Received: 16 February 2022 | Accepted: 20 July 2022

DOI: 10.1111/2041-210X.13964

Methods in Ecology and Evolution BRITISH ECOLOGICAL SOCIETY

APPLICATION

soundClass: An automatic sound classification tool for biodiversity monitoring using machine learning

Bruno Silva[1] | Frederico Mestre[1] | Sílvia Barreiro[1] | Pedro J. Alves[2] | José M. Herrera[1]

[1]Mediterranean Institute for Agriculture, Environment and Development (MED), University of Évora (UE), Évora, Portugal

[2]PLECOTUS, Estudos Ambientais, Pombal, Portugal

Correspondence
Bruno Silva
Email: bmsasilva@gmail.com

Funding information
Fundação para a Ciência e a Tecnologia, Grant/Award Number: IF/00001/2015, PTDC/BIA-CBI/1365/2020 and SFRH/BD/137803/2018; H2020 European Institute of Innovation and Technology, Grant/Award Number: 862480

Handling Editor: Luis Cayuela

Abstract

1. Passive acoustic monitoring, a non-invasive technique, is increasingly used to study animal populations and habitats at much larger spatial and temporal scales than standard methods. However, easy to apply tools for reliable detection and classification of signals of interest among hundreds or even thousands of hours of recording are still lacking.
2. We introduce the R package soundClass, a tool to train convolutional neural networks, and employ them to classify sound events in recordings. soundClass provides a sound event classification pipeline, from annotating recordings to automating trained networks usage in real-life situations.
3. We illustrate the package functionality on bat echolocation calls, bird songs and whale echolocation clicks, showing that the package can be used to train networks for several types of sound events, taxonomic groups and environments; and exemplify its application.
4. This tool facilitates the creation and usage of trained networks and was developed with a strong focus on graphical user interfaces to be used by non-specialist scientists in statistics and programming.

KEYWORDS
biodiversity monitoring, convolutional neural networks, machine learning, passive acoustic monitoring, sound classification

1 | INTRODUCTION

Passive acoustic monitoring (PAM) is increasingly used to study animal populations and habitats (Browning et al., 2017). Concurrently, rapid technological advances are prompting the appearance of low-cost recording hardware, better storing and raw data management as well as extended battery life. These advances lead, for example, to the possibility of deploying multiple recording devices simultaneously in the field for progressively longer periods of time. PAM is a non-invasive technique enabling the detection of species presence and richness, the monitoring of population density and inferring several environmental metrics (acoustic diversity, acoustic entropy, etc.) at much larger spatial and temporal scales than standard methods (Merchant et al., 2015). However, it is important to note that the target species must make detectable sounds that can be identifiable to a useful class (a set of cases sharing common characteristics, e.g. taxonomic group, species, behaviour) for PAM to be effectively employed (Browning et al., 2017).

Adopting PAM over broad spatial and temporal scales, however, still faces a pressing issue: the reliable detection and classification

of acoustic signals of interest among hundreds or even thousands of recording hours. Hand-browsing of recordings is in many cases infeasible and the diversity of acoustic signals and environmental soundscape hinders the possibility of a single solution (algorithm) for every problem. In ecology, project-specific or site-specific audio classification tools are commonly built and trained on data representative of the actual survey datasets. Such tools, besides lacking transferability, are extremely time-consuming to develop and rely on statistics and programming experts to be developed, employed and maintained (Gibb et al., 2019).

Existing proprietary software offer a growing range of inbuilt automated tools for large taxonomic groups and geographical regions. However, although user-friendly, they are pay-per-use and present clear risks due to poorly reported underlying methods, making the transferability to novel datasets unclear. Alternative, open-source or free-ware tools have no cost but are generally focused on specific taxonomic groups. Within the R environment, which is one of the most used and flexible open-source languages for statistical analysis, there are several packages for analysing acoustic datasets but they tend to require medium to advanced programming skills to be effective (Browning et al., 2017; Gibb et al., 2019). Moreover, the ability to scale up is essential as automatic recording stations tend to generate huge amounts of data. This implies that any feasible method must work without manual intervention, in particular in the detection of meaningful temporal segments in audio recordings—a process known as segmentation (Stowell & Plumbley, 2014)—a feature many software/packages lacks.

We introduce a new R package, soundClass, developed to tackle the limitations of existing tools for PAM projects. By using convolutional neural networks (CNNs) as the base algorithm, manual segmentation is unnecessary. Moreover, the features that discriminate between classes are learned automatically by the network, thereby effectively extending the functionalities of the package for non-specialist scientists in the vocalizations of the taxonomic group being analysed or in general features of sound analysis. soundClass is open source and was developed to implement in a single tool the ability to (a) manage and annotate recordings (i.e. create timed-text notes) in a database with *structured query language* (SQL) capabilities, (b) be generic so that it can be employed for several taxonomic groups, (c) use CNNs for creating custom classification tools, as they can easily evolve from new data for better transferability to novel datasets, and (d) be used by non-specialist scientists in statistics and programming due to its user-friendly graphical user interfaces (GUIs).

2 | SOUNDCLASS DESCRIPTION AND WORKFLOW

The main purpose of soundClass is to provide a sound classification pipeline, from annotating sound events in recordings to automating the usage of fitted CNN models in real-life situations. This package is aimed at biologists and ecologists working with sound events that could benefit greatly from machine learning algorithms applied to their research. Using the package requires a pre-compiled collection of recordings and it can be employed for (Figure 1):

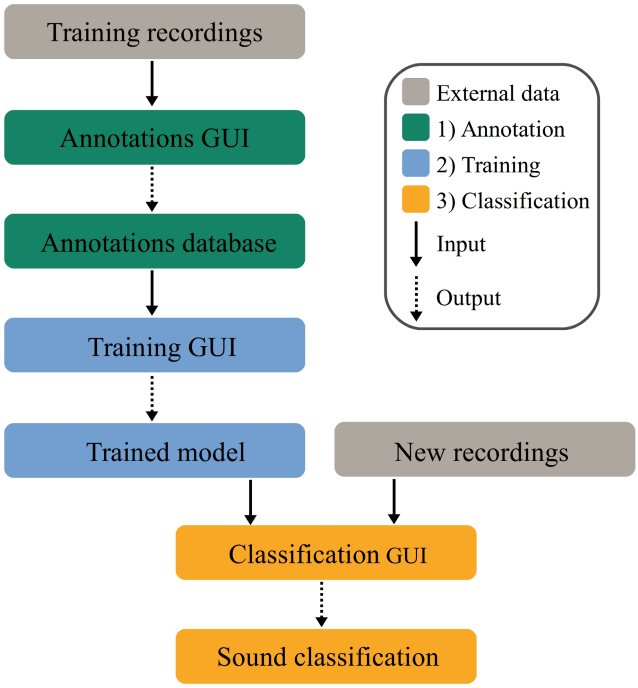

FIGURE 1 Package usage workflow. Training recordings are annotated (1) and a training dataset is created and used to fit a convolutional neural network (CNN) (2). The fitted CNN is then used to classify new recordings (3).

1. Annotation: create a database of annotated recordings,
2. Training: prepare training data from annotated recordings and fit/train CNN models,
3. Classification: automate the use of the fitted CNN model for classifying new recordings.

To be accessible to a larger number of users, soundClass was developed with a focus on GUIs and does not require advanced coding or statistical skills. Nevertheless, advanced users may also use the package through scripting for training and classification, which provides a reproducibility advantage. The functions composing the package and a brief description are found in Tables S1 and S2.

1. Annotation: Create a database of annotated recordings

Sound recordings generally have many sound events. Therefore, annotating sound events in recordings with the correct class label is the first step for obtaining a trained model for classification purposes. To automate the classification process with the fitted CNN in new, non-segmented recordings (i.e. recordings where meaningful temporal segments for classification are unknown), an 'irrelevant' class (referring to sound events that are not the target of our research) must be included in the training data (Marques et al., 2013; Stowell et al., 2019). This class should include typical 'background' sounds such as environmental noise (e.g. rustling foliage, wind, rain), other animal vocalizations (e.g. insects, birds, mammals) or anthropogenic

noise (Mac Aodha et al., 2018). Please note that this class must have the label '0' to be correctly processed by the package.

To facilitate the annotation of sound files for posterior processing, the 'Annotations GUI' converts the original sound wave with the Fast Fourier Transform into a spectrogram image (Mac Aodha et al., 2018). By clicking on the spectrogram on events of interest, annotations are created (Figure 2) and stored in a relational database with SQL capabilities (Figure S1). For better results, it is recommended to place annotations approximately in the middle of the sound events.

2. Training: Prepare training data from annotated recordings and fit/train CNN models

The training data are composed of spectrogram images computed for a time frame centred at the annotations in the training recordings, as CNNs classify images and not raw sound waves. A denoising method to filter out background noise by removing the mean amplitude in each frequency band is applied to each spectrogram (Aide et al., 2013; Mac Aodha et al., 2018). Manual segmentation and feature extraction are unnecessary as the features that discriminate between classes are learned automatically by the CNN from the image as a whole. The computation of the spectrograms of sound events providing images to train the network is done in conformation with the requirements of Keras (Chollet, 2015), the open-source library used in this package to create and train CNNs. This task can be done through the 'Training GUI' which requires the parameters for computing the spectrograms and the path to the folder where the recordings are stored (Figure S2).

When choosing the spectrogram parameters for a type of sound event, it should be noted that there is a trade-off between temporal and frequency precision, as they are inversely related. Shorter windows will give a more accurate temporal precision at the cost of frequency precision. The length should be small enough to capture the change in time, without sacrificing too much frequency precision (Figure 3). Additionally, the parameter 'spec_size' should be large enough to encompass the maximum duration of the events and the parameter 'freq_range' should cover their typical frequencies range. The other parameters are more generalist and the same values can be used for different sound events, as they only change the resolution of the images created.

Once the spectrograms of individual sound events have been computed, the CNN model can be fitted. A model architecture with

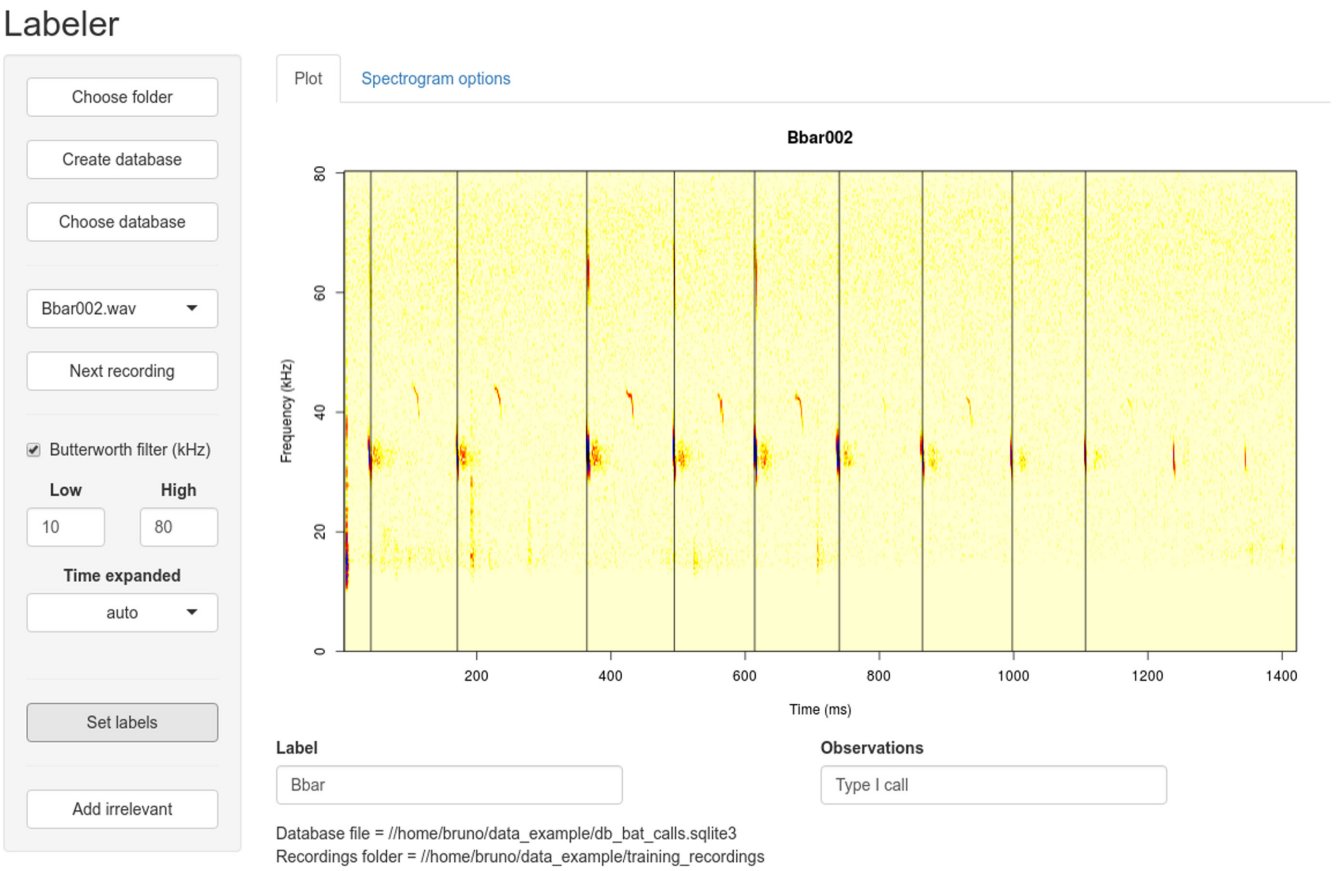

FIGURE 2 Annotations graphical user interface (GUI) with an example of an annotated recording of bat echolocation calls (*Barbastella barbastellus*). To create a new annotation, the user must click on the sound events, indicate the correct class label in the text box 'Label', and press the 'Set labels' button to enter the annotations in the database. The black lines will then be drawn to show the position of the annotations entered in the database. The labels must be unique for each class, as they will be used in the next step to create the training data. The time expanded factor must be also set: for real-time recordings, the value must be 1 and for bat recordings—which sometimes are 10× time-expanded—'auto' can be used. The Butterworth filter is optional as it is only used for visualization purposes.

360,000 parameters (Simonyan & Zisserman, 2015), which can be trained on a CPU relatively quickly and is adequate for many sound events classification (Figure S3), is bundled with the package. Other networks can also be used, the user can adjust the pre-defined network or build a new one. The CNN model can be fitted with the 'Training GUI' (Figure S4), using a stochastic gradient descent optimizer to adjust the network parameters (LeCun et al., 2012). The inherent randomness of the optimizer implies that each fitted CNN is unique.

3. Classification: Automate the use of the fitted model for classifying new recordings

To automate the classification process, each new recording is split into multiple chunks without overlap. For each chunk, the location of maximum energy is detected, converted to a spectrogram, and classified into one of the known classes of the trained model. Irrelevant events are discarded and the class obtained in the majority of the chunks is considered the class of the recording. With this approach, no human intervention is required, rendering the classification process fully automatic (Figure S5).

The classification process outputs a database, in the sqlite3 format, with the sound events detected and the respective class labels and probabilities. Additionally, a file in CSV format is saved to disk, containing summary statistics per recording: the average probability of the class with most events detected and their average frequency of maximum energy. Additionally, the annotated spectrograms of the classified recordings may also be plotted and saved to disk.

3 | WORKING EXAMPLE

soundClass depends on several R packages that are automatically installed. Nevertheless, before first use, Keras must be manually installed. After loading soundClass for the first time run:

```
keras::install_keras()
```

To demonstrate the package functionality and versatility we trained three CNN models to classify: (a) European bat echolocation calls, (b) owl songs (*Strix aluco* and *Athene noctua*), and (c) sperm whale echolocation clicks (*Physeter macrocephalus*). A detailed description of the first model is presented as an example and the code related to the other models is found in Supporting Information (Figures S6 and S7). Before running this example, please download and extract the necessary data: https://doi.org/10.6084/m9.figshare.19550605.v2.

The downloaded data includes an annotations database and the recordings for training and validation. The database was created with the 'Annotations GUI' and contains the annotations of bat echolocation calls and irrelevant sound events in the supplied training recordings. We start this example by setting the extracted folder as the working directory and defining the path to both the training recordings and database:

```
setwd("/path_to_extracted_folder/")
train_recs_folder <- "./training_recordings/"
label_database <- "./db_bat_calls.sqlite3"
```

To create the training data, spectrogram images with pre-defined size and centred at the annotations of the recordings are calculated with the function spectro_calls(). This function outputs a list with four components: (a) an array with the spectrogram matrices; (b) the class labels for each matrix in one-hot-encoded format (i.e. each class label is converted into a new column and each observation is assigned a binary value: 1 in its class label column, 0 otherwise); (c) the parameters used to create the matrices; and (d) the class labels with their respective numeric index. In this example, we are going to train a CNN model to detect European bat echolocation calls (excluding *Rhinolophus* genus). We use a 'freq_range' between 10 and 80 kHz to cover the emitting frequency range, 'spec_size' of 20 ms to encompass the total length of calls from all species and a 'window_length' of 0.5 ms as bat calls have a high rate of change. We decided to exclude the *Rhinolophus* genus as their calls are quite idiosyncratic: they emit longer calls (>40 ms) at higher frequencies (>80 kHz) and with a lower rate of change (Russo & Jones, 2002):

```
train_calls <- spectro_calls(
        files_path = train_recs_folder,
        db_path = label_database,
        spec_size = 20,
        window_length = 0.5,
        overlap = 0.5,
        dynamic_range = 100,
        freq_range = c(10, 80),
        tx = "auto")
```

Once the data has been prepared, a blank model must be loaded into R and compiled. Two variables must be created in the global environment before loading the model: the input shape (input_shape) with the format 'c(number of rows, number of columns, number of channels)' and the number of classes (num_classes). These values can be found in the third component of the list train_calls (note that the number of channels is always 1, as we are working with pseudo-coloured images).

```
input_shape <- c(
        train_calls$parameters$img_rows,
        train_calls$parameters$img_cols,
        1)
num_classes <- train_calls$parameters$num_classes
```

With the training data prepared and the global model variables defined, a model can now be loaded with the base function source(). In this example we use the bundled model:

```
model_path <- system.file(
        "model_architectures", "model_vgg_sequential.R",
        package = "soundClass")
source(model_path, local = TRUE)
```

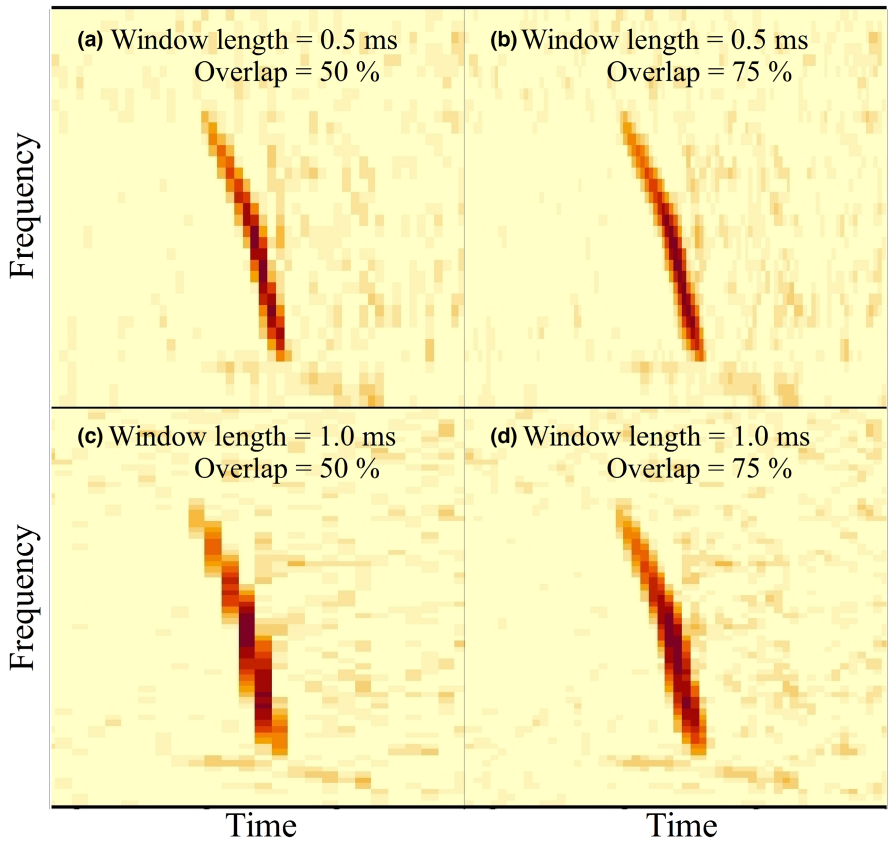

FIGURE 3 Spectrogram parameters effect on the training images. (a) A smaller window leads to less frequency precision but captures quick changes in time; (c) a larger window leads to better frequency precision at the cost of temporal precision; (b) and (d) greater overlap leads to better resolution, but there may be an impractical increase in computation and memory resources required.

At this point, the parameters for compiling and fitting the model must be set. Please note that as we are using classification models, both 'loss' and 'metrics' should always be set as they are in this example. The other parameters can be tuned as desired. For further information about the parameters in the optimizer, please refer to Keras documentation (https://keras.io/api/).

```
model %>% keras::compile(
        optimizer = keras::optimizer_sgd(
            learning_rate = 0.01,
            momentum = 0.9,
            nesterov = TRUE),
        loss = "categorical_crossentropy",
        metrics = "accuracy")
```

The model is now ready to be fitted. Three callbacks (objects that can perform actions at various stages of training) are specified: (a) early stopping, (b) model checkpoint and (c) CSV logger. These objects permit, respectively, to (a) stop the fitting process if there is no improvement in the validation dataset accuracy for a defined number of epochs, (b) save the partially fitted model to disk after each iteration and keep only the best model after training for further use and (c) save to disk the log of the training process. For further information about the callbacks available and their usage, please refer to Keras documentation (https://keras.io/api/). To fit the model the function fit() from package keras is used:

```
model %>% keras::fit(
        x = train_calls$data_x,
        y = train_calls$data_y,
        batch_size = 128,
        epochs = 20,
        callbacks = list(
            keras::callback_early_stopping(patience = 4, monitor =
            "val_accuracy"),
            keras::callback_model_checkpoint("./fitted_model.hdf5",
                monitor = "val_accuracy", save_best_only = TRUE),
            keras::callback_csv_logger("./fitted_model_log.csv")),
        shuffle = TRUE,
        validation_split = 0.3,
        verbose = 1)
metadata <- train_metadata(train_calls)
save(metadata, file = "./fitted_model_metadata.RDATA")
```

The fitted model and an additional file with fitting and training data parameters are saved to disk in the working folder for further use in the classification of novel recordings. To validate the performance of the model, we use recordings not used to calibrate the model. To evaluate model transferability, we use recordings with the same species used for training but also new ones, obtained with a different recorder. The validation dataset is composed of 76 recordings, each 0.5 s long: *Barbastella barbastellus* was present in 13, *Myotis escalerai* in 9, other bat species in 24, and 30 without bats. The

TABLE 1 Confusion matrix with the performance of the trained convolutional neural network (CNN)

	Bat	No bat	Observed	Recall	Precision	F1-score
Bat	45	1	46	98%	79%	87%
No bat	12	18	30	60%	95%	73%
Predicted	57	19	76			

classification is applied to a folder containing these recordings with function auto_id() and the results are saved to the 'out_dir' folder:

```
auto_id(
        model_path = "./fitted_model.hdf5",
        metadata = "./fitted_model_metadata.RDATA",
        file_path = "./validation_recordings/",
        out_file = "id_results",
        out_dir = "./output/",
        save_png = TRUE,
        win_size = 40,
        remove_noise = TRUE,
        tx = "auto")
```

The model class predictions were evaluated using (a) Recall—the proportion of correctly classified samples of a specific class against the total number of samples of that class; (b) Precision—the proportion of correctly classified samples of a specific class against the total number of predicted samples of that class and (c) F1-score—the harmonic mean of recall and precision (or accuracy) for a particular class (Tharwat, 2018). The results show that the fitted model was able to classify bat calls with reasonable accuracy (Table 1). For the class 'bats', this metrics can be interpreted as: 98% of all recordings containing bats were predicted to have bats (Recall); of all bat predictions made, 79% are correct (Precision); the accuracy of a prediction for this class is 87% (F1-score).

4 | DISCUSSION

PAM is rapidly expanding with the recent boom in low-cost, open-source passive acoustic sensors resulting in massive acoustic audio datasets that still present serious analytical difficulties (Aide et al., 2013; Gibb et al., 2019). Despite the increasing number of algorithms to automate acoustic species identification (e.g. Mac Aodha et al., 2018; Rasmussen & Širović, 2021; Stowell et al., 2019), many users do not have the programming or statistical skills to adapt those algorithms to their particular needs (Aide et al., 2013).

Here we present the R package soundClass, which provides a simple and reliable approach to analyse sound recordings resulting from passive acoustic biodiversity monitoring schemes and, importantly, it does not require that the user has extensive coding or statistical skills. Given the ongoing concerns with biodiversity, but also the availability of increasing computational power, these methods are bound to become increasingly used by researchers.

We demonstrate that the soundClass package is a versatile tool for biodiversity monitoring, being able to cover several taxonomic groups, as it is adequate for classification of a variety of sound events, in broad frequency ranges and multiple environments. Indeed, we underline the multi-functionality of soundClass using bat echolocation calls, bird songs, and whale echolocation clicks as study taxa in our examples. It should be noted, however, that the performance of the fitted CNNs in the examples has much space for improvement. To a large extent, this is because we used limited training datasets for demonstration purposes and the size of the training dataset is crucial for CNNs performance, to account for within-class variability and diversity of background noise (Gibb et al., 2019). Additionally, the parameters used for creating the spectrograms can be tweaked to find optimal values for the target species and using different CNN architectures could also lead to better performance. Furthermore, as each fitted CNN is unique due to the inherent randomness of the training algorithm, fitting multiple CNNs to obtain one with better performance is also a possibility.

Nevertheless, by encapsulating the complex steps of data preparation and model fitting in a small number of functions and by creating easy to use GUIs, the package presents a straightforward approach to the process of obtaining trained CNNs for sound events classification and applying them in real-life situations, even for those with little coding or statistical skills. In this sense, soundClass brings to a broader number of users the ability to use machine learning methods, particularly CNNs, for their particular research objectives.

AUTHOR CONTRIBUTIONS

Bruno Silva, Sílvia Barreiro, Pedro J. Alves and José M. Herrera conceived the ideas; Bruno Silva designed the package; Bruno Silva and Frederico Mestre contributed to the development of the functions and the documentation of the package. Bruno Silva, José M. Herrera and Frederico Mestre led the writing of the manuscript. All authors contributed critically to the drafts and gave final approval for publication.

ACKNOWLEDGEMENTS

This work was supported by the projects OLEAdapt (PTDC/BIA-CBI/1365/2020) funded by the Portuguese National Public Agency for Science, Technology and Innovation, and SHOWCASE (ref. 862480) funded by the Horizon 2020 Research and Innovation programme from the European Union. JMH and BS were supported by FCT contracts/grants IF/00001/2015 and SFRH/BD/137803/2018, respectively.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

DATA AVAILABILITY STATEMENT

soundClass is hosted on CRAN (https://cran.r-project.org/web/packages/soundClass/index.html) and the development version is on GitHub (https://github.com/bmsasilva/soundClass). The example data can be downloaded at: https://doi.org/10.6084/m9.figshare.19550605.v2.

ORCID

Bruno Silva https://orcid.org/0000-0002-7323-513X
Frederico Mestre https://orcid.org/0000-0002-7390-1120
Sílvia Barreiro https://orcid.org/0000-0002-8263-6373
Pedro J. Alves https://orcid.org/0000-0002-7884-0857
José M. Herrera https://orcid.org/0000-0001-7968-3438

REFERENCES

Aide, T. M., Corrada-Bravo, C., Campos-Cerqueira, M., Milan, C., Vega, G., & Alvarez, R. (2013). Real-time bioacoustics monitoring and automated species identification. *PeerJ*, *1*, e103. https://doi.org/10.7717/peerj.103

Browning, E., Gibb, R., Glover-Kapfer, P., & Jones, K. E. (2017). Passive acoustic monitoring in ecology and conservation. *WWF Conservation Technology Series*, *1*(2), 75. https://doi.org/10.13140/RG.2.2.18158.46409

Chollet, F. (2015). *Keras*. GitHub Retrieved from https://keras.io

Gibb, R., Browning, E., Glover-Kapfer, P., & Jones, K. E. (2019). Emerging opportunities and challenges for passive acoustics in ecological assessment and monitoring. *Methods in Ecology and Evolution*, *10*(2), 169–185. https://doi.org/10.1111/2041-210X.13101

LeCun, Y., Bottou, L., Orr, G. B., & Müller, K. R. (2012). Efficient backprop. In *lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics): Vol. 7700 LECTU* (issue 1998). https://doi.org/10.1007/978-3-642-35289-8_3

Mac Aodha, O., Gibb, R., Barlow, K. E., Browning, E., Firman, M., Freeman, R., Harder, B., Kinsey, L., Mead, G. R., Newson, S. E., Pandourski, I., Parsons, S., Russ, J., Szodoray-Paradi, A., Szodoray-Paradi, F., Tilova, E., Girolami, M., Brostow, G., & Jones, K. E. (2018). Bat detective—Deep learning tools for bat acoustic signal detection. *PLoS Computational Biology*, *14*(3), 1–19. https://doi.org/10.1371/journal.pcbi.1005995

Marques, T. A., Thomas, L., Martin, S. W., Mellinger, D. K., Ward, J. A., Moretti, D. J., Harris, D., & Tyack, P. L. (2013). Estimating animal population density using passive acoustics. *Biological Reviews*, *88*(2), 287–309. https://doi.org/10.1111/brv.12001

Merchant, N. D., Fristrup, K. M., Johnson, M. P., Tyack, P. L., Witt, M. J., Blondel, P., & Parks, S. E. (2015). Measuring acoustic habitats. *Methods in Ecology and Evolution*, *6*(3), 257–265. https://doi.org/10.1111/2041-210X.12330

Rasmussen, J. H., & Širović, A. (2021). Automatic detection and classification of baleen whale social calls using convolutional neural networks. *The Journal of the Acoustical Society of America*, *149*(5), 3635–3644. https://doi.org/10.1121/10.0005047

Russo, D., & Jones, G. (2002). Identification of twenty-two bat species (Mammalia: Chiroptera) from Italy by analysis of time-expanded recordings of echolocation calls. *Journal of Zoology*, *258*(1), 91–103. https://doi.org/10.1017/S0952836902001231

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015—Conference Track Proceedings*, 1–14.

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., Pamuła, H., Stylianou, Y., Glotin, H., & Stowell, D. (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

Tharwat, A. (2018). Classification assessment methods. *Applied Computing and Informatics*, *17*(1), 168–192. https://doi.org/10.1016/j.aci.2018.08.003

SUPPORTING INFORMATION

Additional supporting information can be found online in the Supporting Information section at the end of this article.

How to cite this article: Silva, B., Mestre, F., Barreiro, S., Alves, P. J., & Herrera, J. M. (2022). soundClass: An automatic sound classification tool for biodiversity monitoring using machine learning. *Methods in Ecology and Evolution*, *13*, 2356–2362. https://doi.org/10.1111/2041-210X.13964