

APPLICATION

pykanto: A python library to accelerate research on wild bird song

Nilo Merino Recalde 

Department of Biology, University of Oxford, Oxford, UK

Correspondence

Nilo Merino Recalde

Email: nilo.recalde@biology.ox.ac.uk

Funding information

Clarendon-Mary Frances Wagley Graduate Scholarship; Edward grey Institute, University of Oxford

Handling Editor: Sarab Sethi

Abstract

1. Studying the vocalisations of wild animals can be a challenge due to the limitations of traditional computational methods, which often are time-consuming and lack reproducibility.
2. Here, I present pykanto, a new software package that provides a set of tools to build, manage, and explore large sound databases. It can automatically find discrete units in animal vocalisations, perform semi-supervised labelling of individual repertoires with a new interactive web app and feed data to deep learning models. pykanto can be used to streamline research on, for example, individual vocal signatures and acoustic similarity between individuals and populations.
3. To demonstrate its capabilities, I put the library to the test on the vocalisations of male great tits in Wytham Woods, near Oxford, UK.
4. The results show that the identities of individual birds can be accurately determined from their songs and that the use of pykanto improves the efficiency and reproducibility of the process.

KEYWORDS

animal vocalisations, bioacoustics, bird song, python

1 | INTRODUCTION

Collecting large amounts of acoustic data from wild bird populations has traditionally been very difficult. Due to technical limitations, studies have often been constrained to tens of individuals and tens, or at best hundreds of vocalisations. But this has changed rapidly within the last decade: compact and economic autonomous recording units, such as the AudioMoth (Hill et al., 2019), now make it possible to collect orders of magnitude more data from many more individuals at once—and to do so much more cheaply. As a direct consequence, many of the computational tools traditionally employed with bioacoustic data have quickly become obsolete: they require manual curation, segmentation, and labelling of data, which are extremely time-consuming and prone to errors.

To illustrate this point, as part of our research on a wild population of great tits *Parus major*, we record around 50,000 songs every year, which translates to well over half a million discrete acoustic units. Any analysis that required finding, labelling and characterising them, if done manually—as is still often the case in wild bird vocalisation research (Beecher et al., 2020; Demko & Mennill, 2018; McLean & Roach, 2020; Pipek et al., 2018; Youngblood & Lahti, 2022)—would take a very long time to complete. This bottleneck, in turn, severely limits researchers' ability to ask questions that require large datasets to answer—such as those about social learning, vocal development, large-scale cultural diversity, and the syntactic organisation of animal vocalisations (Aplin, 2019; Kollmorgen et al., 2020; Lachlan et al., 2018; Sainburg et al., 2019).

This is an open access article under the terms of the [Creative Commons Attribution-NonCommercial](https://creativecommons.org/licenses/by-nc/4.0/) License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited and is not used for commercial purposes.

© 2023 The Author. *Methods in Ecology and Evolution* published by John Wiley & Sons Ltd on behalf of British Ecological Society.

In addition to concerns over the scalability of existing data analysis pipelines, there is now a demand for tools that are freely accessible and promote transparent, reproducible research. Existing proprietary software, such as the widely used Raven Pro (up to \$800, K. Lisa Yang Center for Conservation Bioacoustics, 2019) and Avisoft-SASLab Pro (up to \$2835, Specht, 2002) are difficult to reconcile with contemporary data science practices that rely on open-source programming languages such as R (R Core Team, 2021) and Python (van Rossum, 1995). There exist some excellent open-source options, such as Luscinia (Lachlan, 2016), Sound Analysis Pro (Tchernichovski et al., 2000) and the more recent Koe (Fukuzawa et al., 2020). However, these were generally not designed to cope with large volumes of data, and their reliance on point-and-click graphical user interfaces limits their flexibility and hinders reproducibility.

As a response to the need for scalable and open-source tools for vocalisation data analysis and related issues, the field of bioacoustics has recently started to experiment with a new suite of methods based on deep-learning artificial neural network architectures, the same that excel at, for example, computer vision and speech recognition tasks (Stowell, 2021). Segmentation and annotation pipelines based on deep neural networks have already been shown to work well in laboratory settings, where three conditions hold: (i) acoustic data have a high signal-to-noise ratio, (ii) there are orders of magnitude more examples per vocalisation type than there are vocalisation types and (iii) vocalisations are produced by relatively few individuals (fewer than ten to a few tens) that do so in a stereotyped manner (Coffey et al., 2019; Cohen et al., 2022; Steinfath et al., 2021). Unfortunately, none of these conditions tend to be the case in field studies, and this creates a barrier to the adoption of new methods by researchers working with natural populations.

This is the context in which I present *pykanto* (pronounced pl-'kænt@U). This software library was born of three needs, which can be summarised as follows.

First, it needed to provide the infrastructure necessary to catalogue, explore and label large acoustic datasets collected in often suboptimal field conditions.

Second, it had to serve as a flexible starting point that would allow researchers to perform both traditional analyses (such as extracting hand-picked features from the vocalisations) and to use machine learning algorithms to learn low-dimensional representations of the data (Goffinet et al., 2021; Kollmorgen et al., 2020; Morfi et al., 2021; Sainburg et al., 2020), train classifiers or detect vocalisations in unseen recordings (Cohen et al., 2022; Kahl et al., 2021; Stowell & Plumbley, 2014).

Third, I wanted to build a tool that was free, open source, followed sustainable software practises, and geared toward computational reproducibility and transparency.

2 | *pykanto*: IMPLEMENTATION

pykanto is a software library designed to streamline the process of analysing animal vocalisations. It is programmed in Python and

offers various modules to assist users in their work (see Figure 2). The central module is *pykanto.dataset*, which serves as a database for vocalisations and includes methods to visualise, segment and label them. The *pykanto.signal* module provides tools for signal processing and creating spectrograms, while *pykanto.parameters* contains classes and functions for managing parameters. The web application *pykanto.app* allows users to explore and label large numbers of vocalisations (Figure 1) and *pykanto.plot* provides functions for plotting spectrograms. Finally, *pykanto.utils* includes parsers, I/O tools, custom typing and general computing functions. The documentation for *pykanto* is available at nilomr.github.io/pykanto.

2.1 | Dependencies

pykanto was written in Python 3.8 and tested in Python 3.8, 3.9 and 3.10. Its interactive web application also relies on JavaScript, HTML and CSS. External dependencies are automatically downloaded during package installation (see the [pyproject.toml](#) file for a full list of dependencies).

2.2 | API and documentation

pykanto is a well-documented code library, making it easier to use and contribute to its development. The methods and functions in *pykanto* have clear and concise documentation, including type annotations and descriptions of their intended use. Its API (application programming interface) reference, along with tutorials and practical examples, can be found in the online documentation at nilomr.github.io/pykanto.

2.3 | Reproducibility and open research

pykanto encourages the user to create reproducible data science projects. For example, one of its modules is dedicated to creating consistent project structures, inspired by popular utilities such as [cookiecutter](#). Using the library requires writing simple scripts in Python, which allows every step of the research, from data ingestion to eventual model training and reporting, to be explicitly reproduced. The documentation includes a complete user guide with examples of best practices.

The input and output files use open data formats, and all code is available under the [MIT licence](#) (a simple and very permissive licence). Where applicable, we have followed the guidelines and recommendations of the Software Sustainability Institute, a UK-based facility dedicated to research software sustainability (software.ac.uk).

Many of the processes that *pykanto* carries out are computationally intensive, such as calculating spectrograms, performing operations on large arrays, and running dimensionality reduction and clustering algorithms. High-level, interpreted languages—like R or Python—are notoriously slow: where possible, we have



FIGURE 1 Interface of the interactive web app in pykanto. This app can be used to explore datasets as well as to review and correct automatically assigned class labels in bulk.

optimised performance by both (a) translating functions to optimised machine code at runtime using Numba (Lam et al., 2015) and (b) parallelising tasks using Ray, a state-of-the-art platform for distributed computing (Moritz et al., 2018). As an example, the `segment_into_units()` function can find and segment 20,000 discrete acoustic units in approximately 16 s on a desktop, 8-core machine; a dataset with over half a million (556,472) units takes 132 s on a standard 48-core compute node. If pykanto detects a suitable GPU unit and the optional dependencies are installed, algorithms such as UMAP (McInnes et al., 2018) switch to their GPU implementation, which provides a 15–100x speedup (Nolet et al., 2021; Raschka et al., 2020). The library has a module dedicated to making it easy for users to run their scripts in a high-performance computing context (for example, a university compute cluster), and its documentation includes examples of configuration and submission scripts.

2.4 | Limitations

This final section discusses some of the main limitations of pykanto. Although it will hopefully offer a flexible solution for researchers, it is also limited in important ways.

Limitation 1: Vocalisation unit segmentation via the very simple amplitude thresholding algorithm will not work well with species whose vocalisations vary greatly in amplitude, or with very

noisy datasets. In those cases, and depending on data volume, segmentation might better be performed either manually or in a semi-automated way. For example, one could use *chipper* (Searfoss et al., 2020) or train a neural network like *TweetyNet*, (Cohen et al., 2022) on a manually annotated subset of the data.

Limitation 2: pykanto has been tested on species that produce vocalisations made up of a small or moderate number of different but distinct elements (variously referred to as notes or syllables). It will be useful for researchers working with any species, but the automatic part of the clustering process will work increasingly poorly with those that have a large number of very variable elements. This is true of any clustering method: they will fail or produce spurious results if variation in the data is continuous.

Limitation 3: The library does not include methods to train models intended to find analysable vocalisations in long recordings of entire soundscapes. This is a particularly challenging problem (Priyadarshani et al., 2018) without a universal solution. However, pykanto can be used to generate and organise the training data required by these models (Kahl et al., 2021; Stowell & Plumbly, 2014; Stowell et al., 2019), and to work with their output annotations.

Limitation 4: pykanto is intended as a flexible solution for managing and preparing animal vocalisation data for further analysis. It provides tools that can save researchers a great deal of time while making analysis pipelines more reproducible. However, it does not implement any specific analysis or feature extraction

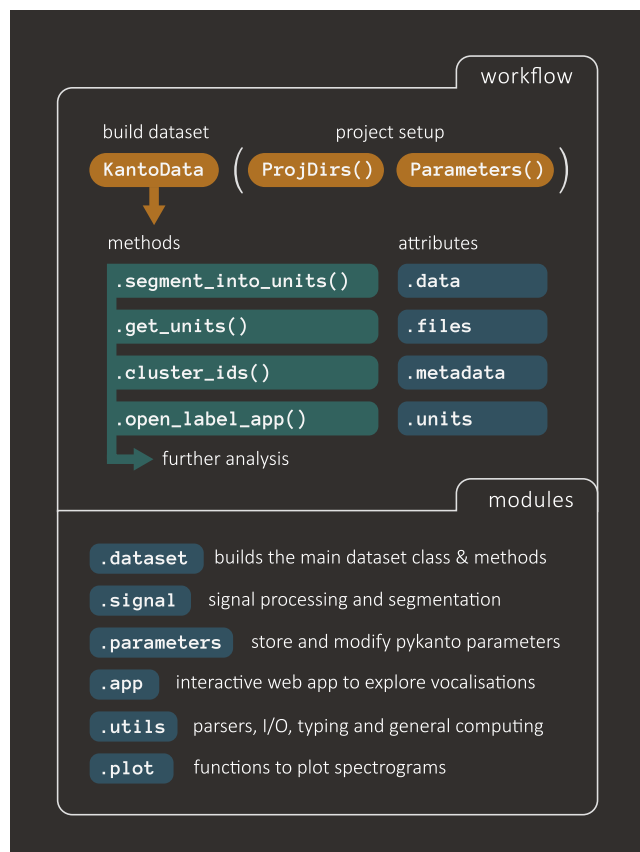


FIGURE 2 pykanto is written around a central dataset class, *KantoData*, which provides methods to segment, visualise and label vocalisations. The library contains six modules with functions and classes to carry out common tasks in animal vocalisation analysis.

methods, since these will vary greatly by use case. This means that researchers using the library as part of their work will need to either have or develop familiarity with bioacoustic analysis and scripting in Python.

3 | USING pykanto: CAN INDIVIDUAL BIRDS BE IDENTIFIED FROM THEIR SONGS?

I now provide a worked example of how pykanto can be used to help answer real research questions vocalisations—bird song in this case:

3.1 | Introduction

Great tits are small, short-lived birds (average lifespan: 1.9 years) that sing acoustically simple yet highly diverse songs. In Wytham Woods, Oxfordshire (UK), a population of these birds has been the focus of a long-term study that is now in its 75th year. For the past 3 years, we have recorded the song repertoires of hundreds of individual males when they sing close to their nest before their partner begins laying. With the help of these data, we are trying

to answer questions about song learning and cultural change in natural populations.

To do this, we first need to know which individuals are present in the breeding population for the first time, and which were already around in previous years. However, individual survival over the winter months is low and detection by traditional means—such as mist-netting or identification in the nest—is imperfect. So we would first like to test whether individual birds can be identified based on their songs alone, and then quantify how much variation in song types occurs within and between years.

Our example dataset consists of 5293 songs from 12 males that were known (from physical recaptures) to be present in the breeding population in two different years, 2020 and 2021. Although this is a small subset of our data, it is large enough that it would still take weeks to process and analyse using traditional methods. We demonstrate the use of pykanto to (a) organise, segment and label the dataset, and (b) prepare it so that we can train a deep neural network to recognise song types. The entire process, which takes under an hour to complete, can be computationally reproduced using its [dedicated repository](#). The repository includes raw data, auxiliary scripts and detailed instructions. Below is a short narrative description of the process.

3.2 | Data collection

Most great tits in our population nest in nest boxes with known locations. Every year, fieldworkers record the identities of breeding males and females, clutch initiation and egg-hatching dates, clutch size and fledgling success using standardised protocols. A significant number of birds in the population are fitted with a unique British Trust for Ornithology (BTO) metal leg ring as nestlings or adults. During the breeding season (March–June), great tit pairs are socially monogamous and protect territories around their nest boxes (Hinde, 1952).

We collected data during the breeding seasons of 2020 and 2021, from early April to late May, using a dense sampling design with multiple recorders placed in nest boxes throughout the study site. Fieldworkers checked every nest box in the study site at least once a week before and during egg laying, which can last from one to 14 days (Perrins, 1965). Once a nest box was believed to be in use by a great tit, we placed an autonomous sound recorder in its vicinity, either in the same tree or in a suitable neighbouring tree. We left each recorder in the same location for at least three consecutive days before moving it to a different nest box. Throughout the recording period, we relocated 20 recorders every day.

We used 60 AudioMoth recorders (Hill et al., 2019) in 2021 and 30 in 2020, which were housed in waterproof custom-built enclosures. Recording began about an hour before sunrise (from 05:36 to 04:00 UTC during the recording period) and consisted of seven 60-min recordings with a sample rate of 48 kHz. Since the recording process was automated, there is a possibility that some of the songs recorded in the immediate vicinity of a given nest box do not belong

to the focal bird. To reduce the risk of false positives, we discarded recordings with more than one vocalising bird, unless one was distinctly louder than the others. We also discarded all songs with a maximum amplitude below -16dB , calculated as $20\log_{10}\left(\frac{A}{A_0}\right)$, with $A = 5000$ and $A_0 = 32767$ (the maximum value for 16-bit digital audio). This threshold was determined from the observation that, in cases where we had simultaneous recordings of close neighbours from the centres of their respective territories, an amplitude cut-off greater than 4000 always separated a focal bird from its neighbours. It should be noted that these values are not calibrated and are relative to the recording equipment and settings used, as well as other factors such as sound directionality and vegetation cover.

3.3 | Running the analysis

3.3.1 | Installation

pykanto can be used outside a virtual environment, but this is not encouraged. Using clean environments for each project will allow you to avoid dependency issues. Once inside a new environment with Python 3.8 or above, you can install pykanto by simply running pip install pykanto, then install the package containing this example. See detailed installation and use instructions in the [.README](#).

3.3.2 | Creating a new project and dataset

Our first step will be to define a directory structure for our project and a ProjDirs object to hold everything together. Then, we can test and set adequate parameters for our dataset. These include things like low- and high-cut filters, spectrogram settings, amplitude thresholding and whether the analysis will be carried out at the song or note level. The data folder in the project already contains .wav audio files and their corresponding .json with annotations; therefore, we can create a KantoData instance: this will be our database.

3.3.3 | Segmenting songs and using the interactive app

Then, using the .segment_into_units() method, we find segment onsets, offsets, unit and silence durations and add them to KantoData, data, the main data frame in our database object. At this point, we could already carry out most of the analyses common in the bird song literature, for example, by extracting some simple acoustic parameters from the segmented data. Instead, we want to preserve all the temporal and spectral information that is available in the spectrograms to train a more accurate classifier.

The next step is to compute and store spectrograms for each unit under examination, and then reduce their dimensionality and group them into clusters. This can be achieved by using the .get_units() and .cluster_ids() methods, which employ algorithms such as

UMAP (McInnes et al., 2018) and HDBSCAN (McInnes et al., 2017). Afterwards, we can launch the interactive web app by calling .open_label_app(). Using this app, we can review the automatic labels for up to tens of thousands of vocalisations at once, splitting or combining clusters as needed. Once completed, we will have a fully annotated dataset, which can be divided into training and testing sets and exported as labelled spectrograms using pykanto.

3.3.4 | Training a convolutional neural network classifier

Our goal is to generate compressed representations of songs that can facilitate comparisons and identification of those sung by the same individual, even in the presence of variations in performance and noise. To do this, we can train a model to distinguish between different song types, which are categorically distinct within individual song repertoires, without providing the model with information about who sang them.

In this example, we use weights from a pre-trained ResNet50 backbone (He et al., 2015) and gradually unfreeze the earlier layers of the network during training. By doing so, we can fine-tune the network to attain better performance in our task, while still benefiting from the weights learned on a much larger dataset (Zhuang et al., 2021).

The distribution of song sample sizes per individual approximately follows a power law, so there is a very large amount of data for a few birds and very little for most. This imbalance of data is problematic for learning algorithms, as they may develop a bias toward larger classes and perform poorly on rarer ones. There are different ways to deal with this (see, e.g. Krawczyk, 2016; Thabtah et al., 2020); however, to keep things simple, here we will just under-sample majority classes so that all birds have the smallest common sample size for each song type.

Background noise can also bias our analysis. Each bird's acoustic environment is unique, and the network may learn to distinguish between songs based on this noise rather than the signal of interest. To address this issue, we remove most background noise by thresholding the spectrograms, taking advantage of the difference in amplitude between the focal bird singing near the recording and its acoustic background. Additionally, we apply a series of data augmentation techniques during the training process to prevent over-fitting and ensure that the algorithm does not memorise irrelevant or highly variable elements. These techniques include semi-random cropping in the time domain, blurring and sharpening, random erasing of parts of the spectrogram, and contrast and brightness changes. After training the model, we can verify that background noise is not causing bias by checking whether songs recorded in the same location are classified as more similar than expected by chance. In addition, we can create class activation maps to identify the regions of the spectrograms used by the model to generate predictions.

Once the model is trained, we can assess its ability to classify unseen songs drawn from the held-out dataset, which can reach an

accuracy of around 92%. As described below, most of the remaining 8% can be attributed to confusion between the same song types sung by the same birds in different years.

Finally, we can use the model to extract a compressed representation of each song in the entire dataset. This is achieved by passing each song through the trained network and obtaining the output from the last hidden layer. This output consists of a feature vector that captures the most relevant information to distinguish between different types of songs; these feature vectors can then be used to determine whether two songs are very similar. In our case, we do this by taking the inverse of the cosine distance between each pair to build a similarity matrix.

The repository that supports this paper contains a streamlined way of doing this using PyTorch 276 (2019) and PyTorch Lightning (2019) that can be easily adapted for use with other datasets.

4 | RESULTS & DISCUSSION

After calculating the similarity scores between all pairs of songs, we grouped them according to song type, the bird that sang them, and the year they were sung. For each bird in the first year, we identified the individual who sang the song with the highest similarity to any within its repertoire during the following year. We found that we were able to successfully identify the correct bird in all cases, even though the baseline probability was only 2.27% (1 out of 44 song types). This suggests that we would have been able to re-identify the individuals even if they had not been observed or captured again.

As shown in Figure 3, the highest similarity values correspond to comparisons of the same song types within years and birds. The similarity between the same song types sung by the same bird across different years is consistently higher than that between different

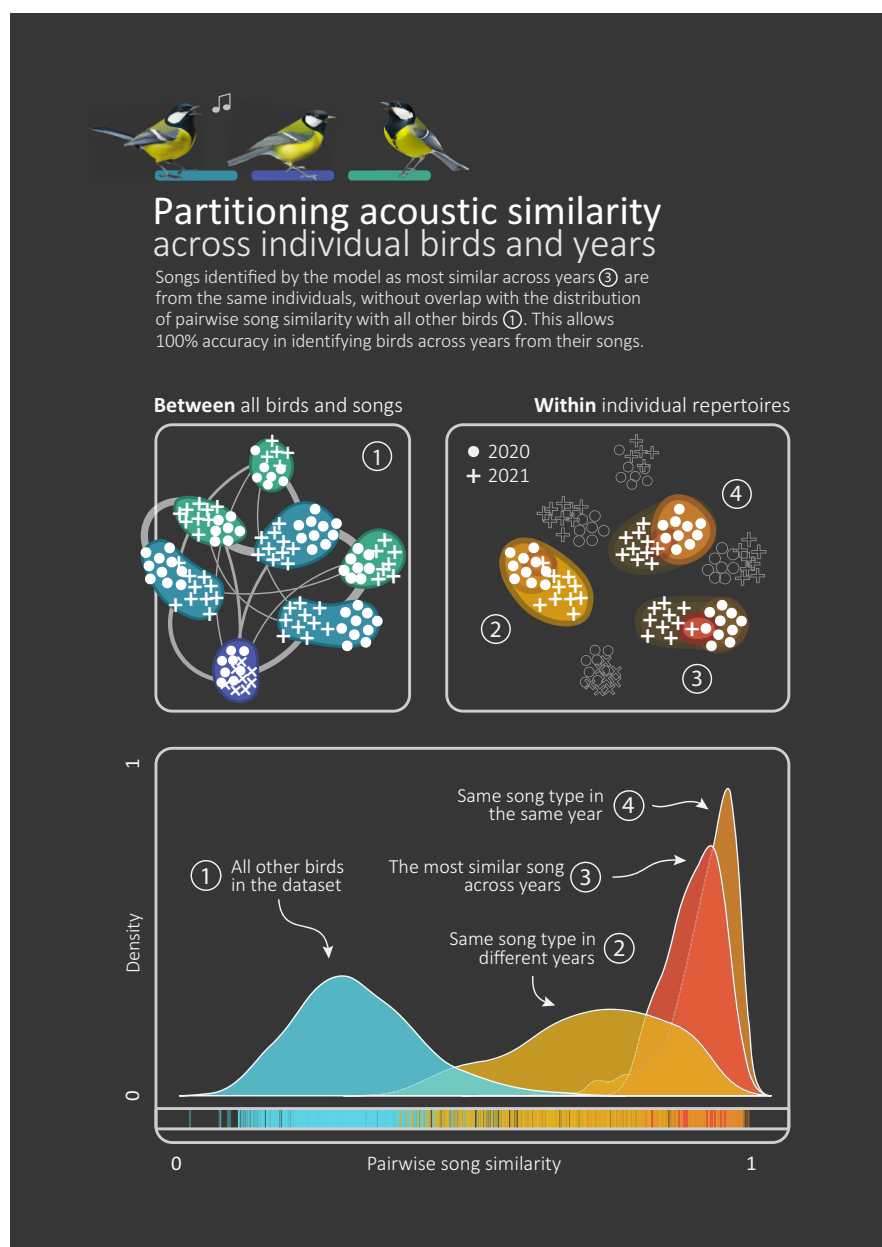


FIGURE 3 (From left to right and top to bottom.) First, we calculate the pairwise similarity between the songs of all birds, which provides a baseline distribution of similarity in the population (1). Then, we compare songs from the same bird in different years (2), find the pair of songs that are most similar across years (3), and compare songs within the same year (4). The probability density estimates in the bottom panel show how pairwise similarities in (3) allow us to re-identify birds across as they do not overlap with any other birds (1).

birds, even when some song types were shared by individuals: this means that individual vocal signatures are at least partly maintained across their lifespan.

The conclusions drawn from this analysis are limited by the small size of the dataset: including more birds would likely lead to noisier results, as it increases the chances of finding a second bird with even more similar songs. Nonetheless, in combination with other information (such as spatial location), they might allow high-confidence identification of individuals between years without physical capture.

This example illustrates how *pykanto* can be used to help address a specific research question. The model-based feature vectors used to describe each song can be imported back into the *KantoData* database as a new column, enabling a wide range of research possibilities while maintaining a clear project structure.

AUTHOR CONTRIBUTIONS

Nilo Merino Recalde wrote the software library and its documentation, collected the data, conducted the analyses and wrote the manuscript.

ACKNOWLEDGEMENTS

I thank Ben Sheldon and the Sheldon lab for their support and patience. Ben Sheldon, Carys Jones and Andrea Estandia provided useful comments on a draft of this manuscript. Carys Jones and Antoine Vansse tried early versions of the interactive app in *pykanto* and provided valuable feedback. Some of the methods in *pykanto* are directly inspired by or adapted from Sainburg et al. (2020). I have indicated where this is the case in the relevant method's docstring. The dereverberation function is based on code by Robert Lachlan that is part of *Luscinia* (Lachlan, 2016), a software for bioacoustic archiving, measurement and analysis. Please consider citing these two publications if you use *pykanto* on your own projects. I have learnt a great deal about packaging and developing in Python by browsing the structure of existing open source projects, for example, some by David Nicholson (@NickleDave). I only became aware of *VocalPy*, a project that aims to “develop an ecosystem of interoperable packages” for “computational vocal communication and learning research” when I had already written most of *pykanto*, but eventually, I would like to re-write *pykanto* to increase compatibility with it: standardisation is direly needed in the field and I do not want to contribute to the chaos. This work was supported by a Clarendon-Mary Frances Wagley Graduate Scholarship and an EGI scholarship to Nilo Merino Recalde and made use of the University of Oxford Advanced Research Computing Facility (Richards, 2015).

CONFLICT OF INTEREST STATEMENT

The author declares no conflict of interest.

PEER REVIEW

The peer review history for this article is available at <https://www.webofscience.com/api/gateway/wos/peer-review/10.1111/2041-210X.14155>.

DATA AVAILABILITY STATEMENT

I distribute *pykanto* with three sample datasets that are used to run unit tests and as examples in the documentation (Merino Recalde, 2023b). **Great tit songs:** 20 songs recorded from male birds during the dawn chorus in a population in Oxford, UK. Recorded by the author and accessible at [pykanto/data/great_tit](https://pykanto.com/data/great_tit). **European storm-petrel purr songs:** Two males singing from burrows in the Shetland and Faroe islands. Source: [XC46092](https://doi.org/10.1016/j.anbehav.2018.05.001) (© Dougie Preston), [XC663885](https://doi.org/10.1016/j.anbehav.2018.05.001) (© Simon S. Christiansen). Under [CC BY-NC-ND 2.5 licence](https://creativecommons.org/licenses/by-nc-nd/2.5/). **Bengalese finch songs:** Recordings from two isolated Bengalese finches. Originally published in Tachibana, Koumura and Okanoya (Tachibana et al., 2015), data can be accessed at [OSF](https://osf.io/8qz3p/). They can be found under *pykanto/data* when you install the package, as well as in the [GitHub repository](https://github.com/merino/pykanto). Additionally, the worked example in this article uses 5293 songs from male great tit songs recorded by the author between 2020 and 2021 in Wytham Woods, Oxfordshire, UK. They are available from [pykanto-example/data](https://github.com/merino/pykanto-example) on GitHub, along with detailed meta-data (Merino Recalde, 2023a).

CODE AVAILABILITY

The latest version of *pykanto* is available from PyPI (pip install *pykanto*) and its source repository (github.com/pykanto). See the repository for detailed installation instructions. *pykanto* and the example in this article rely on the following open-source scientific libraries or tools: *numpy* (Harris et al., 2020), *scipy* (Virtanen et al., 2020), *pandas* (The Pandas Development Team, 2023), *numba* (Lam et al., 2015), *pytorch* (Paszke et al., 2019), *torchvision* (TorchVision Maintainers and Contributors, 2016), *pytorch lightning* (Falcon and The PyTorch Lightning Team, 2019), *tqdm* (da Costa-Luis, 2019), *ray* (Moritz et al., 2018), *soundfile* (Bechtold & Geier, 2022), *umap* (McInnes et al., 2018), *joblib* (Joblib Development Team, 2020), *hdbscan* (McInnes et al., 2017), *seaborn* (Waskom, 2021), *scikit-image* (van der Walt et al., 2014), *librosa* (McFee et al., 2015), *bokeh* (Bokeh Development Team, 2018), *ujson* (van Kemenade et al., 2023), *psutil* (Rodola, 2023) and *attrs* (Schlawack, 2019).

ORCID

Nilo Merino Recalde  <https://orcid.org/0000-0003-3903-1288>

REFERENCES

- Aplin, L. M. (2019). Culture and cultural evolution in birds: A review of the evidence. *Animal Behaviour*, 147, 179–187. <https://doi.org/10.1016/j.anbehav.2018.05.001>
- Bechtold, B., & Geier, M. (2022). Soundfile.
- Beecher, M. D., Akçay, Ç., & Campbell, S. E. (2020). Birdsong learning is mutually beneficial for tutor and student in song sparrows. *Animal Behaviour*, 166, 281–288. <https://doi.org/10.1016/j.anbehav.2020.05.015>
- Bokeh Development Team. (2018). Bokeh: Python Library for Interactive Visualization.
- Coffey, K. R., Marx, R. G., & Neumaier, J. F. (2019). DeepSqueak: A deep learning-based system for detection and analysis of ultrasonic vocalizations. *Neuropsychopharmacology*, 44(5), 859–868. <https://doi.org/10.1038/s41386-018-0303-6>

- Cohen, Y., Nicholson, D. A., Sanchioni, A., Mallaber, E. K., Skidanova, V., & Gardner, T. J. (2022). Automated annotation of birdsong with a neural network that segments spectrograms. *eLife*, 11, e63853. <https://doi.org/10.7554/eLife.63853>
- da Costa-Luis, C. O. (2019). Tqdm: A fast, extensible progress meter for python and CLI. *Journal of Open Source Software*, 4(37), 1277. <https://doi.org/10.21105/joss.01277>
- Demko, A. D., & Mennill, D. J. (2018). Rufous-capped warblers *Basileuterus rufifrons* show seasonal, temporal and annual variation in song use. *Ibis*, 161, 481–494. <https://doi.org/10.1111/ibi.12666>
- Falcon, W., & The PyTorch Lightning Team. (2019). PyTorch Lightning. <https://doi.org/10.5281/zenodo.3828935>
- Fukuzawa, Y., Webb, W. H., Pawley, M. D. M., Roper, M. M., Marsland, S., Brunton, D. H., & Gilman, A. (2020). Koe: Web-based software to classify acoustic units and analyse sequence structure in animal vocalizations. *Methods in Ecology and Evolution*, 11(3), 431–441. <https://doi.org/10.1111/2041-210X.13336>
- Goffinet, J., Brudner, S., Mooney, R., & Pearson, J. (2021). Low-dimensional learned feature spaces quantify individual and group differences in vocal repertoires. *eLife*, 10, e67855. <https://doi.org/10.7554/eLife.67855>
- Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., ... Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825), 357–362. <https://doi.org/10.1038/s41586-020-2649-2>
- He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. *arXiv* <https://doi.org/10.48550/arXiv.1512.03385>
- Hill, A. P., Prince, P., Snaddon, J. L., Doncaster, C. P., & Rogers, A. (2019). AudioMoth: A low-cost acoustic device for monitoring biodiversity and the environment. *HardwareX*, 6, e00073. <https://doi.org/10.1016/j.ohx.2019.e00073>
- Hinde, R. A. (1952). The behaviour of the great tit (*Parus major*) and some other related species. *Behaviour Supplement*, 2, III–201. <https://doi.org/10.2307/4510389>
- Joblib Development Team. (2020). Joblib: Running python functions as pipeline jobs.
- K Lisa Yang, Center for Conservation Bioacoustics. (2019). *Raven Pro: Interactive sound analysis software*. The Cornell Lab of Ornithology.
- Kahl, S., Wood, C. M., Eibl, M., & Klinck, H. (2021). BirdNET: A deep learning solution for avian diversity monitoring. *Ecological Informatics*, 61, 101236. <https://doi.org/10.1016/j.ecoinf.2021.101236>
- Kollmorgen, S., Hahnloser, R. H. R., & Mante, V. (2020). Nearest neighbours reveal fast and slow components of motor learning. *Nature*, 577(7791), 526–530. <https://doi.org/10.1038/s41586-019-1892-x>
- Krawczyk, B. (2016). Learning from imbalanced data: Open challenges and future directions. *Progress in Artificial Intelligence*, 5(4), 221–232. <https://doi.org/10.1007/s13748-016-0094-0>
- Lachlan, R. F. (2016). *Luscinia: A bioacoustics Analysis computer program*. <https://rflachlan.github.io/Luscinia/>
- Lachlan, R. F., Ratmann, O., & Nowicki, S. (2018). Cultural conformity generates extremely stable traditions in bird song. *Nature Communications*, 9(1), 2417. <https://doi.org/10.1038/s41467-018-04728-1>
- Lam, S. K., Pitrou, A., & Seibert, S. (2015). Numba: A LLVM-based python JIT compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC LLVM'15* (pp. 1–6). Association for Computing Machinery. <https://doi.org/10.1145/2833157.2833162>
- McFee, B., Raffel, C., Liang, D., Ellis, D., McVicar, M., Battenberg, E., & Nieto, O. (2015). Librosa: Audio and music signal analysis in python. In *Python in Science Conference Austin* (pp. 18–24). SciPy 2015. <https://doi.org/10.25080/Majora-7b98e3ed-003>
- McInnes, L., Healy, J., & Astels, S. (2017). hdbSCAN: Hierarchical density based clustering. *The Journal of Open Source Software*, 2(11), 205. <https://doi.org/10.21105/joss.00205>
- McInnes, L., Healy, J., & Melville, J. (2018). UMAP: Uniform manifold approximation and projection for dimension reduction. *The Journal of Open Source Software*, 3(29), 861.
- McLean, L. C., & Roach, S. P. (2020). Markov dependencies in the song syntax of hermit thrush (*Catharus guttatus*). *Journal of Ornithology*, 162, 469–476. <https://doi.org/10.1007/s10336-020-01840-2>
- Merino Recalde, N. (2023a). nilomr/pykanto-example: Publication-ready release of pykanto (use example). Zenodo. <https://doi.org/10.5281/zenodo.7941355>
- Merino Recalde, N. (2023b). nilomr/pykanto: V0.1.5. Zenodo <https://doi.org/10.5281/zenodo.7888656>
- Morfi, V., Lachlan, R. F., & Stowell, D. (2021). Deep perceptual embeddings for unlabelled animal sound events. *The Journal of the Acoustical Society of America*, 150(1), 2–11. <https://doi.org/10.1121/10.0005475>
- Moritz, P., Nishihara, R., Wang, S., Tumanov, A., Liaw, R., Liang, E., Elibol, M., Yang, Z., Paul, W., Jordan, M. I., & Stoica, I. (2018). Ray: A distributed framework for emerging AI applications. In *Proceedings of the 13th USENIX Conference on Operating Systems Design and implementation OSDI'18* (pp. 561–577). USENIX Association.
- Nolet, C. J., Lafargue, V., Raff, E., Nanditale, T., Oates, T., Zedlewski, J., & Patterson, J. (2021). Bringing UMAP closer to the speed of light with GPU acceleration. *arXiv:200800325 [cs, stat]*.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., ... Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems* 32 (pp. 8024–8035). Curran Associates Inc.
- Perrins, C. M. (1965). Population fluctuations and clutch-size in the great tit, *Parus major* L. *Journal of Animal Ecology*, 34(3), 601–647.
- Pipek, P., Petrusková, T., Petrušek, A., Diblíková, L., Eaton, M. A., & Pyšek, P. (2018). Dialects of an invasive songbird are preserved in its invaded but not native source range. *Ecography*, 41(2), 245–254. <https://doi.org/10.1111/ecog.02779>
- Priyadarshani, N., Marsland, S., & Castro, I. (2018). Automated birdsong recognition in complex acoustic environments: A review. *Journal of Avian Biology*, 49(5), jav-01447. <https://doi.org/10.1111/jav.01447>
- R Core Team. (2021). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing.
- Raschka, S., Patterson, J., & Nolet, C. (2020). Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. *arXiv Preprint arXiv:200204803*.
- Richards, A. (2015). University of Oxford advanced research computing. <https://doi.org/10.5281/zenodo.22558>
- Rodola, G. (2023). Psutil.
- Sainburg, T., Theilman, B., Thielk, M., & Gentner, T. Q. (2019). Parallels in the sequential organization of birdsong and human speech. *Nature Communications*, 10(1), 3636. <https://doi.org/10.1038/s41467-019-11605-y>
- Sainburg, T., Thielk, M., & Gentner, T. Q. (2020). Finding, visualizing, and quantifying latent structure across diverse animal vocal repertoires. *PLOS Computational Biology*, 16(10), e1008228. <https://doi.org/10.1371/journal.pcbi.1008228>
- Schlawack, H. (2019). Attrs.
- Searfoss, A. M., Pino, J. C., & Creanza, N. (2020). Chipper: Open-source software for semi-automated segmentation and analysis of bird-song and other natural sounds. *Methods in Ecology and Evolution*, 11(4), 524–531. <https://doi.org/10.1111/2041-210X.13368>
- Specht R. (2002). *Avisoft-SASLab Pro* (2002, pp. 1–723). Avisoft Bioacoustics.
- Steinfath, E., Palacios-Muñoz, A., Rottschäfer, J. R., Yuezak, D., & Clemens, J. (2021). Fast and accurate annotation of acoustic

- signals with deep neural networks. *eLife*, 10, e68837. <https://doi.org/10.7554/eLife.68837>
- Stowell, D. (2021). Computational bioacoustics with deep learning: A review and roadmap. arXiv:211206725 [cs, eess, q-bio].
- Stowell, D., & Plumbley, M. D. (2014). Automatic large-scale classification of bird sounds is strongly improved by unsupervised feature learning. *PeerJ*, 2, e488. <https://doi.org/10.7717/peerj.488>
- Stowell, D., Wood, M. D., Pamula, H., Stylianou, Y., & Glotin, H. (2019). Automatic acoustic detection of birds through deep learning: The first bird audio detection challenge. *Methods in Ecology and Evolution*, 10(3), 368–380. <https://doi.org/10.1111/2041-210X.13103>
- Tachibana, R. O., Koumura, T., & Okanoya, K. (2015). Variability in the temporal parameters in the song of the Bengalese finch (*Lonchura striata* var. *Domestica*). *Journal of Comparative Physiology A*, 201(12), 1157–1168. <https://doi.org/10.1007/s00359-015-1046-z>
- 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(6), 1167–1176. <https://doi.org/10.1006/anbe.1999.1416>
- Thabtah, F., Hammoud, S., Kamalov, F., & Gonsalves, A. (2020). Data imbalance in classification: Experimental evaluation. *Information Sciences*, 513, 429–441. <https://doi.org/10.1016/j.ins.2019.11.004>
- The Pandas Development Team. (2023). Pandas-dev/pandas: Pandas. Zenodo. <https://doi.org/10.5281/zenodo.7549438>
- TorchVision Maintainers and Contributors. (2016). *TorchVision: PyTorch's computer vision library*. GitHub.
- van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., Yu, T., & the scikit-image contributors. (2014). Scikit-image: Image processing in python. *PeerJ*, 2, e453. <https://doi.org/10.7717/peerj.453>
- van Kemenade, H., Woodsend, B., Hamrén, J., JustAnotherArchivist, Crall, J., O'Mahony, K., Lay, E. L., HMychev, M., Swenson, D. W. H., Dawborn, T., Garcia-Armas, M., Ceccon, R., Moiron, J., Górný, M., NaN-git, Naoki, I., Beasley, B., Zhou, L., Borisov, M., ... CozyDoomer. (2023). Ultrajson. Zenodo. <https://doi.org/10.5281/zenodo.7510698>
- van Rossum, G. (1995). *Python reference manual*. CWI.
- Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., ... SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 17, 261–272. <https://doi.org/10.1038/s41592-019-0686-2>
- Waskom, M. L. (2021). Seaborn: Statistical data visualization. *Journal of Open Source Software*, 6(60), 3021. <https://doi.org/10.21105/joss.03021>
- Youngblood, M., & Lahti, D. C. (2022). Content bias in the cultural evolution of house finch song. *Animal Behaviour*, 185, 37–48. <https://doi.org/10.1016/j.anbehav.2021.12.012>
- Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., & He, Q. (2021). A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1), 43–76. <https://doi.org/10.1109/JPROC.2020.3004555>

How to cite this article: Merino Recalde, N. (2023). pykanto: A python library to accelerate research on wild bird song. *Methods in Ecology and Evolution*, 14, 1994–2002. <https://doi.org/10.1111/2041-210X.14155>