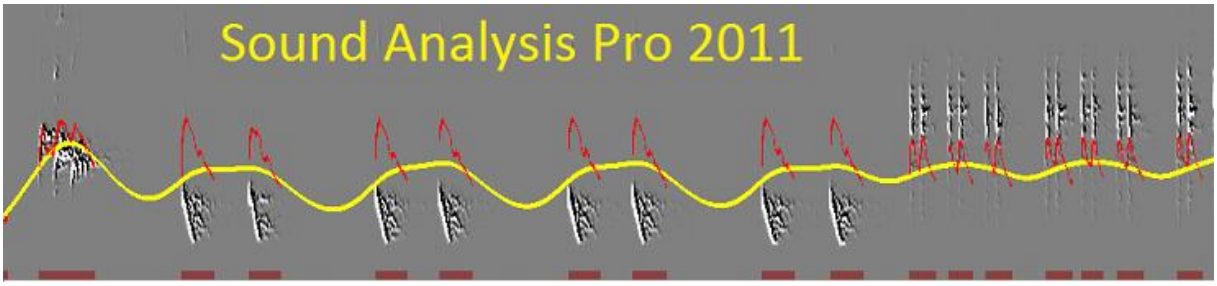


User Manual

Ofer Tchernichovski

Compiled from

http://soundanalysispro.com

September 2012

License

Sound Analysis Pro 2011 (SAP2011) is provided under the terms of GNU GENERAL PUBLIC LICENSE Version 2

Note: Sound Analysis Pro is provided "as is" with NO WARRANTY

However please feel free to email Ofer Tchernichovski (tchernichovski@gmail.com) if you have any questions, suggestions or concerns. When possible we will try to help and provide advice and guidelines.

About Sound Analysis Pro 2011

Sound Analysis Pro is a free software that can be used for studying animal vocalization. It includes:

- Multi-channel recorder threshold-activated sound detector
- Automated playbacks, operant (e.g., by key-peck) or passive
- Modern spectral analysis allowing flexible exploration of vocalization structure
- Extracting spectral features that are correlated with articulation
- Computing similarity measurement
- Clustering vocal sounds to types
- Presenting scatter plots (DVD maps) of millions of vocal sounds over extended periods
- Full integration with mySQL database
- Data exporting to Excel and to Matlab.

Important: when using SAP2011 please refer to:

Tchernichovski, O., Nottebohm, F., Ho, C.E., Bijan, P., Mitra, P.P. (2000). A procedure for an automated measurement of song similarity. *Animal Behaviour 59, 1167-1176*

Acknowledgments

SAP2011 is supported by a R01 grant from the **NIH** Institute of Deafness and other Communication Disorders (**NIDCD) from 2000-2014**.

Programming: Ofer Tchernichovski, Elena Kashtelyan & David Swigger

Algorithms: Partha Mitra, Ofer Tchernichovski & others (see below)

SAP2011 is based on a digital signal processing (*DSP*) engine called *ztBirdEngine*, developed for us by ***David Swigger*** (**ZettaTronic Inc**). It handles sound input from up to 10 channels and performs real time signal processing, recording and playback control. It is encapsulated into an **ActiveX** component, which is included in this package together with its entire source code and documentation.

Spectral analysis is done with the public domain **FFTW** (http://www.fftw.org) algorithm, including features for optimizing performance of frequency analysis to the hardware configuration of each computer (all those features are automatically and transparently implemented in *SAP2011*, but can be used for other applications). We included the implementation of these FFTW routines, together with routines of randomly accessing wave files in a Dynamic Link Library with open source code and appropriate documentation.

Calculation of acoustic features (pitch, Wiener entropy, FM, etc.) is now encapsulated in a C++ class developed by **David Swigger**, which is also included with this package with proper

documentation.

Many of the algorithms used in SAP were developed by **Partha P. Mitra.** The algorithms for classification of syllable types were implemented by **Aylin Cimenser** at Columbia University.

All database routines were programmed using **MySQL** (http://www.MySQL.com) database engine. MySQL is also a public domain and open-code application. Implementation of the MySQL engine was done through the **Core Lab MyDAQ3** package (http://www.crlab.com).

We used the Embarcadero RAD Studio 2010 to program SAP2011 (sing the **C++ Builder module).**

Many GUI features were programmed with **SDL Component Suite** (http://www.lohninger.com).

The Excel exporting was implemented using the **XL Report** by **Afalina Co., Ltd** (http://www.afalinasoft.com).

All other algorithms, implementations to C++ code, integration of components, software design as well as the GUI and help system were developed by **Ofer Tchernichovski** and **Partha P. Mitra**.

Many of the improvements in software design should be credited to user feedback and data sharing: we would like to thank **Cheryl Harding**, **Michael Brainard**, **Heather Williams**, **Santosh Helekar**, **David Vicario**, and **Noam Leader** for their help and data sharing.

Ofer Tchernichovski

September 2011

A bird's eye view of Sound Analysis Pro 2011

Sound Analysis Pro 2011 performs automated recording and analysis of animal vocalization. It can record and manage sound data over prolonged periods. It can be used to train animals with playbacks while recording their vocalization, e.g., throughout vocal development of a bird. The basic analyses include automatic segmentation of vocal data to syllables, calculation of acoustic features such as amplitude, pitch, frequency modulation and entropy, to summarize the spectral structure of vocal sounds. Based on those features, advanced analyses include similarity measurements, cluster analysis (to identify syllable types), and several methods for displaying distributions of vocal sounds over long time scales (.e.g, scatter plots of an entire song development).

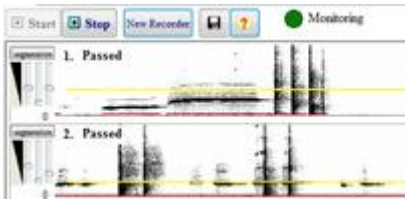
SAP2011 can be installed in any *Windows 7, 2000 or XP* computers. Other operating systems are not supported. For specific hardware requirements (depend on what you want to do) see **Chapter 2: Installation of SAP2011**.

SAP2011 includes six modules: Recorder, Live Analysis, Explore & Score, Batch, Clustering, and DVD maps:

- **Recorder (10 channels):**

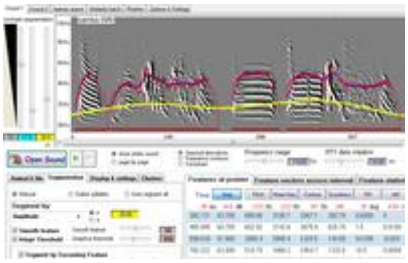
The recorder performs multi-channel (up to 10) recording using most standard and ASIO sound cards. Recording modes include i) *Triggered recording:* records vocalization that passes an amplitude threshold, including pre-buffering and duration control. ii) *C*ontinuous recording: records continuously into files of fixed duration, and iii) Master-slave recording: for synchronized recordings across channels. The recorder allows training with sound playbacks, e.g., by the animal pressing on a lever, etc). It includes an accurate time code, and it can transfer data to the Live Analysis module, which performs spectral analysis and saves sounds in daily folders. For more information see **Chapter 7: Recording Vocal Sounds**

- **Live Analysis**



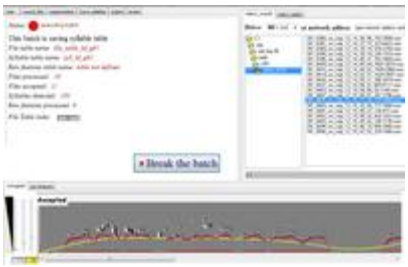
Live Analysis is the companion module of the *Recorder*. It processes sound data that passed the recorder thresholds and performs spectral analysis and calculation of acoustic features, followed by segmentation to syllable units. The sonogram is displayed in nearly real-time to allow simple sound detection settings. In addition to saving sounds, it can also save tables with syllable features, and raw tables of continuous features. Files are automatically arranged in data folders of appropriate capacity, .e.g., of daily folders.

- **Explore & Score**



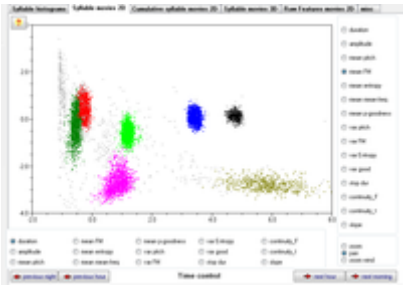
Explore & Score is used to browse sounds, examine their features, segment them (manually or automatically), and to perform a variety of measurements that can be easily stored and exported to Matlab or to Excel. It provides variety of graphical representations to examine distribution of features. It is used to score the similarity between sounds, with several alternative methods for scoring similarity.

- **Batch**



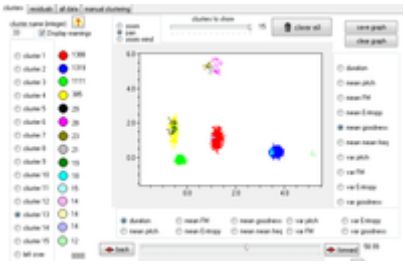
The *features batch* can analyze large amount to sound data to calculate acoustic features and save them into syllable tables and into raw features tables. The tables can include millions of records that can be presented graphically in the DVD module. The *similarity batch* can be used to perform a large set of similarity measurements. It supports two batch modes: one is for comparing ordered pairs of sounds, and the other is for comparing matrixes (M x N) of sounds.

- **Dynamic Vocal Development maps**



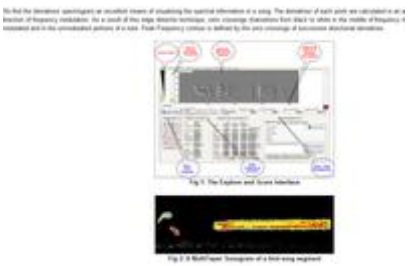
SAP2011 automatically generates and updates a *syllable-table* for each bird, which summarizes every song syllables produced during vocal development (in a zebra finch, it is typically 1-2 million syllables). Obviously there is a lot of information in those syllable tables. To make this information easily accessible we developed a descriptive model called the *Dynamic Vocal Development (DVD) map*. *DVD maps* are presented as movie clips showing how syllable features change during song learning (or as a result of experimental manipulation). In the adult bird, the distribution of syllable structure is highly clustered, and the *DVD maps* show how these clusters (syllable types) come about. We developed several types of such maps to show different aspects of song development including syntax, circadian factors, and cross time-scales vocal changes. The different modes of *DVD maps* use shape, color and even sound-clicks to represent different aspects of song structure. Importantly, *DVD maps* can be played in nearly real-time, so that you can see a vocal change as it occurs.

- **Clustering syllables**



Clustering is used to detect *syllable types* and to automatically trace vocal changes during song development.We implemented a *nearest-neighbor hierarchal clustering* method into an extensive graphic user interface including a display of clusters in color code, assessment of residuals, and an account of the number of members in each syllable type. The procedure performs the cluster analysis recursively, throughout song development. It provides online visual assessment of the outcome in each stage of analysis. The results of the clustering are automatically registered into the *syllable table*, so that as you do the cluster analysis you can play DVD maps, and ensure, by inspecting the color-code of each cluster, that the tracing procedure is indeed 'locked on the target'. The tracing of each syllable type progresses from the mature song and back until the clustering procedure fails to trace the syllable type. As long as a cluster can be traced, it is easy to detect *vocal changes* that occur as the feature of a cluster approaches the final (target) state. Cluster Analysis is therefore a formal way of analyzing (and parameterizing) the *DVD map*.

- **Online help**



SAP2011 now includes many functions and to make them easily accessible to the user we developed a web help system, that allows quick update and posting of comments. Clicking on the help links in SAP2011 will take you directly to the help pages you need to look at in this site.

Chapter 2 - Installing Sound Analysis Pro

Chapter 2 - Content:

Hardware Installation: Building Sound Attenuation Boxes

In the previous chapter we provided a brief summary of SAP2011 features. If you are only interested in using SAP2011 to analyze data (without the recorder and live analysis components), you may skip this section and go directly to *Software Installation*. If you already have a recording system that includes a multi-channel sound card, microphones connected to an existing experimental setup, and you want to implement *SAP*2011 as a recorder - but not for operant training - you should read parts II and III of this section. Otherwise, you should read the entire chapter.

Training boxes

Building your own sound boxes is easy and will save you thousands of dollars. The sound isolation quality will depend on the thickness of the foam used, and on the accuracy of application, but in general, it should be easy to achieve at least 30dB of attenuation, which would usually prevent sound leakage from one box to the next as long as both doors are closed. In zebra finches, we did not detect any penetration of bird vocalization sounds across boxes (as long as both boxes are closed and airflow is intact).

Coolers:

There are many brand names (cost is bout $40-$60 per box). There are certain differences in the quality of hinges and ease of handling. We used Thermos 110 qt Coolers (Outside: 34 x 16 x 17; Inside: 32 x 14 x 15). It is very likely that you will have to replace the hinges after a few years of usage.

Sound isolation foam:

The material cost is about $50 per box. We recommend **SOUNDFOAM "M" from** http://www.soundcoat.com/absorption.htm:

which is an acoustic quality, open cell, flexible polyether based urethane foam designed specifically to provide maximum sound absorption in minimum thickness. It is characterized by outstanding humidity resistance, excellent chemical resistance, fine and uniform cell size and consistent sound absorption. It is available plain, embossed, or with decorative and protective surface finishes of aluminized polyester film, Tedlar, and matte urethane film. Soundfoam "M" is supplied with one of several high performance pressure sensitive adhesives in “ready to use custom die-cut parts. Glazed ¾ and ½ “: thick for the sides and front lead:Use silicon glue to suture the joints between sheets (you will need glue gun and silicon glue tubes).

Latching:

Remove the original latches from the cooler and replace them by metal locks. We use Stanley LifeSpan USA+ from Home Depot.

Cages:

18 x 9 x 10 Previw-Hendrix Model DB (Breeding small cages with a split, which is not used for our application).
* comment: those cages are now difficult to find, and most commercially available cages are too large. We now custom build cages of the appropriate size for our experiments.

Lights:

We use LB1-W12, 12-LED Light Bars (http://www.superbrightleds.com) 12 VDC Power Supply (http://www.superbrightleds.com\specs\ps_specs.htm) of 5 Amp can be used to connect at least 4 bars (namely for 4 cages). You will also need a 36" long LB1 Jumper – Power cable and at liest 2 Light Bar Mounting Tracks per cage. Overall cost is around $30 per cage (including the partial coast of power supply). You will need to do some soldering and make sure that the power supply is secured in an appropriate enclosure (the provided enclosure is not secure enough – place it inside an appropriate electric box).

Air supply:

The airflow system should be centralized. We use 100W or 120W aquarium air pump model #LPH120.
http://www.Jehmco.com http:\\www.jehmco.com\products_\hardware_\central_air_pumps\
This should suffice for 20 boxes or so. These pumps are very quiet since they are surrounded by a 'sound box'. You can choose between a few different capacities. The highest one (xxx Amp) should suffice for 20 training boxes, whereas the smaller ones () can support 10 or 15 boxes.

Airline Silicon tubing:

250 feet, small diameter tubing with good damping of sound noises, from Foster & Smith.

Racks:

You may use any comfortable shelving solution, and get the coolers fixed (e.g., by bungee cords or by screws).

Hardware Installation: Sound hardware

Sound Hardware <u>Microphones:</u> There are many good (and expensive!) choices, but we feel that most condensed microphones can provide reasonable sound quality. We like the EarthWorks microphones, (SR69 or SRO). They have high sensitivity and a very flat frequency response up to 20kHz. Another good, and less expensive option is the Audio-Technica http://www.audio-techica.com. Some sound cards supplies appropriate phantom power and the combination works very well. One thing we discovered though is that those microphones should be protected from bird feces. We discovered that zebra finches can spray their feces in a straight upward direction (they do so while performing wheelbarrow jumps in the case!) and a direct hit from a single piece of feces can destroy the microphone. A piece of cloth set loosely around the recording tube solved that problem.

<u>Sound card or Analog Input card</u>

<u>Option 1:</u> Use a regular sound card for two channels recording. Such inexpensive cards would usually work fine, but you might experience about 5% leaking of sound across stereo channels. You can install two such cards to allow recording from 4 channels but in most cases you will have to use cards of different brands to avoid driver conflict (which can be very nasty). In addition, you will need to buy a preamplifier for each microphone and for your speakers as well

<u>Option 2:</u> Use a multi channel sound card that are ASIO or DirectSound compatible. The card that was successfully tested in our system is M-Audio ProFire 2626 http://www.m-audio.com/products/en_us/ProFire2626.html:

It is a decent and relatively inexpensive card (about $600) with 8 preamplified microphone channels with phantom power, and 8 output line level audio channels. It requires a firewire connector (or a firewire PCA card if the motherboard does not support it). We did not try the equivalent USB card.

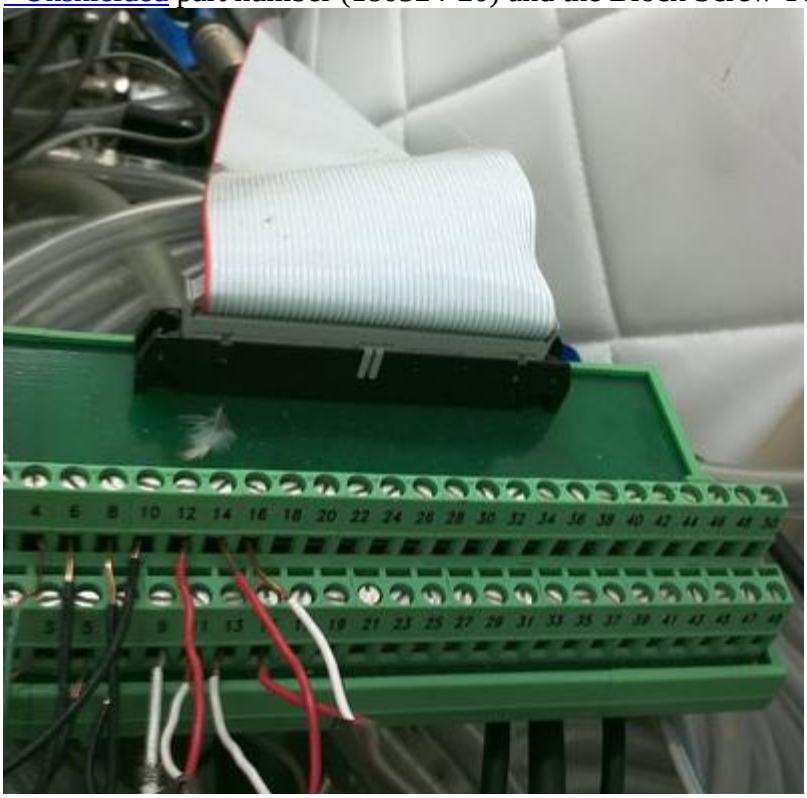
Hardware installation: storage media

We use USB2 external hard discs for storage. We like the Lacie hard drivers: they are very reliable and simple. It is advised to keep the operating system on a separate internal hard disk, and use either the USB Lacie drives for recording, or have an additional internal hard disks, or even better -- a RAID. We custom build our computers with internal RAID system that gives effective 4Tb of storage space (with redundancy) so that bad hard disks can be replaced (swapped) on the fly.

Setting National Instrument Card

Our results suggest that operant training facilitates song imitation in zebra finches compared to yoked trained birds. We are still working on improving the training regimen, and users should consult our website for the most current updates. In particular, we found that replacing the keys by strings, and adding mirrors in specific locations might facilitate vocal learning. At this time, we are using a key-pecking system as our standard training method as documented below.

Digital I/O card: if you want to set an operant training you will need a National Instruments PCI-6503 I/O card. This is the least expensive NI card, and it is great. You will have to install the NIDAQ-traditional driver of this card. By the NB1 Ribbon Cable (1m), (2m) & CB-50LP - Unshielded part number (180524-20) and the Block Screw Terminals 777101-01



Install the card (follow instructions!), connect the cable to the screw terminal and then connect the screw terminal to the keys:

Low cost alternative: A $100 USB card called NI USB-6501 seems to work too without any additional software or hardware (no cable or screw terminal needed, just connect the keys directly to the card), but we did not test it extensively yet. Since it is USB you can try it with laptops.

Keys: Cherry 1g lever key can be any key, you may use any lever key, we had good experience with the Chery 1g lever keys. Use a standard low voltage electrical cable to connect them. Connect to the NO (normally open) connectors. You may join all the grounds together, as long as all of them are connected to the same NI card. We attach a small round disk to the end of the lever, and fix the lever on the wall of the cage, next to a perch, so that the bird can peck on the disk with its beak (some birds I saw also jump at it). Here are pics of

the lever in a cage. We mount them with binder clips that we connect to the cage bars with cable ties.

Screw terminal configuration: The screw terminal has even numerals (2-50) at the back row (closer to the ribbon cable connection). Those are all grounds. The distant row with the even numerals has the actual channels. The channels are divided into three ports (port 0, 1, 2) and each port has eight lines (lines 0-7). Number 47 in the screw terminal corresponds to port 0 line 0. the next screw to the right, number 45, corresponds to port 0 line 1, and so on all the way to number 1, which corresponds to port 2 line 7. First, connect each terminal screw to the to the appropriate device. For example, port 0 line 0 to key 1 of training box 1, port 0 line 1 to key 2 of training box 1, port 0 line 2 to key 1 of training box 2, and so forth. , line 2 to key 2, etc. Input wires such as lever keys should be connected via a long lamp-cable (polarity does not matter in this case). Now you need to connect each device to the *ground*. All the even channels are common ground – connect whichever way you like. No resistor is necessary using this configuration.

Now you should test your system (before connecting the other keys). Turn your computer on and start the NIDAQ instrumentation panel. Double-click on the devices icon and you should see the PCI icon. Right click on it and choose 'test panel'. You should see a panel with buttons. Set the top row as input port 0 and the bottom row as input port 1. If the buttons of port 0 are not red, click 'generate output'. Make sure that the port is set to 0 and input, and then, while looking at the panel, click on the key. You should see that the first gray button has turned gray – you just activated line 0. Every time the bird pecks on a key, when a motion detector is activated and so forth, *SAP2011* will capture the event via one of these lines and will respond appropriately.

From here an on you continue with every other wire: connecting wire 6 to a key, and then back to the common should activate line 1, connecting wire 8 will activate line 2, and so forth. Keep track of your connections; you will have to remember them when setting up the input/output configuration of the SAP2011 recorder.

Installation of National Instruments analog card to work with SAP2011: SAP2011 allows recording via most NI analog data cards. You will have to set up a data neighborhood for each channel and then use them to record with the recorder just as you do with sound cards. This is particularly useful for recording combined sound and electrophysiology data.

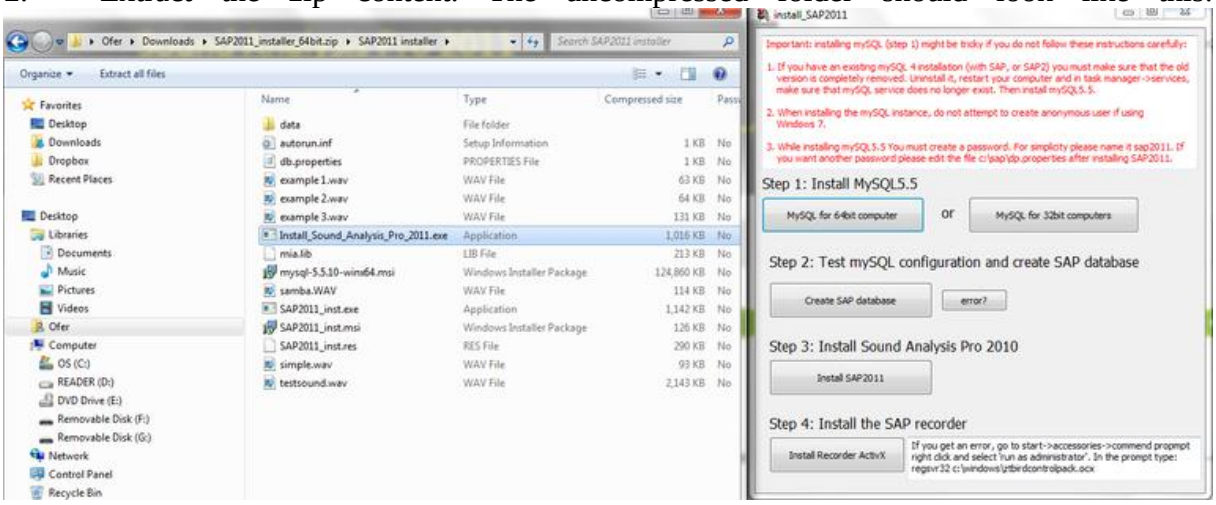
Software installation step by step

System requirements: Windows 7 or Windows XP.

Important: If your computer language is not English, you might need to set your computer date format, language and regional setting to US English to avoid floating point errors when opening sound files.

To install the beta test of SAP2011 please follow these steps:

1. Download the installer zip folder 64bit installer or 32bit installer (to determine your system type see Control Panel\System and Security\System)
2. Extract the zip content. The uncompressed folder should look like this:



3. Run Install_Sound_Analysis_Pro_2011.exe and follow instructions carefully.
4. When installing mySQL make sure you choose the correct version (64bit or 32bit) and follow the default settings. Keep the default mySQL user name as root, and set password sap2011. **Do not create anonymous account.**

General guidelines:

* Avoid uninstalling mySQL5 -- the uninstaller is buggy and re installation might be difficult. SAP2011 requires mySQL 5, if you have mySQL4 installed you should first uninstall it and make sure that the service (in task manager) no longer exists.

To upgrade from mySQL 4 to mySQL 5:

a. Uninstall mySQL4 including the mySQL control center.
b. Delete c:\ProgramData\MySql directory (it's hidden).
c. If that doesn't work, try to find and delete any directory containing mySQL in it's name in Programs folder. You may also need to clean up the registry.

If you upgrade from mySQL4, It is also recomended to select mySQL5 as the instance (service) name (default is mySQL).

5. Create SAP database, make sure you enter the password selected during mySQL installation (sap2011).
6. Install SAP2011.
7. Installation of Connection to Matlab is optional.
8. Install the SAP recorder.

If you run into any problem, please email tchernichovski@gmail.com

MySQL optimization

Optimization of MySQL:

Install the mysql qui tools from http://dev.mysql.com/downloads/gui-tools/5.0.html

Go to Start->MySQL->MySQL Administrator

Click the Startup variables (left tab) and select the advance networking tab. Check wait timeout and set it to 9999999 and Then click “Apply changes”

In the performance tab and set **cache Query** and **cache limit** to at least 256M (or try 500M)

Then click “Apply changes”

Next go to Advance tab, scroll all the way down and set **Bulk insert buffer** to at least 256M

Install SAP in Mac

SAP is incompatible with Mac OS. However, you can run SAP in Macs using Windows desktop, by following these steps:

1. Install a *Parallels* platform for windows (version 4 or higher, acadmic version should cost about $40). There are other virtualization software available which could, perhaps work too. The Parallels coherence mode is useful as it allows Mac and Windows apps to be open at the same time in the Mac OSX environment.

2. Install windows 7 (home version will do).

3. Install SAP, then MySQL workbench or whatever SQL clients are preferred.

—

Important comments:

- When installing SAP, it is advised to take windows to full screen mode.

- Windows 7 should have full access to all Mac hardware (drives, wireless, data ports) and it should share these with the Mac OS

- Windows XP may not work, we strongly recommend using Windows 7

- Avoid an installation of Windows from a disk image (which is what your IT department might recommend). This seems to result in windows with poor integration with existing hardware.

Please let us know if you have encountered any issues. We thank Christopher R. Olson (OHSU) for providing this information.

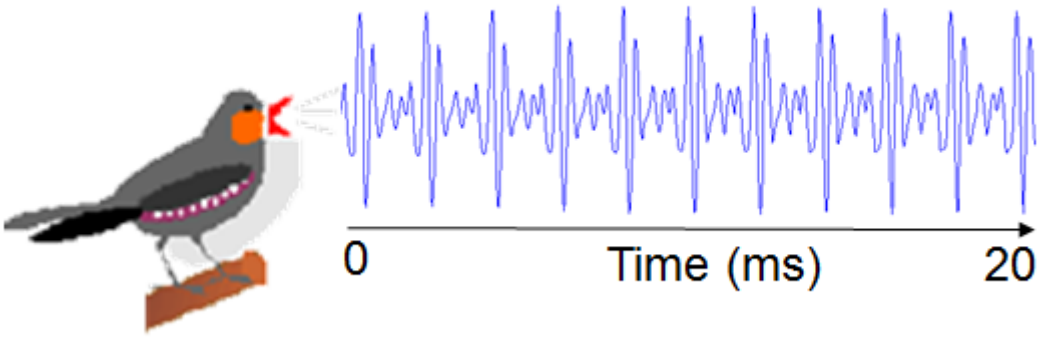
Chapter 3: Spectral Analysis

Chapter 3 - Content:

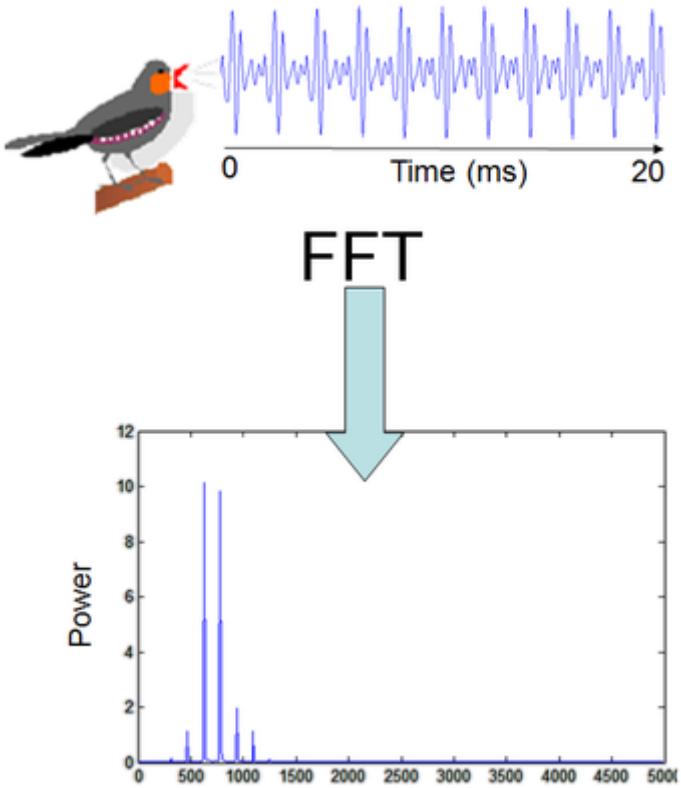
Introduction to Spectral Analysis

This chapter presents some concepts of spectral analysis and acoustic features including some knowledge base that might help you get the most out of SAP2011. We will use the Explore & Score module to present those concepts. This module is similar to the previous versions of Sound Analysis with several new features.
When listening to birdsong, it is immediately apparent that each song has a distinct rhythmic and sometimes even melodic structure. Songs of individual birds in a flock sound similar to each other and differ from those of other flocks. As early as the 18th century, Barrington noted that the songs of cross-fostered birds differed from the species-typical song, suggesting a role for vocal learning. However, until the late 1950s, there had been no objective way of confirming these observations by physical measurements of the songs themselves. The invention of the sound spectrograph (sonogram) at Bell Laboratories was a significant breakthrough for quantitative investigation of animal vocal behavior. The sonogram transforms a transient stream of sound into a simple static visual image revealing the time-frequency structure of each song syllable. Sonogram images can be measured, analyzed, and compared with one another. This allows the researcher to quantify the degree of similarity between different songs by inspecting (or cross-correlating) sonograms and categorizing song syllables into distinct types. Each song is then treated as a string of symbols, corresponding to syllable types, e.g., a, b, c, d..., and song similarity is estimated by the proportion of shared syllable types across the sonograms of the two songs. The procedure is equally useful in comparing the songs of different birds and that of the same bird at different ages or after control and experimental treatments.
Spectral analysis is less than intuitive, and here is a little technical tutorial about how sonograms are computed:

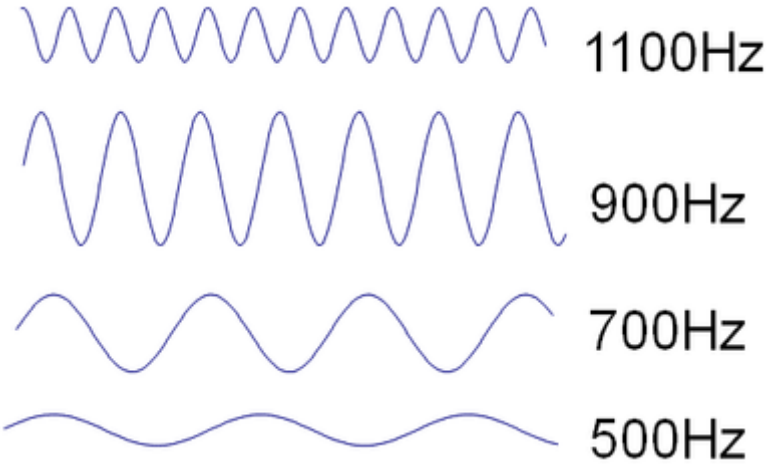
When recording a singing bird, the microphone capture tiny fluctuations in air pressure we call sound waves and turn those into an electrical current, which might look like this over 20 milliseconds:



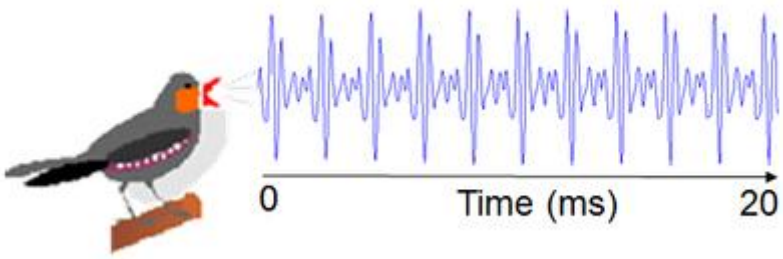
The fast Fourier transform (FFT) is an algorithm to compute the periodic structure in the signal, but representing it by a set of sine waves, called frequencies. Plotting the power of each one of those frequencies gives the Power Spectrum:



Each of of the peaks in the Power Spectrum above, corresponds to a sine wave, e.g.,:

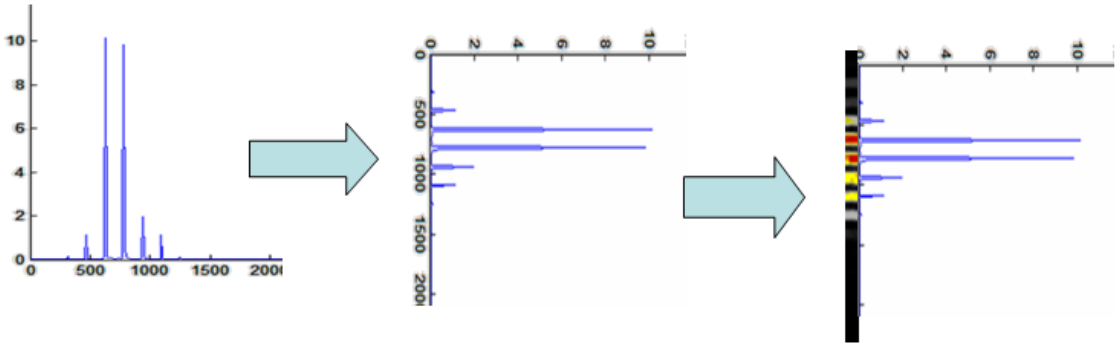


Now comes the interesting part: if we take those waves and add them, we will get a combo wave that looks like this:

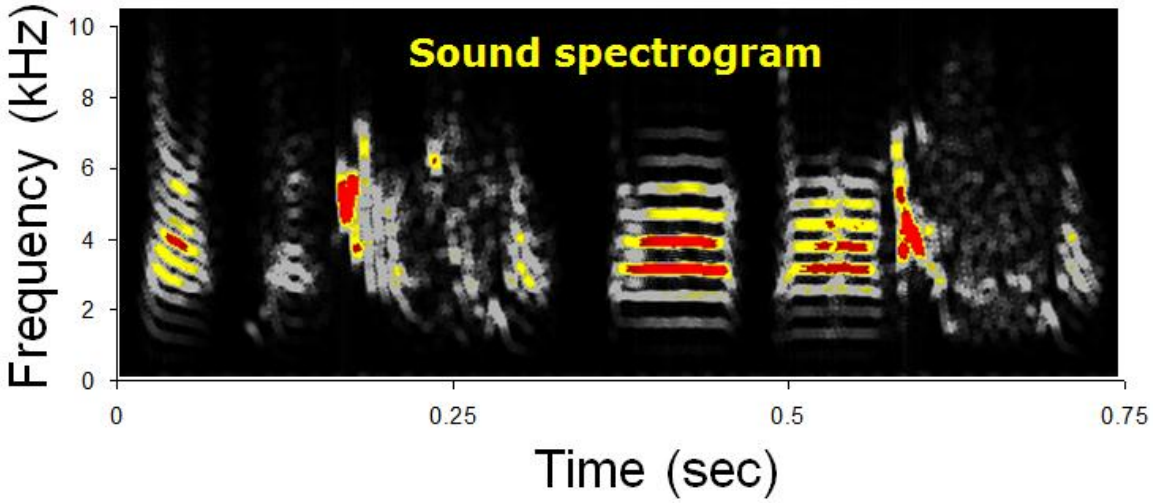


Exactly like the original wave, that is, the Power Spectrum (including phase, that is usually not shown) is a complete representation of the sound. We can look at the time domain (wave form) or frequency domain (the Power spectrum), and it summarizes its periodic structure.

Computing power spectrum works best when the signal is periodic and stationary, which is why in sound it usually makes sense to use short time windows, e.g., 5-20ms. To look at an entire song we have to summarize power spectra of many small time windows. This is how it is done: Take the power spectrum of one window, say from the first 20ms of the song, flip it and replace the graph with a color bar:



and now repeat this procedure for overlapping time windows, i.e., 1-20ms, 2-21ms, 3-22, etc, and stack the color bars next to each other. The outcome might look like this:



This is the famous sonogram! It is a set of vertical strips, each one represent the power spectrum of one time window of sound.

Glossary of terms

Sound Data: Sound data are defined as files of digital sound. See chapter 8 for information about digital recording, digitizing and re-sampling sounds.

Digital recording: is a sequence of sound pressure data (waveform) in a digital format. The two parameters that determine the quality of the data are sampling rate and sampling accuracy.

Sampling rate: determines how many samples of sound data are collected during one second (Hz), e.g. 44100Hz is sometimes referred to as 'CD quality'. The accuracy of the digital representation of each sound pressure data is measured in bits, e.g. 16 bits means that each sample of sound is represented by one of 216 = 65536 possible values. Sound Analysis require accuracy of 16 bits and sampling rate of either 22050, 44100 or 88200 Hz.

Hertz: (symbol: Hz) is the SI unit of frequency defined as the number of cycles per second of a periodic phenomenon.

Sound units: Sound Analysis distinguishes between silence intervals and signal.

Syllable: is defined as a continuous sound, bounded by silence intervals. Depending on the task, Sound Analysis sometimes treats the sound as a continuous analog signal and sometimes as a set of syllables.

Fourier transformation: Fourier transformation (FT) transforms a short segment of sound to the frequency domain. The FT is implemented algorithmically using the Fast Fourier Transformation technique (FFT).

Time window (frame): Time window is the duration of the segment of sound upon which Fast Fourier Transformation technique (FFT) is performed. Sound Analysis default is 409 samples of sound (9.3ms) and the next window starts 1.4 ms after the beginning of the previous one and has therefore an 85% overlap with the previous window.

Spectrogram: is a sequence of spectra computed on such windows, typically represented as an image where power is represented on a scale of gray ranging from white to black. Because frequency resolution is finite, the spectrogram does not capture a 'pure sine wave' but represents the 'frequency' as a 'trace' of a certain distribution in time and frequency. Using long time windows improves the frequency resolution at the expense of time resolution.

MultiTaper (MT) spectral Analysis: MultiTaper methods are a modern framework for performing spectral analysis. In particular, they lead to spectral estimates that are similar but superior to the traditional spectrogram. Apart from the estimation of the spectrogram, MultiTaper methods also provide robust estimates of derivatives of the spectrogram as well as provide a framework for performing harmonic analysis (detection of sine waves in a broadband noisy background).

Spectral derivatives: The traditional sonogram represents the power of sound in a time-frequency plan, while the spectral derivatives represent the change of power. The derivatives behave as 'edge detectors' of frequency traces in the time-frequency plan, and provide a superior spectral image.

Spectral Derivatives

What are Spectral Derivatives? The traditional sonogram represents the power of sound in a time-frequency plane, while the spectral derivatives represent the change of power. For each point of the two-dimensional time-frequency plane of a sonogram, one can measure change of power from left to right (on time), from bottom to top (on frequency) or at any arbitrary direction. So, spectral derivatives are derivatives of the spectrogram in an 'appropriate' direction in the time-frequency plane. Spectral derivatives can be estimated using MultiTaper spectral methods, they have the same resolution and are not artificially broadened.

Sound Analysis Pro uses spectral derivatives to track frequency traces in the spectrogram as follows. As one cuts across a horizontal frequency trace, from low to high, there is a sharp increase in power, then a plateau, then a decrease in power. The same cuts are first positive and than negative, passing through zero at the peak power location. A useful property of these derivatives is that they show a sharp transition from positive to negative values, providing a contour that is more accurately defined than just the frequency trace.

If the frequency trace is not horizontal, then the direction of maximum change in power is not in the frequency axis, but rather at an angle to both time and frequency axes. To capture the direction of maximal power change in the frequency trace, it is then natural to take a directional derivative perpendicular to the direction of frequency modulation (think about detecting waves in the ocean by cutting through the surface in many arbitrary direction until you hit the wave). The directional derivative is easily computed as a linear combination of the derivatives in the time and frequency directions, and may be thought of as an edge detector in the time-frequency plane.

We find the derivatives spectrogram an excellent means of visualizing the spectral information in a song. The derivatives of each point are calculated in an angle that is perpendicular to the direction of frequency modulation. As a result of this edge detector technique, zero crossings (transitions from black to white in the middle of frequency traces) look equally sharp in the modulated and in the unmodulated portions of a note. Peak Frequency contour is defined by the zero crossings of successive directional derivatives.

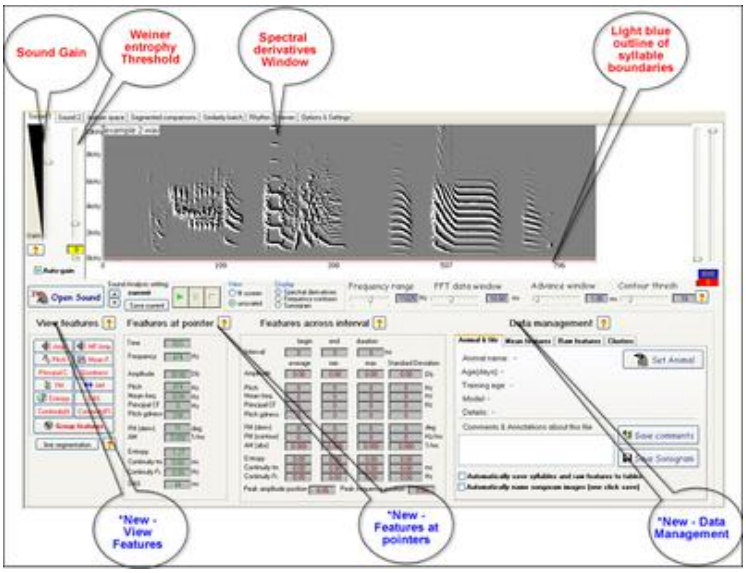


Fig 1: The Explore and Score Interface

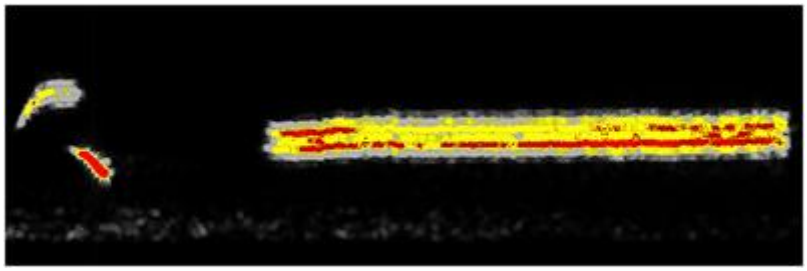


Fig 2: A MultiTaper Sonogram of a bird song segment

Spectral derivatives of the same sound in Fig 2 here the frequency traces are more distinct. Since the derivatives are calculated in a different direction for each point, subtle modulations are also visible Estimates of frequency and time derivatives of the spectrum may be robustly obtained using quadratic inverse techniques (Thomson, 1990, 1993).

These estimates have the general form: An approximation of the above matrix is define by:

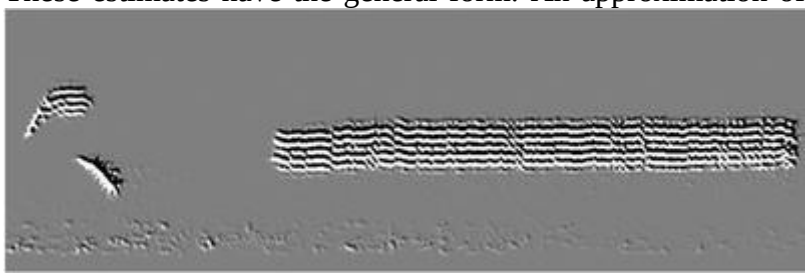


Fig 3: Spectral Derivatives of the same sound

Estimates of frequency and time derivatives of the spectrum may be robustly obtained using quadratic inverse techniques (Thomson, 1990, 1993).

These estimates have the general form:

$$\sum_{k,k'} A_{k,k'} \tilde{x}_k(f) \tilde{x}^*_{k'}(f)$$

An approximation of the above matrix is defined by:

$$z(f,\theta) = \sum_{k=1}^{k-1} \tilde{x}_k(f)\, \tilde{x}_{k+1}(f) e^{i\theta}$$

Empirically:

$$\frac{\partial S(f,t)}{\partial S} \propto \mathrm{Re}(z(f,\theta))$$

Where:

$$\partial S(f,t) / \partial S$$

is a directional derivative of the spectrogram in the time-frequency plan, the direction being specified by the angular parameter:

$$\theta$$

In particular, the time and frequency derivatives of the spectrogram may be obtained by setting:

$$\theta = 0, \pi$$

Segmentation of Syllable Units

One of your most important decisions when analyzing vocal sounds is choosing **methods** and setting **parameters** for distinguishing vocalization from background noise. At the macro scale, it allows you to detect bouts of vocalization, and at shorter time scales, to segment the sound to syllables (vocal events with short stops between them) and sub-syllabic units. Even though some analysis can be done on the continuous signal in file, once vocal events are identified and segmented it is possible to do much more, e.g. identify and compare units of vocalization, classify sounds and compare them by similarity measurements and clustering methods.

SAP2011 offers a few alternative approaches to segmentation, as demonstrated below:
Open Explore & Score --> Open Sound --> Select file you wish to explore (a quick preview of the sound is provided in the lower portion of the screen) & choose the Pagination option in the lower right corner; full file (recommended for smaller files) or page by page (recommended for larger files of 5 seconds or more) -->Click OK

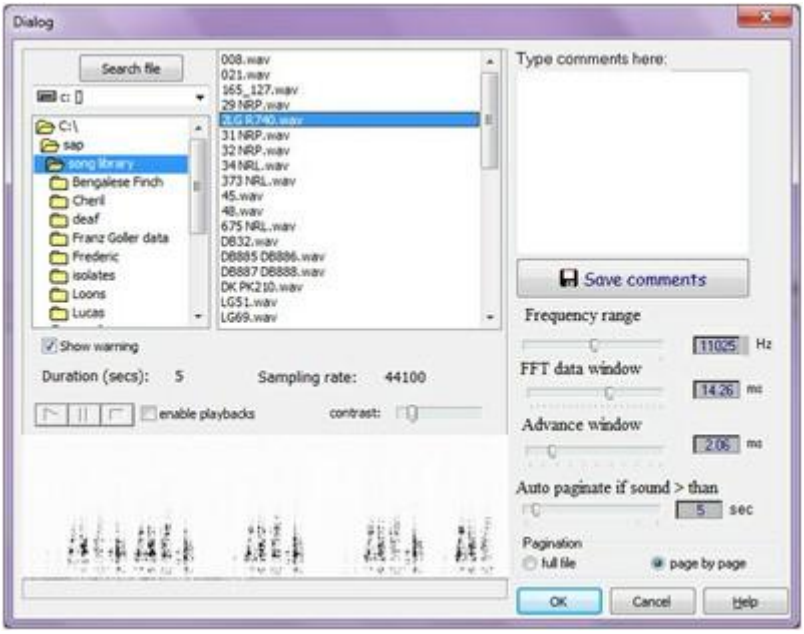


Fig 1: Open Sound window

- **Using a fixed amplitude threshold to segment sounds**

One of the simplest and most widely used methods for segmenting sounds is by a fixed amplitude threshold:
Select Segmentation Panel tab on the left side of the window.

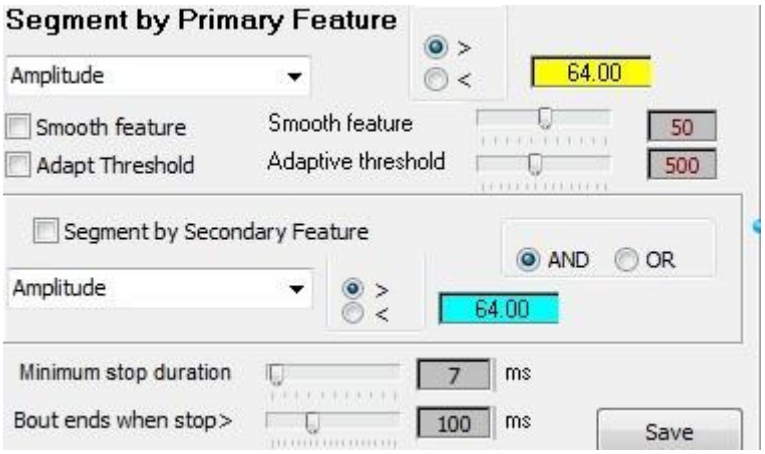
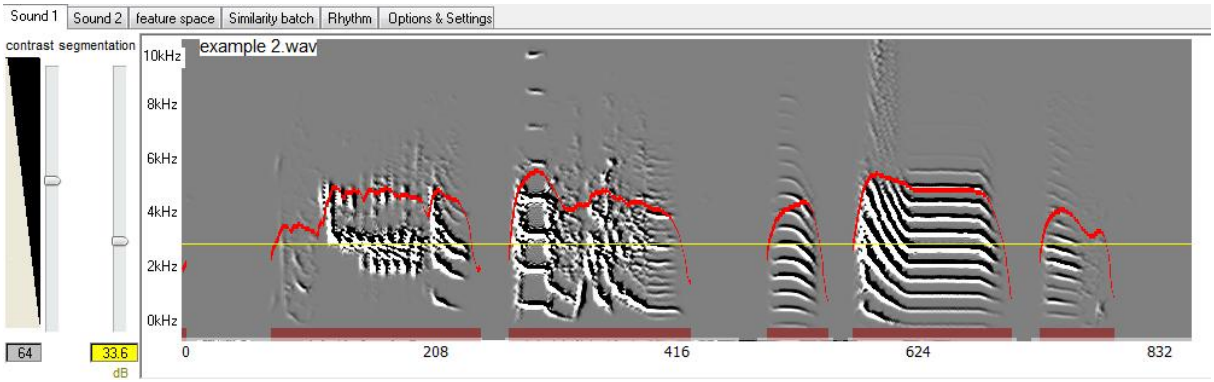


Fig 2: Segmentation Tab

Select primary Feature to segment; there are nine available options:

1. Amplitude
2. Pitch
3. Mean Frequency
4. Goodness of pitch
5. AM
6. FM
7. Weiner Entropy
8. Continuity over time (t)
9. Continuity over frequency (f)

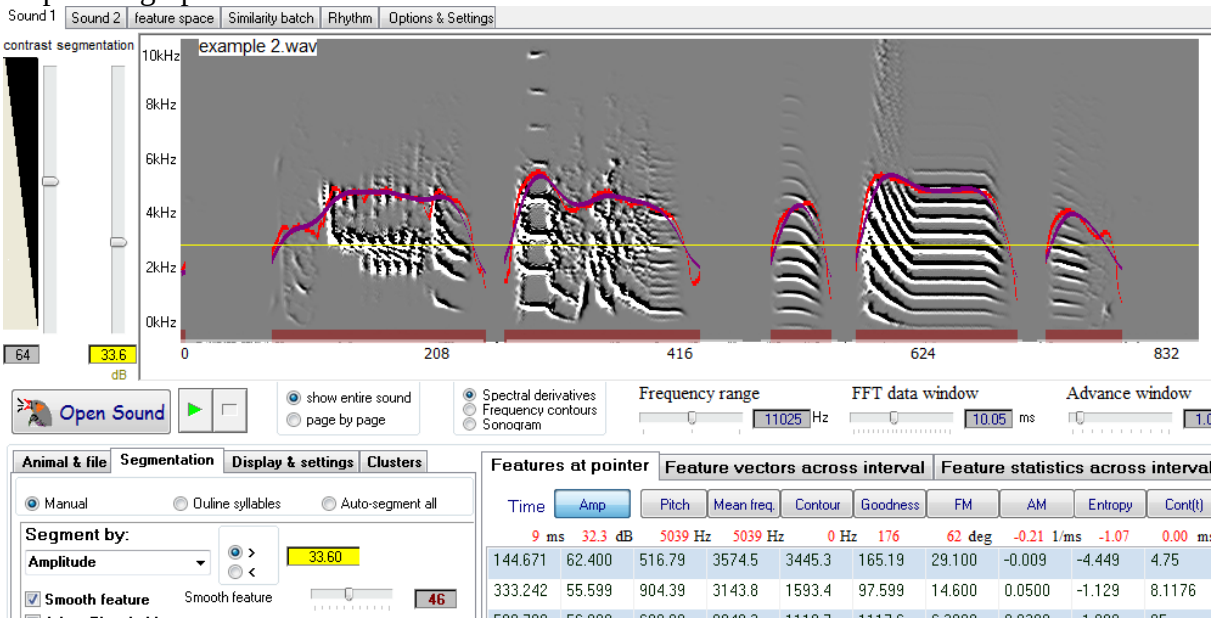
Move amplitude threshold slider located next to the sonogram screen to the appropriate level, while watching the segmentation changing below the sonogram. The yellow horizontal line on the graph indicates the level of the threshold. The feature can be shown when greater than the threshold (if ">" option is selected next to threshold) or less than the threshold (if "<" option is selected). Syllables are shown on the bottom of the graph screen in thick red line segments.



- **Using a smooth feature option to segment sounds**

Check "Smooth feature" checkbox, the Smooth feature slider will show. Select the level to which to smooth the feature. The lowest level is 0, which means that the feature will not be smoothed. The highest level is 100, which means that smoothing window is 100 datapoints. For example, if the advance window param is set to 1 ms, then the smoothing window of 100 is 100 ms. The smoothed feature graph will appear in different colors depending on which

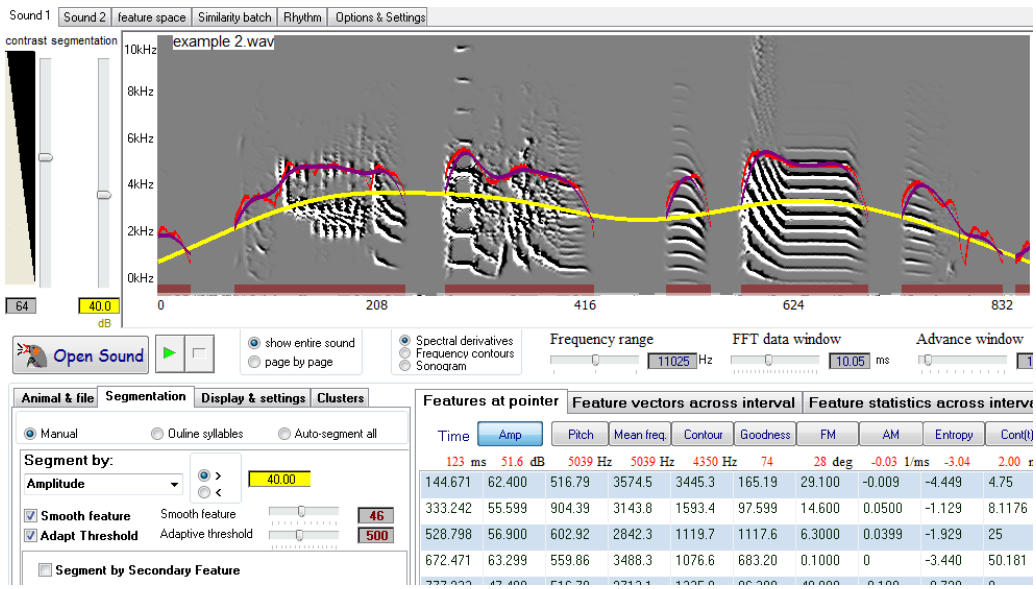
feature is smoothed. For example, smoothed amplitude will appear in purple, while original amplitude graph is red.



- **Using an adapted amplitude threshold to segment sounds**

Check "Adapt Threshold" checkbox, the Adaptive threshold slider will show. Select the level to which to adapt the threshold, such that threshold increases with the mean amplitude. This allows better detection of low amplitude sounds (e.g., in the beginning of the file). The lowest level is 200, which means that threshold is adapted every 200 slices, making it of the similar shape as smoothed feature graph. The highest level is 900, the threshold is adapted every 900 slices, making it almost a straight horizontal line. That is, at the extreme, smooth threshold becomes identical to the fixed threshold. Adjusting the threshold level with the slider will move the adapted threshold so that the mean of the threshold is on the level selected by the slider (i.e. the adjusted threshold will move up and down as slider is moving). The threshold is indicated by the thick yellow line. As with fixed amplitude, the feature can be shown when greater than the threshold or less than the threshold.

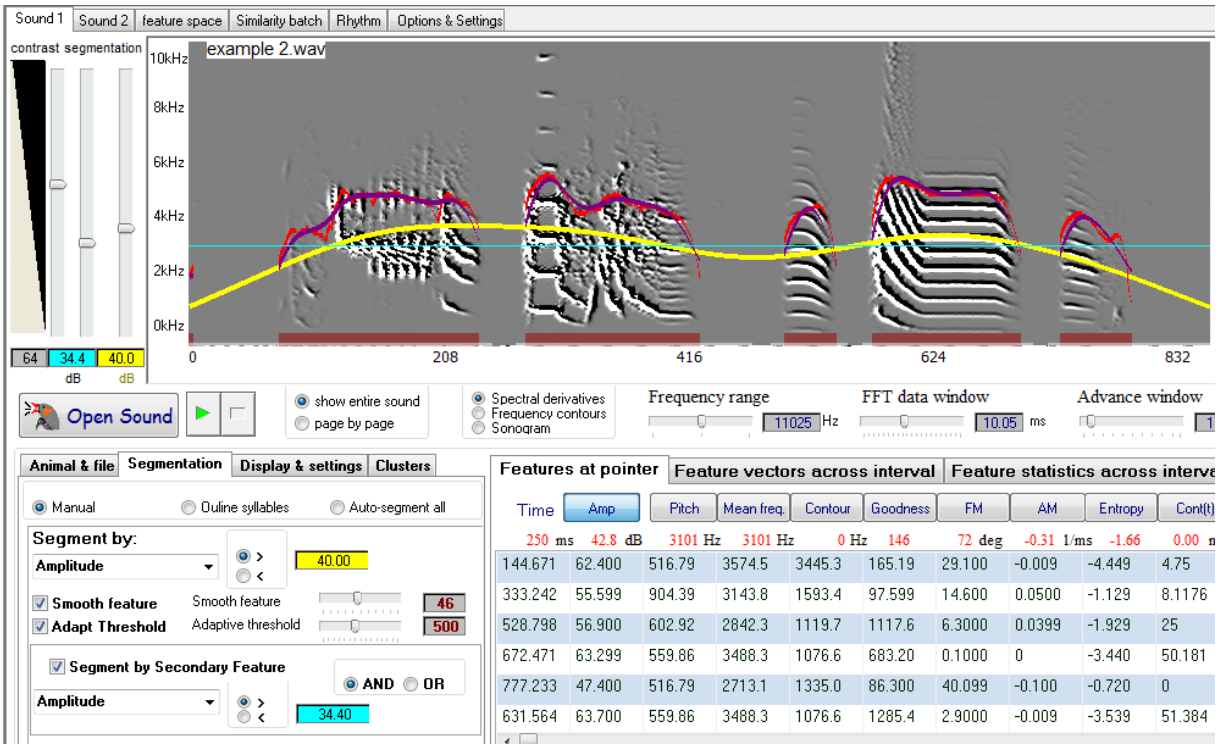
warning: using adapt threshold in pagination mode (small windows) will affect the segmentation.



- **Using secondary segmentation to segment sounds**

Check "Segment by Secondary Feature" checkbox, the secondary feature controls as well as secondary threshold slider will show. Select a secondary feature to segment by. Select whether you want to segment by combining both features (Primary feature **AND** Secondary feature), or segment by either feature (Primary feature **OR** Secondary feature). Use the secondary threshold slider to select the threshold level. The secondary feature can be shown when greater than the threshold (if ">" option is selected next to secondary threshold) or less than the threshold (if "<" option is selected). The secondary feature may be the same as the primary feature (usually smoothed) or set up the threshold min and max by using primary and secondary thresholds.

For example, as seen in the figure above, our segmentation is too liberal in the edges. Using fixed amplitude threshold as a secondary feature solves this problem by imposing a minimum amplitude threshold:



- **Using Preview Mode**

Click Preview button located below the graph screen. The preview of the entire file will appear. In Page by Page mode, click on the portion of the sound in the preview screen that you wish to view in detail and/or segment. The graph screen will show the portion of the file highlighted in Preview screen.

- **Going forward/backward (page by page mode)**

Move the cursor to the right edge of the graph to make the Forward button 

appear. The button is not visible if the sound file is at the end.

Move the cursor to the left edge of the graph to make the Back button 

appear. The button is not visible if the sound file is at the beginning.
Open Explore & Score, Ensure that "fine segmentation" is turned off (see Fig 1 below)

Fig 1: Fine Segmentation "off"

Open your sound file or use Example1 (found in the sap directory) and then move the amplitude threshold slider (the one closest to the frequency axis) up to about 43Db:

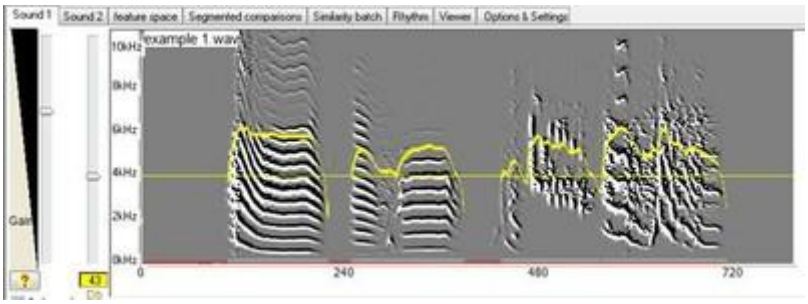


Fig 2: Amplitude Threshold Slider

The **yellow curve** shows the **amplitude**, and the **strait yellow line** is the **threshold**. Amplitude is shown only when above threshold. **Syllable units** are underlined by a light blue color below them, and **bouts** are underlined by a red color.

Note segmentation outlines at bottom of sounds:

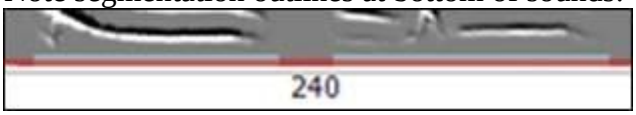


Fig 3: Segmentation outlines

Additional constraints on segmentation can be set, so at to reject some sources of noise. Here is an example:
Set the "advance window" slider to 2ms, and set the amplitude threshold to 30Db. Open example3:

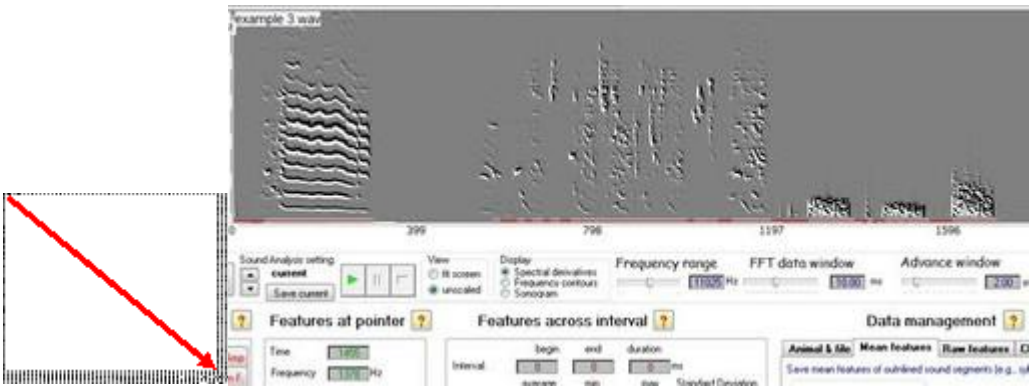


Fig 4: Frequency of syllables

As shown, the last 3 ‘syllables’ are actually low frequency cage noise. Move the mouse to just above the noise level while observing the frequency value at the Features at Pointer panel (see red arrow). As show, most of the noise is below 1500Hz, whereas most of the power of the syllables is above that range.
We are not going to filter out those low frequencies. Instead, we will use this threshold to make a distinction between cage noise and song syllables: Click the “Options & Settings” tab. Turn the high pass noise detector on and change frequency to 1500Hz:

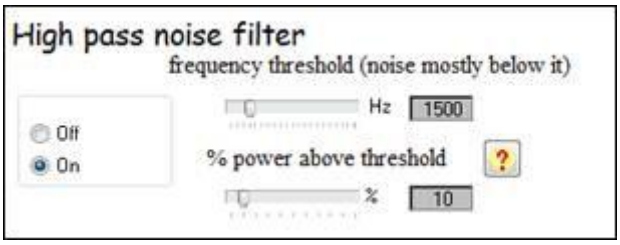


Fig 5: High Pass - Noise Detector

Go back to sound 1, and click update display below the sonogram image:

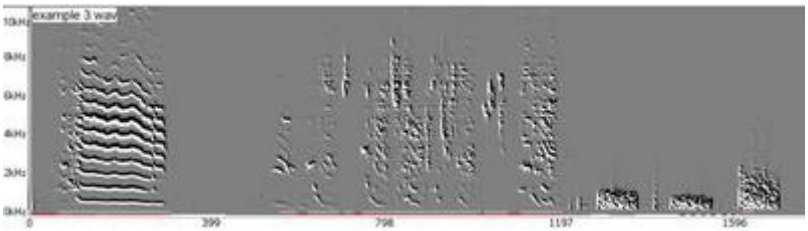


Fig 6: Noise - No longer detected

Note that the most of the noise is no longer detected as vocal sound:

Fig 7: Noise isolated from vocal sounds

This filter does not affect any analysis of the remaining vocal sounds. This is because we set the noise detector filter as an additional criterion (on top of the amplitude threshold) to eliminate ‘syllables’ where at more than 90% of the energy is at the noise range. There are several other controls that affect segmentation indirectly. Those include the FFT

window, advance window, the band-pass filters on feature calculation, etc. Here is an example of using the band-pass filter: turn the noise detector off and update the display so that the noise is once again detected as vocal sound. Then move the right sliders as shown:

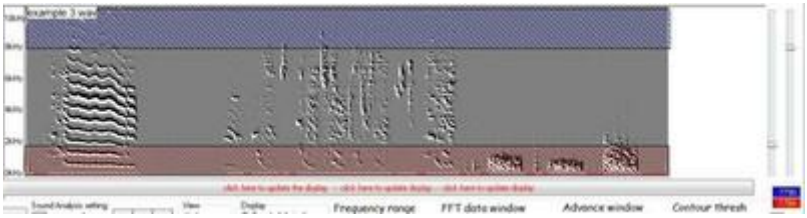


Fig 8: Noise isolated from vocal sounds

Now click update display:

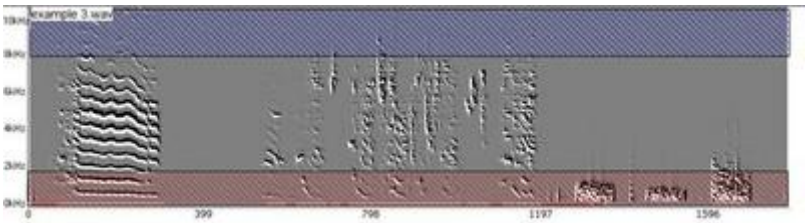


Fig 9: Noise isolated from vocal sounds

And the outlines under the noise that is below the detection band should disappear. Note, however, than now all features for all syllables are only computed based on the band-pass filter that you set. Namely, frequencies outside the band are ignored across the board.

1. Segmentation by a dynamic amplitude threshold

One limitation of static amplitude threshold is that when an animal vocalizes the "baseline" power often change as vocalization becomes more intense. For example, open the file "thrush nightingale example 1" with 3ms advance window and 0 amplitude threshold. Let's observe the amplitude envelope of this nightingale song sonogram:

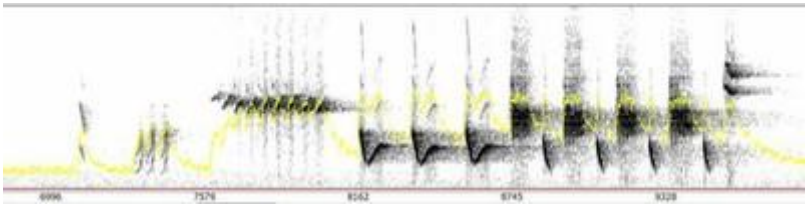


Fig 10: Noise isolated from vocal sounds

And let's also look at the spectral derivatives, and a certain threshold indicated by the black line:

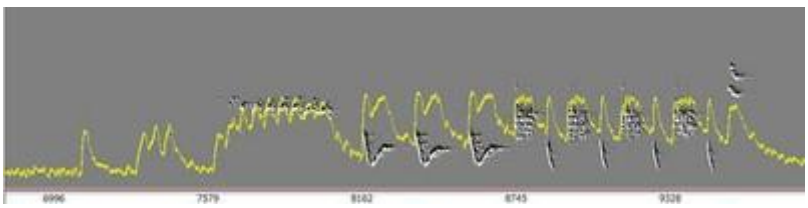


Fig 11: Noise isolated from vocal sounds

It is easy to see that no fixed threshold can work in this case (see arrows). To address this, turn "fine segmentation" on. A new slider – called Diff – should appear between the amplitude threshold slider and the display contrast slider. Set it to zero (all the way up). In the fine segmentation box (bottom left of the SAP2 window) set the course filter to 500, fine filter to 0, update display and click filters:

Fig 12: White curve - coarse amplitude filter, black line - fine filter, and segmentation

The white curve shows the coarse amplitude filter, which is the dynamic (adaptive) threshold. The black line is the fine filter, which is the same as amplitude in this case. The segmentation is set by the gap between them, where diff=0 means that we segment when the black line touches the white line, namely vocal sound is detected when the fine filter is higher than the course filter.

We can see that all syllables are now detected and segmented, but there are two problems:
1. The diff detect low amplitude sounds, but also falsely detect small changes in background noise as sounds (look at the beginning of the file).
2. Segmentation to syllables is often too sensitive and unreliable because each small modulation of amplitude may cause segmentation.

A simple way of avoiding false detection of silences is to impose some minimal fixed amplitude threshold on top of the filters. To do this, set the Db threshold to 24:

Fig 13: No more silence is detected

As shown, no more silences are detected as sounds.

To decrease the sensitivity of segmentation we can use two methods. One is to make the Diff more liberal – allowing the detection of sounds even when the fine filter is slightly below the coarse one. Set the diff to -2.5 gives this result:

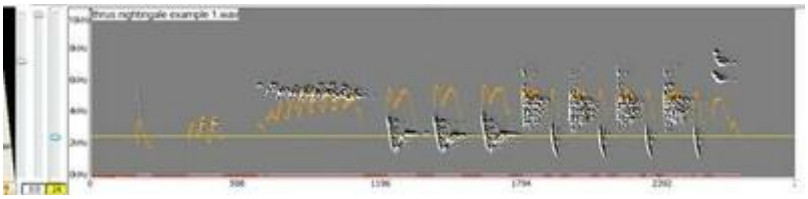


Fig 14: Setting the "diff filter to -2.5"

It is often better approach is to set the fine filter a bit coarser. For example setting fine filter to 5, keep course filter at 500, and setting the diff slider to -1.5 gives this segmentation:

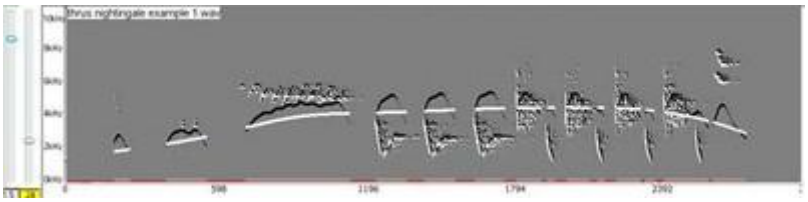


Fig 15: Sound with fine filter set a to coarser setting

As shown, we have achieved a rather reliable segmentation despite the wide range of amplitudes in this song.

Setting the Spectral Parameters

SAP2011 supports sounds digitized at any rate, including ultra sound. The frequency range is adjustable in this version to 3 ranges: full spectral range (sampling rate/2, which is as high as frequency analysis can go), half range, or quarter range.

You can change the contrast of the derivative display without changing the contrast of the feature curves by moving the Gain slider on the left.

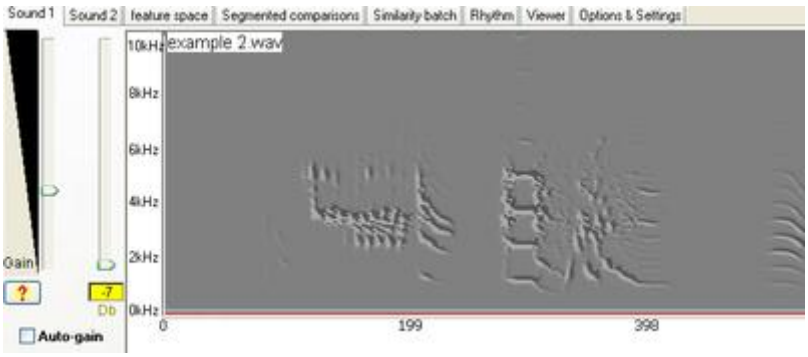


Figure 1: Contrast slider at the minimum

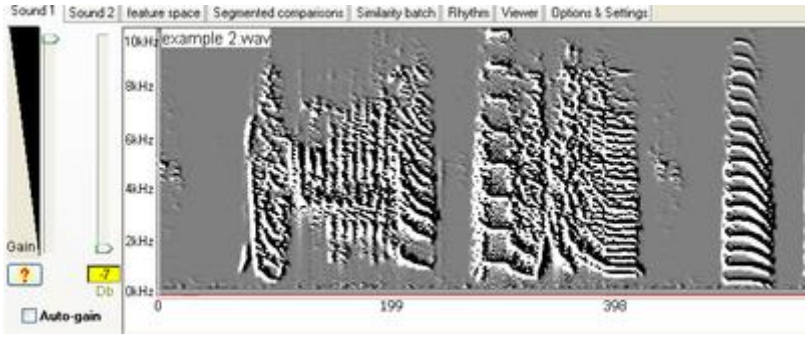


Fig 2: Contrast slider at the maximum

This manipulation has only visual effect.

If however, the sounds you are analyzing are of very low amplitude (e.g., forcing you to increase the display gain all the way up), you may want to boost them prior to analysis. You can do this by going to the Display & Settings tab > check "boost sound amplitude". Be aware though, that this might virtually "clip" sounds of high amplitude.

Note that changing the amplitude of wave files data should have no effect on song features except for amplitude of course, as long as data are not clipped (namely, as long as the numbers are within the 16bit range).

Spectral parameters

There are 4 parameters that you can change within the 'Explore & Score' module, the Frequency range, FFT data window, Advance window and Contour threshold. Be aware that the scope of those changes is global – that is all other modules will apply this change.

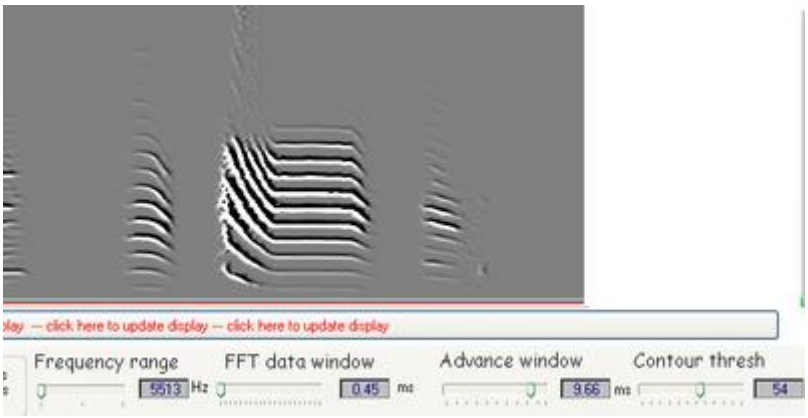


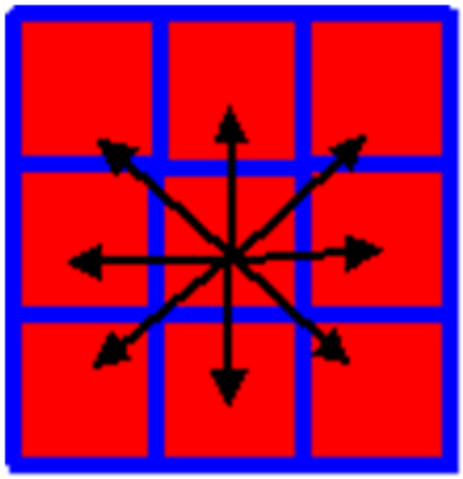
Figure 3: Four parameters that you can change within the 'Explore & Score' module

Start by changing the size of the FFT data window. Reducing it to 5ms, and reopening the sound will present a different time-frequency compromise. Note that feature calculation & scale are affected by such manipulation.

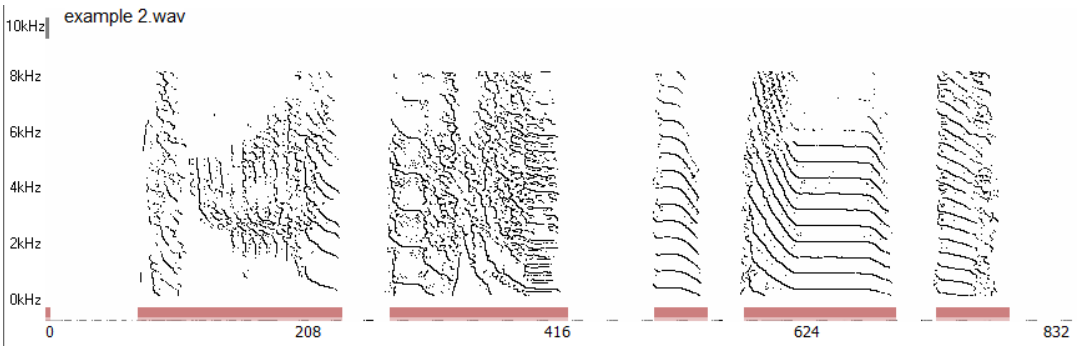
Return the data window size to 9.27ms, and change the advance window to 0.4ms, and sound display, as well as feature calculation, becomes more elaborate. The only feature that will change scale is AM: Try identifying and comparing the first two syllables across the two displays. Change the advance window back to 1.2ms.

Frequency Contours

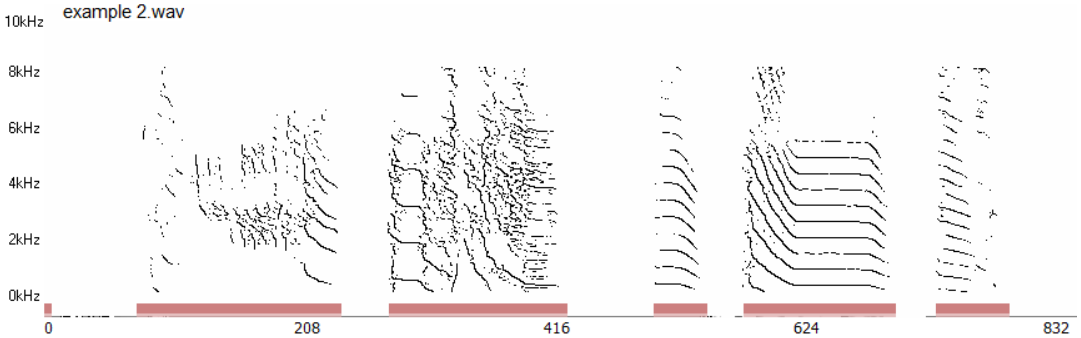
SAP2011 calculates frequency contours by detecting zero crossings of the spectral derivatives. In order to reject artifact, we require that the contours passed a dynamic contrast threshold, T, calculated for each time window t_i *and frequency* f_i as:
$T(t_i,f_i)=abs(Wiener\_entropy(t_i)) / abs(f_i-mean\_frequency(t_i))$,
were T' is a user defined threshold. Therefore, the detection threshold is weighted by the distance from the mean frequency (the gravity center of the frequencies) and by the width of the power spectrum. A pixel in the time frequency space is defined as contour if i) there is zero crossing between the neighboring pixels at any one of the 8 possible directions (see diagram below) and ii) both neighboring pixels (in the direction of the zero crossing) are larger than *T*.



Here is an example with T'=10:



and with T'=50:



Chapter 4: The Song Features

Chapter 4 - Content:

Introduction to song features

We now take a deeper look into the acoustic features and the measures we derive from them. The first step of the analysis is to reduce the sound spectrograph to a few simple features. The entire SAP2011 analysis is based on those features – that is, the features replace the sonogram.
Why use features? Many of the previous attempts to automate the analysis of sound similarity used a sound-spectrographic cross-correlation as a way to measure the similarity between syllables: correlation between the spectrograms of the two notes was examined by sliding one note on top of the other and choosing the best match (the correlation peak). However, measures based on the full spectrogram suffer from a fundamental weakness: the high dimensionality of the basic features. For example, cross-correlations between songs can be useful if the song is first partitioned into its notes, and if the notes compared are simple. But even in this case, mismatch of a single feature can reduce the correlation to baseline level. For example, a moderate difference between the fundamental frequencies of two complex sounds that are otherwise very similar would prevent us from overlapping their spectrogram images (a vertical translation will not help since the harmonics won't match).

The cross-correlation approach (as mentioned above) requires, as a first step, that a song be partitioned into its component notes or syllables. This, in itself, can be a problem. Partitioning a song into syllables or notes is relatively straightforward in a species such as the canary in which syllables are always preceded and followed by a silent interval. Partitioning a song into syllables is more difficult in the zebra finch, whose song includes many changes in frequency modulation and in which diverse sounds often follow each other without intervening silent intervals. Thus, the problems of partitioning sounds into their component notes and then dealing with the complex acoustic structure of these notes compound each other. The analytic approach of Sound Analysis addresses both of the above difficulties. It achieves this by reducing complex sounds to an array of simple features and by implementing an algorithm that does not require that a song be partitioned into its component notes.
In recent years, several alternative approaches to sound analysis were published and made available, including improved cross correlation methods, and other means of extracting features. Other approaches are based on compression by principal component analysis, and other objective methods for dimensionality reduction. If there is one advantage to using Sound Analysis Pro, it is the usage of simple, and well understood features. When you see that two syllables differ in pitch, Wiener entropy or in frequency modulation, it is easy to imagine how the vocal quality varies, and so it is more likely that you will be able to develop a good intuition, and perhaps some understanding of the difference between the two sounds.

Articulation based analysis

The analytic framework of *Sound Analysis* is rooted in a robust spectral analysis technique that is used to identify acoustic features that have good articulatory correlates. The acoustic features that we chose to characterize zebra finch song are represented by a set of simple, one-dimensional measures designed to summarize the multidimensional information present in a spectrogram. A procedure for measuring similarity, based on such an analytic framework has the following advantages:

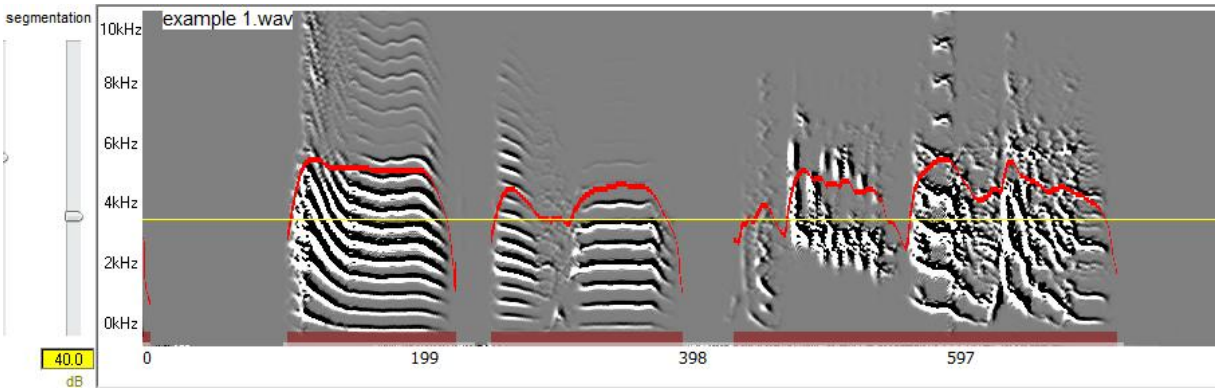
1. It enables the examination of one acoustic feature at a time, instead of having to cope with the entire complexity of the song of two birds. A distributed and then integrated assessment of similarity across different features promotes stability of scoring.

2. It is analytic: it evaluates each feature separately and tells you not only that two sounds are similar or different - by also in what sense they are similar/different. For example, it allows examination of how each of the features emerges during development and is affected by different experimental manipulations.

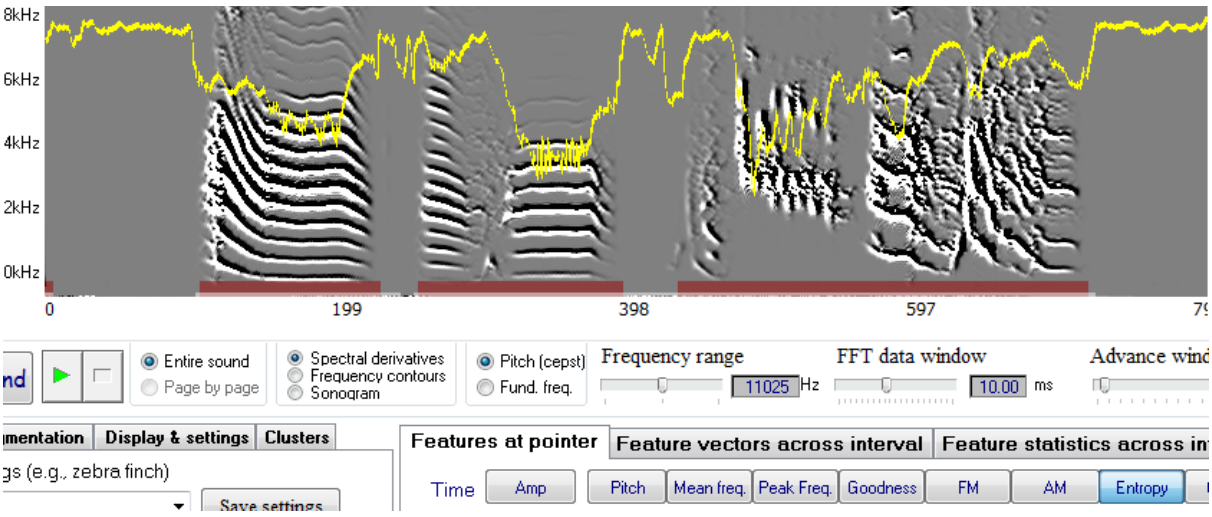
Note that all the features presented here (except for amplitude) are *amplitude invariant*. That is, the amplitude of the sound recorded does not affect them and hence the distance between the bird and the microphone should have only minor effect as long as the shape of the sound wave has not been distorted by the environment.

Viewing the features

SAP2011 allows you to visualize features both graphically and numerically as shown below (see also video tutorial).
In Explore & Score, open sound 'example 1' and set segmentation amplitude threshold to 40.0 as shown below:



The thin yellow line represents the amplitude threshold, and the red curve is the amplitude. The brass color outlines represent syllable boundaries as defined by the threshold. To see graphs of other features, select the 'features at pointer' tab, and click on the features buttons one by one, while observing the sonogram image:



Just below those buttons you will see numbers that represent the feature values at the location of the pointer. Move the pointer around the spectral image and observe the changing values of pitch, FM, as well as song time and frequency at the current pointer position. Clicking on the sonogram saves the feature values into the table below. Clicking on the saved values will show a red outline at the location of each feature that you saved.

Amplitude

At the time domain, the amplitude of the sound waveform is the absolute deviation from the zero, which can be estimated by the root mean square (RMS) of the signal. Of course, determining the zero baseline (silence) is important, but calibration it is not always an issue while recording animal vocalization recording, and usually we care about the relative intensity of the sound, which is estimated at the frequency domain in units deciBels dB

$$Amp = 10\log_{10}\sum_{f} P_f - baseline$$

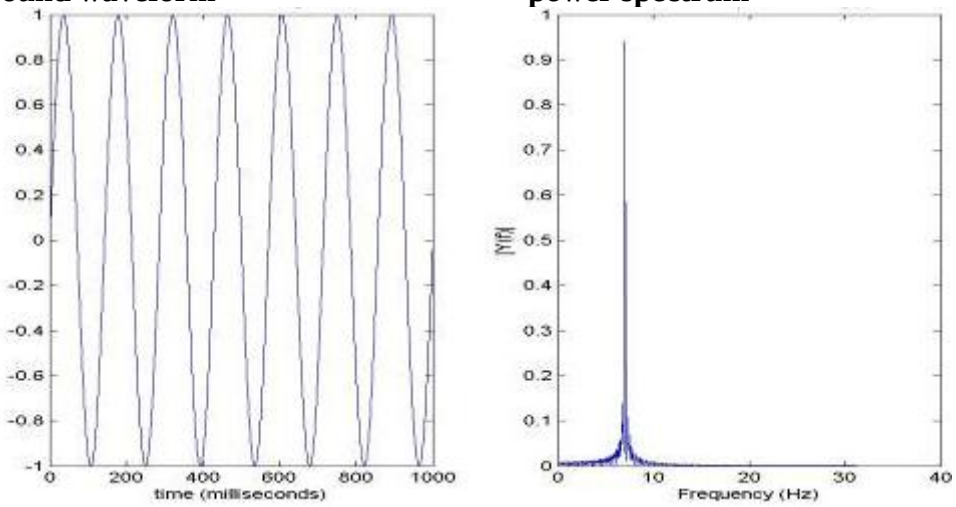
where P_f is the power at any one frequency and were baseline is (arbitrarily) set to 70dB as a default.

Amplitude is measured in dB scale, but note that the power is un-scaled, hence the baseline is arbitrary. You can adjust the baseline value in options->feature calculation->amplitude baseline. *SAPII* uses amplitude as one of the criteria for segmenting syllables and adjustments are based on relative values (namely, you adjust the amplitude slider to be slightly above the cage noise observed in the spectral image). Other than that, however, amplitude is not used for any other procedure.

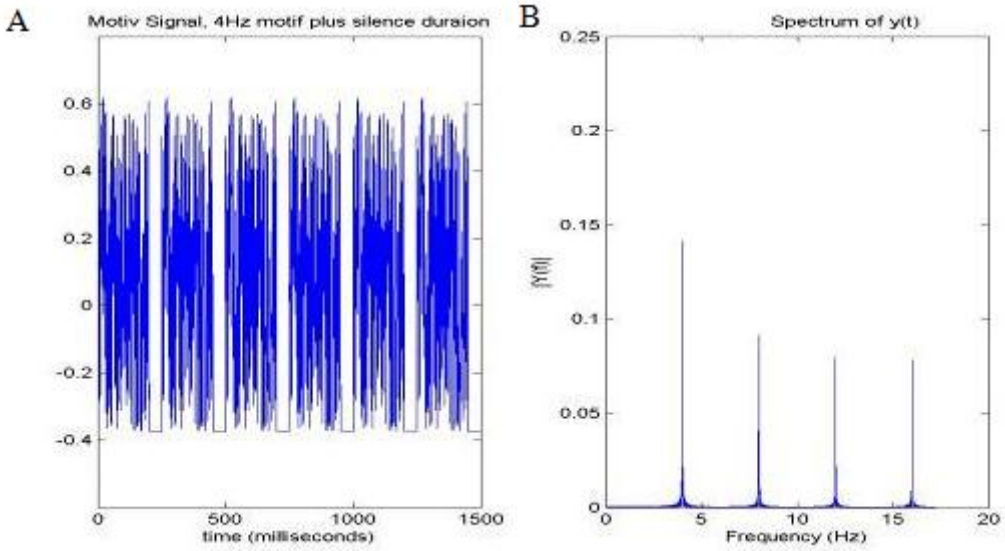
Pitch related measures

The term pitch is used to describe the perceived tone of sounds (high, low, etc). Quantitatively, pitch estimates are measures of the period of oscillation. It is the only feature that requires careful adjustments depending on the species and acoustic structure. When the spectral structure is simple, as in a whistle, the pitch can be easily estimated as the (only) peak in the power spectrum. the location of this peak can be assessed by one of two features: peak frequency -- the frequency of highest power, or the mean frequency -- the gravity center of the power spectrum. However, things get more complicated when the power spectrum is composed of several peaks:
In the case of a pure sinusoid, all the energy is centered in one frequency and we see a sharp peak in the power spectrum:

sound waveform **power spectrum**



However, a non-sinusoidal periodic signal appears as an harmonic series of peaks in the power spectrum. For example, if we create a short vector of random numbers, create several copies of that vector, and glue those head to tail to create a long vector, and then compute the spectrum of that vector, the result looks like this:



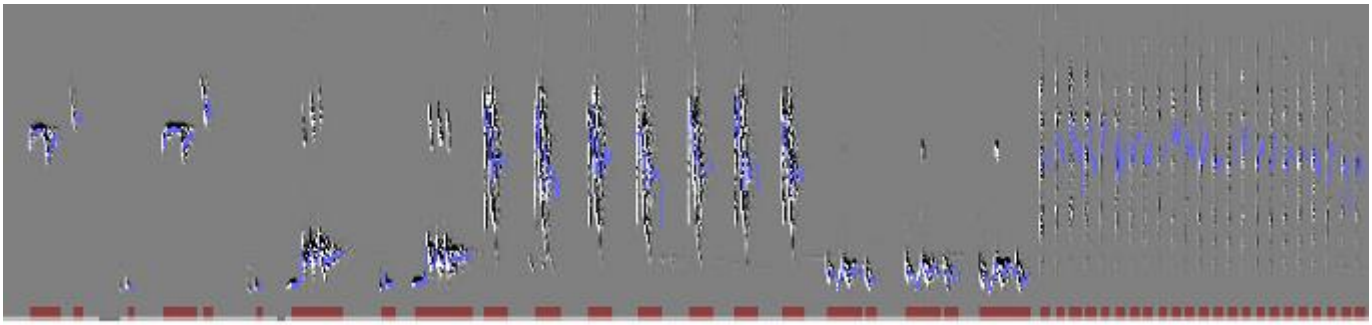
It this case, the perceived pitch is usually the lowest common denominator of these peaks, called fundamental frequency. Harmonic pitch is an estimate of the fundamental-frequency of a complex sound composed of harmonically related frequencies.

The main challenge is the automatic distinction between tonal pitch and harmonic pitch. In previous version of SAP we used the cepstrum (spectrum of log spectrum) to detect harmonic pitch, and dynamically shifted our pitch estimate to the mean frequency when harmonic pitch was undetectable. In the current version of SAP we include the YIN method for fundamental frequency estimate (by Alain de Cheveigne), which is also used by the Michael Brainard lab to estimate pitch. In SAP2011 we use the term pitch for our cepstrum based estimate, and 'fundamental frequency' for the YIN estimate.

In general, pitch is difficult to estimate, but nevertheless, pitch is a central feature of song, and with careful adjustments, one can often obtain a good estimate. Furthermore, when we calculate the mean pitch of a syllable, we adjust the weight of each time window by the goodness of pitch, which often stabilizes the mean pitch estimate of that syllable type.

Mean frequency

Mean frequency is a pitch measure, that assesses the center of the distribution of power across frequencies.
Here is an example of mean frequency estimates in a thrush nightingale song:



As indicated by the purple curve, the mean frequency provides a smooth estimate of the concentration of spectral power. This estimate is often (but not always) superior to peak frequency, which might 'jump' from one harmonic to the next.
Formal definition: Mean frequency is an estimate of the central tendency of the derivative power distributions. Derivative power is calculated by adding the partial derivatives of powers, namely:

$\partial P_f dt$ as the time-derivative, and

$\partial P_f df$ as the frequency-derivative of frequency f.

Note that the mean is of the squared spectral derivative. A similar result could be obtained with power, but empirically, we find the derivative to be

$$\hat{f} = \frac{\sum_f f(\partial P_f^2 dt + \partial P_f^2 df)}{\sum_f (\partial P_f^2 dt + \partial P_f^2 df)}$$

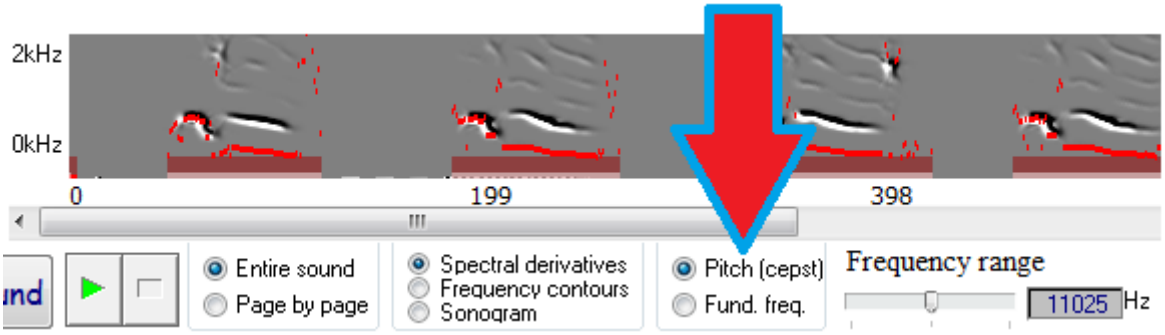
In contrast to peak-frequency, mean frequency provides a smooth estimate of the center of derivative power. In contrast to peak frequency, mean-frequency does not ‘stick’ to any frequency trace, as shown in the example above.

Peak frequency

Peak frequency is simply the frequency of maximum power. For vocal sounds composed of a pure tone (sine wave) embedded in some environmental noise this is often the best pitch estimate. For other broadband and harmonic sounds, mean frequency and fundamental frequency often work better.

Fundamental frequency

Fundamental frequency is an estimate of the periodicity of the sound. In the power spectrum it is the lowest common denominator of the harmonic peaks. SAP2011 offers two methods for computing it, to select in Explore & Score, use this radio box:



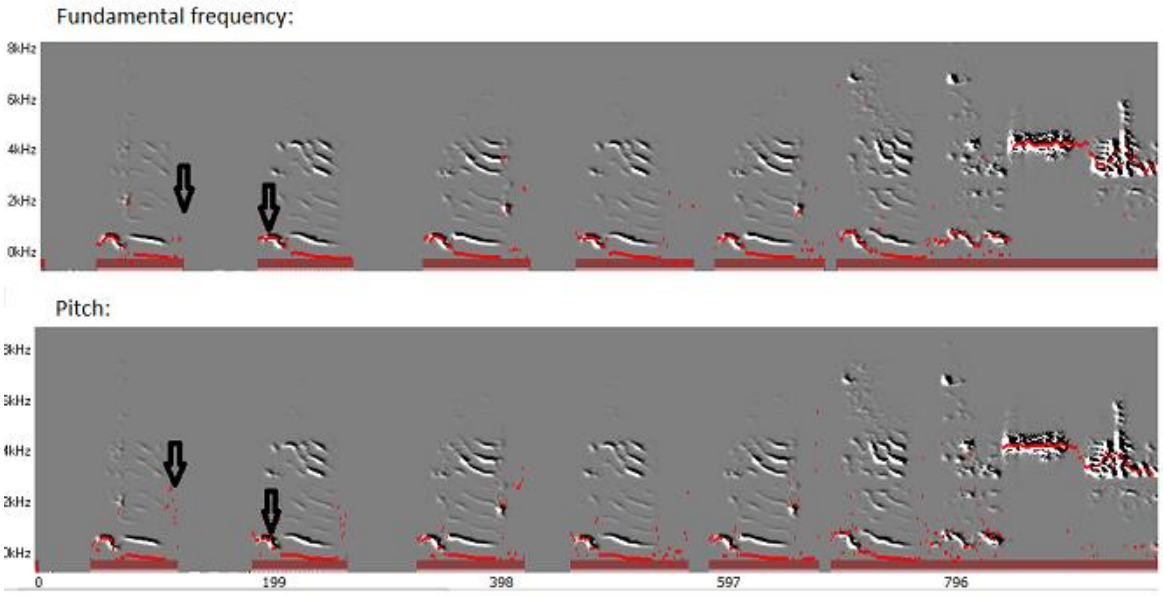
Pitch: a measure used in previous versions of Sound Analysis, is a combo-measure of harmonic pitch estimate (cepstrum) and mean frequency. The cepstrum peak is usually calculated upon the spectrum of log spectrum, but instead of log spectrum we use the derivative spectrum. At any given time window, pitch might be either harmonic, sinusoidal (whistle) or not well-defined. In the two later cases, mean frequency provides an appropriate pitch estimate.
Hence, at time window t, we calculate pitch according to three threshold parameters T1, T2, T3.

ff_t =period where FFT(spectral_derivatives $_t$) is the highest. (this is our cepstrum pitch estimate), and then:
if $\{ ff_t > T1 \;||\; (\text{Wiener entropy} < T2 \;\&\; \text{Goodness of pitch} < T3) \}$ *Pitch* $_t$ *= mean frequency*
Otherwise Pitch $_t = ff_t$

SAP2011 distinguishes between the two based on three considerations: first, harmonic pitch is often frequency bounded, e.g. in zebra finches we rarely see harmonic sounds with a fundamental higher than 1800Hz. Therefore, we reject a cepstral estimate higher than this threshold and prefer the mean frequency estimate. Second, if the goodness of pitch is very low, pitch is unlikely to be harmonic. Third, if both goodness of pitch and Wiener entropy are low, pitch is even less likely to be harmonic. You can manipulate those parameters in the options. Note that harmonic pitch is typically low, hence we reject harmonic pitch estimates that are higher than the natural range (e.g., in zebra finch harmonic pitch rarely approaches 2kHz).

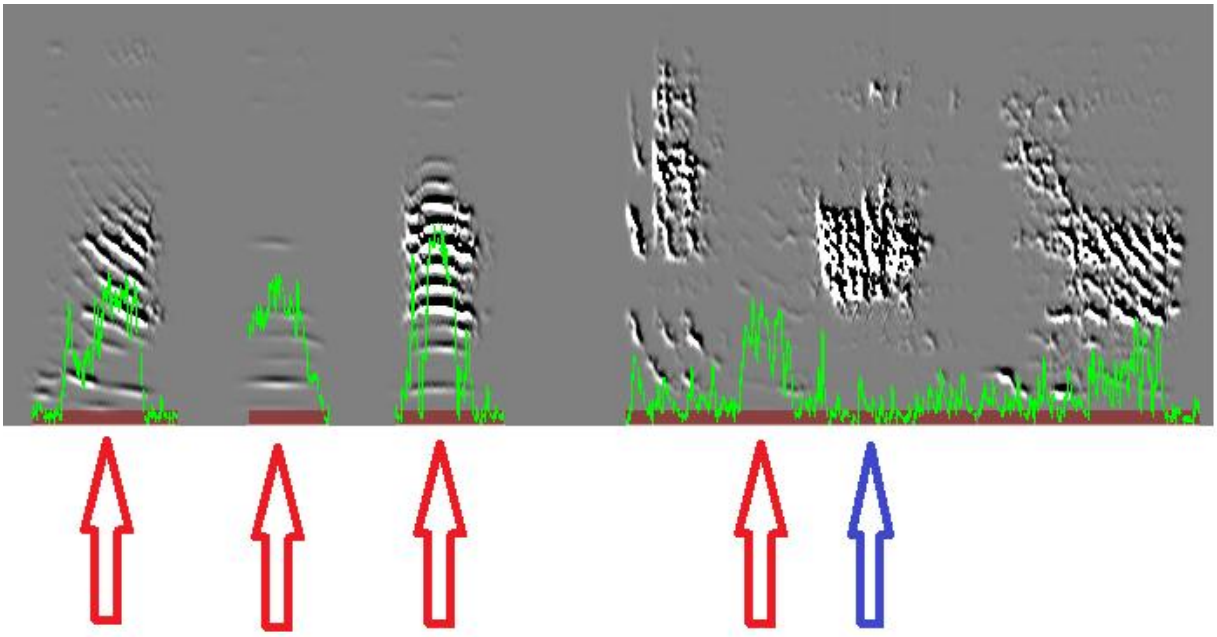
Fundamental frequency (YIN algorithm): This measure is often (but not always) more accurate than the pitch, and it is more computationally expensive (about 40% increase in feature computation time of SAP2011). The algorithm is a so called 'time domain approach', which is based on auto-correlation and does not involve explicit spectral analysis (although auto-correlation and power spectrum and linear transformations of each other). It also include a smoothing algorithm that can, in principle, provide accuracy that might be better than the nyquist limitation by means of interpolation. The only YIN algorithm parameter is minimum frequency. By default SAP2011 set it at 300Hz. For more details, see A. de Cheveigné and H. Kawahara. YIN, a fundamental frequency estimator for speech and music. The Journal of the Acoustical Society of America, 111:1917, 2002. doi:10.1121/1.1458024.

Pitch version Fundamental frequency comparison: The following examples demonstrate the pros and cons of each approach. As shown, the fundamental frequency estimate is somewhat more accurate and less noisy than the pitch (compare values around the arrows)



Goodness of pitch

Goodness of Pitch is an estimate of harmonic pitch periodicity. High goodness of pitch can be used as a detector of harmonic stack, modulated or not. For example, in the figure below the red arrows indicate harmonic sounds and the blue one a clearly non-harmonic one:



Formal definition: *Goodness of pitch is the peak of the derivative-cepstrum calculated for harmonic pitch*

Units are comparable to amplitude and can be converted to dB by subtracting a baseline and converting to log scale (not implemented in current version).

This is a measure of how periodic the sound is. It only captures the 'goodness' of harmonic pitch, whereas both noisy sounds and pure tones give low values. Noisy sounds, however, also give high entropy.

Therefore the combination of Wiener entropy and goodness of pitch is useful.

Wiener entropy

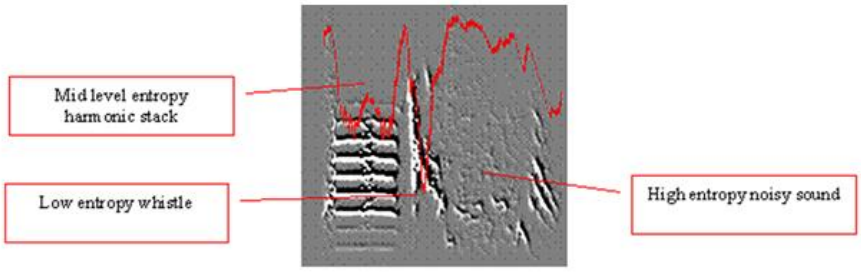
Wiener entropy is a measure of the width and uniformity of the power spectrum. Noise is typically broadband with sound energy smeared rather smoothly within the noise range, whereas animal sounds, even when multi-harmonic, are less uniform in their frequency structure. Wiener entropy is a pure number, that is, it does not have units. On a scale of 0-1, white noise has an entropy value of 1 and complete order, and a pure tone has an entropy value of 0.

To expand the dynamic range, the Wiener entropy is measured on a logarithmic scale, ranging from 0 to minus infinity (white noise: *log*1=0; complete order: *log*0=minus infinity). The Wiener entropy of a multi-harmonic sound depends on the distribution of the power spectrum (see figure below).

A narrow power spectrum (the extreme of this is a pure tone) has a large, negative Wiener entropy value; a broad power spectrum has a Wiener entropy value that approaches zero. The amplitude of the sound does not affect its Wiener entropy value, which remains virtually unchanged when the distance between the bird and microphone fluctuates during recording. Yet, the entropy time series (or curve) of a song motif is negatively correlated with its amplitude time series. This is because noisy sounds tend to have less energy than tonal sounds. A similar phenomenon has also been observed in human speech, where unvoiced phonemes have low amplitude.

Wiener entropy may also correlate with the dynamic state of the syringeal sound generator, which shifts between harmonic vibrations and chaotic states. Such transitions may be among the most primitive features of song production and maybe of song imitation.
Formal definition: Wiener entropy is a pure number defined as the ratio of geometric mean to arithmetic mean of the spectrum:

$$W = \log\left(\frac{\exp[\int df Log(S(f))]}{\int df S(f)}\right)$$



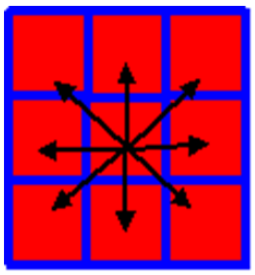
Spectral continuity

SAP2011 uses two contour-derived features: continuity over time, and continuity over frequency. Frequency contours can be detected by the zero crossings of the spectral derivatives. In order to reject artifact, we require that the contours pass a certain contrast threshold. We use a dynamic threshold T calculated for each time window t_i *and frequency* f_i as:

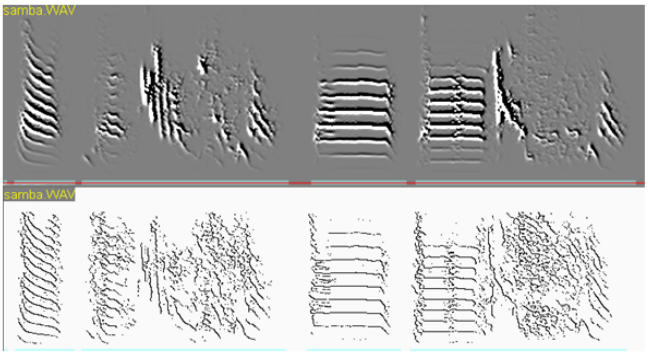
$$T(t_i, f_i) = T' \frac{abs\{Wiener\_entropy(t_i)\}}{abs\{f_i - mean\_frequency(t_i)\}}$$

were T' is a user defined threshold

Therefore, the detection threshold is weighted by the distance from the mean frequency (the gravity center of the frequencies) and by the width of the power spectrum.



A pixel in the time frequency space is define as contour if i) there is zero crossing between the neighboring pixels at any one of the 8 possible directions (see diagram above) and ii) both neighboring pixels (in the direction of the zero crossing) are larger than *T*.



Continuity Over Time: For each time window t_i we detect all the contours and measure the duration of each contour. The mean duration across the contours is the continuity over time.

Continuity Over Frequency: For each time window t_i we detect all the contours and measure the frequency range of each contour. The mean frequency range across the contours is the continuity over frequency.

Frequency & amplitude modulation

Frequency Modulation (Derivative Estimates) is defined as

$$FM = \arctan\left(\frac{Max_f \partial P_f^2 dt}{Max_f \partial P_f^2 df}\right)$$

that is, FM is the angular component of squared time and frequency derivatives. This measure gives an absolute (unsigned) estimate of frequency modulation.

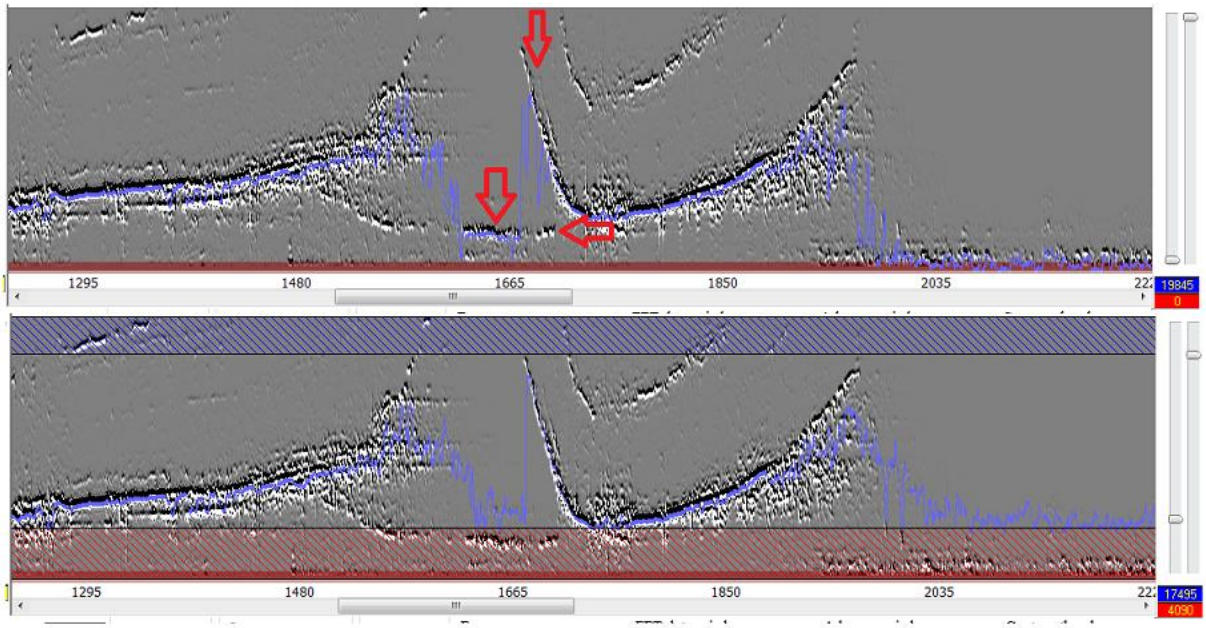
Frequency modulation is estimated based on time and frequency derivatives across frequencies. If the frequency derivatives are much higher than the time derivatives, we say that FM is low and vice versa. Visually, FM is an estimate of the (absolute) slope of frequency traces in reference to the horizontal line. Note: *SAP2* calculates an absolute estimate of FM, but it is easy to change the definition to obtain signed FM.

Amplitude Modulation: *AM is simply the overall time-derivative power across all frequencies within a range. Units of AM are 1/t, SAP2011 does not scale AM, and time units are defined by the 'advance window' parameter.*

Amplitude modulation captures changes in the amplitude envelope of sounds. It is positive in the beginning and negative at the end of each sound, with the sound unit being either a syllable or a change of vocal state within a syllable.

Bandpass frequency filtering

Many vocal sounds are strongly band-limited, namely the frequency range is narrow relative to the sonogram range (which you can only coarsely control in SAP2011). Take for example this vocalization by a dolphin in a social environment:



As shown in the top panel, the mean frequency estimate of a whistle (blue curve, see red arrows) is contaminated by a low frequency sound, which we are not interested in. Using the two band-sliders on the right, we limit the range of feature calculation to that of our whistle of interest, click recalculate and this mean is sufficient to obtain a more reasonable estimate of mean frequency.

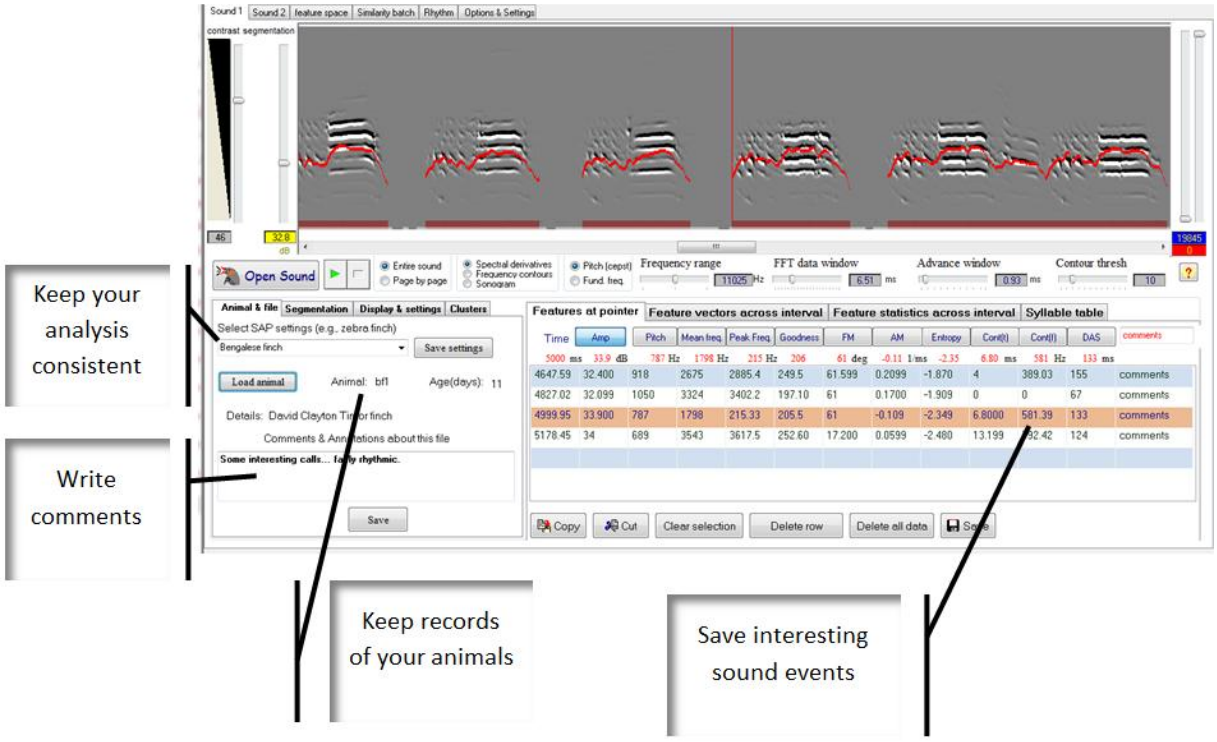
Chapter 5 - Content:

Introduction to Explore & Score

Explore & Score is where you open sound files, explore their structure, keep records of interesting events, score similarity, etc.
If you are new to SAP2011, we recommend you start by watching this video clip of Explore & Score. The next sections of this chapters will help you figure out who to get your analysis done most efficiently. But most important: SAP2011 is designed to facilitate your observations. It allows you to see more, and it provides quantitative measures that are always intuitive. So, when we are providing means of segmenting sounds we also make it easy to see how the algorithm works. When we compute features, we provide means of seeing their distributions in different types of sounds, etc. And the more you take advantage of these means you will start getting the most of SAP2011.

Here are some highlights of the Explore & Score interface:

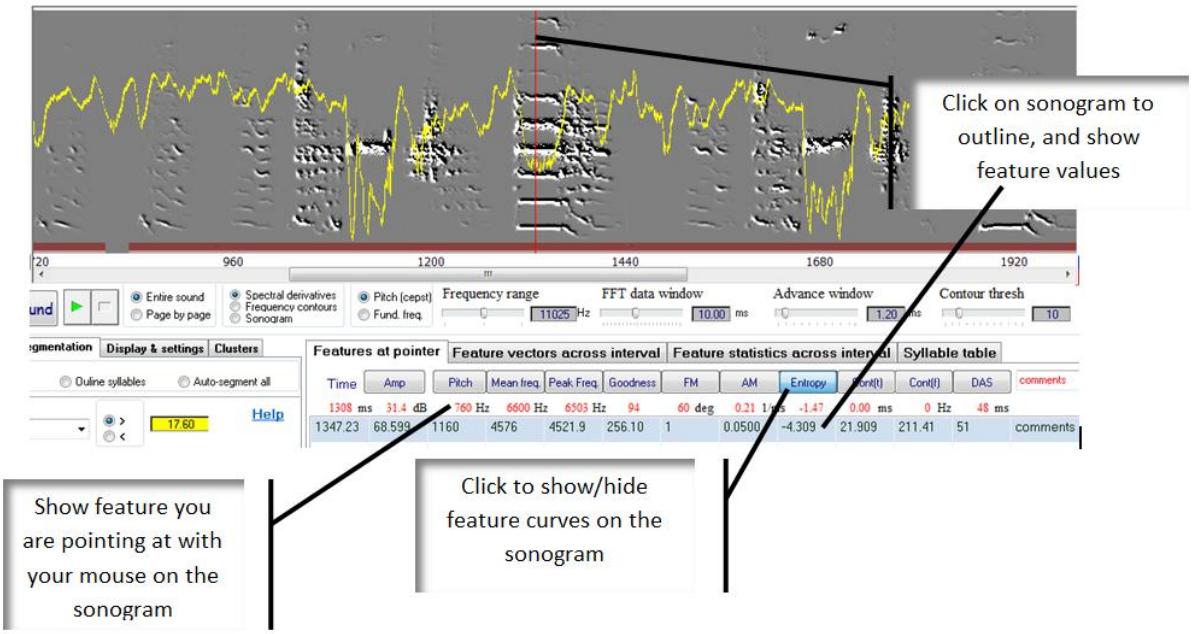
- SAP2011 provides you with several new ways of organizing your data and keeping track of your investigation. The Animal & file tab allows you to save different settings of SAP2011 parameters for different projects to keep your analysis consistent. Keeping data about the identity of your animal will allow automatic detection of the age of your animal at the time this file was created. You can write general comments about this file, and save interesting sound events by simply clicking on the sonogram image, as shown below.



Exploring song features

Almost everything that SAP2011 does is based on the features presented in chapter 4, including segmentation and charectarization of syllable structure, similarity measurements, clustering of syllables to type, Dynamic Vocal Development (DVD) maps, etc.

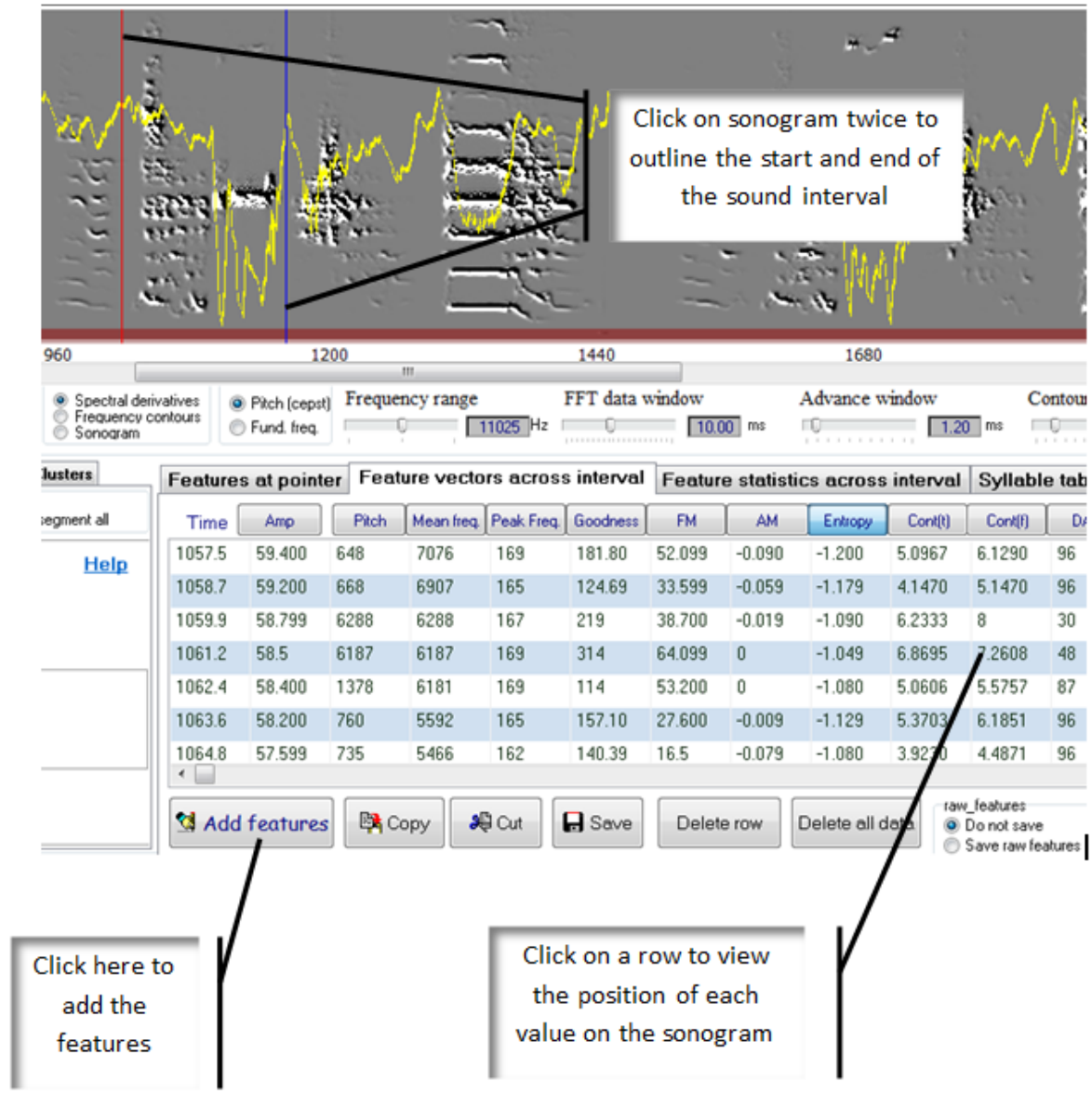
Here we present a few methods for exploring feature, viewing them using various methods and saving them.
SAP2011 provides several ways of looking at features. Graphically, you can look at feature curves on the sonograms, or look at feature scatter plots in 3D. Numerically, you can look at features values at certain time events, or look at feature time series over an interval of sound, or look at features first and second order statistics of song syllables or of intervals of sounds.
The simplest way of observing features is by clicking on a feature button, e.g., Wiener entropy, look at its curve on the sonogram, and observe the numerical value (using the "features at pointer" tab) as shown below:



you can then click on the feature data rows, to see the location of each feature value on the sonogram.
Writing comments is also advisable -- do it before you click on the sonogram.

Feature vectors

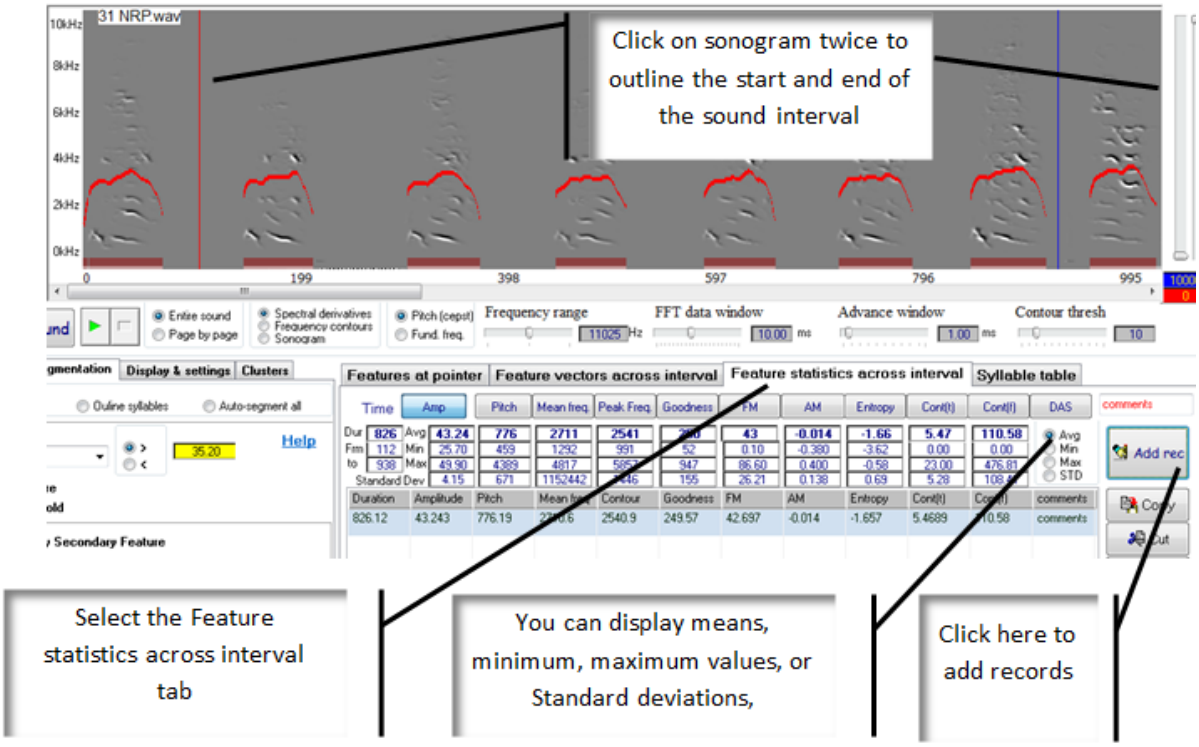
Viewing and saving feature time series is very easy in SAP2011: click on the 'feature vector' tab, and then click the beginning and end of a sound interval to outline it. Then click the 'add features' button as shown below:



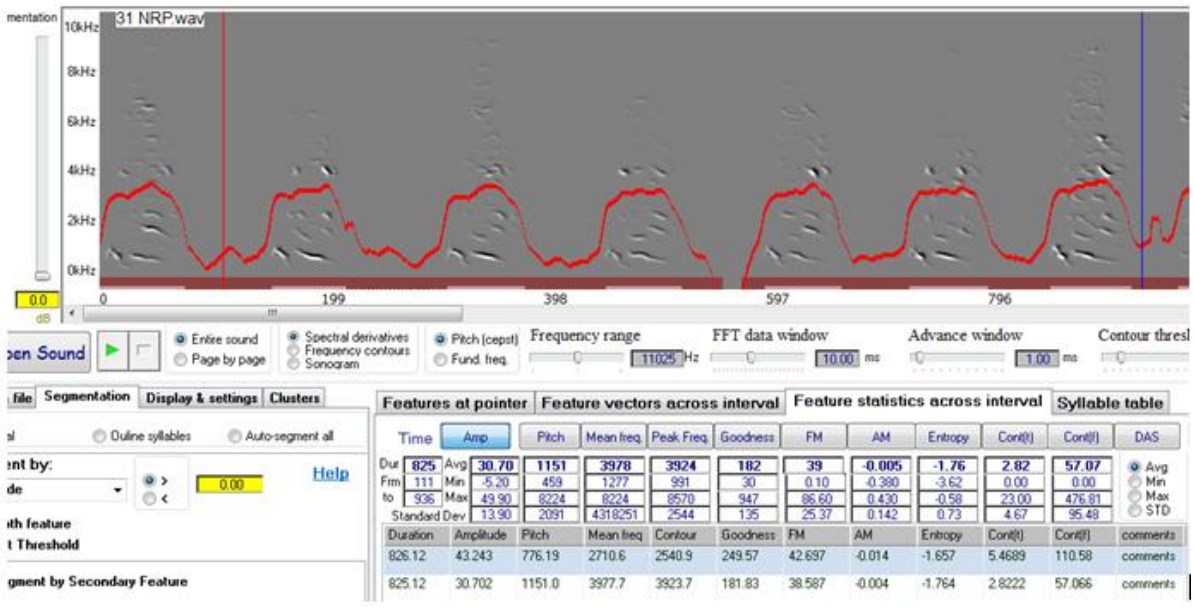
In the tabular list each record (line) represent features calculated for each FFT window within the sound interval you outlined. For example, setting 'FFT window' to 10ms and 'advance window' to 1ms, each record is of features calculated in time windows 1-10ms, 2-11ms, 3-12ms, etc. That is, for every one millisecond in time windows of 10 ms. Therefore, the curves of feature value you can see on the sonogram is presented here as a vector. You can copy it and paste it to Excel, or save it as an XML file and open it in Matlab.

Feature statistics across intervals

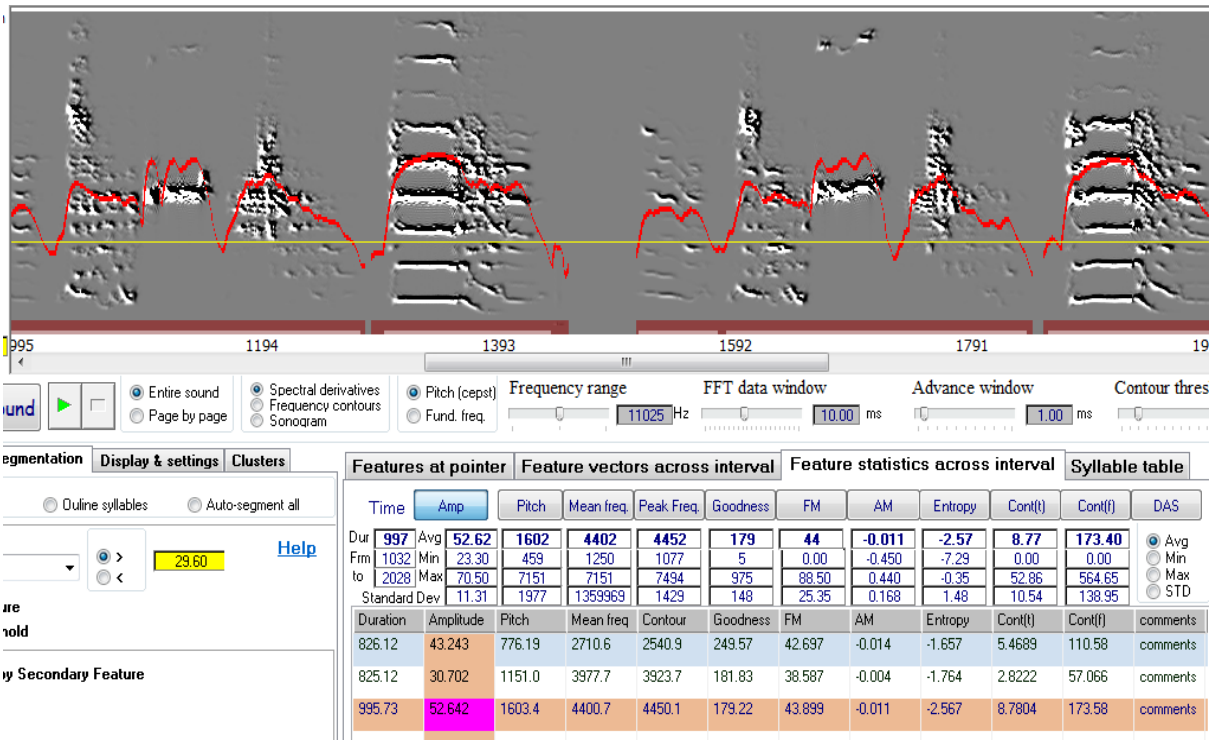
Until now we have only presented instantaneous feature values: either features at the mouse cursor pointer, or vectors of features over an interval. However, it is often desired to examine the distribution of features using simple first and second order statistics, so as to summarize the acoustic structure by calculating first and second order statistics, such as means and variance. This can be used to characterize natural sound units (notes, syllables, motifs, bouts) or just to look at the overall distribution of features over many sounds.
First select the 'Feature statistics across interval' tab. Then, to outline an interval, either click on the sonogram in two places, or click and drag. Click & Drag will automatically add record, whereas clicking in two positions must be followed by clicking 'Add record' to add data to the table:



The feature distribution of the introductory notes we outlined is now summarized in our record (bottom of the image above).
Important: Although we outlined an interval containing both sounds and silences, only features of the sounds are included in the calculation of the features statistics. If we now change the segmentation criteria, to include silences as shown below, the feature statistics would change:



We can now test how features of the introductory notes might differ than those of the song motifs that the bird sings immediately after them:

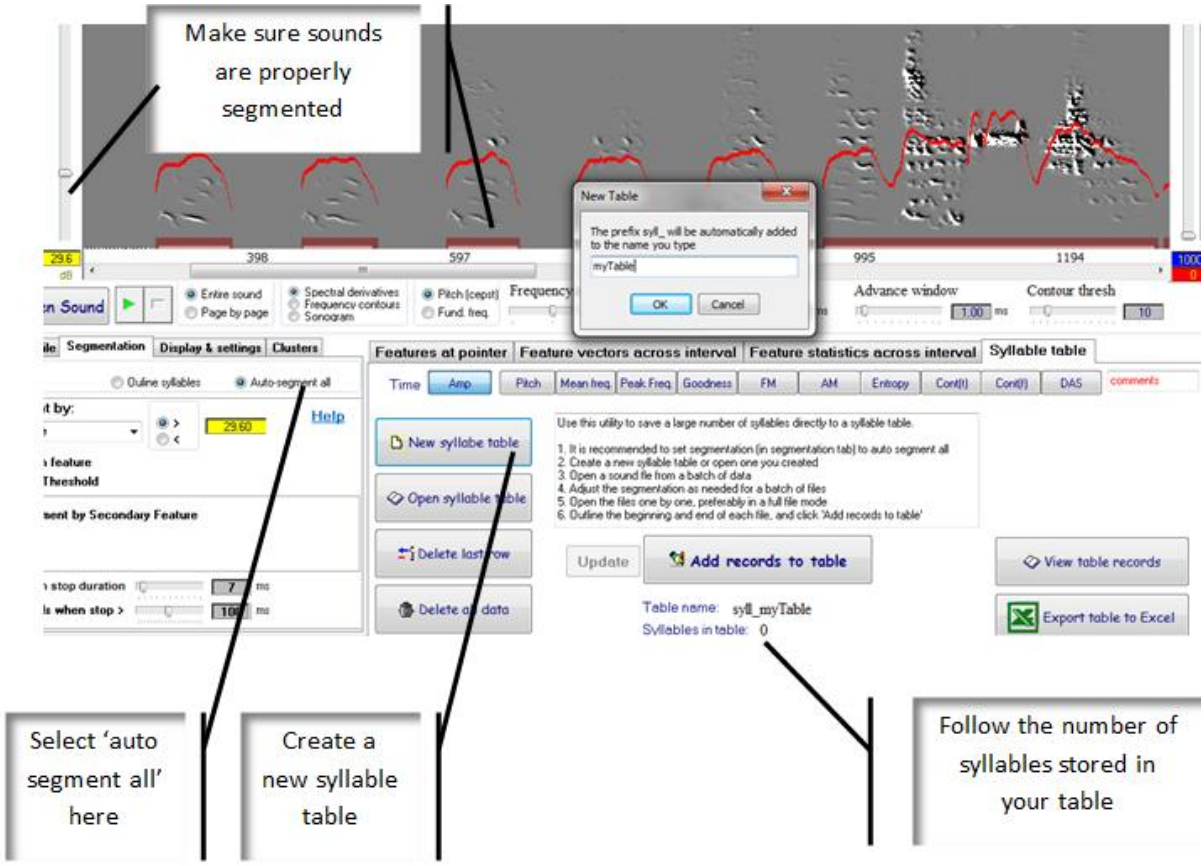


As shown, mean feature values now look very different, with higher amplitude, higher pitch, lower Wiener entropy, etc.

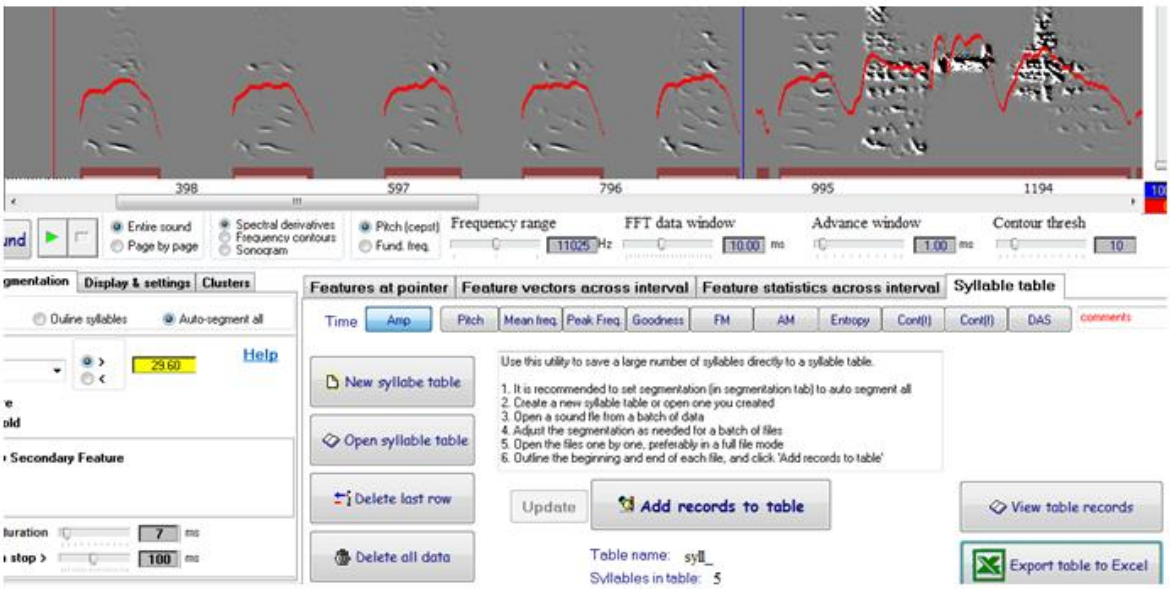
Syllable features

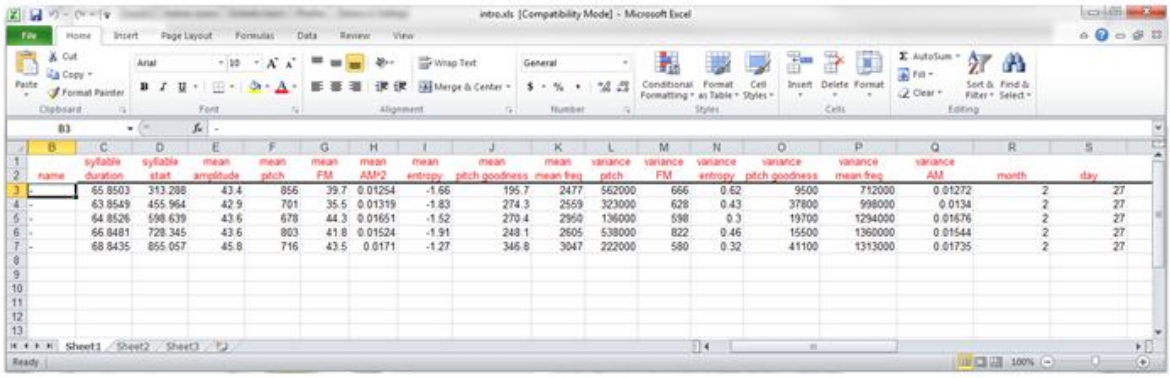
In the previous section we showed how to obtain simple statistics (mean, var, etc) from any arbitrary interval of vocal sounds, e.g., to compare features of introductory notes segment versus the features of song motifs of the same vocalization bout. A specific case of such intervals is the syllable or call, defined as a continuous sound. Because sound analysis at the syllable level is among the most powerful and frequently used approaches, SAP2011 provides several levels of automation, starting from automatic segmentation, and ending with feature batch that is used to compute syllable features over huge quantity of syllables (millions) fully automatically. However, using automated methods is dangerous and might introduce horrible artifacts. To help you avoiding it, SAP2011 uses graphical displays and other visual validations. We strongly recommend that you start extracting syllable features manually in Explore & Score, and once you are confident about the segmentation and feature calculation approach, consider the batch.

The first step is, of course, the segmentation: please go over the different approach in the segmentation section of this manual, and confirm by visual inspection that syllable units are properly segmented in your file. Then follow the procedure below in Explore & Score for several files, and only then try the batch:



Now outline a few syllables as shown below, and then click the 'Add records to table' button. You should see the syllable count increasing. Then click 'Export to Excel', and name the file. It should open in Excel as shown below:


398
597
796
995
1194
Entire sound
Page by page
Spectral derivatives
Frequency contours
Sonogram
Pitch (cepst)
Fund. freq.
Frequency range
11025 Hz
FFT data window
10.00 ms
Advance window
1.00 ms
Contour thresh
10
Display & settings
Clusters
Outline syllables
Auto-segment all
Help
29.60
Secondary Feature
7 ms
100 ms
Features at pointer
Feature vectors across interval
Feature statistics across interval
Syllable table
Time
Amp
Pitch
Mean freq
Peak Freq
Goodness
FM
AM
Entropy
Cont(t)
Cont(f)
DAS
comments
Use this utility to save a large number of syllables directly to a syllable table.
1. It is recommended to set segmentation (in segmentation tab) to auto segment all
2. Create a new syllable table or open one you created
3. Open a sound file from a batch of data
4. Adjust the segmentation as needed for a batch of files
5. Open the files one by one, preferably in a full file mode
6. Outline the beginning and end of each file, and click 'Add records to table'
New syllabe table
Open syllable table
Delete last row
Delete all data
Update
Add records to table
View table records
Export table to Excel
Table name: syll_
Syllables in table: 5


intro.xls [Compatibility Mode] - Microsoft Excel
name
syllable duration
syllable start
mean amplitude
mean pitch
mean FM
mean AM^2
mean entropy
mean pitch goodness
mean freq
variance pitch
variance FM
variance entropy
variance pitch goodness
variance mean freq
variance AM
month
day
- 65.8503 313.288 43.4 856 39.7 0.01254 -1.66 195.7 2477 562000 666 0.62 9500 712000 0.01272 2 27
- 63.8549 455.964 42.9 701 35.5 0.01319 -1.83 274.3 2559 323000 628 0.43 37800 998000 0.0134 2 27
- 64.8526 598.639 43.6 678 44.3 0.01651 -1.52 270.4 2950 136000 598 0.3 19700 1294000 0.01676 2 27
- 66.8481 728.345 43.6 803 41.8 0.01524 -1.91 248.1 2605 538000 822 0.46 15500 1360000 0.01544 2 27
- 68.8435 855.057 45.8 716 43.5 0.0171 -1.27 346.8 3047 222000 580 0.32 41100 1313000 0.01735 2 27

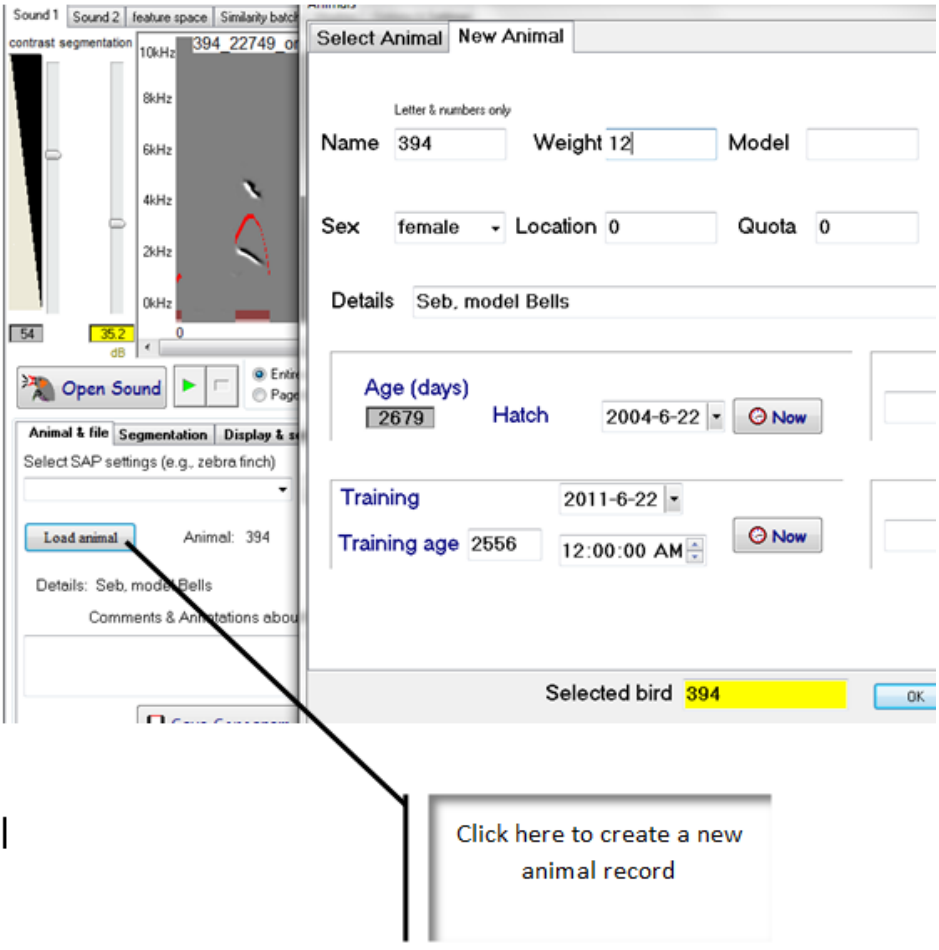
Saving animal & file information

In the Explore & Score module of SAP2011, we introduced several mechanisms for displaying and saving data, including saving the sonogram image, saving 'time events' (a certain moment in the sonogram), saving feature vectors (time course of features over an interval), and saving syllable features:

Saving sonogram images: in Sound1 or Sound2, place the cursor on the sonogram image and right click. Select 'Save this image' and name it. The image (including any visible curves) will be saved as a bitmap.

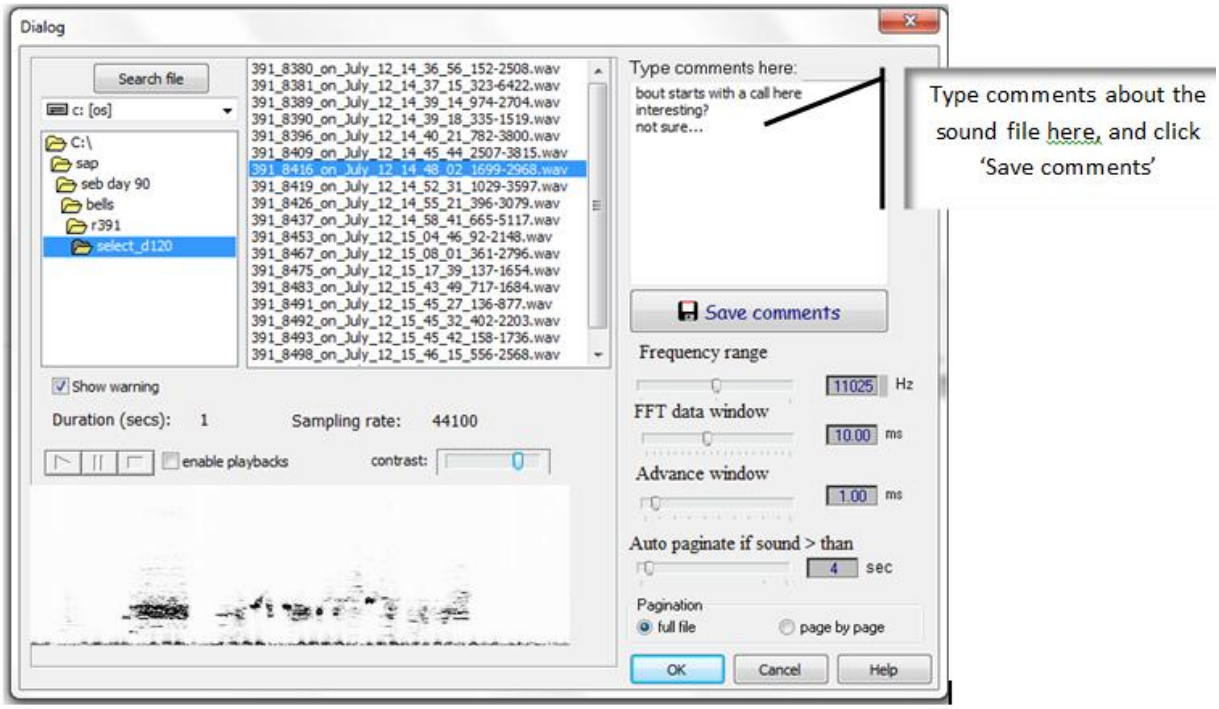
Metadata about the file & animal: Each time you open a sound file, SAP2011 can automatically retrieve information about the animal and about the specific sound file. It also automatically saves information you stored about time events.

Information about your animal: If you use files generated by Sound Analysis Pro, the first part of the file name is the animal name. SAP2011 will use this information to automatically retrieve information about that animal every time you open a file from that animal. This will include information including the experiment and also about the age of the animal when recording was done (this is achieved by comparing the file age to the animal age). Of course, you need to provide the information about the animal once. To do this, click 'load animal' in the Animal & file tab as shown below.

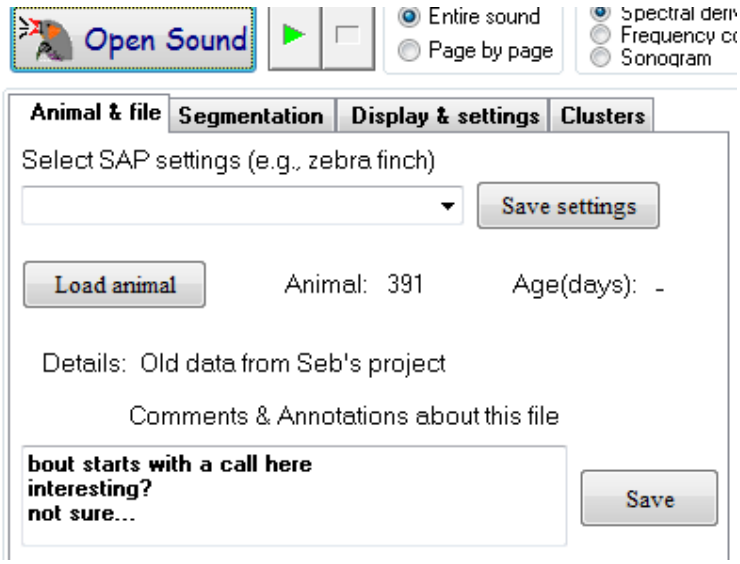


Select 'new animal' and add the information (it's all optional, but the more the better). From now and on, every time you open a file of this animal, you will see the information you inserted in the Animal & file tab.

Information about the file: There are two equivalent methods provided for inserting and retrieving information about the specific file: one is in the 'open sound' window:



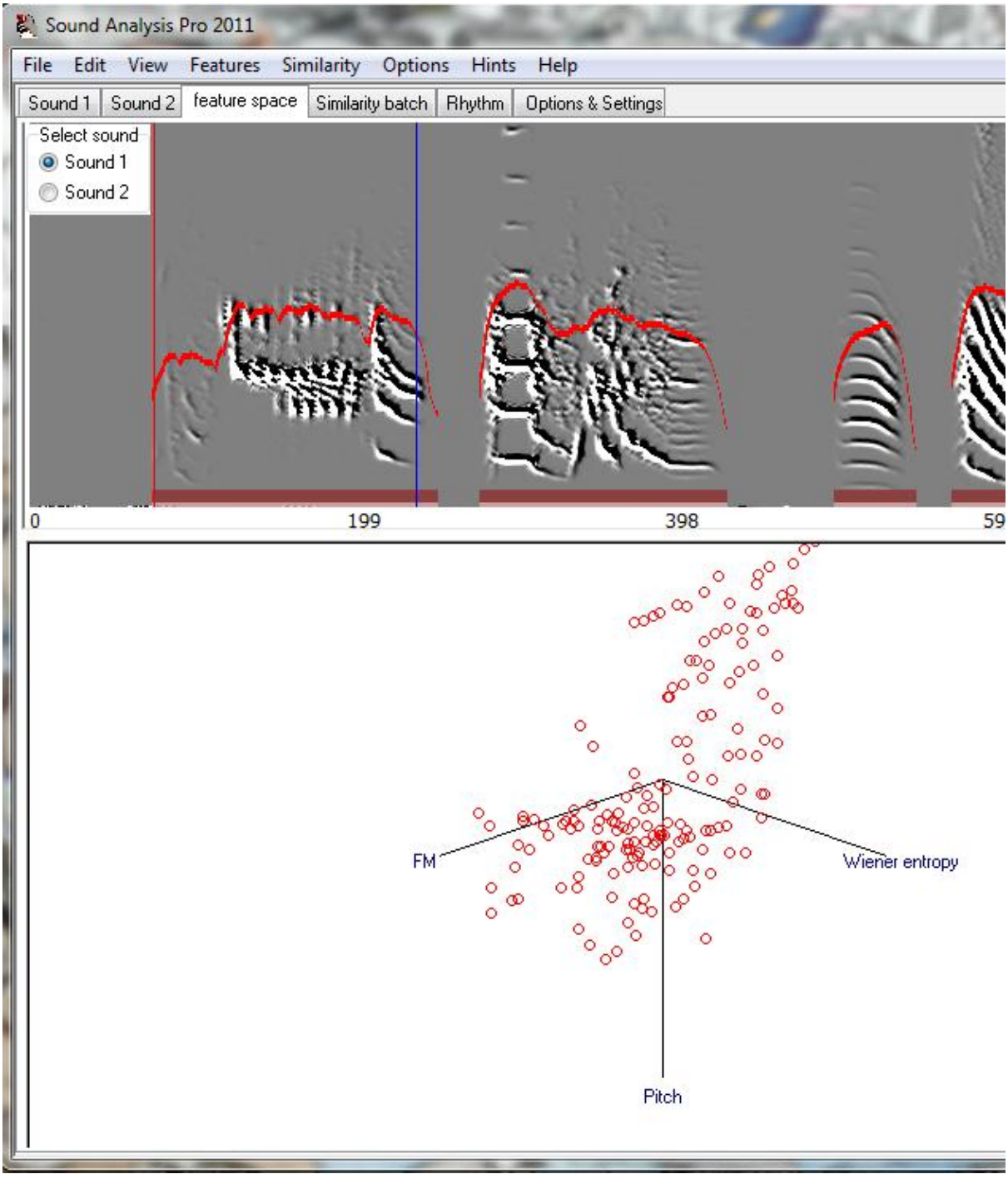
As you browse from one sound to the next, feel free to type comments on the right as shown. Next time you will browse those comments will show up and will remind you which file to open. The same comments will then appear in the main Explore & Score window:



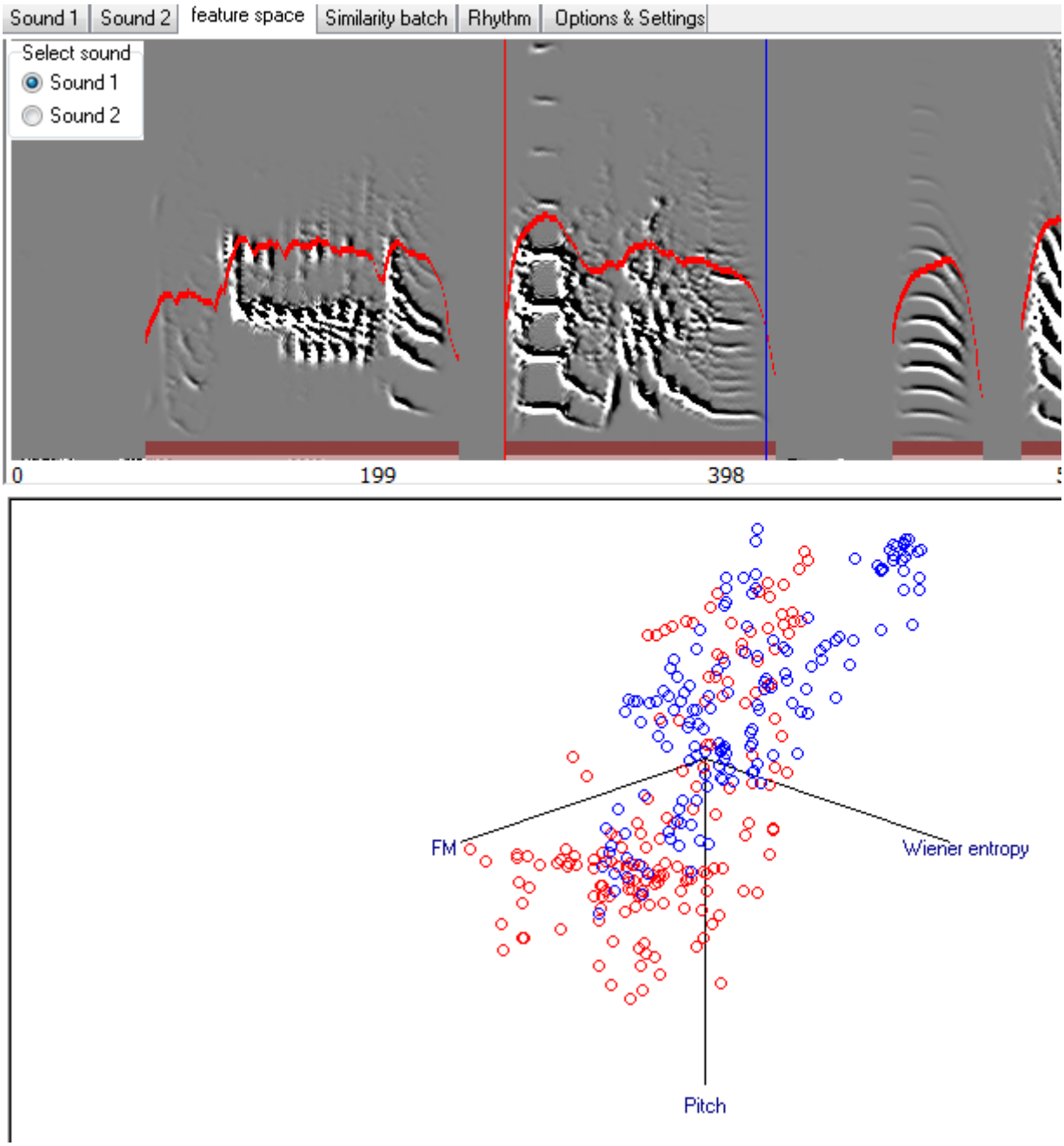
And here too, you may insert and edit them as needed. SAP2011 does not remember the history of comments. A new comment will overwrite the previous one.

Feature space

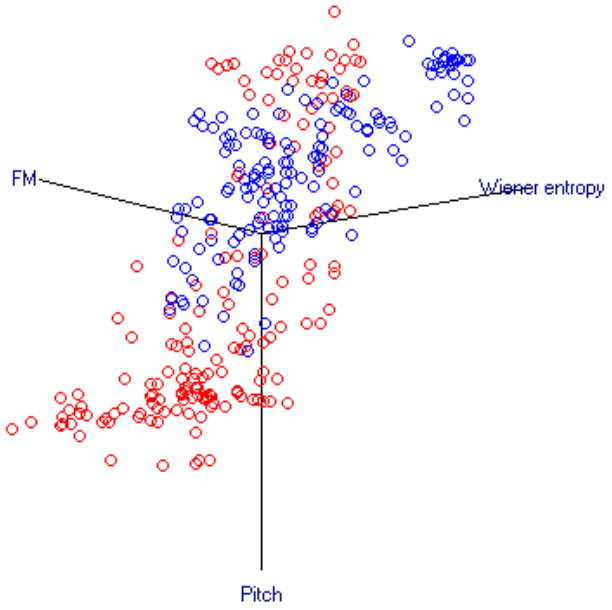
Feature space is a simple graphical method for viewing the distribution of features within and across sounds. As simple as it is, it can be quite useful. For example, you can compare features between two call types, and also judge if over time there is any directional changes in sound features. Here is a little demonstration of how it can be used: open SAP2011 Explore & Score and open a sound file that include several syllables. Let's start with the sound Example 2. Click on the 'feature space' tab, outline the first syllable and select 'draw' in the popup menu:



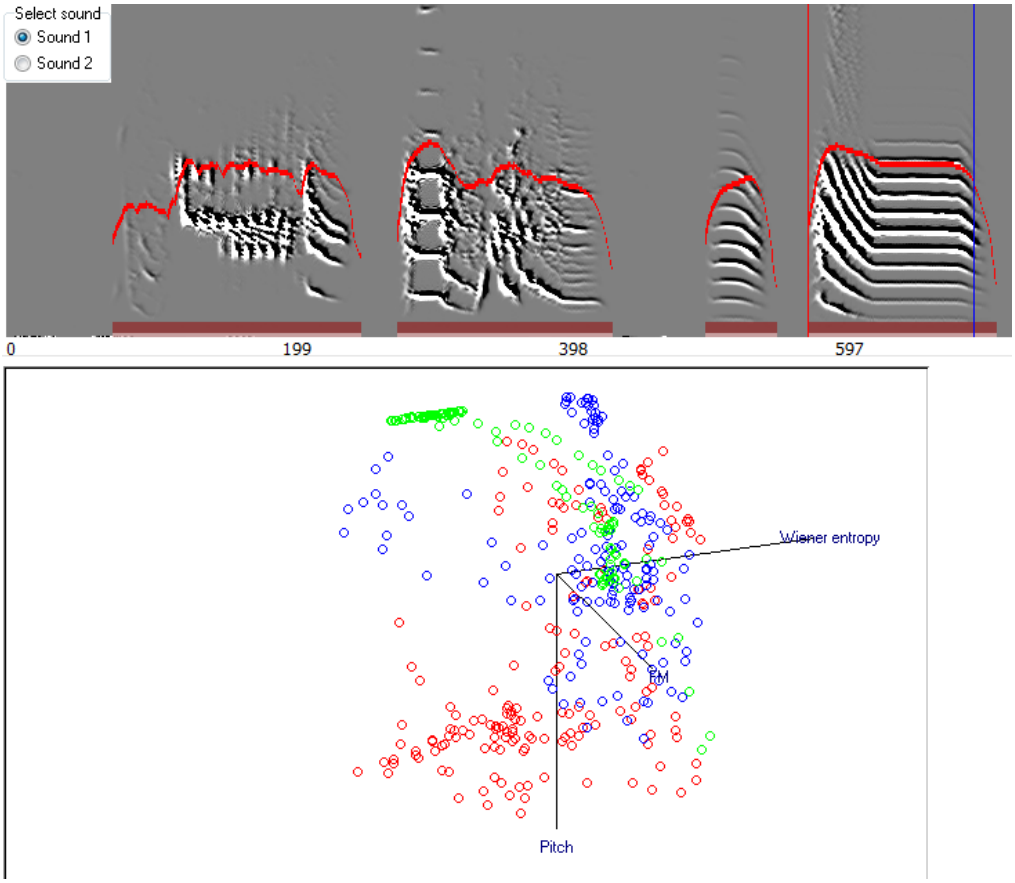
Next, outline the second syllable, but this time select 'pick color & draw'. Select blue, and the outcome should look like this:



Now rotate the image and pan it with the mouse to try to obtain a better separation of the colors:

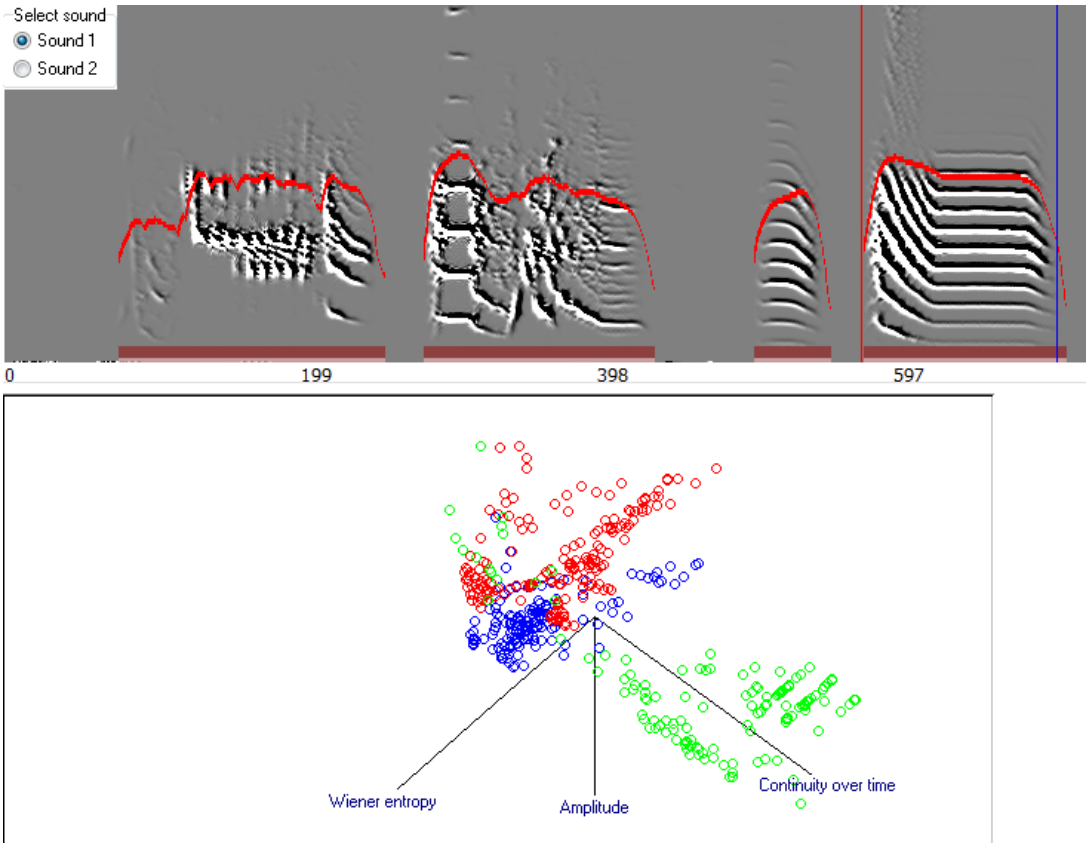


Clearly, the two syllables differ in pitch and in Wiener entropy, but less different in FM. Now, outline syllable syllable 3, select Green and Draw:



What is this exercise good for? Well, it is a way of examining the emergence of categories of sounds in a continuous space. For example, if you suspect that '*Dee'* notes of two populations

of chickadees are distinct – you may examine the data in feature space. Outline each note type and use a color code to distinguish between the populations. Select several examples from each population and see how it looks like before you get yourself committed into heavy quantitative analysis.
Last trick: clear the image and use different features, and outline the sounds again, e.g.:



Chapter 6 Content:

MySQL in a nutshell

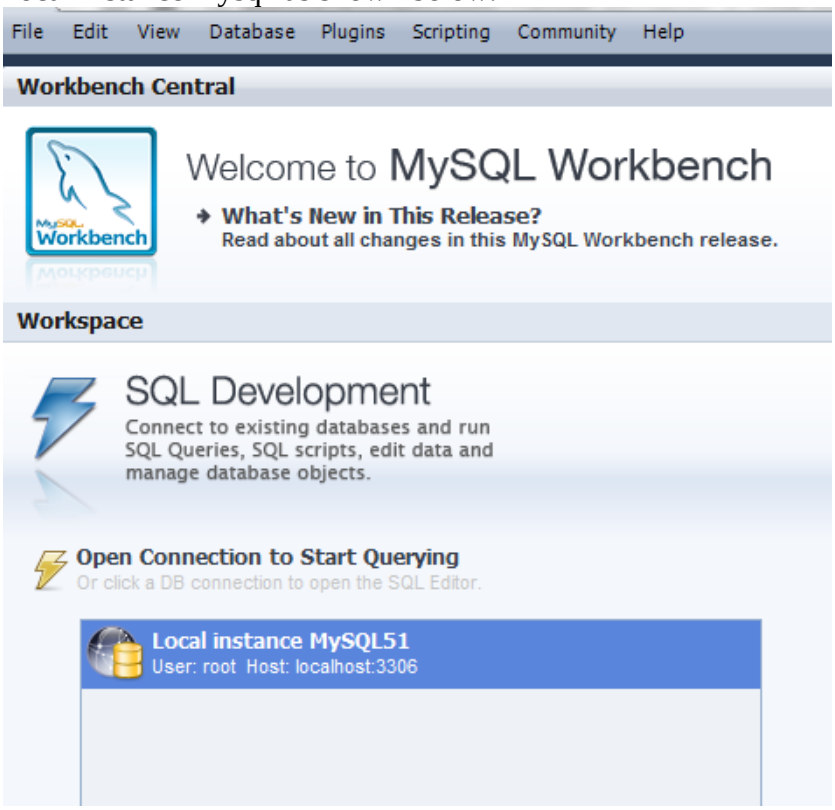
Why should you learn some Standard Query Language (SQL)
Standard Query Language (SQL) will allow you to retrieve and manipulate SAP2 data easily, and very quickly. MySQL is one of the most standard implementation of SQL and it is (practically) free. Since MySQL is used heavily in SAP2, it is not a bad idea to know just enough SQL to be able to manipulate the tables and export data efficiently to other applications. This is particularly important because the SAP2 tables often contain large quantity (even millions) of records. There are very few applications that can efficiently manage large amount of automatically generated data. Therefore, you should look at SQL as an efficient data pump between the tables and other applications. For example, Matlab does not have any (respectable) database functionality, and it stores all variables in memory. It is therefore inefficient to import an entire table to Matlab, but you can use SQL to pump data epochs into Matlab, calculate something and move to the next epoch. Furthermore, those data epochs can be very flexible in content. For example, obtaining epochs containing the pitch of the first 100 syllables produced at around 9AM is a very simple command:

select pitch from my_table where hour=9 limit 100;

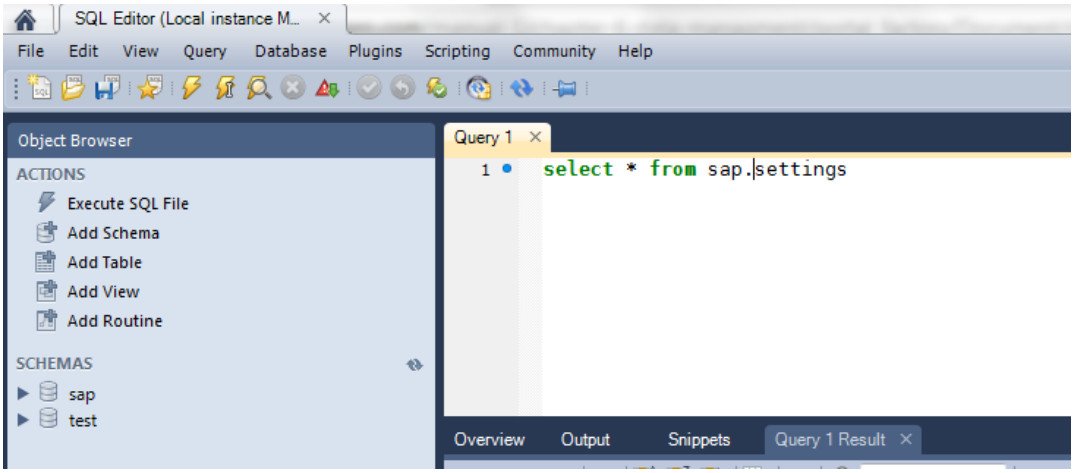
as you can see, it is very simple and intuitive to write SQL queries.

SQL is used not only to retrieve but also to process data from your tables, but here we will only teach you how to select data from a table, how to copy data into a new table, how to change table values, and how to merge tables together.
How to practice this tutorial in MySQL Workbench (WB):

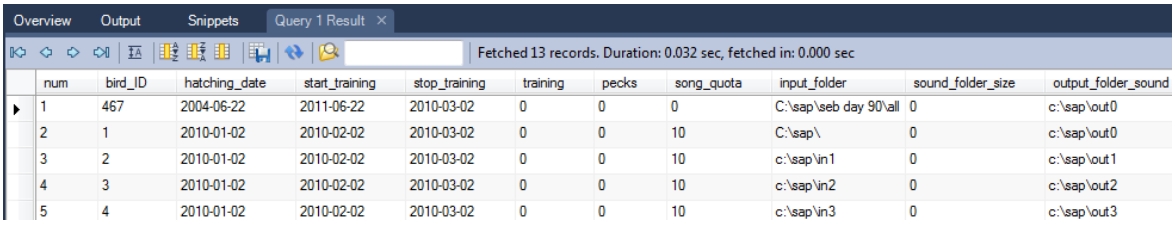
You can download the workbench here. Once installed open the workbench and double click 'local instance mysql' as shown below.



Then type a query:



and you should see the results (or error message) below:



num	bird_ID	hatching_date	start_training	stop_training	training	pecks	song_quota	input_folder	sound_folder_size	output_folder_sound
1	467	2004-06-22	2011-06-22	2010-03-02	0	0	0	C:\sap\seb day 90\all	0	c:\sap\out0
2	1	2010-01-02	2010-02-02	2010-03-02	0	0	10	C:\sap\	0	c:\sap\out0
3	2	2010-01-02	2010-02-02	2010-03-02	0	0	10	c:\sap\in1	0	c:\sap\out1
4	3	2010-01-02	2010-02-02	2010-03-02	0	0	10	c:\sap\in2	0	c:\sap\out2
5	4	2010-01-02	2010-02-02	2010-03-02	0	0	10	c:\sap\in3	0	c:\sap\out3

To use Matlab: download the mySQL Matlab connector here. Then open Matlab and type:
>>
mysql('open', 'localhost', 'root', 'sap2011');
Connecting to host=localhost user=root password=sap2011
Uptime: 745762 Threads: 1 Questions: 17 Slow queries: 0 Opens: 42 Flush tables: 1 Open tables: 0 Queries per second avg:
>>
mysql('use sap');
Current database is "sap"
and then type a query , e.g.,
>>
x=mysql('select bird_ID from sap.settings');

We will start by creating a very small Syllable Table. Open SAP2 'explore & score' and click 'New table'. Call the table table1. Now set the amplitude (yellow slider) threshold to 42Db. Next, click 'open sound' and open the sound 'bird102_long.wav' (you may download this file from http://). Select “graphic” and then outline the entire sound (click to the left of the first syllable and then move the scroll bar below all the way to the right before clicking at the end of the last syllable). You should now see 45 records (syllables) in the table.

The SELECT keyword
Once you understand how to use SELECT keyword, you can already achieve a lot. Open WB (mySQL workbench) and navigate to sap database, and then click the SQL button or open Matlab (after installing the mySQL interface), open sap as shown above and then type
WB: **select duration, mean_pitch from table1;** and click **!**
Matlab: **[duration pitch]=mysql('select duration, mean_pitch from table1');**

If you use WB select the entire output table, click Control-C (to copy the fields to the clipboard) and then open Excel and type Control-V to paste. Click data->Text to column and choose delaminated and then 'next' and in 'other' type '|' And then click finish. You can now create a X-Y plot of duration versus pitch.

Although the sample size is really small you can already see that some of the data are clustered. Let's query to obtain data that belong to one cluster, for example we can ask for data where duration is between 100-150ms and pitch is between 600-1100Hz:

WB: **select duration, mean_pitch from table1 where duration>100 and duration<150 and mean_pitch>600 and mean_pitch<1100;**
Matlab: **[c1_duration, c1_pitch]=mysql('select duration, mean_pitch from table1 where duration>100 and duration<150 and mean_pitch>600 and mean_pitch<1100;');**

Advance usage of the SELECT keyword
Table1 includes only three song bouts produced by a bird named b109. In our database you will find () a table called b109clustered, which contains the entire song development of this bird. In this table you may not want to run the query shown above, because this query would return hundreds of thousands of records.
In this case, you should always using the keyword **limit**, e.g., (Matlab):

[c1_duration, c1_pitch]=mysql('select duration, mean_pitch from b109clustered where duration>100 and duration<150 and mean_pitch>600 and mean_pitch<1100 limit 5000;');

will execute promptly and will return the first 5000 syllables that match your criteria. Because this table is already "clustered", namely, the syllable types are already identified (see chapter xx) you could have simplified this query since this cluster is already identified as cluster number 4, so this query can turn to

select duration, mean_pitch from b109clustered where cluster=4 limit 5000;

Now, this query will return the first 5000 syllables of type 4, but what if you want not the first 5000 syllables by a random sample of 5000 syllables? That's easy:

select duration, mean_pitch from b109clustered where cluster=4 order by rand() limit 5000;

What if you only want to count how many syllables of each count exists in the table?
select count(*) from b109clustered where cluster=4;

will return 173581, so you can now repeat this query for clusters 0 (not clustered), 1 (introductory notes) and 3, 4 (song syllable types of this bird) and tell that in this bird the frequency of types over development is:

0: 279354
1: 337884
3: 198997
4: 173581

We can now see if those types are present in similar proportion throughout development. To do so, we will limit our queries to different dates.

select count(*) from b109clustered where month=8 and day<19 and cluster=4;

will return 0, but

select count(*) from b109clustered where month=8 and day=20 and cluster=4;

will return 858, telling us that (using our clustering method) this cluster was not identified before August 19.

It should be now quite easy for you to realize how to select data by any combination of criteria. Using a simple for loop, you can pump the results of these queries into Matlab functions. When properly used, the MySQL server is a very potent device, capable of amazing performance returning complicated queries in a remarkable speed.

SELECT and CREATE tables

It is sometimes useful to create new tables from subsets (queries) of an existing table (or even by consolidating data from several tables). We will cover this subject briefly here:

The simplest approach is demonstrated in this query:

create table b109cluster4 select * from b109clustered where cluster=4 limit 9999999;

creates a new table, called b109cluster4, that includes only records of syllable type 4. Note that if you do not use the **limit** keyword, mySQL will create a table using some default limit (usually 1000 records) so we advise that you **always use limit**.

Now, a slightly more sophisticated case is when you need to create a new table from certain fields, combining data from a few different tables. For example, say that you have a raw feature table that is a bit faulty (my_faulty_table) and you suspects that some of the records in that table has Wiener entropy values which are positive (Wiener entropy must be negative!). Now detecting those records is easy:

SELECT entropy FROM my_faulty_table where entropy>0;

Of course, you can now create a new table excluding the faulty records:

create table my_right_table SELECT * FROM my_faulty_table where entropy<0 limit 999999;

However, you also want to identify the wave file that those faulty data belongs to, so as to figure out what happened. The problem is that the name of the wave file is not included in the raw features table. In that table you only have the file_index field. This field, however, is pointing at the appropriate wave file name in another linked table, called file_table. This data structure saves much space since we have thousands of records from each file in the raw features table. So we now need to join information across those related table, and mySQL makes it very easy. All you need to do is identify the table where the field comes from by the syntax table_name.field_name.

For example, the query

SELECT entropy, file_table.file_name FROM my_faulty_table, file_table where entropy>0 and my_faulty_table.file_index=file_table.file_index limit 99999;

returns the positive entropy values side by side with the file names that relate to those values.

The simple examples above demonstrated how a new table that include only some of the table records (row) and fields (columns) can be generated from an already existing table using the **Select** command combined with other commands such as **create table**.

Altering, cleaning and merging tables

Mistakes happens, and tables need to be altered or updated occasionally. You can alter data manually in the MCC, but as your tables become larger, it becomes impractical to correct values by hand. The simplest way of altering a table is to apply a change to an entire field, e.g.,

Update my_syllables set cluster=0;

is going to erase all your cluster values, setting them to zero. So, if you actually like most of your clusters, but not cluster 4, you could type:

Update my_syllables set cluster=0 where cluster=4;

In short, you can alter subset of your data using the exact same method for selecting subset of your data.

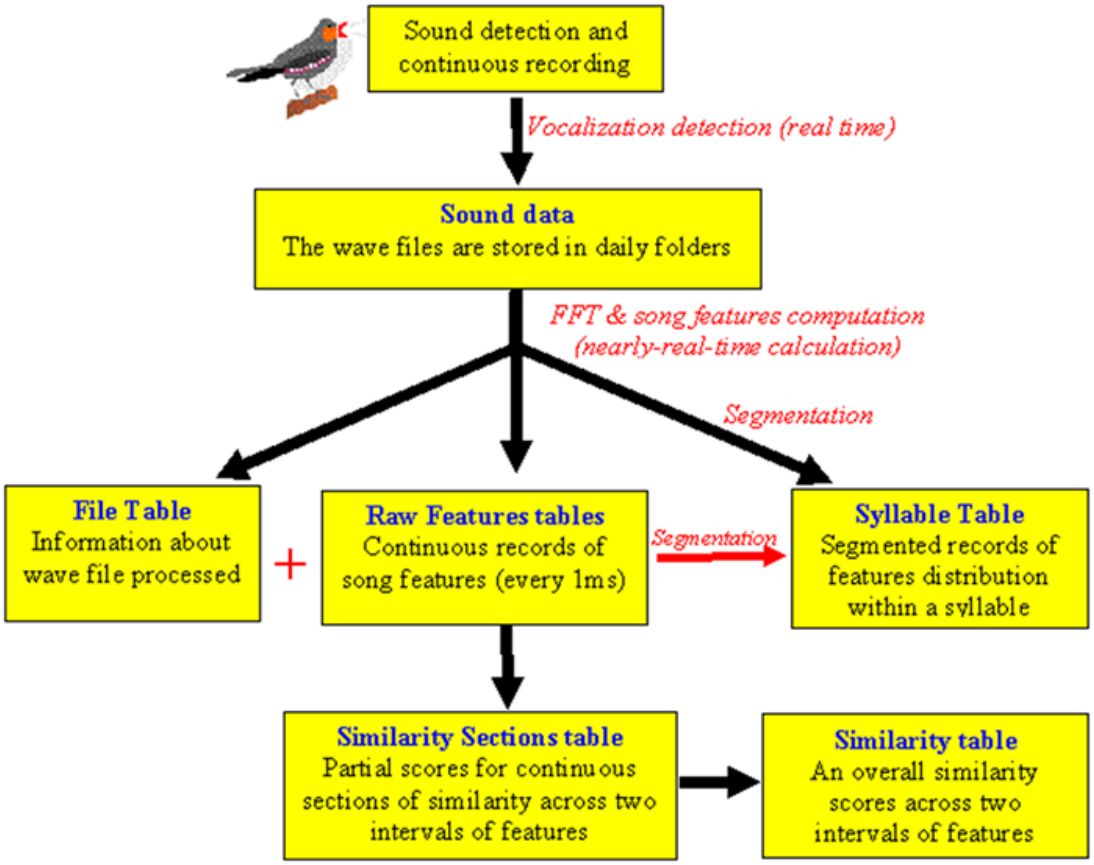
Eliminating duplicates:

Finally, it is quite easy to merge two tables into a single, large table:

Insert into my_first_table select * from my_second_table;

The SAP2011 data structure

Here is a brief introduction to the SAP2011 database structure
How analysis results are stored in the database: SAP2011 automatically creates wave files and analyzes them. The results are saved in a database called SAP2011, where each table summarizes data of different types. Here is a brief description of the data tables according to their types:



The Sound Analysis Recorder detects vocalization and saves the sound data in wave files. Wave files are stored in daily folders. During recording, in nearly real time, the Sound Processing Live engine performs multi taper spectral analysis, and then computes the song feature. The features, usually computed in steps of 1ms, are stored in Raw Features tables. Note that these records are continuous, with no distinction between "silences" and "sounds". By default, SAP2011 creates a single Raw Feature table per day, per bird. It is often a large table, including up to hundreds of millions of records. As the data from the (many) wave files flows into the Raw Features table, SAP2011 updates a small File Table, which contains indexes and file names of each processed wave file. Indexes for those wave files are saved in each of the Raw feature table records.The Raw Feature Table, together with the File Table, can be used to generate a Syllable Table. SAP2011 automatically create one Syllable Table that includes records of the entire song syllables the bird produced during an entire experiment.

The transition from Raw Feature Tables to the Syllable table is computationally cheap. However, segmentation of sound data is a difficult task, which you might want to revisit. This

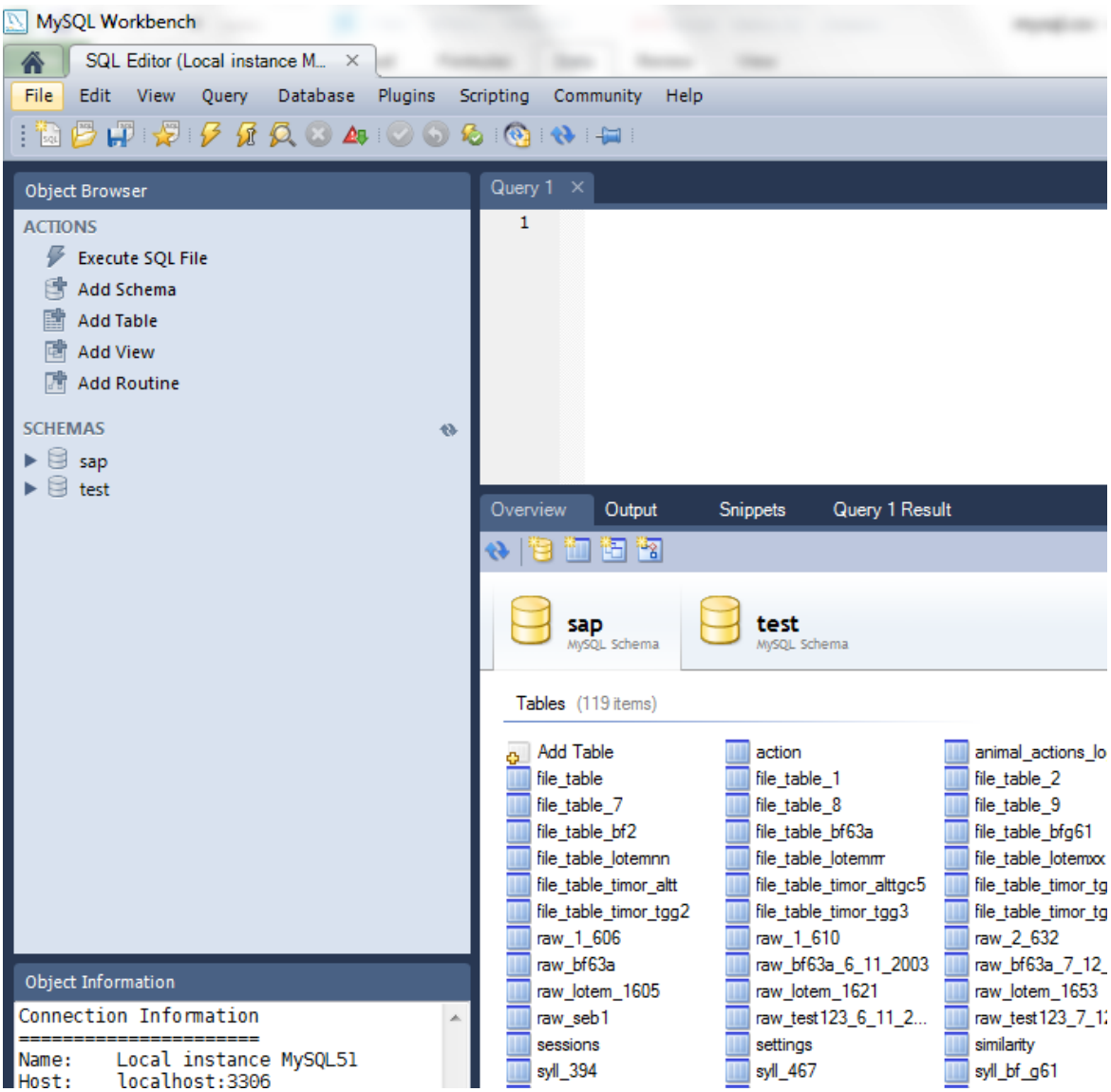
is one reason why it is useful to have the Raw Feature Tables - it is an analyzed version of your raw data that is still very raw. A second reason to have the Raw Feature Tables is that it is something meaningful to analyze sound data "as is" without any segmentation. Such analysis is particularly useful for so called "graded" signal, where segmentation is not necessarily meaningful and might induce artifacts. It is easy to compare raw features using the KS statistic. The more complicated Similarity Measurements of SAP2011 are also based on raw features.

In the previous chapter we presented examples of how SAP2011 data can be exported easily to MS Excel. This method, however, is very limited in its capacities and flexibilities. In order to take full advantage of SAP2011 you will need to understand some of its database structure and you will need to know how to access raw and processed data using using MySQL and Matlab. If you do not use Matlab it is recommended that you take a look at this chapter, and learn something about how to use MySQL to visualize and manipulate the data. If you do use Matlab, this chapter is very important, and also, we strongly recommend that you will learn just a little SQL (we will help you), which will help you to swiftly retrieve the appropriate data into matlab variables. In this chapter we present the database structure in a simplified manner. We then describe the structure of the feature tables, and show how those tables can be accessed in Matlab. SAP2011 raw data are saved as wave files, whereas all the derived data (features, similarity measures, clusters) are saved in a MySQL database tables.

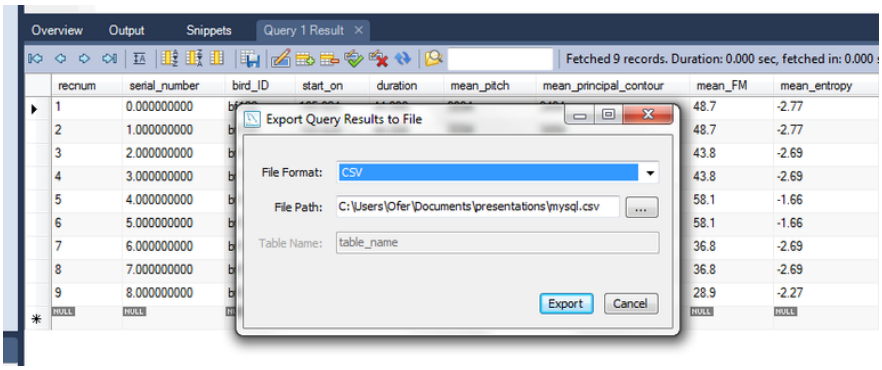
SAP2 automatically generates tables of several different types, but you only need to know about some of them. Here is a simplified flowchart of what happens to sound data in SAP2011:
The SAP2011 recorder detects and save wave files into daily folders. Each daily folder might contain thousands of wave files. As wave files are recorded, "sound processing live" engine.

Exploring SAP2011 tables

SAP2011 can auto-generate all the tables if they are not found in the database.
You can access the tables via Matlab or via Excel, but the native table-browser is the mySQL WorkBench, which you can download from the mySQL website.
Once you set up the workbench, click the local instance to open this window:



Then double-click on a table you want to view or export, e.g., try recs0:



and use the save button to export the table (or a query from the table) into a CSV file, that you can then open in Excel or in Matlab.
If you do not like the workbench, an alternative, free product is mySQL query browser. When opening Workbench for the first time you will need to start a new session (just give it a name keeping all the default settings) and then you can "connect" and observe the databases. Double click the SAP database to observe the tables:
The Table Structure

An example of table structure. Here is a summary of the features, as they appear in in a raw features table:

feature	description
time	time elapsed since midnight in 0.1ms units
file_index	an index to the wave file (lined to the birds file table
amplitude	the momentary sound amplitude in Db
mean_frequency_amp	the amplitude at the 500Hz range around mean frequency
pitch	the momentary pitch
mean_frequency	the center of gravity of the power spectrum
FM	frequency modulation (0-90 degrees)
am	amplutude modulation
goodness	the goodness of pitch (cepstrum peak value)
entropy	Wiener entropy
peak_frequency	The frequency of highest power
DAS	Duration of acoustic state (ms)
continuity_t	continuity of contours over time
continuity_f	continuity of contours over frequency

SAP2011 can perform several different tasks and our database structure is shared across those tasks.

Parameter settings & feature scale

The Settings Table

This is an automatically generated input/output table that in most cases you will not need to access directly, but when encountering "strange behavior" of SAP2011 it might become useful to know something about it. The Settings replaces the "birds" table used in previous versions of SAP. There is only one "settings" table. As shown below, this table has a major role in SAP2011 "remembering" its previous state upon restart. However, not all the states are captured in Settings - the state of the recording channels (of the recorder) are stored in a table called Channels, and the settings of keys and other digital devices (if present) are stored in the NIDAQ table.

Settings table is 12 records: the first is used to store the settings of single analysis modules such as the Explore & Score and Feature Batch. The other 11 records are used to store multiple analysis modules as in Sound Live.

The fields (entry types) and only 12 records (one record per recording channel). The first record is used as the default state for all the modules except for the live modules (you may think about this as 'channel 0'). The other 11 records are used for the real recording and processing "channels" of the recorder and of the live processing modules. Settings table communicates with all the SAP2011 modules and maintaining birds IDs, ages, recording and segmentation parameters, playbacks, etc.

The Feature_Scale Table

The feature_scale table is an automatically generated table that contains one or several schema for scaling features. Features are scaled for several purposes, including similarity measurements, based their distribution. We are interested in two types of distributions: distribution of raw features (pitch, FM…) and the distribution of syllable features (mean_pitch, mean_FM, var_pitch, etc). To scale features we need to know the central tendency and width of the distributions in a representative sample of data (e.g., within the songs of 100 zebra finches). We use the median and the median absolute deviation (MAD) of each feature to characterize those distributions. SAP2011 comes with a zebra finch settings and users can easily add settings for other species, song types, etc.

Animals table: handling animals' information

How SAP2011 handles Animal Information
Until now we looked at derivatives and features without identifying the animals that the data belong to. Once you insert animal information into SAP2011, such as animal name, date of birth, training method, etc, this information is automatically retrieved each time you access data from that animal. This automatic retrieval has different implementations in different modules, but the example below will clarify the principle and the advantages of this mechanism. Overall, we recommend that each animal you analyze be identified, as shown below.

Let us start by looking at some song development data of a bird called R109. When SAP2011 recorded this bird, the user already inserted the name of the bird, but your new SAP2011 does not have this information. To insert a new bird using the Explore & Score module, click the "New/Change bird" button. You will see an "animals" window that allows you to either select an animal or to insert a new animal.

Click the "New Animal" tab and fill in the fields as shown below:
From now and on, each time you record from this bird, and each time you open a wave file of this bird, and each time you access any kind of data derived from this bird (e.g., syllable table), SAP2 will use the information you just inserted to make more information available to you.

For example, each time you record from this bird, SAP2011 will generate wave files that start with the bird's name, e.g.,:

R109_39152.86207609_3_11_23_56_47.wav

We will describe the SAP2011 files and table names annotation method in the next chapter.

Each time you open one of those files, SAP2011 will recognize the file as belonging to bird R109, and based on the file attributes that indicate when the data were generated, and based on the hatching date information, will present to you the age of the bird at the time of recording this file. Let's try it:
Click "Open Sound" and open the sound
The age 66 indicate that this bird was 66 days old when these data were recorded. Once you have several birds in the SAP2011 database it is easy to look at one, or a few of them, according to any search and filter criteria. We will start by uploading my CCNY animals database.
The first page, Animals, present a list of animals we experimented with. Use the mouse to click on different records, and scroll up and down, note that the data entry fields below are updated accordingly.

If you like to change a bird entry, or to add a new bird (perhaps based on some of those values already entered for another bird) click "Allow Add/Update". The table and fields now turned yellow, which indicates that you can now edit the data fields. If you change any field other than the "Name", and then click Add/Update Record, the information about the bird will be updated according to your changes. If you changed the "Name" field and then click Add/Update Record, SAP2011 will add a new bird at the end of the table.

Playbacks: Channels, Key-peck & NIDAQ

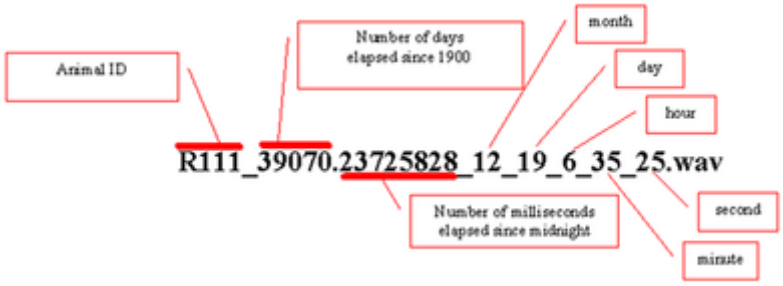
The Channels Table is an automatically generated table that contains the state of each SAP2 Recorder channel. You do not need to know much about it to use SAP2011.

The Key-peck table: Is an automatically generated table that contains information about activation of detectors in operant song training. If you are using operant playbacks, this table is used by the recorder to save events of key activation.

The NIDAQ table: This automatically generated table maintain the identity of the National Instrument detectors and effectors. If you are using operant playbacks, the configuration of the National Instruments ports and channels (for more information, see the installation and Recording Vocal Sound chapters.

Sound data: Wave files & features tables

The Wave Files
All the raw data generated by SAP2011 are saved as wave files of the following file naming format:

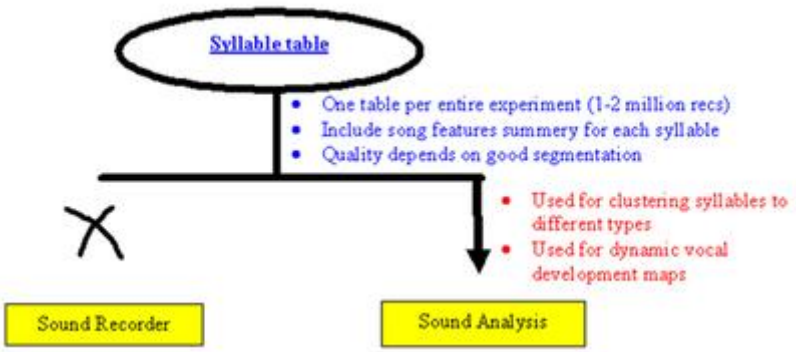


When multiple channels are recorded (in a master-slave mode), e.g., when recording electrophysiology data, the master channel (typically the sound) has the name annotation shown above, and the slave channels will have the same name as the master, but starts with Sn~ where n is the channel number (e.g., S3~109_39013.... is the third slave channel of this bird, in this case an electrode located in RA).

When SAP2011 records & processes data from a certain animal (life recording) it identifies vocalization events (e.g., a song bout) and saves it into a wave file. Over a few days of recording, the number of these files might reach tens of thousands. Files are automatically organized into daily folders. The name of each folder is the age of the bird, e.g., the file is from bird 111 when 62 days old. The age is calculated by subtracting the age of the file from the hatch date of the bird (hatching date is automatically retrieved from the Settings table).

Having many, relatively short files could be difficult to maintain, but SAP2011 provides, in addition to the folder and name annotation, an important utility called the File Table. The File table contains information about all the files recorded for a certain bird, and it can be used to query an arbitrary subset of the data - e.g., the first 100 files of the morning. This is why it is convenient to keep the SAP2 raw data in small packages.

The Syllable Table



Syllable table is perhaps the most useful table the SAP2011 generates (for example, see DVD maps). What makes the syllable table so useful is that it captures much of the dynamics of vocal changes at the syllabic level during an entire development or during an entire experiment and have it all in a single file. It allows you to cluster syllables to type, watch how

syllable types 'move' in their feature space until model match is achieved, and it provides an extensive summary of different features of different syllables, namely summarizing their structure. In fact, you already created a syllable table in the previous chapter and exported to Excel.

recnum is the global index (and the primary key) of the syllable table. SAP2 uses recnum to retrieve data epochs from the table, e.g., when displaying a DVD map. Also, the recnum tells you in what order the syllables were originally inserted into the table (and this order should match the order of serial number (the date & time 'age' stamp) of each syllable.

serial_number is the date & time stamp of each syllable, and it is identical to the age of the sound (wave) file that the syllable belongs to. As in the file name encoding, the integer number is the number of days elapsed since 1900. The fraction, however, is not the milliseconds elapsed since midnight but the fraction of the day elapsed (both numbers are given by the 'FileAge Windows function).

bird_ID is a number that identifies the bird. In Explore & Score it is zero by default, but in the Live mode, it corresponds to the bird's name. In Batch mode, user is always asked to provide a bird ID.

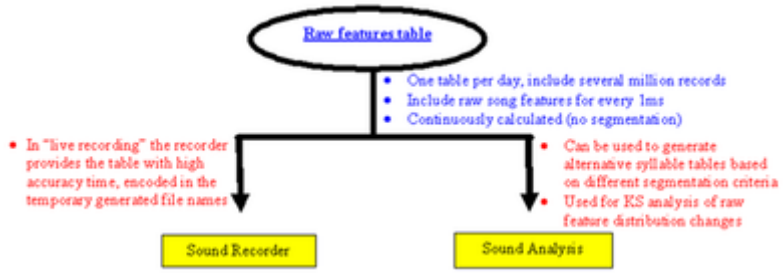
start_on is the time (in milliseconds) within the sound file where the syllable begin. It is an important feature that allows SAP2 (or you) to accurately identify the syllable in the raw data. For example, SAP2 can perform batch similarity measurements within and across syllable types, automatically retrieving sound files and outlining a specific syllable according to start_on and duration.

The features first and second order statistics fields are duration, mean features, minimum and maximum number of features, as well as the variance of features.

Date & time fields albeit redundant, it is often convenient to have date and time (month, day of month, hour, etc) represented separately in fields, so as to allow simple querying of data subsets.

file_name is the name of the sound file that the syllable belongs to.

The Raw Features Table



SAP2011 performs spectral analysis and calculate acoustic (and other) features across partially overlapping spectral (FFT) windows. By default those windows are 9.27ms long and we advance the window 1ms in each step. Namely, we start from calculating features in the first interval of the sound file (1-9.27ms), then we calculate features in the interval 2-10.27, 3-11.27, and so forth. The raw features table presents those raw features as records. Let's make

an example: open SAP2011, click “Explore & Score” and then in at the bottom right check “save raw features”. In the popup window type “raw_features1” and click Ok. Now click “open sound” and open the sound “example1.wav”. This sound is about 820ms long. Next open the MySQL Control Center (if already open, right click the database SAP and click “refresh”). The new table you just created, raw_features1, should show up in the list of tables in SAP and it should include 812 records, one for each 1ms in the file excluding edges (note that the number of records is determined by the “advance window” not by the FFT window size!). Double click this table and you should see a display like this:
Note that 'time' and 'file_index' fields have a little key above them. Those fields are the '***primary key***' of the raw feature table. ***Primary keys*** must be unique, namely the table cannot include duplicate of the same time within the same file_index (each file_index identifies a wave file, and indeed, the same time should not occur twice within a file). The primary key is an index used by MySQL to sort the table and to accelerate access to different locations in the table.

We will now go through the fields:

Time: display the time of day in units of 0.1milliseconds elapsed since midnight. For example, the first number 391,740,010 can be divided by 860,400,000 (the number of 0.1milliseconds in each day) to obtain the fraction of the day elapsed = 0.45. So in units of hours (24 x 0.45) the time is 10.8 (10:48AM). In other words, we are saying that this sound was recorded at 10:48AM - how can we tell? In this case (the default mode of the “Explore & Score” module) the time was retrieved by the Windows API utility FileAge(). FileAge() has the advantage of returning the time of when the original file was created. It remains unchanged in any copy of the original file, which is nice. There are two issues you should keep in mind regarding FileAge():

1. If your original data are not wave files (e.g., mat files) then the generated wave files will give meaningless time stamp. In such cases, the solution is to generate file names of appropriate annotation (see section 6b) and then instruct SAP2 to extract “time” from the file name.
To manipulate SAP2 method of extracting time information from wave files go to options -> input & output settings, and check the appropriate “extract time stamp” option.

2. The Windows time stamp is only about 2 seconds accurate. That is, our time estimate can be very accurate within each file, but across files we have considerable time jitter. In the SAP2011 recorder, we overcame this limitation by implementing an accurate millisecond counter. The accurate milliseconds count is then displayed in the recorder generated file names, and then the sound processing live uses this information to use raw features table of 1ms accuracy across files. Note that raw feature tables generated using the new Recorder are indistinguishable in their time format from raw features tables generated using Explore & Score or Batch - it is the user responsibility to keep track of the cross-file accuracy of the time field in any given data. For example, all the song development data generated in our lab were, unfortunately, created prior to SAP2 and therefore, our cross-file time field is of 2 second accuracy and there is nothing we can do about it.

Note that the raw features table does not contain any information the date but only the time of day. The file_index field (see below) allows you to retrieve the information if needed, however, in most cases you will not need to: When the raw features table is generated in “live recording”, SAP2 creates one raw features table in each day for each bird. The (automatically

generated) name of the daily table (e.g., b109_76) will tell you the identity of the bird and its age (bird 109 on day 76 post hatch).

File_index field points to an automatically generated File Table (see below), which provides several details about the wave file, so as to make easy to link the features at the sound data.

The features fields: In order to minimize storage space (Raw Features Tables are huge) and decrease computation time (for some features) we encoded some of the features to units that allow efficient storage and quick calculations. Here is a list of those decoding together with decoding (namely, the procedure that will transform the features back to their appropriate units):

Feature	**Original units**	**Raw Features units**	**Decoding**
Amplitude	Db	Db	None
Mean Frequency Amplitude	Db>	Db	None
Pitch	Hz	Hz	None
Mean Frequency	Hz	Hz	None
FM	Degrees (0^0-90^0)>	Degrees x 10	/10
AM	1/t	1/t x 100	/100
Goodness of pitch	None	None	None
Wiener entropy	None	x 100	/100
peak frequency	Hz	Hz	-
DAS	ms	ms	
Continuity over time	milliseconds	Milliseconds x 100	100
Continuity over frequency	Hz	Hz x 100	/100

A wave file of, say, 10s contains 441,000 samples of sound (sampling rate is 44,100Hz). Each sample is a 16 bit number. When we analyze this file and save raw features table the number of records is only 10,000 (this is the number of milliseconds of sound we analyzed). However, each record contains several numbers, and therefore keeping the number of bits per record reasonably low can save much storage space. The field types we chose, in addition to some simple encoding of units reducing the size of the raw feature tables to about one third of the raw data, as described below.

Sound metadata

Several meta data about recorded sounds might be stored by SAP2011:
The File Table: A file table is generated each time a live or a batch module create syllable tables or raw features tables. The file index then appear in those feature tables as an index to the file table, which tells you more about the identity, location and attributes of each of the sound files. As mentioned earlier, the File Table is an important utility that allows you to query arbitrary subsets of your raw data from any set of processed data. Here is the File Table structure:

Note that the file_age and the bird_age can be used to design queries for retrieving subsets of data. However, the file age is a bit less accurate than the ms table serial number time stamp. That is, the number of days elapsed from 1900 is accurate but the fraction of the day is not the same as number of milliseconds elapsed from midnight but the DOS time prompt, with accuracy of about 2 seconds. To overcome this, the recorder generates its own time code -- more about this in the next chapter.

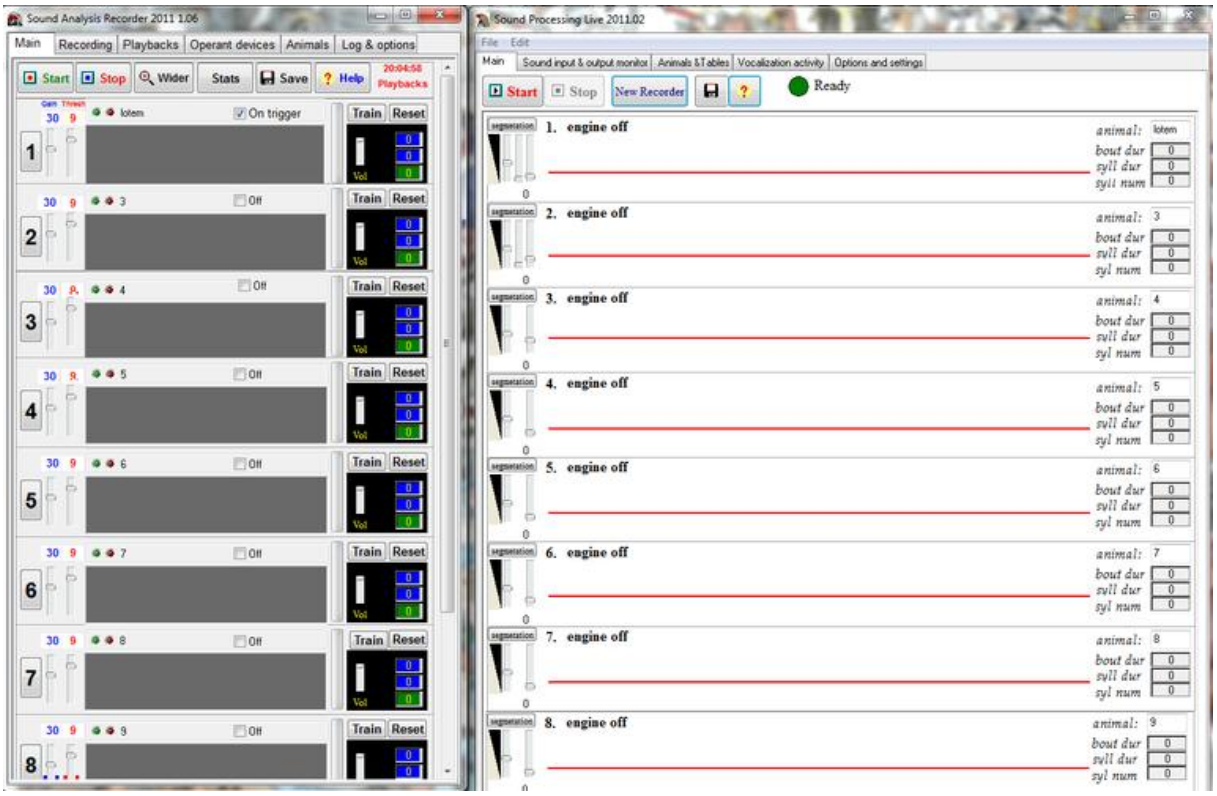
The Bouts table contains the duration of the longest bout in each recorded file. It is used in the recorder self-feedback training, so as to pick playbacks form a recent file which contains the longest song bout (so as to avoid playing back too much noise. Fields included are:
recnum: the record number (auto-increment)
bird_name
bout_duration: in milliseconds
file: include both file name and folder

Chapter 7 Contents:

Recorder overview

The Recorder Features

Start *SAP2011* and click the *Live Analysis* button. You should see a screen similar to the one below:



In this chapter we cover the Sound Analysis Recorder (left window) functions, whereas the Sound Processing Live (right window) is documented in the next chapter. Note that the recorder can be used without the Sound Processing Live if all you want to do is to record sounds into a folder, and you can use it either as a simple recorder or as a threshold activated recorder as a stand alone application. However, quite often you will also want to analyze the sounds you record, and perhaps to refine your selection about what sound files to keep and what to discard. For those purposes you can use the Sound Processing Live, which will also create tables of sound features that you can analyze.

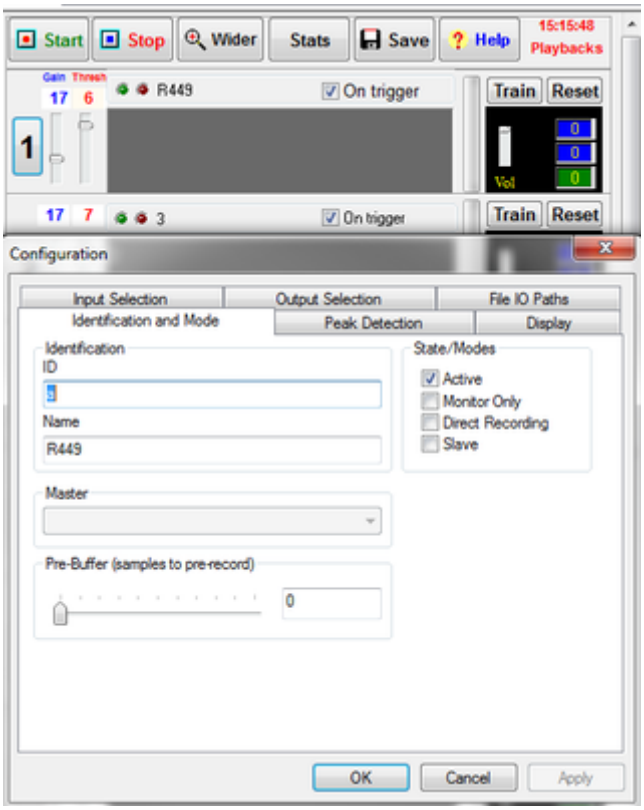
First, please familiarize yourself with the *Main* window of the recorder. It provides you with an overall view of the recorder functionality. You can record from up to 10 independent channels. The sound waves are displayed in the oscilloscope view on the left. Playbacks activity (operant of passive) can be seen on the right side of the window. If keys are used, those are displayed as circles in the training box representation. The Main tab also provides you with the most important controls, both global ones (start, stop) and channel specific (start playbacks, control the automated recording trigger of a channel).

Step by step setup

Here we demonstrate how to set one channel for basic recording. Once you get one channel to work, you can repeat it for all others and use the advance options in the other tabs. But first, get the channels you need to work using the following three steps:

Step 1 - Setting Channel Identification

Click on the channel number-button (left side) to open the configuration window of the channel:



The first tab is Identification & Mode, please go carefully through the options:

ID: this is the internal identification of the channel -- do not change it unless you want to save several different configurations for the same channel. Changing ID will reset all channel options.

Bird Name: type the bird name here (e.g., R503). Note that this name will be overwritten ever time you change animal via the the "Animals" tab.

States & Modes: each channel can record in one of the following modes: Active: The channel record sound when triggered by an appropriate input; Monitor only: The channel display the sound but does not record or trigger other channels; Direct recording: The channel behaves like a simple tape recorder (remember those devices?): it turns on by clicking start and off by clicking stop. It records files of certain duration, e.g., a new file every minute.

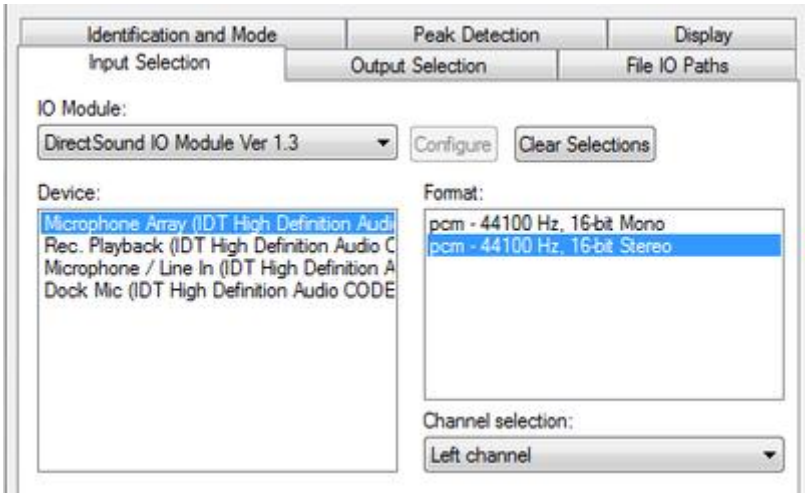
Master & Slave recording: In this mode, recording is triggered by another (master) channel. This is useful when recording combined sound and electrophysiology. Note that the sampling and accuracy quality of most sound cards is good enough for capturing single and multiunit activity. For EEG recording you must take into account that low frequencies (2Hz and lower)

are not captured by most sound cards. In order to automatically save recorded data on a slave channel, you must check 'slave' in the Sound Processing Live module of this channel. Otherwise, the content of that channel will be judged separately than that of the master. For example, if the slave channel is used to capture neural data, you want those data to be coupled to the sound so that a decision to save or delete the sound (master) channel will apply to the slave channels.

Pre-Buffer: In the Active mode recording is triggered by 'sound events' but quite often it is desired to have the recording starting a little bit prior to the trigger. This control allows you to set how many samples are recorded prior to triggering. Setting it to 44100 will record sound starting 1 second before triggering occurred (e.g., about 1s before the song begins)..

Step 2- Setting the Input & Output Channel

Select the Input Selection tab:



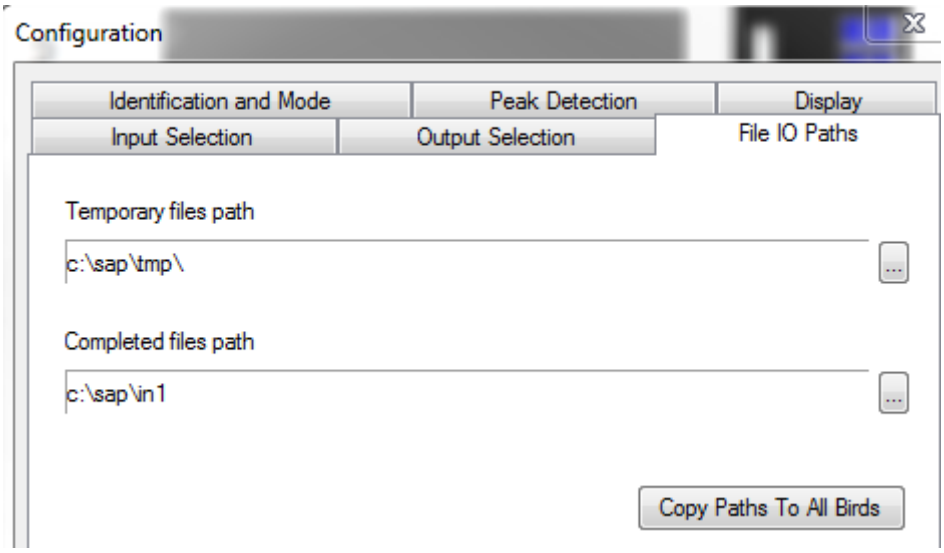
IO Module: You may select between DirectSound and ASIO driver modules. Each time you make a selection you will see the available devices (if any) that are supported by those drivers. Most simple (stereo) sound cards will appear when selecting DirectSound. Multi-channel sound cards will usually appear as both DirectSound and ASIO devices, but in most cases you will need to use the ASIO module to access individual channels.

Device: select the sound device of your choice.

Format: for DirectSound select stereo and then in Channel selection go for left or right. This will result in a mono-left or mono-right recording (do not select mono format unless you want to join the channels). In the ASIO simply select the channel you need (stereo mode does not apply).
If you want to play sounds while recording, go to the Output selection and repeat your choices. The File Selection allows you to select a sound file to play during recordings.

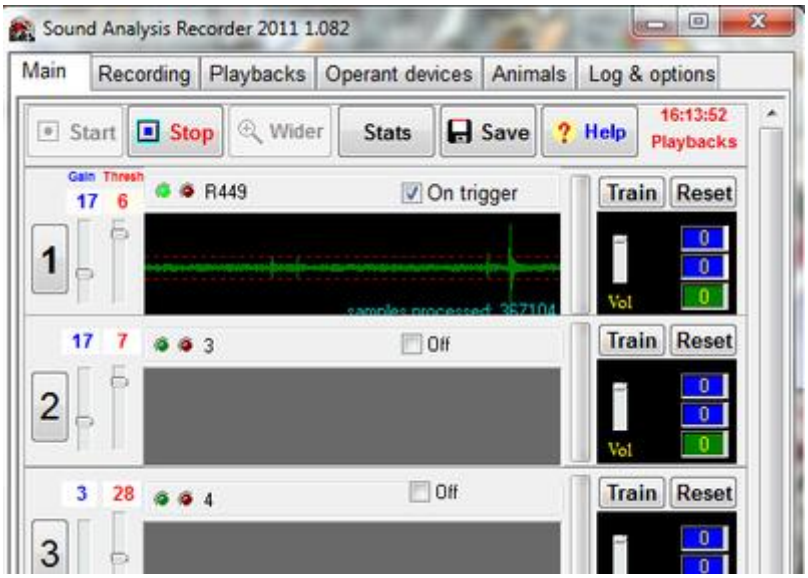
Step 3 - Setting the Sound Storage Folder

Go to the File IO Paths tab. By default, the path should look like this:



Once (actually a little bit before) sound recording is triggered, the SAP2011 recorder starts saving a temporary wave file into the hard disk. This storage folder (temporary file path) must exist, otherwise, nothing will be saved ever. Once recording is triggered off, the recorder "decides" to either delete the temporary file, or to move it into a "Complete files path". Note, however, that "complete" from the point of view of the recorder, is the input folder of the "Live" module, which perform spectral analysis and made a final decision about the storage. If you have two hard disks in your computer, it is recommended to use the other hard disk (D) for both temporary and complete storage, so as to leave the system hard disk (C) protected.

That's it! Click OK to exit the configuration window (for now, ignore the warning about changing bird's name) and click "Start". If all is cool, you should see the oscilloscope display moving for the channel you just set as shown below.



now stop the recording, and repeat for all other channels.

Once done, please continue to the next session about recorder control, it will provide you with important information about how to managed the recording parameters.

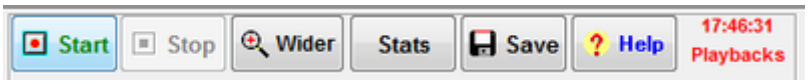
Troubleshootings

1. Nothing happens, no green oscilloscope can be seen: First is 'active' checked? Try clicking "stop" and start again. If nothing happened, make sure that there are no duplicate channels and that the channel name is valid. If the problem is not solved, click stop, go to channel configuration, click output selection and click 'clear selections'. Click OK and try again.

2. The oscilloscope is frozen: Click "Start" and look at the bottom right of the channel: Are the numbers moving? If yes, recording is happening, but the gain is too low. Move the gain slider up (blue label) and see if any signal appears. If this does not help, it is likely that you have a hardware problem (microphone not connected?). If the numbers are not moving, the channel itself is not working – try restarting your computer.

Recorder controls

The Recorder Control Panel allows you to start and stop the engine, to change the overall appearance of the recorder, and to save the recording and playbacks configuration.



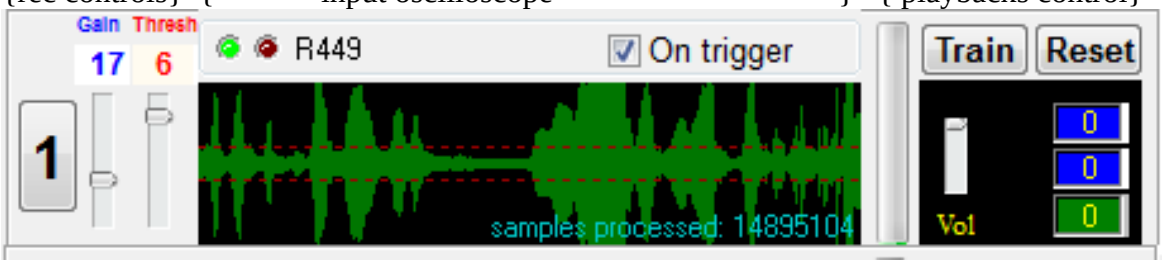
Start & Stop: All the 10 recording and playback channels are controled by a single engine. Starting the engine and stopping it therefore apply to ALL the functionality of the recorder. Each channel (recording and playbacks) is enabled and dissabled seperately in the channel control.

Wider/Smaller: This button tuggles between compact display and large display where oscilloscope is twice larger for each channel.

Stats: Clicking stats will display several statics about the recording events. It is better to display those in the Wider mode:

Save: Save all channels configurations including recording inputs, outputs, sessions, and playbacks.

{rec controls} {----------input oscilloscope---------- } { playbacks control}



The Oscilloscope Display The scrolling display allows you to see the amplitude envelope of the signal scrolling (slowly or fast) from right to left. It will allow you to identify, for example, song syllables. The horizontal red doted lines represent the recording threshold. We will talk about this threshold in details shortly, but note how easy it is to see the cases where signal amplitude passed that threshold. Peak Count is showing you the number of samples that passed the threshold during the current "session".

Gain: moving the gain slider changes the gain of the display of the oscilloscope. It has no effect on the recording, so feel free to play with it. You want to set it such that vocal sounds are clearly visible.

Thresh: In the default 'on trigger' recording mode, recording session starts when the oscilloscope passed a threshold, that you can see as two horizontal red dotted lines. Set the threshold just above the noise level.

LDE lights (green, red): When a recording session takes place, the green LED light will turn on. When sound level is too high and sounds are getting clipped (this has nothing to do with the gain, but with your hardware recording amplification level) the red LED will turn on.

Keep in mind that SAP2011 allows *pre-triggering* of recording, namely, the recorder keeps a

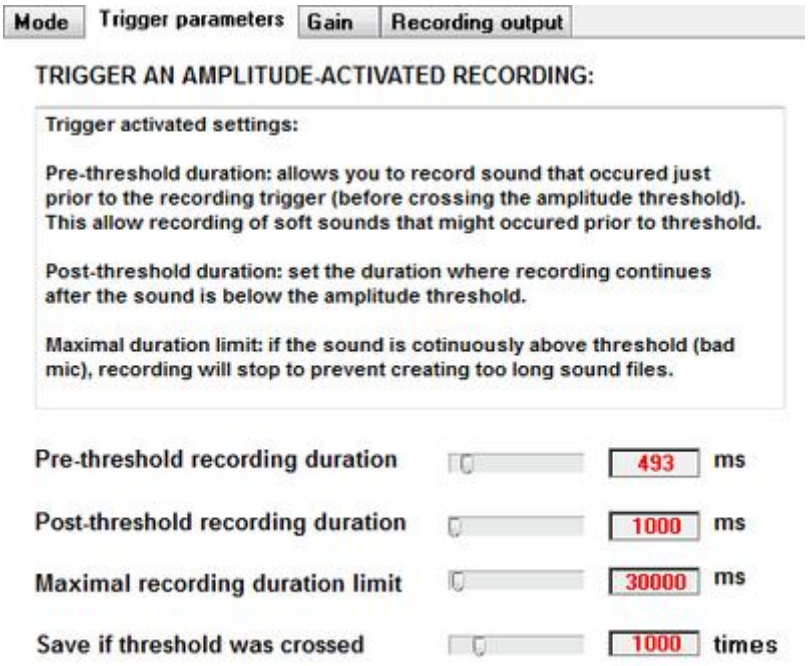
ring buffer of the signal so that it can decide if to save or discard a recording session retroactively (a few seconds later). Therefore, depending on how selective you like the recorder to be, you might want to see a larger chunk of sound in the display mode. Moving the scroll slider to the left allows you to see several seconds of sound signal, e.g., to detect a song bout by visual inspection. Of course, the display is irrelevant to the detection algorithm, but it allows you to judge the behavior of the detector based on your visual inspection.

Setting recording channels

Click the Recording tab. Here you can control important recording parameters including recording mode for each channel, trigger parameters, and recording output.

Mode: here you can quickly set recording modes of each channel. Your changes only takes effect after stopping and restarting the recorder.

Trigger parameters:



These parameters apply to all channels and it is very important to set those right. The pre-threshold control will allow you to record soft sounds prior to the trigger event. The post threshold has a similar function and we recommend to have it no less than 1000ms. Mximal recording duration is there to avoid continuous recording of a very long file into your hard disk, and the save threshold can be used to avoid recording of very short sound events.

Gain: these sliders control the recording gain -- we recommend that you keep them at 1 and use your hardware amplifier controls instead.

Recording output: For each recording channel you must set two folders -- one for temporary storage of 'real time' sounds and another for the recorder output, which we call 'stage 1'. In most cases, you will want to use the accompanied Sound Live module to analyze those stage 1 recording and make a final decision what to save.

Calibrating threshold to background noise level: in general, you should set the detection of a sound event (triggering) to a level only slightly higher then that of the background noise level. Remove the animal from the training box, start the *SA+ Recorder*. Make sure that the oscilloscopic view turned on.

Click the Recording tab.

Change options and check the monitor only checkbox (both Active and Monitor only should be checked).

Click ok, and then change the peaks value (top of the Recording window) to 1, and the occurred within value to 0.

Now go to the main window and click Record.

Locate the Thresh slider just below the oscilloscopic view and turn it all the way to the left -- a green light should turn on above the oscilloscopic view.

Now move the slider slowly to the right until the green light turns off, and remains off (make sure that no sound enters the training box). Repeat for all boxes. This is the correct threshold level (you can go a bit above it, but not too much).

Set the mode of each channel back to Recording, and turn back the recording setting to normal (do not leave the occurred within level at zero!). Place the animal in the box, and as long as the animal stays quiet, you should see that *SA+* records nothing.

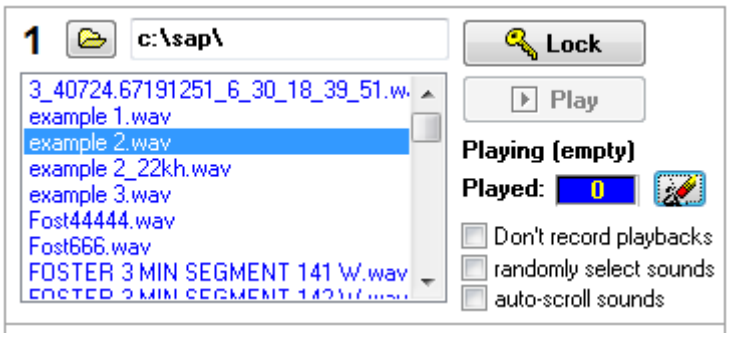
Any sound produced by the animal (including cage noise) will trigger a recording session, indicated by the green light.

Note: that if you decide to change gain, you must recalibrate the threshold level, so it is recommended to first set the gain to convenient levels and keep it unchanged thereafter.

Setting playbacks

Setting the Playbacks regime is quite complicated, and in SAP2011 we developed several utilities aiming at simplifying it. To set playbacks you need to determine the playbacks sounds, the sessions, and the triggering mechanism. The setting tab includes buttons that lead you to the tabs as following:

***Select playback sounds*:**
You may select one or several sound files, in one or two lists for each animal.



First, arrange the wave file or files you want to play in one folder and use the folder selection button to present the list of wave files as shown above. Select one of those files with you mouse. Start the recorder (playback cannot work while recorder engine is off) and then click Play. Once everything is set right press the Lock button to prevent accidental change of the playbacks.

Don't record playbacks: Checking this option will mute the channel just before playback starts and until the Gap timer (see in sessions tab) is elapsed. This will avoid contaminating the recording with the playbacks, at the cost of losing data about vocal interaction of the animal with the playbacks sounds.

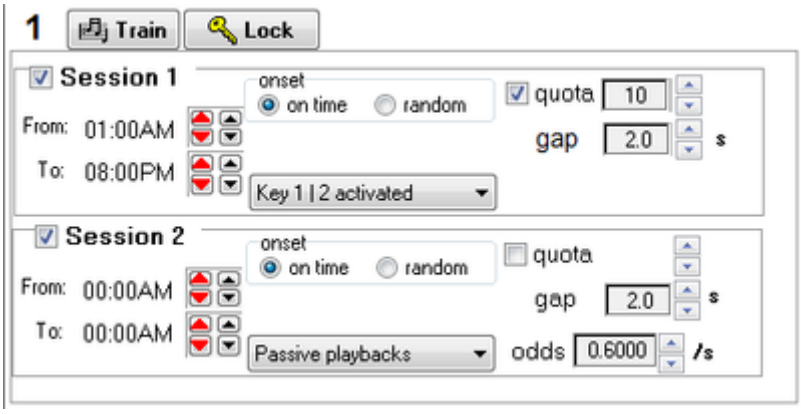
Randomly select sounds: This option will select a wave file randomly from the folder each time playback is triggered.
Go to the "Playbacks List" tab.

Auto-scroll sounds: This option will scroll the playbacks one by one to the end of the list, and then back to the first sound etc.

One/two lists per session: This option allows toggling between a simple, one list training, and a two lists training. The latter can be used for operant training or for preference tests (e.g., key 1 -> list one; key 2 -> list two). See more details below.

Set playback sessions:

Click the sessions tab, and for each animal you should see a control panel with two sessions:



For each session you should set start and end time, and then determine the training mode:

Passive Playbacks: This mode will play sounds randomly according to settings of the odds/second, gap and quota, during each session.

Key 1 activated: You may set up to two keys per bird (see operant training below). This option has Key 1 as the trigger of playbacks, within the quota for each session.

Key 2 activate: same, but for the second Key.

Key 1 | 2: In this mode, pressing on either key will trigger the same song playback.

Quota: maximum number of songs per session.

Gap: Refractory time between playbacks. Should be at least as long as the longest playback.
Note: if the key is blue that means that playback are not active. This could be because the playbacks list is empty or because training session is not currently active. Click the 'playbacks quota' tab and set the training session times:

Testing the playbacks: In the main window, press down the training button:

The color of the rectangle indicates the playbacks state:
Teal - session is on
Black - session is off
Olive -- session is on, but quota has expired
Aqua -- playback is happening right now

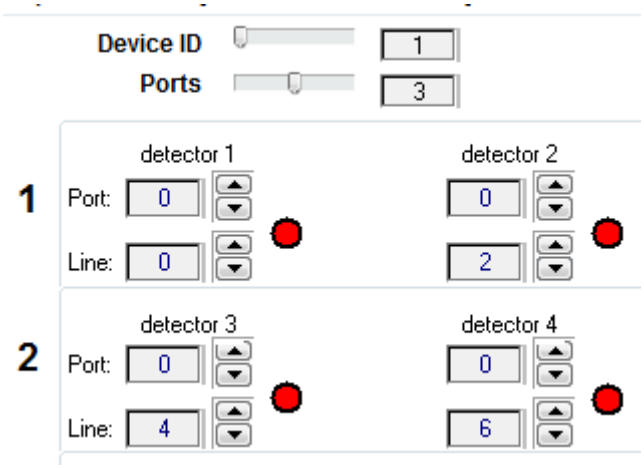
The objects within the rectangle indicate what playback types are enabled:
top circle: key 1 playbacks enabled.

bottom circle: key 2 playbacks enabled.
speaker: passive playbacks enabled.

The color of the circle indicates:
Yellow -- key position is off
Red -- key is pressed
Blue -- key is disabled (e.g., nothing to play).
Important: you can only click Play when the engine is started! Go to the main tab and click "Start".

Setting operant devices

Assuming you already installed a digital I/O National Instruments card and set up the keys as documented in the installation chapter, here you set SAP2011 to recognize those keys. In the Operant device tab, check 'enabled operant training' and you should see this display:



Device ID: Usually set to 1, unless you have more than one NI card in your computer.
Ports: Number of digital I/O ports in your card (usually at least 3).
For each key, you need to set a port number and a line number.

Chapter 8 Contents:

Introduction to live analysis

The automated sound recognition system
A brief review of the Recording & Live processing functionality

Automated animal-voice recognition makes *it possible to record and analyse an entire song development.* Here is the vocalization detecting algorithm. Steps 1-2 take place at the Recorder, and the rest takes place at the processing live module.

1. The *Recorder* captures sounds from an audio channel into a memory ring buffer. It examines sound amplitude in real-time, and if the amplitude is higher than background noise level, a recording session starts (including the pre-trigger ring buffer). Recording to a temporary wave file continues until sound amplitude is below threshold for a certain duration (1s by default). During the session the recorder counts the number of peaks in the waveform the passed an amplitude threshold.

2. Once a recording session has ended, the recorder decides if to save the file based on the number of peak amplitude events that passed the threshold during the recording. The wave file is then either moved to the stage 1 folder, or deleted. The stage 1 folder is the input folder of the Sound Processing Live module.

3. The 'Sound Processing Live' application now reads the sound files from the stage 1 folder. It was separated from the recorder to enhance recording stability (usually ensuring that recording will persist no matter what happened during the later stages of analysis).

*4. M*ulti taper spectral analysis is performed over the entire sound file, and song features are calculated.

5. The sound is segmented into syllables and bouts units.

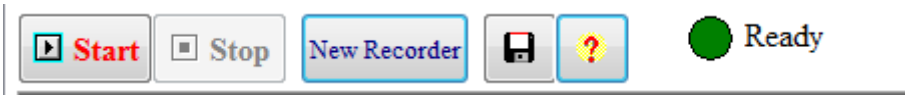
6. A final decision if to accept of reject the sound is made based on the number of syllable, their duration, and the bout duration.

7. If sound was rejected, the file is deleted.

8. Otherwise, the module saves all or some of the following:
- The wave file (under a new name)
- A table of syllable features
- A table of raw features (every 1ms)
Based on this analysis (that occurs about 10-20 times faster than the real-time progression of the sound) *SAP2011* decides whether to save the sound or not.

The main window

The Main Processing Live window



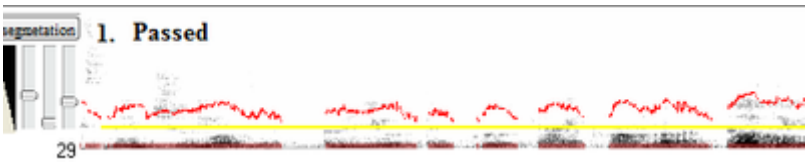
The control panel includes a status indicator: red = not ready, green=ready. If all is okay, you should see a message "Ready" next to the green indicator. If this did not happened, the most common reason for the delay is a large number of "waiting files" in the input folder. The Processing Live module has several state parameters for each channel. To save those parameter values click on the save button.

Each channel shows the name of the bird (in the example below "R449") and you should see the same name on top of the recorder channel as well. The sliders are as in the Explore & Score module, leftmost for display contrast, right most for segmentation threshold, and middle one for a secondary segmentation. All segmentation parameters can be set via the segmentation button at the to right. Note that changes in those sliders position takes an effect for the next sound, and not in real time. We therefore recommend that you first make yourself familiar with how these sliders work in the 'Explore & Score' module.

Starting the automated sound processing for the first time
When you click the 'start' button, *SAP2011 Live Analysis* start monitoring the input folders to detect new sound files. Then, it processes each sound file separately. It first performs spectral analysis and feature calculation, segmentation and computation of syllable features. Based on those measures it 'decides' if to save or discard the data. A decision to save means that 1) the sound file will be moved to a permanent storage address, 2) the raw feature file will be created and saved in a permanent storage address, and 3) the detected syllable features will be saved by adding records to the syllable-table of this animal.

Vocalization or noise? The *Sound Analysis Recorder* already eliminated all the 'empty' sound streams from the recording channel based on simple amplitude filtering. We now need to make an accurate determination for each time window, if it contains animal vocalization or not. *SAP2011* uses several criteria to make this determination, and here we only discuss (in a nutshell) the two central controls, which are amplitude and Wiener entropy thresholds. The yellow slider at the left of each spectral display is for controlling amplitude threshold, whereas the red slider is for controlling the Wiener entropy threshold. In other modules of *SA+* you can change these controls 'live' while *SAP2011* interactively updates the display and the calculation of features.

Setting the amplitude threshold:
Once recording starts, you should see sonograms appearing in the channel window. Set the segmentation threshold (right slider) to a position when only visible vocal sounds are segmented, as shown below:



Analysis summary:
On the right side of the channel you can see the outcome of segmentation statistics, with number of syllables, duration of longest syllable, and duration of longest song bout. Those results are used as criteria for saving or deleting the sound file (see next section).

Save the sound? Decision parameters and options:
For each channel you can select the following options (top right, red font):
pause: the channel processing is off
save only sounds: save and delete sound files based on saving criteria (see below) but delete the song features and segmentation data.
save syllable table: in addition to saving the sound file, it add recorder of syllable features (time, duration, mean amplitude, mean pitch, etc) into a syllable table.
save raw table: in addition to saving the sound file, save vectors of raw features (e.g., amplitude, pitch, etc).
save all tables: save sound file, syllable table, and raw features table.

Input and output settings

To check for this, go to the SAP recorder and click the Recording tab. The Sound input and output monitor shows the cue of sound files generated by the recorder that are waiting to be processed. In case the cue is empty it could be either because all sounds have been processed already or because the output folder of the SAP recorder (labeled as "Stage 1 sound folder" is not the same as the input folder of this module. You should first validate that the input and output chains are correctly specified.

The blue strings in the box represent the wave files cue waiting for analysis. Below, you can see the last saved file.
Important: Please make sure that input folder and output folder ends with "\".

Input settings: Remember that the input of this application is the output of the *Recorder*. Any sound file saved by the recorder should appear on the file-list window. You must make sure that the input folder is correctly specified. You can click the buttons to navigate to the appropriate folder, or type the folder address in the edit box. However you can do this only when the engine is stopped or when the channel is paused.

Output settings: the output of the sound processor is divided into three categories.

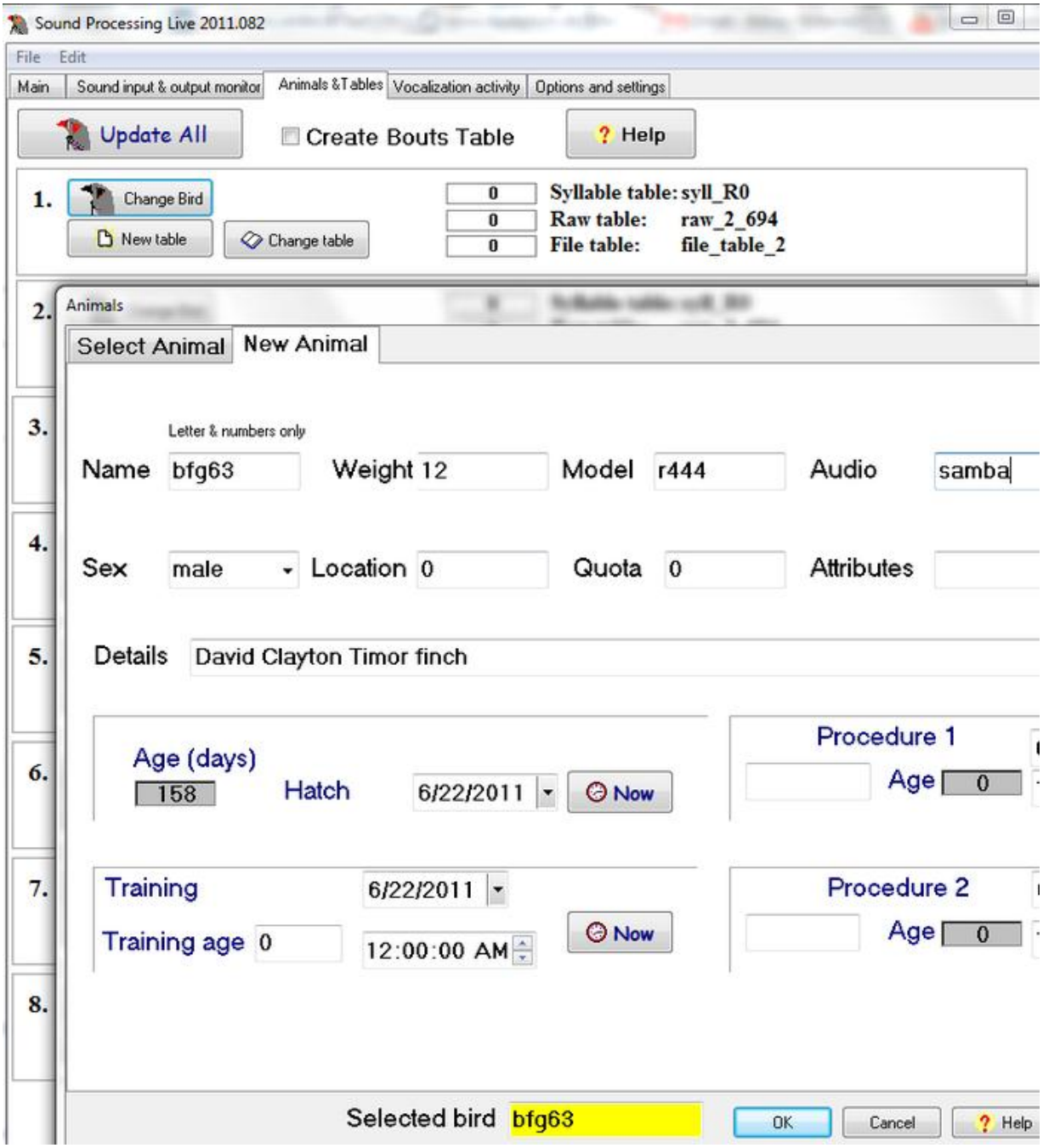
Sound output: The first channel in the sound output is where sound files that matched a criterion (e.g., sound files that contain song bouts) will be moved to. Note that other files are permanently deleted (see batch operations for some alternative approaches of sorting rather than deleting files). Sound files are not only moved, but also renamed. By default, *SAP2011* ignores the original input file name and generates a file name with the following file annotation format: R449_39365.70262828_10_10_19_31_2.wav
Note that the same date & time stamp is also used as a serial_number in the syllable database, allowing easy access to the raw sound data of each syllable.

Should I save all the sound files? In many cases, you will find that the most significant issue is the load of accumulating sound files on your hard disk. You may ask yourself if saving all the raw data is indeed necessary. The practical answer to this question is that with appropriate maintenance, saving the entire sound data is inexpensive and easy. The cost of a DVD/R disk is now down to $1 (when buying in bulk), and the overall cost of saving the entire vocal ontogeny is about $15. *SAP2011* provides an easy way of saving raw sound files and keeping track of the data. Raw data are an asset that our community may use and reuse and saving raw data is generally a good idea.

Saving sound output through the network: You may define the output folder pass as a standard network address. E.g., //Peterson_computer/c/*SA*P/bird1/ will work just like a local folder address. Note however, that the network might crush. If *SAP2011* detects a network crush or inaccessibility to a computer, it will revert to save the sounds to a default local drive, called c:\rescue\. If this has happened, take your time, solve the network problem, and then click ‘stop’ and then ‘start’ to resume the selected destination you previously set to the Sound Processing Live engine.

Animals & Tables

Click on the 'Animals & Tables' tab, and click 'change animal':



Here you can set new animals:

Setting Animals

Setting the animal: Before recording and processing you must make sure that animals are appropriately identified by each channel. The Database module of SAP2011 provides you with the most complete set of tools to add animals to the database. You then only have to click "Change Bird" to switch. The recommended method is to set the animal for each channel using the Recorder, which also allows you to add new birds. Once done, just click "Update All" in this module so as to synchronize the Processing live with the Recorder. Note

that if there is a cue of waiting files waiting unprocessed from a previous session, these data will be falsely added to the new bird table.

If you are processing several birds at the same time, it often happens that you change one bird "on the fly". In this case do not click "Update All" and instead click "Change Bird" in the appropriate channel.

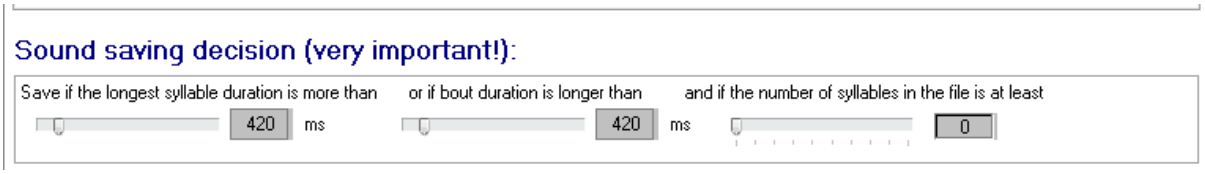
Saving Tables

By default SAP2011 Sound Processing Live saves only the wave files that were "accepted" and all the features that were calculated are discarded. Of course, you can always recalculate them using the batch module, but you should decide of you want to save them in real time. There are many reasons to save, but keep those two issues in mind:

1. If you want to save the raw features, namely the continuous feature vector, you are saving a record of 14 fields (about 200 bits) every millisecond of vocalization, which is quite a lot. For example, the raw wave files (your sound data) is about 700 bit / ms). Taking other database issues into account, you are increasing the storage need by maybe 30-40%. Another consideration is that the database storage at this fast rate takes a toll on your memory and processor, which can be quite significant.
2. If you want to save the syllable table features, then storage, memory and processing costs are negligible. However, the syllable table is only as good as your segmentation is. This is why we often recalculate the syllable table off line using more careful choices of segmentation parameters. We also have Matlab routines that can explore different thresholds and optimize the segmentation. So, in most cases, you should consider the real time syllable table as crude, and try to improve it off line.

Saving data criteria

Based on the segmentation criteria, the sound is segmented into syllable and bout units. A final decision if to accept of reject the sound is made based on the number of syllable, their duration, and the bout duration. Click the options & settings tab:



The criteria for saving a sound file (and features) is a combination of these conditions: Note that longest syllable & bout duration are determined in part by the amplitude and entropy thresholds. Make sure that those threshold are appropriate by observing the light-blue && red lines at the bottom of spectral images.

Graphs & Displays

Around the clock monitoring of vocal activity

It is often useful to be able to see how vocal activity is distributed around the clock. Many animals vocalize more in certain hours, and also, the health of the animal, as well as the health of your system (bad microphones…) will reflect in the circadian distribution of sounds. Therefore, SAP2011 provides an automatically updated online display of vocal activity. Furthermore, it shows 6 different curves, to capture distribution of sounds of different 'kind'.

Note: In the coming two chapters, we will present methods of classifying syllables to types, based on their feature distribution. Here we only present a first-pass, and rather arbitrary categorization of sounds. To do off-line assessment of vocal activity, we suggest you use the cluster analysis approach.

The 'around the clock' page tab includes 8 graphs, one for each bird. In each graph you will find a 24h count of six syllable types. Each count is performed within intervals (bins) of 15 minutes. Syllables are automatically (and arbitrarily) categorized into the following types:

Type	Definition	Comments
1	Duration < 100ms, mean FM<10	Non-modulated short notes
2	Not type 1 and mean pitch > 3000	High-pitch notes
3	All other sounds with duration<100ms	Introductory notes and cage noise.
4	Duration > 100ms, mean FM<10	Non-modulated long calls
5	Duration>100ms, mean pitch>2000	High-pitch song syllables
6	All other sounds with duration > 100ms	Other syllables and cage noise

Note: because types 4-6 are often much more rare than types 1-3 (and are often of main interest), the display shows a 2x multiplication of their values.

Chapter 9: Batch & Automation

Chapter 9 Contents:

Introduction

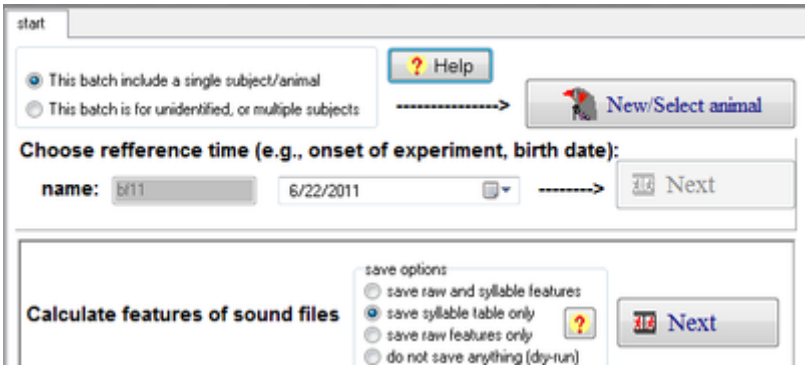
In this chapter we document two types of batch operations: *features batch* and *similarity batch*. The features batch is design to process a large amount of sound data stored as wave files. Similarity batch can perform many thousands of similarity measurement automatically. In addition to adding sound files manually, SAP2011 allows you to use the output of other modules to target specific sounds, e.g., those that belong to a certain cluster. In general, all the fields in a syllable table can be used to quarry retrieval of two sets of sounds followed by MxN similarity measurements.

The batch can be used to:

- Segment the sound to syllables, compute syllable features and store them in 'syllable tables'
- Calculate raw features vectors from sound data and store them in 'raw feature tables'
- Compute similarity between vocal sounds in a batch
- Sort sound files according to content

Feature batch modes

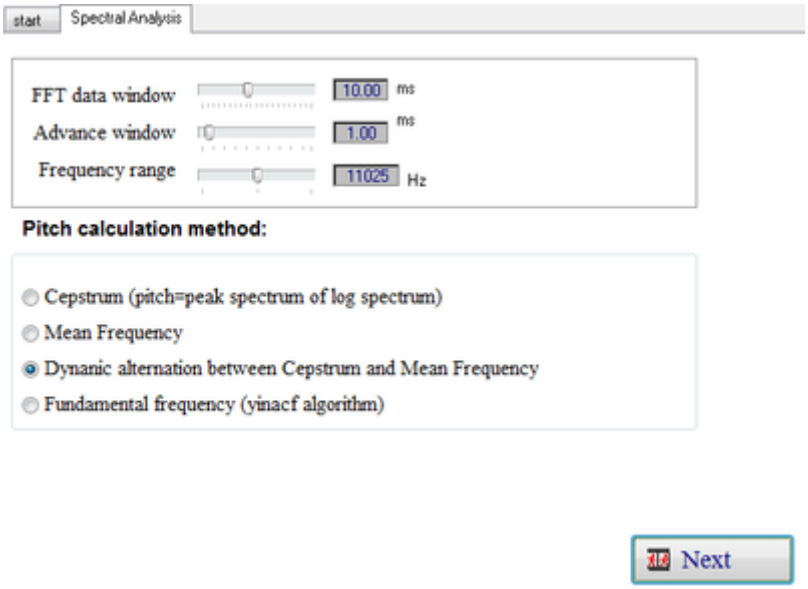
Open the feature batch, and select from the following modes:



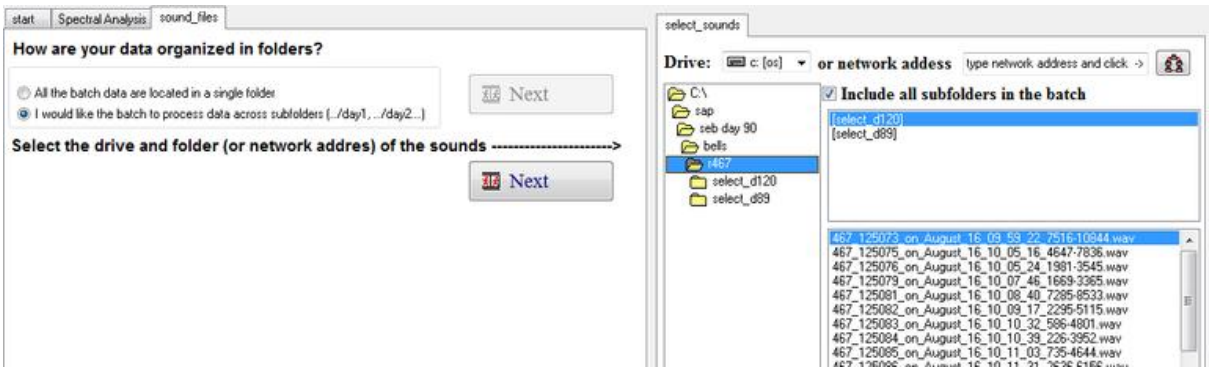
Select if the batch includes data from one specific animal -- if so, you will need to select that animal or create a new record for that animal. Otherwise you may simply name the batch and click next.
The choices that now appear below will affect what is calculated and what is saved: if you select saving of syllable table (options 1 and 2) you will need to determine the segmentation criteria for syllables. Otherwise, you may save only feature vectors of unsegmented data. The last option is for 'dry run' without saving anything.

Feature batch step by step

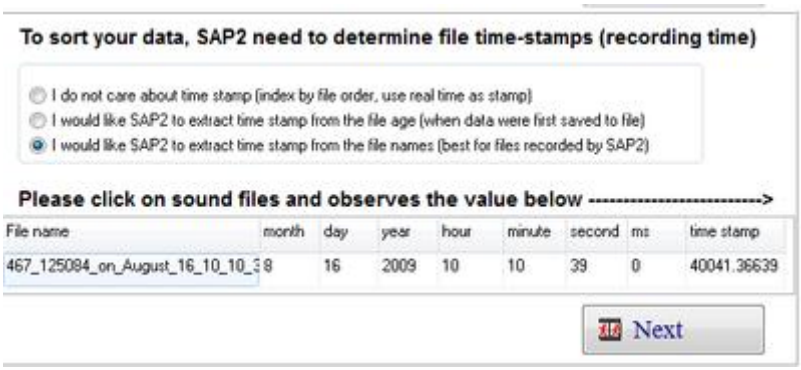
We will now go over the steps of the feature batch with the option of saving both syllable tables and raw features table. Other modes are similar, but skip some of the stages if they are not necessary. After selecting the animals and batch mode you should see this screen:



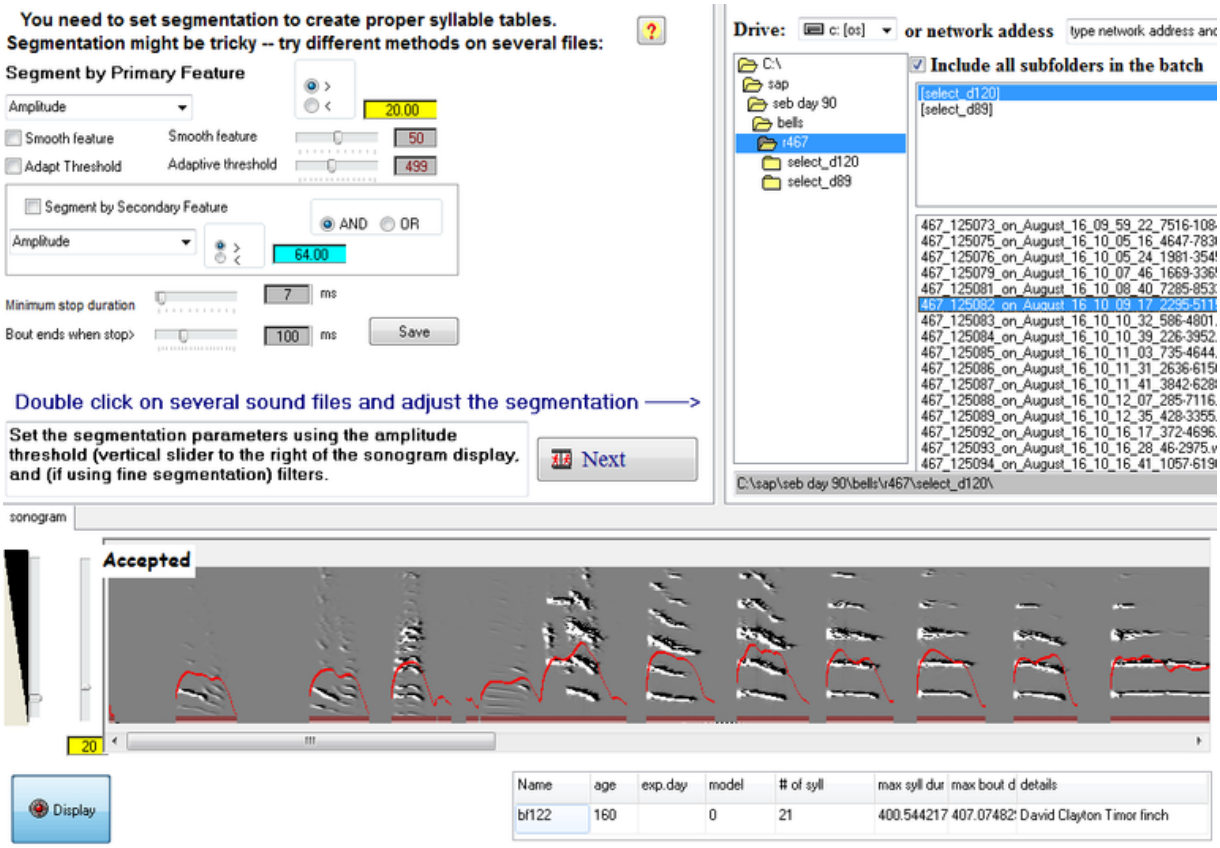
These choices are presented for your convenience but the parameters of spectral analysis should be set in Explore & Score, and then this window should show you the preset options (as you set in Explore & Score, Sound 1).
Next you will need to identify the location of the sound files:



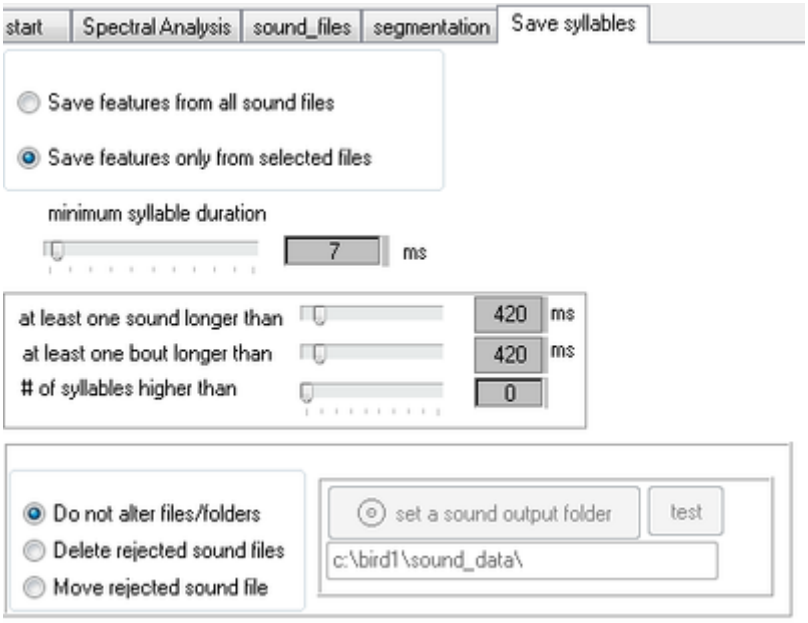
If you select the sub-folders option, make sure that the root folder is highlighted on the left, subfolders appear on the top right, and wave file on the bottom right as shown above.
Next, you need to determine the appropriate time stamp for each wave file and test it. This is an important step since the time stamp will determine the order of data in your tables.



If you do not care about order, or like the order of record to simply follow the order of files without any date & time information select the first option. If the wave file you use has no date & time information in the file name, but the age of the files mirror the age of the data (which is often the case) you might select option 2 above. Note, however, that Windows file age time stamp is not very accurate. Therefore, for files generated with SAP2 or later, selecting the third option will often provide the most accurate time stamp.
Once you made your decision, try it by clicking on a few files and observe the time stamps output to ensure accuracy and data integrity.
Next we set the segmentation criteria (this option show only if you choose to create syllable tables).



Note the "Accepted" label on the sonogram: it means that this sonogram passed a criteria for saving data. Note also the segmentation outlines. You should inspect the segmentation carefully across several sounds and adjust it appropriately.
The criteria for saving data are presented in the next step (this too, only applies if you chose to save syllable tables):



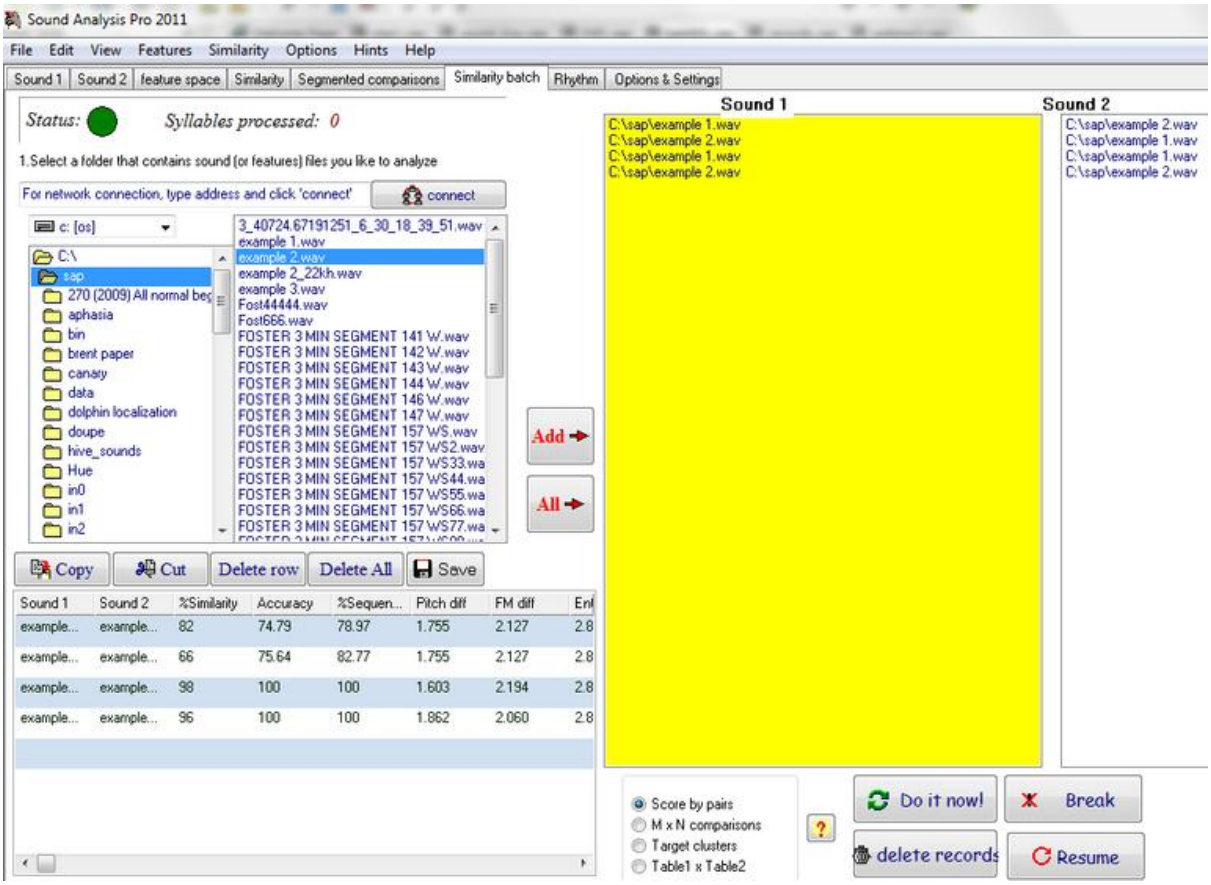
The next stages will allow you to choose the database tables output of your batch. This includes:

File table: with indexes for file names and locations, that include information about all the wave files used in the batch. This table is linked to the raw features table, so that each data point in the feature vector (those can be very long vectors with millions of data) can be linked to the wave files it was extracted from.

Syllable table: there is one syllable table per batch. You may change its name, but better keep the automatically generated name starting with syll_label used to identify syllable tables.

Raw features tables: If your data includes daily subfolders and a large amount of data, it is often useful to have SAP2011 automatically created daily tables. Even those daily tables can be quite big; e.g., a day of singing activity in a zebra finch often contains 20 or 30 million records in each daily table.

Similarity batch



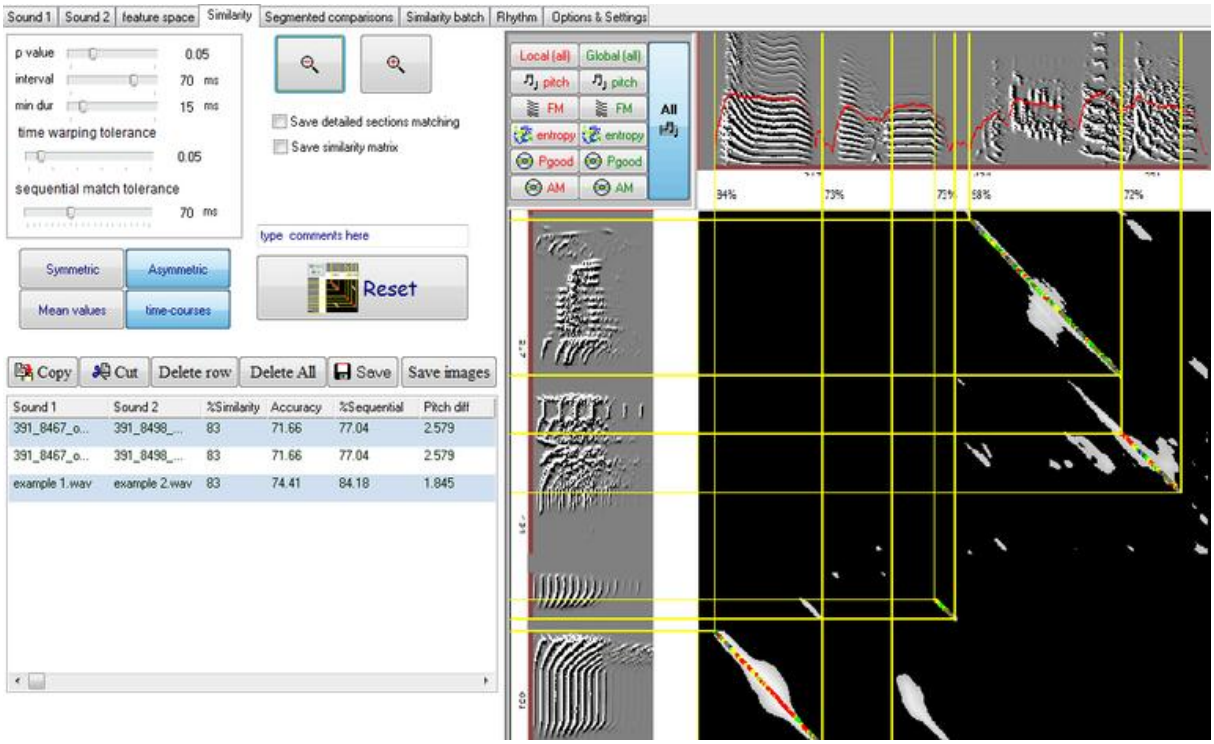
The similarity batch is run by selecting wave files from the file list and adding them to Sound 1 and Sound 2. The active sound list is yellow. You can switch from one list to the next by clicking on it (and it will turn yellow). You can select one file, or several files using control-click, and then move them together to the appropriate sound list.
Note: similarity score batch might take a long time. In particular, we recommend using short files (of not more than one or two seconds of sound).

Chapter 10: Similarity Measurements

Table of contents:

Introduction to similarity measurements

The similarity measurements implement the same feature-based metrics of Euclidean distances used for the cluster analysis, except that here we do not look across thousands of syllables, but rather at the time course of feature values across two specific sounds. Similarity measurements should not be used for comparing simple sounds (such as two tones), in such cases, differences across mean and range of features is preferred. Similarity measurements are necessary for comparing two complex sounds, e.g., a pair of songs or several pairs of complex syllables. Although *SAP2011* segments sounds to syllables, this segmentation is not used as a unit of similarity analysis, that is, we compare everything to everything in the two sounds regardless of syllable structure.



Limitation: Similarity measures must be tuned to each species. The current version is tuned to zebra finches. *SAP2011* makes it very easy to set and save feature scales to other species. Setting the feature scale for a new species is easy. First, set all the sliders to the new position and then click 'save new scale' and type the new species name. You can then browse between the settings by clicking the up and down arrowheads. Note that SAP2011 will always start with the default setting (zebra finch) and you should remember to set it each time to the appropriate species.
The aim of analysis is to address three issues:

- · Assessing the likelihood that two sounds are related to each other.
- · Quantifying the accuracy of the vocal match (assuming that sounds are related).
- · Accounting for the temporal order (syntax) of sounds when scoring similarity.

Similarity measurements in *SAP2011* are quite different than those used in previous versions of Sound Analysis:

- · Similarity measurements are more accurate and more transparent to the user.
- · Both symmetric and asymmetric similarity measurements methods are implemented.
- · Partial similarity scores are shown for each syllable.
- · Users can open two long recording sessions (of a few minutes each) and perform partial. similarities much more efficiently without draining memory.
- · New features include amplitude modulation (AM) and goodness of pitch. We eliminated the spectral continuity feature
- · Automated batching provides a link between cluster analysis and similarity measurements, allowing a fully automated measurement of thousands of sounds, which are automatically opened, outlined and scored.
- · The memory management scheme has been altered to reduce memory allocation error and optimize the section detection.

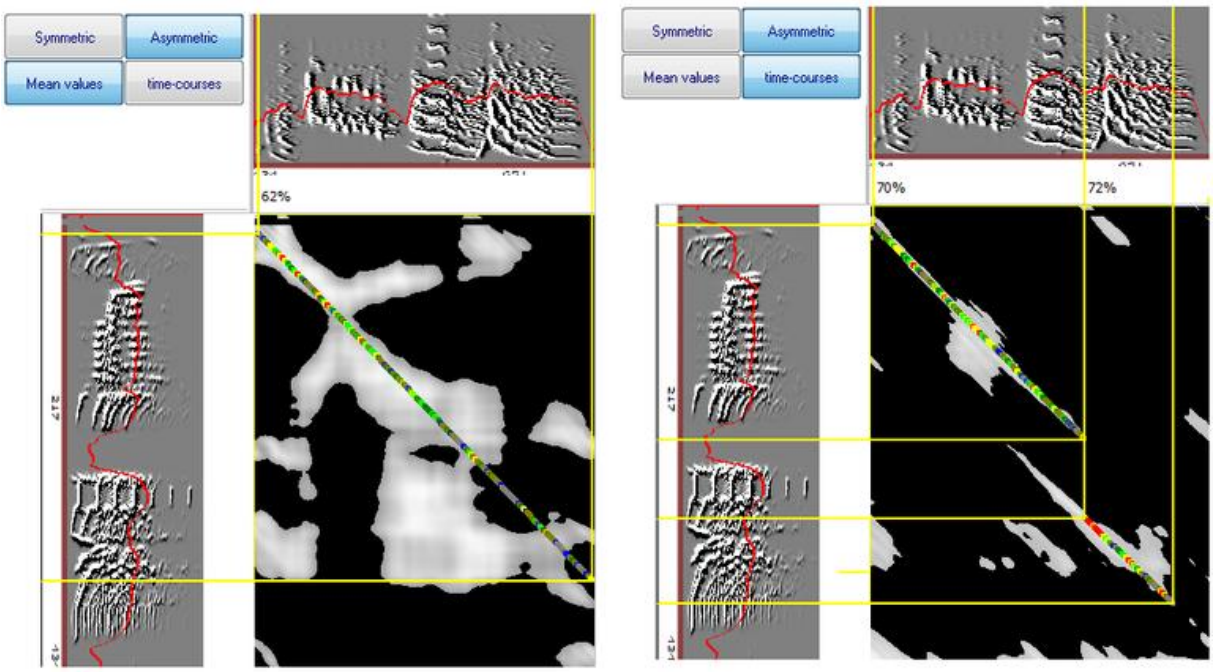
The metric system

An FFT data window, or frame, is a short (e.g., 10ms) interval of raw sound data (a waveform of time varying amplitude curve), which is the unit of spectral analysis. The spectral structure of each frame is summarized by measurements of song features (e.g., Pitch, FM, AM, Wiener entropy, and goodness of pitch). Each of these features has different units and different statistical distributions in the population of songs studied. To arrive at an overall score of similarity, we transformed the units for each feature to units of statistical distances. One can transform the units of pitch, for example, from Hz to units of standard deviation. Instead of SD we use a similar (and sometimes better) measure of deviation called MAD (median absolute deviation from the median). We can then compute Euclidean distances across all features. A similar procedure can be used to compare larger units of time, which we shall call intervals. *SAP2011* uses two methods to estimate Euclidean distances across intervals.

Euclidean distances across mean values: given two intervals, A and B, we first calculate the mean feature values for each feature, and then compute Euclidean distances across the mean features, just as we would have done for a single frame. For example, consider two intervals of 3 frames in each, and (for simplicity) we shall consider only a single feature: A=[10, 20 ,30] ; B=[30, 20,10]. We first average across frames, which gives 20 for both A and B, and obviously, the Euclidian distance between the means is zero. That is, this approach looks at the overall interval, allowing local differences to cancel each other.

Euclidean distances across time courses: given two intervals, A and B, we compute Euclidean distances across pairs of features, A1 against B1, A2 against B2, and so forth. We then calculate the mean Euclidean distance across all pairs. Now consider the same example: A=[10, 20 ,30] ; B=[30, 20,10], the Euclidian difference is
SQRT[(10-30)^2 +(20-20)^2 + (30-10)^2]= 28.3 MADs.

As shown, when we compared single frames, it is not unlikely to obtain short, or even zero distances, but comparing time series, a distance of zero requires that all the pairs of distances are zero. Hence, when examining the cumulative distribution of Euclidean distances across the two methods in a large sample of sounds, the two methods give different results:



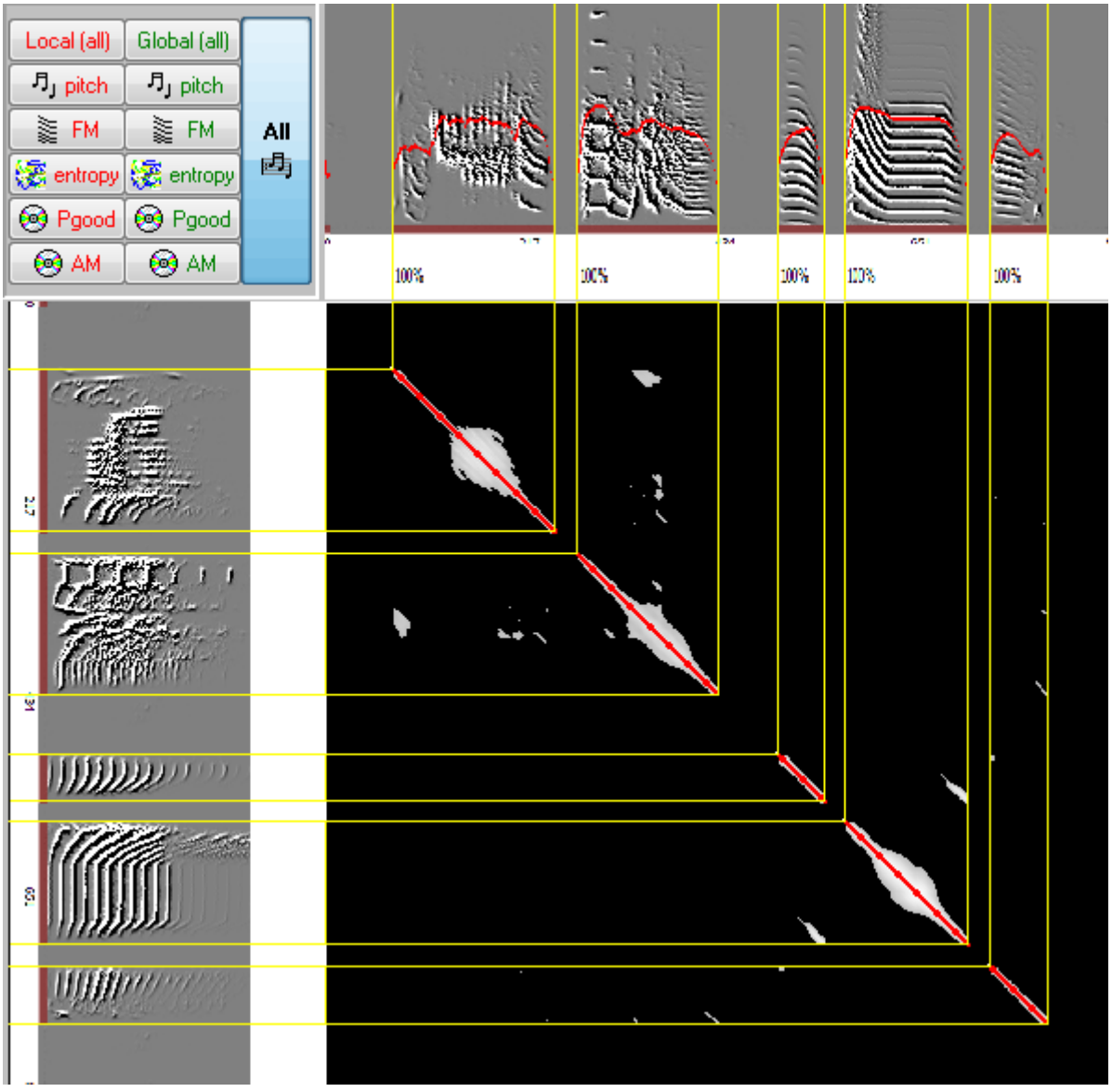
This difference has a very practical implication when comparing songs: the time course approach is good for detecting similarity between two sequences of features that show similar curves of feature values. Note that moving an interval even by a single frame changes the entire frame of comparison. By comparing all possible pairs of intervals between two sounds, we can detect the rare pairs of intervals where the sequential match between all (or most) frames is high. Euclidean distance across mean values achieves exactly the opposite: dependency between neighboring intervals is high and we are looking for high similarity between distributions regardless of the short scale differences.

Note: The difference between those approaches applies also to other SAP modules: for example, the syllable table is based on mean and variance feature values calculated for each syllable, and hence all the table-based methods (DVD maps, cluster analysis) are based on Euclidean distances across mean values. Therefore, when we identify a nice cluster of syllables, we should not assume that similarity measurements based on the Euclidean distances across time series will show high similarity across members of the cluster. In fact, current findings suggest to us that birds stabilize the overall (mean) values of syllable features at a time when the frame-to-frame feature values are dissimilar across syllables.

Asymmetric measurements

Asymmetric similarity measurements are those where sound 1 is the model (or template) and sound 2 is the copy, and we want to judge how good the copy is in reference to the model. For example, if a bird has copied 4 out of 5 syllables in the song playbacks it has heard, we will say that 80% of the model was copied. However, what should we say had the bird produced a song of 10 syllables, including accurate copies of the 5 model syllables and 5 improvised syllables? It makes sense to state that all (100%) of the model song was copied, but the two songs are only 50% similar to each other. To capture both notions we will say that asymmetrically, the similarity to the model is 100%, and that symmetrically, the similarity between the two songs is 50%. We shall start with asymmetric comparisonsl

Start *SAP2011*, open 'Example 2' and outline the entire song. Click the 'Sound 2' tab, open 'Example 2' . For both sounds set amplitude threshold to 34dB to obtain a good segmentation to syllable units. Click the 'Similarity' tab and click 'Score'. The following image should appear within a few seconds (click the +/- zoom buttons to see the entire picture):



The gray level of the similarity matrix represents the Euclidean distances: the shorter the distance the brighter the color; intervals with feature distances that are higher than threshold are painted black. The numeric values of the similarity matrix are not useful to save in most cases, but those can be saved by checking the "Save similarity matrix" box. Then, after clicking score, a popup menu will prompt you to name the text file, where comma deliminated values of the matrix will be stored.

Similarity sections are neighborhoods of intervals that passed the threshold (e.g., when the corresponding p-value of Euclidean distance is less than 5% for all neighbors). Check 'Save detailed section matching' to have those sections automatically saved into the mySQL table 'similarity_sections'. As noted, the gray level represents the distance calculated for each pair of intervals. However the only role of the distance calculation across (70ms) intervals is to set a threshold based on 'viewing' features across a reasonably long interval. The actual similarity values are calculated frame-to-frame within the similarity section, where p-value estimates are based on the cumulative distribution of Euclidean distances across a large sample (250,000) of random pairs of frames obtained from comparisons across 25 random pairs of zebra finch songs. The gray level of the similarity matrix represents the Euclidean distances: the shorter the distance the brighter the color; intervals with feature distances that are higher than threshold are painted black.

Local (frame level) similarity scores: Based on this distribution, we can endow each pair of frames with a local similarity score, which is simply the complement of the Euclidean distance p-value. That is, if a single-frame p-value is 5% we say that the similarity between the two frames is 95%. Local similarity is encoded by colors in the similarity matrix as follows:

Score (1-p)%	Color
95-100	red
85-94	yellow
75-84	lime
65-74	green
50-64	olive
35-49	blue

Section-level similarity Score: We now turn to the problem of estimating the overall similarity captured by each section. First, *SAP2011* detects the boundaries of each section. Then, single frame scores are calculated for each pixel and finally, *the algorithm* searches for the best 'oblique cut' through the section, which maximizes the score. In the simplest case (e.g., of two identical sounds) similarity will maximize on a 45^0 angle at the center of the section. In practice, it is not always the center of the section that gives the highest similarity, and the angle might deviate from 45^0 if one of the sounds is time warped in reference to the other. We therefore need to expand in different displacement areas and at different angles. The default 'time warping tolerance' is set to 5% by default, allowing up to 5% angular deviation from the diagonal. Note that computation time increases exponentially with the tolerance. The search for best match is illustrated below:
We next consider only the frames that are on the best-matching diagonal, and calculate the average score of the section. This score is plotted above the section. Boundaries of similarity sections can be observed more clearly by clicking the 'Global' button. The light blue lines show the boundaries of each section and the rectangles enclose the best diagonal match of each section

Similarity across sections: Note that there are several sections with overlapping projections on both songs. To obtain a unique similarity estimate, *we* must eliminate redundancy by trimming (or omitting) sections that overlap with sections that explain more similarity. We call the former '*inferior sections*' (blue rectangles) and the latter (red rectangle) '*superior sections*'.

Final sections: once redundancy has been trimmed, it often makes sense to perform one final filtering, by omitting similarity sections that explain very little similarity (which are likely to be 'noise'). By default, *SAP2011* omits sections that explain less than the equivalent of 10ms x 100% similarity. Superior similarity sections that passed this final stage are called *final sections*.

The similarity score

In asymmetric comparisons, the similarity score has two major components: the percentage of similarity, which computed over intervals of sounds, and the accuracy -- which is the local, fine grained similarity. The %similarity measure, being computed across relatively long intervals (typically including 50 FFT windows or so), is designed to capture patterns in feature values. Once similar patterns are detected, one may look at the details, just like comparing pictures looking for similar faces (is in % similarity), and then comparing similar faces pixel by pixel (as in accuracy).

Scoring similarity between songs on the scale of a single window is hopeless, as is comparing pictures one pixel at a time. The solution is to compare intervals consisting of several windows. If such intervals are sufficiently long, they will contain enough information to identify a unique song segment. Yet, if the intervals are too long, similarities that are real at a smaller interval size may be rejected and that would reduce the power of analysis. We found empirically that comparisons using 50-70ms intervals, centered on each 10-ms time window were satisfactory. Perhaps not surprisingly, the duration of these song intervals is on the order of magnitude of a typical song note. Our final score of similarity combined the two scales: the 'large scale' (usually 50-70 ms) is used for reducing ambiguity with a measure we call *%similarity*, while the 'small scale' (usually 5-10 ms) is used to obtain a fine-grained quantification of similarity, which we call *accuracy*.

For each pair of time windows labelled as 'similar' for two songs being compared, SAP2011 calculated the probability that the goodness of the match would have occurred by chance as described above. We are left, then, with a series of P values, and the lower the P, the higher the similarity. For convenience we transform these P values to 1-P; therefore, a 99% similarity between a pair of windows means that the probability that the goodness of the match would have occurred by chance is less than 1%. In this case, 99% similarity does not mean that the features in the two songs being compared are 99% similar to each other. In practice and because of how our thresholds were set, songs or sections of songs that get a score of 99% similarity tend, in fact, to be very similar. The SAP2011 procedure requires that there be a unique relation between a time window in the model and a time window in the pupil. Yet, our technique allows that more than one window in the pupil song will meet the similarity threshold. The probability of finding one or more pairs of sounds that meet this threshold increases with the number of comparisons made and so, in some species at least, the duration of the pupil's song will influence the outcome. When a window in a tutor's song is similar to more than one window in the pupil's song, the problem is how to retain only one pair of windows. Two types of observations helped us make this final selection: the first is the magnitude of similarity, the second one is the length of the section that met the similarity criterion.

Windows with scores that meet the similarity threshold are often contiguous to each other and characterize discrete 'sections' of the song. In cases of good imitation, sections of similarity are interrupted only by silent intervals, where similarity is undefined. Depending on the species, a long section of sequentially similar windows (i.e. serial sounds similar in the two songs compared) is very unlikely to occur by chance, and thus the sequential similarity we observed in zebra finches was likely the result of imitation. Taken together, the longer the section of similarity and the higher the overall similarity score of its windows, the lower the likelihood of this having occurred by chance. Therefore, the overall similarity that a section captures has preeminence over the local similarity between time windows.

To calculate how much similarity each section captured SAP2011 used the following procedure. Consider for example, a tutor's song of 1000 ms of sound (i.e. excluding silent intervals) that has a similarity section of 100 ms with the song of its pupil, and the average similarity score between windows of that section is 80%. The overall similarity that this section captures is therefore 8%.

This procedure is repeated for all sections of similarity. Then, we discarded parts of sections that showed overlapping projections, either on the tutor or on the pupil's song. Starting from the section that received the highest overall similarity score (the product of similarity[1]duration), we accepted its similarity score as final and removed overlapping parts in other sections. We based the latter decision on the overall similarity of each section and not on the relative similarity of their overlapping parts. We repeated this process down the scoring hierarchy until all redundancy was removed. The remainder was retained for our final score of %similarity.

Therefore:

% Similarity: is the percentage of tutor's sounds included in *final sections*. Note that the *p-value* used to detect sections is computed across intervals. This similarity estimate is asymmetric and it bears no relation to the local similarity score we discussed above.

Accuracy: is the average local similarity (frame by frame) across final sections.

Sequential match: is calculated by sorting the *final sections* according to their temporal order in reference to sound 1, and then examining their corresponding order in sound 2. We say that two sections are sequential if the beginning of in sound 2 occurred between 0-80ms after the end of S. This tolerance level accounts for the duration of stops and also for possible filter effect of very short sections that are not sequential. This procedure is repeated for all the consecutive pairs of sections on sound 1 and the overall sequential match is estimated as:
Note that multiplying by 2 is offsetting the effect of adding only one (the smallest) of two sections in the numerator. This definition is used for asymmetric scoring, whereas for symmetric scoring the sequential match is simply the ratio between the two outlined intervals on sound 1 and sound 2. Weighting the sequential match into the overall score: In the case of symmetric scoring only the sequentially matching parts of the two sounds can be considered, so it makes sense to multiply the sequential match by the combined score. In the case of time-series comparison, it does not make sense to multiply the numbers, because this will mean that we give 100% weight to sections that are sequential, and 0% weight to those that are not. Therefore, you have to decide what weight should be given to non-sequential sections. The problem is that sequential mismatches might have different meanings. For example, in extreme cases of 'low contrast' similarity matrices (with lots of gray areas) the sequence might be the only similarity measure that captures meaningful differences, but when several similar sounds are present in the compared intervals, it might be shear luck if *SAP2011* will sort them out sequentially or not. In short - we cannot advise you what to do about it, and the default setting of 50% weight is arbitrary.

Symmetric measurements

In symmetric comparisons, the similarity is estimated from the beginning of the two sounds. Comparison is done on the 45^0 of the matrix (diagonal). However, we allow "tolerances levels" (as if the diagonal has some 'thickness) within which we select the highest similarity score. The user can control with the ‘min dur’ slider.

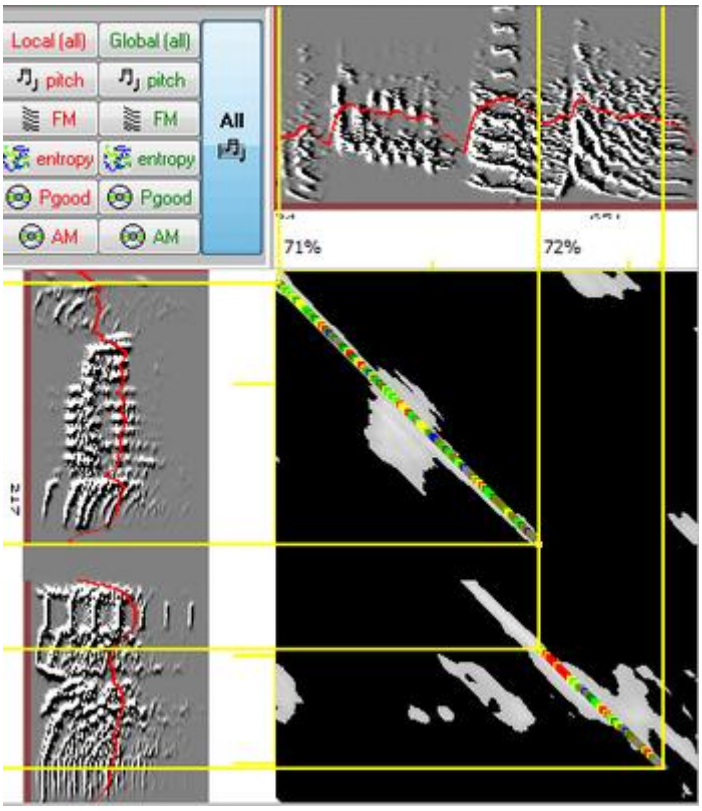
The symmetric measurements are very simple, those are the mean values of similarity scores across the diagonal line. As before, we use both global and local scale, where the global scale is called 'similarity' and the local scale is called 'accuracy':

%similarity: the mean global similarity score (usually across 50ms time windows) across the diagonal with tolerance set by 'min dur'

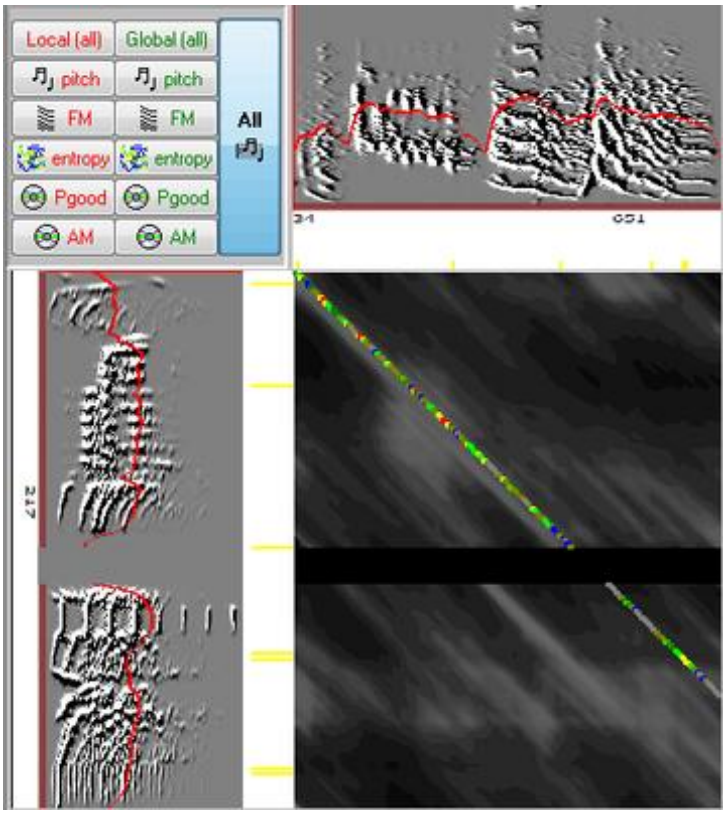
%accuracy: the mean local similarity score (at the millisecond level) across the diagonal with tolerance set by 'min dur'

%Sequential match: note that the shortest interval selected by the user (in sound 1 or sound 2) determine the intervals of comparisons. The ratio between the two intervals is shown in 'sequential match'. Namely, it is min(duration1,duration2)/max(duration1,duration2). It is not a similarity estimate here, just an indicator for how much of the outlined intervals were actually compared.

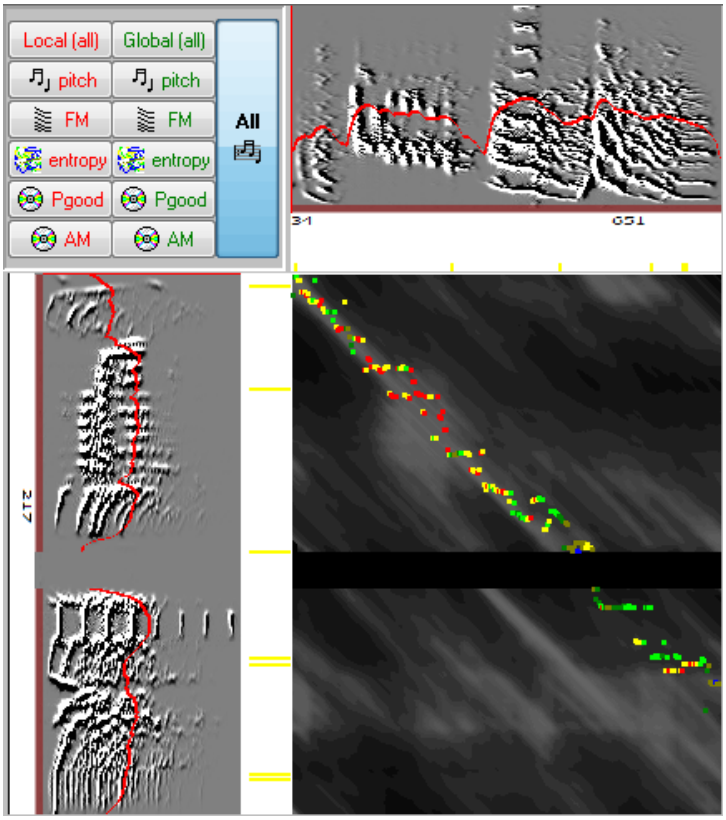
A typical symmetric comparison is done at the syllable level, e.g., take a look at this asymmetric comparison of these sounds:



There are two sections of similarity that form parallel lines with a gap. In symmetric comparison (and no tolerance) the comparison is strictly on the diagonal:



and adding 15ms tollerance looks like this:



and we can see that the similarity is higher in the first part.

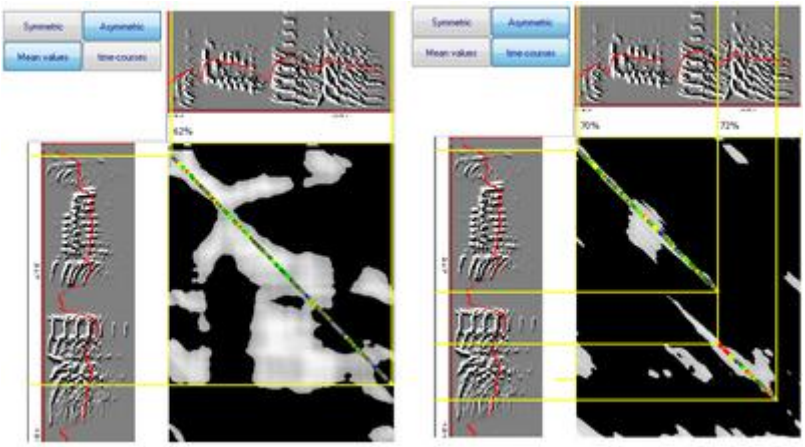
Note that in symmetric comparison reversing the order of Example 1 and Example 2 should not affect the similarity values (including % similarity).

To get the symmetric comparison to behave properly in this case, we need to manually segment the syllable in Example 2:

So, when should one use symmetric comparisons? These measurements are the best choice if we know in advance the sound units that should be compared and if we do not categorize one sound as a model for the other. For example, comparing two calls of the same type across different members of a colony calls for symmetric scoring. Also, once we have identified a cluster of syllables we might want to measure the similarity across sounds of this cluster, here too, a symmetric approach is most appropriate. *SAP2011* provides a method for automatically identifying clusters across files and scoring their similarity (allowing batching of many thousands of comparisons, see batch operations for details).

Time course & mean values

An example of time-course versus mean values comparison:



Above are similarity matrices of intervals of sounds from example1.wav and example2.wav. Note that although we did not change the threshold p-value, the sections are very different in shape and size. The time-course method gives narrower sections that capture the sequential (diagonal) match, whereas the mean-values method shows big blobs of similarity. In both cases we see that the top section is surrounded by a red rectangle, indicating that the final section is identical to the original similarity section. This is not the case with the bottom section. You can see how the original section (blue) was modified (red) so as to trim the redundancy with the top section -- which is also the superior one. It is superior in the sense that it explains more similarity even though its local score is lower. This is because multiplying the duration of the top section by its local score gives higher similarity than that of the lower section. That is, the algorithm takes into account the overall similarity explained by each section. Note also, that although the shape of the sections is very different, the oblique cuts through the sections are similar across methods, hence the overall score and the partial scores are very similar. You should examine the matrix shown in the combo button quite often, since it will help you understand what's going on 'behind the scenes' of the similarity measurements. The white cross shown on top of some of the blue rectangles signifies that this section was excluded because of redundancy.

Exploring similarity across features

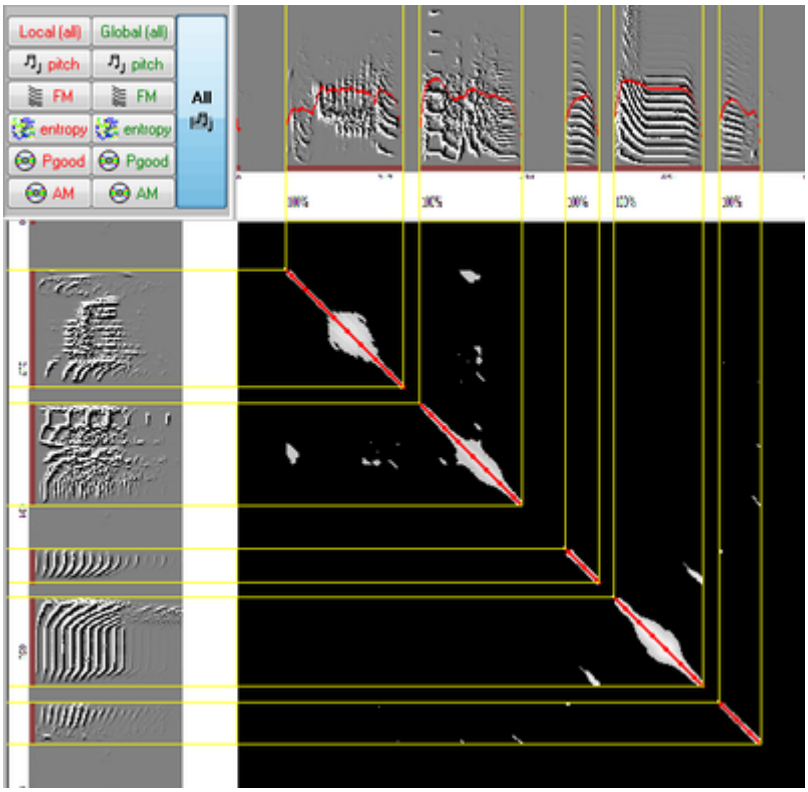
The global similarity estimate is based on five acoustic features: pitch,FM, AM, Goodness of pitch and Wiener entropy. You can view the similarity across each feature separately by clicking one of the buttons in the similarity display group. As noted earlier, asymmetric similarity is estimated in two stages: first global comparisons across 70ms intervals are used to threshold the match and detect similarity sections. Then, similarity is estimated locally, based on frame-by-frame scores. Both local and global distances can be viewed, and those views are useful to assess what might have gone wrong when the similarity results do not seem reasonable. For example, you might discover that the pitch is not properly estimated, which will show similarity across all features but pitch.

The effect of global versus local estimates can be seen in the example below showing FM and AM partial similarities on local and global scales. Note that locally, FM shows a similar area in the middle of the matrix where the two sounds are not modulated, and we can see four bulges emerging from each corner of the central rectangle. Those are the similarities between the modulated parts of the syllable. Since the syllable is frequency modulated both in its onset and offset, we have similarity between beginning and end parts. Now look at the global similarity and note how the rectangle turned into a diagonal line, which captures the similarity in the transitions from high-low-high FM. In addition, we see short sidebands, indicating the shorter scale similarity between the beginning of one syllable and the end of the other. Now examine the partial similarity of AM. Here the local similarity does not show any similarity between the beginning of one sound and the end of the other sound, but it does show strong similarity between the two beginnings and the two ends. This is because the sign of amplitude modulation is positive in the onset and negative at the offset of each sound. Hence, when looking at the global AM matrix we do not have side-bands.

Overall, the message is that by comparing similarity across different features we capture different aspects of the similarity. By taking all those features into account, we can often obtain a reasonable overall assessment of how good the similarity is, and we might then also develop some understanding of meaningful articulatory variables that are similar or different across the two sounds. However, it might also happen that the similarity is good in some features and poor with respect to others, and in such cases, it might be desired to omit some features from the global estimate (this is not something you want to do just in order to obtain a better match!). In the options (similarity tab), you can set different scales and exclusion of features.

Self similarity

It is often useful to compare the similarity of a sound to itself and see similarity of 100% (or at least close to this). There are several reasons that the similarity estimate might deviate from 100%, some are important and others can often be ignored. In the example below, we see a nice 100% diagonal similarity across all but one syllable.



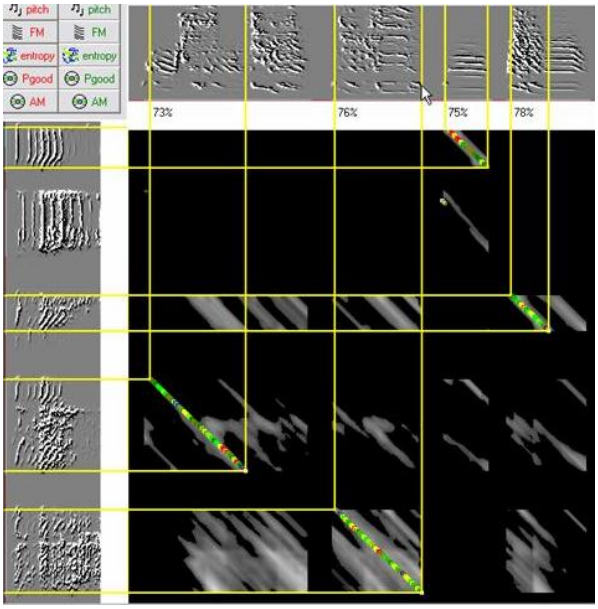
Reducing the min duration with the slider will eliminate this problem, but at the cost of adding 'noise' of many short (and mostly meaningless) sections. Therefore, what you should have in mind is that optimizing the similarity contrast is more important than reaching 100% (as opposed to 95%) during self-similarity tests.

Another issue is setting the amplitude and Wiener entropy thresholds: if those settings differ across Sound 1 and Sound 2, self-similarity might deviate from 100% but usually not by more than 10%. Any self similarity test that gives less than 90% similarity requires a careful investigation of what went wrong.

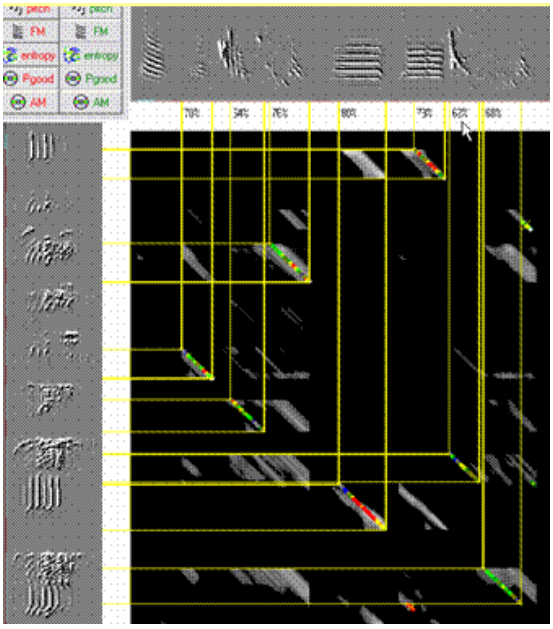
Related and unreleated sounds

Comparing related and unrelated songs

In this example we compare the song model Samba to the song of a bird that was trained with this model.
In the example below, we examine two songs that share a few syllable types, but in a different sequential order. Note that the low sequential match will reduce the overall similarity score to very low levels. In general, there are cases when sequential match is the most robust marker of song imitation, whereas in other cases, it is not. A lot depends on the spectral diversity across syllables. In some cases, and particularly if the recording quality is poor, you will find it very difficult to obtain a p-value that is selective enough to reject and accept sounds 'properly' but even in such cases, one can hope that songs that are 'really' similar, will show more sequential match than other songs.

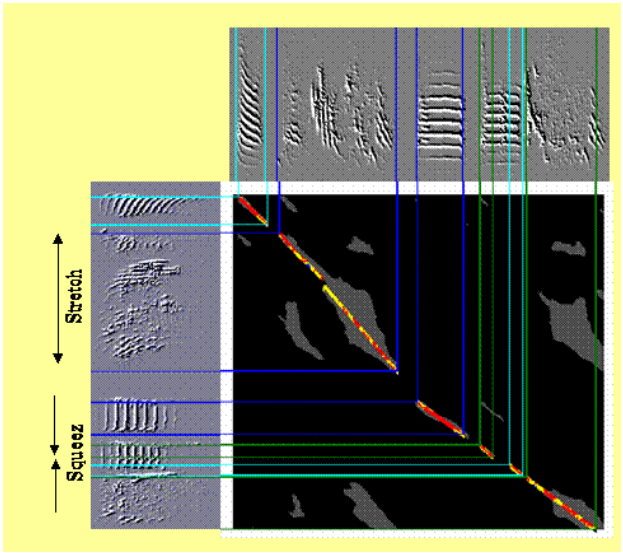


For example, those two unrelated songs have several syllables of complex harmonic structure. Obviously, there are several similar sections, but their sequential match is extremely poor. In sum: in difficult cases, your choices are either to reduce the p-value threshold, or to take the sequential match as a criterion.



In some cases we need to examine similarity between two sounds that differ in their time scale, such that one sound is a stretched or a squeezed version of the other, while the acoustic features of the two sounds (pitch, FM, etc) are otherwise similar. Although *Sound Analysis* does not perform any calculations of time warping, it can capture similarity across time-warped sounds, where the warp is represented as the slope of the similarity section.

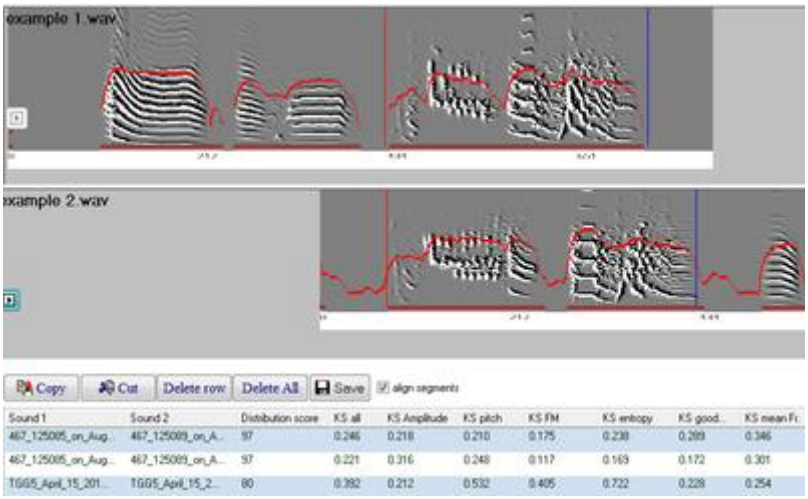
One strength of scoring similarity by sections is that by setting appropriate levels of time warping tolerance *SA+* can easily trace time-warped versions of a syllable type, because the similarity sections will 'aim' at the best angle that maximizes the similarity as demonstrated below:



As shown, warping the sounds did not have much effect on the similarity score, and we can clearly see the different warps as different slopes of sections in the similarity matrix. You can change the tolerance of *Sound Analysis* to time warps: Click options and find the time warping slider at the bottom left. Moving the slider all the way to the left will cause rejection of similarity across warped sounds. After changing the threshold you can click **Reset**, and **Score** to see how the results change.

Segmented comparisons

The segmented comparisons tab allows you to explore two sounds, align segments, play them and compute differences between them. This procedure is conceptually similar to similarity measurements described earlier, but here we are only performing comparisons over the means of features across the two intervals, which does not guarantee that the spectral shapes are similar, but instead provides some overall 'distance' estimates between the two intervals. SAP2011 includes several improvements to the ‘segmented comparisons’ window. First, it is now easier to slide sounds in reference to each other. Clicking just below the sound moves the onset to the mouse position. Double clicking below the sound shifts to ‘sticky mouse’ mode, allowing continuous sliding of the sound. Double clicking again below the sound will release the image.



The next feature you may find useful is auto-alignment: outline an interval in the top sound and then in the bottom sound. You will see that the two outlined intervals have moved to provide perfect alignment. This works nicely using the ‘auto segment a single syllable’ mode (you can set it in either ‘sound 1’ or ‘sound 2’ window). You can then open two song bouts and align syllables to each other in a single click while monitoring changes in the ‘distribution similarity’ score.

Feature distribution scores: based on the MAD table of syllables (see Options -> Feature Normalization -> Normalization of syllable features) we can calculate a distance score, just as we did at the level of frames. When outlining arbitrary intervals of sound, *SA+* scores the similarity between them as if they were syllables, which is not very meaningful as a p-value, but still, it can be used as a non-metric score (just as human score is). To obtain a more generic statistical distance estimate, in a way that is not related to any assumption about the distribution of syllables (that is, without any normalization), the Kolmogorov Smironov statistic should be used.

Kolmogorov-Smirnov statistic: is simply the distance between two cumulative histograms. For each given interval, we can calculate the cumulative histogram of feature distribution and compare it to the cumulative distribution of the same feature in the other interval. KS values are unit-less and additive across features. The KS statistic is simple, valid for all species and has very little assumptions in it (in contrast to all other methods we use to score similarity). It does have one major disadvantage, which is – it uses the interval chosen to estimate some ‘momentary’ distribution of features, but this estimate is not robust for short intervals because

in a time series of feature values, each feature value is strongly correlated with other feature values (those that occur just prior to, or after it) and this problem cannot be solved by reducing the overlap between features. Therefore, the KS statistic is only meaningful statistically when the two intervals chosen include several syllables. In this case too, you can use the KS statistic between syllables as a non-metric estimate.

Interpretation of scores

Scoring similarity between two sounds as described above might work well in some cases, and less well in other cases. It should be used carefully and wisely. The outcome of similarity scoring depends heavily on appropriate scaling of the features to units of median absolute deviation from the average in the 'population'. The next chapter explains how to scale features and when new feature scaling should be considered. A related factor is feature weight: the default assumption is that the five features are equally important. This assumption has not been tested, but empirically, giving an equal weight to the five features works well for scoring song similarity across adult zebra finches. Each feature has different strengths and weaknesses and together, they complement each other. The feature that is most likely to cause you troubles is pitch: pitch is sometimes difficult to calculate, and an unstable pitch estimate might bias the scoring procedure.

Reading about the complexities involved in calculating similarity scores you might wonder about the consequences of improper use of these methods. Compared to the human-observer scoring method, the automated approach has pros and cons. No doubt, the judgment of any human observer is preferred over automated methods. The main difference is that automated methods provide well defined metrics for distances between sounds and can quantify subtle differences. Statistically, however, you should handle automated similarity scores just as you would handle human scores, except that you might consider using parametric methods (if the scores distribution appears to be normal) – but this is not a big issue. If at the end of the day all you care about is whether two groups of animals differ in their sounds – it does not matter how the scores were calculated, under what assumptions, etc. For example, if you use the feature scale of zebra finches on monkey sounds, and find strong differences in similarity scores across two groups of animals using some non-parametric estimate of the scores, the difference is real regardless of the strong biases injected by using a wrong normalization. However, you do not want to use wrong normalization since this might reduce the sensitivity and reliability of scoring method, making it most-likely that significant differences will be found. Overall, in most cases, you will want to use the scoring method that maximizes the difference between your groups. The actual p-value used for threshold is just a yardstick, and it has nothing to do with statistical significance.

Period of repetition & state transitions

Period-of-repetition is somewhat similar to the concept of a song motif in that it is a repeated unit; it is an estimate of the typical interval duration between two consecutive renditions of similar sounds. A song motif, however, may contain repeating sounds, so the two measures may differ (*SAP2011* does not calculate *motif duration* -- only *period*). Earlier we presented the similarity score as a mean of evaluated similarity between two different sounds base on the *Euclidean distance* between their features. The same idea can be applied for finding similar sections within a single interval of sound (typically, a long one): We start by choosing a frame of sound (insert point) at a random location within an interval of sound outlined by the user and observe the features in a 70ms window around it. A 'silence' is not allowed to be an insert point. We first need to depart from the insert point by moving the window forward in time until we encounter sound that is different enough from the original sound to meet a *similarity rejection threshold*. The default value is 75% similarity. We then keep sliding the window until we find a sound that is similar to the insert point to meet a *similarity acceptance threshold*. The default is 95% similarity. To obtain a stable estimate of the typical period, *the algorithm* examines a random sample of 50 random *insert points*, and measures the interval between each *insert point* and the next rendition of a similar sound. The output is the median of those measurements; that is, it is an estimate of the typical *period* of sound repetition. To obtain a good estimate of the time interval between two similar renditions you will need to outline a relatively long sound interval. We cannot tell you exactly how long the interval should be – but it should contain several renditions of syllables, e.g., 10s is reasonable for most bird songs. You can combine several song bouts into one file and then calculate the period. The stability of the measurements should be validated experimentally (e.g., try it on different samples of sounds and make sure that the results are similar). To sum up, *Period is an estimate of the* typical interval between two occurrences of 'the same' sound, starting from a random insert point (anywhere in the outlined interval).

In addition to the period measure, *SAP2011* also calculates 'random period' measure: here instead of progressing from the insert until we find a match, *SA+* makes a random search in the outlined interval, until it finds a matching sound. In the ideal case of fully periodic syntax, e.g., 2,3,4,5,2,3,4,5,2,3,4,5… and given an insert of 5, the odds of obtaining 5 in each random draw is 25%, and the mean random period is 4, which is identical to the period. However, if the sound syntax is 2,2,2,3,3,3,4,4,4,5,5,5,… then the random period will remain 4, but the median estimate of period will be much shorter than the random period.

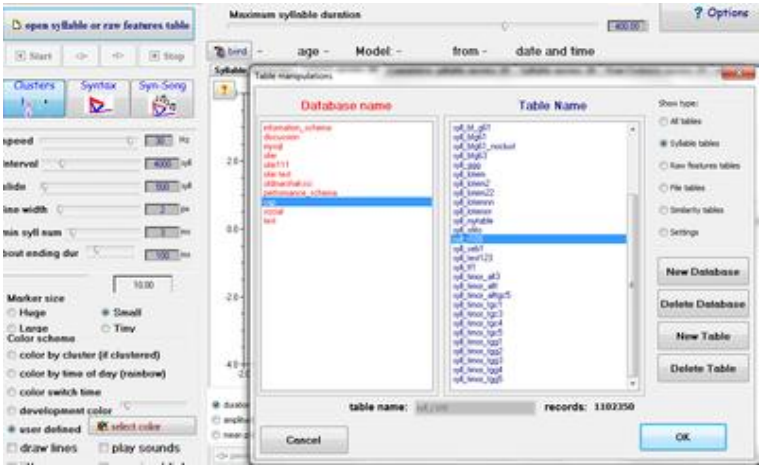
Note: both Period and Random-period are based on random sampling of the data, and will never give you the same answer twice. You must compute them several times and make sure that the estimates give you a stable central tendency.

Chapter 11: Dynamic Vocal Development (DVD) maps

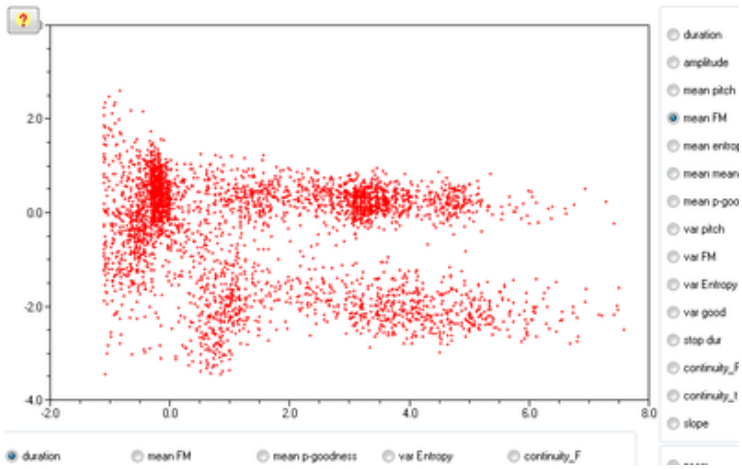
Table of contents:

Introduction to DVD maps

We use the term DVD maps for a variety of graphical methods to view vocal events dynamically at multiple time scales. Like the single frame of video, the sonogram presents a static representation of a dynamic process. The sonogram can only capture short term changes in sound (over time scales of milliseconds), whereas song development is a process occurring over time scales of vocal change ranging from minutes to weeks. The study of that process requires the ability (a) to store large amounts of vocal data, (b) to analyze that data, and (c) to display the results of that analysis as visual representations that highlight key features of the development process as they occur over time. The availability of digital recording and the low cost of digital data storage have now made it possible to meet these requirements. The feature batch module of SAP2011 creates syllable tables including the entire vocal ontogeny of a bird on file. The DVD maps will allow you to see those data for tracing vocal changes over time.
We will use a syllable table of song development of a bird called 109. Please start by downloading it here:
http://soundanalysispro.com/download/download-syllable-table-example.
Then open SAP2011 and select the 'DVD maps' option. Click 'Open syllable or raw features table' and select the table syll_R109:



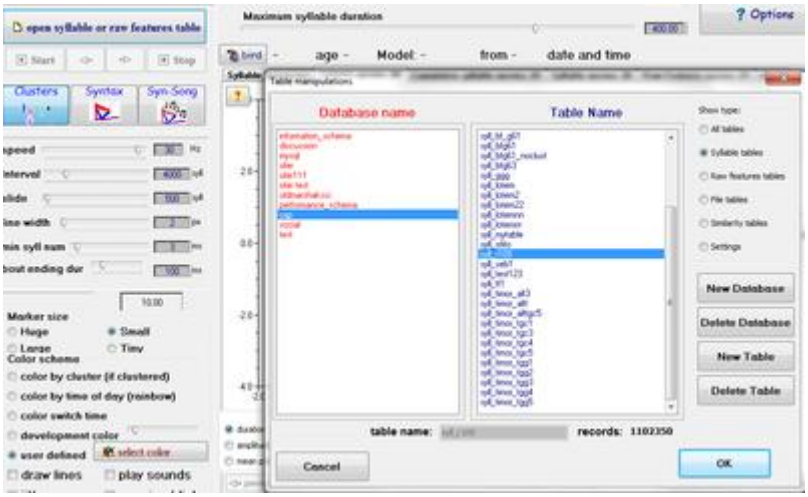
Note that the syll_phrase indicates that this is a syllable table, where each record (line) summarize features of one syllable (duration, mean pitch, etc). Now click start, and you should see a movie of scatter plots of the syllable features distribution in 2D, keep looking how it changes over time.



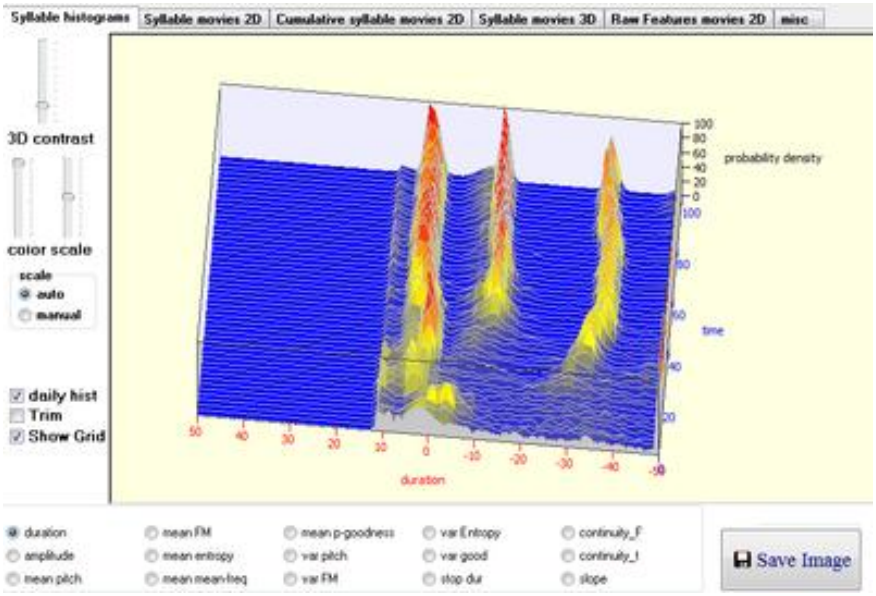
The online analysis of vocalization and the automatic generation of the syllable table is the corner stone of *SAP2011*, and one of the most useful descriptive model of such tables is displaying dynamic vocal development (DVD) maps, showing vocal changes as movies. You should keep in mind, however, that segmentation of sound to syllable units is a double-edge sword: it can uncover song structure if properly used, but it might also inappropriately chop the sound into inappropriate units. This is not only because the criteria for segmenting sounds are not always robust, but mainly because the deep structure of vocalization bout is not trivially related to syllable units. The binary feature files that are automatically created during the live analysis keep record of non segmented (continuous) feature curves. *SAP2011* does not provide a means of explicitly analyzing these data, but you can easily export them to Matlab.

Developmental histograms

One-dimensional DVD-maps display changes in the values of a single feature, capturing the entire table (e.g., the whole song development) in a single image. Petr Janata was the first to suggest this idea, and showed that presenting the evolution of duration histograms during development has traceable structure. The plot shows the distribution of syllable durations during any one day in each raw using gray-scale and the entire matrix span that entire song development. SAP2011 uses a similar approach, except that the gray-scale is replaced by a 3D reconstruction.
Open SAP2011 DVD-map module, Click the '1D developmental histograms' tab and open the table of syll_R109.



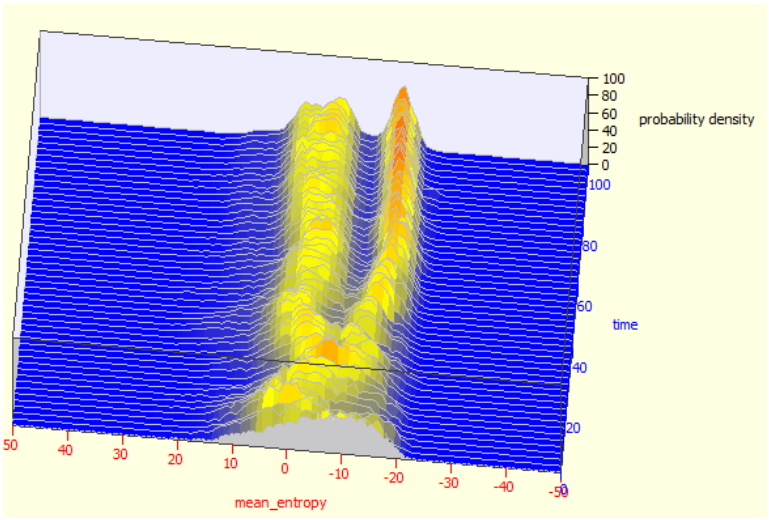
Click the start button, and wait several seconds:



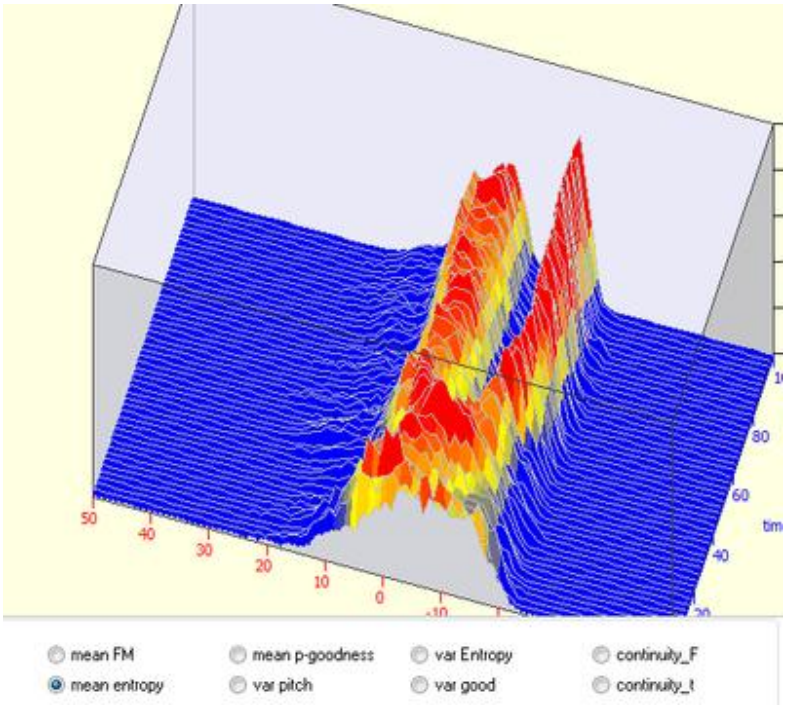
This is a developmental histogram of syllable durations. Each row in the 3D histogram shown below represents a single day of song development, and the height of the mountains represents the abundance (frequency) of a certain duration. The far side of the histogram represents the end of song development, and the 3 peaks stand for the 3 syllable types produced by this bird. As shown, it is very easy to trace the peaks backwards to the point of

origin. The advantage of this representation is that it shows us the entire song development at a glance, based on the entire song development data. It is also making it obvious why is it so important to record and analyze the entire song development data - otherwise, computing such histograms won't be as robust and won't work on a short time scale. Note how one of the peaks 'takes a turn' during early song development (this is the vocal change of time-warping). You can see that the vocal change occurs within a few days, whereas thereafter, duration did not change much. As we shall discuss later on, vocal changes are often like this - they can take a few hours, a few days or several days, and then nothing happens during the next several days. Given this hectic nature of vocal change, DVD-maps give us an easy way of detecting critical moments in song development.

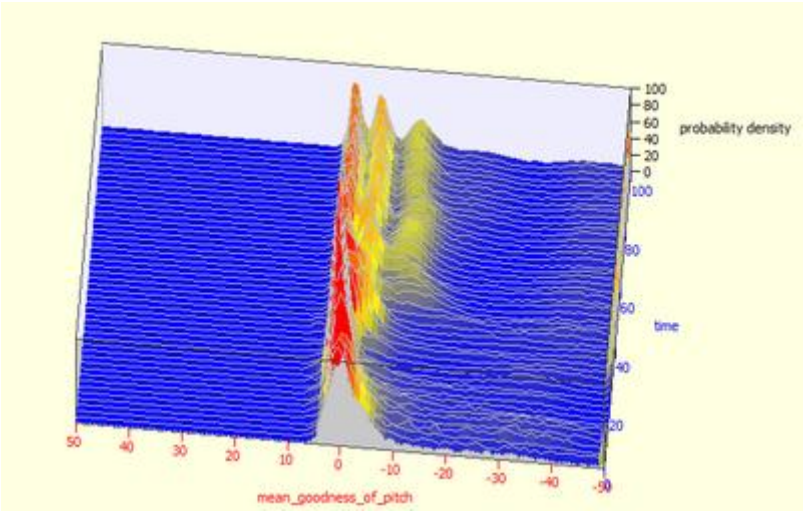
Finally, histograms can be computed for any features, here are a few examples: Wiener entropy developmental histogram shows an early bifurcation:



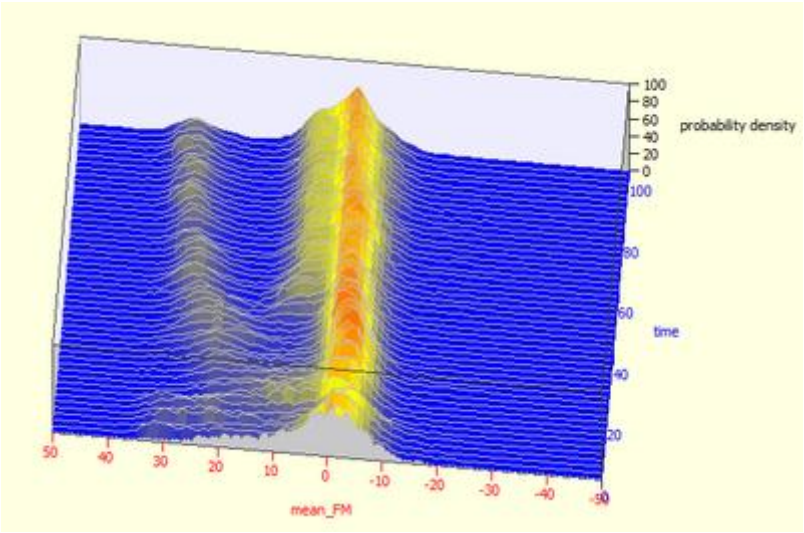
Use the mouse to rotate the image, and the sliders of the left to change display contrast and color scale:



Here is goodness-of-pitch histogram, note that the changes in this feature occur much later in development:



and here is FM-histogram, again showing a very different time course of vocal changes:

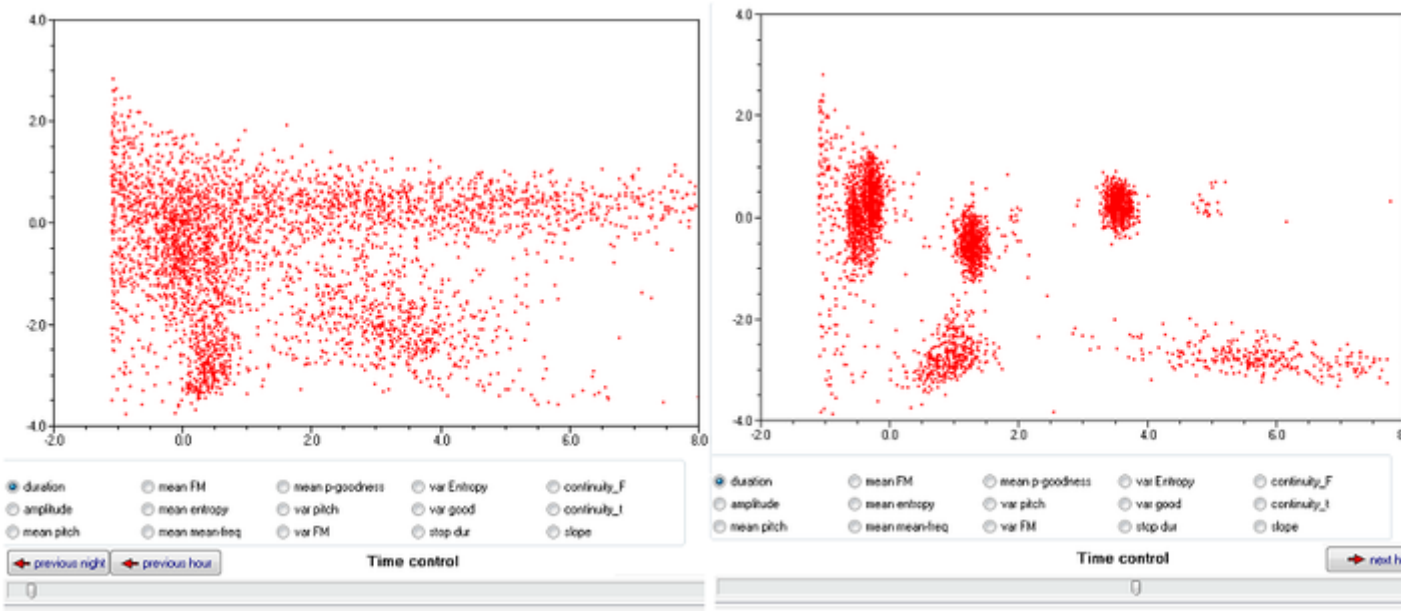


The range can be automatic or manual. When selecting a manual scale the minimum and maximum range is set by the user. The number of bins, however, is fixed and equals 100, so selecting a duration range of 0-500ms will give bins of 5ms. If you choose auto-scale, *SA+* will scale the features based on the normalization of syllable features (see in options -> feature normalization).

Aliasing issues: never let the bin-size approach the encoding accuracy of a feature. The likely problem is with duration. The actual encoding is the window-step taken, which is about 1.4ms by default. Since we use 100 bins, the default definition is about 5ms when the range is set between 0-500ms. Setting it from, say 50-300ms is taking us down to 2.5ms - which is still fine, but going down to, say, 100-200 is taking us down to 1ms - which is below the critical (Nyquist) value - which will result in aliasing: a nasty, unresolvable artifact in the duration histogram.

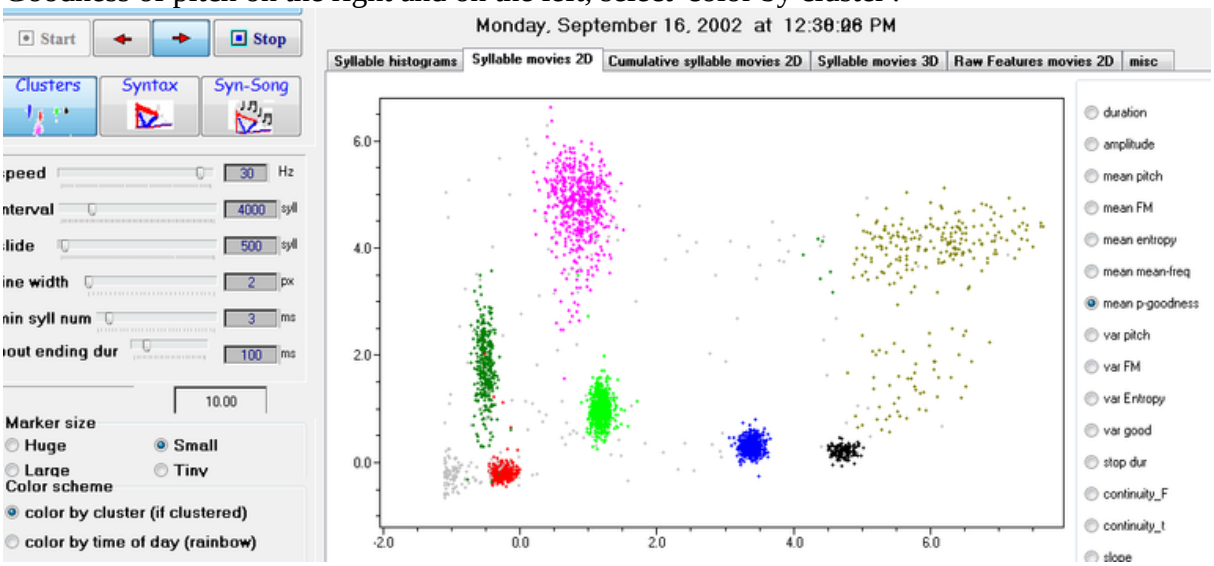
2-D DVD maps

2D DVD maps allow you to observe scatter plots of vocal development (or any other vocal process) as a movie. Start SAP2011 and select DVD maps. Open the table syll_R109, and click start. You should see an evolving movie (here are initial and later snapshots):



Each red dot represents a syllable, the X axis is 'duration' by default, and the Y axis is mean FM. Note that feature units are scaled, with median at 0 and with a spread of a single MAD (median absolute deviation) unit. The beginning of the movie is also the beginning of song development, when songs syllable are relatively unstructured.

Click at the time control (long slider at the bottom), it should turn yellow, and move its thumb to the right, about ¾ of the way. The movie should now look very different as you go.
Below and to the right of the image you can select the features to be observed. Try the Goodness of pitch on the right and on the left, select 'color by cluster':

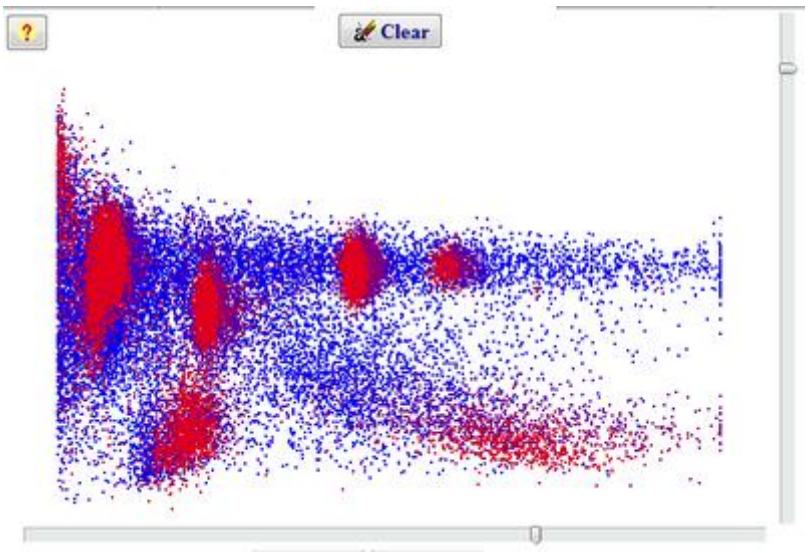


We can see 3 dense clusters, which correspond to the 3 syllable types of the song produced. To see other features of those clusters click on the Y-axis feature choice (you can do this as the movie is playing). Clicking on the pitch shows different projections of the clusters. Moving the mouse into the movie image will show a hand pointer, indicating that you can move the image as desired. On the bottom right you will see a radio-group with 3 choices: rotate, pan and zoom - try them all during the movie play.

At this point, our notion of clusters is informal, but in the next chapter you will learn how to automatically identify the clusters. In this example, we already clustered the data. Looking at DVD maps after clustering is a useful method for validating that the outcome of the clustering is reasonable.

Cummulative DVD maps

Looking at the movie, you can see how clusters (syllable types) emerge and change with development. In order to obtain some solid evidence to this change, it is often useful to compare two 'frames' of the DVD map. To best observe the long-term effect of time on the distribution of syllable features, select 'color by linear time' from the color scheme, move the 'Time control' slider to the beginning of song development and then select the 'cumulative movies' tab. Select duration for the X axis and FM for the Y axis. In the marker size selection, click 'tiny' and then click 'DVD Play/Stop' to start the movie. Color will slowly change from yellow to red. The little slider to the right of the 'color by linear time' control can be used to change this rate. The result should look like the figure below. Note that you can also change colors manually at any time during the movie, as well as moving the 'Time control' slider to different developmental time, either during play or offline.

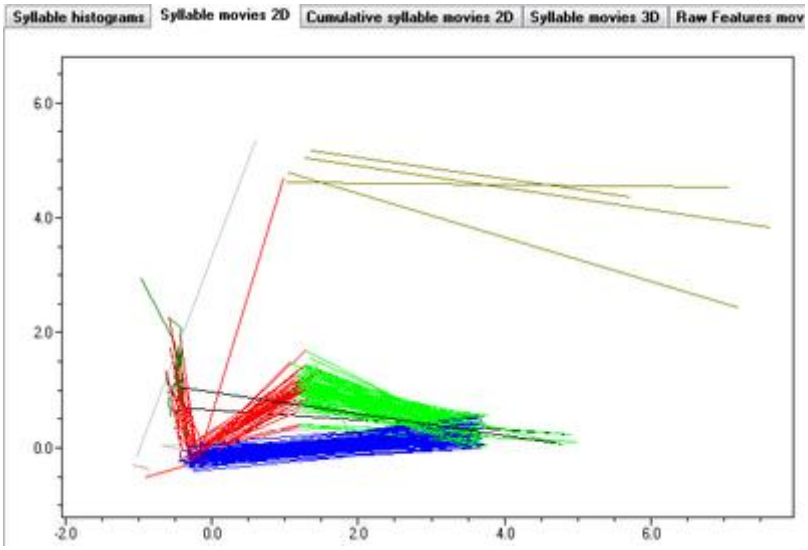


Changes in feature distribution occur also during short time-scales, e.g., after night sleep or during certain hours of the day. Click 'clear' and change the color scheme to 'color by time of day'. *SA+* will now code circadian time by color - red for early morning, orange for late morning, green for noon, blue for the afternoon and black for the evening song. Set the time control to late song development and click 'DVD Play/Stop'. You should be able to see how the duration fluctuates with the time of day. Looking at other features during early development will show even stronger changes - try the Wiener entropy variance. Feel free to switch back and forth between dynamic movies and cumulative movies on the fly.

Syntax DVD maps

The syntax mode

Click on the 'Syllable movies in 2D' tab and select the syntax mode. Note that the interval and slide positions have changed, and that 'Draw lines' is now checked. Open syll_R109 and move the time control slider to the middle of the range. Set color scheme to color by cluster and the Y axis feature (right of the chart) to mean p-goodness. Click start:



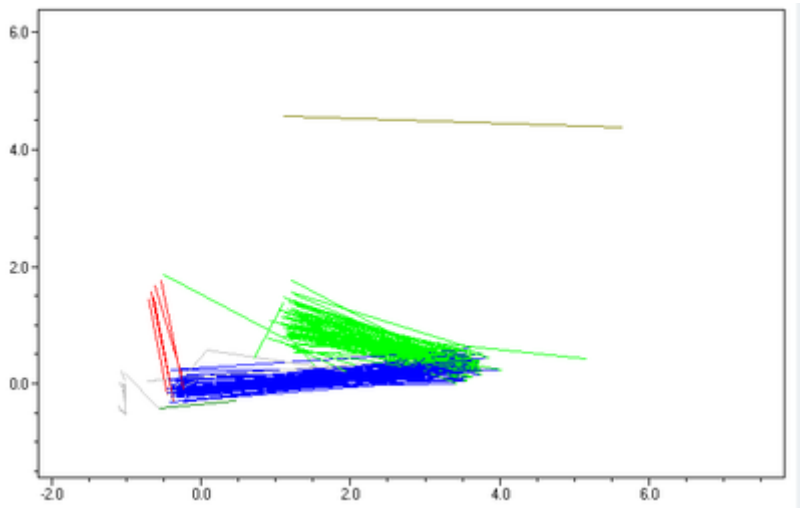
Move the time control to the later stages of song development, select duration for X and pitch for Y axis. Set the color scheme to user defined and select red. Then click 'DVD Play/Stop'. Instead of looking at the syllables as dots, we now look at the trajectories that connect them in time. That is, the order of singing the song syllables is now represented by lines that connect those syllables. When a short duration syllable is followed by a longer duration syllable, *SAP2011* will paint the line red, and when a long syllable is followed by a shorter one, the line is blue. It is immediately apparent by the movie, that the song is stereotyped; however, it is also easy to see drifts in the pattern. As shown, the shape equivalent of this song is a triangle. Each projection (using different features) will give a different view of this syntax patters, obviously, some projections are nicer than others. We can now explore the syntax development of this song. Moving the slider back to the original setting, duration versus FM, shows nice transitions of syntax structure during development as exemplified in the figure to your right. Note how the blue and red 'streams' get separated during development as the third song syllable appears, turning the shape into triangle.

Now say that you want to observe the possible effect of circadian time on song syntax: selecting 'color by time of day' will present the trajectories with circadian color, indicating that for some features (e.g., pitch) evening (blue) and morning (red) trajectories differ during early development.

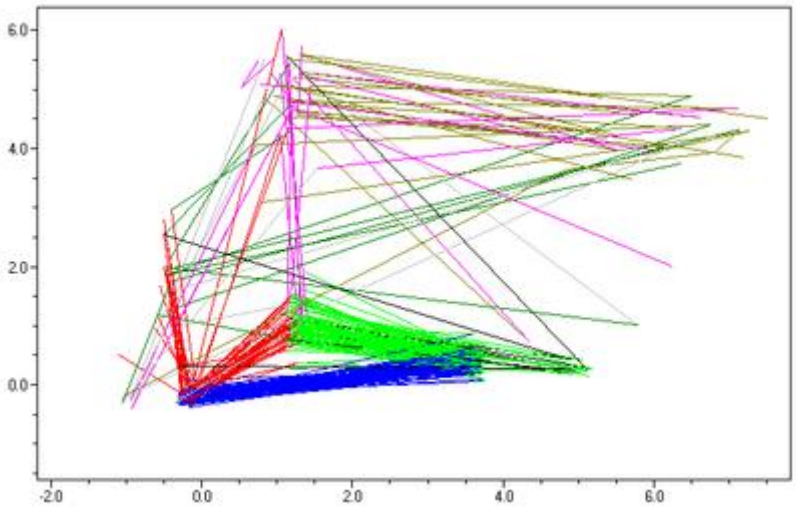
Note that although we call this 'syntax mode' what you actually see is not really syntax, but some combo display capturing both syntax and feature (sort of phonetic) changes. We find this representation particularly useful because there are reasons to believe that the changes of feature structure (within a syllable) and changes of syntax might be linked. For example, a prototype sound can give rise to two different types of sounds (see details in Tchernichovski et al 2001, sound differentiation in situ).

This is a good time to elaborate about the relations between the observed clusters and the actual syllables produced by the bird. The clusters are at best as robust as the segmentation method is, and unfortunately, our current methods of segmentation, based on amplitude and Wiener entropy thresholds are not always robust. The most common type of measurement noise is inconsistency in the segmentation of short attenuations. For example, if the 'real' pattern of syllable is ABC,ABC,ABC… and the pause between B and C is very short, *SAP2011* might sometimes join B and C, to give an additional 'combination cluster' that we shall call D. hence, what we observed as ABC,AD,AD,ABC should actually be ABC,ABC,ABC… This problem, once detected, is not too difficult to solve. For example we can re-segment the data (using the binary files rather than the raw sound files) using a more aggressive criteria for partitioning sounds.

It is often useful to change the bout ending dur parameter -- it determine when to cut the lines that connect the syllables. Setting a conservative threshold of, say, 40ms, looks like this:



indicating that short latency transitions are restricted to blue-green syllable types. And setting it high will show this view:

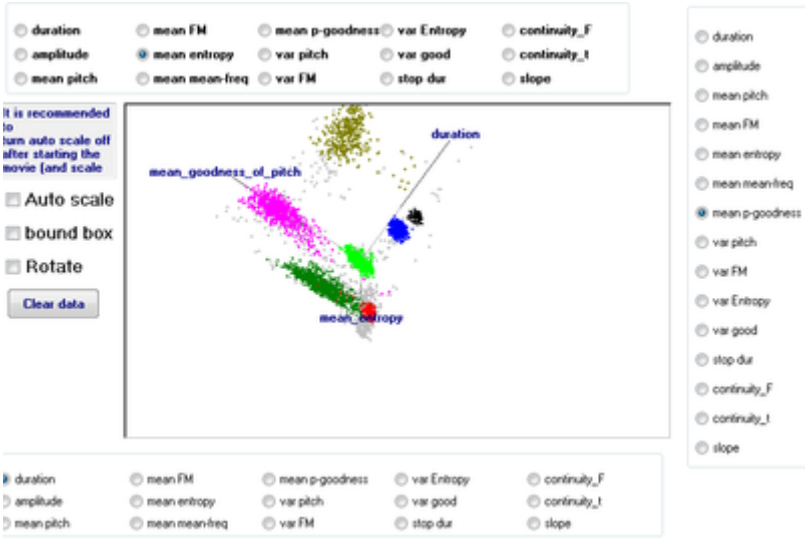


indicating that the transition to call elements usually occurs between the green and red clusters.

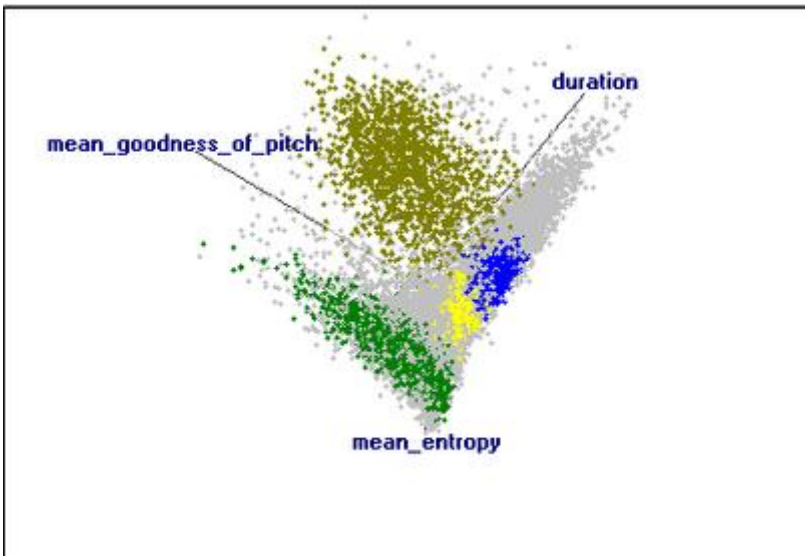
3D DVD maps

The 3D DVD maps are similar to the 2D ones except that you can see three features at a time in 3D and rotate the view.

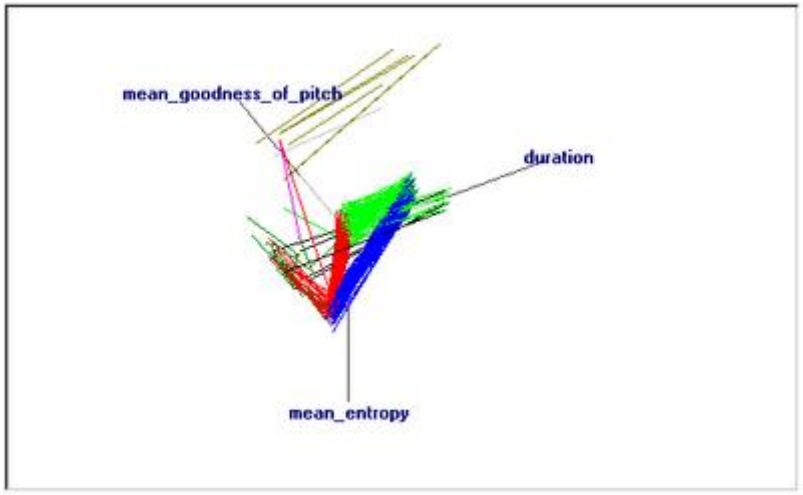
You should often increase the interval size to see better in 3D (e.g., try going up to 5000 or 7000 syllables). Start with Auto scale on, and then once the move is running turn it off to obtain a stable image of the appropriate scale (example is from syll_R109 toward the end of development):



and using different rotations in earlier days can provide a nice image of how those clusters emerge:



you can also look at syntax in 3D:


mean_goodness_of_pitch
duration
mean_entropy

Raw features DVD maps

Under construction

Chapter 12: Clustering Syllables

Table of contents:

Introduction to Clustering

Background & computational approach

In the previous chapter (DVD-maps) we saw an example of song development where syllable-feature distribution is broad during early stages, and then feature distribution becomes clustered. Most clusters (or types) appear to stay, and one can often keep track of each cluster throughout the rest of the movie. This cluster analysis module performs the (difficult) task of automatically classifying syllables into types and tracking each type during time. We use the terms 'type' and 'cluster' interchangeably, but one should keep in mind that a cluster is not a generic type, but a transient entity that is only defined in one animal (during a particular time). More often, the term 'syllable type' refer to species-specific sound, sometimes in relation to perceptual categories or other behaviors. The narrow definition of type has some advantages, however, of being simple, robust and 'analytic'.

At best, the clustering method is as good as the segmentation method is. We have no doubt that both cluster analysis and segmentation methods presented here are sub-optimal. Aylin Cimenser, a Physics PhD student at Columbia University is now in the process of optimizing both methods, and we believe that it should be possible to properly segment and cluster vague sounds that the current technology cannot handle.

The cluster analysis methods used here were implemented by Aylin Cimenser at Columbia University under the supervision of Partha P Mitra and Tim Halperin Hilly. The algorithm is based on the *Nearest Neighbor Hierarchal Clustering (NNC)* method. This is a non-parametric, density-based method (in contrast Gaussian mixture methods and K-means, which are parametric). This works nicely when clusters are reasonably segregated, regardless of their shape, and the back-tracing method is fully automated and self-checking. We cannot claim, however, that this approach always works. Rarely, clusters that can be identified visually are not easily identified. Users should keep in mind that a perfect solution for clustering problems should not be expected. It is no coincidence that there are many alternative cluster analysis methods in the 'market' - there is no generic solution to this problem. Namely, one cannot cluster data without making some strong assumptions about either shape or density of clusters, and those assumptions are sometimes false. For example, NNC does not handle well cases where some dense clusters are too close to sparse clusters.

More significantly, in song development, clustering has to be performed by recursive steps, starting from the end of song development, where identifying clusters is very easy, and concluding at an early stage of song development, when clustering becomes increasingly difficult, and finally impossible. The difference between approaches is not if the procedure will fail - but when.

NNC is a very simple algorithm and we encourage you to understand it:

1. Obtain a sample of a few thousand (say 3,000) syllables produced at a given time and compute features.
2. For each feature (e.g., syllable duration), calculate Euclidean distances between all possible (3000 x 3000) pairs of different syllables. Repeat for all features, scale units and add. This step includes about 50 million multiplications (but only about 1s).
3. Sort syllable pairs according to Euclidean distances in ascending order and discard syllable-pairs with a distance greater than a certain (say 0.01 MAD) threshold.

4. Now the best-best-friends, the syllable pair of shortest distance (nearest-neighbor pair), are establishing the first cluster.
5. Examine the next pair of nearest neighbors and consider the following scenarios:
n **Case A**. If both syllables are new (not clustered yet), define another cluster with this pair.
n **Case B**. If one of the syllables is already a member of a cluster (remember that any one syllable can appear in many pairs!), and its pair is not a member of any cluster, add this pair to that same cluster (that is: a friend of my friend is my friend!).
n **Case C**. If one syllable belongs to one cluster, and the second one to a different cluster - merge the two clusters.
n **Case D**. If both syllables belong to the same cluster, do nothing.
6. Repeat step 5 for all the remaining pairs (with a distance above threshold) according to distance order.

In many cases, you will find that the default threshold suffices to properly cluster the data, but be aware that the choice of threshold is very critical. Setting the threshold too high will join all the data into a single cluster. The threshold is determined empirically, and if not optimal you will experience one of two problems: if the threshold is too conservative, you will see too many clusters and many residuals (not clustered) syllables. If the threshold is too liberal, you will see 'false joining' of distinct clusters. In some cases (the most difficult ones) you will see both false joining and many residuals. To address these cases, you can endow a ***veto power*** to certain features. The meaning of *veto* is: 'do not merge two clusters if the difference between them is too big with respect to a specific feature', so that even if the pair of syllables (in case C) is really tight, *SAP2011* won't join their clusters, if the centers of those clusters are too far away with respect to one feature. For example, you can tell *SA+* to refuse merging two clusters if the duration difference between them is more than 0.5 MAD (which is 36ms in the zebra finch). If this 'distance gap' remains empty of other clusters, those two clusters will never merge.

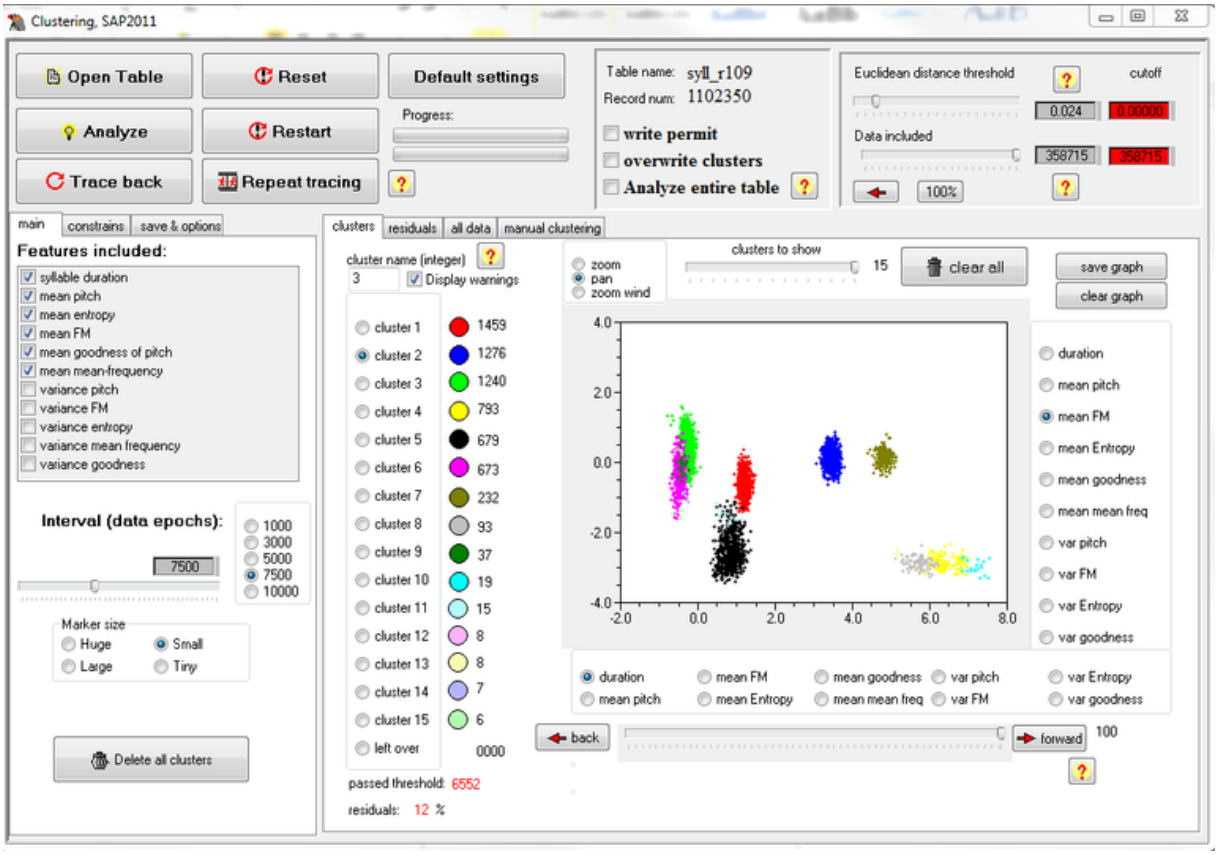
Step by step clustering

Scaling syllable features

Scaling of syllables, based on 'maximum likelihood estimates' is the same as that used for similarity measurements at the levels of frames and intervals.

Example case: Syll_R109

Open the clustering module, open the table syll_R109. To prevent overwriting of existing clusters uncheck 'write permit', and click 'Analyze'. The result should look like this:

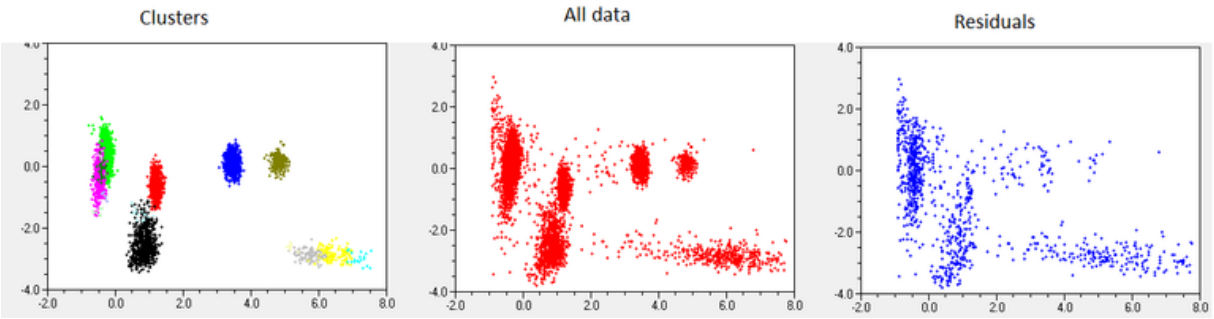


Note that this representation is basically the same as a 2D DVD-map, with a default of duration for the X axis and mean FM for the Y axis. Not all features are used for clustering: by default, *SAP2011*uses syllable duration and the mean values of pitch, Wiener entropy, FM and goodness of pitch. Feature units are scaled to MAD also in the display, as noted, proper scaling is essential for calculating meaningful Euclidean distances across features. For every one animal, you might find biases, but overall, all clusters should live in the neighborhood of 4 MADs and have a mean spread between 1-2 MADs (averaged across features). The colors identify the clusters, but the initial color identity is determined by the population of the cluster (how many members). Therefore, the color is not yet identifying any cluster in the long run (only at the current moment). Shortly, we will discuss the techniques of marking (identifying) a cluster for tracing. The legend on the left allows you to pick a cluster or to view some of its properties. For example, the most abundant cluster is painted red, and you can see near the red legend that this cluster has 606 member syllables. Once you identified a cluster and clicked at the appropriate legend - you should give the cluster an ID. The ID must be an integer number,

type it in the edit box placed on top of the legend. Once you ID a cluster and start tracing it back, it will keep its original color (that is, we uncoupled the abundance from the color).

SAP2011 presents the actual Euclidean distance cutoff of the most distant pair of syllables included in the analysis in addition to the threshold. The 'data included' results show the number of paired syllables that passed that threshold. The upper bound for 3000 syllables is 3000x3000=9 million. The threshold reduces this number to about 50,000 syllable pairs that passed. You can reduce it even further by moving the 'data included' slider to the left and observe the gray display of 'data included' changing as you go. Now click restart and observe the consequence of this action on the cutoff. This technique allows you to quickly test the consequences of changing the cutoff without re-calculating Euclidean distances (which is the time-limiting step). Note that the table of syllable pairs has still a lot of redundancy with 50,000 syllables in pairs that are extracted from no more 3000 different syllables (and often, much less). In fact, looking below the legend will show you that only about 2500 different syllables passed the threshold. A syllable in a 'crowded area of feature space' will participate in many pairs, whereas in a sparse area, a syllable might have no neighbor close enough to join a pair. Also, remember that *SA+* only analyzes the 10 largest clusters. If you want to cluster more then 10 types, you can do so exhaustively as described later. You should be aware that filtering the table (removing clusters) is a non-linear operation with regards to clustering. That is, the results might change abruptly with filtering. In practice, this is more often a plus than a minus, since it can turn an unstable performance into a stable one.

Before we get into the tracing technique, let's explore the different displays that will help you judge how good the clustering is. Click on the 'all data' tab, then click the 'residuals' tab, and move back and forth from cluster to both of those displays. As you can see, most but not all the syllables were clustered.



A careful look at the outcome raises a few questions about the clustering performance.

First, how come that the yellow and green clusters where not joined into a single cluster? The answer to this question becomes clear when looking at different projections of this cluster in feature space. Changing the Y axis to pitch shows that the two clusters overlap in their frequency modulation but not in their pitch.Second, what sounds compose the 27% residuals? Looking at the residuals shows that some belong to 'sparse clouds' that have not been clustered. These 'clouds' are often long and short calls, which are less stereotyped and less abundant in our recording (the lower abundance of calls is, in fact, an artifact of our recording method, which intends to preserve song data and eliminate isolated calls). Other residuals belong to the category of cage noise - these are often characterized by low-pitch and broad distribution of duration, as shown in the example above. Finally, some residuals are left-overs of clusters - these residuals can be reduced by having a more liberal threshold. Similarly, one can cluster a 'sparse cloud' of data by having a more liberal threshold.

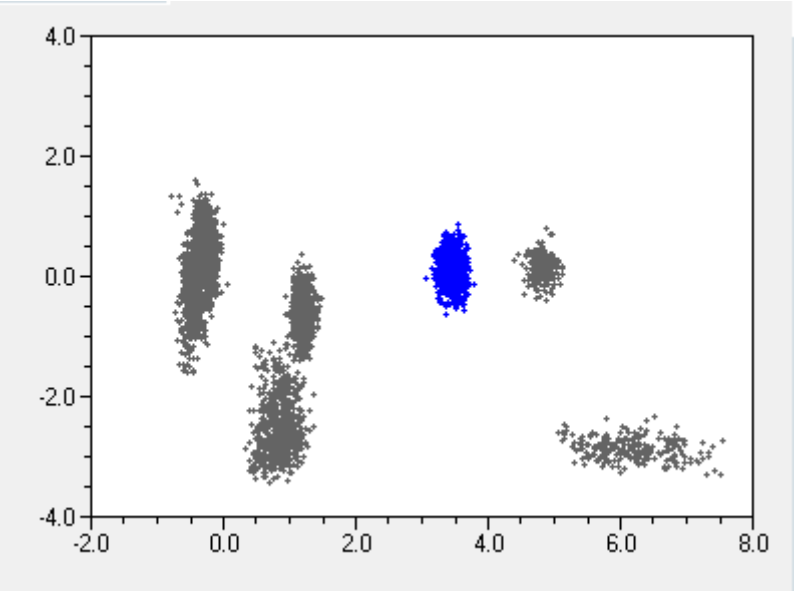
You might ask - how can one decide what should the threshold value be? The answer is that the ideal threshold value is 'cluster dependent'. If a cluster is very close to another cluster, having a too liberal threshold will join them. The point is you do not want to try to cluster all your data at once. Instead, the strategy we implemented is of clustering types one by one. This requires more work, but it gives you the freedom to set appropriate conditions, that works well for this particular type of sound.

We will now start the process of tracing back syllables, but first, let us illustrate some of the problems involved in trace-back. As noted earlier, the major issue is that the nature of the task changes as we step backwards in song development, as we expect clusters to eventually fall apart when we reach the beginning of song development. What we are trying to trace is, in fact, the structure of this 'falling apart' (actually, the 'build up' when forward-tracking) process. During later stages of song development, we will typically observe movement of clusters in feature space. This process is easy to trace since we have a complete recording of ontogeny, and since most features change slowly compared to the size of time-slices we try to bridge across (typically, 3000 syllables occurs in time scales of several minutes to a few hours). Even non-linear vocal changes, such as period-doubling, will rarely cause problems since other features of the syllable will remain stable during this event. During early stages of song development, we often see how clusters merge - since in almost every bird, different syllable types emerge from a smaller number of prototype syllables in the process of 'syllable differentiation'. Detecting the point of transition is a challenging task.

Let's look at two clusters of bird 109, which are shown as fuchsia and green in the figure above. We noted that those clusters are close to each other. They have similar FM but different pitch, and there is also a slight duration difference between them. Move the 'Time control' slider 2/3 to the left and click 'Analyze'. Note, that since we are not back-tracing, *SAP2011* will make no attempt to re-identify clusters, so the colors will change arbitrarily (according to the number of members in each cluster).

Note that although we stepped several weeks, the two images are similar, and we can see that the blue cluster is still there, but stained yellow, and the red one has turned blue (and is somewhat larger). The problem is that the yellow and green clusters have merged - and are both red now. The question is - is this a false merging, or something that the bird did? Looking at the raw data (right panel) shows clearly that the clusters are indeed merged. This example demonstrates some of the difficulties you might encounter while back-tracing - now let's try it.

Pull the Time control slider to the end of song development and click 'Analyze'. We will start with the easiest cluster - say the blue one. Now we need to tell *SAP2011* that this is the cluster we want to trace, and we need to give it a permanent name. This name will appear in the appropriate records of the database table as we do the procedure, unless you uncheck the 'write permit' check box (please do uncheck it!). Since this cluster appeared blue we check the blue radio-button in the legend, and then on top of the legend we type the permanent cluster name. The cluster name must be an integer number, and we suggest starting with 1. Now you should see that the track-back button (top) became enabled. Click it.



Note that the 'Time control' did not take a step back yet - it only identified this cluster in the current slice and (if write permitted), registered it in the table of bird 109 so that each occurrence of this cluster is marked as 1 (the default value for a cluster is 0). Now click track-back once more. Note that the Time control has moved a tiny bit to the left.

Now click 'Repeat tracing back' and you will see that tracing back occurs automatically, step after step, until you click this button again - or - until something bad happens…

Let's try to understand more formally what is happening here. *SAP2011* did clustering and you have chosen a cluster to trace, we will call it the reference cluster. When tracing back, *SA+* does a similar clustering on a slightly earlier time window. The algorithm then computes the centroid of each cluster (that is, the mean duration of syllables in the cluster, the mean mean-pitch of syllables in the cluster and so forth. Then, the centroids of each of those new clusters are compared to the centroid of the reference cluster. The cluster with the most similar centroid to that of the reference cluster is assumed to be an earlier version of that cluster - but this is only if it is similar enough to the reference. The default threshold for this comparison is 0.2 MADs (across all features chosen).

Tracing this cluster should work very well for several weeks of song development, but eventually, it is doomed to fail.
You will need to define the cluster again - based on its location (change the radio-choice in the legend to the appropriate color to re-activate the 'trace-back' button). Then keep tracing back, with some 'playing around' you should be able to trace it back until August 10 or so, which is 3 days after the onset of training.

Try to trace other clusters of bird 109. You will find some of them easy and others more tricky. For example, this yellow cluster will cause frequent troubles by merging with the one below it:

To solve such problems, your first line of defense is decreasing the Euclidean distance threshold, e.g. to 0.01
To check quickly if threshold reduction can solve a problem, click analyze and then click the 'data included' left-arrow followed by 'restart'.

This approach also fails from time to time - but do not give up - reduce threshold to 0.008 to regain hold, back-trace once and try 0.01 again, and then auto-trace until the next failure. By the time you approach the beginning of September, tracing this cluster becomes really difficult.

This experience might (and should) have raised some concerns about the objectivity of this clustering method. Indeed, one would like to be able to set the parameters once, rather then keep playing with them. The reality of cluster analysis, however, is that one often needs to adjust parameters. We suggest you document your adjustments, and also, try it more than once in such difficult cases. In this particular cluster, the problem is that the only good distinguishable feature is pitch - all other features of these two clusters overlap. Trying to distinguish between them can only work for some time, but as the pitch values approach each other, the mission is turning impossible. Furthermore, you pay a toll of high percentage of residuals.

The solution is therefore to also cluster the two clusters together, and consider the time when they are separated as two descendent clusters of a main branch (as in a dendrogram).

To do this, move the time control to the end of song development, return the threshold to 0.015 and in the features included, uncheck the 'mean pitch'. Check the 'write permit' but uncheck the 'overwrite clusters'. Now give the joined cluster a different name (say 2) and click 'analyze' and the clusters will immediately merge. You will see that the two clusters immediately join, since when pitch is not taken into account, other features they have in common prevail.

Correcting errors

It occasionally happens that clustering goes wrong, and the wrong data are written into the table. This does not occur when *SAP2011* warns you about the back-tracing problem, but rather when *SAP2011*does not warn you but visual inspection indicates that clustering went wrong. To correct a false clustering click the forward arrow of the 'time control', change the cluster name to 0, and click trace-back twice.

Looking at the clusters is quite useful for correcting errors:

To see the clusters on the sonograms: open Explore & Score, go to the clusters tab, and open the syllable table. Then open any sound files used for the clustering and click 'view clusters'.

You should be able to see the cluster identity as a colored line above the syllables.

Exhaustive clustering

Exhaustive clustering

As stated earlier, the cluster analysis method included in *SAP2011* is far from perfect, and we hope to incorporate the new approach that Aylin Cimenser is currently developing shortly. In difficult cases, one can still succeed to cluster using the exhaustive clustering approach as following:

· Start from the easiest (most prominent and distinct) clusters and cluster them throughout development.
· Use the mySQL Workbench to split the data into unclustered and clustered tables:

CREATE TABLE bird109_unclustered SELECT * FROM syll_r109 where cluster=0;
CREATE TABLE bird109_clustered SELECT * FROM syll_r109 where cluster>0;
ALTER TABLE bird109_unclustered CHANGE recnum recnum PRIMARY KEY;

· Open *SAP2011* and cluster bird109_unclustered - you might find that this is an easier task now
· Finally, merge the two tables in mySQL Control Center typing

INSERT INTO b109clustered SELECT * FROM b109unclustered

Comment: always make sure that recnum is the primary key and that the table is sorted by recnum. You can always change sorting by editing the table and changing sorting in
Also, you can sort by any field after deleting recnum, and then recreate recnum as the primary key and auto-increment to change sorting order as desired.