



Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

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



Ken Soderstrom*, Ali Alalawi

Department of Pharmacology and Toxicology, Brody School of Medicine, East Carolina University, Greenville, NC 27834, United States

ARTICLE INFO

Article history:

Received 27 June 2017

Received in revised form 11 September 2017

Accepted 9 October 2017

Keywords:

Vocal learning

Acoustic analysis

Kullback–Leibler Distance

Phonology

ABSTRACT

KLFromRecordingDays allows measurement of Kullback–Leibler (KL) distances between 2D probability distributions of vocal acoustic features. Greater KL distance measures reflect increased phonological divergence across the vocalizations compared. 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 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. It may be useful in studies employing other species, and as part of speech therapies tracking progress in producing distinct speech sounds in isolation.

© 2017 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

Software metadata

| | |
|---|--|
| Current software version | 1.1 |
| Permanent link to executables of this version | https://github.com/soderstromk/KLFromRecordingDays/tree/master/cx_freeze/KLFromRecordingDays |
| Legal Software License | MIT |
| Computing platform / Operating System | Microsoft Windows, 64-bit |
| Installation requirements & dependencies | Requires Sound Analysis Pro 2011, http://soundanalysispro.com/ The utility ParseSAPRecorderWavs is very helpful if Sound Analysis Recorder is used to generate *.wav files, https://github.com/soderstromk/KLFromRecordingDays/tree/master/cx_freeze/ParseSAPRecorderWavs Documentation and manual are incorporated to the manuscript. Installation instructions are included in the README.md file on the GitHub repository, https://github.com/soderstromk/KLFromRecordingDays/blob/master/README.md |
| If available Link to user manual-if formally published include a reference to the publication in the reference list | soderstromk@ecu.edu |
| Support email for questions | |

* Corresponding author.

E-mail address: soderstromk@ecu.edu (K. Soderstrom).

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

2011

<http://dx.doi.org/10.1016/j.softx.2017.10.003>2352-7110/© 2017 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

Code metadata

| | |
|---|---|
| Current Code version | 1.1 |
| Permanent link to code / repository used of this code version | https://github.com/ElsevierSoftwareX/SOFTX-D-17-00050 |
| Legal Code License | MIT |
| Code Versioning system used | git |
| Software Code Language used | Python 3.5 |
| Compilation requirements, Operating environments & dependencies | Requires Microsoft Windows, 64-bit |
| If available Link to developer documentation / manual | Documentation and manual are incorporated to the manuscript |
| Support email for questions | soderstromk@ecu.edu |

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 [SAP, [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 in a manner allowing efficient analysis of the totality of vocalizations produced over experiments lasting several weeks.

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 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 analyses 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 [3,4]. 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].

In using songbirds as a preclinical animal model to screen drugs for effects on vocalizations, our initial intent was to apply the excellent SongSeq software already developed and made freely available by [4] (https://www.math.fsu.edu/~bertram/software/birdsong/JNM_12/, last accessed 9/5/2017). This software calculates KL distances in a manner similar to the KLFromRecording-Days program that we have developed. In addition, it features a powerful method to distinguish individual syllables and analyze the degree to which they are produced in a consistent sequence within songbird vocalizations. We continue to use this sequence analysis feature of SongSeq in our project.

In the case of SongSeq's KL distance measurement capability, two issues made the feature impractical for our work. The first issue is related to SongSeq's production of KL distance measures one probability distribution at a time. That is, when using syllable duration as an x-axis, only one of the remaining 13 acoustic parameters may be used as the y-axis to generate a distribution that is compared across days. Additional parameters require separate, sequential analyses that, due to the scale of our project, were time-consuming and created problems with organizing and summarizing results. The second and more significant problem we encountered in applying SongSeq to generate KL distance measures is attributable to its use of SAP-generated spreadsheets for data input. These spreadsheets are of the older *.xls file format limited to a maximum of 65,536 rows. As acoustic measures from each syllable processed by SAP occupies a spreadsheet row, and it is not uncommon to have hundreds of thousands of syllables produced during a single day of recording, in order to objectively analyze data from every syllable uttered 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 (version 3.5) for development of our application. 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

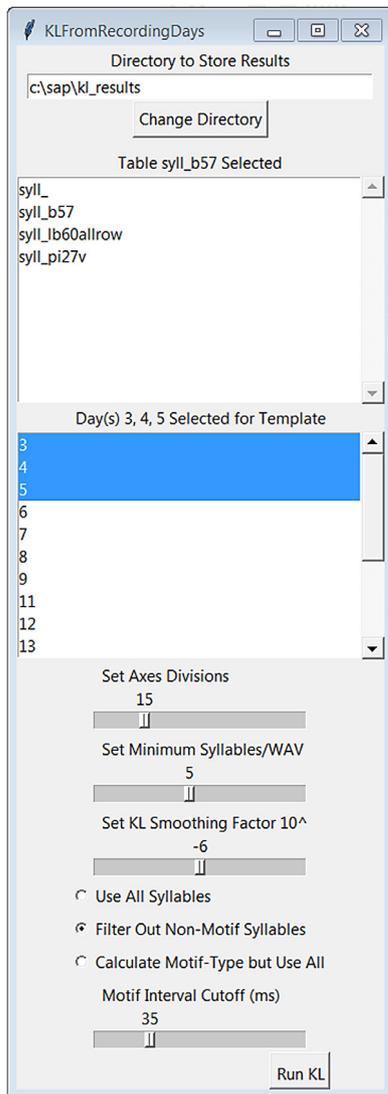


Fig. 1. The KLFromRecordingDays GUI. The Tkinter-derived window allows selection of directory to use for storage of Excel format output; the SAP-generated MySQL syllable table to analyze; and days to use for as a template for comparison to target recording days. Parameters to control aspects of the analysis are available in the lower half of the window and are explained in detail in Section 3.2.1 User Controlled Variables.

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 Fig. 2.

The program uses PyMySQL to query the SAP database for a list of syllable tables. Once a syllable table is selected for analysis, a list of all of the recording days represented in the selected syllable table is created. This list is presented in a 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 initiates a check for table and template entries, returning control to the GUI in case of errors. If there

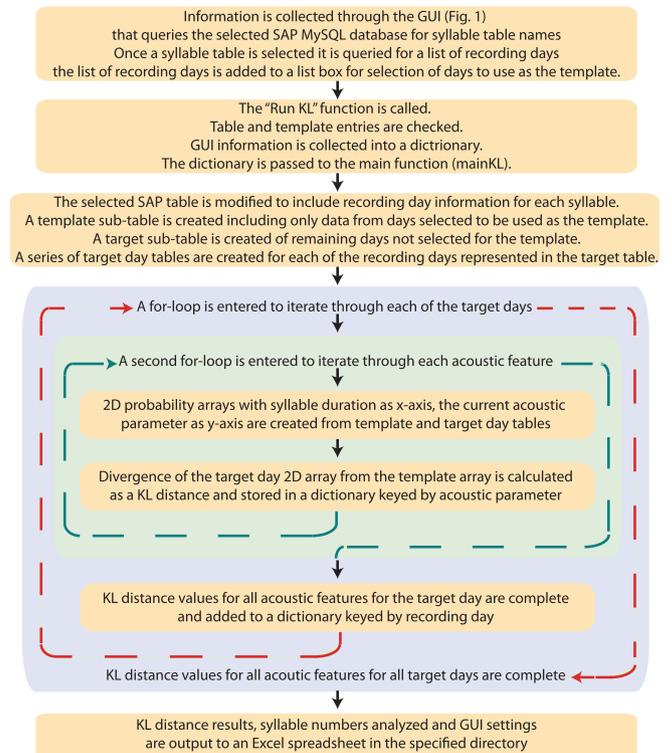


Fig. 2. Flow diagram illustrates the architecture and execution sequence of KLFromRecordingDays.

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 acoustic 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 equal width bins to create a virtual 2D array of cells. The default axes division value = 15, the default number of these cells = $15^2 = 225$. 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. A user selectable smoothing factor (default value of 10^{-6}) is added to each cell to avoid division by zero caused by empty cells. 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, and this 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 KL distance 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 6/27/2017). 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/>

Code Snippet 1: Class to Generate KL Distance Measures

```

# Encapsulated class that generates probability tables for Template and a Target day and uses
# these 2D arrays for KL distance measures for an acoustic parameter
class GenerateKL:
    def __init__(self, TempX, TempY, TargetX, TargetY, axediv, SmoothFactor):

        # Establishes min and max values for Template and Target
        # These are used to define the range of the axes values that will be divided into
        # axediv number of bins
        # Axes bins will be used to generate a 2D array of cells = axediv^2
        # For default axediv = 15, the number of 2D array cells = 225
        maxy = max(max(TempY),max(TargetY))
        miny = min(min(TempY),min(TargetY))
        maxx = max(max(TempX),max(TargetX))
        minx = min(min(TempX),min(TargetX))

        # The numpy procedure histogram2d is used to generate Template and Target incidence
        # distributions
        # Each incidence represents a syllable that falls into a cell of the 2D array
        self.TemplateArray, TempXEdge, TempYEdge = numpy.histogram2d(TempX, TempY, bins = \
            axediv, range = [[minx, maxx], [miny, maxy]],normed=False) # axediv x axediv array
        # w/counts of sylls
        self.TargetArray, TargetXEdge, TargetYEdge = numpy.histogram2d(TargetX, TargetY, \
            axediv,range = [[minx, maxx], [miny, maxy]],normed=False)

        # To each cell of the 2D Template and Target incidence distributions is added
        # a smoothing factor.
        # The magnitude of the smoothing factor can be controlled by the user through the GUI.
        # Default = 10^-6
        # A smoothing factor is necessary as the KL distance equation uses probability
        # distribution factors as denominators
        self.TemplateArray = [(x + SmoothFactor) for x in self.Get_TemplateArray()]
        self.TargetArray = [(x + SmoothFactor) for x in self.Get_TargetArray()]

        # Incidence distributions are converted to probability distributions where the sum
        # of all cells of the 2D distribution = 1
        self.TemplateProbs = [x / sum(sum(self.Get_TemplateArray())) for x in \
            self.Get_TemplateArray()]
        self.TargetProbs = [x / sum(sum(self.Get_TargetArray())) for x in \
            self.Get_TargetArray()]

        #Probability distributions are then passed to a function that calculates KL distance
        self.KL = self.KLCalc(self.TemplateProbs, self.TargetProbs, axediv)

# Get function passes the TemplateArray from the GenerateKL object
def Get_TemplateArray(self):
    return self.TemplateArray

# Get function passes the TargetArray from the GenerateKL object
def Get_TargetArray(self):
    return self.TargetArray

# Get function passes the KL distance measure from the GenerateKL object
def Get_KLDistance(self):
    return self.KL

# This function uses Template and Target arrays to calculate KL distance
# This gives results identical to the scipy.stats.entropy function
# This measure was first applied to evaluating differences in acoustic features by
# Wu W, Thompson JA, Bertram R, Johnson F. J Neurosci Methods. 2008 Sep 15;174(1):147-54
# http://www.sciencedirect.com/science/article/pii/S0165027008003907
def KLCalc(self, TemplateArray, TargetArray, AxesDiv):
    import math
    SumMN = 0
    for i in range(AxesDiv):
        for j in range(AxesDiv):
            q_1 = TemplateArray[i][j]
            q_k = TargetArray[i][j]
            qratio = q_1 / q_k

```

soderstrom/KLFromRecordingDays.git (last accessed 6/27/2017). To install, follow instructions in the README.md file. Both executables should run on any 64-bit Windows computer. Source code is also available within the GitHub repositories.

Sequential steps for using KLFromRecordingDays are illustrated in a flow diagram as Fig. 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

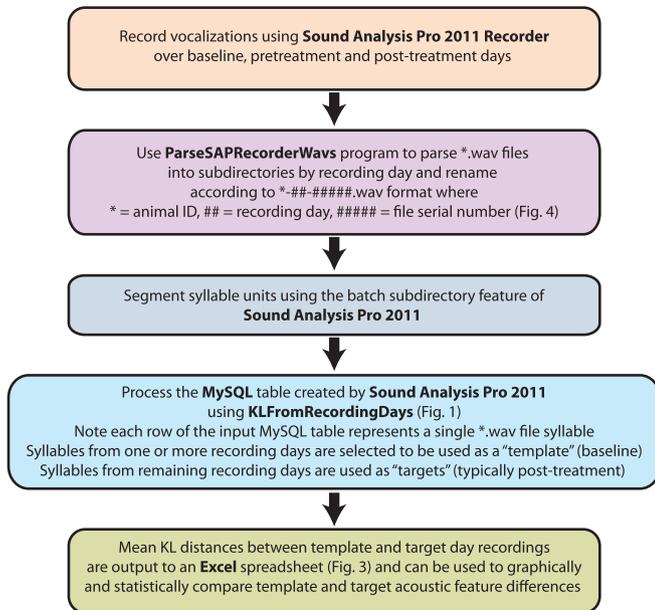


Fig. 3. Summary of analysis steps. Recordings are made over several days with Sound Analysis Pro 2011 recorder. These recordings in *.wav file format are processed into a series of subdirectories according to recording day and renamed according to the indicated file name format. The batch subdirectory feature of Sound Analysis Pro 2011 is then used to segment each syllable recorded and measure a series of acoustic features described in Section 2.2 Background. These acoustic feature measures for each syllable are stored in a MySQL syllable table that is then accessed by KLFromRecordingDays and template and a series of daily target tables are created. Each acoustic parameter for each target day is used to create a 2D probability plot that is used to compare to a comparable plot generated for the template. KL distance, the degree to which template and target 2D arrays diverge are calculated and stored in an Excel spreadsheet.

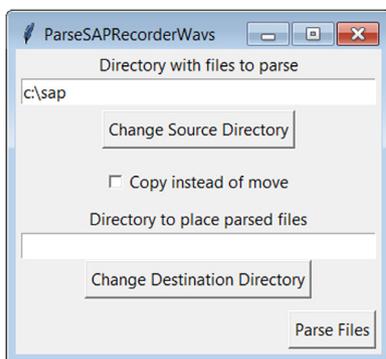


Fig. 4. The GUI for the companion program ParseSAPRecorderWavs that renames files according to our convention (described in Fig. 3, second panel) and moves them into subdirectories by recording day. This utility has options to store renamed files in a different directory (default is to create recording day subdirectories in the source directory) and to copy files instead of moving them (the default is to move files as they are generally large and their number consumes significant disk space). It is important that files for input into KLFromRecordingDays are named according to our convention as animal ID and recording day are accessed from the filename.

to our conventional format: *-##-#####.wav, where * = animal ID, ## = recording day, ##### = *.wav file number. This utility software is called “ParseSAPRecorderWavs” and is available in the `cx_freeze\ParseSAPRecorderWavs` GitHub subdirectory (see Fig. 4). KLFromRecordingDays is dependent upon SAP determining syllable acoustic features from *.wav files in this file format to read animal ID and recording day information. The source code for this utility is also provided so that changes to accommodate other *.wav file naming conventions can be made.

4.2. Empirical results

Depending upon the size of the MySQL table being processed, KLFromRecordingDays can take several hours to complete 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 Fig. 5.

5. Illustrative examples

The project for which this software was developed involves determining the extent to which an experimental new drug may improve recovery from fractional damage (a microlesion of about 10%) to a pre-vocal-motor region of zebra finch cortex. This microlesion procedure was originally developed by [6] and results in a temporary disruption of vocal patterns for about seven days in typical animals. A drug capable of mitigating effects of these lesions should reduce the time to recovery and/or magnitude of vocal pattern disruption. Such mitigation of phonological effects should be evident from reduced KL distance measures in animals treated with drug compared to vehicle controls.

In an initial dose–response experiment, animals ($n = 24$) were randomly assigned to receive either 0, 1, 10, or 100 mg/kg once every morning of the new drug. These dosage groups were further divided into three surgery conditions: No Microlesion; Sham Microlesion and; Microlesion groups ($n = 8$ for each drug dosage). Animals were housed in single cages placed into recording chambers ($30 \times 24 \times 16$ ” plastic box fitted with a light and microphone suspended from the center of the top panel). Microphones were monitored via computer continuously, and sounds meeting song-like criteria were saved to hard drives in *.wav format using Sound Analysis Recorder (see Fig. 3 for analysis steps, <http://soundanalysispro.com/manual-1/chapter-7-recording-vocal-sounds/recorder-overview>, last accessed 9/11/2017). Baseline recordings were made for three days. Daily drug treatments began six days prior to surgical procedures and were given for the entire 17 day post-baseline experimental period. Surgical procedures were done on day 7 post baseline. Recovery was followed for an additional 10 days. Effects of drug treatments prior to surgical procedures can be evaluated using baseline recordings as the “template” and the following six pre-surgery drug treatment days as the “target”. Recovery of vocal patterns following microlesions can be assessed using recordings made during the six pre-surgery drug treatment days as the “template” and ten post-surgery days as “target”.

Recordings in *.wav format were copied to either USB hard drives or to cloud storage for analysis. The utility program ParseSAPRecorderWavs (available on the same GitHub repository as KLFromRecordingDays) was used to parse *.wav files into subdirectories by recording day, and SAP was used to segment recorded sounds into distinct syllables. The totality of recordings made were analyzed by SAP. The MySQL table produced by SAP was used for input to KLFromRecordingDays and KL distance values for each of the 13 non-duration acoustic parameters were calculated and stored in Excel spreadsheet format. All syllables segmented by SAP were used by KLFromRecordingDays with a cut-off of a minimum of five syllables per *.wav file (specified in the GUI). Thus, the analysis was completely objective with all recordings processed, and all *.wav files consisting of at least five segmented syllables used for KL distance calculations. Resulting values were summarized across dosage and surgical condition groups, and statistically compared via 2-way ANOVA using Holm–Sidak post-hoc comparisons (Statistical comparisons were made using SigmaStat 3.1 run on a Windows XP emulation).

Preliminary analyses indicated significant differences across both dosage and surgical conditions. In examining results from the

Bird = B57
 Day(s) used for template: 3_4_5
 15 axes divisions, 1e-06 smoothing factor, 5 syllables/wav minimum , only motifs, 35 ms interval
 Day 3: 3975 total, 2560 motif, Day 4: 1082 total, 716 motif, Day 5: 1332 total, 818 motif,

| Day# | mean_amplitude | mean_pitch | mean_FM | mean_am2 | mean_entropy | mean_goodness | mean_mean_fmvar_pitch | var_FM | var_entropy | var_goodness | var_mean_fm | var_am | average | Total Syllables | Motif Syllables |
|------|----------------|------------|------------|------------|--------------|---------------|-----------------------|------------|-------------|--------------|-------------|------------|------------|-----------------|-----------------|
| 6 | 0.41170797 | 0.30004697 | 0.9099079 | 0.5591309 | 0.59995836 | 0.57670725 | 0.91888541 | 0.05890259 | 0.50775488 | 0.70919959 | 0.30101948 | 0.34648041 | 0.43385551 | 0.51027363 | 850 |
| 7 | 0.70225125 | 0.60982574 | 0.85678063 | 0.44451513 | 0.75610775 | 0.532971 | 1.05242611 | 0.11336834 | 0.79035482 | 0.76247171 | 0.33471599 | 0.89005328 | 0.49360834 | 0.64134232 | 772 |
| 8 | 1.77510642 | 2.32268539 | 2.53867311 | 3.02374776 | 1.90166379 | 2.21385103 | 3.06384151 | 0.31462053 | 1.77086163 | 2.10941265 | 1.07571198 | 3.3328589 | 2.50427399 | 2.14979298 | 1074 |
| 9 | 2.03529589 | 1.78917565 | 4.08882934 | 2.83427719 | 2.89960625 | 3.91831331 | 2.41164627 | 1.32077468 | 3.22410906 | 3.31344644 | 3.01519114 | 2.52691187 | 2.9433725 | 2.7939192 | 1048 |
| 11 | 1.2226753 | 1.24771185 | 2.13667734 | 1.95364278 | 1.88298603 | 1.47495789 | 2.47605898 | 0.29175624 | 1.45667055 | 1.27310974 | 0.91596017 | 1.91727427 | 2.22318382 | 1.57482038 | 1171 |
| 12 | 3.00140362 | 0.5786027 | 0.96162891 | 0.81923325 | 1.09006272 | 1.12176808 | 1.22046477 | 0.35261535 | 1.35331023 | 1.00453715 | 1.33950398 | 1.00637253 | 0.72469464 | 1.12109215 | 1083 |
| 13 | 1.05604933 | 0.25800933 | 0.98645533 | 0.48691678 | 0.53885079 | 0.48424885 | 1.23075825 | 0.13766907 | 1.21189744 | 1.00548648 | 0.60316399 | 0.50803894 | 0.39966446 | 0.68516993 | 899 |
| 14 | 2.37175613 | 0.55695903 | 1.18959833 | 1.62771249 | 0.77637219 | 0.98741559 | 1.54770063 | 0.3280457 | 1.51648041 | 1.2877688 | 0.79813057 | 1.20707478 | 1.38906774 | 1.19877557 | 1049 |
| 15 | 1.73647556 | 0.41450361 | 1.11969827 | 1.02976481 | 0.76329111 | 0.67798203 | 1.05806933 | 0.20834104 | 0.9648392 | 0.80897774 | 0.68592713 | 0.70980697 | 0.79261561 | 0.84386865 | 1159 |
| 16 | 1.78101044 | 0.27027075 | 0.52579451 | 1.15169362 | 0.63822853 | 0.78579471 | 1.08696031 | 0.13893558 | 1.21156071 | 1.07614335 | 0.6365955 | 0.54142333 | 0.91285272 | 0.82748185 | 1114 |
| 17 | 0.96519335 | 0.42245829 | 1.06610355 | 0.81078634 | 0.87079461 | 0.57540911 | 1.17450415 | 0.12853329 | 1.07510919 | 0.98884178 | 0.72888032 | 1.1656394 | 0.87500049 | 0.83440414 | 1190 |
| 18 | 1.2924973 | 0.3511464 | 0.67069834 | 0.71520515 | 0.83255842 | 0.62657466 | 1.17610901 | 0.22197503 | 1.44748069 | 0.99635807 | 0.60774974 | 0.81125236 | 0.97327366 | 0.86637536 | 1163 |
| 19 | 0.73325751 | 0.43017573 | 0.7207695 | 1.97422759 | 0.51846358 | 0.38461705 | 1.15180644 | 0.07793011 | 1.41536363 | 0.72437293 | 0.41394512 | 0.72908394 | 1.6864798 | 0.84311484 | 813 |
| 20 | 1.41297229 | 0.16107148 | 0.24911432 | 0.27826616 | 0.39072841 | 0.26667668 | 1.03270151 | 0.07224992 | 0.86903033 | 0.46980652 | 0.31556464 | 0.34940432 | 0.23995237 | 0.46981069 | 1090 |

Fig. 5. Typical KLFromRecordingDays results in Excel spreadsheet format. Target recording days are listed by rows starting with row five. The 13 acoustic features for which KL distances are calculated are written to columns C–O. Column P contains mean KL distance measures across all acoustic features. Column Q shows the number of individual syllables used for analysis for that recording day. The header contains animal ID, GUI settings including days used for the template.

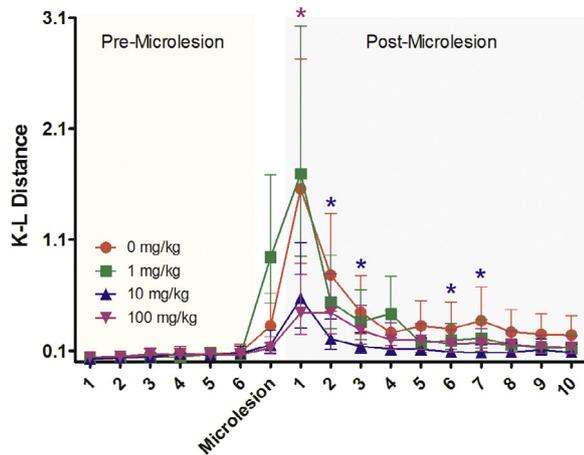


Fig. 6. Experimental results generated using KLFromRecordingDays shows drug effects on phonetic divergence when comparing Pre-Lesion recordings (as template) to Post Lesion recording days (targets). Lower KL distance measures indicate a lower degree of lesion-induced change. Thus, drug treatments appear to significantly reduce lesion-induced phonetic disruption (* = significant difference from 0 mg/kg control, 2-way ANOVA, Holm–Sidak post tests).

group of animals that received microlesions (Fig. 6, graph produced using GraphPad Prism software <https://www.graphpad.com/>, last accessed 6/27/2017) significant reductions from the 0 mg/kg control group in mean KL distance measures across all acoustic parameters are evident in the 100 mg/kg group on recovery day 1, and in the 10 mg/kg group on recovery days 2, 3, 6, 7 (* $p < 0.05$, Holm–Sidak post-hoc test). These preliminary results suggest that both 10 and 100 mg/kg dosages of the experimental drug reduce the magnitude of lesion effects (no differences in infarct size were observed across dosage groups, data not shown). Significantly lower KL distance measures in the 10 mg/kg drug group than 0 mg/kg controls out to 7 days post-microlesion suggests decreased time to recovery of song phonology.

6. Conclusions

Development of KLFromRecordingDays allows us to employ the entirety of the massive amount of syllable acoustic 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 (see Fig. 6), and should increase the value of songbirds as a preclinical model. We have also begun to apply the software to assess phonological effects of drug treatments delivered during development, and to those following restraint- and chronic mild, unpredictable stress in songbirds.

In addition to our application of this software to zebra finch song, it may have applicability in assessing phonetic changes over time in other species with vocalizations amenable to SAP segmentation into syllables. This includes other vocal learners like bats and cetaceans, and a large number of animals that produce call-like vocalizations. Given challenges in segmenting human language into syllables, and irregularity of speech, the software may be limited in ability to analyze most naturally-produced human vocalizations. However, analysis of those types of human vocalizations amenable to segmentation remains possible (including laughter and production of distinct phones and phonemes). Thus, the software may be useful to the small but potentially important field studying laughter. It may also be of clinical utility in assessing progress of speech therapies wherein distinct speech sounds are practiced repetitively in isolation.

The software is currently set up for day-by-day analyses and should be suitable as-is for typical songbird 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.

Acknowledgments

This work was financially supported by GW Pharmaceuticals. We are indebted to the group led by Ofer Tchernichovski who created and made freely available the Sound Analysis Pro 2011 software upon which our work is based (<http://soundanalysispro.com/credits>, last accessed 6/27/2017).

References

- [1] Shriberg LD, Tomblin JB, McSweeney JL. J. Speech Lang. Hear. Res. JSLHR 1999;42:1461–81.
- [2] Tchernichovski O, Nottebohm F, Ho CE, Pesaran B, Mitra PP. Anim. Behav. 2000;59:1167–76.
- [3] Wu W, Thompson JA, Bertram R, Johnson F. J. Neurosci. Methods 2008;174:147–54.
- [4] Daou A, Johnson F, Wu W, Bertram R. J. Neurosci. Methods 2012;210:147–60.
- [5] Thompson JA, Basista MJ, Wu W, Bertram R, Johnson F. J. Neurosci. Off. J. Soc. Neurosci. 2011;31:322–30.
- [6] Thompson JA, Wu W, Bertram R, Johnson F. J. Neurosci. Off. J. Soc. Neurosci. 2007;27:12308–20.