seewave

A very short introduction to sound analysis for those who like elephant trumpet calls or other wildlife sound

Jérôme Sueur
Muséum national d'Histoire naturelle
CNRS UMR 7205 OSEB, Paris, France
http://sueur.jerome.perso.neuf.fr

April 9, 2013

This document is a very brief introduction to sound analysis principles. It is mainly written for students starting with bioacoustics. The content should be updated regularly. Demonstrations are based on the package seewave.

Contents

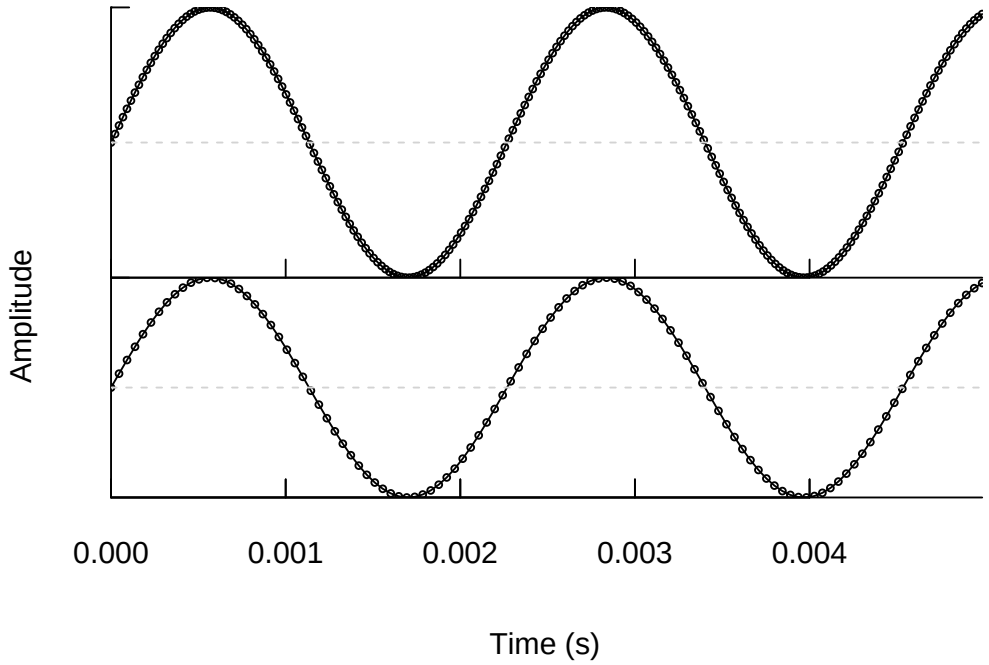

Figure 1: Digital sound is a discrete process along a time scale: the same sound sampled at two different rates: 44.1 kHz (above) and 22.05 kHz (bottom) respectively.

1 Digitization

1.1 Sampling

Digital recording is not a continuous but a discrete process of data acquisition. Sound is recorded through regular samples. These samples are taken at a specified rate, named the sampling frequency or sampling rate f given in Hz or kHz. The most common rate is 44,100 Hz = 44.1 kHz but lower rate can be used for low frequency sound (*e.g.* 22.05 kHz) or higher rate can be used for high frequency sound (up to 192 kHz).
Figure 1 shows 5 ms of a pure tone sound (440 Hz) sampled at 44.1 kHz and 22.05 kHz respectively. The discretization of sound digitization should not be underestimated as a too low sampling rate can lead to frequency artefacts.

1.2 Quantisation

Another important parameter of digitization is the process of quantisation that consists in assigning a numerical value to each sample according to its amplitude. These numerical values are attributed according to a bit scale. A quantisation of 8 bit will assign amplitude values along a scale of $2^8 = 256$ states around 0 (zero). Most recording systems use a $2^{16} = 65536$ bit system. Quantisation can be seen as a rounding process. A high bit quantisation will produce values close to reality, *i. e.* values rounded to a high number of significant digits, when a low bit quantisation will produce values far from reality, *i. e.* values rounded a low number of significants digits. Low quantisation can lead to impaired quality signal.

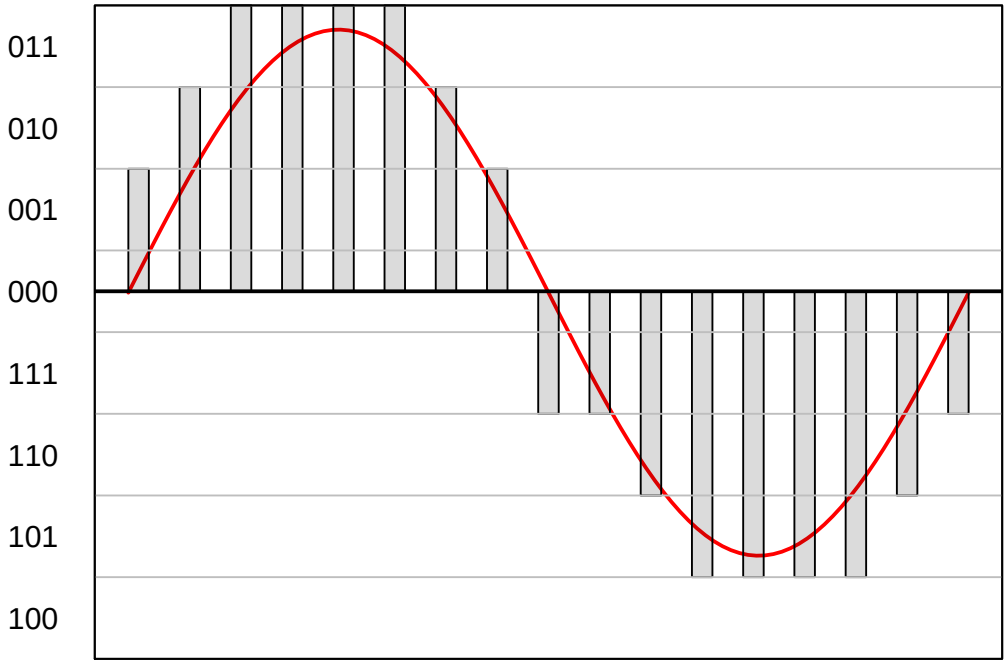

Figure 2: Digital sound is a discrete process along amplitude scale: a 3 bit ($= 2^3 = 8$) quantisation (grey bars) gives a rough representation of a continuous sine wave (red line).

1.3 File format

Wihtin R, digitized sound can be stored in three categories of files:

- uncompressed format (`.wav`): the full information is stored in a heavy file.
- lossy compressed format (`.mp3`): the information is reduced. Time, amplitude and frequency parameters can be impaired.
- losslessly compressed format (`.flac`): the full information is stored in a reduced size file.

All these formats generate binary files, sound being encoded into a succession of `0` and `1`. When importing these formats into R through `tuneR` the data are transformed into a decimal format. This implies an important increase in data size.

2 Amplitude envelope

The amplitude envelope or amplitude contour is the profile of sound energy over time. The envelope can be expressed along a relative or an absolute energy scale. There are two ways to obtain a relative amplitude envelope (see Figure 3):

- by computing the absolute value of the waveform,
- by computing the Hilbert transform of the waveform.

An example of the two envelopes types is shown in the figure 3 for the song of the bird *Zonotrichia capensis* (Figure 4) included in the `tico` data.

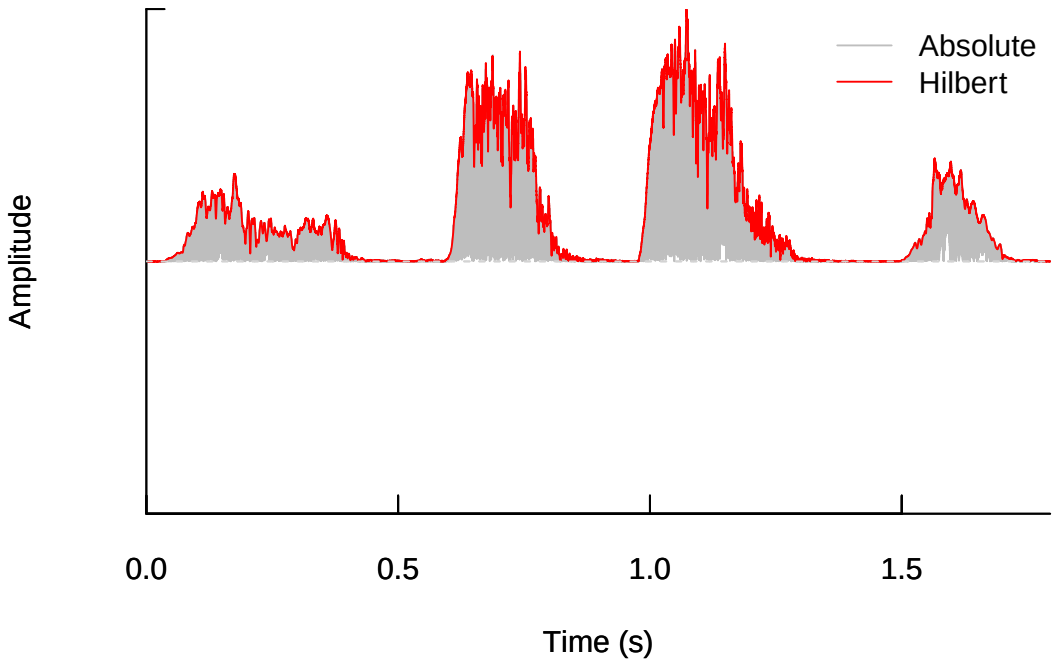

Figure 3: Two ways to compute the amplitude envelope of a sound: the absolute value or the Hilbert transform of the time wave.

Figure 4: The rufous-collared sparrow *Zonotrichia capensis* also named tico-tico in Portuguese. Picture by Ladislav Nagy, Wikimedia Commons.

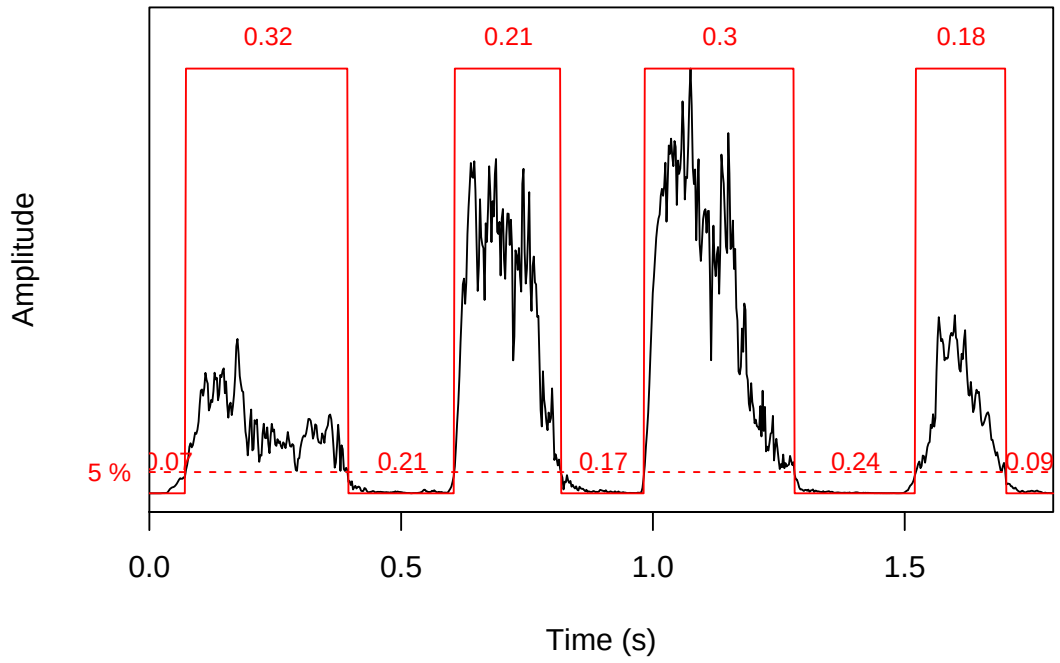

Figure 5: Use of the amplitude envelope to automatically measure the temporal pattern of a sound.
`timer(tico,f=22050,threshold=5,msmooth=c(50,0))`

The envelope can then be used to measure the duration of the different temporal parts of the sound as shown in figure 5 using the function `timer()` or to analyse the amplitude modulation rates with the function `ama()`.

3 Discrete-time Fourier Transform (DTFT)

3.1 Definitions and principle

Start first with some terminology:

Fourier transform (FT) This is a reversible mathematical transform named after the French mathematician Joseph Fourier (1768-1830) (Figure 6). The transform decomposes a time series into a sum of finite series of sine or cosine functions.

Fast Fourier Transform (FFT) This is an algorithm to compute quickly the FT.

Discrete-time Fourier transform (DTFT) This is a specific form of the FT applied to a time wave, typically a sound. Each sine / cosine function has a specified frequency and a relative amplitude. These two parameters are used to build the frequency spectrum of the original time wave. The DTFT is then a way to switch from the *time domain* to the *frequency domain*.

The signal s depicted in the figure 7 was made by the addition of three original waves with three different carrier frequencies ω_i: $\omega_1 = 1$ kHz, $\omega_2 = 2$ kHz, and $\omega_3 = 3$ kHz. The waves were added in phase ($\Phi = 0$) but with three different relative amplitudes : $a_1 = 1$, $a_2 = 0.5$, and

Figure 6: Joseph Fourier around 1823. Engraving by Jules Boilly (Public Domain)

$a_3 = 0.25$). The carrier frequencies ω_i and the relative amplitude of each sine function can be plotted in X-Y graph as shown in figure 8. This graph is a **frequency spectrum**.

3.2 Complete sound

The number of sine functions n is determined by the number of samples N of the original time wave following $n = 0.5 \times N$. If the DTFT is computed on `tico` data, which includes 39,578 samples, the DTFT will decompose the sound into $0.5 \times 39578 = 19789$ sine functions. The first sine function will have a frequency $w_1 = f_s/N = 22050/39578 = 0.557$ Hz (Figure 10). This is equivalent to the frequency resolution Δ_f of the decomposition.

3.3 Sound section

Such a high frequency resolution is often not required, if not irrelevant. In addition, computing the FFT of the whole sound might not be appropriate if there is frequency modulation along time, *i. e.* the frequency of the sound is not constant along the time scale.
A first solution is to compute the DTFT locally, on a specific sound section. The size of this section, or window, can be set up in seconds or in number of samples, a more accurate solution. We can, for instance, compute the DTFT in the middle of the third note produced by the tico bird that is at 1.1 s (Figure 10). The length of the FFT is controlled with the argument `wl` for **w**indow **l**ength. If we choose a window size of 512 samples, we will end up with a decomposition into $0.5 \times 512 = 256$ sine functions with a frequency precision $\Delta_f = 22050/512 = 43.07$ Hz.
Increasing the window size will increase frequency resolution but the decomposition will be less accurate in terms of time as more signal will be selected. Inversely, reducing the window size will be more specific in terms of time (position) but the frequency resolution will decrease. This trade-off is an example of the uncertainty or Heisenberg principle that stipulates that there is a limit in the precision of pairs of parameters, here the time and frequency parameters.

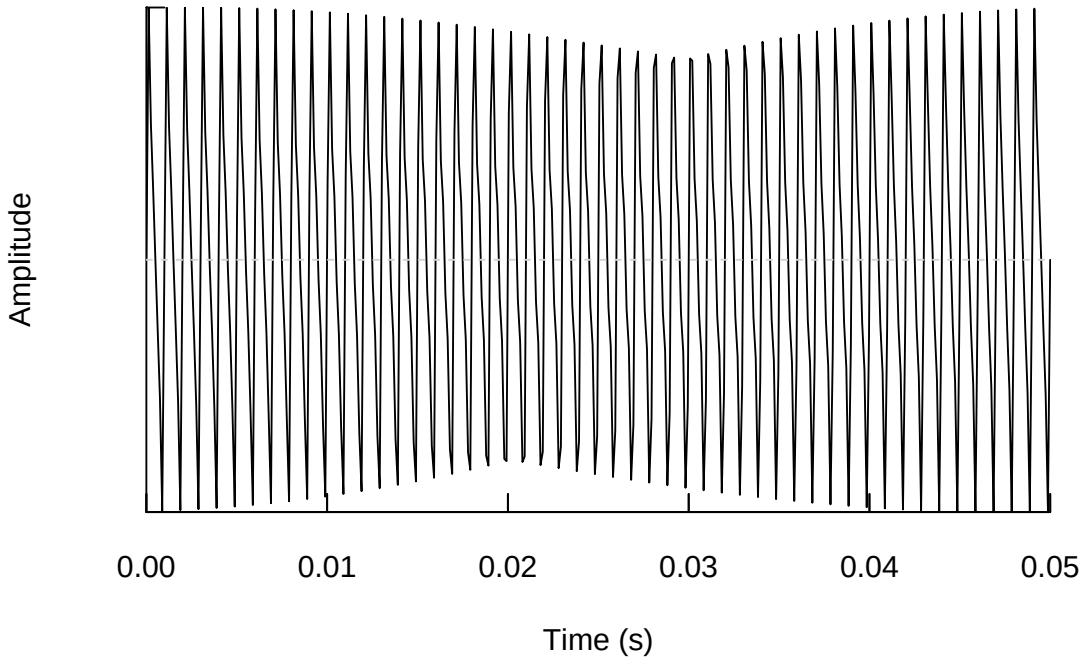

Figure 7: A time wave s sampled at 8000 Hz for 0.05 s.

```
NULL
```

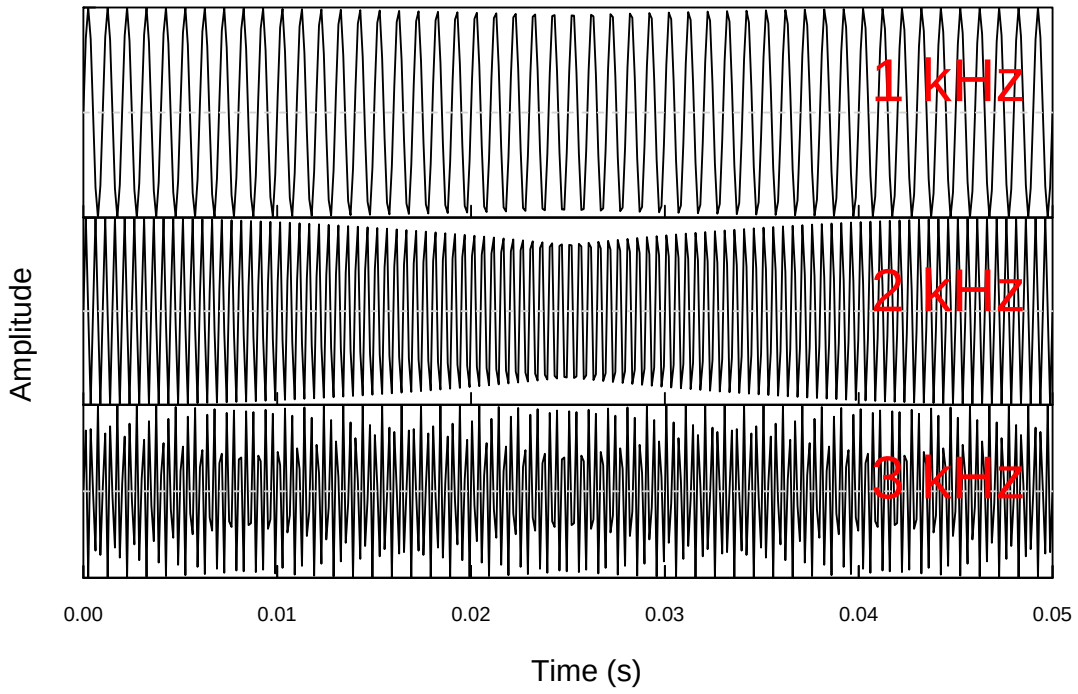

Figure 8: Decomposition of the time wave s into three sine functions. See figure 7.

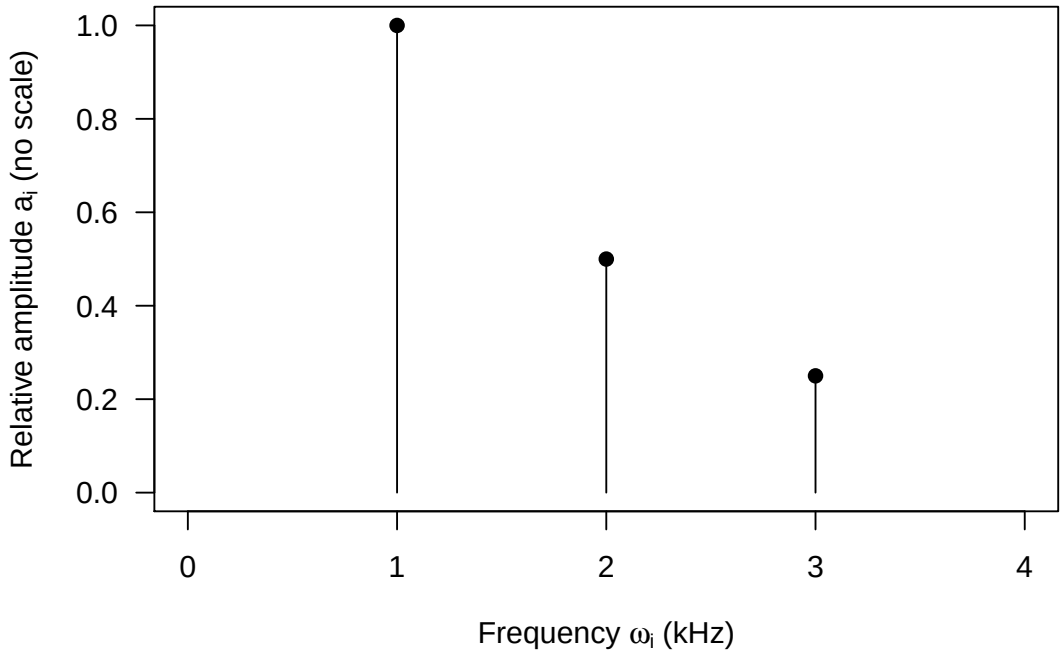

Figure 9: Frequency spectrum of the signal s. See figures 7 and 8.

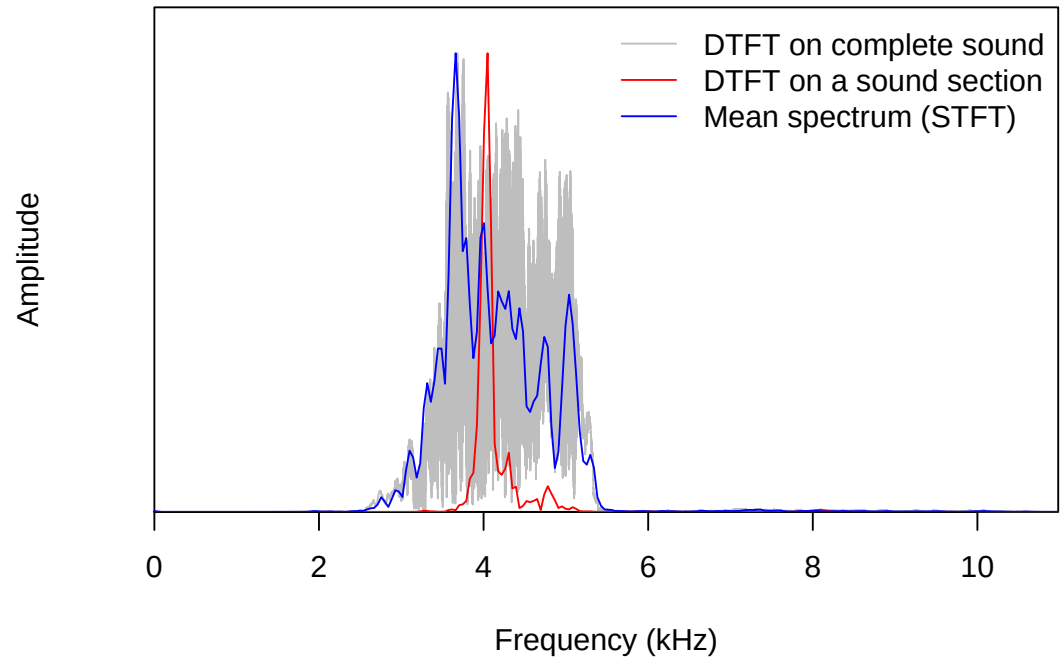

Figure 10: Three categories of frequency spectra computed on `tico` : (*1*) the spectrum of complete sound, (*2*) the spectrum computed at 1.1 s with a 512 samples window, and (*3*) the mean spectrum computed with the STFT (see section 4).

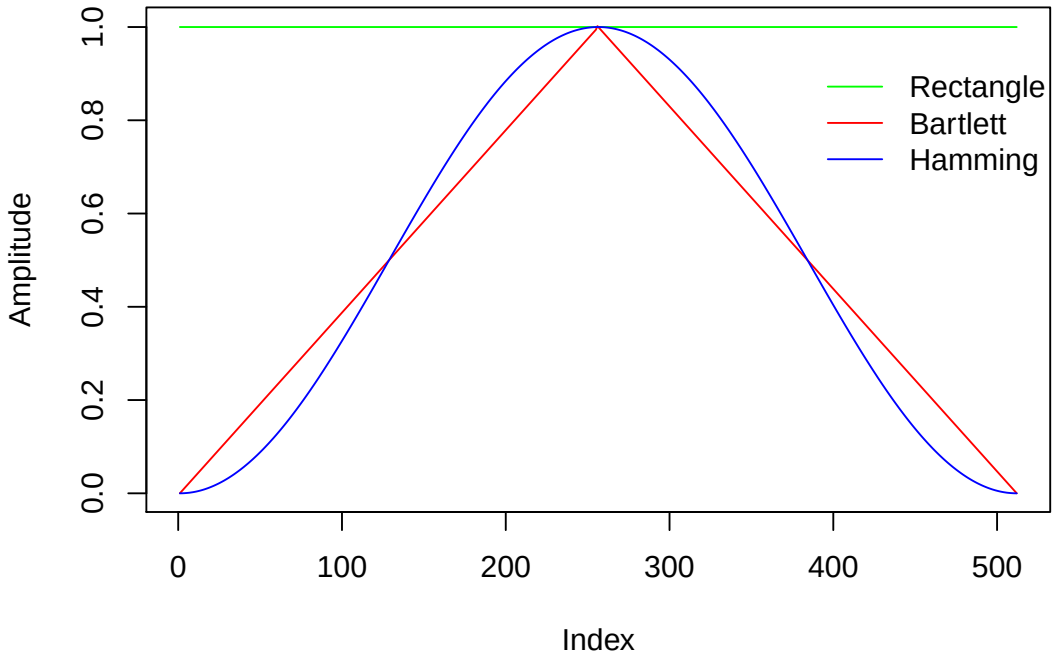

Figure 11: Three different Fourier window shapes. Try `example(ftwindow)` for other shapes.

3.3.1 Window shape

When computing the DTFT, the shape of analysis window is by default a rectangle. However this shape is not always appropriate as it induces artefacts like side frequency lobes. A way to avoid this is to multiply the original window with a function of the same length with a particular shape. This shape can be rectangular – in that case nothing is changed to the original signal – triangular (Bartlett window), or sinusoidal (Blackman, Hamming, flat top, and Hanning windows) (see figure 11 for three examples of window shapes and figure 12 for a test on a simple signal).
The default window shape used in seewave is the Hanning window but other windows could be more appropriate depending on main signal features.

4 Short-time Fourier Transform (STFT)

4.1 Principle

Computing the FFT on the whole sound or a single section might not be informative enough. An intuitive solution is to compute the DTFT on successive sections along the signal. A window is then slided along the signal and a DTFT is computed at each slide or jump. This is what the short-time Fourier transform (STFT) does.
A good way to understand how it works is to use the function `dynspec()`. The successive DTFT can be tracked when moving along the signal with a sliding cursor. Here is an example with a DTFT window of 1024 samples:

```
> dynspec(tico, wl=1024, osc=TRUE)
```

Basically the STFT returns a matrix of values where columns are the successive spectra along time. This can be summarized as a a_{np} matrix : with a_{ij} the Fourier coefficients, n the number

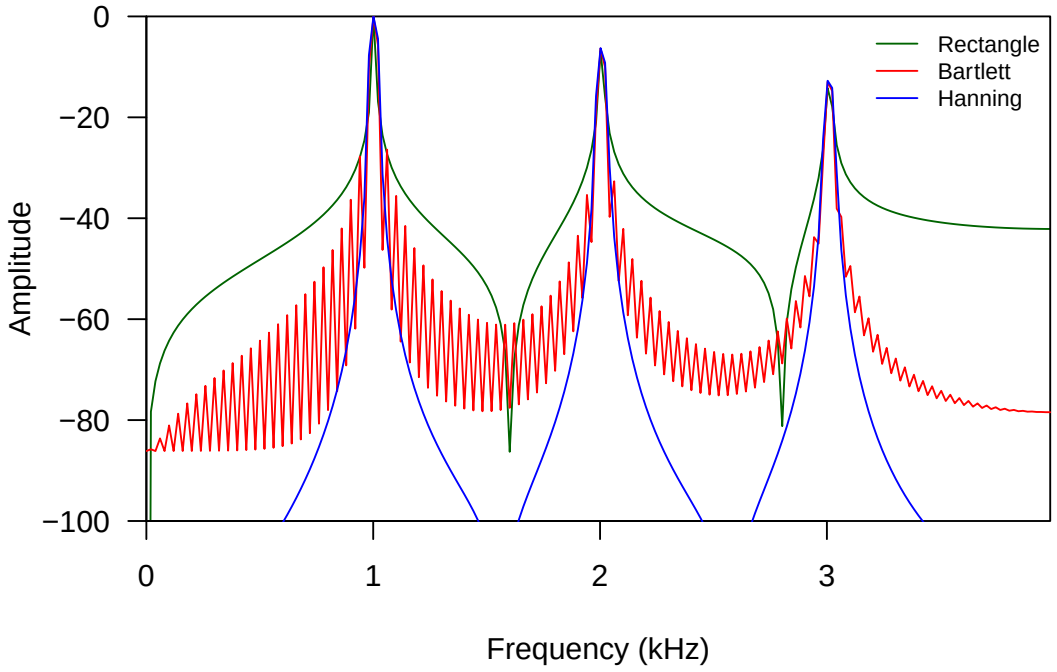

Figure 12: Using different DTFT window shapes can lead to different spectral profiles. The frequency spectrum of the signal s (see figure 7) was computed with three different window shapes. Side lobes differ between the three spectra

of frequency ω (window size / 2) and p the number of Fourier windows computed along the signal:

$$\begin{array}{c} \\ \omega_1 \\ \vdots \\ \omega_i \\ \vdots \\ \omega_n \end{array} \begin{array}{c} \begin{array}{ccccc} t_1 & \dots & t_j & \dots & t_p \end{array} \\ \begin{pmatrix} a_{11} & \dots & a_{1j} & \dots & a_{1p} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{i1} & \dots & a_{ij} & \dots & a_{ip} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{n1} & \dots & \dots & \dots & a_{np} \end{pmatrix} \end{array}$$

This matrix is nice but a plot of it would even be better. There are three ways to friendly visualize this matrix:

- a waterfall plot, see the function `wf()`,
- a density plot, or spectrogram, see the function `spectro()`.
- a 3D plot, or 3D-spectrogram, see the function `spectro3D()`,

4.2 Spectrogram

4.2.1 3D in a 2D plot

The density plot option, or **spectrogram**, is the most popular representation used in bioacoustics. It has the main advantage not to be based on a 3D representation that is not appropriate for human eye inspection. The principle is quite simple: the successive DTFT are plotted against

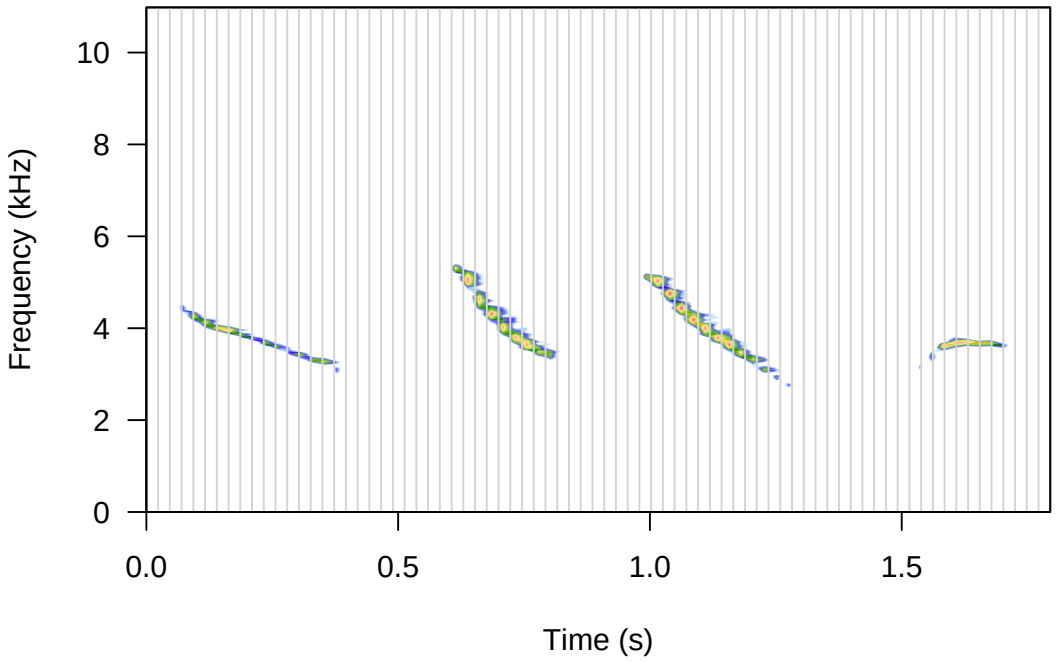

Figure 13: STFT on `tico` song showing 77 successive FFT windows containing 512 samples each.

time with the relative amplitude of each sine function of each DTFT being depicted in reference to a colour scale. The usual way is to use a dB scale for amplitudes with negative dB values refering to a maximum of 0 dB.
If we keep the `tico` example, applying a STFT with a sliding window of 512 samples will result in the computation of $39578/512 = 77$ FFTs. The spectrogram will therefore be made of 77 sections (columns) as shown in figure 13.

4.2.2 Overlap

The temporal and the frequency resolutions of the spectrogram are linked, with $\Delta_f = \Delta_t^{-1}$. In the latter case, the frequency resolution is $22050/512 = 43.06$ Hz and the time resolution is $512/22050 = 0.0232s$. As mentionned above, increasing the size of the window will increase frequency resolution but decrease time resolution.
However, there is a trick to counteract this two-dimension precision limit. In the first example, the DTFT window was coarsely jumping from a position to another but we can make the jump slightly better. The solution is to simply allow an overlap between successive windows. This overlap is usually set up in percentage: the default value is 0% as in figure 13. A percentage of 50% will double the number of DTFTs, hence increasing the time resolution by a factor of 2 (now 153 FFTs) when the frequency resolution is not reduced. The overlap parameter is set up with the argument `ovlp` of the function `spectro()`(see figure 14). A value of 100% is of course a non-sense as the sliding window will stay on the spot. Increasing the overlap inscrease computing time as more FFT are computed. We advice to keep reasonable values for computing efficiency.

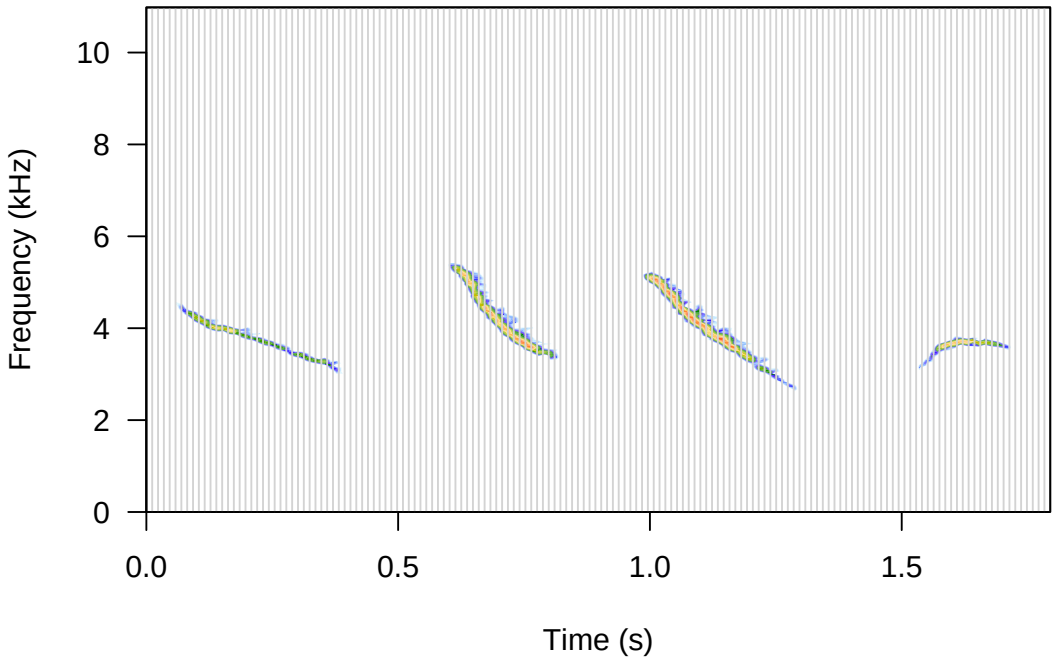

Figure 14: STFT on `tico` song showing the 50% overlapping FFT windows. Each of the 144 FFTS contains 512 samples.

4.2.3 Values

The spectrogram is a graphical function but the values along the three scales can be saved. The value of `spectro()` is a list containing three components:

- `$time` or `[[1]]` returns the values of the time axis,
- `$frequency` or `[[2]]` returns the values of the frequency axis,
- `$amp` or `[[3]]` returns the amplitude values of the successive FFT decompositions or spectra.

These components can be used to plot the spectrogram manually (Figure 15). The successive spectra computed by the successive DTFT can also be picked up and plot as the function `spec()` would do (Figure 16).

4.3 Mean spectrum

The columns of the STFT matrix can be averaged giving the so-called mean or average spectrum as shown in the figure 10. The frequency resolution and the shape of the mean spectrum will of course change when changing the window size (`wl`) and overlap (`ovlp`) arguments.
The function to compute the mean spectrum is `meanspec()`.

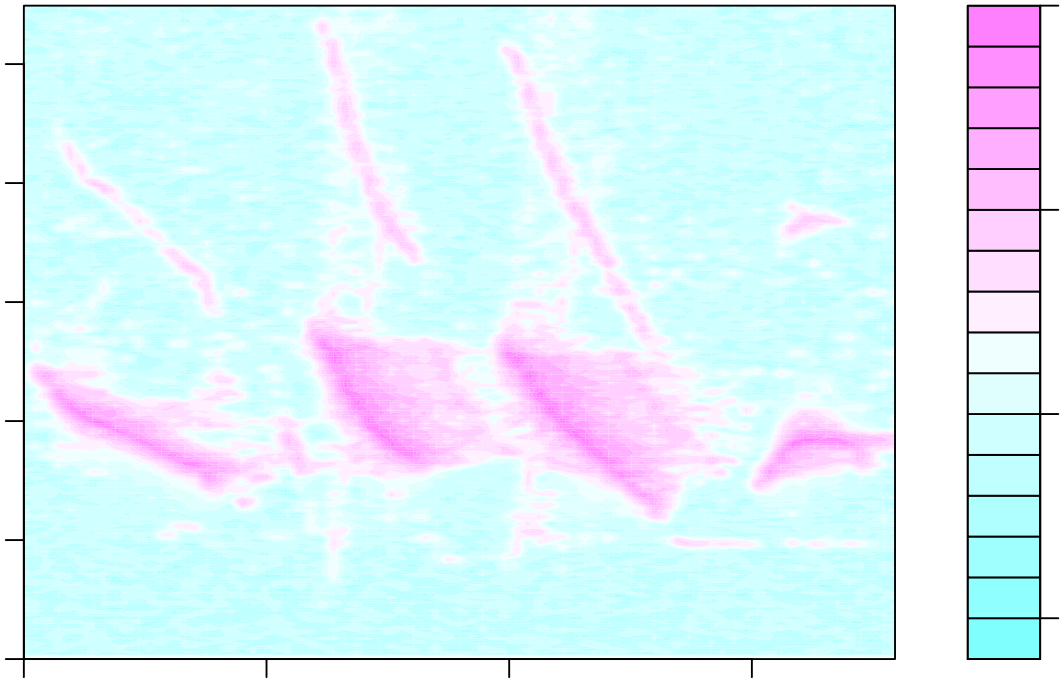

Figure 15: Redrawing the graphical output of the function `spectro()` with the native `filled.contour()` function, with the command: `filled.contour(x=spectro[[1]], y=spectro[[2]], z=t(spectro[[3]]))`.

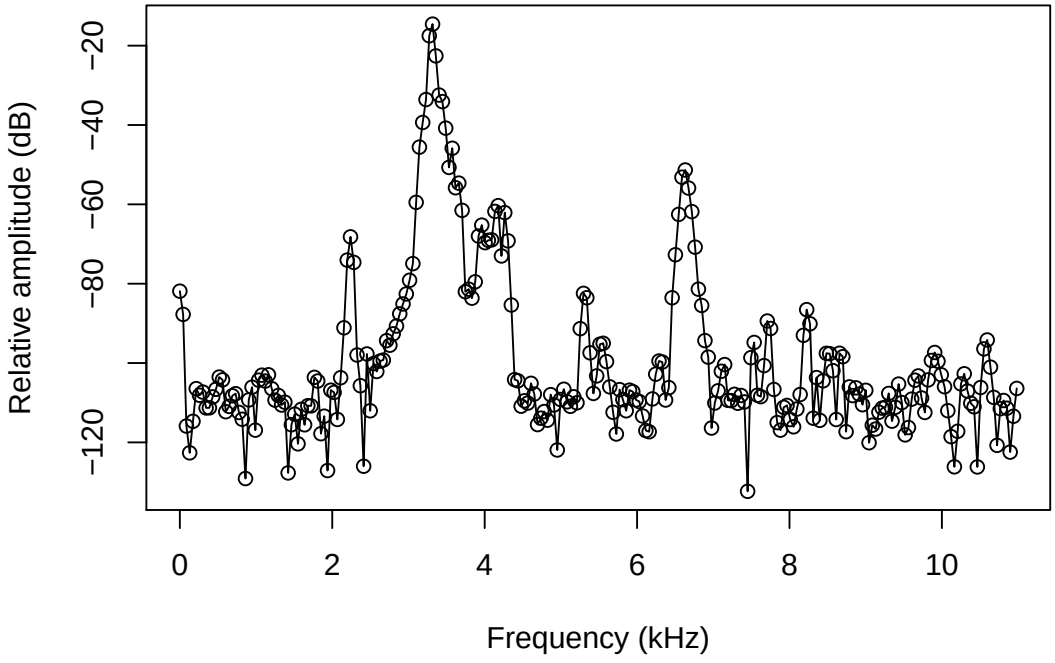

Figure 16: One of the dB spectra computed by the STFT.
`plot(x=spectro[[2]], y=spectro[[3]][,15], type="o", xlab="Frequency (kHz)", ylab="Relative amplitude (dB)")`

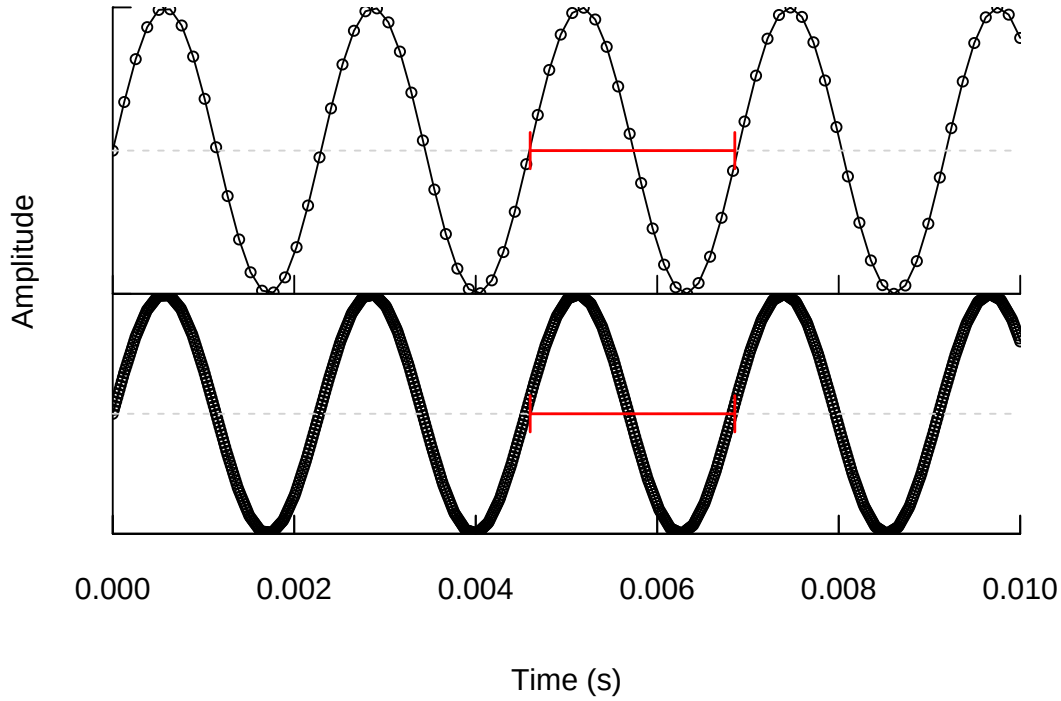

Figure 17: Zero-crossing: principle and interpolation to reduce innacurracy of measurement. The upper panel shows a 440 Hz signal sampled at 8000 Hz. The sampling is too low to measure properly the period betwee successive cycles. The lower pannel plots the same wave with $\times 10$ interpolation factor. New samples are added, it is now possible to measure the periodicty of the wave.

5 Instantaneous frequency

5.1 Zero-crossing

The zero-crossing is a rather simple technique which consists in measuring successive time intervals at which the wave crosses the zero amplitude line. This gives a measure of the period T of a full cycle and the instantaneous frequency is obtained by simply computing $f = T^{-1}$. The signal has to be quite periodic to make the method reliable.
The main problem in zero crossing procedure is linked to the discrete process of sound sampling. The signal to be analysed might not always have values equal or very close to zero. This makes the zero-crossing results quite approximative. An example of this issue is illustrated in the upper panel of the figure 17. It is therefore sometimes necessary to oversample the signal by interpolation. This process adds values closer to zero and then increase the accuracy of the measure as shown in the lower panel of the figure 17. An example of such measure on the usual `tico` song in the figure 18.

5.2 Hilbert transform

The Hilbert transform is a decomposition of a signal $x(t)$ into the amplitude envelope and the instantaneous frequency. More specifically, the amplitude envelope is the modulus of the analytic signal, defined as $z(t) = x(t) + iy(t)$ where $y(t)$ is the Hilbert transform, and the instantaneous frequency is the derivative of the phase of $z(t)$ with respect to time.
The Hilbert transform can be thus used to track both amplitude and frequency modulations.

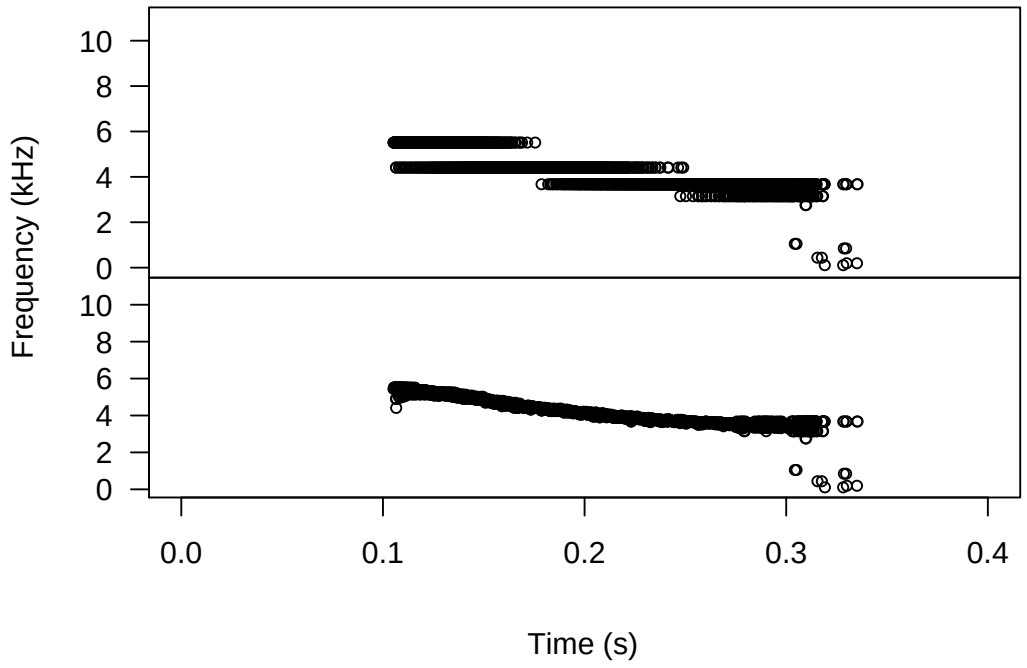

Figure 18: Zero-crossing: measuring the instantaneous frequency of a note the `tico` song without (upper panel) and with interpolation (bottom panel).

The amplitude enveloppe is obtained with the function `env()` and the instantaneous frequency is obtained with the function `ifreq()`.

5.3 Cepstral transform

The Cepstral transform is the inverse Fourier transform of the logarithm of the spectrum. The real cepstrum (an anagram of spectrum) is the real part of the Cestral transform. The scale of the independent variable (usually the y-axis) of the cepstrum is named quefrequency. The quefrequency scale is not intuitive but can be transformed in frequency (Hz). The cepstrum is useful for detecting the fundamental frequency of an harmonic series, it corresponds to the first peak of the cepstrum as shown in the figure 19. Note that this dectection will work properly with harmonic signals only. The function `cepstro()` is short-term version of the Cepstral function: successive cepstrum are computed along the signal with a sliding window in a similar way as the STFT (see section 4).

6 Other transforms

There are several other options to analyse a signal. Among others, we could list the following ones that are not included in seewave:

- Mel-frequency cepstral transform, see the function `melfcc()` of the package `tuneR` [not tested]
- Wavelet transform, see the packages `biwavelet`, `rwt`, `wavelets`, `waveslim`, `wavethresh` and `wmtsa` [not tested].
- Gabor transform – not yet implemented in R.

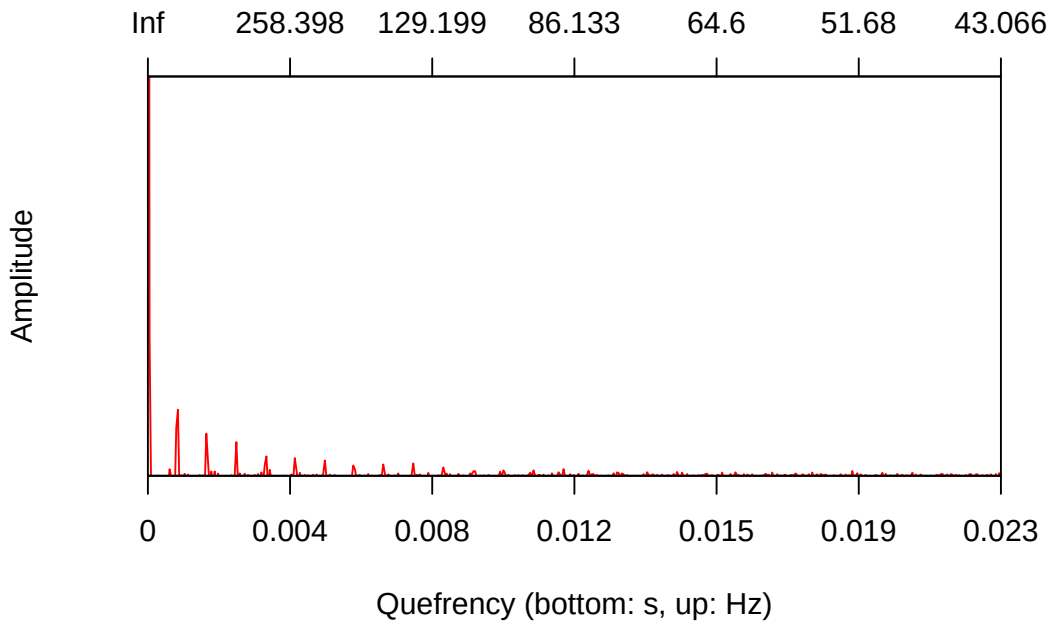

Figure 19: A cepstral analysis on a *Vanellus vanellus* call (data `peewit`) with the following call: `ceps(peewit,at=0.4,wl=1024, col=2)`.

- Wigner-Ville transform – not yet implemented in R.

7 References

7.1 Books

Au WWL, Hastings MC (2008) *Principles of marine bioacoustics*, Springer.
Bradbury JW, Vehrencamp SL (1998) *Principles of animal communication*, Sinauer Associates.
Fletcher NH (1992) *Acoustic systems in biology*, Oxford University Press.
Gerhardt HC, Huber F (2002) *Acoustic communication in insects and anurans*, University of Chicago Press.
Hopp, SL, Oweren MJ, Evan CS (1998) *Animal acoustic communication*, Springer.
Marler P, Slabbekoorn H (2004) *Nature´s Music. The Science of Birdsong*, Academic Press, Elsevier.
Rossing TD (2007) *Handbook of acoustics*, Springer.
Rumsey F, McCormick T (2002) *Sound and recording - an introduction*, Elsevier.
Speaks CE (1999) *Introduction to sound*, Singular.

7.2 Dedicated journals

Animal Behaviour – http://www.journals.elsevier.com/animal-behaviour/
Bioacoustics – http://www.tandfonline.com/toc/tbio20/current
Journal of the Acoustical Society of America – http://asadl.org/jasa/