**USV Analysis Codes**

Contents

[Script\_ run 1](#_Toc472443445)

[Skeleton 1](#_Toc472443446)

[1. Signal Processing 1](#_Toc472443447)

[1.1. Enframe 1](#_Toc472443448)

[1.2. Framing 2](#_Toc472443449)

[1.3. Pre-processing 2](#_Toc472443450)

[1.4. ReArr 2](#_Toc472443451)

[1.5. Rearrange\_signal 2](#_Toc472443452)

[1.6. TimeAndFormVec 2](#_Toc472443453)

[2. Feature Extraction 2](#_Toc472443454)

[2.1. Exclude\_too\_little\_data 2](#_Toc472443455)

[2.2. Get\_stats 2](#_Toc472443456)

[2.3. getTemporalFeatures 2](#_Toc472443457)

[2.4. Preprocess\_3DFormant 3](#_Toc472443458)

[2.5. Real\_time 3](#_Toc472443459)

[2.6. Feature\_test\_limits 3](#_Toc472443460)

[2.6.1. Check\_length\_Call 3](#_Toc472443461)

[2.6.2. diffOfVector 3](#_Toc472443462)

[2.6.3. maxentropymat 3](#_Toc472443463)

[2.7. Filter Bank 4](#_Toc472443464)

[2.7.1. ERBFilterBank 4](#_Toc472443465)

[2.7.2. Frequency\_continuity 4](#_Toc472443466)

[2.7.3. MakeERBFilters 4](#_Toc472443467)

[2.7.4. Segmentation4Signal\_2 4](#_Toc472443468)

[2.7.5. Syllable\_Detection2 4](#_Toc472443469)

[2.7.6. Syllable\_Detection\_ERB 4](#_Toc472443470)

[2.8. LPC - unused 4](#_Toc472443471)

[3. Classification 4](#_Toc472443472)

[3.1. ANN 4](#_Toc472443473)

[3.1.1. Organize\_for\_ann 4](#_Toc472443474)

[3.1.2. Preprocess4ann 4](#_Toc472443475)

[3.2. Create\_feature\_models\_code 4](#_Toc472443476)

[3.3. DTW\_classification 4](#_Toc472443477)

[3.4. Plot\_gold 4](#_Toc472443478)

[4. Classification models 4](#_Toc472443479)

[5. ROC code analysis 4](#_Toc472443480)

[6. User Interface 4](#_Toc472443481)

[6.1. Usv\_analysis\_ver2 4](#_Toc472443482)

[6.2. Outcome\_per\_signal 4](#_Toc472443483)

[6.3. Supported\_Gui\_functions 4](#_Toc472443484)

[6.3.1. Create\_spec\_mat 4](#_Toc472443485)

[6.3.2. Get\_info\_from\_file\_path 4](#_Toc472443486)

[6.3.3. Spectrogram\_disp 4](#_Toc472443487)

[6.3.4. uipickfiles 5](#_Toc472443488)

[7. Gold standard 5](#_Toc472443489)

[8. Create data base 5](#_Toc472443490)

[8.1. Data\_struct 5](#_Toc472443491)

[8.2. Datastruct2file 5](#_Toc472443492)

[8.3. Dir2txtlog 5](#_Toc472443493)

[9. Classes 5](#_Toc472443494)

[9.1. Added\_info 5](#_Toc472443495)

[9.2. Added\_info4disp 5](#_Toc472443496)

[9.3. bigData\_class 5](#_Toc472443497)

[9.4. LPC\_Formants\_class 5](#_Toc472443498)

[9.5. Parameters\_usv 5](#_Toc472443499)

[9.6. ROC\_class 5](#_Toc472443500)

[9.7. Temporal\_features 5](#_Toc472443501)

[9.8. Import\_txt2param 5](#_Toc472443502)

[10. Code for documentation 5](#_Toc472443503)

[11. Documentation 5](#_Toc472443504)

[12. Xls\_tables 5](#_Toc472443505)

[Processes 6](#_Toc472443506)

[Segmentation process 6](#_Toc472443507)

[Creating Gold Classification 7](#_Toc472443508)

# Script\_ run

Script\_run is a Matlab script that calls the main functions for USV analysis. The script lets the user choose which process to begin:

1. Run analysis for new signals – begins analysis for new signals by running the **Usv\_analysis\_ver2** GUI.
2. Run viewer for previously processed signals – opens **Outcome\_per\_signal** GUI to allow viewing the signal and analysis results.
3. Analyze classification algorithm for previously processed signals – performs "Leave on out" process for the classification algorithm on a pre-classified data set.
4. Write parameters – runs **getTemporalFeatures** to extract the features and write them to an excel

# Skeleton

The skeleton function activates all the main processes of the analysis. The function receives a **Parameters\_usv** class variable and carries out processes according to the flags in it.

Step 1 - create output directory according to results folder indicated in parameters.

Step 2 – perform segmentation (if segmentation flag is "True")

Pre-Step3 – not in use

Step 3 - Create models (not in use)

Step 4 – Classification (not updated)

Step 5 – ROC (not in use)

Step 6 – Temporal Features extraction (not updated)

Step 7 – Open **Outcome\_per\_signal** GUI

# Signal Processing

## Enframe

Enframe (Copyright Mike Brookes 1997) splits signal up into overlapping frames. The function's inputs are the signal, the length of the frames and the overlap between frames.

## Framing

Framing receives the signal and sends it to Enframe for framing. In addition the function performs windowing by use of a hamming window.

## Pre-processing

Pre-processing filters the given signal with a pre-emphasis filter and DC removal. The function is not enabled in the current version.

## ReArr

Currently unused

## Rearrange\_signal

Rearrange\_signal receives the time frames in which frequencies were found and rearranges them into syllables. The function's outputs are the Start and End times of each syllable.

## TimeAndFormVec

Currently unused

# Feature Extraction

## Exclude\_too\_little\_data

Currently unused

## Get\_stats

Get\_stats receives a vector of frequencies and calculates the descriptive statistics:

* Start frequency
* End frequency
* Mean
* STD
* Min
* Max
* Median
* Mode

## getTemporalFeatures

getTemporalFeatures calculates descriptive statistics of the duration of the calls, the intervals between the calls and bursts of calls. The function receives a **Parameters\_usv** class and navigates to the linked output folder. The segmentation information is loaded from the output folder. In addition, getTemporalFeatures calls get\_stats to receive spectral information. All the statistics are written to two separate excel files:

Features.xlsx – containing all the temporal statistics. This file is shared for all information from getTemporalFeatures. Further information is in **xls\_tables** section.

Freq\_info.xlsx – containing all the spectral statistics. This file is created for each pup recording separately. Further information is in **xls\_tables** section.

## Preprocess\_3DFormant

The function receives the frequencies of a call from the segmentation and rearranges them. The processing includes:

* Re- arranging the frequencies to the minimum Energy. This is performed by calling the **maxentropymat** function.
* Erasing all the singulars frequency data.
* Executing a median filter to exclude the outliers.

## Real\_time

Real\_time adjusts the times attached to the detected formants. The function's has two modes:

0 – The input times are relative to the beginning of the specific track and need to be adjusted to be relative to the beginning of the recording.

1 - The input times are relative to the beginning of the recording and need to be adjusted to be relative to the beginning of the specific track.

The function uses the log file that was automatically created during the recording. The log contains the time the recording started and the time each track started.

## Feature\_test\_limits

### Check\_length\_Call

Check\_length\_Call enforces minimum call length of 0.016 [sec] by deleting shorter calls; and a minimum interval between calls 0.006 [sec] by unifying close calls.

### diffOfVector

aaaaa

### maxentropymat

aaaaa

## Filter Bank

### ERBFilterBank

ERBFilterBank receives the parameters for each filter in the filter bank and filters the input signal. The input is a signal frame of the track.

### Frequency\_continuity

Frequency\_continuity forces continuity of harmonics in syllables. The input is a matrix of all the time frames in which harmonies were detected in the track. The condition for a harmony is at least four sequential time frames where a harmony was found.

### MakeERBFilters

MakeERBFilters (©1993, Apple Computer, Inc.) computes the filter coefficients for a bank of Gammatone filters. These filters were defined by Patterson and Holdworth for simulating the cochlea. The inputs are the sampling frequency, the number of wanted filters and the lowest frequency needed.

### Segmentation4Signal\_2

Segmentation4Signal\_2 begins the segmentation process for each track. The function loads the signal and creates a log file for the segmentation output. Framing parameters and threshold parameters are defined in this function and sent to Syllable\_Detection2. Once frequencies are detected the function sends the information to Rearrange\_signal and saves the final outputs – matrix of start and end times of each syllable and the frequencies of each syllable, to the defined output folder.

### Syllable\_Detection2

In Syllable\_Detection2 the signal is framed and a filter bank is created by sending relevant parameters to **MakeERBFilters**. Each frame is sent, along with the filer bank to **Syllable\_Detection\_ERB** for frequency detection. Frames in which a frequency was found are sent to Syllable\_Detection\_ERB again with a modified filter bank to try to detect harmonies. After each frame was sent to Syllable\_Detection\_ERB the **Frequency\_continuity** function is activated to remove un-continuous harmonies.

### Syllable\_Detection\_ERB

Syllable\_Detection\_ERB receives a frame and the parameters of a filter bank and sends them to **ERBFilterBank**. The output of ERBFilterBank is a matrix of outputs for each of the filters in the bank. Syllable\_Detection\_ERB calculates the energy of each output and searches for peaks. If there are under 20 peaks that surpass 20% of the full energy in the frame, then the frame is classifies as a silent frame. Otherwise, the peaks that surpass 90% of the full energy are detected as the filters in which a frequency component exists and the corresponding frequencies are sent back to Syllable\_Detection\_ERB.

## LPC – unused

Currently unused

# Classification

## ANN

### Organize\_for\_ann

jjj

### Preprocess4ann

jjj

## Create\_feature\_models\_code

Currently unused

## DTW\_classification

Currently unused

## Plot\_gold

jjj

# Classification models

# ROC code analysis

# User Interface

## Usv\_analysis\_ver2

## Outcome\_per\_signal

## Supported\_Gui\_functions

### Create\_spec\_mat

### Get\_info\_from\_file\_path

### Spectrogram\_disp

### uipickfiles

# Gold standard

# Create data base

## Data\_struct

## Datastruct2file

## Dir2txtlog

# Classes

## Added\_info

## Added\_info4disp

## bigData\_class

## LPC\_Formants\_class

## Parameters\_usv

## ROC\_class

## Temporal\_features

## Import\_txt2param

# Code for documentation

# Documentation

# Xls\_tables

# Processes

## Segmentation process

Segmentation4Signal\_2.m

Parameters

Pre-Processing.m

(not in use)

Rearrange\_signal.m

Check\_length\_Call.m

Real\_time.m

Syllable\_Detection2.m

1

2

3

4

5

Framing.m

MakeERBFilters.m

Syllable\_Detection\_ERB.m

Frequency\_continuity.m

1

2

3

4

ERBFilterBankm

The segmentation process is initiated in **Skeleton** by running Segmentation4Signal\_2 for each track separately.

**Segmentation4Signal\_2** loads the track and creates a log file. Frame length, overlap and thresholds for the filter bank algorithm are defined in Segmentation4Signal\_2 and sent to Syllable\_Detection2 along with the signal.

In **Syllable\_Detection2** the signal is framed according to the defined frame length and overlap using the **Framing** function. A Filter bank is created by **MakeERBFilters**, consisting of 90 filters between 35kHz-125kHz. Each frame is sent to Syllable\_Detection\_ERB.

In **Syllable\_Detection\_ERB** the frame is sent to **ERBFilterBank** to be filtered by each of the filters in the bank. The energy of each filter's output is calculated and energy peaks are detected. Peaks that surpass the defined threshold are detected and the corresponding frequency is returned to Syllable\_Detection2.

If a frequency is detected, Syllable\_Detection2 searches for a harmony frequency by adjusting the thresholds and the filter bank (using again MakeERBFilters) and sending the frame again to Syllable\_Detection\_ERB. Continuity of the detected harmonies is enforced in **Frequency\_continuity**.

Syllable\_Detection2 returns all the frequencies that were detected along with the end time and start time of the corresponding time frames to Segmentation4Signal\_2. The information is organized into separate syllables by the **Rearrange\_signal** function.

The next step in Segmentation4Signal\_2 is to send the detected syllables to **Check\_length\_Call** to enforce minimum call length and minimum interval length. The start-end information is written to the log file.

At this point the start and end times are relative to the beginning of the specific track. In **Real\_time** the time is calculated relative to the beginning of the 10min recording. After the start and end times are adjusted they are saved along with the detected frequencies in the output directory folder.

## Creating Gold Classification

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Recording Date** | **Name** | **Session** | **Recording Number** | **Start Point** | **End Point** | **Structure** | **Grade [%]** | **Classification date** |
|
| **22.11.15** | **Newborn1** | **Session1** | **T0000004** | **0.5166** | **0.5658** | **Freq Steps** | **95** | **25Jul16** |

Manual classification is performed in the **Outcome\_per\_signal** GUI (see fig no). The user clicks the syllable in the graph, chooses a class from the drop-down menu and enters the certainty grade of the classification. Once the user presses the save button the following information is saved to an excel file:

Example:

Using the excel file, a **bigData\_class** variable can be created which contains the manual classification and can be entered to the classification functions. The variable is created by the **Data\_struct** function which receives the parameters class of the segmented syllables and the path to the excel file.