Journal of
open research software

Bas, Y et al 2017 Tadarida: A Toolbox for Animal Detection on Acoustic Recordings.
Journal of Open Research Software, 5: 6, DOI: https://doi.org/10.5334/jors.154

SOFTWARE METAPAPER

Tadarida: A Toolbox for Animal Detection on Acoustic Recordings

Yves Bas,[1,2] Didier Bas[3] and Jean-François Julien[1]

[1] CESCO, MNHN, FR
[2] CEFE, CNRS, FR
[3] none (volunteer), FR
Corresponding author: Yves Bas
(ybas@mnhn.fr)

Passive Acoustic Monitoring (PAM) recently extended to a very wide range of animals, but no available open software has been sufficiently generic to automatically treat several taxonomic groups. Here we present **Tadarida,** a software toolbox allowing for the detection and labelling of recorded sound events, and to classify any new acoustic data into known classes. It is made up of three modules handling **D**etection, **L**abelling and **C**lassification and running on either Linux or Windows. This development resulted in the first open software (1) allowing generic sound event detection (multi-taxa), (2) providing graphical sound labelling at a single-instance level and (3) covering the whole process from sound detection to classification. This generic and modular design opens numerous reuse opportunities among (bio)acoustics researchers, especially for those managing and/or developing PAM schemes.

The whole toolbox is openly developed in C++ (Detection and Labelling) and R (Classification) and stored at https://github.com/YvesBas.

Keywords: Acoustic Recording Unit (ARU); bioacoustics; animal calls classification; ecoacoustic indices; environmental sound; machine learning; random forest; sound event detection; species monitoring
Funding Statement: This software has been partially funded by Holcim patronage of "Vigie-Chiro" monitoring scheme.

(1) Overview

Introduction

Reduced cost of acoustic recorders and increase in storage capacity has resulted in an exponential development of Passive Acoustic Monitoring (PAM) of a very wide range of animals in a few years [1, 2, 3], opening up a new field of research [4] and responding to an urgent need for biodiversity monitoring [5]. However, data analysis of this emerging big data is now impeded by limited software development and availability [6, 7].

To date, no available software allows researchers to perform PAM on a wide range of taxonomic groups, though numerous classification methods of animal vocalizations have been developed in recent years [8]. Here we present a software toolbox that enables the building of an automatic classifier from available sound reference data, and then to classify recorded sound events into known classes. This toolbox was called Tadarida (which is incidentally a widespread bat genus), for Toolbox for Animal Detection on Acoustic Recordings Integrating Discriminant Analysis. Discriminant analysis is not defined here as Fisher's LDA but rather any type of automatic classification.

This toolbox was initially developed to support the French bat monitoring scheme "Vigie-Chiro" launched in 2006 [9]. Like other PAM schemes during this period, Vigie-Chiro experienced an exponential increase in recording data. In addition, a large volume of acoustic data has been recorded for other taxa without being identified, especially for bush-crickets which did not benefit from any monitoring scheme [10, 11].

Until now, all available software for bat automatic identification were commercial and based on a very specific sound event detection process that prevented efficient bush-cricket detection. Tadarida development focused on detecting every sound event, even if they are structurally very different (e.g. bats and bush-crickets) and overlapping in time or frequency.

This generic detector enables complex delimitations of the sound events in time and frequency so that the least possible amount of noise, i.e. spectrogram elements that do not contain signal, are included. It then performs a broad range of feature extraction on all detected sound events (DSEs). This first part of the toolbox was named Tadarida-D (for Detection).

Because of a lack of sufficient available reference sound data on bats and bush-crickets, we needed to

build our own training dataset for automatic identification (machine learning). However, no available software allowed for sound annotation at the scale of our detected sound events. Massive sound annotation of animal vocalizations was limited to a larger time scale (several seconds) so that most sound reference data contained multiple species with sound events not attributed to a single label with certainty. Although recent developments have significantly improved the machine learning process for multi-label multi-instance data [12], we circumvent such limitations by developing a Graphical User Interface called Tadarida-L (for Labelling) that allows a user to generate single-label single-instance training data.

The last part of the toolbox, Tadarida-C (for Classification), uses features extracted by Tadarida-D and labels generated through Tadarida-L to build and use a random forest automatic classification that handles strong unevenness in class sample size.

This modular toolbox has been designed for a community of users sharing two interacting roles (e.g. PAM managers and participants). First, "PAM managers" go through the whole process: automatic detection and labelling to build automatic classification. Then, "PAM participants" can use the detection and classification processes to apply the automatic classifier provided by PAM managers. This development resulted in the first open software (1) allowing generic sound event detection (multi-taxa), (2) providing graphical sound labelling at a single-instance level and (3) covering the whole process from sound detection to classification.

Implementation and architecture

Tadarida toolbox consists of 3 software modules (see **Fig. 1** for the structure):

1) Tadarida-D **detecting** sound events and extracting their features.
2) Tadarida-L facilitating the **labelling** of sound events and generating a sound reference database.
3) Tadarida-C building and using a supervised **classifier** to automatically sort sound events into defined classes (e.g. bat and bush-cricket species).

Tadarida-D and Tadarida-L implementation

TADARIDA-D and TADARIDA-L have been developed in Qt C++, because it combines clean writing, based on the concepts of object-oriented programming, and allows for fast processing.

TADARIDA-D runs faster than real time on recordings even with a basic configuration and under the most demanding conditions: 500 kHz sample rate .wav files, detecting 100 sound events per seconds and extracting 269 features.

These constraints have sometimes led us to focus on solutions that save time at the expense of code readability. For example, we used pointers in a number of cases and tables instead of vectors. This applies to shapesDetects and detectsParameter2 (see below).

Qt 5.5 framework was chosen for its portability across different operating systems. Initially developed for Windows, Tadarida-D has also been successfully deployed on diverse Linux systems (see below).

Tadarida-D uses two external libraries:

– libsndfile to read .wav files.
– FFTW3f-3 to compute time-frequency matrices (spectrograms) from. wav files, through Fast Fourier Transformation.

To exploit the potential of multi-core computers, Tadarida-D uses parallel computing, launching one to eight threads according to user settings. These threads simultaneously process a fraction of the .wav files.

Tadarida-D architecture

TadaridaD is a non-graphical application that is executed from the command line in scripts, particularly on servers.

This application handles the sound events detection and features extraction, and is structured around 4 classes:

1) DetecLaunch (main class)

DetecLaunch operates the input parameters: directories of .wav files to process, number of threads to be executed, etc.

The process first distributes .wav files among the threads. Next, variables that will be used by the threads are initialized (to manage the variables that do not support multithreading) and the threads created (objects of Detec class) with the necessary settings, and then executed.

2) Detec (thread treatment)

Detec class retrieves provided settings and creates the object of DetecTreatment class that will handle the output processing (generating features tables).

The run() method (executed by pdetec [i] → start() ;) organizes treatment.

It calls for each .wav file to be processed, the treatOneFile method, which launches CallTreatmentsForOneFile(...) method of DetecTreatment class (see below).

Note that the division of tasks between the Detec and Detectreatment classes avoids a duplication of code in the DetectTreatment class between Tadarida-D and Tadarida-L projects (see below).

3) Classe DetecTreatment

After initialization, the CallTreatmentsForOneFile method handles the processing of each .wav file by successively calling the following methods:

– openWavFile and ComputeFFT which produces a spectrogram (time-frequency matrix) for each.wav file.
– CorrectNoise skipping silence and equalizing frequency bands (Note that this operation has strong consequences for sound event detection and several advantages: reducing bias generated by heterogeneity in recorders output, suppressing narrowband electronical noise and separating sound events even

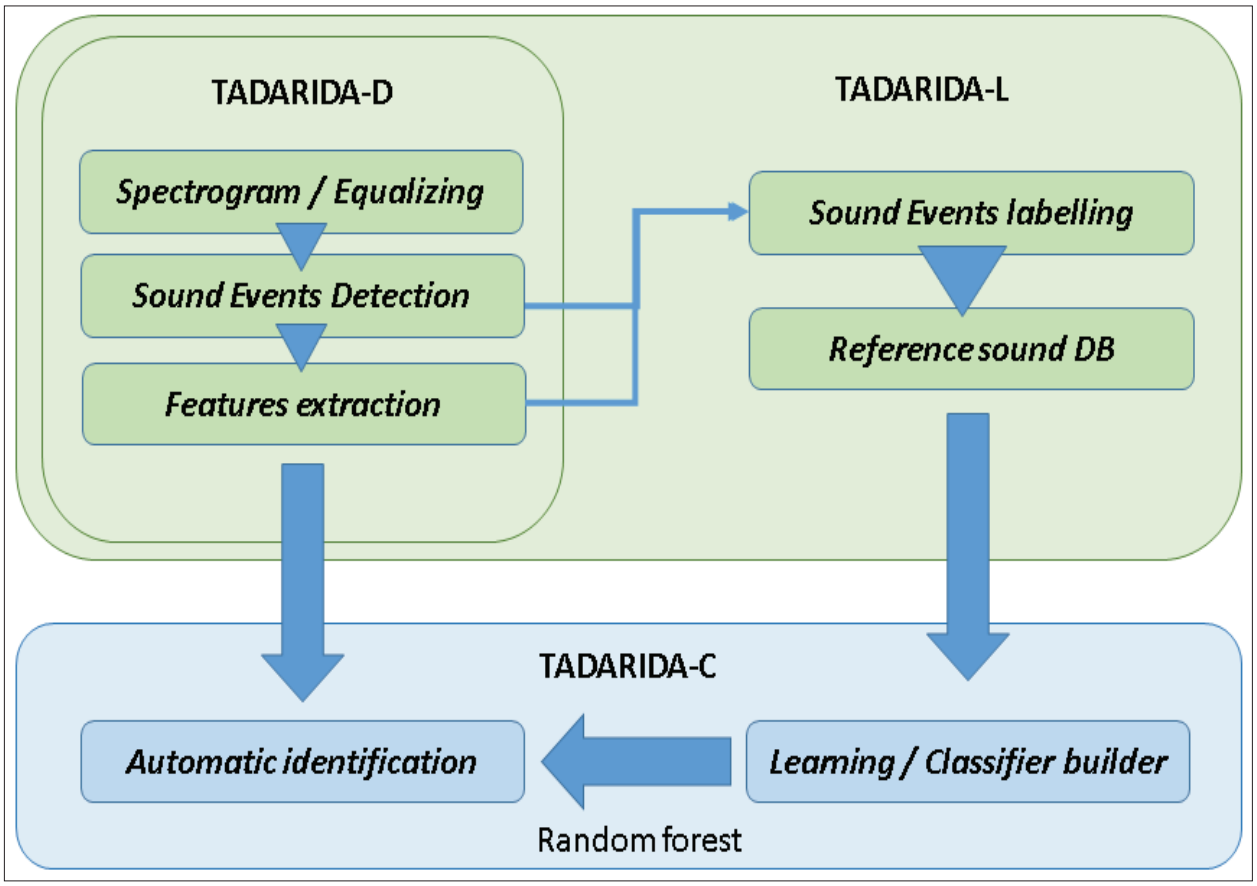

Figure 1: Overview of Tadarida functionalities. Left half corresponds to end use, whereas expert users go through the whole process.

when a chorus of individuals produce constantly a large amount of amplitude on a frequency band).
- shapesDetects detecting sound events, defined as a set of elements of the frequency-time matrix isolating an acoustic signal in both frequency and time coming from a single acoustic source (this detection algorithm follows an hysteresis function, see https://github.com/YvesBas/Tadarida-D/blob/master/Manual_Tadarida-D.odt for details).
- detectsParameter2 and saveParameters computing and saving the 269 numerical features on each DSE in tables within .ta files (tab separated text format, **Fig. 2**). See https://github.com/YvesBas/Tadarida-D/blob/master/Manual_Tadarida-D.odt for the description of each numerical features.

Note that shapesDetects defines the master point of each DSE as its element with the highest amplitude value, then DSE are filtered according to the frequency of this master point: 8–250 kHz in HF mode and 0.8–25 kHz in LF mode. Note that the HF range covers every known vocalization of bats and bush-crickets and the LF range covers most other animal vocalization.

Tadarida-L architecture

Tadarida-L is an extended version of Tadarida-D to handle sound events labelling and reference sound database (RSDB) constitution (in addition to sound events detection and features extraction as done by Tadarida-D).

This application is structured around 4 classes (corresponding to each .cpp file):

1) TadaridaMainWindow

This class generates the main window of the software and displays main user settings.

The left half of the window contains all the input settings of the detection process: the directory path of the .wav files to be processed, and optional settings (**Fig. 3**). It calls threads of the Detec class for .wav files treatment.

The right half of the window displays the labelling interface. It calls the following Fenim class.

2) Detec and Detectreatment

Both Detec and Detectreatement classes have the same purpose in Tadarida-L as they have in Tadarida-D (see above). Detectreatment class is strictly identical. Detec class here has one additional function: it handles the generation of supplementary output files (in extended mode): spectrogram images (.jpg) and DSEs data (.da2). These files are used by Fenim class as input information (see **Fig. 2**).

3) Fenim

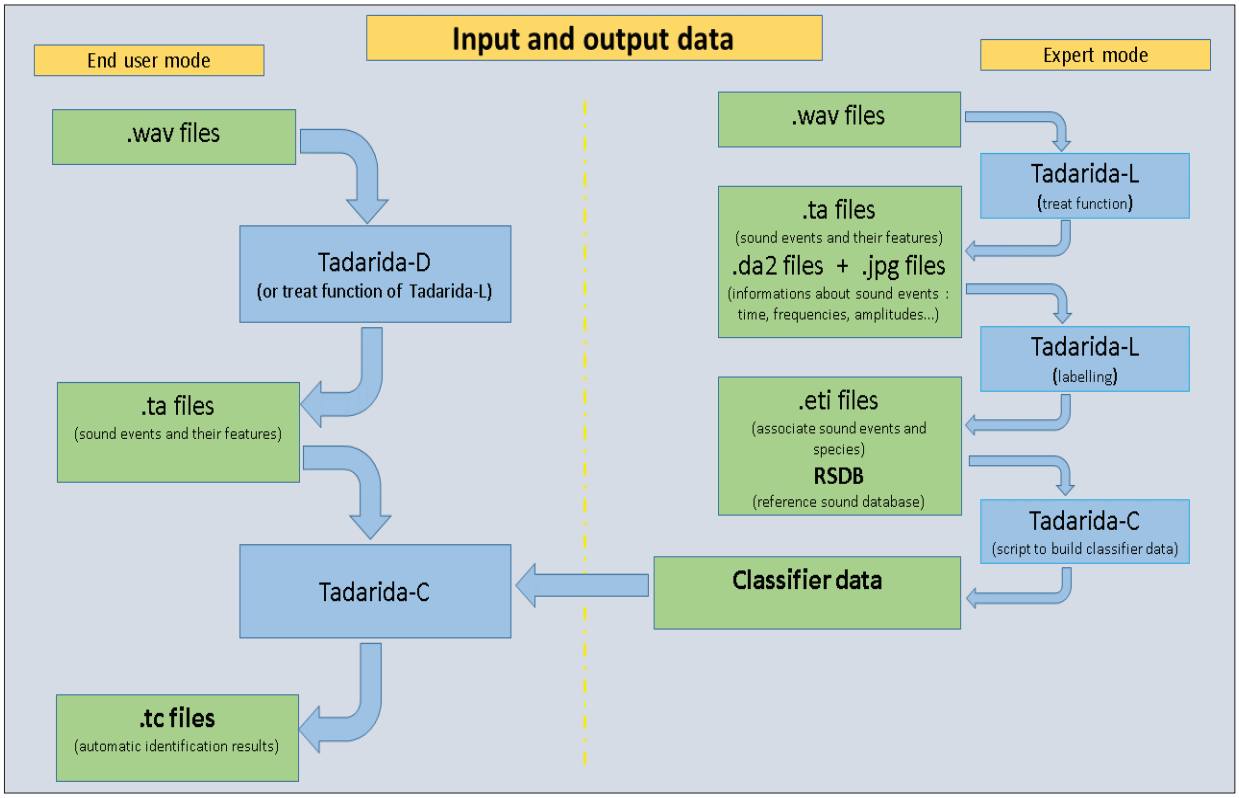

Figure 2: Overview of the data process in both end and expert use modes, detailing input and output file formats of the three modules.

Fenim class handles the labelling man-machine interface (**Fig. 4**), involving:

- spectrogram display that can be zoomed in and out on two separate Windows.
- graphical multiple selection of DSEs.
- label data entry (11 fields: species, confidence, location, etc. see https://github.com/YvesBas/Tadarida-L/blob/master/Manual_Tadarida-L.odt for details).
- saving data in standard tables (.eti) linked to extracted features.

This labelling process allows the user to quickly generate a standardized RSDB that can be shared with other users, and/or used to build their own classifier (**Fig. 2**).

Note that Tadarida-L has been built in an evolutive way so that both detection and feature extraction process can be drastically changed with no loss of information in users RSDBs.

Tadarida will indeed propose an entire base reprocessing to upgrade and homogenize the database, if the defined RSDB corresponds to an anterior version of the software. New DSEs will be matched according to their previous and new structural data (.da2), see https://github.com/YvesBas/Tadarida-L/blob/master/Manual_Tadarida-L.odt for details.

Tadarida-C implementation and architecture

Tadarida-C handles the classification of DSE, based on features extracted by Tadarida-D and RSDB collected by Tadarida-L, and thus provides the final output of the Tadarida toolbox.

It contains two R-scripts: one for building a classifier (for expert users) and another using it (for end users, see **Fig. 2**). Tadarida-C has been developed in R because it allows the use and optimisation of the Random Forests (RF) algorithm [13], with numerous built-in statistical tools (summaries, graphs, etc). Both scripts use randomForest and data.table packages [14, 15]. This latter package is only used through its "rbindlist" function that allows for fast aggregation of large data frames.

1) Building a classifier

The "buildClassif.r" script should be run each time user's RSDB has been significantly improved or when there is any need for a new classifier (different species list, settings, etc). It first aggregates and merges label data (.eti files, see Tadarida-L) and features data (.ta files) from a specified RSDB (**Fig. 2**).

This data frame may then be filtered on a species list (if provided) and is finally used to build a series of 50 RFs [13] of 10 classification trees each. The authors found out that combining trees with different subsampling levels better handled the trade-off between classification error rates on common species (i.e. species with a large number of labelled DSEs in the RSDB) and error rates on rare species. Thus, a gradient of subsampling is set so that the first trees of the series use most of the available DSEs in the RSDB, whereas last trees use an equal number of DSEs for most species. The strength of the subsampling can be tuned through two input settings (see https://github.com/

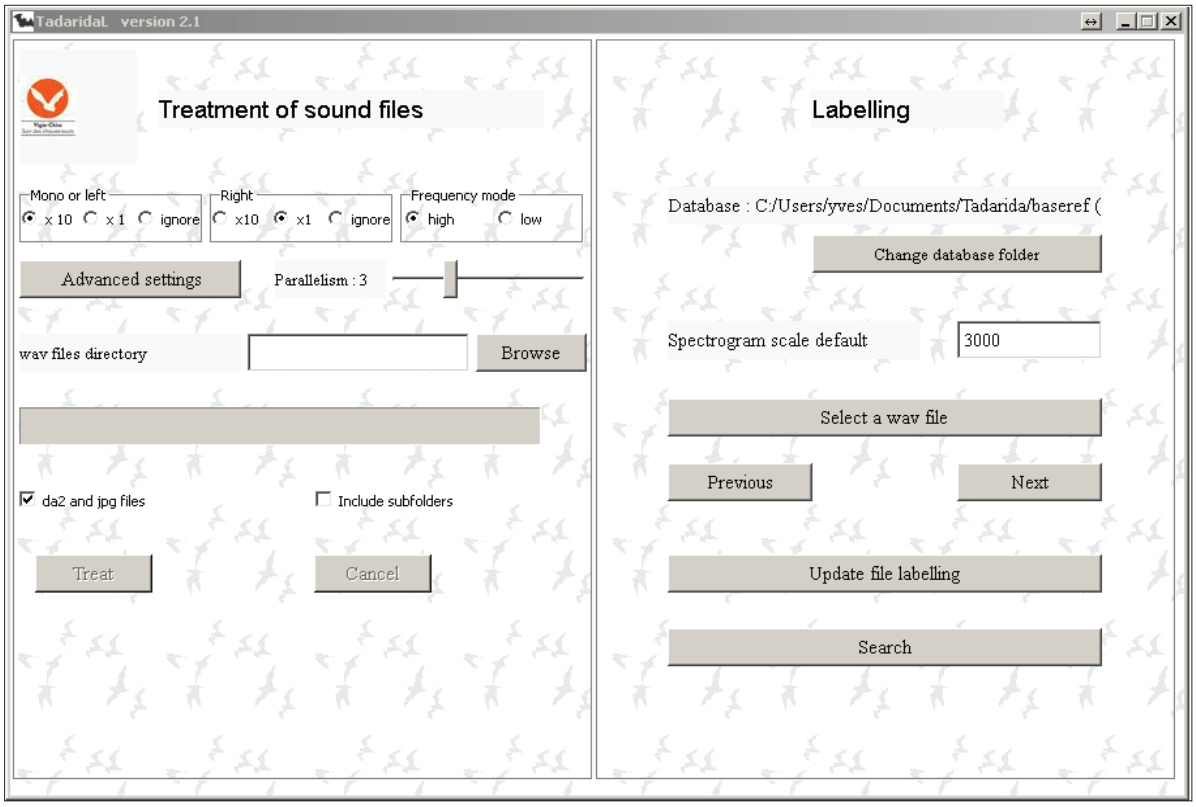

Figure 3: The graphical user interface of Tadarida-L.

YvesBas/Tadarida-C/blob/master/Manual_Tadarida-C.odt for details on all RF parameters).

The "site" field of the label data is also used for subsampling, each RF using 63% of available sites. This makes the training and testing independent for each tree. To implement this subsampling, authors have modified the randomForest function so that size of sample class (sampsize) could be equal to 0.

The output file is named "ClassifEspHF3.learner" to be used as input in the following script (**Fig. 2**).

2) Using the classifier

The "TadaridaC.r" script aims to return to the user the list of species present within each treated .wav file, and a confidence index of each automatic identification. For that purpose, it applies the previously built RF classifier (ClassifEspHF3.learner) to any list of .ta files (output from Tadarida-D or Tadarida-L, see **Fig. 2**).

With the "predict" function from randomForest R package [14], it first converts features values extracted on each DSE to a matrix of class probabilities named "ProbEsp0" (i.e. the probabilities for each DSE to belong to each potential species).

Most users will not be interested to an identification on such a small scale since many DSEs within a .wav file could come from the same source. Thus, the next commands aim at (1) summarizing this information within a .wav file and (2) defining the number of species that are present within the recording.

This latter objective is accomplished through a simple and robust algorithm functioning in a loop:

1) The maximum score within the class probability matrix gives the identity of the first species automatically identified (species A).
2) DSE returning a probability lower than 0.02 for species A are considered as not being from this species and this subset goes through step 1 again. If this condition is not met for any DSE, the loop is ended.

Some ancillary data is also computed for end users to get summary information on the species likely to be present in their recordings, where in time and frequency the identified vocalizations occur and how confident the identification is (see https://github.com/YvesBas/Tadarida-C/blob/master/Manual_Tadarida-C.odt for details).

Quality control

The detection, feature extraction and classification processes have been extensively tested by its automated use on French Bat Monitoring data since October 2015. 296 participants uploaded 12 952 016 wave files on a web portal (www.vigiechiro.herokuapp.com; see [16, 17] for source code), and all files were successfully processed by Tadarida-D and Tadarida-C to get species identification on all sound events recorded. The few bugs that remained after the development phase were then quickly discovered and fixed.

Moreover, Tadarida toolbox has been used to build a specific classifier for the Norfolk Bat Survey, then to process the whole data, i.e. 1.1 million .wav files. Newson et al. [11]

manually checked 328 291 of these files to evaluate classifier performance on bush-crickets, showing variable but sufficient performance to perform large-scale monitoring. Appendix S1 of this latter paper further shows the relative importance of features extracted by Tadarida in species classification. Therefore, both Vigie-Chiro and Norfolk Bat Survey provide valuable proofs of concept that Tadarida fulfils its purpose.

Any upgrade in detection and/or feature extraction processes resulted in the reprocessing of authors' RSDB (today counting 17,773 .wav files and 863,730 labelled DSEs) that constituted each time a robust test able to detect any new bugs before a release to the users. This reprocessing happened 8 times since the start of the authors' RSDB constitution (01/08/2014).

All three modules (Tadarida-D, -L and -C) have thus been intensively tested through this production phase. Moreover, use cases of Tadarida HMI are relatively few and then easily tested before each release to the users. For that purpose, each code repository contains a test plan document describing manual test scenarios, and their associated input and output files.

(2) Availability

Operating system

Tadarida-D and -C have been successfully tested on various versions of Windows (both 32- and 64-bit, and from XP to 8.1) and Linux (Ubuntu 14.04LTS and 14.6, Debian 8.0 and Scientific Linux 6.8).

Code sources are the same for both OS, except for one line which is disabled in the Windows version ("typedef int64_t__int64;" in detec.h).

Tadarida-L have been successfully tested on various versions of Windows (from Windows XP to Windows 8.1). It has not been built on Linux yet.

Programming language

C++ Qt (5.4 and 5.5) for Tadarida-D and Tadarida-L. Both have been successfully built from Qt 4.8.5 to Qt 5.7. R 3.3.0 for Tadarida-C.

Additional system requirements

Memory consumption is approximately 150 MB for threads of Tadarida-D, but is much higher for Tadarida-C. Memory consumption of the "builder" script depends on the RSDB size and the number of defined species. With more than 800 000 DSEs and 91 defined species in current authors' RSDB, the builder script consumes approximately 15 GB of memory. "TadaridaC" script consumes a lower amount of memory unless a very large number of DSEs is provided in the input (more than 1 million).

All modules have a dedicated manual that details the install procedure and the requirements.

Dependencies

Tadarida-D and Tadarida-L use the "sndfile" and "fftw3" libraries while Tadarida-C use "randomForest" and "data.table" libraries.

List of contributors

- Yves Bas (contributed to Tadarida-D, -L and -C).
- Didier Bas (main developer of Tadarida-D and Tadarida-L).
- Jean-François Julien (contributed to Tadarida-C).
- Emmanuel Leblond (helped to first implementation of Tadarida-D and -C on Linux).

Software location

Archive

Name: Figshare

Persistent identifier: https://doi.org/10.6084/m9.figshare.4212837.v1

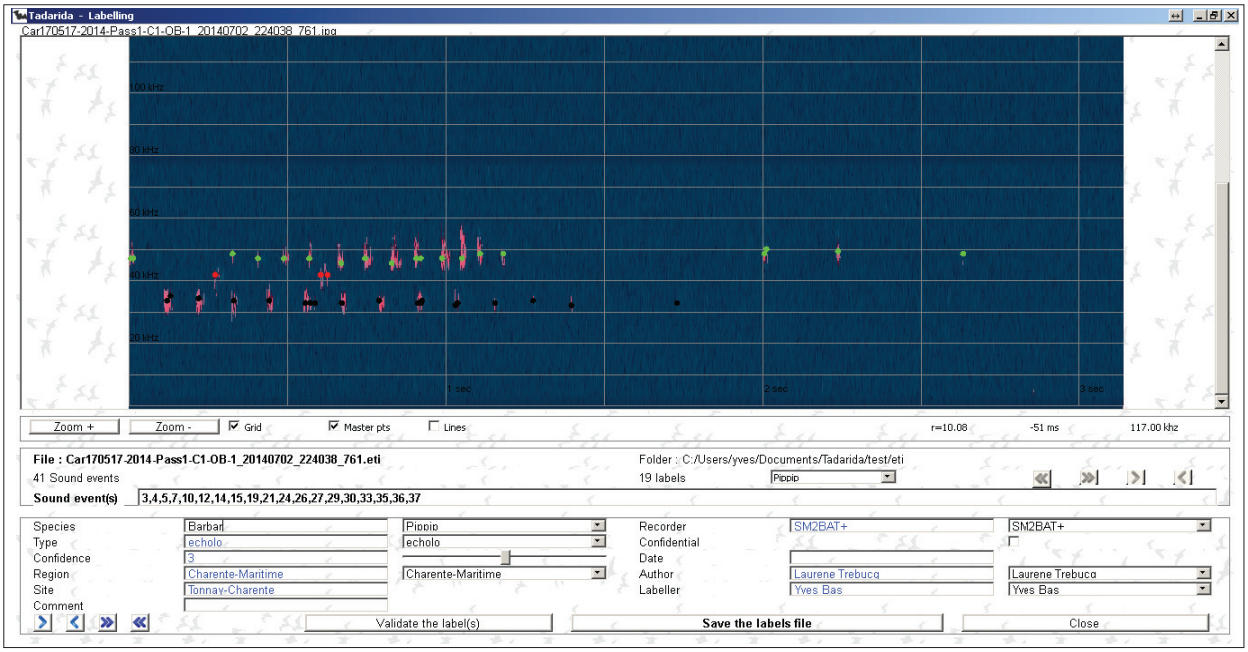

Figure 4: The graphical user interface of the labelling mode of Tadarida-L.

Licence: CC-BY
Publisher: Yves Bas
Date published: 07/11/16

Code repository

Name: Github
Identifier: github.com/YvesBas
Licence: LGPL-3.0 – GNU for Tadarida-D and Tadarida-L, GPL-3.0 – GNU for Tadarida-C
Date published: 07/11/16

Language

English

(3) Reuse potential

Tadarida toolbox has been first designed for the specific need of the monitoring of French common bats and bush-crickets (243 end users so far), and is also used to analyse Norfolk Bat Survey data [11, 18]. Both schemes have a longer term objective to monitor a wide range of other acoustically active taxa (birds, frogs, other insects…). Thus, this very generic design makes it useful to any researchers involved in PAM by saving time at several stages of the process (sound event detection, RSDB constitution, etc) and potentially broadening their range of monitored species.

Tadarida may even be used in other intensive research topics in acoustics such as multi-source acoustic detection in urban environments [19].

Tadarida is made up of three different software packages: Tadarida-D, -L and -C. This modular architecture further broadens the community of researchers likely to be interested in Tadarida toolbox because each package can be used independently. For example, the prolific development of acoustic indices in ecoacoustics can benefit from Tadarida-D by the way it detects and extract numerous features on every animal vocalization in a soundscape, thus allowing measures of alpha and beta diversity [5, 20].

Tadarida is not only modular but also interoperable with other software. For example, since sound events detected by Tadarida-D are precisely located in time and frequency (stored in .ta files), they can easily be linked to sound events detected by other software (e.g. SonoBat™). Thus, a researcher willing to use the feature extraction of such software can nonetheless benefit from the functionalities of Tadarida-L (sound event labelling and RSDB constitution) and/or Tadarida-C (building and using a classifier). To integrate Tadarida-L with other sound detection softwares, all that is needed is to link output tables of both softwares thanks to frequency and time variables, namely Fmin, Fmax, StTime and Dur for Tadarida-L features. To integrate Tadarida-C with external sound detection softwares, all that is needed is (1) to take external software's feature tables as input (object param3) and (2) to adapt FormulCrit object to external software's features list.

Tadarida code is built and documented so that third-party developers can also modify or improve it according to their needs. The detectsparameter2 method is especially designed in such a way that other features can easily be added.

Support is not guaranteed for this code, but users are encouraged to contact the authors to discuss specific application possibilities and issues.

Acknowledgements

The authors would like to thank an anonymous reviewer for useful comments, Christian Kerbiriou and Grégoire Loïs for advices and help in the project management, Stuart Newson, Emmanuel Leblond, Jérôme Landieth and the 251 Vigie-Chiro participants for extensive testing, and finally Stuart Newson for careful rereading of the article.

Competing Interests

The authors have no competing interests to declare.

References

1. **Selby, T H**, **Hart, K M, Fujisaki, I, Smith, B J, Pollock, C J** and **Hillis-Starr, Z** et al 2016 Can you hear me now? Range-testing a submerged passive acoustic receiver array in a Caribbean coral reef habitat. *Ecol Evol*; 6: 4823–35. DOI: https://doi.org/10.1002/ece3.2228
2. **Kalan, A K, Mundry, R, Wagner, O J J, Heinicke, S, Boesch, C** and **Kuehl, H S** 2015 Towards the automated detection and occupancy estimation of primates using passive acoustic monitoring. *Ecol Indic*; 54: 217–26. DOI: https://doi.org/10.1016/j.ecolind.2015.02.023
3. **Froidevaux, J S P, Zellweger, F, Bollmann, K,** and **Obrist, M K** 2014 Optimizing passive acoustic sampling of bats in forests. *Ecol Evol*; 4: 4690–700. DOI: https://doi.org/10.1002/ece3.1296
4. **Sueur, J** and **Farina, A** 2015 Ecoacoustics: the Ecological Investigation and Interpretation of Environmental Sound. *Biosemiotics*; 8: 493–502. DOI: https://doi.org/10.1007/s12304-015-9248-x
5. **Harris, S A, Shears, N T** and **Radford, C A** 2016 Ecoacoustic indices as proxies for biodiversity on temperate reefs. *Methods Ecol Evol*; 7: 713–24. DOI: https://doi.org/10.1111/2041-210X.12527
6. **Roch, M A, Batchelor, H, Baumann-Pickering S, Berchok, C L, Cholewiak, D** and **Fujioka, E** et al 2016 Management of acoustic metadata for bioacoustics. *Ecol Inform*; 31: 122–36. DOI: https://doi.org/10.1016/j.ecoinf.2015.12.002
7. **Merchant, N D, Fristrup, K M, Johnson, M P, Tyack, P L, Witt, M J** and **Blondel, P** et al 2015 Measuring acoustic habitats. *Methods Ecol Evol*; 6: 257–65. DOI: https://doi.org/10.1111/2041-210X.12330
8. **Stowell, D** and **Plumbley, M D** 2014 Automatic large-scale classification of bird sounds is strongly improved by unsupervised feature learning. *Peerj*; 2: E488. DOI: https://doi.org/10.7717/peerj.488
9. **Azam, C, Le Viol, I, Julien, J-F, Bas, Y** and **Kerbiriou, C** 2016 Disentangling the relative effect of light pollution, impervious surfaces and intensive agriculture on bat activity with a national-scale

monitoring program. *Landsc Ecol*: 1–13. DOI: https://doi.org/10.1007/s10980-016-0417-3

10. **Jeliazkov, A, Bas, Y, Kerbiriou, C, Julien, J-F, Penone, C** and **Le Viol, I** 2016 Large-scale semi-automated acoustic monitoring allows to detect temporal decline of bush-crickets. *Glob Ecol Conserv*; 6: 208–218. DOI: https://doi.org/10.1016/j.gecco.2016.02.008
11. **Newson, S, Bas, Y, Murray, A** and **Gillings, S** 2017 Potential for coupling the monitoring of bush-crickets with established large-scale acoustic monitoring of bats. *Meth Ecol Evol*; 27 JAN 2017. DOI: https://doi.org/10.1111/2041-210X.12720
12. **Briggs, F, Lakshminarayanan, B, Neal, L, Fern, X Z, Raich, R** and **Hadley, S J K** et al 2012 Acoustic classification of multiple simultaneous bird species: A multi-instance multi-label approach. *J Acoust Soc Am;* 131: 4640–50. DOI: https://doi.org/10.1121/1.4707424
13. **Breiman, L** 2001 Random Forests. *Mach Learn*; 45: 5–32. DOI: https://doi.org/10.1023/A:1010933404324
14. **Liaw, A** and **Wiener, M** 2002 Classification and regression by randomForest. *R News*; 2:18–22.
15. **Dowle, M, Short, T, Lianoglou, S, Saporta, R** and **Srinivasan, A** and **Antonyan, E** 2014 data.table: Extension of data.frame.
16. **Leblond, E** and **Landieth, J** 2014–2017 vigiechiro-api. URL https://github.com/Scille/vigiechiro-api.
17. **Landieth, J** and **Leblond, E** 2014–2017 vigiechiro-front. URL: https://github.com/Scille/vigiechiro-front.
18. **Newson, S E, Evans, H E** and **Gillings, S** 2015 A novel citizen science approach for large-scale standardised monitoring of bat activity and distribution, evaluated in eastern England. *Biol Conserv*; 191: 38–49. DOI: https://doi.org/10.1016/j.biocon.2015.06.009.
19. **Mesaros, A, Heittola, T, Eronen, A** and **Virtanen, T** 2010 Acoustic event detection in real life recordings. Signal Process. Conf. 2010 18th Eur., IEEE; p. 1267–1271.
20. **Depraetere, M, Pavoine, S, Jiguet, F, Gasc, A, Duvail, S** and **Sueur, J** 2012 Monitoring animal diversity using acoustic indices: implementation in a temperate woodland. *Ecol Indic*; 13: 46–54. DOI: https://doi.org/10.1016/j.ecolind.2011.05.006

How to cite this article: Bas, Y, Bas, D and Julien, F J 2017 Tadarida: A Toolbox for Animal Detection on Acoustic Recordings. *Journal of Open Research Software,* 5: 6, DOI: https://doi.org/10.5334/jors.154

Submitted: 08 November 2016 **Accepted:** 10 February 2017 **Published:** 21 February 2017

]u[*Journal of Open Research Software* is a peer-reviewed open access journal published by Ubiquity Press

OPEN ACCESS