# Original Software Publication (OSP) Template

# Title / name of your software

Software for objective comparison of vocal acoustic features over weeks of audio recording: KLFromRecordingDays

# Authors / main developers

Ken Soderstrom and Ali Alalawi

Department of Pharmacology and Toxicology

Brody School of Medicine

East Carolina University

Greenville, NC 27834

Corresponding author: Ken Soderstrom, soderstromk@ecu.edu

# Abstract.

This program allows measurement of Kullback-Leibler (KL) distances between 2D probability distributions of vocal acoustic features. The software has been used to compare \*.wav file recordings made by Sound Analysis Recorder 2011 of songbird vocalizations pre- and post-drug and surgical manipulations. Recordings from individual animals in \*.wav format are first organized into subdirectories by recording day and then segmented into individual syllables uttered and the acoustic features of these syllables using Sound Analysis Pro 2011 (SAP). KLFromRecordingDays uses syllable acoustic feature data output by SAP to a MySQL table to generate and compare “template” (typically pre-treatment) and “target” (typically post-treatment) probability distributions. These distributions are a series of virtual 2D plots of the duration of each syllable (as x-axis) to each of 13 other acoustic features measured by SAP for that syllable (as y-axes). Differences between “template” and “target” probability distributions for each acoustic feature are determined by calculating KL distance, a measure of divergence of the target 2D distribution pattern from that of the template. KL distances and the mean KL distance across all acoustic features are calculated for each recording day and output to an Excel spreadsheet. Resulting data for individual subjects may then be pooled across treatment groups and graphically summarized and used for statistical comparisons. Because SAP-generated MySQL files are accessed directly, data limits associated with spreadsheet output are avoided, and the totality of vocal output over weeks may be objectively analyzed all at once. The software has been useful for measuring drug effects on songbird vocalizations and assessing recovery from damage to regions of vocal motor cortex.

# Keywords:

Vocal Learning; Acoustic Analysis; Kullback-Leibler Distance

# Abbreviations:

Sound Analysis Recorder 2011, SAR; Sound Analysis Pro 2011, SAP

# Free text.

# *Description of your software in maximum 5 pages for first Original Software Publication – see suggested format;*

# *1 Introduction*

Speech disorders afflict approximately 4 % of children by the time they reach six years of age, and are often secondary to other more severe developmental disorders including intractable childhood epilepsy and autism [1]. These disorders are typically treated with speech-language physical therapy. Effective pharmacological interventions remain unavailable, possibly due to a lack of appropriate pre-clinical animal models. Very few animals learn a form of vocal communication in a manner similar to humans. Songbirds are among this small group of vocal learners. Thus, we have begun to develop a songbird, the zebra finch, as a laboratory animal model to evaluate drugs for effects to improve vocal learning and recovery from CNS damage. This model depends upon development of methods to objectively compare quality of vocalizations pre- and post-treatment over the course of weeks. Excellent software exists for recording and analyzing acoustic features of animal vocalizations (e.g. Sound Analysis Recorder and Sound Analysis Pro 2011 [2], and good methods for comparing phonetic quality of vocalizations by KL distance measures have been described [3]. The goal in developing the current software was to integrate these approaches.

# *2 Problems and Background*

**2.1 Problems solved**

The software described solves the problem of applying KL distance methods of quantifying phonology (developed by [3]) to the literally millions of acoustic feature measures calculated by SAP that are derived from every syllable recorded from animals over a multi-week experimental period. The analysis is accomplished entirely objectively, from audio recording to final output of KL distance measures.

**2.2 Background**

Sound Analysis Pro (SAP) is powerful, free, open source software developed to study songbird vocal development [2]. It includes modules that allow both recording and management of recordings in \*.wav format, as well as analysis software that automatically segments sounds into the individual syllables uttered and analyzes their spectral structure through calculation of 14 acoustic features (syllable duration, mean amplitude, mean pitch, mean FM, mean AM2, mean entropy, mean goodness of pitch, mean mean frequency, pitch variance, FM variance, entropy variance, goodness of pitch variance, mean frequency variance, AM variance). SAP uses the open source MySQL relational database system to manage data. SAP provides for the export of these data in either native MySQL table or Excel spreadsheet formats. Others have developed an excellent approach to use SAP spreadsheet output to statistically compare changes in songbird phonology over time using KL distance measures (Wu et al., 2008). These methods have already been successfully applied to studying effects of surgical manipulation of brain regions important to vocal learning and motor production [4–6]. However, these prior studies relied upon manual analysis of SAP spreadsheet output derived from selected recordings made during a subset of pre- and post-operative periods. Our goal was to develop a completely objective system that automates analysis of the totality of vocal production over the entire experimental period.

Our initial efforts to develop an analysis system based upon Excel spreadsheet output from SAP failed due to the 65536 row number limit of the \*.xls file format. As acoustic measures from each syllable analyzed by SAP occupies a data table row, and it is not uncommon to have hundreds of thousands of syllables analyzed over the course of a typical experiment (in our case experiments last 20 days) we needed to establish KL distance measurement software that directly accesses the MySQL database system employed by SAP. The size of MySQL database tables are limited only by the memory resources of the computer used. Given the open source nature of Python and availability of PyMySQL and openpyxl packages to interface with MySQL and Excel respectively, we chose this programming language for development of our application (version 3.5). The software was developed to run on a 64-bit Windows system.

# *3 Software Framework*

**3.1 Software Architecture**

The software is controlled by a graphical user interface derived from the Tkinter Python package (see Fig. 1). This interface collects information from the user including: The directory to store the resulting Excel spreadsheet of KL distance measures; the SAP-produced syllable table to analyze; the recording day(s) to use as the “template” (remaining days will be used as the “target”); the number of divisions to divide each axis of virtual 2D plots into (explained below); the minimum number of syllables required before \*.wav files are used for analysis; the smoothing factor to employ for empty cells (explained below); the syllable interval cutoff (ms) to estimate if a syllable is part of a motif and; whether or not to modify the SAP syllable table with estimated syllable type and whether or not to exclude non-motif syllables from the analysis. A flow diagram of the program is presented as Figure 2.

The GUI function (GetTableNames) uses PyMySQL to query the SAP database for a list of syllable tables. This function takes advantage of the SAP naming convention that prefixes syllable tables with “syll\_”. This list is used to populate the table selection list box of the GUI. Once a syllable table is selected for analysis, a function is called by the GUI class (PopulateDayValues) to use PyMySQL to query, calculate and return a list of all of the recording days represented in the selected syllable table. This list is used to populate the selection box from which the user designates which day(s) to use as the “template”. Template days are typically the pre-treatment or pre-manipulation baseline recordings. Each day listed in the selection box that is not designated for inclusion in the “template” is used as a “target” day for comparison to the template and calculation of KL distance values.

Once information is entered, the user activates the “Run KL” button on the interface. This calls the “RunKL” function of the GUI object that checks for table and template entries, returning control to the GUI in case of errors. If there are not table or template selection omissions, “RunKL” creates a dictionary of values set by the user through the GUI and passes this dictionary to the main calculation function “mainKL”.

“MainKL” then calls “ModifyTableWithDay” that adds a new column to the syllable database selected and enters recording day information for each syllable in the table. If the user has elected to analyze only motif-type syllables (Filter Out Non-Motif Syllables) or to calculate the syllable type but use all for the analysis (Calculate Motif-Type but Use All) a function is called to calculate pre- and post-syllable intervals for each syllable and add this information to the syllable table (Syll\_Duration). Functions are then called to create new Template and Target MySQL tables that contain syllable information from the corresponding recording days (MakeTemplateTable and MakeTargetTable functions). The Target table is then used to create a series of sub-tables for each Target recording day (CreateTargetDayTables). Each of these target day-specific tables will be compared to the Template table and KL distances generated for each acoustic parameter.

Once the template and series of recording day-specific target MySQL tables have been created, a for loop is entered for each target day. A second, nested, for loop is entered that iterates over each of the 13 acoustic parameters that will be used as y-axes in 2D arrays (listed above, excluding syllable duration that is always used as the x-axis). The values for each acoustic parameter calculated by SAP for each syllable are accessed from template and target day tables using the “TableParameters” function that is passed the template or target day table name, and the name of the acoustic parameter to access. The function then uses PyMySQL to query the table for the parameter values for each syllable and stores these values as a list in a dictionary keyed by the parameter name. “TableParameters” returns this parameter name-keyed dictionary. Four of these “TableParameters” dictionaries are ultimately produced: two each for the template and target day tables; one for the x-axis of a 2D plot; the other for the y-axis. Parameter-keyed dictionary data for template and target day 2D plots are then passed to a class object to calculate KL distance for that parameter: “GenerateKL”.

GenerateKL creates a class object for each acoustic parameter. It is initialized with the four “TableParameter” dictionaries described above, the axes divisions value, and the smoothing factor. From “TableParameter” data, the range between minimum and maximum values across template and target day values are determined for both x-axis (syllable duration) and y-axis (the currently iterated parameter). These minimum and maximum values are then passed with either template or target parameter values and the axes division value to the Python numpy package function, “histogram2d”. Histogram2d uses the ranges established by minimum and maximum values for each axis and divides them into the number of equal width bins indicated by the axes divisions value obtained from the GUI. Histogram2d uses these equal width bins to create a virtual 2D array of cells. The default axes division value = 15, the number of these cells = 152= 225. Histogram2d then parses each syllable from the parameter value data into the 2D cell corresponding to the appropriate syllable duration and parameter value. The sum of syllables falling within each cell of the virtual 2D array are returned as an incidence array by histogram2D. 2D incidence arrays of syllables are generated for both the template and target parameter values. As not all cells in the 2D array are likely to have at least one syllable assigned to it, and the next step of generating a probability distribution uses cell values as a divisor, a smoothing factor is added to each cell to avoid division by zero. This value is user selectable via the GUI and has a default value of 10-6: Larger values predictably decrease KL measures. Following smoothing, template and target probability arrays are generated by dividing each cell value by the sum of all cells. The sum of all cells of the resulting array = 1. Thus, each cell represents the fractional probability that a template or target syllable will be assigned to it. Once template and target probability arrays have been generated, they are passed to the KLCalc function within the GenerateKL object. This function employs the KL distance calculation equation described by [3]. Note that output of the KLCalc function (snippet below) is equivalent to that of the scipy.stats.entropy function.

The KLCalc function compares divergence of the 2D pattern of the target probability array from that of the template. Higher values represent greater divergence. The function returns the KL distance calculated between template and target distributions for the parameter evaluated. This value is assigned to the GenerateKL object and is available to the mainKL process through the object’s “Get\_KLDistance” method. The KL value is assigned to a dictionary keyed by the parameter name, and the loop moves to the next parameter. Once KL distance values for all parameters have been calculated added to the dictionary, KL distance measures for that recording day are complete, and the day’s dictionary is added to a higher-level results dictionary keyed by recording day. Execution then iterates to the next recording day. Once data for all recording days have been processed, the complete results dictionary keyed by recording day is passed to a function to write results to an Excel spreadsheet: “WriteXL”.

The “WriteXL” function uses the openpyxl package that enables manipulation of Excel spreadsheet objects through Python. The working directory is changed to that specified in the GUI, a new Excel workbook is created and KL distance results for all 13 acoustic parameters are written by row for each target day. Information about the animal recorded, GUI settings and numbers of syllables analyzed for template and target days are also recorded (see Fig. 2). The spreadsheet file is named by the animal ID, the day(s) used for the template and number of target days and saved in the working directory.

**3.2 Software Functionalities**

The software provides a metric (KL distance) of how spectral features of an animal’s vocalizations change over the course of days. It stores these values in an Excel spreadsheet in a directory specified by the user. These measures are useful for studying the phonetic effects of drugs, surgeries or other manipulations over a series of days or weeks.

**3.2.1 User-controlled variables**

The software is controlled by a GUI that collects information from users as described in the Architecture section (3.1) above. In addition to the table to analyze and recording days to use for the template, users can specify the number of bins or divisions that x- and y-axes will be separated into. These bins are used to form cells of 2D probability arrays to which each syllable is assigned based upon syllable duration (x-axis) and the particular parameter being iterated (in the mainKL function, y-axis). The greater the number of axes divisions, the higher the degree of divergence between template and target probability arrays is likely to be, which will tend to produce greater KL distance values. As the number of cells approaches one, KL distances approach zero. Thus, this parameter should be optimized for different experiments that produce different axes ranges and numbers of syllables analyzed. Using zebra finches as an animal model, we and others [6] have found that the default value of 15 axes divisions produces good sensitivity to manipulations of motor cortex, while maintaining consistent pretreatment baseline values.

The GUI also allows specification of the minimum number of syllables in a recording to accept for analysis. Recordings with few syllables are typically not song bouts, but calls, cage bangs, wing flutters or other extraneous sounds. We have found that the default setting of a minimum of five syllables per recording eliminates many non-song bout recordings, without excluding those containing complete motifs.

Motifs are a series of syllables produced by songbirds in a stereotyped order. Motif syllables are typically produced with short periods of silence separating them. Thus, syllables that are produced as part of a motif can usually be reliably estimated and distinguished from calls, introductory notes and other noises using a motif interval cutoff. This parameter can be controlled by the user, and has a default of 35 msec. Individual animals vary in the maximum interval between syllables and so this parameter should be optimized for each subject. Using the motif interval cutoff, users can elect to include only syllables estimated to be part of a motif for analysis. Alternatively, all syllables may be used for analysis, in which case every syllable analyzed by SAP will be used for calculation of KL distance values, including calls and introductory notes. Note that selection to use all syllables avoids triggering of functions to estimate motif type and modify syllable tables with the information – increasing analysis speed. To satisfy cases where users may wish to use all syllables for KL distance analyses, but still have motif-type predicted and added to syllable tables, the “Calculate Motif-Type but Use All” option is provided on the GUI (Fig. 1).

**3.3 Sample Code Snippet**

Code snippet 1 presents the class object (GenerateKL) that calculates KLdistance for a particular acoustic parameter for a single recording day. The operation of this class is described in detail in the Architecture section (3.1) above.



# *4 Implementation and Empirical Results*

**4.1 Implementation Details**

SAP should be installed prior to KLFromRecordingDays. This will also ensure that MySQL is installed. SAP is available from: <http://soundanalysispro.com/> (last accessed 3/20/2017). The Sound Analysis Pro Recorder (SAR) package is also available at this same website. KLFromRecordingDays and ParseSAPRecorderWavs are available as executables generated by the Python package cx\_freeze on the GitHub repository <https://github.com/soderstromk/KLFromRecordingDays.git> (last accessed 3/20/2017). To install, create subdirectories for each program, access the cx\_freeze subdirectory and unpack the zip files within to the respective subdirectory. Both executables should run on any 64-bit Windows computer. Source code is included in the zip files.

Sequential steps for using KLFromRecordingDays are illustrated in a flow diagram as Figure 3. To accommodate the recording system file format used in our laboratory we first process recordings generated by SAR with a utility that renames \*.wav files according to our conventional format: \*-##-#####.wav, where \* = animal ID, ## = recording day, ##### = \*.wav file number. This utility software is called “ParseSAPRecorderWavs” and is included with instructions for use as part of the distribution package (see Fig. 4). KLFromRecordingDays is dependent upon \*.wav files in this file format to read animal ID and recording day information.

**4.2 Empirical Results**

Depending upon the size of the MySQL table being processed, KLFromRecordingDays can take several hours to compete an analysis. Progress is indicated in red font at the base of the GUI, which will change to “Done” upon completion. Typical KLFromRecordingDays output in Excel spreadsheet format is illustrated in Figure 5.

# *5 Illustrative Examples*

# Results of an experiment employing songbirds and investigating effects of drug treatments on recovery of song following partial lesion of the pre-vocal motor cortical region, HVC are summarized in Figure 6. KLFromRecordingDays-generated spreadsheet data were pooled for the number of individuals indicated and graphically summarized using GraphPad Prism software <https://www.graphpad.com/> (last accessed 3/20/2017). Statistical comparisons were made using SigmaStat 3.1 run on a Windows XP emulation.

# *6 Conclusions*

Development of KLFromRecordingDays allows us to employ the entirety of the massive amount of syllable data produced by SAP for objective measurement of phonetic change over time. Its availability has allowed us to use zebra finches to study effects of drugs on recovery following damage to vocal motor brain regions, and should increase the value of songbirds as a preclinical model. The software is currently set-up for day-by-day analyses and should be suitable as-is for typical experiments based upon phonetic change over days. For adaption to different experimental designs, source code is available on the repository and modification using Python should be straight-forward.

# *\*Acknowledgements*

# This work was financially supported by GW Pharmaceuticals

# *References*

# [1] L.D. Shriberg, J.B. Tomblin, J.L. McSweeny, J. Speech Lang. Hear. Res. JSLHR 42 (1999) 1461–1481.

# [2] O. Tchernichovski, F. Nottebohm, C. Ho, B. Pesaran, null Mitra, Anim. Behav. 59 (2000) 1167–1176.

# [3] W. Wu, J.A. Thompson, R. Bertram, F. Johnson, J. Neurosci. Methods 174 (2008) 147–154.

# [4] A. Daou, F. Johnson, W. Wu, R. Bertram, J. Neurosci. Methods 210 (2012) 147–160.

# [5] J.A. Thompson, M.J. Basista, W. Wu, R. Bertram, F. Johnson, J. Neurosci. Off. J. Soc. Neurosci. 31 (2011) 322–330.

# [6] J.A. Thompson, W. Wu, R. Bertram, F. Johnson, J. Neurosci. Off. J. Soc. Neurosci. 27 (2007) 12308–12320.

# B- Required Metadata

# B1 Current executable software version

# *Ancillary data table required for sub version of the executable software: (x.1, x.2 etc.) kindly replace examples in right column with the correct information about your executables, and leave the left columns as they are*

# *Table 1 – Software metadata*

|  |  |  |
| --- | --- | --- |
| **Nr** | **(executable) Software metadata description** | ***Please fill in this column*** |
| S1 | Current software version | *1.1* |
| S2 | Permanent link to executables of this version | [*KLFromRecordingDays*](https://github.com/soderstromk/KLFromRecordingDays)*/*[*cx\_freeze*](https://github.com/soderstromk/KLFromRecordingDays/tree/master/cx_freeze)*/****KLFromRecordingDays cx\_freeze.zip*** |
| S3 | Legal Software License | *MIT* |
| S4 | Computing platform / Operating System | *Microsoft Windows, 64-bit* |
| S5 | Installation requirements & dependencies | *Requires Sound Analysis Pro 2011,* [*http://soundanalysispro.com/*](http://soundanalysispro.com/)  *The utility ParseSAPRecorderWavs is very helpful for use,* [*KLFromRecordingDays*](https://github.com/soderstromk/KLFromRecordingDays)*/*[*cx\_freeze*](https://github.com/soderstromk/KLFromRecordingDays/tree/master/cx_freeze)*/****ParseSAPRecorderWavs.zip*** |
| S6 | If available Link to user manual - if formally published include a reference to the publication in the reference list | *Documentation and manual are incorporated to the manuscript* |
| S6 | Support email for questions | *soderstromk@ecu.edu* |

# B2 Current code version

# *Ancillary data table required for subversion of the codebase. Kindly replace examples in right column with the correct information about your current code, and leave the left columns as they are*

# *Table 2 – Code metadata*

|  |  |  |
| --- | --- | --- |
| **Nr** | **Code metadata description** | ***Please fill in this column*** |
| C1 | Current Code version | *1.1* |
| C2 | Permanent link to code / repository used of this code version | *https://github.com/soderstromk/KLFromRecordingDays.git* |
| C3 | Legal Code License | *MIT* |
| C4 | Code Versioning system used | *git* |
| C5 | Software Code Language used | *python* |
| C6 | Compilation requirements, Operating environments & dependencies | *Requires Microsoft Windows, 64-bit* |
| C7 | If available Link to developer documentation / manual | *Documentation and manual are incorporated to the manuscript* |
| C8 | Support email for questions | *soderstromk@ecu.edu* |