**Biogents Wingbeat Analyzer CC**

Author: Michael Weber, michael.weber@biogents.com

Posted: 2/4/24

Copyright and License:

Biogents Wingbeat Analyzer  
© 2014 by Michael Weber, onVector Technology  
© 2019 by Biogents AG  
is licensed under Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International.  
To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/4.0/> or open license.txt

# Overview

The Biogents Wingbeat Analyzer is a Python-based interactive analysis program for mosquito wingbeat recordings stored as WAV files (one channel, mono). WAV files can be of any length and sampling rates; and are re-sampled during reading to all be the same frequency (default: 8 kHz). Recordings can be grouped either by storing them in different subdirectories or by using one of several clustering methods.

Various methods are provided for feature extraction and clustering. Results can be exported in CSV format for further analysis.

A selection of charts, graphs and tables is provided for data visualization and tabulation of results. If at least two groups are provided, a machine learning algorithm (XGBoost) can be used to train a ML model for species or sex classification; the model can be saved and used to analyze unknown recordings. The user can choose and compare the effects of the feature extraction methods.

The code was developed and optimized (graphics, multiprocessing) on MacOS using Python 3 and a stack of scientific libraries (see next section) using PyCharm Professional. It also runs on Windows but graphics and table may look different and the Python interpreter and/or libraries may throw errors especially related multiprocessing and a feature for sound playback.

# Installation

It was verified that the code runs on MacOS Sonoma 14.2.1 with the current Anaconda distribution and the versions of the scientific libraries in Anaconda. However, no warranty is made that the code is compatible with other OS versions, Python distributions and library versions.

Step 1: Install PyCharm

Step 2: Install Anaconda: <https://www.anaconda.com/download>

Step 3: Create a conda virtual environment using Python 3.9  
 Note: newer versions of Python have not been tested.

Step 4: Install the following libraries:

* matplotlib
* pyqt
* numpy
* pandas
* PyWavelets
* OpenCV  
  scikit-learn
* xgboost
* librosa
* peakutils (\*)
* playsound (\*) (\*\*)

(\*) pypi channel (\*\*) may not function depending on OS/version.

# Program execution and basic navigation

The Python script file is “WingbeatAnalyzer.py” in the src folder.

The script is launched using the PyCharm Run command (in Windows, a shortcut using the Python executable could be used). There will be several notices in the console (including deprecation warnings depending on library versions which can be ignored).

After the script has launched successfully, the main window and menu bar are displayed, and a pop-up window will ask for the WAV directory to select data.

Following data selection, recordings are read in and processed. If there is a lot of recordings, this can take several minutes! The progress is displayed in bottom right of the status bar.

When the read is complete, the main window will show three rows of charts and two tables on the right (one for groups, the other one for the selected recording).

Chart display:

* Top chart row: Graphs displaying group data, color-coded by group.
* Middle chart: Spectrograms for each recording in the sequence they were read in.
* Bottom chart row: Graphs displaying data for the currently selected recording.

Table display:

* Top table: Statistics for the groups
* Bottom table: Parameters for the selected recording.

Selecting a recording for display:

* Click on the spectrogram chart
* Click on a point in a scatter plot (handy to figure out what’s going on with outliers)
* Press the left or right cursor key to advance to the previous or next recording

Deleting a recording:

* Press the Del(ete) key to delete the currently selected recording: the WAV file is removed from the respective directory, results will be recalculated and re-displayed, and recording display will advance to the next recording.

# Menu Overview and Analysis Functions

**File Menu** > Load WAV directory…: Browse to a directory containing WAV files. All WAV files in the subdirectory tree under this directory will be loaded. If the directory has subdirectories, the WAV files will be grouped accordingly. In the scatter plots and tables, the groups are displayed with different colors. The file menu also has several choices for scaling recordings as they are being read, and for saving processing results.

**Clean Menu**: Several methods to remove files with poor signal quality based on certain criteria. Use with caution:

* the WAV file(s) will be deleted from their respective subdirectory!
* the criteria area are simple thresholds - a file with a low amplitude may still contain good data.

Therefore, when experimenting, keep the original data in a separate directory.

**Feature Menu**: The choices here allow features to be extracted which are used to generate graphs and perform clustering and machine learning. A good starting point is the default (MFCC). Bernoulli RBM and PCA are not well tested might bomb.

**Classifier Menu**: If there is more than one group of recordings, the menu items to train one of the classifiers become enabled. XGBoost has consistently produced good results. The model can be stored and used subsequently for classification on newly loaded data.

Note: make sure the training sets are balanced, i.e. the groups to be trained have approximately the same number of recordings. Else metrics such as accuracy may look good but unknown recordings are likely to be misclassified, especially if they belong to the minority class.

**Cluster Menu**: Select a clustering algorithm to find clusters of recordings that “belong together” using the currently selected features. This is donw for all the currently displayed recordings, ignoring the original group (if any). Again, the results can be misleading if there is an imbalance between recordings, for example a lot more female mosquitoes than male mosquitoes. DBScan, in particular, can crash the code depending on parameters and data.

If a model is loaded, there is a choice to classify the recordings based on the model.

**View Menu**: The choice here configures the second graph from the left in the top row of the window. These graphs are useful for looking for trends in the data (based on the selected features) and for differences between groups. The t-SNE graph is particularly informative: if there are clear clusters in the data, they will emerge in this graph (for example, male and female mosquitoes of the same species). However, the absence of clusters in t-SNE does not mean that a trained model has low accuracy.

Note: the higher the number of recordings, the longer the t-SNE calculations will take; for several thousand datapoints it could be half an hour or more.

# Tuning

The script has a number of global parameters which are parameters for tuning algorithms and displays. For example, the parameter “sr\_librosa” sets the sample frequency used across the board when WAV files are read in. These are found starting at line 4040 in the script.

Two algorithms for estimating the fundamental frequency f1 are provided in the script:

* Autocorrelation
* “Harmonic Analysis” (finding and including overtones)

Both are calculated when a recording is processed; the one chosen for display and further calculation is programmatically selected in lines 2303-2307. The determination is not easy because (depending on the experimental conditions) wingbeat recordings can be short (a few 10s of ms), long (100’s of milliseconds or seconds), show periods of silence or large variations of amplitude, changes in frequency, a missing fundamental frequency, or no good flight tone at all.

The most robust algorithm for general use is “Harmonic Analysis”; Autocorrelation is useful if the flight tone persists for the most of the recording duration but may incorrectly produce twice the fundamental frequency as its result.