Received: 13 January 2023 | Accepted: 7 June 2023

DOI: 10.1111/2041-210X.14170

Methods in Ecology and Evolution

APPLICATION

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, New York, USA; [5]Rockefeller University Field Center, Millbrook, New York, USA; [6]Department of Biological Sciences, University of Cincinnati, Cincinnati, Ohio, USA; [7]Smithsonian Tropical Research Institute, Panama City, Panama; [8]Department of Biology, University of Washington, Seattle, Washington, USA and [9]Burke Museum of Natural History and Culture, University of Washington, Seattle, Washington, USA

Correspondence
Marcelo Araya-Salas
Email: marcelo.araya@ucr.ac.cr

Funding information
CONARE-Max Planck, Grant/Award Number: C0754; Universidad de Costa Rica; Walt Halperin Endowed Professorship; Washington Research Foundation as Distinguished Investigator

Handling Editor: Thomas White

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 analyse promptly. Although many tools are available for the automated detection of sounds, choosing the right approach can be difficult and only a few tools provide a framework for evaluating detection performance.
2. Here, we present *ohun*, an R package intended to facilitate automated sound event detection. *ohun* provides functions to diagnose and optimize detection routines, compare performance among different detection approaches and evaluate the accuracy in inferring the temporal location of events.
3. The package uses reference annotations containing the time position of target sounds in a training data set to evaluate detection routine 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. In addition, *ohun* includes energy-based and template-based detection methods, two commonly used automatic approaches in bioacoustics research.
4. We show how *ohun* can be used to automatically 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.

KEYWORDS
animal vocalizations, bioacoustics, sound event detection

1 | 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 (Gwee et al., 2019; Köhler et al., 2017), community ecology (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 bioacoustics approaches may also imply large amounts of data (i.e lots of recordings), which can be challenging to analyse 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 (Balantic & Donovan, 2020; Knight et al., 2017). 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).

The fact that sound events are not always structured as single acoustic units and that they are embedded within a continuous string of sound in some cases creates the need for 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 challenges 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 bioacoustics analysis: energy-based detection and template-based detection (Aide et al., 2013; Charif et al., 2010; Hafner & Katz, 2015; Mellinger & Clark, 2000). 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). See the package vignette (https://marce10.github.io/ohun/articles/ohun.html), the package website (https://marce10.github.io/ohun) and the R package documentation in CRAN (https://CRAN.R-project.org/package=ohun) for additional details and examples.

2 | FORMATTING ACOUSTIC DATA SETS

The format and size of the acoustic data to be analysed needs to be standardized to avoid downstream errors and to inform expectations for computational time performance. Several functions in *ohun* can facilitate double-checking the format of acoustic datasets 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):

```
# path to files directory
path_zebra_finch <- "path_to_zebra_finch_files"
feature_acoustic_data(path = path_zebra_finch)
## Features of the acoustic data set in './data/raw/taeniopygia':
## * 72 sound files
## * 1 file format(s) (.wav (72))
## * 1 sampling rate(s) (44.1 kHz (72))
## * 1 bit depth(s) (16 bits (72))
```

```
## * 1 number of channels (1 channel(s) (72))
## * File duration range: 0.2-21.76 s (mean: 6.07 s)
## * File size range: 0.02-1.92 MB (mean: 0.54 MB)
## (detailed information by sound file can be obtained with 'warbleR::info_sound_files()')z1
```

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 analyse 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, # path to recordings
 sgmt.dur = 5) # duration of clips
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, # data frame with reference annotations
 path = path_zebra_finch # path to recordings
 )
```

##	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	43.28	65.19	88.69
## bottom.freq	0.50	0.50	0.50
## top.freq	10.00	10.00	10.00

3 | DIAGNOSING DETECTION PERFORMANCE

The *ohun* package uses signal detection theory indices to evaluate detection performance. Signal detection theory deals with the process of recovering signals (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 events can be classified into four classes: true positives (TPs, detections that overlap with reference events), false positives (FPs, detections that do not overlap with reference events) and false negatives (FNs, reference events that do not overlap with any detection). True negavites cannot 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 the proportion of target sounds that were correctly detected), precision (i.e proportion of correct detections relative to total detections) and F score (combined recall and precision as the harmonic mean of these two, which provides a single value for evaluating performance, a.k.a. *F*1 score, *F*-measure or Dice similarity coefficient). Note that in *ohun* overlap is measured as 50% temporal intersection over union but it be modified by users.

The package also offers three additional metrics related to the accuracy of the time location of a sound event detection: 'splits, 'merges' and 'overlap'. 'Splits' refers to the number of redundant detections, i.e. those detections overlapping reference sounds that also overlap with other detections. 'Merges' is the number of detections that overlap with more than one reference sound, and 'overlap' quantifies the

mean overlap between detections and reference sounds (1 means complete overlap). These three indices are relevant for instances in which the temporal location of events has to be accurately determined. 'Splits' and 'merges' also enable users to infer whether target signals are being detected as discrete units. This can be particularly helpful for recordings with a high duty cycle. A perfect routine should present no split or merged detections and an overlap equals to 1.

```
head(raven_detec)
diag_raven <- diagnose_detection(
 reference = manual_ref_tae, # data frame with annotations
 detection = raven_detec, # detection data frame to be diagnosed
 by = "tuning_parameters" # categorical column name indicating
detections from same run
 )
```

tuning_parameters	detections	true.positives	false.positives	false.negatives	splits	merges	overlap	recall	precision
band: 1–10 kHz; sep: 0.005 s	574	448	126	143	0	0	0.832	0.758	0.780
band: 1–10 kHz; sep: 0.02322 s	155	30	125	561	0	0	0.743	0.051	0.194
band: 1–15 kHz; sep: 0.02322 s	632	375	257	216	0	0	0.833	0.635	0.593
band: 1–22 kHz; sep: 0.02322.txt s	1020	423	597	168	0	0	0.833	0.716	0.415

A perfect detection should also lack any 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 annotation 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: 0.54989 s; minimum separation: 0.02268 s; values obtained through manual optimization) on a subset of the zebra finch recordings described below (example data included in the Supporting Information):

```
# reading data
raven_detec <- read.csv("combined_raven_detection.csv")
# checking data structure
```

The diagnose_detection function make use of the maximum bipartite matching algorithm (Csardi & Nepusz, 2006) in conjunction with the push-relabel algorithm (Goldberg & Tarjan, 1988) to optimize the assignment of detections to target sounds. This algorithm aims to maximize the matching between detected and reference events, ensuring that each reference sound is exclusively associated with a single detection (Lostanlen et al., 2019). Our implementation weights the matching process with the amount of overlap. Thus, detections with higher overlap to reference events are given higher priority. The 'by' argument in the provided code enables users to specify a column indicating which detections (i.e rows) belong to the same run.

By default the function computes indices across all sound files in the data set. However, the function also allows detailing those indices separately for each sound file (argument 'by.sound.file') The following code shows the first ten files detailed by the column 'tuning_parameters', which contains the combined detection parameter values used in Raven:

```
diag_raven <- diagnose_detection(
    reference = manual_ref_tae, # data frame with reference
    annotations
    detection = raven_detec, # detection data frame to be
    diagnosed
    by = "tuning_parameters", # categorical column name indicating
    detections from same run
```

sound.files	selec	start	end	bottom.freq	top.freq	tuning.parameters
Ag13_43421.27975590_11_17_7_46_15.wav	1	0.3715	0.423744897	0.8	10	band: 1–10 kHz; sep: 0.005 s
Ag13_43421.27975590_11_17_7_46_15.wav	2	1.0913	1.17837483	0.8	10	band: 1–10 kHz; sep: 0.005 s
Ag13_43421.27975590_11_17_7_46_15.wav	3	3.2218	3.26243492	0.8	10	band: 1–10 kHz; sep: 0.005 s
Ag13_43421.27975590_11_17_7_46_15.wav	4	3.6107	3.668749887	0.8	10	band: 1–10 kHz; sep: 0.005 s
Ag13_43421.27975590_11_17_7_46_15.wav	5	5.1084	5.172254875	0.8	10	band: 1–10 kHz; sep: 0.005 s
Ag13_43421.27975590_11_17_7_46_15.wav	6	6.5596	6.681504762	0.8	10	band: 1–10 kHz; sep: 0.005 s

```
    by.sound.file = TRUE # for detailing indices by sound file
    )
head(diag_raven, 10)
```

uning_parameters	sound.files	detections	true. positives	false. positives	false. negatives	splits	merges	overlap	recall	precision
band: 1–10 kHz; sep: 0.005 s	Ag13_43421.27975590_11_17_7_46_15.wav	40	35	5	0	0	0	0.856	1.000	0.875
band: 1–10 kHz; sep: 0.005 s	BRN7_43435.27985312_12_1_7_46_25.wav	34	21	13	30	0	0	0.747	0.412	0.618
band: 1–10 kHz; sep: 0.005 s	Blk109Brn_43559.32349131_4_4_8_59_9.wav	30	27	3	7	0	0	0.852	0.794	0.900
band: 1–10 kHz; sep: 0.005 s	DB118_43568.33139566_4_13_9_12_19.wav	18	18	0	1	0	0	0.808	0.947	1.000
band: 1–10 kHz; sep: 0.005 s	DB15HP_43450.29217192_12_16_8_6_57.wav	23	21	2	0	0	0	0.902	1.000	0.913
band: 1–10 kHz; sep: 0.005 s	DB7_43357.58119484_9_14_16_8_39.wav	18	18	0	1	0	0	0.882	0.947	1.000
band: 1–10 kHz; sep: 0.005 s	DG124DB_43559.30160960_4_4_8_22_40.wav	39	29	10	0	0	0	0.815	1.000	0.744
band: 1–10 kHz; sep: 0.005 s	GRY37HP_43442.27431670_12_8_7_37_11.wav	54	45	9	9	0	0	0.875	0.833	0.833
band: 1–10 kHz; sep: 0.005 s	Gold183_43555.5549813_3_31_1_32_29.wav	2	2	0	0	0	0	0.827	1.000	1.000
band: 1–10 kHz; sep: 0.005 s	Gry35HP_43455.29800260_12_21_8_16_40.wav	18	15	3	11	0	0	0.819	0.577	0.833

Diagnostics from routines utilizing different tuning parameters 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*.

4 | 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 (see Stowell, 2022 for a detailed review of available methods).

5 | STUDY CASES

5.1 | 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 × 4 × 3 m) for 5 min, 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 (a time window of 200 samples and 70% overlap), 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) 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 (Figure 1, Araya-Salas et al., 2020).

Template-based detection uses spectrographic cross-correlation to find sounds resembling an example target sounds (i.e template) across sound files. The method produces vectors of correlation values through time, in which a correlation threshold can be applied to separate detections from background noise. It is a useful approach when there are minimal structural differences in the target sounds (e.g. when signals are produced in a highly stereotyped manner: Balantic & Donovan, 2020; Knight et al., 2017). 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 centred 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 (argument 'n.sub.spaces'):

```
# path to files directory
path_bats <- "path_to_bat__files"
```

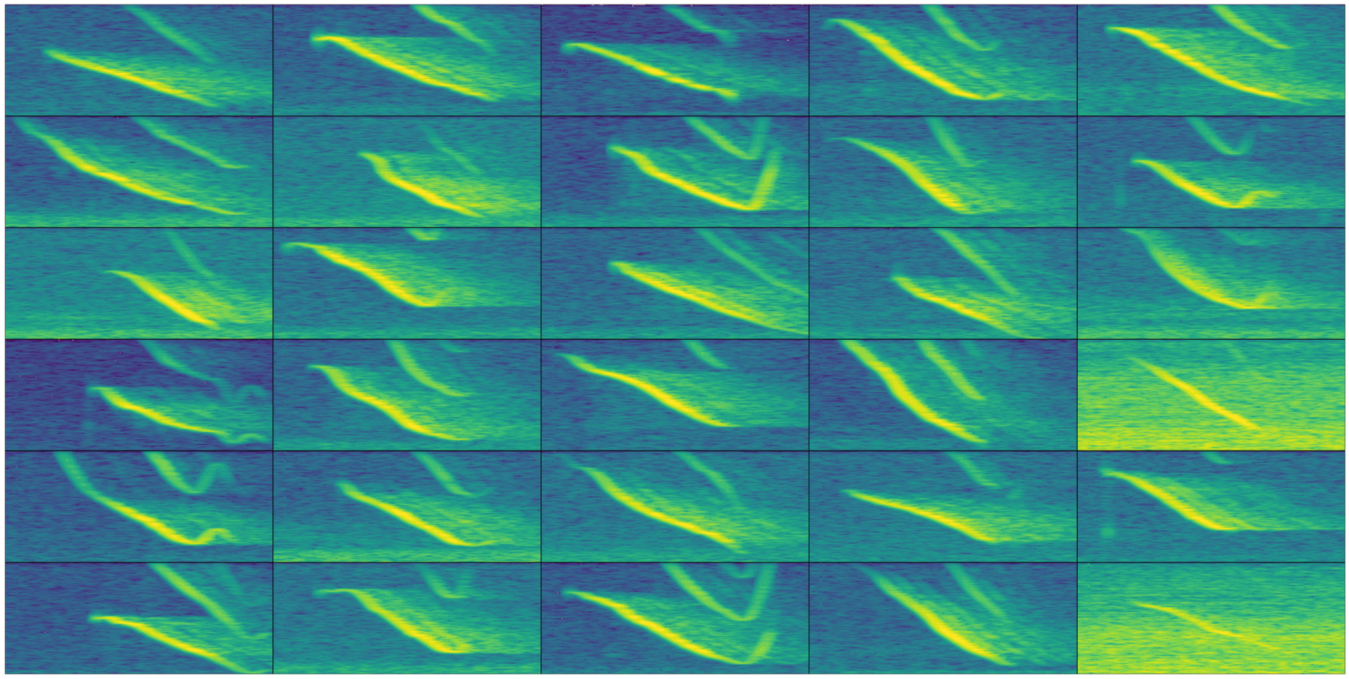

FIGURE 1 Example spectrograms of Spix's disc-winged bats social calls for each of the 30 recordings used in the analysis. The highest signal-to-noise ratio call by sound file are shown. The time scale range is 71 ms and the frequency range 10–44 kHz.

```
# read manual annotations
manual_ref_thy <- read.csv(file.path(path_bats, "manual_annotations_thyroptera.csv"))
# get random subset of 5 sound files for training

set.seed(1) # use seed to allow replication
train_files <- sample(unique(manual_ref_thy$sound.files), size = 5)
train_ref <- manual_ref_thy[manual_ref_thy$sound.files %in% train_files, ]

# use the rest of the data for testing
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(
    reference = train_ref, # data frame with reference annotations
    path = path_bats, # sound file directory
    bp = c(10, 50), # bandpass filter (kHz)
    ovlp = 70, # overlap between spectrogram time windows (%)
    hop.size = 10, # size of the spectrogram time window (in ms)
    n.sub.spaces = 3 # number of sub-spaces
    )
```

This method might perform better on acoustic spaces in which sounds are homogeneously distributed, as templates might represent similar portions of the overall population of sounds in the data. However, it can still be useful for identifying structurally diverse templates in irregularly distributed acoustic spaces. The output of the get_templates function includes an acoustic space plot (Figure 2)

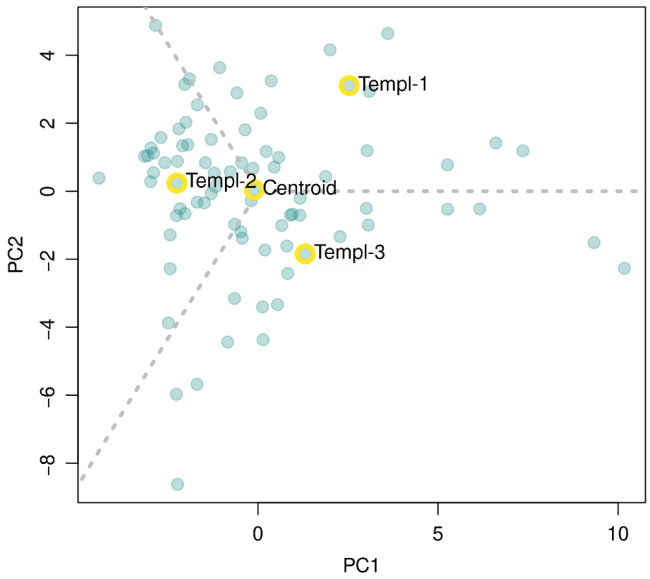

FIGURE 2 Acoustic space defined as the first two components of a Principal Component Analysis on spectrographic parameters. Templates are selected as those closer to the centroid within sub-spaces. Grey dashed lines delimit the region of sub-spaces. Yellow circles around points highlight the position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

in which the position of the sounds selected as templates is highlighted. Users can also provide their own acoustic space dimensions (argument 'acoustic.space'), which allows users to customize the acoustic space by specifying the features used for projecting it. 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, # data frame with annotations to be used as templates
    path = path_bats, # sound file directory
    files = unique(train_ref$sound.files), # set of files in which to run correlation
    hop.size = 10, # size of the spectrogram time window (ms)
    ovlp = 70 # overlap between spectrogram time windows (%)
    )
# evaluate detection for different correlation thresholdsopt_detec_train <- optimize_template_detector(
    reference = train_ref, # data frame with reference annotations template.correlations = corr_templ_train, # output from template_correlator()
    threshold = seq(0.05, 0.5, 0.01) # correlation treshold values to evaluate
    )
```

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 (identified as the highest F score) for each template.

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

In this example, the "centroid" template produced the best performance (although not drastically different from other templates; Table 1; Figure 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$template == "centroid", ], # template annotation
    path = path_bats, # sound file directory
    files = unique(test_ref$sound.files), # set of files in which to run correlation
    hop.size = 10, # size of the spectrogram time window (ms)
    ovlp = 70 # overlap between spectrogram time windows (%)
    )
# detect on test filesdetec_test <- template_detector(
    template.correlations = corr_templ_test, # output from template_correlator()
    threshold = 0.45 # correlation threshold
    )
diagnose_detection(
    reference = test_ref, # data frame with reference annotations
    detection = detec_test # detection data frame to be diagnosed
    )
```

detections	true.positives	false.positives	false.negatives	splits	merges	overlap	recall	precision	f.score
532	522	10	41	0	14	0.835	0.927	0.981	0.953

The last line of code evaluates the detection on the test data set, which shows a good performance for both recall and precision (0.93 and 0.98 respectively). An alternative to this approach would be to run detections using all templates and then generate a consensus table. This last step can be done using the function consensus_detection, which combines detections from several templates and, when several templates match the same reference sound, only the template with the highest correlation score is kept. This method can improve detection performance, but note that it will also increase computational time.

TABLE 1 Performance diagnostics 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	f.score
0.45	centroid	75	1	6	0.926	0.987	0.955
0.50	centroid	74	0	7	0.914	1.000	0.955
0.50	templ-1	71	1	10	0.877	0.986	0.928
0.45	templ-1	74	5	7	0.914	0.937	0.925
0.40	templ-2	73	0	8	0.901	1.000	0.948
0.35	templ-2	74	2	7	0.914	0.974	0.943
0.45	templ-3	66	7	15	0.815	0.904	0.857
0.40	templ-3	68	10	13	0.840	0.872	0.855

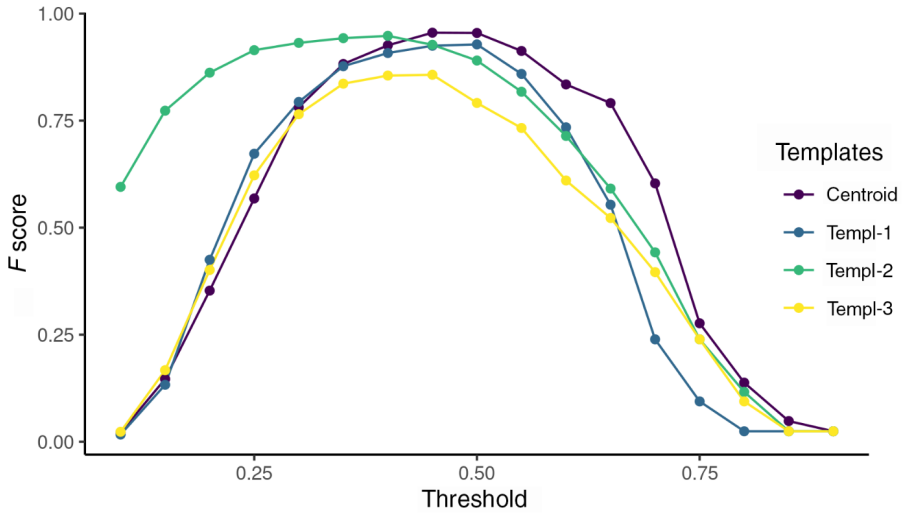

FIGURE 3 Changes in *F* score across the range of cross-correlation threshold values for four sound templates.

5.2 | Energy-based detection on zebra finch vocalizations

This method applies a threshold to amplitude envelopes to infer the temporal position of sound events. 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 (Figure 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.

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):

```
set.seed(450) # use seed to allow replication
train_files <- sample(unique(manual_ref_tae$sound.files), 3) # get subsample of files for training
test_files <- setdiff(manual_ref_tae$sound.files, train_files) # keep the rest of files for testing

# subset data
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. Several values for each tuning parameter can be evaluated in a single run. The following code tries three minimum duration and maximum duration values and two hold time values:

```
opt_det_train <- optimize_energy_detector(
  reference = train_ref, # annotation data frame
  files = train_files, # sound files on which to optimize detection
  threshold = c(1, 5), # amplitude threshold (in %)
  hop.size = 11.6, # size of the spectrogram time window (ms)
  smooth = c(5, 10), # size (in ms) of the sliding window use for smoothing
  hold.time = c(0, 5), # time range in which to merge detections into a single one (ms)
  min.duration = c(5, 15, 25), # minimum duration of detections to keep (ms)
  max.duration = c(275, 300, 325), # maximum duration of detections to keep (ms)
  bp = c(0.5, 10), # bandpass filter (kHz),
  path = path_zebra_finch
  )
```

The output (opt_det_train) shows the performance indices for each of those combinations. Here, we show the 10 combinations with the highest F score:

```
# subset with highest performance
opt_det_train <- opt_det_train[order(opt_det_train$f.score, decreasing = TRUE), ]
head(opt_det_train, 10)
```

threshold	smooth	hold.time	min.duration	max.duration	true.positives	false.positives	false.negatives	recall	precision	f.score
1	5	0	25	300	89	19	16	0.848	0.824	0.836
1	5	0	25	325	89	19	16	0.848	0.824	0.836
1	5	5	25	300	86	16	19	0.819	0.843	0.831
1	5	5	25	325	86	16	19	0.819	0.843	0.831
1	5	5	15	300	86	22	19	0.819	0.796	0.808
1	5	5	15	325	86	22	19	0.819	0.796	0.808
1	5	0	15	300	89	29	16	0.848	0.754	0.798
1	5	0	15	325	89	29	16	0.848	0.754	0.798
1	10	0	25	300	84	22	21	0.800	0.792	0.796
1	10	0	25	325	84	22	21	0.800	0.792	0.796

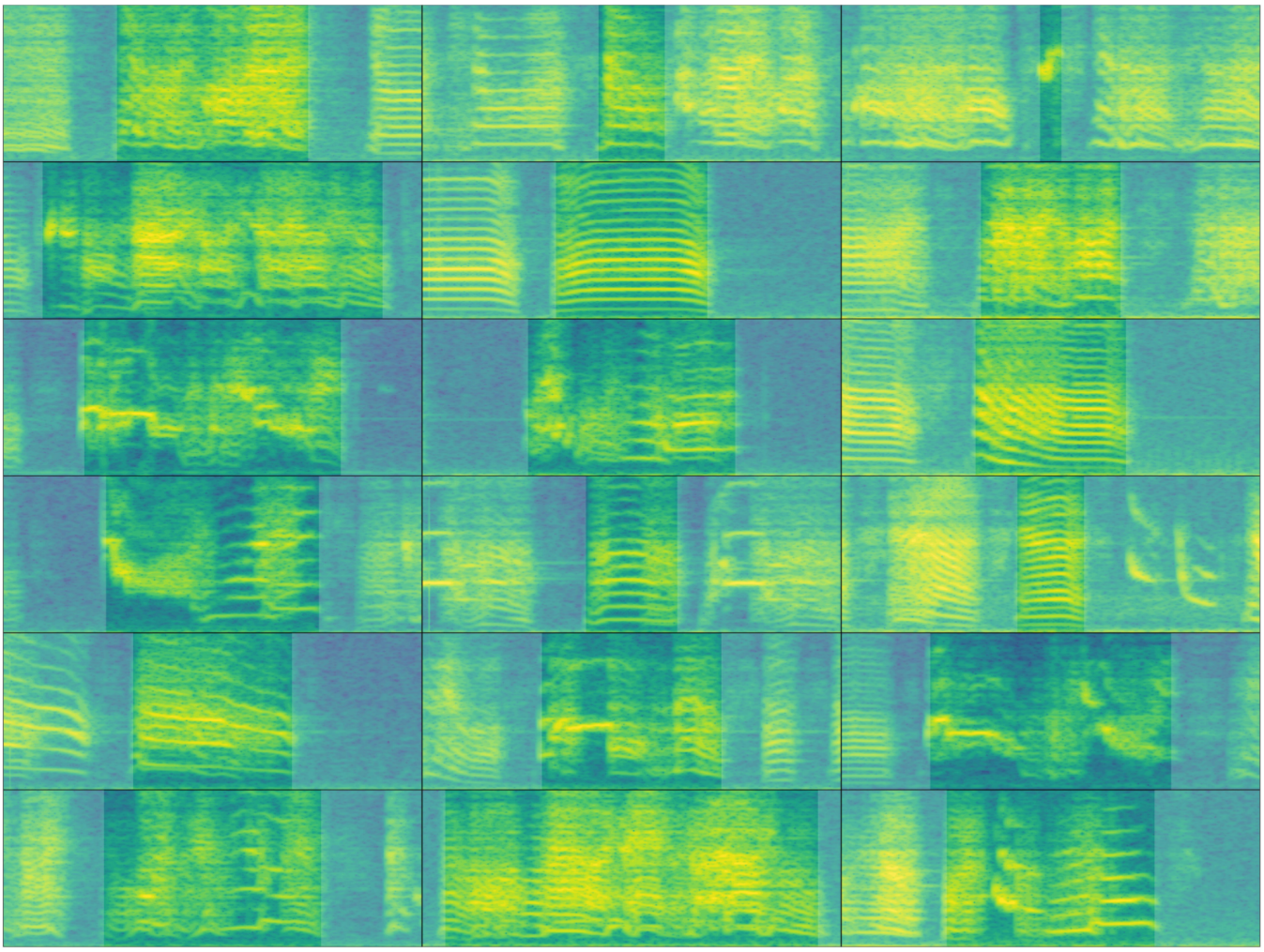

FIGURE 4 Example spectrograms of male zebra finch songs for each of the 18 sound files used in the analysis. The highest signal-to-noise ratio call by sound file are shown. The time scale range is 359 ms and the frequency range 0–11 kHz. Signals have been highlighted for visualization purposes only.

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

```
# keep only the highest f.score
best_param <- opt_det_train[which.max(opt_det_train$f.score), ]

det_test <- energy_detector(
  files = test_files, # set of files in which to run detection
  threshold = best_param$threshold, # threshold from best detection
  hop.size = 11.6, # size of the spectrogram time window (ms)
```

```
  smooth = best_param$smooth, # size (in ms) of the sliding
  window use for smoothing
  hold.time = best_param$hold.time, # size of the spectrogram
  time window (ms)
  min.duration = best_param$min.duration, # minimum duration
  of detections to keep (ms)
  max.duration = best_param$max.duration, # maximum duration
  of detections to keep (ms)
  bp = c(0.5, 10), # bandpass filter (kHz)
  path = path_zebra_finch
  )
```

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, # data frame with reference annotations
  detection = det_test, # detection data frame to be diagnosed
  by.sound.file = FALSE # summarize across sound files
  )
```

detections	true.positives	false.positives	false.negatives	splits	merges	overlap	recall	precision	f.score
505	452	53	34	0	0	0.924	0.93	0.895	0.912

The performance on the test data set was also acceptable, with an F score of 0.91. 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 measures might be needed when working with unbalanced datasets. For instance, the function can use macro-averaging for summarizing performance indices while giving equal weight to each sound file (Mesaros et al., 2016):

```
diagnose_detection(
  reference = test_ref, # data frame with reference annotations
  detection = det_test, # detection data frame to be diagnosed
  by.sound.file = FALSE, # summarize across sound files,
  macro.average = TRUE # calculates average of within sound file
  averages
  )
```

Detections	True.Positives	False.Positives	False.Negatives	Splits	Merges	Overlap	Recall	Precision	f.score
505	452	53	34	0	0	0.922	0.923	0.894	0.908

Stratified sampling that accounts for additional structure in the data (e.g. several individuals, populations, days) might also help to deal with unbalanced data and ensure similar performance on unseen data (see *createDataPartition* in the R package *caret* for creating stratified training samples).

5.3 | Additional tools and tips

The *ohun* package offers additional tools to simplify sound event detection. Detected sounds can be labelled 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 '*consensus_detection*' can remove ambiguous sounds (i.e those labelled as split or merged detections), keeping only those that maximize a specific criterion (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 stereotyped multi-element repertoires when running template-based detection ('template_detection' function).

Detection routines can take a long time when working with large amounts of acoustic data (e.g. long recordings or many files). These are some practices that can help make a sound event detection routines 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) 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. (3) Try using a computer with lots of RAM or a computer cluster for working on large amounts of data. (4) 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. 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. Lastly, ensure that the 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.

6 | 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, there is only one other performance-evaluating software developed in a free, open-source platform (sed_eval, Mesaros et al., 2016). These two software packages can provide a common framework for evaluating sound event detection that 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 behaviours in video analysis of animal motor activity (e.g. Bohnslav et al., 2021; Hsu & Yttri, 2021; Sturman et al., 2020). 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 bioacoustics community (Aide et al., 2013; Charif et al., 2010; Hafner & Katz, 2015; Mellinger & Clark, 2000; Specht, 2002) 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 (Mesaros et al., 2021), and, in some cases, more complex training routines (e.g. data augmentation, 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 as well as making 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.

The implementation of detection diagnostics that can be applied to both built in detection methods and to those obtained from other software packages makes the package ohun an useful tool for conducting direct comparisons of the performance of different routines. This feature enables users to precisely identify the detection approaches that better align with their specific needs. The package also offers a range of complementary functions that allow users to inspect and format acoustic data sets, and extract structural features of sound events from training data sets in order to inform tuning parameter values for automatic detection routines. Furthermore, ohun introduces new performance indices that are focused on the accuracy of temporal location in detected sound events. These indices can prove particularly useful in studies where further measurements need to be taken from the detected events. As sound event detection techniques continue to advance, these new indices can be instrumental for evaluating an additional performance dimension, temporal accuracy, which has remained relatively unexplored. Finally, the compatibility of 'ohun' with data formats already used by other sound analysis R packages (e.g. seewave, warbleR) make possible the integration of 'ohun' into more complex acoustic analysis workflows in a popular programming environment within the research community. We expect these contributions to make automatic sound event detection more accessible to the wider audience in the scientific community, facilitating the implementation of automated detection routines in bioacoustics research.

AUTHOR CONTRIBUTIONS

Marcelo Araya-Salas, 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, Fabiola Chirino and Jorge Elizondo-Calvo 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.

ACKNOWLEDGEMENTS

We thank 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. M.A.-S. would like to thank the Programa de Posdoctorado at the University of Costa Rica for funding. A.R-G. is supported by the Walt Halperin Endowed Professorship and the Washington Research Foundation as Distinguished Investigator.

CONFLICT OF INTEREST STATEMENT

The authors declare that there is no conflict of interest that could be perceived as prejudicing the impartiality of the research reported.

PEER REVIEW

The peer review history for this article is available at https://www.webofscience.com/api/gateway/wos/peer-review/10.1111/2041-210X.14170.

DATA AVAILABILITY STATEMENT

The ohun package is available on CRAN (https://cran.r-project.org/package=ohun). The development version of the package used in the application and the source code can be found at https://github.com/maRce10/ohun. The scripts and data for running the example code in the paper are available at https://doi.org/10.6084/m9.figshare.21675692.v9.

ETHICS STATEMENT

All sampling protocols followed guidelines approved by the American Society of Mammalogists for the capture, handling and care of mammals (Sikes & The Animal Care and Use Committee of the American Society of Mammalogists, 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).

ORCID

Marcelo Araya-Salas https://orcid.org/0000-0003-3594-619X
Juan C. Brenes https://orcid.org/0000-0003-3746-7272

REFERENCES

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

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.

Araya-Salas, M., Hernández-Pinsón, H. A., Rojas, N., & Chaverri, G. (2020). Ontogeny of an interactive call-and-response system in Spix's disc-winged bats. *Animal Behaviour*, *166*, 233–245.

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*, e63377.

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.

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.

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

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.

Goldberg, A. V., & Tarjan, R. E. (1988). A new approach to the maximum-flow problem. *Journal of the ACM (JACM)*, *35*(4), 921–940.

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*, 725–741.

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), 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*. https://doi.org/10.5751/ACE-01114-120214

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), 1–124.

Lostanlen, V., Salamon, J., Farnsworth, A., Kelling, S., & Bello, J. P. (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., Heittola, T., & Virtanen, T. (2016). Metrics for polyphonic sound event detection. *Applied Sciences*, *6*(6), 162.

Mesaros, A., Heittola, T., Virtanen, T., & Plumbley, M. D. (2021). Sound event detection: A tutorial. *IEEE Signal Processing Magazine*, *38*(5), 67–83.

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., de Andrade, L. G., Hedwig, D., Keen, S. C., Shiu, Y., Charif, R. A., Webster, M. S., & Rice, A. N. (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., & The Animal Care and Use Committee of the American Society of Mammalogists. (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., von Ziegler, L., Schläppi, C., & Akyol, F. (2020). Deep learning-based behavioral analysis reaches human accuracy and is capable of outperforming commercial solutions. *Neuropsychopharmacology*, *45*(11), 1942–1952.
Sugai, L., Silva, T., Ribeiro, J. W., Jr., & Llusia, D. (2019, 1). Terrestrial passive acoustic monitoring: Review and perspectives. *BioScience*, *69*, 15–25.
Tchernichovski, O., Eisenberg-Edidin, S., & Jarvis, E. D. (2021). Balanced imitation sustains song culture in zebra finches. *Nature Communications*, *12*(1), 1–14.
Tiwari, C., & Diwakar, S. (2022). The katydid country: Bioacoustics and ecology of tettigoniid communities from the indian subcontinent. *Bioacoustics*, *2022*, 1–25.

How to cite this article: Araya-Salas, M., Smith-Vidaurre, G., Chaverri, G., Brenes, J. C., Chirino, F., Elizondo-Calvo, J., & Rico-Guevara, A. (2023). ohun: An R package for diagnosing and optimizing automatic sound event detection. *Methods in Ecology and Evolution*, *14*, 2259–2271. https://doi.org/10.1111/2041-210X.14170