Received: 17 September 2019 | Accepted: 6 January 2020

DOI: 10.1111/2041-210X.13368

Methods in Ecology and Evolution BRITISH ECOLOGICAL SOCIETY

APPLICATION

Chipper: Open-source software for semi-automated segmentation and analysis of birdsong and other natural sounds

Abigail M. Searfoss[1,2] | James C. Pino[1,3] | Nicole Creanza[2]

[1]Program in Chemical and Physical Biology, Vanderbilt University, Nashville, TN, USA

[2]Department of Biological Sciences, Vanderbilt University, Nashville, TN, USA

[3]Center for Structural Biology, Vanderbilt University, Nashville, TN, USA

Correspondence
Nicole Creanza
Email: nicole.creanza@vanderbilt.edu

Funding information
Division of Graduate Education; Vanderbilt University

Handling Editor: Veronica Zamora-Gutierrez

Abstract

1. Audio recording devices have changed significantly over the last 50 years, making large datasets of recordings of natural sounds, such as birdsong, easier to obtain. This increase in digital recordings necessitates an increase in high-throughput methods of analysis for researchers. Specifically, there is a need in the community for open-source methods that are tailored to recordings of varying qualities and from multiple species collected in nature.
2. We developed Chipper, a Python-based software to semi-automate both the segmentation of acoustic signals and the subsequent analysis of their frequencies and durations. For avian recordings, we provide widgets to best determine appropriate thresholds for noise and syllable similarity, which aid in calculating note measurements and determining song syntax. In addition, we generated a set of synthetic songs with various levels of background noise to test Chipper's accuracy, repeatability and reproducibility.
3. Chipper provides an effective way to quickly generate quantitative, reproducible measures of birdsong. The cross-platform graphical user interface allows the user to adjust parameters and visualize the resulting spectrogram and signal segmentation, providing a simplified method for analysing field recordings.
4. Chipper streamlines the processing of audio recordings with multiple user-friendly tools and is optimized for multiple species and varying recording qualities. Ultimately, Chipper supports the use of citizen-science data and increases the feasibility of large-scale multi-species birdsong studies.

KEYWORDS

acoustic signals, birdsong, citizen science, Python, recording, segmentation, software, syntax

1 | INTRODUCTION

Acoustic communication is one of the few natural behaviours that can be easily recorded, digitized and studied (Catchpole & Slater, 2008; Cocroft & Rodríguez, 2005; Garland, Rendell, Lamoni, Poole, & Noad, 2017; Ryan & Guerra, 2014). Often, behavioural studies involve laboratory observations, which can lead to fundamental insights but may disrupt natural animal behaviour (Fehér, Wang, Saar, Mitra, & Tchernichovski, 2009; Marler & Peters, 1977; Searcy, 1984). In addition, scientists can collect acoustic sounds in the wild without disturbing animals, eliminating potential influences of the laboratory environment on behaviour but limiting the types of experiments possible (Grant & Grant, 1996; Lachlan, Ratmann, & Nowicki, 2018; Shizuka, Ross Lein, & Chilton, 2016; Williams, Levin, Ryan Norris, Newman, & Wheelwright, 2013).

Moreover, recordings can be pooled across sources—professionals and hobbyists, analogue and digital, old and new—providing vast datasets that span many years and large geographic scales (Bolus, 2014; Roach & Phillmore, 2017). Thus, audio recordings are an advantageous resource for broad-scale animal behaviour research.

Birdsong has been studied in ecology and evolution for decades (Marler & Tamura, 1964; Thorpe, 1958). Historically, field studies of birdsong have provided insights into mating and territory-defense behaviours, evolutionary events such as speciation and hybridization and environmental adaptation (Grant & Grant, 1997; Liu & Kroodsma, 2007; Mason et al., 2017; Nowicki & Searcy, 2004; Robinson, Snyder, & Creanza, 2019; Slabbekoorn & Peet, 2003; Snyder & Creanza, 2019). These studies are often conducted with banded birds and direct recordings using parabolic microphones. Some song analysis software is well-suited to these studies, allowing users to visualize and manually select songs from their field recordings for analysis (Boersma & Weenink, 2019; Burt, 2001; Center for Conservation Bioacoustics, 2019; Lachlan, 2007). On the other hand, laboratory experiments often use individual sound-attenuating recording chambers. Such experiments have greatly extended our understanding of the neurobiology of learning and development (Tchernichovski, Mitra, Lints, & Nottebohm, 2001). Alongside laboratory work, song analysis software has been developed to provide quantitative comparisons between individuals from a specific species, such as pupils and tutors in song-learning experiments (Lachlan, 2007; Tchernichovski, Nottebohm, Ho, Pesaran, & Mitra, 2000). In sum, fieldwork and laboratory experiments, particularly when paired with software, have made large contributions in understanding acoustic communication.

Concurrently, portable audio recording devices have changed significantly over the last 50 years, moving from large reel-to-reel devices to handheld digital recorders and smartphones, which has made collecting natural recordings much easier (Sullivan et al., 2009; Vellinga & Planqué, 2015). This new technology has improved collection of both wild and laboratory recordings and led to an active world-wide community of citizen scientists who record and archive birdsong (Bonney et al., 2009; Silvertown, 2009; Sullivan et al., 2009; Wood, Sullivan, Iliff, Fink, & Kelling, 2011). Although there are many scientific questions that can be answered using these expanding citizen-science datasets of birdsong or other natural sounds (e.g. Xeno-canto, eBird, Macaulay Library at the Cornell Lab of Ornithology), there is still a need for high-throughput and automated methods of song analysis that address the varying quality and multi-species nature of citizen-science recordings. One R package, WarbleR, has made progress by facilitating the retrieval and analysis of songs from the Xeno-canto repository (Araya-Salas & Smith-Vidaurre, 2017). Existing signal processing toolboxes in Python are neither optimized for natural recordings nor user-friendly for researchers unfamiliar with computer programming. To reduce and streamline the manual work involved in processing databases of natural recordings, we developed Chipper, an open-source Python-based (v3.6.2) software with a Kivy-based (v1.10.0) graphical user interface, to semi-automate the segmentation and analysis of acoustic signals.

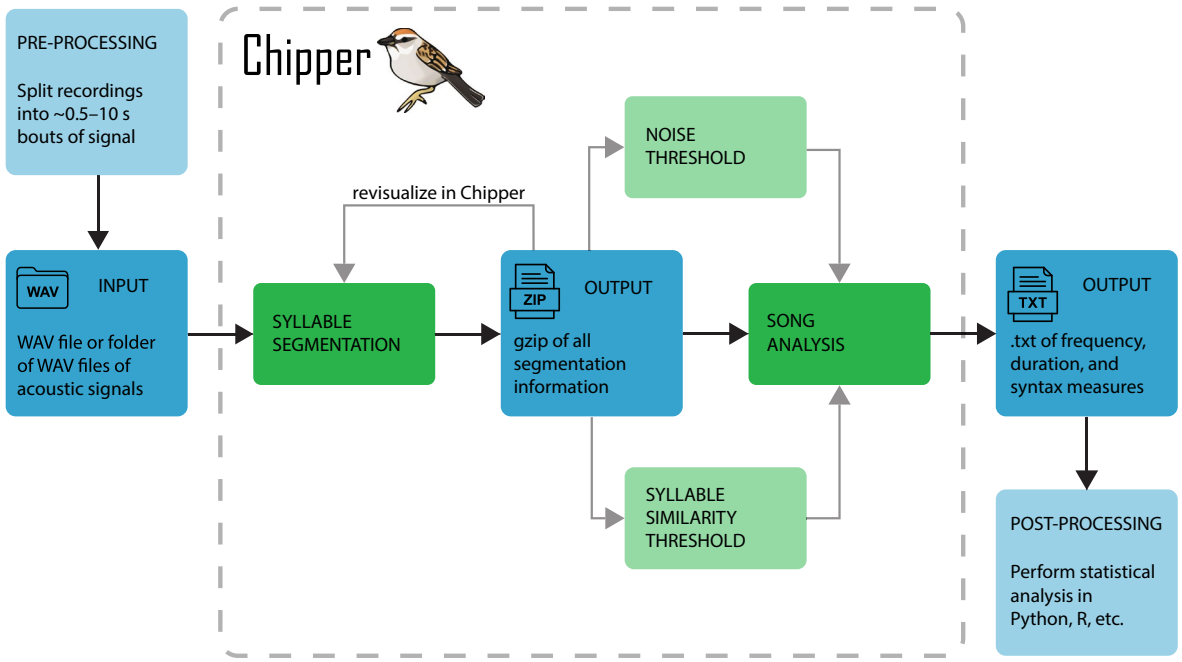

FIGURE 1 Chipper's streamlined process of segmenting and analysing recordings. Blue steps indicate inputs and outputs; green steps indicate Chipper widgets. Navigate through Chipper as follows: (a) Split recordings into ~0.5–10 s of signal, often a bout of singing. (b) Gather bouts of acoustic signals (WAV files) to input into Chipper. (c) Load files and begin with the default syllable segmentation. Alter segmentation parameters, viewing how this changes the quality of the signal and the segmentation. (d) Segmentation results in zipped files with all necessary information. (e) Use widgets to determine the best thresholds for noise and syllable similarity. (f) Run song analysis using these thresholds. Measurements characterizing frequencies and durations for the song, syllables and notes are calculated. (g) All measurements are output into two text files. (h) Perform statistical analysis on song measurements in Python, R, etc

Chipper facilitates syllable segmentation and subsequent analysis of frequency, duration and syntax, improving efficiency in using citizen-science recordings and increasing the feasibility of multi-species studies. While primarily designed for birdsong, Chipper can also process other natural sounds, such as frog or insect vocalizations. Our software is open-source and user-friendly, allowing seamless integration into research and STEM education. In particular, Chipper streamlines the song analysis process, eliminating the need to manually handle each song multiple times (Figure 1). In addition, we created synthetic datasets of birdsong for testing acoustic software and conducted a thorough test of Chipper's accuracy, repeatability and reproducibility (Supporting Information).

2 | CHIPPER'S INTERFACE AND CAPABILITIES

Chipper is primarily designed to parse syllables from a bout of birdsong, but any sound recording can be processed. We suggest recordings <3 MB or ~0.5–10 s, but the optimal value will differ between projects (based on sampling rate, syllable duration, computing resources, etc.). For long songs, we advise splitting the recording into multiple files before processing in Chipper. Therefore, users should first select and export song bouts (as WAV files) from in-house recordings or citizen-science data; we recommend Audacity for manual pre-processing and monitoR or Kaleidoscope Pro Analysis Software for semi-automated selection. Chipper guides the user through two main steps to extract information from WAV files of song bouts: syllable segmentation and song analysis (Figure 1).

2.1 | Syllable segmentation

On the Chipper landing page (Figure 2a), the user can adjust the defaults for the automated segmentation. Next, a single WAV file or an entire folder of WAV files can be selected to begin segmentation. Chipper will then semi-automate the process of noise reduction and syllable parsing of each recorded bout of song. The syllable segmentation window (Figure 2b) shows two images: the top image is the spectrogram of the file and the bottom shows a binary image calculated based on user-informed parameters, with onsets (short green lines) and offsets (tall green lines) depicting the automated syllable segmentation.

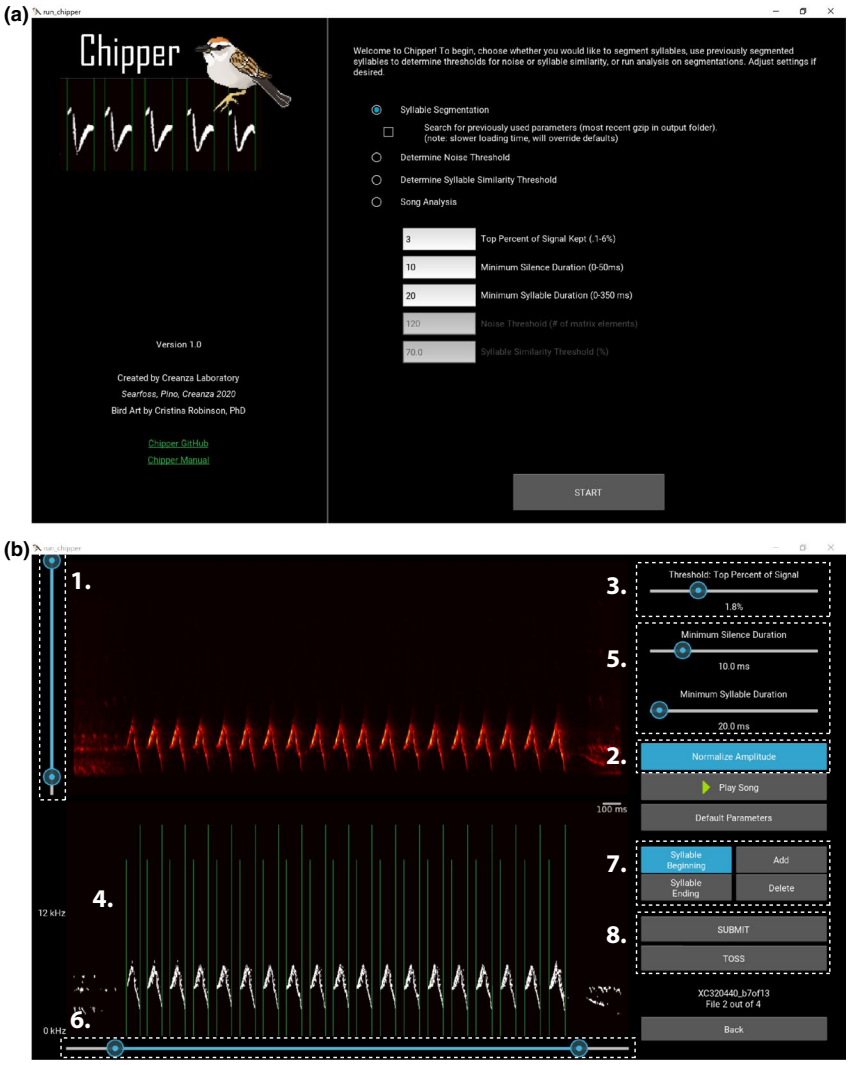

FIGURE 2 Chipper interface. (a) Landing page of Chipper. Here the user can choose to segment songs, visualize the already segmented songs to choose thresholds for noise and syllable similarity or run song analysis with default or user-defined thresholds. (b) Segmentation window with parameters labelled in the order that they are applied to the spectrogram and segmentation calculations (see Section 2.1). For images of other Chipper windows and more detailed instructions on the use of Chipper, see the manual at https://github.com/CreanzaLab/chipper/tree/master/docs

TABLE 1 Chipper's output measurements

Measurement	Calculation
Song duration	(time of last syllable offset − time of first syllable onset)
Number of syllables	number of syllable onsets in a song
Syllable rate	(number of syllables)/(song duration)
Average syllable duration	mean(time of syllable offset − time of syllable onset)
SD of syllable duration	standard deviation(time of syllable offset − time of syllable onset)
Average silence duration	mean(time of syllable onset − time of previous syllable offset)
SD of silence duration	standard deviation(time of syllable onset − time of previous syllable offset)
Largest syllable duration	max(time of syllable offset − time of syllable onset)
Smallest syllable duration	min(time of syllable offset − time of syllable onset)
Largest silence duration	max(time of syllable onset − time of previous syllable offset)
Smallest silence duration	min(time of syllable onset − time of previous syllable offset)
Average syllable frequency range	mean(maximum frequency − minimum frequency for each syllable)
SD syllable frequency range	standard deviation(maximum frequency − minimum frequency for each syllable)
Average syllable lower frequency	mean(minimum frequency of each syllable)
Average syllable upper frequency	mean(maximum frequency of each syllable)
Largest syllable frequency range	max(maximum frequency − minimum frequency for each syllable)
Smallest syllable frequency range	min(maximum frequency − minimum frequency for each syllable)
Maximum syllable frequency	max(maximum frequency of each syllable)
Minimum syllable frequency	min(minimum frequency of each syllable)
Overall syllable frequency range	max(maximum frequency of each syllable) − min(minimum frequency of each syllable)
Syllable stereotypy	list of the mean(pairwise percent similarities) for each repeated syllable, where percent similarity is the maximum(cross-correlation between each pair of syllables)/maximum(autocorrelation of each of the compared syllables) × 100 Note: For the definition of repeated and unique syllables, see Section 3.2.
Mean syllable stereotypy	mean(stereotypy values for each repeated syllable)
SD syllable stereotypy	standard deviation(stereotypy values for each repeated syllable)
Syllable pattern	list of the syllables in the order that they are sung, where each unique syllable is assigned a number (i.e. the song syntax)
Number of unique syllables	number of unique values in syllable pattern
Degree of repetition	(number of syllable onsets in a song)/(number of unique syllables)
Degree of sequential repetition	(number of syllables that are followed by a repeat of the same syllable)/(number of syllables − 1)
Number of notes	number of 4-connected elements of the spectrogram with an area greater than the noise threshold
Number of notes per syllable	(total number of notes)/(total number of syllables)
Average note duration	mean(time of note ending − time of note beginning)
SD of note duration	standard deviation(time of note ending − time of note beginning)
Largest note duration	max(time of note ending − time of note beginning)
Smallest note duration	min(time of note ending − time of note beginning)
Overall note frequency range	max(maximum frequency of each note) − min(minimum frequency of each note)
Average note frequency range	mean(maximum frequency − minimum frequency for each note)
SD note frequency range	standard deviation(maximum frequency − minimum frequency for each note)
Average note lower frequency	mean(minimum frequency of each note)
Average note upper frequency	mean(maximum frequency of each note)
Largest note frequency range	max(maximum frequency − minimum frequency for each note)
Smallest note frequency range	min(maximum frequency − minimum frequency for each note)
Maximum note frequency	max(maximum frequency of each note)
Minimum note frequency	min(minimum frequency of each note)

The user can adjust the segmentation parameters using the sliders. With each parameter adjustment, a new binary image and corresponding onsets and offsets are calculated in the following order (numbered as in Figure 2b):

1. The spectrogram of the recording is created from the WAV file (method adapted from Gardner & Magnasco, 2006), and low- and high-frequency noise can be removed with high- and low-pass filters respectively. Colours in the resulting spectrogram are rescaled based on the remaining signal.
2. Selecting 'Normalize Amplitude' rescales the amplitude across the spectrogram.
3. The 'Threshold: Top Percent of Signal', q, is used to find the $(100 - q)$ th percentile of signal. Only signal above this percentile is retained and plotted in the binary image; all other signal is set to zero.
4. Syllable onsets (beginnings) and offsets (endings) are calculated by summing the columns of the spectrogram, creating a vector of total signal intensity over time. Then, an onset is defined as the position of the first element in the matrix where signal is present after silence and an offset as the position of the first element of the matrix with no signal after prolonged signal.
5. Two parameters act as constraints on the list of onsets and offsets—'Minimum Silence Duration' and 'Minimum Syllable Duration'. If the time between the offset of one syllable and the onset of the next syllable is less than or equal to the minimum silence duration, these boundaries are removed, combining the two syllables. Similarly, if the duration between an onset and offset of one syllable is less than the minimum syllable duration, the onset–offset pair is removed.
6. If any onsets or offsets are outside the time range of interest (determined by the slider below the binary image), they will be removed.
7. The user can manually add or delete onsets and offsets to adjust segmentation, such as adding a missing onset or offset or altering an incorrect placement due to noise.
8. Lastly, the user can submit the parameters, the final binary matrix and syllable onsets and offsets. If a satisfactory segmentation was not reached, the file can be tossed.

2.2 | Quantitative analysis

As syllable segmentation is completed for each song, Chipper generates an output file (gzip) containing all necessary information on the binary image, segmentation and conversion factors for both time and frequency space. These output files can then be processed using Chipper's analysis tool. This portion of Chipper is fully automated; the window serves to show the number of files processed out of the total selected by the user. For each song being processed, Chipper produces multiple song, syllable, note and syntax measurements (Table 1). Many of these outputs rely on the input parameters for noise and syllable similarity thresholds; thus, we recommend using our widgets in Chipper to determine appropriate thresholds for each species-specific set of songs studied.

3 | ADDITIONAL NOTE AND SYNTAX ANALYSES FOR BIRDSONG APPLICATION

For a subset of measurements provided by Chipper's analysis tool, the user can improve measurement accuracy by setting a noise threshold and syllable similarity threshold. The noise threshold affects any note-related and frequency-related calculations, since any signal smaller than the noise threshold is removed from the binary spectrogram. For example, low-frequency noise in a syllable that is not removed either in the segmentation process or by the noise threshold will affect multiple frequency measurements (minimum syllable frequency, average syllable frequency range, etc.) Any calculation that specifically uses the onsets and offsets, such as song, syllable and silence durations, will not be affected by the noise threshold. The syllable similarity threshold only affects syntax-related calculations (number of unique syllables, syllable pattern, syllable stereotypy, etc.). Since it is useful to set these thresholds based on multiple songs, we have provided widgets to visualize these thresholds.

3.1 | Determine threshold for noise

Chipper's quantitative analysis uses connectivity to classify signal within a syllable as either a note or noise. Specifically, any signal within the syllables (defined by onsets and offsets) in the binary image that is connected by an edge (not corner, i.e. 4-connected) and has an area greater than the user-specified threshold is labelled as an individual note, and any signal with an area less than or equal to the threshold is considered noise. Since signal connectivity is highly dependent on signal-to-noise ratio or amplitude, we provide a widget to determine the best threshold for a set of songs. In the noise threshold widget, the user selects a folder of multiple gzips (the output from syllable segmentation) as a representative subset of the songs being analysed. For each song, the user can change the threshold to visualize areas being classified as notes (coloured) versus noise (white) and then submit a threshold for that song. After going through the selected songs, a summary is provided including the average, minimum and maximum thresholds selected for noise and, if enough songs are processed in the widget, a histogram. This information is provided to guide the user in choosing a single threshold that will be used in song analysis for the entire set of song files. We advise caution in using the output from note analysis for low-quality recordings: whereas high-quality recordings will have syllables in which signal is only disconnected at true notes, the degraded signal in low-quality recordings can lead to many false notes.

3.2 | Determine threshold for syllable similarity

For each pair of syllables, a percent syllable similarity is calculated by sliding one syllable's binary matrix across another syllable's binary matrix and finding the maximum overlap (cross-correlation). This is then repeated for each syllable compared to itself, providing

an autocorrelation for each syllable. We scale the maximum overlap between the two syllables by dividing by the maximum of the two syllables' autocorrelations; multiplied by 100, this results in a percent of the maximum possible overlap or percent syllable similarity for the syllable pair. Similar methods of spectrographic cross-correlation have been previously demonstrated as a useful method in determining syllable types (Clark, Marler, & Beeman, 1987). Applying the user-defined syllable similarity threshold to the resulting pairwise matrix, we establish the syntax for the recording by considering two syllables to be repeats of the same syllable if their similarity is greater than or equal to the user-specified threshold. If two syllables are considered to be the same type and the second one of those is considered the same as a third syllable, then the third syllable is classified as the same type as the first two. This prevents groups of similar syllables from being separated but also means that the first and third syllables could have a percent similarity below the threshold but still be considered the same type. Chipper's syllable similarity threshold widget guides the user in deciding an appropriate value. The binary song and the corresponding syllable onset and offset lines from syllable segmentation are plotted. Based on the threshold, the syntax is displayed in text as well as visually, with repeats of the same syllable shown in the same colour. The user can change the threshold to see how this will change the syntax of the song, submitting the threshold when satisfied. When all of the sample songs have been processed, a summary will be displayed with the average, minimum and maximum thresholds selected for syllable similarity and, if enough songs have been processed, a histogram. Once again, this information is provided to guide the user in choosing a threshold to process the entire set of song files of interest.

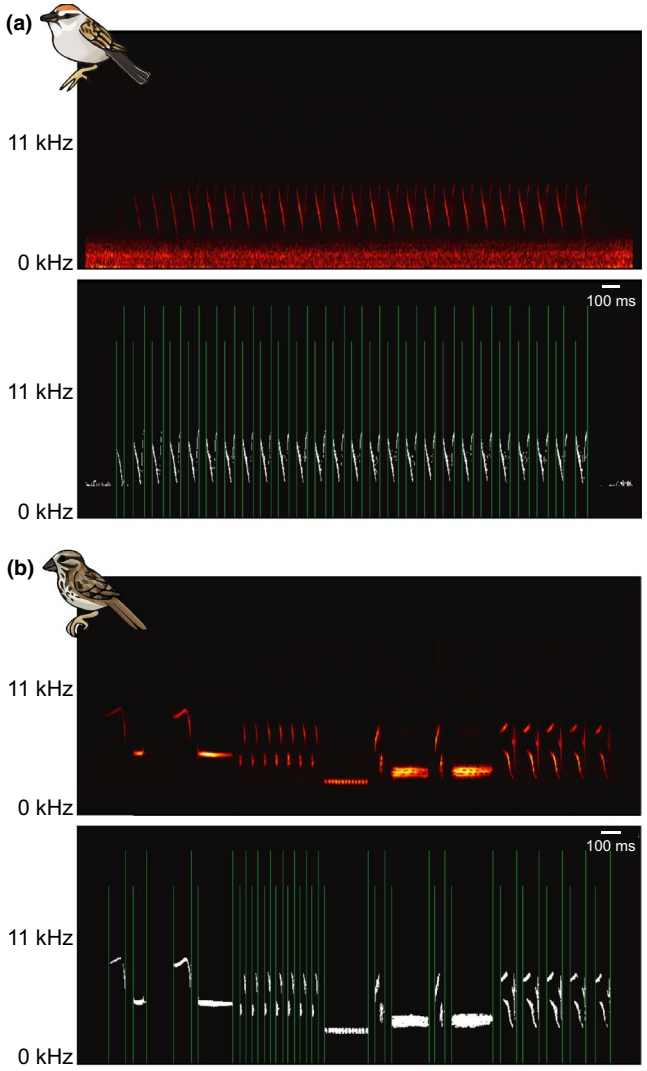

FIGURE 3 Chipper can segment songs of various qualities and from different species. Example song of (a) chipping sparrow and (b) song sparrow. The top images are the spectrograms when they are originally loaded into Chipper. The bottom images are the binary signal after parameters have been adjusted to optimize segmentation. The green lines show the syllable onsets (shorter lines) and offsets (taller lines)

4 | CONCLUSIONS

With the ever-growing repositories of citizen-science recordings, a new software was needed to handle the various recording qualities and vast species coverage (Figure 3). Thus, we developed Chipper as a free, open-source software to improve the workflow of audio signal processing with particular application to high-throughput analysis of citizen-science recordings. With its user-friendly graphical user interface, Chipper can be used by researchers, students in classrooms and curious citizen scientists alike. In testing Chipper, we found that it produced robust estimates of sound properties for a set of synthetic recordings, and these results were consistent within and between users and in the presence of natural and white noise (Supporting Information). We hope Chipper, in tandem with citizen-science data, can aid in large-scale spatiotemporal studies of acoustic signals, particularly global inter- and intra-species studies of birdsong (Searfoss, Liu, & Creanza, 2020). Chipper's song measurements could also prove valuable in studying the complex temporal variations associated with duets or coordinated singing. With open-source code on GitHub, we welcome users to extend and contribute to Chipper, tailoring it to additional projects and data types. In the future, as we continue to maintain and develop Chipper, we aim to add additional song measurements, such as syllable entropy, as well as functionality to accommodate longer recordings. Ultimately, using Python and Kivy, we have developed an application that facilitates audio processing of natural recordings, extending the utility of rapidly growing citizen-science databases and improving the workflow for current birdsong research in ecology and evolution.

ACKNOWLEDGEMENTS

We thank the following students for testing Chipper during its development: Vanderbilt University BSCI 1512L Fall 2018 class, Vanderbilt University BSCI 3965 Spring 2019 class, Megan

Mitchell, Nyssa Kantorek, Maria Sellers and Emily Beach. In addition, we thank Megan Mitchell for editing the Chipper Manual and Dr. Cristina Robinson for the bird art used in this paper and in the Chipper logo.

AUTHORS' CONTRIBUTIONS

A.M.S. and N.C. conceived and designed the project; A.M.S. and J.C.P. developed the software; Parts of the software were based on earlier code written by N.C. in MATLAB; A.M.S. translated this code to Python and built the graphical user interface in Kivy; J.C.P. packaged the software; A.M.S. led the writing of the manuscript with assistance and revision from J.C.P. and N.C. All authors contributed to drafts, edited the final manuscript, tested and verified the software and gave final approval for publication.

DATA AVAILABILITY STATEMENT

Chipper v1.0 can be downloaded for Mac, PC and Linux at https://github.com/CreanzaLab/chipper/releases. The DOI for the Chipper v1.0 release is https://doi.org/10.5281/zenodo.3620939 (Searfoss, Pino, & Creanza, 2020). The Chipper manual can be found at https://github.com/CreanzaLab/chipper/tree/master/docs. Code for Chipper and to create and analyse synthetic songs can be found at https://github.com/CreanzaLab/chipper. Chipper leverages several existing Python packages including SciPy (Jones, Oliphant, & Peterson, 2001; http://doi.org/10.5281/zenodo.817564), Pandas (McKinney, 2010; https://github.com/pandas-dev/pandas/releases/tag/v0.20.3), Matplotlib (Hunter, 2007; http://doi.org/10.5281/zenodo.573577), NumPy (Oliphant, 2006; van der Walt, Colbert, & Varoquaux, 2011; https://github.com/numpy/numpy/releases/tag/v1.13.1) and scikit-image (van der Walt et al., 2014; https://github.com/scikit-image/scikit-image/releases/tag/v0.13.0). We also use the Python library Kivy v.1.10.0 for building the graphical user interface (Virbel, Hansen, & Lobunets, 2011; https://github.com/kivy/kivy/releases/tag/1.10.0). The recordings used in this paper are freely available in the Xeno-canto repository: Jonathon Jongsma, XC320440, accessible at www.xeno-canto.org/320440; Chris Parrish, XC13690, accessible at www.xeno-canto.org/13690; Allen T. Chartier, XC16985, accessible at www.xeno-canto.org/16985. The folder and file images used in Figure 1 are adapted from icons found at the Noun Project: tab file document icon by IYIKON, .WAV Folder by Linseed Studio, Audio by Ben Avery, zip file document icon by IYIKON, wax file document icon by IYIKON and csv file document icon by IYIKON.

ORCID

Abigail M. Searfoss https://orcid.org/0000-0002-5417-7827
Nicole Creanza https://orcid.org/0000-0001-8821-7383

REFERENCES

Araya-Salas, M., & Smith-Vidaurre, G. (2017). warbleR: An r package to streamline analysis of animal acoustic signals. *Methods in Ecology and Evolution*, *8*, 184–191.

Boersma, P., & Weenink, D. (2019). *Praat: Doing phonetics by computer (Version 6.1.03)* [Computer software]. Retrieved from http://www.praat.org/

Bolus, R. T. (2014). Geographic variation in songs of the Common Yellowthroat. *The Auk*, *131*, 175–185. https://doi.org/10.1642/auk-12-187.1

Bonney, R., Cooper, C. B., Dickinson, J., Kelling, S., Phillips, T., Rosenberg, K. V., & Shirk, J. (2009). Citizen science: A developing tool for expanding science knowledge and scientific literacy. *BioScience*, *59*, 977–984. https://doi.org/10.1525/bio.2009.59.11.9

Burt, J. (2001). SYRINX-PC–A windows program for spectral analysis, editing, and playback of acoustic signals (Version 2.6h). [Computer software]. Retrieved from http://www.syrinxpc.com

Catchpole, C. K., & Slater, P. J. B. (2008). *Bird song: Biological themes and variations*. Cambridge, UK: Cambridge University Press.

Center for Conservation Bioacoustics. (2019). *Raven Pro: Interactive sound analysis software (Version 1.6)*. [Computer software]. Ithaca, NY: The Cornell Lab of Ornithology. Retrieved from http://ravensoundsoftware.com/

Clark, C. W., Marler, P., & Beeman, K. (1987). Quantitative analysis of animal vocal phonology: An application to swamp sparrow song. *Ethology*, *76*, 101–115. https://doi.org/10.1111/j.1439-0310.1987.tb00676.x

Cocroft, R. B., & Rodríguez, R. L. (2005). The behavioral ecology of insect vibrational communication. *BioScience*, *55*, 323. https://doi.org/10.1641/0006-3568(2005)055[0323:TBEOIV]2.0.CO;2

Fehér, O., Wang, H., Saar, S., Mitra, P. P., & Tchernichovski, O. (2009). De novo establishment of wild-type song culture in the zebra finch. *Nature*, *459*, 564–568.

Gardner, T. J., & Magnasco, M. O. (2006). Sparse time-frequency representations. *Proceedings of the National Academy of Sciences of the United States of America*, *103*, 6094–6099. https://doi.org/10.1073/pnas.0601707103

Garland, E. C., Rendell, L., Lamoni, L., Poole, M. M., & Noad, M. J. (2017). Song hybridization events during revolutionary song change provide insights into cultural transmission in humpback whales. *Proceedings of the National Academy of Sciences of the United States of America*, *114*, 7822–7829. https://doi.org/10.1073/pnas.1621072114

Grant, B. R., & Grant, P. R. (1996). Cultural inheritance of song and its role in the evolution of Darwin's Finches. *Evolution*, *50*, 2471–2487. https://doi.org/10.2307/2410714

Grant, P. R., & Grant, B. R. (1997). Mating patterns of Darwin's Finch hybrids determined by song and morphology. *Biological Journal of the Linnean Society*, *60*, 317–343. https://doi.org/10.1111/j.1095-8312.1997.tb01499.x

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*, 90–95. https://doi.org/10.1109/MCSE.2007.55

Jones, E., Oliphant, T., & Peterson, P. (2001). *SciPy: Open source scientific tools for Python*. Retrieved from http://www.scipy.org

Lachlan, R. F. (2007). *Luscinia: A bioacoustics analysis computer program (Version 1.0)*. [Computer software]. Retrieved from www.lusciniasound.org

Lachlan, R. F., Ratmann, O., & Nowicki, S. (2018). Cultural conformity generates extremely stable traditions in bird song. *Nature Communications*, *9*, 2417. https://doi.org/10.1038/s41467-018-04728-1

Liu, W.-C., & Kroodsma, D. E. (2007). Dawn and daytime singing behavior of chipping sparrows (*Spizella passerina*). *The Auk*, *124*(1), 44–52. https://doi.org/10.1093/auk/124.1.44

Marler, P., & Peters, S. (1977). Selective vocal learning in a sparrow. *Science*, *198*, 519–521. https://doi.org/10.1126/science.198.4316.519

Marler, P., & Tamura, M. (1964). Culturally transmitted patterns of vocal behavior in sparrows. *Science*, *146*, 1483–1486.

Mason, N. A., Burns, K. J., Tobias, J. A., Claramunt, S., Seddon, N., & Derryberry, E. P. (2017). Song evolution, speciation, and vocal learning in passerine birds. *Evolution*, *71*, 786–796. https://doi.org/10.1111/evo.13159

McKinney, W. (2010). Data structures for statistical computing in Python. In S. van der Walt & J. Millman (Eds.), *Proceedings of the 9th Python in science conference*, pp. 51–56. Retrieved from http://conference.scipy.org/proceedings/scipy2010/mckinney.html

Nowicki, S., & Searcy, W. A. (2004). Song function and the evolution of female preferences: Why birds sing, why brains matter. *Annals of the New York Academy of Sciences*, *1016*, 704–723. https://doi.org/10.1196/annals.1298.012

Oliphant, T. E. (2006). *A guide to NumPy*. Provo, UT: Trelgol Publishing.

Roach, S. P., & Phillmore, L. S. (2017). Geographic variation in song structure in the hermit thrush (*Catharus guttatus*). *The Auk*, *134*, 612–626. https://doi.org/10.1642/auk-16-222.1

Robinson, C. M., Snyder, K. T., & Creanza, N. (2019). Correlated evolution between repertoire size and song plasticity predicts that sexual selection on song promotes open-ended learning. *eLife*, *8*. https://doi.org/10.7554/eLife.44454

Ryan, M. J., & Guerra, M. A. (2014). The mechanism of sound production in túngara frogs and its role in sexual selection and speciation. *Current Opinion in Neurobiology*, *28*, 54–59. https://doi.org/10.1016/j.conb.2014.06.008

Searcy, W. A. (1984). Song repertoire size and female preferences in song sparrows. *Behavioral Ecology and Sociobiology*, *14*, 281–286. https://doi.org/10.1007/bf00299499

Searfoss, A. M., Liu, W. C., & Creanza, N. (2020). Geographically well-distributed citizen science data reveals range-wide variation in the chipping sparrow's simple song. *Animal Behaviour*, *161*, 63–76. https://doi.org/10.1016/j.anbehav.2019.12.012

Searfoss, A., Pino, J., & Creanza, N. (2020). Data from: CreanzaLab/chipper: Version 1.0 (Version 1.0). *Zenodo*, https://doi.org/10.5281/zenodo.3620939

Shizuka, D., Ross Lein, M., & Chilton, G. (2016). Range-wide patterns of geographic variation in songs of golden-crowned sparrows (*Zonotrichia atricapilla*). *The Auk*, *133*, 520–529. https://doi.org/10.1642/auk-16-27.1

Silvertown, J. (2009). A new dawn for citizen science. *Trends in Ecology & Evolution*, *24*, 467–471. https://doi.org/10.1016/j.tree.2009.03.017

Slabbekoorn, H., & Peet, M. (2003). Birds sing at a higher pitch in urban noise. *Nature*, *424*, 267–267. https://doi.org/10.1038/424267a

Snyder, K. T., & Creanza, N. (2019). Polygyny is linked to accelerated birdsong evolution but not to larger song repertoires. *Nature Communications*, *10*, 884. https://doi.org/10.1038/s41467-019-08621-3

Sullivan, B. L., Wood, C. L., Iliff, M. J., Bonney, R. E., Fink, D., & Kelling, S. (2009). eBird: A citizen-based bird observation network in the biological sciences. *Biological Conservation*, *142*, 2282–2292. https://doi.org/10.1016/j.biocon.2009.05.006

Tchernichovski, O., Mitra, P. P., Lints, T., & Nottebohm, F. (2001). Dynamics of the vocal imitation process: How a zebra finch learns its song. *Science*, *291*, 2564–2569.

Tchernichovski, O., Nottebohm, F., Ho, C. E., Pesaran, B., & Mitra, P. P. (2000). A procedure for an automated measurement of song similarity. *Animal Behaviour*, *59*, 1167–1176. https://doi.org/10.1006/anbe.1999.1416

Thorpe, W. H. (1958). Further studies on the process of song learning in the chaffinch (*Fringilla Coelebs Gengleri*). *Nature*, *182*, 554–557. https://doi.org/10.1038/182554a0

van der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, *13*, 22–30. https://doi.org/10.1109/MCSE.2011.37

van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., … Yu, T. (2014). scikit-image: Image processing in Python. *PeerJ*, *2*, e453. https://doi.org/10.7717/peerj.453

Vellinga, W.-P., & Planqué, R. (2015). The Xeno-canto collection and its relation to sound recognition and classification. In *Proceedings of conference and labs of the evaluation forum (CLEF 2015)*, Toulouse, France.

Virbel, M., Hansen, T., & Lobunets, O. (2011). Kivy – A framework for rapid creation of innovative user interfaces. *Workshop-Proceedings Der Tagung Mensch & Computer*. Retrieved from http://kivy.org

Williams, H., Levin, I. I., Ryan Norris, D., Newman, A. E. M., & Wheelwright, N. T. (2013). Three decades of cultural evolution in Savannah sparrow songs. *Animal Behaviour*, *85*, 213–223. https://doi.org/10.1016/j.anbehav.2012.10.028

Wood, C., Sullivan, B., Iliff, M., Fink, D., & Kelling, S. (2011). eBird: Engaging birders in science and conservation. *PLoS Biology*, *9*, e1001220. https://doi.org/10.1371/journal.pbio.1001220

SUPPORTING INFORMATION

Additional supporting information may be found online in the Supporting Information section.

How to cite this article: Searfoss AM, Pino JC, Creanza N. Chipper: Open-source software for semi-automated segmentation and analysis of birdsong and other natural sounds. *Methods Ecol Evol*. 2020;11:524–531. https://doi.org/10.1111/2041-210X.13368