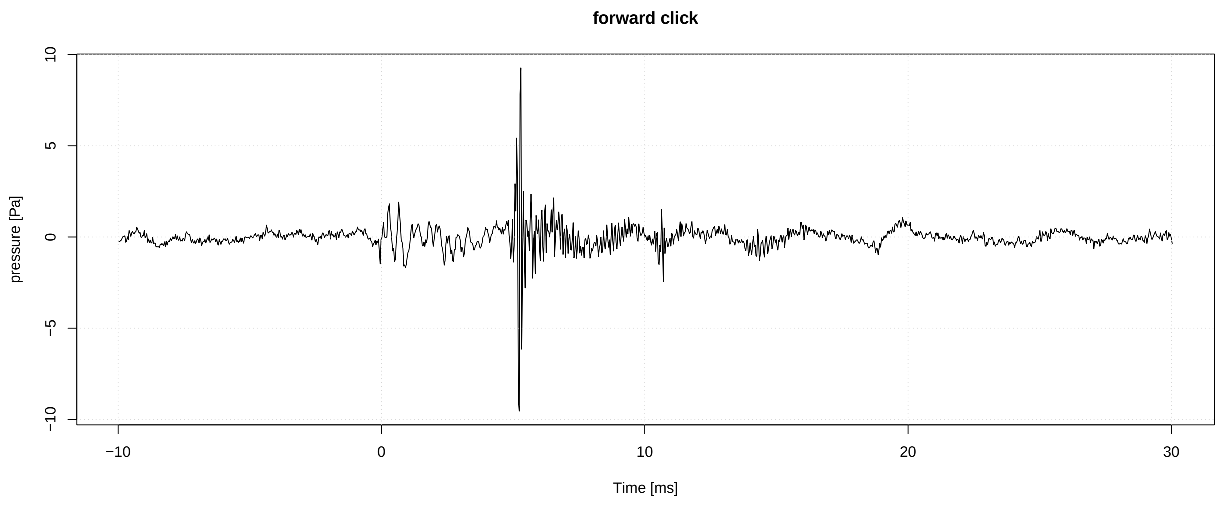
The ICI is at the beginning pretty predictable, but as the time goes by and the whale is moving more horizontally, the ICI may become more variable.

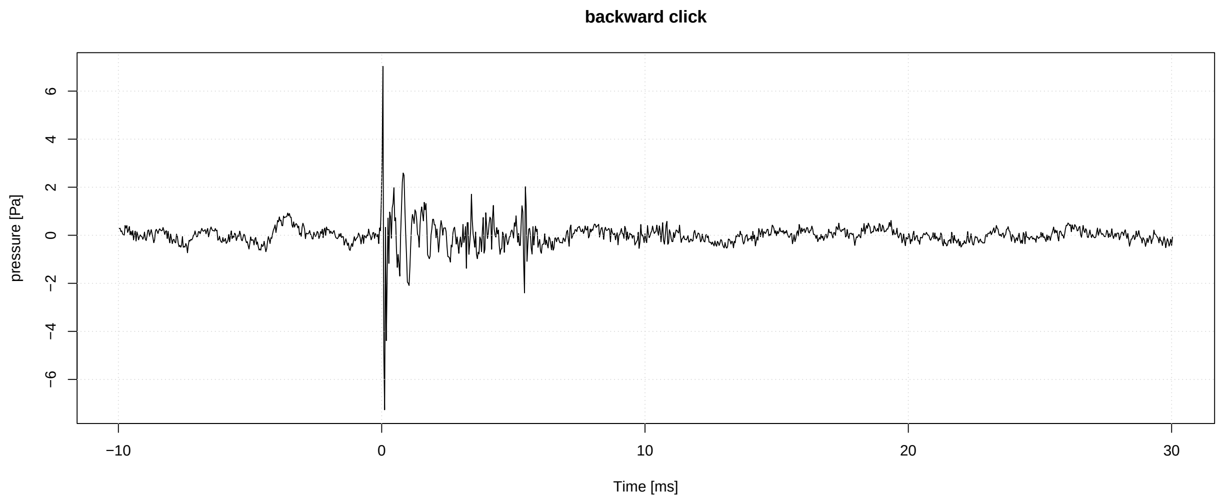
Please note that NO human interaction occurred to ‘clean-up’ the detections, and consequently, there can be additional variation in the ICI due to false or missed detections. More improved detection algorithms may remove false and avoid missed detections.

4.7 Step 6

The internal pulse structure is characteristic for sperm whale clicks. In the final step we plot two isolated echolocation clicks, one observed in front of the whale and one from behind. The two clicks are given as two different file in the `./data` directory

```r
library(tuneR)

cal <- (1/0.1)*(1/2^15)

dat1 <- readWave('./data/forward_click.wav')
x1 <- dat1@left*cal
t1=-10+seq(1,length(x1))/dat1@samp.rate*1000

dat2 <- readWave('./data/backward_click.wav')
x2 <- dat2@left*cal
t2=-10+seq(1,length(x2))/dat2@samp.rate*1000

par(mfrow = c(2, 1))

plot(t1,x1,type='l',xlab='Time [ms]',ylab='pressure [Pa]',main='forward click')
grid()

plot(t2,x2,type='l',xlab='Time [ms]',ylab='pressure [Pa]',main='backward click')
grid()
```





These two visualizations confirm what is called the bent-horn hypothesis of sperm whale echolocation clicks. The top click is seen in front of the animal, where the original pulse generated by the phonic lips, called also `P0`, is weak (shielded by the distal air sac) and the primary echolocation pulse (after reflection from the frontal air sac and leaving the junk area in forward direction), called also `P1`, is very strong. When seen from behind the animal, the original pulse is the strongest pulse. Common to both clicks is the low frequency pressure oscillation after the original Pulse. The very weak pulse at about 10.5 ms is also called `P2`.

More information on the relationship between sperm whale click structure and anatomy as discovered by bio-acoustic analysis can be found in the two publications in the `./doc` folder.

Overall, can easily conclude that:

- as the IPI is related to the overall body length and the size of this whale can be estimated from the acoustics and results here in about 12 m.
- the unique way of sperm whale click generation allows an estimation of the whale orientation in space (e.g whale orients toward hydrophone or away from it)
- as the whale dives, boundary reflections occur with increasing distances from the echolocation click repeating similar pulse structure.
- multiple boundary reflection sequences indicate complex propagation environment.
- clicks carry also low frequency energy the duration of which reduces as the whale dives
- the ICI of sperm whale echolocation clicks is in the order of 1 s, but may vary significantly during the foraging process.
- foraging attempts are characterized by a decrease of ICI (up to 50 clicks per second). However, due to decreased sound pressure of these fast ‘foraging’ clicks, they are difficult to detects, especially if the whale is at distance.

5 Summary

Acoustics

- Sound propagates as pressure wave through gas,liquid and solids.
- The sound speed in water is about 1500 m/s and 4.5 times faster than in air (330 m/s).
- Sound speed increases with temperature,depth and salinity.
- Typical frequencies of cetacean sound vary from 20 Hz to over 100kHz,
- The sound pressure is measured in units of Pascal (Pa) and is typically much smaller than the ambient pressure of 100 kPa.
- Due to wide range of pressure values, they are frequently expressed as logarithmic ratio relative to a reference pressure, which in underwater scenarios is typically 1μPa, for terrestrial application 20 μPa.
- The same sound pressure corresponds to much less sound intensity in water than in air (less damaging in water than in air).
- Sound pressure decreases as function of distance due to spreading of the sound energy and absorption.
- Typical spreading laws are spherical spreading and cylindrical spreading when propagation is constrained by surface and bottom.
- Absorption depends strongly on frequency and weakly on temperature, depth and salinity.

Acoustic Analysis

- Hydrophones convert pressure to Volts and Analog-Digital-Converter (ADC) convert Volt to digital numbers as used by computers.
- Calibration is the conversion from digital numbers used by computer to pressure values in Pascal.
- It is important to listen to sound files as the human brain is very good in audio signal processing, but sound outside human hearing range can only visualized using computer programs
- Sound files can be plotted as time series or as spectrograms.
- spectrograms show in form of an image the temporal variation of the spectral components of the sound.
- Transients, like clicks, are easier to see in the series plots, while tonal sound, like dolphin whistles, are easier to see and identify in spectrograms.
- Statistical analysis along the temporal axis of spectrograms indicate that the median (50%-tile) is a good proxy for the background noise.
- Difference between 99%-tile and the median is a good indicator if short term acoustic activities.

Sperm Whale Bio-Acoustics

- ICI may vary over time complicating the detection process.
- The IPI is related to the overall body length and the size of this whale can be estimated from the acoustics.
- The ICI of sperm whale echolocation clicks is in the order of 1 s, but increases during foraging attempts.
- The unique way of sperm whale click generation allows an estimation of the whale orientation in space.
- As the whale dives, boundary reflections occur with varying time distances from the primary click.

Signal processing

- Extraction of echolocation clicks require click detector that determine the onset of the clicks.
- A detector separates clicks from background noise, typically using a threshold.
- As different clicks are separated in time, a-priori knowledge on the inter-click-interval is needed.
- Smart detectors are needed to address variability in echolocation click trains.
- To improve computer response, it is wise to limit graphics to a limited number of data points or to resample/compress the data.