ohun: an R package for diagnosing and optimizing automatic sound event detection

Journal:	*Methods in Ecology and Evolution*
Manuscript ID	Draft
Wiley - Manuscript Type:	Application
Date Submitted by the Author:	n/a
Complete List of Authors:	Araya-Salas, Marcelo; Universidad de Costa Rica, Centro de Investigación en Neurociencias; Universidad de Costa Rica, Biologia; Universidad de Costa Rica, Sede del Sur Smith-Vidaurre, Grace; The Rockefeller University, Laboratory of Neurogenetics of Language; The Rockefeller University, Rockefeller University Field Center; University of Cincinnati, Department of Biological Sciences Chaverri, Gloriana; Universidad de Costa Rica, Sede del Sur; Smithsonian Tropical Research Institute Brenes, Jan; Universidad de Costa Rica, Centro de Investigación en Neurociencias Chirino, Fabiola; Universidad de Costa Rica, Biologia Elizondo-Calvo, Jorge; Universidad de Costa Rica, Biologia Rico-Guevara, Alejandrio; University of Washington, Department of Biology; University of Washington, Burke Museum of Natural History and Culture
Keywords:	bioacoustics, automatic detection, optimization, animal vocalizations
Abstract:	1. Animal acoustic signals are widely used in diverse research areas due to the relative ease with which sounds can be registered across a wide range of taxonomic groups and research settings. However, bioacoustics research can quickly generate large data sets, which might prove challenging to analyze promptly. Although many tools are available for the automated detection of sounds, choosing the right approach can be difficult only a few tools provide a framework for evaluating detection performance. 2. Here we present ohun, an R package intended to facilitate automated sound detection. ohun provides functions to diagnose and optimize detection routines and compare performance among different detection approaches. 3. The package uses reference annotations containing the time position of target sounds in a training data set to evaluate detection routines performance using common signal detection theory indices. This can be done both with routine outputs imported from other software and

detections run within the package. The package also provides functions to organize acoustic data sets in a format amenable to detection analyses. ohun also includes energy-based and template-based detection methods, two commonly used automatic approaches in bioacoustic research.
4. We show how ohun automatically can be used to detect vocal signals with case studies of adult male zebra finch (Taenopygia gutata) songs and Spix's disc-winged bat (Thyroptera tricolor) ultrasonic social calls. We also include examples of how to evaluate the detection performance of ohun and external software. Finally, we provide some general suggestions to improve detection performance.

ohun: an R package for diagnosing and optimizing automatic sound event detection

Marcelo Araya-Salas[1, 2, 3] *, Grace Smith-Vidaurre[4, 5, 6], Gloriana Chaverri[3, 7], Juan C. Brenes[1], Fabiola Chirino[2], Jorge Elizondo-Calvo[2] & Alejandro Rico-Guevara[8, 9]

[1] Centro de Investigación en Neurociencias, Universidad de Costa Rica, San José, Costa Rica

[2] Escuela de Biología, Universidad de Costa Rica, San José, Costa Rica

[3] Sede del Sur, Universidad de Costa Rica, Golfito, Costa Rica

[4] Laboratory of Neurogenetics of Language, Rockefeller University, New York, NY, USA

[5] Rockefeller University Field Center, Millbrook, NY, USA

[6] Department of Biological Sciences, University of Cincinnati, Cincinnati, OH, USA

[7] Smithsonian Tropical Research Institute, Panama City, Panamá

[8] Department of Biology, University of Washington, Seattle, USA

[9] Burke Museum of Natural History and Culture, University of Washington, Seattle, USA

* *To whom correspondence should be addressed*

Keywords: bioacoustics, automatic detection, animal vocalizations

Abstract

Animal acoustic signals are widely used in diverse research areas due to the relative ease with which sounds can be registered across a wide range of taxonomic groups and research settings. However, bioacoustics research can quickly generate large data sets, which might prove challenging to analyze promptly. Although many tools are available for the automated detection of sounds, choosing the right approach can be difficult only a few tools provide a framework for evaluating detection performance. Here we present *ohun*, an R package intended to facilitate automated sound detection. *ohun* provides functions to diagnose and optimize detection routines and compare performance among different detection approaches. The package uses reference annotations containing the time position of target sounds in a training data set to evaluate detection routines performance using common signal detection theory indices. This can be done

both with routine outputs imported from other software and detections run within the package. The package also provides functions to organize acoustic data sets in a format amenable to detection analyses. ohun also includes energy-based and template-based detection methods, two commonly used automatic approaches in bioacoustic research. We show how ohun automatically can be used to detect vocal signals with case studies of adult male zebra finch (*Taenopygia gutata*) songs and Spix's disc-winged bat (*Thyroptera tricolor*) ultrasonic social calls. We also include examples of how to evaluate the detection performance of ohun and external software. Finally, we provide some general suggestions to improve detection performance.

Introduction

Animal acoustic signals are widely used to address a variety of questions in highly diverse areas, ranging from neurobiology (Burgdorf *et al.* 2011; Schöneich 2020) to taxonomy (Köhler *et al.* 2017; Gwee *et al.* 2019), community ecology (Zsebők *et al.* 2021; Tiwari & Diwakar 2022), and evolutionary biology (Medina-García *et al.* 2015; Odom *et al.* 2021). The profuse usage of animal sounds in research relates to the fact that they can be easily collected using non-invasive methods. In addition, animal sounds can be obtained in various natural and unnatural settings, with equipment that has become increasingly inexpensive and broadly accessible (Blumstein *et al.* 2011; Sugai *et al.* 2019). Online repositories have also facilitated the study of these communication signals at larger taxonomic and geographic scales. However, adopting bioacoustic approaches may also imply large amounts of data (i.e., lots of recordings), which can be challenging to analyze manually (Gibb *et al.* 2019). As a result, a growing number of computational tools for automatically-detected animal sounds is increasingly available (reviewed by Stowell 2022), reflecting the need for better and more efficient automated approaches (Gibb *et al.* 2019).

Most available tools for the automatic detection of acoustic events are free software accessible to a wider range of users and scientific questions. However, this diversity of automated detection tools also posits a challenge, as it can be difficult to navigate (Stowell, 2022). In this regard, using standard approaches for evaluating the performance of automatic detection tools might prove helpful in informing researchers' decisions about which method better fits a given question and study system (Knight *et al.* 2017). The performance of automated sound event detection routines has typically been evaluated using standard indices from signal detection theory (Knight *et al.* 2017; Balantic & Donovan 2020). In its basic form, performance is assessed by comparing the output of a detection routine against a 'gold-standard' reference in which all the target sounds have been annotated (hereafter called 'reference annotation'). This comparison facilitates quantifying the number of sounds detected correctly (true positives), wrongly (false positives), and missed (false negatives), as well as additional metrics derived from these indices (e.g., recall, precision).

Nevertheless, the fact that sound events are not evaluated as discrete classification units but are instead embedded within a continuous string of sound demands additional information to fully diagnose the temporal precision of the detection performance. This is particularly relevant if identifying the precise time position of sounds is needed, which is often required when the main goal is measuring the acoustic structure of sounds (Araya-Salas & Smith-Vidaurre 2017). In this line, several detection problems can be encountered; for instance, the same signal can be detected as several separated sounds, the inferred time position can be offset from the target signal position, or several sounds can be detected as one single signal. Therefore, tools containing metrics that account for these additional performance dimensions are valuable for properly diagnosing automatic sound event detection.

Here we present the new R package *ohun*. This package is intended to facilitate the automatic detection of sound events, providing functions to diagnose particular aspects of acoustic detection routines to simplify their optimization. The package uses reference annotations containing the time position of target sounds that, along with the corresponding sound files, serve as a training data set to evaluate the performance of detection routines. This can be done with routine outputs imported from other software and detection routines run within the *ohun* package. The package also provides a set of functions to explore acoustic data sets and organize them in an amenable format for detection analyses. In addition, it offers implementations of two automatic detection methods commonly used in bioacoustic analysis: energy-based detection and template-based detection (Mellinger & Clark 2000; Charif *et al.* 2010; Aide *et al.* 2013; Bioacoustics 2014; Hafner & Katz 2015). Here, we explain how to explore and format acoustic data sets and how sound event detection routines can be evaluated.In addition, we showcase the package usage with study cases on male Zebra-finch songs (*Taenopygia gutata*) and Spix's disc-winged bat calls (*Thyroptera tricolor*), which correspond to different recording settings (i.e., lab and flight cages) and signal types (i.e., sonic mating sounds and ultrasonic social calls).

Formatting acoustic data sets

The format and size of the acoustic data to be analyzed needs to be standardize to avoid downstream errors and to inform expectations for computational time performance. Several functions in *ohun* can facilitate double-checking the format of acoustic data sets prior to automatic detection. The function feature_acoustic_data prints a summary of the duration, size, and format of all the recordings in a folder. Here we explore the acoustic data set of zebra finch's songs (Supporting Information):

```
# working directory
path_zebra_finch <- "path_to_zebra_finch_files"

feature_acoustic_data(path = path_zebra_finch)
```

```
## Features of the acoustic data set in '/home/neuro/Dropbox/Projects/ohun_paper/data/raw/taeniopygi
a':
## * 18 sound files
## * 1 file format(s) (.wav (18))
## * 1 sampling rate(s) (44.1 kHz (18))
## * 1 bit depth(s) (16 bits (18))
## * 1 number of channels (1 channel(s) (18))
## * File duration range: 2.63-21.76 s (mean: 12.22 s)
## * File size range: 0.23-1.92 MB (mean: 1.08 MB)
##  (detailed information by sound file can be obtained with 'warbleR::info_sound_files()')
```

In this case, all recordings have the same format (.wav files, 44.1 kHz sampling rate, 16-bit resolution, and a single channel). We can also check the files' duration and size. Format information is important as some tuning parameters of detection routines can behave differently depending on file format (*e.g.*, time window size can be affected by sampling rate) or simply because some software might only work on specific sound file formats. In addition, long sound files could be difficult to analyze on some computers and might have to be split into shorter clips. In the latter case, the function split_acoustic_data can be used to produce those clips:

```
split_info <- split_acoustic_data(path = path_zebra_finch, sgmt.dur = 5)

head(split_info)
```

original.sound.files	sound.files	start	end
Ag13_43421.27975590_11_17_7_46_15.wav	Ag13_43421.27975590_11_17_7_46_15-1.wav	0	5.000
Ag13_43421.27975590_11_17_7_46_15.wav	Ag13_43421.27975590_11_17_7_46_15-2.wav	5	10.000
Ag13_43421.27975590_11_17_7_46_15.wav	Ag13_43421.27975590_11_17_7_46_15-3.wav	10	15.000
Ag13_43421.27975590_11_17_7_46_15.wav	Ag13_43421.27975590_11_17_7_46_15-4.wav	15	20.000
Ag13_43421.27975590_11_17_7_46_15.wav	Ag13_43421.27975590_11_17_7_46_15-5.wav	20	21.761
Blk109Brn_43559.32349131_4_4_8_59_9.wav	Blk109Brn_43559.32349131_4_4_8_59_9-1.wav	0	5.000

The output shows the time segments in the original sound files to which the clips belong. If an annotation table is supplied (argument 'X'), the function will adjust the annotations, so they refer to the position of the sounds in the clips. This can be helpful when reference tables have been annotated on the original long sound files.

Annotations can also be explored using the function feature_reference, which returns the mean and range of signal duration and gap duration (e.g., time intervals between selections), bottom and top frequency, and the number of annotations by sound file. If the path to the sound files is supplied, then the duty cycle (i.e., the fraction of a sound file corresponding to target sounds) and peak amplitude (i.e., the highest amplitude in a detection) are also returned:

```
# read reference annotations
manual_ref_tae <- read.csv(file.path(path_zebra_finch, "manual_selections_Taeniopygia.csv"))

# explore annotations
feature_reference(reference = manual_ref_tae, path = path_zebra_finch)
```

```
##                  min   mean     max
## sel.duration   15.54 103.30  319.41
## gap.duration   80.19 352.26 3024.23
## annotations     2.00  32.83   66.00
## duty.cycle      0.09   0.29    0.61
## peak.amplitude 17.26  39.16   62.67
## bottom.freq     0.50   0.50    0.50
## top.freq       10.00  10.00   10.00
```

Diagnosing detection performance

The *ohun* package uses signal detection theory indices to evaluate detection performance. Signal detection theory deals with the process of recovering sounds (*i.e.*, target sounds) from background noise –not necessarily acoustic noise– and it is widely used for optimizing this decision-making process in the presence of uncertainty (Hossin & Sulaiman 2015). During a detection routine, the detected 'items' can be classified into four classes: true positives (TPs, target sounds correctly identified as signal), false positives (FPs, noise incorrectly identified as 'signal'), false negatives (FNs, sounds incorrectly identified as noise), and true negatives (TNs, background noise correctly identified as noise). However, TNs cannot always be easily defined in the context of sound event detection, as noise cannot always be partitioned into discrete units. Hence, the package makes use of TPs, FPs, and FNs to calculate three additional indices that can further assist with evaluating the performance of a detection routine and are widely used in sound event detection (Knight *et al.* 2017): recall (i.e., correct detections relative to total detections), precision (i.e., the proportion of target sounds that were correctly detected) and F1 score (combined recall and precision as the harmonic mean of these two, which provides a single value for evaluating performance, a.k.a. F-score, F-measure or Dice similarity coefficient).

A perfect detection will have no false positives or negatives, resulting in both recall and precision equal to 1. However, perfect detection cannot always be achieved. Therefore, some compromise between detecting most target sounds plus some noise and excluding noise but missing target sounds might be warranted. These indices provide a useful framework for diagnosing and optimizing the performance of a detection routine. Researchers can identify an appropriate balance between these two extremes by the relative costs of missing sounds and mistaking noise for target sounds in the context of their specific study goals.

ohun offers tools to evaluate the performance of sound event detection methods based on the indices described above. To accomplish this, annotations derived from a detection routine are compared against a reference table containing the time position of all target sounds in the sound files. For instance, the following code evaluates a routine run in Raven Pro 1.6 (Charif *et al.* 2010) using the "band limited energy detector" option (minimum frequency: 0.8 kHz; maximum frequency: 22 kHz; minimum duration: 0.03968 s; maximum duration: 0.54989s; minimum separation: 0.02268 s) on a subset of the zebra finch recordings described below:

```
raven_detec <- read.csv("combined_raven_detection.csv")

diagnose_detection(reference = manual_ref_tae, detection = raven_detec, by = "tuning_parameters")
```

```
raven_detec <- read.csv("./data/processed/combined_raven_detection.csv")

diagnose_detection(reference = manual_ref_tae, detection = raven_detec, by = "tuning_parameters")
```

tuning_parameters	true.positives	false.positives	false.negatives	split.positives	merged.positives	overlap.to.true.positives	recall	precision	f1.score
band: 1-10 kHz; sep: 0.005 s	540	69	51	19	43	0.912	0.914	0.887	0.900
band: 1-10 kHz; sep: 0.02322 s	591	76	0	0	44	0.983	1.000	0.886	0.940
band: 1-15 kHz; sep: 0.02322 s	590	192	1	11	75	0.909	0.998	0.754	0.859
band: 1-22 kHz; sep: 0.02322.txt s	579	498	12	29	62	0.868	0.980	0.538	0.694

The function diagnose_detection uses bipartite matching graphs (Csardi & Nepusz 2006) to determine optimum matching of detected and reference events, ensuring that each reference sound is only be matched to a single detection (Lostanlen et al. 2019).

The output shows the indices described above, plus three additional metrics specific to sound event detection: 'split positives', 'merge positives', and 'overlap to true positives'. 'Split positives' is the number of reference sounds overlapped by more than one detection, 'merge positives' is the number of detected sounds overlapping with another detection, and 'overlap to true positives' quantifies the mean overlap between detections and reference signal (1 means complete overlap). The function also allows detailing those indices by sound file as well as by additional categorical features. Here we show the first ten files detailed by the column 'tunning_parameters' which contains the combained detection parameter values used in Raven:

```
diag_raven <- diagnose_detection(reference = manual_ref_tae, detection = raven_detec, by = "tuning_para
meters", by.sound.file = TRUE)

head(diag_raven, 10)
```

tuning_parameters	sound.files	true.positives	false.positives	false.negatives	split.positives	merged.positives	overlap.to.true.positives	recall	precision	f1.score
band: 1-10 kHz; sep: 0.005 s	Ag13_43421.27975590_11_17_7_46_15.wav	35	5	0	0	0	0.987	1.000	0.438	0.609
band: 1-10 kHz; sep: 0.005 s	BRN7_43435.27985312_12_1_7_46_25.wav	39	7	12	0	9	0.971	0.765	0.448	0.565
band: 1-10 kHz; sep: 0.005 s	Blk109Brn_43559.32349131_4_4_8_59_9.wav	33	1	1	0	2	0.914	0.971	0.943	0.957
band: 1-10 kHz; sep: 0.005 s	DB118_43568.33139566_4_13_9_12_19.wav	19	0	0	0	1	0.988	1.000	1.000	1.000
band: 1-10 kHz; sep: 0.005 s	DB15HP_43450.29217192_12_16_8_6_57.wav	21	2	0	0	0	0.989	1.000	0.875	0.933
band: 1-10 kHz; sep: 0.005 s	DB7_43357.58119484_9_14_16_8_39.wav	19	0	0	0	1	0.936	1.000	1.000	1.000
band: 1-10 kHz; sep: 0.005 s	DG124DB_43559.30160960_4_4_8_22_40.wav	29	10	0	0	0	0.987	1.000	0.137	0.242
band: 1-10 kHz; sep: 0.005 s	GRY37HP_43442.27431670_12_8_7_37_11.wav	53	6	1	0	3	0.933	0.981	0.707	0.822
band: 1-10 kHz; sep: 0.005 s	Gold183_43555.5549813_3_31_1_32_29.wav	2	0	0	0	0	1.000	1.000	1.000	1.000
band: 1-10 kHz; sep: 0.005 s	Gry35HP_43455.29800260_12_21_8_16_40.wav	26	0	0	0	5	0.993	1.000	1.000	1.000

Diagnostics from routines utilizing different tuning parameters can serve to identify the parameter values that optimize detection. This process of evaluating different routines for detection optimization is incorporated into the two signal detection approaches provided natively by *ohun*, which we depict in the following section. Note that the detection with Raven Pro does not necessarily reflect the best performance of this software and has been included only as an example of evaluating detection from external sources rather than a direct comparison of performance between Raven Pro and *ohun*.

Signal detection with *ohun*

The package offers two methods for automated signal detection: template-based and energy-based detection. These methods are better suited for stereotyped or good signal-to-noise ratio sounds, respectively. If the target sounds do not fit these requirements, more elaborate methods (i.e., machine/deep learning approaches) are warranted.

Study cases

Template detection on ultrasonic social calls of Spix's disc-winged bats

We recorded 30 individuals of Spix's disc-winged bats (*Thyroptera tricolor*) at Baru Biological Station in southwestern Costa Rica in January 2020. Bats were captured at their roosting sites (furled leaves of Zingiberaceae plants). Each bat was released in a large flight cage (9 x 4 x 3 m) for 5 minutes, and their ultrasonic inquiry calls were recorded using a condenser microphone (CM16, Avisoft Bioacoustics, Glienike/Nordbahn, Germany) through an Avisoft UltraSoundGate 116Hm plugged into a laptop computer running Avisoft-Recorder software. Recordings were made at a sampling rate of 500 kHz and an amplitude resolution of 16 bits.

Recordings were manually annotated using Raven Pro 1.6 (Charif *et al.* 2010). Annotations were created by visual inspection of spectrograms, in which the start and end of sounds were determined by the location of the continuous traces of power spectral entropy of the target sounds. A total of 644 calls were annotated (~21 calls per recording). Annotations were made with a time window of 200 samples and 70% overlap and were then imported into R using the package Rraven (Araya-Salas 2020).

Inquiry calls of Spix's disc-winged bats are structurally stereotyped (Chaverri *et al.* 2010). Most variation is found among individuals, although the basic form of a short, downward broadband frequency modulation is always shared (Fig. 1, Araya-Salas et al. 2021).

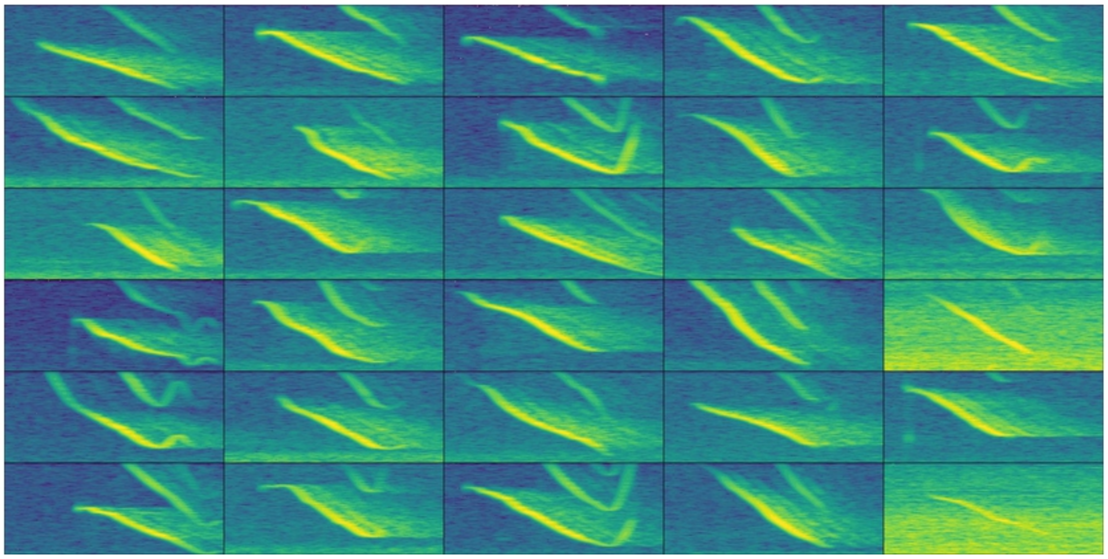


Figure 1. Example spectrograms of Spix's disc-winged social calls for each of the 30 recordings used in the analysis. The highest signal-to-noise ratio call by sound file are shown. The time scale range is 71 ms and the frequency range 10-44 kHz.

A template-based detection is a useful approach when there are minimal structural differences in the target sounds (Knight *et al.* 2017; e.g., when signals are produced in a highly stereotyped manner, Balantic & Donovan 2020). It uses spectrographic cross-correlation to find sounds resembling an example target sounds (i.e., template) across sound files, applying a correlation threshold to separate detections from background noise. We used this approach in *ohun* to detect inquiry calls. To do this, we tested the performance of three acoustic templates on a training subset of five sound files. First, we used the function get_templates to find several sounds representative of the variation in signal structure. This function measures several spectral features, which are summarized using Principal Component Analysis. The first two components are used to project the acoustic space. In this space, the function defines sub-spaces as equal-sized slices of a sphere centered at the centroid of the acoustic space. Templates are then selected as those closer to the centroid within sub-spaces, including the centroid for the entire acoustic space. The user needs to define the number of sub-spaces in which the acoustic space will be split.

```
# read manual annotations
manual_ref_thy <- read.csv("manual_annotations_thyroptera.csv")

# get random subset of 5 sound files for training
set.seed(1)
train_files <- sample(unique(manual_ref_thy$sound.files), size = 5)
train_ref <- manual_ref_thy[manual_ref_thy$sound.files %in% train_files, ]

# the rest for testin g
test_files <- setdiff(manual_ref_thy$sound.files, train_files)
test_ref <- manual_ref_thy[manual_ref_thy$sound.files %in% test_files, ]

# find templates
templates <- get_templates(train_ref, path = data_path,  bp = c(10, 50), ovlp = 70, wl = 200, n.sub.spa
ces = 3)
```

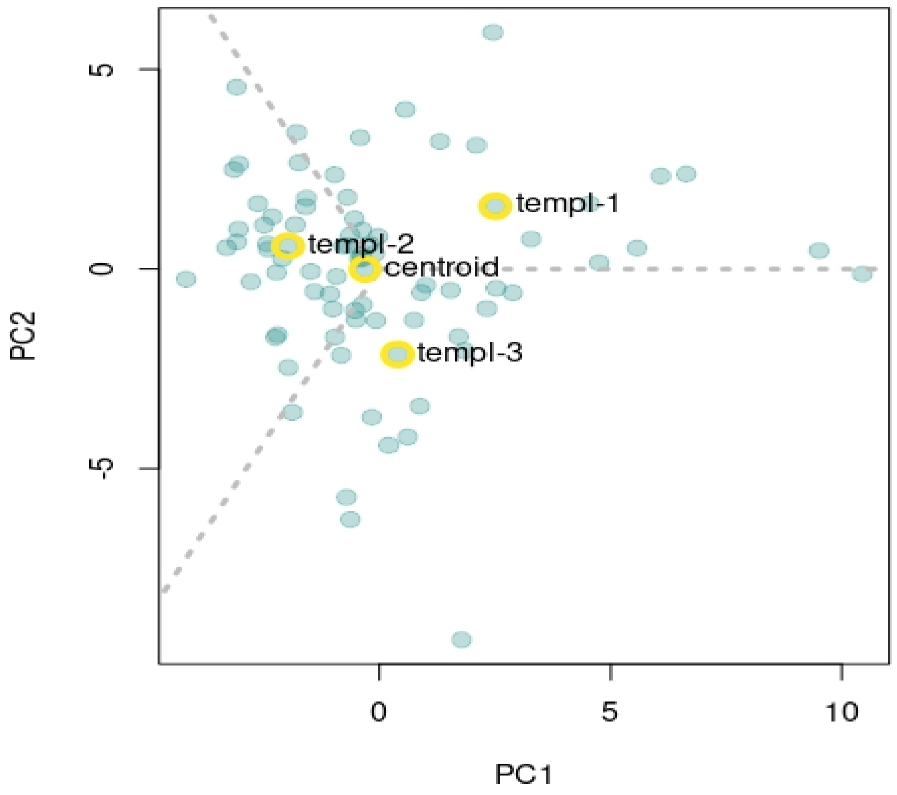


Figure 2. Acoustic space defined as the first two components of a Principal Component Analysis on spectrographic parameters. Templates are selected as those closer to the centroid within sub-spaces. Gray dashed lines delimit the region of sub-spaces. Yellow circles around points highlight the position of the signals selected as templates.

The output of the get_templates function includes an acoustic space plot (Fig. 2) in which the position of the sounds selected as templates is highlighted. Users can also provide their own acoustic space dimensions (argument 'acoustic.space'). In the following code, we used the templates determined above for detecting bat social calls. The code iterates a template-based detection on the training data set across a range of correlation thresholds for each template, in order to find the combination of threshold and template with the best performance.

```
# get correlation vectors
corr_templ_train <- template_correlator(
  templates = templates,
    path = data_path,
  files = unique(train_ref$sound.files),
  hop.size = 10,
  ovlp = 70
  )

# evaluate detection for different correlation thresholds
opt_detec_train <- optimize_template_detector(
  reference = train_ref,
  template.correlations = corr_templ_train,
  threshold = seq(0.05, 0.5, 0.01)
  )
```

Note that the correlation vectors are estimated first (i.e., vectors of correlation values across sound files, template_correlator), and then the correlation thresholds are optimized on these vectors (optimize_template_detector). The output of optimize_template_detector contains the detection performance indices for each combination of templates and thresholds. Table 1 shows the two highest performance runs for each template.

Table 1. Performance diagnostic of template-based detections using four templates across several threshold values. Only the two highest performance iterations for each template are shown.

threshold	templates	true.positives	false.positives	false.negatives	recall	precision	f1.score
0.45	centroid	81	1	0	1.000	0.988	0.994
0.50	centroid	80	0	1	0.988	1.000	0.994
0.50	templ-1	77	1	4	0.951	0.987	0.969
0.45	templ-1	80	5	1	0.988	0.941	0.964
0.40	templ-2	79	0	2	0.975	1.000	0.988
0.35	templ-2	80	2	1	0.988	0.976	0.982
0.45	templ-3	77	2	4	0.951	0.975	0.963
0.40	templ-3	79	5	2	0.975	0.940	0.958

We can explore the performance of each template in more detail by looking at the change in F1 score across thresholds (Fig. 3).

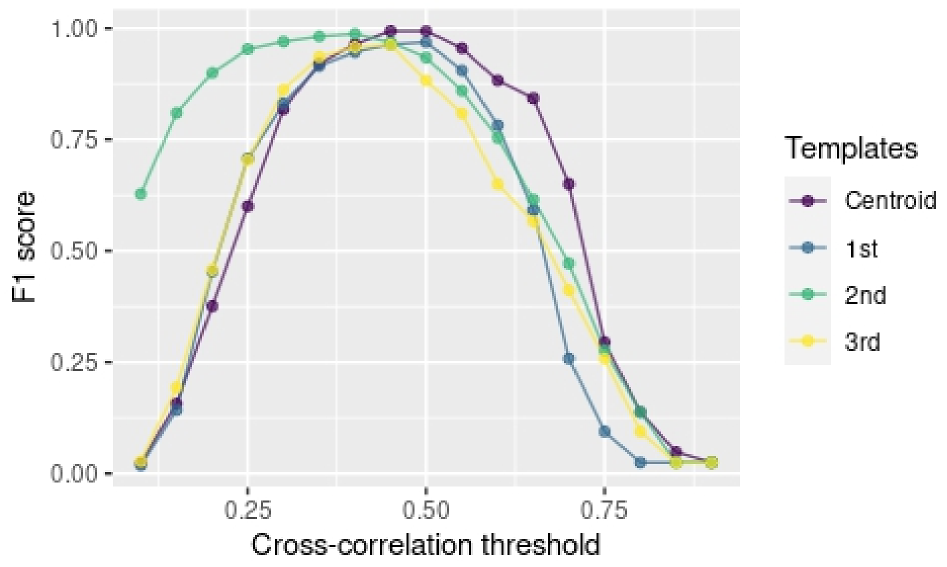


Figure 3. Changes in F1 score across the range of cross-correlation threshold values for four sound templates.

In this example, the "centroid" template produced the best performance (although not drastically different from other templates; Table 1; Fig. 3). Hence, we will use this template for detecting calls on the rest of the data. The following code extracts this template from the reference annotation table and uses it to find inquiry calls on the testing data set:

```
# get correlation vectors for test files
corr_templ_test <- template_correlator(
  templates = templates[templates$sound.file == "centroid", ],
    path = data_path, files = unique(test_ref$sound.files),
  hop.size = 10,
  ovlp = 70
  )

# detect on test files
detec_test <- template_detector(
  template.correlations = corr_templ_test,
    threshold = 0.45
  )

diagnose_detection(reference = test_ref, detection = detec_test)
```

true.positives	false.positives	false.negatives	split.positives	merged.positives	overlap.to.true.positives	recall	precision	f1.score
536	10	27	2	16	0.906	0.952	0.982	0.967

The last line of codes evaluates the detection on the test data set, which shows a good performance for both recall and precision (0.95 and 0.98 respectively).

Energy-based detection on zebra finch vocalizations

We used recordings from 18 zebra finch males recorded at the Rockefeller University Field Research Center Song Library (http://ofer.sci.ccny.cuny.edu/songs, Tchernichovski *et al.* 2021). Recordings contain undirected vocalizations (e.g., songs or calls) of single males recorded in sound attenuation chambers using Sound Analysis Pro. Zebra finch vocalizations are composed of multiple elements (i.e., distinct patterns of continuous traces of power spectral entropy in the spectrogram separated by time gaps) that can vary substantially in key features such as duration and frequency range (Fig. 4) and are not nearly as stereotyped as the Spix's disc-winged bats. However, as recorded sounds show a good signal-to-noise ratio, signals in each recording can potentially be detected using an energy-based approach that does not rely on matching the acoustic structure of a template.

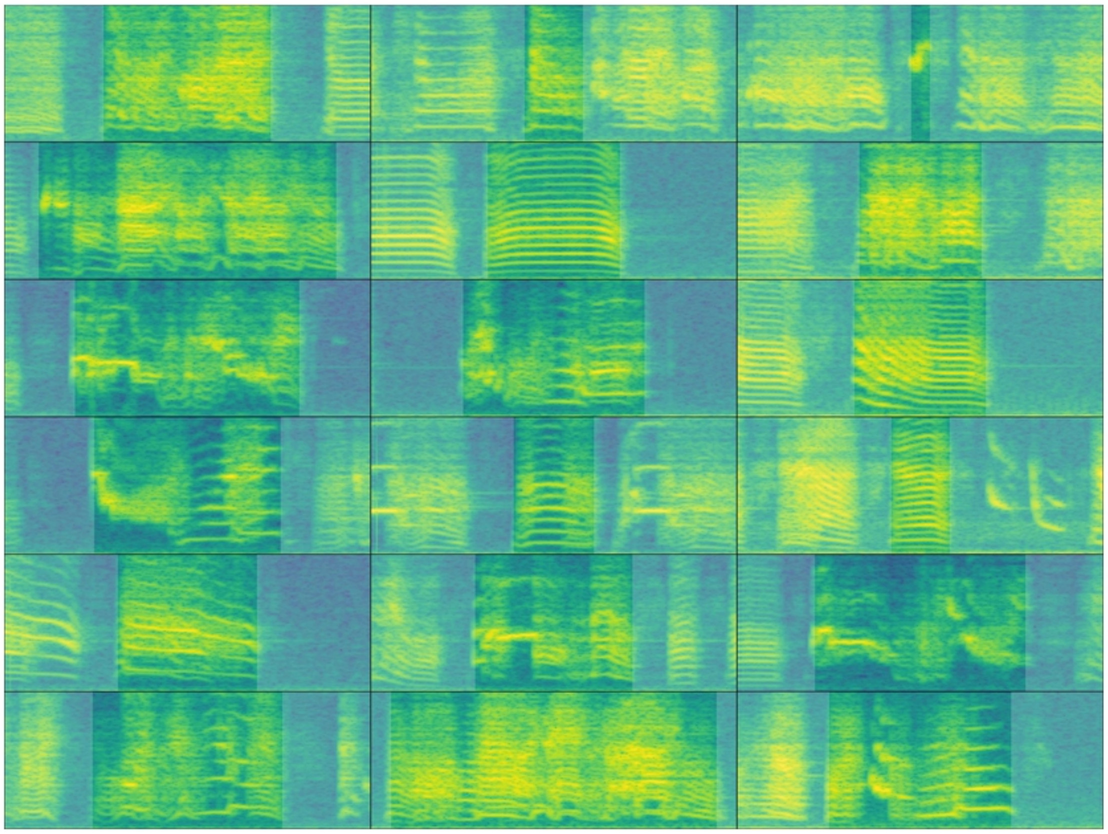


Figure 4. Example spectrograms of male zebra finch songs for each of the 18 sound files used in the analysis. The highest signal-to-noise ratio call by sound file are shown. The time scale range is 359 ms and the frequency range 0-11 kHz. Signals have been highlighted for visualization purposes only.

Reference annotations were made manually on the oscillogram with the spectrogram and audio as a guide using Raven Lite 2.0.1 (Cornell Lab of Ornithology). The following code loads the reference annotations and split them into two data sets for training (3 sound files) and testing (15 sound files):

```r
manual_ref_tae <- read.csv("manual_selections_Taeniopygia.csv")

set.seed(450)
train_files <- sample(unique(manual_ref_tae$sound.files), 3)
test_files <- setdiff(manual_ref_tae$sound.files, train_files)

train_ref <- manual_ref_tae[manual_ref_tae$sound.files %in% train_files, ]
test_ref <- manual_ref_tae[manual_ref_tae$sound.files %in% test_files, ]
```

The detection parameters can be optimized using the function optimize_energy_detector. This function runs a detection for all possible combinations of tuning parameters. The code below tries three minimum duration and maximum duration values and two hold time values (which merges sounds within the specified time interval):

```r
opt_det_train <- optimize_energy_detector(
  reference = train_ref,
  files = train_files,
  threshold = c(1, 5),
  hop.size = 11.6,
  smooth = c(5, 10),
  hold.time = c(0, 5),
  min.duration = c(5, 15, 25),
  max.duration = c(275, 300, 325),
  bp = c(0.5, 10)
)
```

The output (opt_det_train) shows the performance indices for each of those combinations. Here we show the ten combinations with the highest F1 score:

```r
# subset with highest performance
opt_det_train <- opt_det_train[order(opt_det_train$f1.score, decreasing = TRUE), ]

head(opt_det_train, 10)
```

threshold	smooth	hold.time	min.duration	max.duration	true.positives	false.positives	false.negatives	recall	precision	f1.score
1	5	5	25	300	105	9	0	1.000	0.921	0.959
1	5	5	25	325	105	9	0	1.000	0.921	0.959
1	5	0	25	300	105	12	0	1.000	0.897	0.946
1	5	0	25	325	105	12	0	1.000	0.897	0.946
1	10	0	25	300	105	13	0	1.000	0.890	0.942
1	10	0	25	325	105	13	0	1.000	0.890	0.942
1	5	5	15	300	105	15	0	1.000	0.875	0.933
1	5	5	15	325	105	15	0	1.000	0.875	0.933
5	5	5	15	275	96	8	9	0.914	0.923	0.919
5	5	5	15	300	96	8	9	0.914	0.923	0.919

We now can use the tuning parameter values that yielded the best performance to detect sounds on the test data set:

```
best_param <- opt_det_train[which.max(opt_det_train$f1.score), ]

det_test <- energy_detector(
  files = test_files,
  threshold = best_param$threshold,
  hop.size = 11.6,
  smooth = best_param$smooth,
  hold.time = best_param$hold.time,
  min.duration = best_param$min.duration,
  max.duration = best_param$max.duration,
  bp = c(0.5, 10)
)
```

As our reference annotations include all sounds in both the training and test annotations, we can evaluate the performance of the detection on the test set as well:

```
diagnose_detection(reference = test_ref, detection = det_test, by.sound.file = FALSE)
```

true.positives	false.positives	false.negatives	split.positives	merged.positives	overlap.to.true.positives	recall	precision	f1.score
475	27	11	4	16	0.97	0.977	0.946	0.962

The performance on the test data set was also acceptable, with an F1 score of 0.95. Note that in the example we used a small subset of sound files for training. More training data might be needed for optimizing a detection routine on larger data sets or recordings with more variable sounds or background noise levels.

Additional tools

The *ohun* package offers additional tools to simplify sound event detection. Detected sounds can be labeled as false or true positives with the function 'label_detection'. This allows users to explore the structure of false positives and figure out ways to exclude them. The function 'filter_detections' can remove ambiguous sounds (i.e., those labeled as split or merged detections), keeping only those that maximize a specific criterium (i.e., the highest template correlation). Finally, note that several templates representing the range of variation in signal structure can be used to detect semi-stereotyped sounds or sterepotyped multi-element repertoires when running template-based detection ('template_detection' function).

Discussion

Here we have shown how to evaluate the performance of sound event detection routines using the package *ohun*. The package can evaluate detection outputs imported from other software, as well as its own detection routines. The latter can be iterated over combinations of tuning parameters to find those values that optimize detection. Although signal detection indices are commonly reported when presenting new automatic detection methods, to our knowledge, in addtion to *ohun* there is only one other performance-evaluating software developed in a free, open-source platform (sed-eval, Mesaros 2016). These two software packages can provide a common framework for evaluating sound event detection can simplify comparing the performance of different tools and selecting those tools better suited to a given research question and study system. The tools offered by *ohun* for diagnosing detection performance should not necessarily be limited to acoustic data. *ohun* can also be used for cases in which the time of occurrence of discrete events needs to be identified, such as detecting specific behaviors in video analysis of animal motor activity (*e.g.*, Sturman *et al.* 2020; Hsu & Yttri 2021; Bohnslav *et al.* 2021). The detection of such motor events in video recordings can also be evaluated and optimized compared to a reference annotation, as we have shown here for sound events.

The *ohun* package provides two detection methods: template-based and energy-based detection. Compared to new deep learning approaches for finding the occurrence of sound events, the two native methods are relatively simple tools. However, these methods have been widely used by the bioacoustic community (Mellinger & Clark 2000; Charif *et al.* 2010; Aide *et al.* 2013; Specth 2002; Hafner & Katz 2015) and can reach adequate performance under the appropriate conditions, as evidenced by our two study cases and from previous reports (Knight *et al.* 2017). Deep learning methods tend to require greater computational power, larger training data sets, and, in some cases, more complex training routines (but see transfer learning approaches). This might bring unnecessary difficulties when dealing with less challenging detection tasks. Therefore, the availability of a wide range of approaches can simplify finding the most appropriate tool for the intricacies of a study system and research goals and having tools accessible to a broader research community. The tools offered in *ohun* can also be used in a subsequent pipeline in which detected sounds are further classified and false positives are mitigated using more elaborated discrimination algorithms (Balantic & Donovan 2020). Detection performance might be improved by using acoustic structure measurements to distinguish target from non-target sound events.

Detection routines can take a long time when working with large amounts of acoustic data (e.g. long recordings or many files). We provide some additional tips that can help make a routine more time efficient. 1. Always test procedures on small data subsets. Make sure to obtain decent results on a small subset of recordings before scaling up the analysis. 2. Template-based detection is almost always faster than energy-based detection. 3. Run routines in parallel. Parallelization (i.e., the ability to distribute tasks over several cores in your computer) can significantly speed up routines. All automatic detection and performance evaluation functions in *ohun* allow users to run analysis in parallel (see parallel argument in those functions). Hence, a computer with several cores can help improve efficiency. 4. Try using a computer with lots of RAM or a computer cluster for working on large amounts of data. 5. Sampling rate matters. Detecting sounds on low sampling rate files is faster, so we must avoid having Nyquist frequencies much higher than the highest frequency of the target sounds. Lastly, we underscore that these tips are not restricted to *ohun* and can also be helpful to speed up routines in other software packages.

Other things should be considered when aiming to detect sound events automatically. When running energy-based detection routines, try to use your knowledge of the signal structure to determine the initial range of tuning parameters. This can be extremely helpful for narrowing down possible parameter values. As a general rule, if human observers have difficulty detecting where a target sound occurs in a sound file, detection algorithms will likely yield low detection performance. In cases in which occurrences are ambiguous, low performances are expected. Ensure reference annotations contain all target sounds and only the target sounds. Otherwise, performance optimization can be misleading as the performance of a given detection method cannot be better than the reference itself. Lastly, avoid having overlapping sounds or several sounds as a single detection (e.g., a multi-syllable vocalization) in the reference annotation when running an energy-based detector, as they are likely to be identified as separated units.

Acknowledgments

Nazareth Rojas, Silvia Chaves-Ramírez, Mariela Sánchez-Chavarría, Miriam Gioiosa, Cristian Castillo-Salazar, and José Pablo Barrantes for their help collecting acoustic data for Spix's disc-winged bats. We also thank the Centro Biológico Hacienda Barú for their continuous support of our research and Mijail Rojas and Andrey Sequeira for support in the early stages of package development. This study was partly funded by a CONARE-Max Planck grant from the Consejo Nacional de Rectores and the research activity C0754 at Centro de Investigación en Neurociencias, Universidad de Costa Rica. A.R-G. is supported by the Walt Halperin Endowed Professorship and the Washington Research Foundation as Distinguished Investigator.

Ethical note

All sampling protocols followed guidelines approved by the American Society of Mammalogists for the capture, handling, and care of mammals (Sikes *et al.* 2016) and the ASAB/ABS Guidelines for the use of animals in research. This study was conducted in accordance with the ethical standards for animal welfare of the Costa Rican Ministry of Environment and Energy, Sistema Nacional de Áreas de Conservación, permit no. SINAC-ACOPAC-RES-INV-008-2017 (Decree No. 32553-MINAE). Protocols were also approved by the University of Costa Rica's Institutional Animal Care and Use Committee (CICUA-42-2018).

Supporting Information

Supplementary information associated with this article is available in 10.6084/m9.figshare.21675692

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, 2013.

Araya-Salas, M. (2020). Rraven: Connecting r and raven bioacoustic software. R package version 1.0.9.

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

Balantic, C.M. & Donovan, T.M. (2020). Statistical learning mitigation of false positives from template-detected data in automated acoustic wildlife monitoring. BIoacoustics, 29, 293–321.

Blumstein, D.T., Mennill, D.J., Clemins, P., Girod, L., Yao, K., Patricelli, G., Deppe, J.L., Krakauer, A.H., Clark, C., Cortopassi, K.A., Hanser, S.F., Mccowan, B., Ali, A.M. & Kirschel, A.N.G. (2011). Acoustic monitoring in terrestrial environments using microphone arrays: Applications, technological considerations and prospectus. Journal of Applied Ecology, 48, 758–767.

Bohnslav, J.P., Wimalasena, N.K., Clausing, K.J., Dai, Y.Y., Yarmolinsky, D.A., Cruz, T., Kashlan, A.D., Chiappe, M.E., Orefice, L.L., Woolf, C.J. & Harvey, C.D. (2021). DeepEthogram, a machine learning pipeline for supervised behavior classification from raw pixels. eLife, 10.

Burgdorf, J., Panksepp, J. & Moskal, J.R. (2011). Frequency-modulated 50kHz ultrasonic vocalizations: A tool for uncovering the molecular substrates of positive affect. Neuroscience and Biobehavioral Reviews, 35, 1831–1836.

Csardi G, Nepusz T (2006). The igraph software package for complex network research. InterJournal, Complex Systems, 1695.

Charif, R., Waack, A. & Strickman, L. (2010). Raven pro 1.4 user's manual. Cornell Lab of Ornithology.

Chaverri, G., Gillam, E.H. & Vonhof, M.J. (2010). Social calls used by a leaf-roosting bat to signal location. Biology Letters, 6, 441–444.

Gibb, R., Browning, E., Glover-Kapfer, P., Jones, K.E. & Jones, C.E.K. (2019). Emerging opportunities and challenges for passive acoustics in ecological assessment and monitoring. Wiley Online Library, 10, 169–185.

Gwee, C., Eaton, J., Garg, K. & Alström, P. (2019). Cryptic diversity in cyornis (aves: Muscicapidae) jungle-flycatchers flagged by simple bioacoustic approaches. Zoological Journal, 186.

Hafner, S. & Katz, J. (2015). monitoR: Acoustic template detection in r. R package version 1.0.3.

Hossin, M. & Sulaiman, M. (2015). A review on evaluation metrics for data classification evaluations. International Journal of Data Mining. 5.2 (2015): 1.

Hsu, A. & Yttri, E. (2021). B-SOiD, an open-source unsupervised algorithm for identification and fast prediction of behaviors. Nature Communications. 12.1 (2021): 1-13.

Knight, E.C., Hannah, K.C., Foley, G.J., Scott, C.D., Brigham, R.M. & Bayne, E. (2017). Recommendations for acoustic recognizer performance assessment with application to five common automated signal recognition programs. Avian Conservation and Ecology, 12.

Köhler, J., Jansen, M., Rodríguez, A., Kok, P.J.R., Felipe, T.L., Emmrich, M., Glaw, F., Haddad, C.F.B., Mark-Oliver, R., Vences, M., Toledo, L.F., Emmrich, M., Glaw, F., Haddad, C.F.B., Rödel, M.O. & Vences, M. (2017). The use of bioacoustics in anuran taxonomy: Theory, terminology, methods and recommendations for best practice. Zootaxa 4251.1 (2017): 1-124.

Lostanlen V, Salamon J, Farnsworth A, Kelling S, Bello JP (2019). Robust sound event detection in bioacoustic sensor networks. PLoS ONE 14(10): e0214168.

Medina-García, A., Araya-Salas, M. & Wright, T.F. (2015). Does vocal learning accelerate acoustic diversification? Evolution of contact calls in neotropical parrots. Journal of Evolutionary Biology, 28, 1782–1792.

Mellinger, D.K. & Clark, C.W. (2000). Recognizing transient low-frequency whale sounds by spectrogram correlation. The Journal of the Acoustical Society of America, 107, 3518–3529.

Mesaros A, T Heittola, & T. Virtanen. (2016). Metrics for polyphonic sound event detection. Applied Sciences, 6(6):162, 2016

Odom, K.J., Araya-Salas, M., Morano, J.L., Ligon, R.A., Leighton, G.M., Taff, C.C., Dalziell, A.H., Billings, A.C., Germain, R.R., Pardo, M., Andrade, L.G. de, Hedwig, D., Keen, S.C., Shiu, Y., Charif, R.A., Webster, M.S. & Rice, AN (2021). Comparative bioacoustics: A roadmap for quantifying and comparing animal sounds across diverse taxa. Biological Reviews, 96, 1135–1159.

Schöneich, S. (2020). Neuroethology of acoustic communication in field crickets-from signal generation to song recognition in an insect brain. Progress in Neurobiology 194: 101882.

Sikes, R.S., Animal Care, the & American Society of Mammalogists, U.C. of the. (2016). 2016 guidelines of the american society of mammalogists for the use of wild mammals in research and education. Journal of Mammalogy, 97, 663–688.

Specht, R. (2002). Avisoft-saslab pro: sound analysis and synthesis laboratory. Avisoft Bioacoustics. 1-723.

Stowell, D. (2022). Computational bioacoustics with deep learning: A review and roadmap. PeerJ, 10, e13152.

Sturman, O., Ziegler, L. von, Schläppi, C. & Akyol, F. (2020). Deep learning-based behavioral analysis reaches human accuracy and is capable of outperforming commercial solutions. Neuropsychopharmacology. 45.11 (2020): 1942-1952.

Sugai, L., Silva, T., Jr, J.R. & Llusia, D. (2019). Terrestrial passive acoustic monitoring: Review and perspectives. BioScience 69.1 (2019): 15-25.

Tchernichovski, O., Eisenberg-Edidin, S. & Jarvis, E.D. (2021). Balanced imitation sustains song culture in zebra finches. Nature Communications 2021 12:1, 12, 1–14.

Tiwari, C. & Diwakar, S. (2022). The katydid country: Bioacoustics and ecology of tettigoniid communities from the indian subcontinent. Bioacoustics (2022): 1-25.

Zsebők, S., Schmera, D., Laczi, M., Nagy, G., Vaskuti, É., Török, J., & Zsolt Garamszegi, L. (2021). A practical approach to measuring the acoustic diversity by community ecology methods. Methods in Ecology and Evolution, 12(5), 874-884.

Title of submission	ohun: an R package for diagnosing and optimizing automatic sound event detection
Authors	Marcelo Araya-Salas, Grace Smith-Vidaurre, Gloriana Chaverri, Juan C. Brenes, Fabiola Chirino, Jorge Elizondo-Calvo, Alejandro Rico-Guevara
Contact details for issues with the software/application	marcelo.araya@ucr.ac.cr
Software location (including archive name and persistent identifier, e.g. doi)	https://github.com/maRce10/ohun
Operating system	linux, windows, macOS
Programming language and minimum version needed	R >= 3.2.1
Additional system requirements	C++11, fftw3, GNU make, sox, Ghostscript, libsndfile
Dependencies	R (>= 3.2.1), tuneR, warbleR (>= 1.1.28), viridis, cli, methods, stats, utils, seewave (>= 2.0.1), fftw, rlang, sp, igraph, Sim.DiffProc
Installation instructions	remotes::install_github("maRce10/ohun")
Test coverage	78% https://app.codecov.io/gh/maRce10/ohun?branch=master
Continuous Integration	github-actions https://github.com/maRce10/ohun/actions
License	GPL (>= 2)
Version and date published	0.1.0, jan-13-2023

ohun: an R package for diagnosing and optimizing automatic sound event detection

Marcelo Araya-Salas[1, 2, 3] *, Grace Smith-Vidaurre[4, 5, 6], Gloriana Chaverri[3, 7], Juan C. Brenes[1], Fabiola Chirino[2], Jorge Elizondo-Calvo[2] & Alejandro Rico-Guevara[8, 9]

[1] Centro de Investigación en Neurociencias, Universidad de Costa Rica, San José, Costa Rica

[2] Escuela de Biología, Universidad de Costa Rica, San José, Costa Rica

[3] Sede del Sur, Universidad de Costa Rica, Golfito, Costa Rica

[4] Laboratory of Neurogenetics of Language, Rockefeller University, New York, NY, USA

[5] Rockefeller University Field Center, Millbrook, NY, USA

[6] Department of Biological Sciences, University of Cincinnati, Cincinnati, OH, USA

[7] Smithsonian Tropical Research Institute, Panama City, Panamá

[8] Department of Biology, University of Washington, Seattle, USA

[9] Burke Museum of Natural History and Culture, University of Washington, Seattle, USA

* *To whom correspondence should be addressed*

Keywords: bioacoustics, automatic detection, animal vocalizations

Abstract

1. Animal acoustic signals are widely used in diverse research areas due to the relative ease with which sounds can be registered across a wide range of taxonomic groups and research settings. However, bioacoustics research can quickly generate large data sets, which might prove challenging to analyze promptly. Although many tools are available for the automated detection of sounds, choosing the right approach can be difficult only a few tools provide a framework for evaluating detection performance.

2. Here we present *ohun*, an R package intended to facilitate automated sound detection. *ohun* provides functions to diagnose and optimize detection routines and compare performance among different detection approaches.

3. The package uses reference annotations containing the time position of target sounds in a training data set to evaluate detection routines performance using common signal detection theory indices. This can be done both with routine outputs imported from other software and detections run within the package. The package also provides functions to organize acoustic data sets in a format amenable to detection analyses. ohun also includes energy-based and template-based detection methods, two commonly used automatic approaches in bioacoustic research.

4. We show how ohun automatically can be used to detect vocal signals with case studies of adult male zebra finch (*Taenopygia gutata*) songs and Spix's disc-winged bat (*Thyroptera tricolor*) ultrasonic social calls. We also include examples of how to evaluate the detection performance of ohun and external software. Finally, we provide some general suggestions to improve detection performance.

Introduction

Animal acoustic signals are widely used to address a variety of questions in highly diverse areas, ranging from neurobiology (Burgdorf *et al.* 2011; Schöneich 2020) to taxonomy (Köhler *et al.* 2017; Gwee *et al.* 2019), community ecology (Zsebők *et al.* 2021; Tiwari & Diwakar 2022), and evolutionary biology (Medina-García *et al.* 2015; Odom *et al.* 2021). The profuse usage of animal sounds in research relates to the fact that they can be easily collected using non-invasive methods. In addition, animal sounds can be obtained in various natural and unnatural settings, with equipment that has become increasingly inexpensive and broadly accessible (Blumstein *et al.* 2011; Sugai *et al.* 2019). Online repositories have also facilitated the study of these communication signals at larger taxonomic and geographic scales. However, adopting bioacoustic approaches may also imply large amounts of data (i.e., lots of recordings), which can be challenging to analyze manually (Gibb *et al.* 2019). As a result, a growing number of computational tools for automatically-detected animal sounds is increasingly available (reviewed by Stowell 2022), reflecting the need for better and more efficient automated approaches (Gibb *et al.* 2019).

Most available tools for the automatic detection of acoustic events are free software accessible to a wider range of users and scientific questions. However, this diversity of automated detection tools also posits a challenge, as it can be difficult to navigate (Stowell, 2022). In this regard, using standard approaches for evaluating the performance of automatic detection tools might prove helpful in informing researchers' decisions about which method better fits a given question and study system (Knight *et al.* 2017). The performance of automated sound event detection routines has typically been evaluated using standard indices from signal detection theory (Knight *et al.* 2017; Balantic & Donovan 2020). In its basic form, performance is assessed by comparing the output of a detection routine against a 'gold-standard' reference in which all the target sounds have been annotated (hereafter called 'reference annotation'). This comparison facilitates quantifying the number of sounds detected correctly (true positives), wrongly (false positives), and missed (false negatives), as well as additional metrics derived from these indices (e.g., recall, precision).

Nevertheless, the fact that sound events are not evaluated as discrete classification units but are instead embedded within a continuous string of sound demands additional information to fully diagnose the temporal precision of the detection performance. This is particularly relevant if identifying the precise time position of sounds is needed, which is often required when the main goal is measuring the acoustic structure of sounds (Araya-Salas & Smith-Vidaurre 2017). In this line, several detection problems can be encountered; for instance, the same signal can be detected as several separated sounds, the inferred time position can be offset from the target signal position, or several sounds can be detected as one single signal. Therefore, tools containing metrics that account for these additional performance dimensions are valuable for properly diagnosing automatic sound event detection.

Here we present the new R package *ohun*. This package is intended to facilitate the automatic detection of sound events, providing functions to diagnose particular aspects of acoustic detection routines to simplify their optimization. The package uses reference annotations containing the time position of target sounds that, along with the corresponding sound files, serve as a training data set to evaluate the performance of detection routines. This can be done with routine outputs imported from other software and detection routines run within the *ohun* package. The package also provides a set of functions to explore acoustic data sets and organize them in an amenable format for detection analyses. In addition, it offers implementations of two automatic detection methods commonly used in bioacoustic analysis: energy-based detection and template-based detection (Mellinger & Clark 2000; Charif *et al.* 2010; Aide *et al.* 2013; Bioacoustics 2014; Hafner & Katz 2015). Here, we explain how to explore and format acoustic data sets and how sound event detection routines can be evaluated.In addition, we showcase the package usage with study cases on male Zebra-finch songs (*Taenopygia gutata*) and Spix's disc-winged bat calls (*Thyroptera tricolor*), which correspond to different recording settings (i.e., lab and flight cages) and signal types (i.e., sonic mating sounds and ultrasonic social calls).

Formatting acoustic data sets

The format and size of the acoustic data to be analyzed needs to be standardize to avoid downstream errors and to inform expectations for computational time performance. Several functions in *ohun* can facilitate double-checking the format of acoustic data sets prior to automatic detection. The function feature_acoustic_data prints a summary of the duration, size, and format of all the recordings in a folder. Here we explore the acoustic data set of zebra finch's songs (Supporting Information):

```
# working directory
path_zebra_finch <- "path_to_zebra_finch_files"

feature_acoustic_data(path = path_zebra_finch)
```

```
## Features of the acoustic data set in '/home/neuro/Dropbox/Projects/ohun_paper/data/raw/taeniopygi
a':
## * 18 sound files
## * 1 file format(s) (.wav (18))
## * 1 sampling rate(s) (44.1 kHz (18))
## * 1 bit depth(s) (16 bits (18))
## * 1 number of channels (1 channel(s) (18))
## * File duration range: 2.63-21.76 s (mean: 12.22 s)
## * File size range: 0.23-1.92 MB (mean: 1.08 MB)
##  (detailed information by sound file can be obtained with 'warbleR::info_sound_files()')
```

In this case, all recordings have the same format (.wav files, 44.1 kHz sampling rate, 16-bit resolution, and a single channel). We can also check the files' duration and size. Format information is important as some tuning parameters of detection routines can behave differently depending on file format (*e.g.*, time window size can be affected by sampling rate) or simply because some software might only work on specific sound file formats. In addition, long sound files could be difficult to analyze on some computers and might have to be split into shorter clips. In the latter case, the function split_acoustic_data can be used to produce those clips:

```
split_info <- split_acoustic_data(path = path_zebra_finch, sgmt.dur = 5)

head(split_info)
```

original.sound.files	sound.files	start	end
Ag13_43421.27975590_11_17_7_46_15.wav	Ag13_43421.27975590_11_17_7_46_15-1.wav	0	5.000
Ag13_43421.27975590_11_17_7_46_15.wav	Ag13_43421.27975590_11_17_7_46_15-2.wav	5	10.000
Ag13_43421.27975590_11_17_7_46_15.wav	Ag13_43421.27975590_11_17_7_46_15-3.wav	10	15.000
Ag13_43421.27975590_11_17_7_46_15.wav	Ag13_43421.27975590_11_17_7_46_15-4.wav	15	20.000
Ag13_43421.27975590_11_17_7_46_15.wav	Ag13_43421.27975590_11_17_7_46_15-5.wav	20	21.761
Blk109Brn_43559.32349131_4_4_8_59_9.wav	Blk109Brn_43559.32349131_4_4_8_59_9-1.wav	0	5.000

The output shows the time segments in the original sound files to which the clips belong. If an annotation table is supplied (argument 'X'), the function will adjust the annotations, so they refer to the position of the sounds in the clips. This can be helpful when reference tables have been annotated on the original long sound files.

Annotations can also be explored using the function feature_reference, which returns the mean and range of signal duration and gap duration (e.g., time intervals between selections), bottom and top frequency, and the number of annotations by sound file. If the path to the sound files is supplied, then the duty cycle (i.e., the fraction of a sound file corresponding to target sounds) and peak amplitude (i.e., the highest amplitude in a detection) are also returned:

```r
# read reference annotations
manual_ref_tae <- read.csv(file.path(path_zebra_finch, "manual_selections_Taeniopygia.csv"))

# explore annotations
feature_reference(reference = manual_ref_tae, path = path_zebra_finch)
```

```
##                  min   mean     max
## sel.duration   15.54 103.30  319.41
## gap.duration   80.19 352.26 3024.23
## annotations     2.00  32.83   66.00
## duty.cycle      0.09   0.29    0.61
## peak.amplitude 17.26  39.16   62.67
## bottom.freq     0.50   0.50    0.50
## top.freq       10.00  10.00   10.00
```

Diagnosing detection performance

The *ohun* package uses signal detection theory indices to evaluate detection performance. Signal detection theory deals with the process of recovering sounds (*i.e.*, target sounds) from background noise –not necessarily acoustic noise– and it is widely used for optimizing this decision-making process in the presence of uncertainty (Hossin & Sulaiman 2015). During a detection routine, the detected 'items' can be classified into four classes: true positives (TPs, target sounds correctly identified as signal), false positives (FPs, noise incorrectly identified as 'signal'), false negatives (FNs, sounds incorrectly identified as noise), and true negatives (TNs, background noise correctly identified as noise). However, TNs cannot always be easily defined in the context of sound event detection, as noise cannot always be partitioned into discrete units. Hence, the package makes use of TPs, FPs, and FNs to calculate three additional indices that can further assist with evaluating the performance of a detection routine and are widely used in sound event detection (Knight *et al.* 2017): recall (i.e., correct detections relative to total detections), precision (i.e., the proportion of target sounds that were correctly detected) and F1 score (combined recall and precision as the harmonic mean of these two, which provides a single value for evaluating performance, a.k.a. F-score, F-measure or Dice similarity coefficient).

A perfect detection will have no false positives or negatives, resulting in both recall and precision equal to 1. However, perfect detection cannot always be achieved. Therefore, some compromise between detecting most target sounds plus some noise and excluding noise but missing target sounds might be warranted. These indices provide a useful framework for diagnosing and optimizing the performance of a detection routine. Researchers can identify an appropriate balance between these two extremes by the relative costs of missing sounds and mistaking noise for target sounds in the context of their specific study goals.

ohun offers tools to evaluate the performance of sound event detection methods based on the indices described above. To accomplish this, annotations derived from a detection routine are compared against a reference table containing the time position of all target sounds in the sound files. For instance, the following code evaluates a routine run in Raven Pro 1.6 (Charif *et al.* 2010) using the "band limited energy detector" option (minimum frequency: 0.8 kHz; maximum frequency: 22 kHz; minimum duration: 0.03968 s; maximum duration: 0.54989s; minimum separation: 0.02268 s) on a subset of the zebra finch recordings described below:

```
raven_detec <- read.csv("combined_raven_detection.csv")

diagnose_detection(reference = manual_ref_tae, detection = raven_detec, by = "tuning_parameters")
```

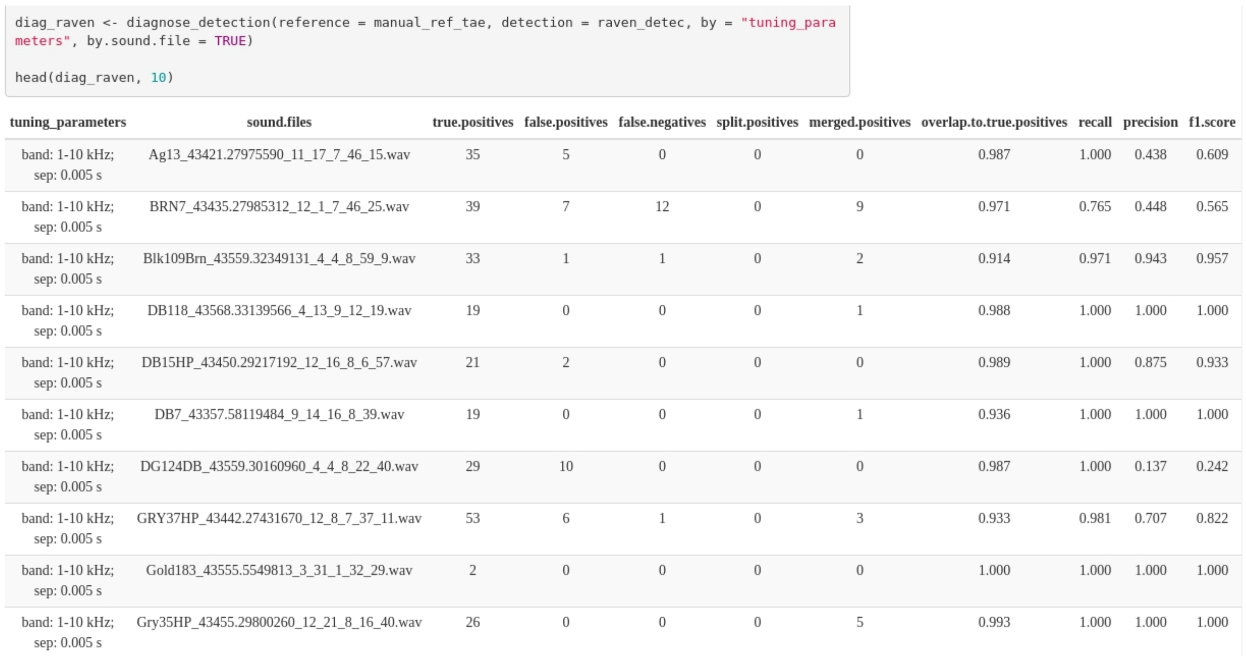
```
raven_detec <- read.csv("./data/processed/combined_raven_detection.csv")

diagnose_detection(reference = manual_ref_tae, detection = raven_detec, by = "tuning_parameters")
```

tuning_parameters	true.positives	false.positives	false.negatives	split.positives	merged.positives	overlap.to.true.positives	recall	precision	f1.score
band: 1-10 kHz; sep: 0.005 s	540	69	51	19	43	0.912	0.914	0.887	0.900
band: 1-10 kHz; sep: 0.02322 s	591	76	0	0	44	0.983	1.000	0.886	0.940
band: 1-15 kHz; sep: 0.02322 s	590	192	1	11	75	0.909	0.998	0.754	0.859
band: 1-22 kHz; sep: 0.02322.txt s	579	498	12	29	62	0.868	0.980	0.538	0.694

The function diagnose_detection uses bipartite matching graphs (Csardi & Nepusz 2006) to determine optimum matching of detected and reference events, ensuring that each reference sound is only be matched to a single detection (Lostanlen et al. 2019).

The output shows the indices described above, plus three additional metrics specific to sound event detection: 'split positives', 'merge positives', and 'overlap to true positives'. 'Split positives' is the number of reference sounds overlapped by more than one detection, 'merge positives' is the number of detected sounds overlapping with another detection, and 'overlap to true positives' quantifies the mean overlap between detections and reference signal (1 means complete overlap). The function also allows detailing those indices by sound file as well as by additional categorical features. Here we show the first ten files detailed by the column 'tunning_parameters' which contains the combained detection parameter values used in Raven:



```
diag_raven <- diagnose_detection(reference = manual_ref_tae, detection = raven_detec, by = "tuning_para
meters", by.sound.file = TRUE)

head(diag_raven, 10)
```

tuning_parameters	sound.files	true.positives	false.positives	false.negatives	split.positives	merged.positives	overlap.to.true.positives	recall	precision	f1.score
band: 1-10 kHz; sep: 0.005 s	Ag13_43421.27975590_11_17_7_46_15.wav	35	5	0	0	0	0.987	1.000	0.438	0.609
band: 1-10 kHz; sep: 0.005 s	BRN7_43435.27985312_12_1_7_46_25.wav	39	7	12	0	9	0.971	0.765	0.448	0.565
band: 1-10 kHz; sep: 0.005 s	Blk109Brn_43559.32349131_4_4_8_59_9.wav	33	1	1	0	2	0.914	0.971	0.943	0.957
band: 1-10 kHz; sep: 0.005 s	DB118_43568.33139566_4_13_9_12_19.wav	19	0	0	0	1	0.988	1.000	1.000	1.000
band: 1-10 kHz; sep: 0.005 s	DB15HP_43450.29217192_12_16_8_6_57.wav	21	2	0	0	0	0.989	1.000	0.875	0.933
band: 1-10 kHz; sep: 0.005 s	DB7_43357.58119484_9_14_16_8_39.wav	19	0	0	0	1	0.936	1.000	1.000	1.000
band: 1-10 kHz; sep: 0.005 s	DG124DB_43559.30160960_4_4_8_22_40.wav	29	10	0	0	0	0.987	1.000	0.137	0.242
band: 1-10 kHz; sep: 0.005 s	GRY37HP_43442.27431670_12_8_7_37_11.wav	53	6	1	0	3	0.933	0.981	0.707	0.822
band: 1-10 kHz; sep: 0.005 s	Gold183_43555.5549813_3_31_1_32_29.wav	2	0	0	0	0	1.000	1.000	1.000	1.000
band: 1-10 kHz; sep: 0.005 s	Gry35HP_43455.29800260_12_21_8_16_40.wav	26	0	0	0	5	0.993	1.000	1.000	1.000

Diagnostics from routines utilizing different tuning parameters can serve to identify the parameter values that optimize detection. This process of evaluating different routines for detection optimization is incorporated into the two signal detection approaches provided natively by *ohun*, which we depict in the following section. Note that the detection with Raven Pro does not necessarily reflect the best performance of this software and has been included only as an example of evaluating detection from external sources rather than a direct comparison of performance between Raven Pro and *ohun*.

Signal detection with *ohun*

The package offers two methods for automated signal detection: template-based and energy-based detection. These methods are better suited for stereotyped or good signal-to-noise ratio sounds, respectively. If the target sounds do not fit these requirements, more elaborate methods (i.e., machine/deep learning approaches) are warranted.

Study cases

Template detection on ultrasonic social calls of Spix's disc-winged bats

We recorded 30 individuals of Spix's disc-winged bats (*Thyroptera tricolor*) at Baru Biological Station in southwestern Costa Rica in January 2020. Bats were captured at their roosting sites (furled leaves of Zingiberaceae plants). Each bat was released in a large flight cage (9 x 4 x 3 m) for 5 minutes, and their ultrasonic inquiry calls were recorded using a condenser microphone (CM16, Avisoft Bioacoustics, Glienike/Nordbahn, Germany) through an Avisoft UltraSoundGate 116Hm plugged into a laptop computer running Avisoft-Recorder software. Recordings were made at a sampling rate of 500 kHz and an amplitude resolution of 16 bits.

Recordings were manually annotated using Raven Pro 1.6 (Charif *et al.* 2010). Annotations were created by visual inspection of spectrograms, in which the start and end of sounds were determined by the location of the continuous traces of power spectral entropy of the target sounds. A total of 644 calls were annotated (~21 calls per recording). Annotations were made with a time window of 200 samples and 70% overlap and were then imported into R using the package Rraven (Araya-Salas 2020).

Inquiry calls of Spix's disc-winged bats are structurally stereotyped (Chaverri *et al.* 2010). Most variation is found among individuals, although the basic form of a short, downward broadband frequency modulation is always shared (Fig. 1, Araya-Salas et al. 2021).

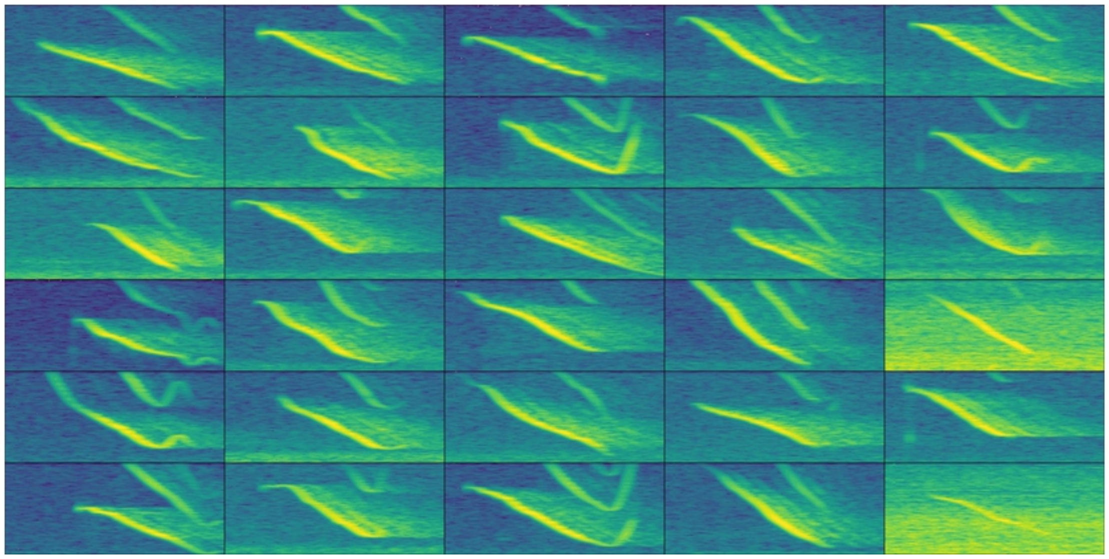


Figure 1. Example spectrograms of Spix's disc-winged social calls for each of the 30 recordings used in the analysis. The highest signal-to-noise ratio call by sound file are shown. The time scale range is 71 ms and the frequency range 10-44 kHz.

A template-based detection is a useful approach when there are minimal structural differences in the target sounds (Knight *et al.* 2017; e.g., when signals are produced in a highly stereotyped manner, Balantic & Donovan 2020). It uses spectrographic cross-correlation to find sounds resembling an example target sounds (i.e., template) across sound files, applying a correlation threshold to separate detections from background noise. We used this approach in *ohun* to detect inquiry calls. To do this, we tested the performance of three acoustic templates on a training subset of five sound files. First, we used the function get_templates to find several sounds representative of the variation in signal structure. This function measures several spectral features, which are summarized using Principal Component Analysis. The first two components are used to project the acoustic space. In this space, the function defines sub-spaces as equal-sized slices of a sphere centered at the centroid of the acoustic space. Templates are then selected as those closer to the centroid within sub-spaces, including the centroid for the entire acoustic space. The user needs to define the number of sub-spaces in which the acoustic space will be split.

```
# read manual annotations
manual_ref_thy <- read.csv("manual_annotations_thyroptera.csv")

# get random subset of 5 sound files for training
set.seed(1)
train_files <- sample(unique(manual_ref_thy$sound.files), size = 5)
train_ref <- manual_ref_thy[manual_ref_thy$sound.files %in% train_files, ]

# the rest for testin g
test_files <- setdiff(manual_ref_thy$sound.files, train_files)
test_ref <- manual_ref_thy[manual_ref_thy$sound.files %in% test_files, ]

# find templates
templates <- get_templates(train_ref, path = data_path,  bp = c(10, 50), ovlp = 70, wl = 200, n.sub.spa
ces = 3)
```

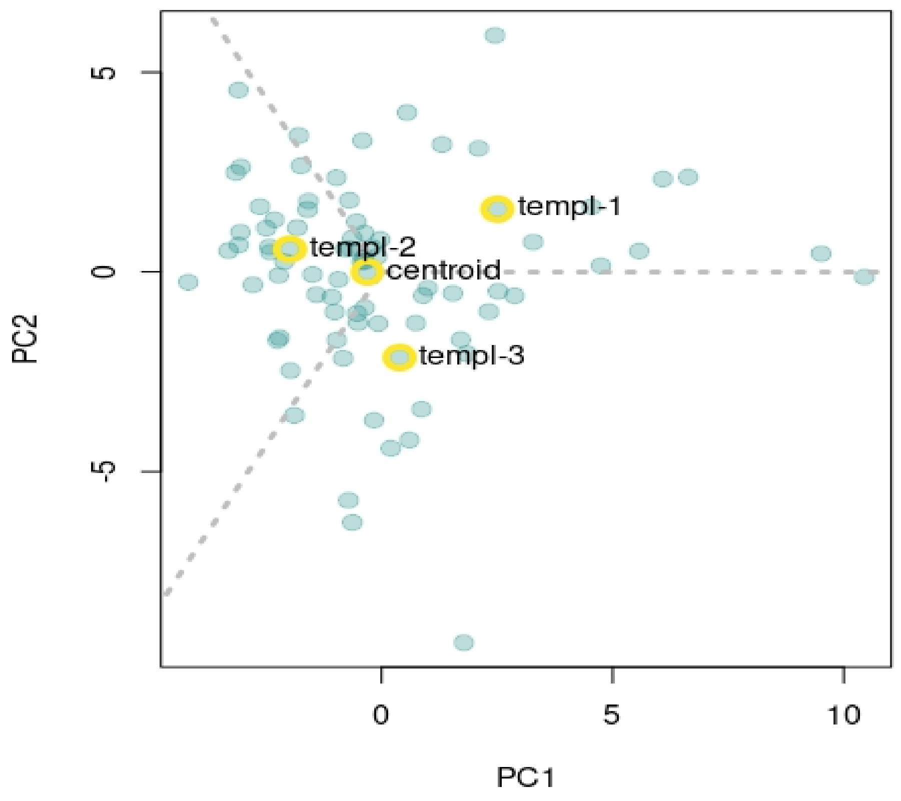


Figure 2. Acoustic space defined as the first two components of a Principal Component Analysis on spectrographic parameters. Templates are selected as those closer to the centroid within sub-spaces. Gray dashed lines delimit the region of sub-spaces. Yellow circles around points highlight the position of the signals selected as templates.

The output of the get_templates function includes an acoustic space plot (Fig. 2) in which the position of the sounds selected as templates is highlighted. Users can also provide their own acoustic space dimensions (argument 'acoustic.space'). In the following code, we used the templates determined above for detecting bat social calls. The code iterates a template-based detection on the training data set across a range of correlation thresholds for each template, in order to find the combination of threshold and template with the best performance.

```r
# get correlation vectors
corr_templ_train <- template_correlator(
  templates = templates,
    path = data_path,
  files = unique(train_ref$sound.files),
  hop.size = 10,
  ovlp = 70
  )

# evaluate detection for different correlation thresholds
opt_detec_train <- optimize_template_detector(
  reference = train_ref,
  template.correlations = corr_templ_train,
  threshold = seq(0.05, 0.5, 0.01)
  )
```

Note that the correlation vectors are estimated first (i.e., vectors of correlation values across sound files, template_correlator), and then the correlation thresholds are optimized on these vectors (optimize_template_detector). The output of optimize_template_detector contains the detection performance indices for each combination of templates and thresholds. Table 1 shows the two highest performance runs for each template.

Table 1. Performance diagnostic of template-based detections using four templates across several threshold values. Only the two highest performance iterations for each template are shown.

threshold	templates	true.positives	false.positives	false.negatives	recall	precision	f1.score
0.45	centroid	81	1	0	1.000	0.988	0.994
0.50	centroid	80	0	1	0.988	1.000	0.994
0.50	templ-1	77	1	4	0.951	0.987	0.969
0.45	templ-1	80	5	1	0.988	0.941	0.964
0.40	templ-2	79	0	2	0.975	1.000	0.988
0.35	templ-2	80	2	1	0.988	0.976	0.982
0.45	templ-3	77	2	4	0.951	0.975	0.963
0.40	templ-3	79	5	2	0.975	0.940	0.958

We can explore the performance of each template in more detail by looking at the change in F1 score across thresholds (Fig. 3).

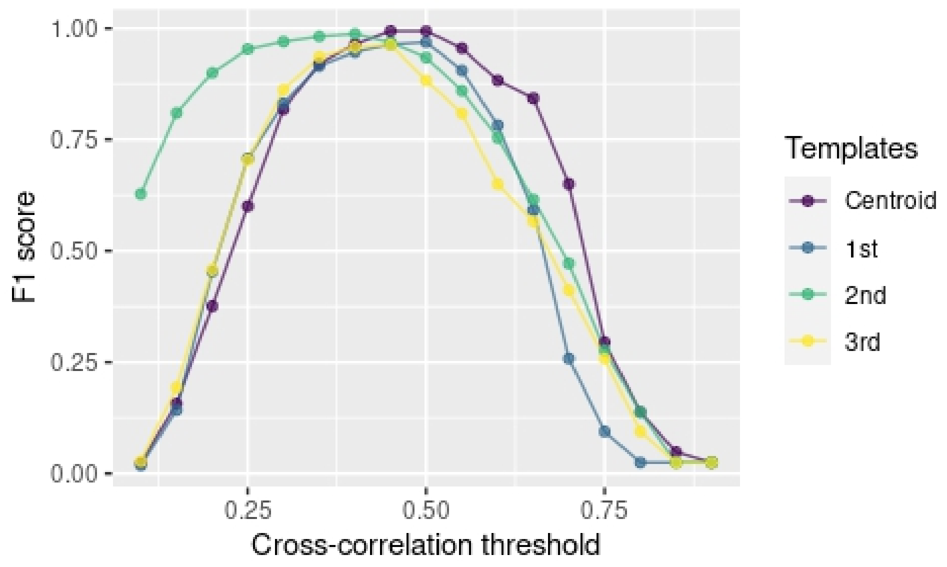


Figure 3. Changes in F1 score across the range of cross-correlation threshold values for four sound templates.

In this example, the "centroid" template produced the best performance (although not drastically different from other templates; Table 1; Fig. 3). Hence, we will use this template for detecting calls on the rest of the data. The following code extracts this template from the reference annotation table and uses it to find inquiry calls on the testing data set:

```
# get correlation vectors for test files
corr_templ_test <- template_correlator(
  templates = templates[templates$sound.file == "centroid", ],
    path = data_path, files = unique(test_ref$sound.files),
  hop.size = 10,
  ovlp = 70
  )

# detect on test files
detec_test <- template_detector(
  template.correlations = corr_templ_test,
    threshold = 0.45
  )

diagnose_detection(reference = test_ref, detection = detec_test)
```

true.positives	false.positives	false.negatives	split.positives	merged.positives	overlap.to.true.positives	recall	precision	f1.score
536	10	27	2	16	0.906	0.952	0.982	0.967

The last line of codes evaluates the detection on the test data set, which shows a good performance for both recall and precision (0.95 and 0.98 respectively).

Energy-based detection on zebra finch vocalizations

We used recordings from 18 zebra finch males recorded at the Rockefeller University Field Research Center Song Library (http://ofer.sci.ccny.cuny.edu/songs, Tchernichovski *et al.* 2021). Recordings contain undirected vocalizations (e.g., songs or calls) of single males recorded in sound attenuation chambers using Sound Analysis Pro. Zebra finch vocalizations are composed of multiple elements (i.e., distinct patterns of continuous traces of power spectral entropy in the spectrogram separated by time gaps) that can vary substantially in key features such as duration and frequency range (Fig. 4) and are not nearly as stereotyped as the Spix's disc-winged bats. However, as recorded sounds show a good signal-to-noise ratio, signals in each recording can potentially be detected using an energy-based approach that does not rely on matching the acoustic structure of a template.

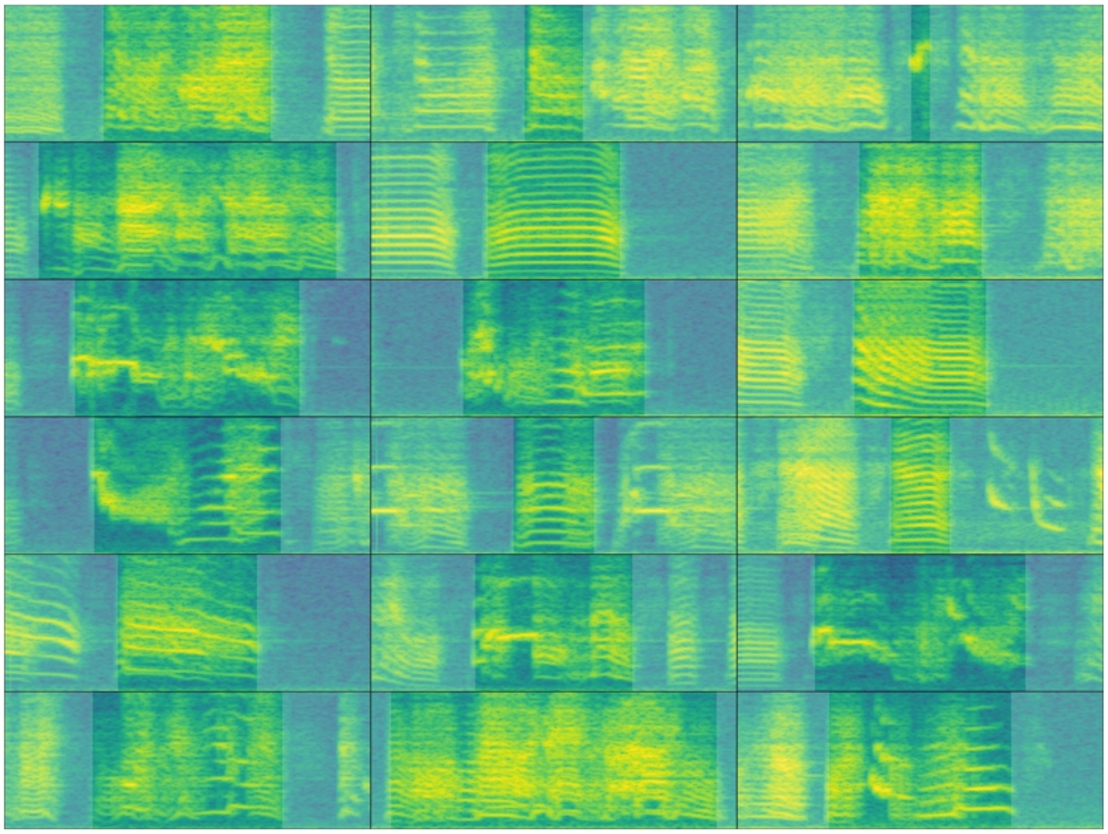


Figure 4. Example spectrograms of male zebra finch songs for each of the 18 sound files used in the analysis. The highest signal-to-noise ratio call by sound file are shown. The time scale range is 359 ms and the frequency range 0-11 kHz. Signals have been highlighted for visualization purposes only.

Reference annotations were made manually on the oscillogram with the spectrogram and audio as a guide using Raven Lite 2.0.1 (Cornell Lab of Ornithology). The following code loads the reference annotations and split them into two data sets for training (3 sound files) and testing (15 sound files):

```r
manual_ref_tae <- read.csv("manual_selections_Taeniopygia.csv")

set.seed(450)
train_files <- sample(unique(manual_ref_tae$sound.files), 3)
test_files <- setdiff(manual_ref_tae$sound.files, train_files)

train_ref <- manual_ref_tae[manual_ref_tae$sound.files %in% train_files, ]
test_ref <- manual_ref_tae[manual_ref_tae$sound.files %in% test_files, ]
```

The detection parameters can be optimized using the function optimize_energy_detector. This function runs a detection for all possible combinations of tuning parameters. The code below tries three minimum duration and maximum duration values and two hold time values (which merges sounds within the specified time interval):

```r
opt_det_train <- optimize_energy_detector(
  reference = train_ref,
  files = train_files,
  threshold = c(1, 5),
  hop.size = 11.6,
  smooth = c(5, 10),
  hold.time = c(0, 5),
  min.duration = c(5, 15, 25),
  max.duration = c(275, 300, 325),
  bp = c(0.5, 10)
)
```

The output (opt_det_train) shows the performance indices for each of those combinations. Here we show the ten combinations with the highest F1 score:

```r
# subset with highest performance
opt_det_train <- opt_det_train[order(opt_det_train$f1.score, decreasing = TRUE), ]

head(opt_det_train, 10)
```

threshold	smooth	hold.time	min.duration	max.duration	true.positives	false.positives	false.negatives	recall	precision	f1.score
1	5	5	25	300	105	9	0	1.000	0.921	0.959
1	5	5	25	325	105	9	0	1.000	0.921	0.959
1	5	0	25	300	105	12	0	1.000	0.897	0.946
1	5	0	25	325	105	12	0	1.000	0.897	0.946
1	10	0	25	300	105	13	0	1.000	0.890	0.942
1	10	0	25	325	105	13	0	1.000	0.890	0.942
1	5	5	15	300	105	15	0	1.000	0.875	0.933
1	5	5	15	325	105	15	0	1.000	0.875	0.933
5	5	5	15	275	96	8	9	0.914	0.923	0.919
5	5	5	15	300	96	8	9	0.914	0.923	0.919

We now can use the tuning parameter values that yielded the best performance to detect sounds on the test data set:

```
best_param <- opt_det_train[which.max(opt_det_train$f1.score), ]

det_test <- energy_detector(
  files = test_files,
  threshold = best_param$threshold,
  hop.size = 11.6,
  smooth = best_param$smooth,
  hold.time = best_param$hold.time,
  min.duration = best_param$min.duration,
  max.duration = best_param$max.duration,
  bp = c(0.5, 10)
)
```

As our reference annotations include all sounds in both the training and test annotations, we can evaluate the performance of the detection on the test set as well:

```
diagnose_detection(reference = test_ref, detection = det_test, by.sound.file = FALSE)
```

true.positives	false.positives	false.negatives	split.positives	merged.positives	overlap.to.true.positives	recall	precision	f1.score
475	27	11	4	16	0.97	0.977	0.946	0.962

The performance on the test data set was also acceptable, with an F1 score of 0.95. Note that in the example we used a small subset of sound files for training. More training data might be needed for optimizing a detection routine on larger data sets or recordings with more variable sounds or background noise levels.

Additional tools

The *ohun* package offers additional tools to simplify sound event detection. Detected sounds can be labeled as false or true positives with the function 'label_detection'. This allows users to explore the structure of false positives and figure out ways to exclude them. The function 'filter_detections' can remove ambiguous sounds (i.e., those labeled as split or merged detections), keeping only those that maximize a specific criterium (i.e., the highest template correlation). Finally, note that several templates representing the range of variation in signal structure can be used to detect semi-stereotyped sounds or sterepotyped multi-element repertoires when running template-based detection ('template_detection' function).

Discussion

Here we have shown how to evaluate the performance of sound event detection routines using the package *ohun*. The package can evaluate detection outputs imported from other software, as well as its own detection routines. The latter can be iterated over combinations of tuning parameters to find those values that optimize detection. Although signal detection indices are commonly reported when presenting new automatic detection methods, to our knowledge, in addtion to *ohun* there is only one other performance-evaluating software developed in a free, open-source platform (sed-eval, Mesaros 2016). These two software packages can provide a common framework for evaluating sound event detection can simplify comparing the performance of different tools and selecting those tools better suited to a given research question and study system. The tools offered by *ohun* for diagnosing detection performance should not necessarily be limited to acoustic data. *ohun* can also be used for cases in which the time of occurrence of discrete events needs to be identified, such as detecting specific behaviors in video analysis of animal motor activity (*e.g.*, Sturman *et al.* 2020; Hsu & Yttri 2021; Bohnslav *et al.* 2021). The detection of such motor events in video recordings can also be evaluated and optimized compared to a reference annotation, as we have shown here for sound events.

The *ohun* package provides two detection methods: template-based and energy-based detection. Compared to new deep learning approaches for finding the occurrence of sound events, the two native methods are relatively simple tools. However, these methods have been widely used by the bioacoustic community (Mellinger & Clark 2000; Charif *et al.* 2010; Aide *et al.* 2013; Specth 2002; Hafner & Katz 2015) and can reach adequate performance under the appropriate conditions, as evidenced by our two study cases and from previous reports (Knight *et al.* 2017). Deep learning methods tend to require greater computational power, larger training data sets, and, in some cases, more complex training routines (but see transfer learning approaches). This might bring unnecessary difficulties when dealing with less challenging detection tasks. Therefore, the availability of a wide range of approaches can simplify finding the most appropriate tool for the intricacies of a study system and research goals and having tools accessible to a broader research community. The tools offered in *ohun* can also be used in a subsequent pipeline in which detected sounds are further classified and false positives are mitigated using more elaborated discrimination algorithms (Balantic & Donovan 2020). Detection performance might be improved by using acoustic structure measurements to distinguish target from non-target sound events.

Detection routines can take a long time when working with large amounts of acoustic data (e.g. long recordings or many files). We provide some additional tips that can help make a routine more time efficient. 1. Always test procedures on small data subsets. Make sure to obtain decent results on a small subset of recordings before scaling up the analysis. 2. Template-based detection is almost always faster than energy-based detection. 3. Run routines in parallel. Parallelization (i.e., the ability to distribute tasks over several cores in your computer) can significantly speed up routines. All automatic detection and performance evaluation functions in *ohun* allow users to run analysis in parallel (see parallel argument in those functions). Hence, a computer with several cores can help improve efficiency. 4. Try using a computer with lots of RAM or a computer cluster for working on large amounts of data. 5. Sampling rate matters. Detecting sounds on low sampling rate files is faster, so we must avoid having Nyquist frequencies much higher than the highest frequency of the target sounds. Lastly, we underscore that these tips are not restricted to *ohun* and can also be helpful to speed up routines in other software packages.

Other things should be considered when aiming to detect sound events automatically. When running energy-based detection routines, try to use your knowledge of the signal structure to determine the initial range of tuning parameters. This can be extremely helpful for narrowing down possible parameter values. As a general rule, if human observers have difficulty detecting where a target sound occurs in a sound file, detection algorithms will likely yield low detection performance. In cases in which occurrences are ambiguous, low performances are expected. Ensure reference annotations contain all target sounds and only the target sounds. Otherwise, performance optimization can be misleading as the performance of a given detection method cannot be better than the reference itself. Lastly, avoid having overlapping sounds or several sounds as a single detection (e.g., a multi-syllable vocalization) in the reference annotation when running an energy-based detector, as they are likely to be identified as separated units.

Acknowledgments

Nazareth Rojas, Silvia Chaves-Ramírez, Mariela Sánchez-Chavarría, Miriam Gioiosa, Cristian Castillo-Salazar, and José Pablo Barrantes for their help collecting acoustic data for Spix's disc-winged bats. We also thank the Centro Biológico Hacienda Barú for their continuous support of our research and Mijail Rojas and Andrey Sequeira for support in the early stages of package development. This study was partly funded by a CONARE-Max Planck grant from the Consejo Nacional de Rectores and the research activity C0754 at Centro de Investigación en Neurociencias, Universidad de Costa Rica. A.R-G. is supported by the Walt Halperin Endowed Professorship and the Washington Research Foundation as Distinguished Investigator.

Ethical note

All sampling protocols followed guidelines approved by the American Society of Mammalogists for the capture, handling, and care of mammals (Sikes *et al.* 2016) and the ASAB/ABS Guidelines for the use of animals in research. This study was conducted in accordance with the ethical standards for animal welfare of the Costa Rican Ministry of Environment and Energy, Sistema Nacional de Áreas de Conservación, permit no. SINAC-ACOPAC-RES-INV-008-2017 (Decree No. 32553-MINAE). Protocols were also approved by the University of Costa Rica’s Institutional Animal Care and Use Committee (CICUA-42-2018).

Supporting information

Supplementary information associated with this article is available in 10.6084/m9.figshare.21675692

Author contribution

Marcelo Araya-Salas, Gloriana Chaverri, Alejandro Rico-Guevara, Juan C. Brenes and Fabiola Chirino and conceived the ideas and designed methods; Marcelo Araya-Salas, Gloriana Chaverri and Grace Smith-Vidaurre collected the data; Marcelo Araya-Salas, Grace Smith-Vidaurre, Juan C. Brenes, Fabiola Chirino and Jorge Elizondo-Calvo[2] analysed the data; Marcelo Araya-Salas, Alejandro Rico-Guevara led the writing of the manuscript. All authors contributed critically to the drafts and gave final approval for publication.

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, 2013.

Araya-Salas, M. (2020). Rraven: Connecting r and raven bioacoustic software. R package version 1.0.9.

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

Balantic, C.M. & Donovan, T.M. (2020). Statistical learning mitigation of false positives from template-detected data in automated acoustic wildlife monitoring. BIoacoustics, 29, 293–321.

Blumstein, D.T., Mennill, D.J., Clemins, P., Girod, L., Yao, K., Patricelli, G., Deppe, J.L., Krakauer, A.H., Clark, C., Cortopassi, K.A., Hanser, S.F., Mccowan, B., Ali, A.M. & Kirschel, A.N.G. (2011). Acoustic monitoring in terrestrial environments using microphone arrays: Applications, technological considerations and prospectus. Journal of Applied Ecology, 48, 758–767.

Bohnslav, J.P., Wimalasena, N.K., Clausing, K.J., Dai, Y.Y., Yarmolinsky, D.A., Cruz, T., Kashlan, A.D., Chiappe, M.E., Orefice, L.L., Woolf, C.J. & Harvey, C.D. (2021). DeepEthogram, a machine learning pipeline for supervised behavior classification from raw pixels. eLife, 10.

Burgdorf, J., Panksepp, J. & Moskal, J.R. (2011). Frequency-modulated 50kHz ultrasonic vocalizations: A tool for uncovering the molecular substrates of positive affect. Neuroscience and Biobehavioral Reviews, 35, 1831–1836.

Csardi G, Nepusz T (2006). The igraph software package for complex network research. InterJournal, Complex Systems, 1695.

Charif, R., Waack, A. & Strickman, L. (2010). Raven pro 1.4 user's manual. Cornell Lab of Ornithology.

Chaverri, G., Gillam, E.H. & Vonhof, M.J. (2010). Social calls used by a leaf-roosting bat to signal location. Biology Letters, 6, 441–444.

Gibb, R., Browning, E., Glover-Kapfer, P., Jones, K.E. & Jones, C.E.K. (2019). Emerging opportunities and challenges for passive acoustics in ecological assessment and monitoring. Wiley Online Library, 10, 169–185.

Gwee, C., Eaton, J., Garg, K. & Alström, P. (2019). Cryptic diversity in cyornis (aves: Muscicapidae) jungle-flycatchers flagged by simple bioacoustic approaches. Zoological Journal, 186.

Hafner, S. & Katz, J. (2015). monitoR: Acoustic template detection in r. R package version 1.0.3.

Hossin, M. & Sulaiman, M. (2015). A review on evaluation metrics for data classification evaluations. International Journal of Data Mining. 5.2 (2015): 1.

Hsu, A. & Yttri, E. (2021). B-SOiD, an open-source unsupervised algorithm for identification and fast prediction of behaviors. Nature Communications. 12.1 (2021): 1-13.

Knight, E.C., Hannah, K.C., Foley, G.J., Scott, C.D., Brigham, R.M. & Bayne, E. (2017). Recommendations for acoustic recognizer performance assessment with application to five common automated signal recognition programs. Avian Conservation and Ecology, 12.

Köhler, J., Jansen, M., Rodríguez, A., Kok, P.J.R., Felipe, T.L., Emmrich, M., Glaw, F., Haddad, C.F.B., Mark-Oliver, R., Vences, M., Toledo, L.F., Emmrich, M., Glaw, F., Haddad, C.F.B., Rödel, M.O. & Vences, M. (2017). The use of bioacoustics in anuran taxonomy: Theory, terminology, methods and recommendations for best practice. Zootaxa 4251.1 (2017): 1-124.

Lostanlen V, Salamon J, Farnsworth A, Kelling S, Bello JP (2019). Robust sound event detection in bioacoustic sensor networks. PLoS ONE 14(10): e0214168.

Medina-García, A., Araya-Salas, M. & Wright, T.F. (2015). Does vocal learning accelerate acoustic diversification? Evolution of contact calls in neotropical parrots. Journal of Evolutionary Biology, 28, 1782–1792.

Mellinger, D.K. & Clark, C.W. (2000). Recognizing transient low-frequency whale sounds by spectrogram correlation. The Journal of the Acoustical Society of America, 107, 3518–3529.

Mesaros A, T Heittola, & T. Virtanen. (2016). Metrics for polyphonic sound event detection. Applied Sciences, 6(6):162, 2016

Odom, K.J., Araya-Salas, M., Morano, J.L., Ligon, R.A., Leighton, G.M., Taff, C.C., Dalziell, A.H., Billings, A.C., Germain, R.R., Pardo, M., Andrade, L.G. de, Hedwig, D., Keen, S.C., Shiu, Y., Charif, R.A., Webster, M.S. & Rice, AN (2021). Comparative bioacoustics: A roadmap for quantifying and comparing animal sounds across diverse taxa. Biological Reviews, 96, 1135–1159.

Schöneich, S. (2020). Neuroethology of acoustic communication in field crickets-from signal generation to song recognition in an insect brain. Progress in Neurobiology 194: 101882.

Sikes, R.S., Animal Care, the & American Society of Mammalogists, U.C. of the. (2016). 2016 guidelines of the american society of mammalogists for the use of wild mammals in research and education. Journal of Mammalogy, 97, 663–688.

Specht, R. (2002). Avisoft-saslab pro: sound analysis and synthesis laboratory. Avisoft Bioacoustics. 1-723.

Stowell, D. (2022). Computational bioacoustics with deep learning: A review and roadmap. PeerJ, 10, e13152.

Sturman, O., Ziegler, L. von, Schläppi, C. & Akyol, F. (2020). Deep learning-based behavioral analysis reaches human accuracy and is capable of outperforming commercial solutions. Neuropsychopharmacology. 45.11 (2020): 1942-1952.

Sugai, L., Silva, T., Jr, J.R. & Llusia, D. (2019). Terrestrial passive acoustic monitoring: Review and perspectives. BioScience 69.1 (2019): 15-25.

Tchernichovski, O., Eisenberg-Edidin, S. & Jarvis, E.D. (2021). Balanced imitation sustains song culture in zebra finches. Nature Communications 2021 12:1, 12, 1–14.

Tiwari, C. & Diwakar, S. (2022). The katydid country: Bioacoustics and ecology of tettigoniid communities from the indian subcontinent. Bioacoustics (2022): 1-25.

Zsebők, S., Schmera, D., Laczi, M., Nagy, G., Vaskuti, É., Török, J., & Zsolt Garamszegi, L. (2021). A practical approach to measuring the acoustic diversity by community ecology methods. Methods in Ecology and Evolution, 12(5), 874-884.

rOpenSci review status

The following changes were requested:

- Including non-standard requirements into the DESCRIPTION file
- replacing the function sapply() by vapply() when possible
- use seq_len() instead of 1:length() or similar when possible

The changes were already made and the package is wating for approval.
The current review status can be checked here:

https://github.com/ropensci/software-review/issues/568