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

Marcelo Araya-Salas 1, 2, 3 \*, Grace Smith-Vidaurre 4, 5, 6, Gloriana Chaverri 3, 7 Juan Carlos Brenes 1 Fabiola Chirino 1 & 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

Check [this link](https://rmarkdown.rstudio.com/authoring_bibliographies_and_citations.html) for details on how to add citations/bibliography in Rmarkdown

## 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](#ref-Burgdorf2011); [S 2020](#ref-schoneich2020)) to taxonomy ([Köhler *et al.* 2017](#ref-Kohler2017); [Gwee *et al.* 2019](#ref-gew2019)), community ecology ([Zsebők *et al.* 2021](#ref-zsebok2021); [Tiwari & Diwakar 2022](#ref-tiwari2022)) and evolutionary biology ([Medina-García *et al.* 2015](#ref-Medina-Garcia2015a); [Odom *et al.* 2021](#ref-Odom2021)). The profuse usage of animal sounds in research is partly due to the fact that they can be easily collected using non-invasive methods. Animal sounds can also be obtained in a variety of settings from laboratories to natural areas, while the equipment required for collecting an analyzing these sounds has become increasingly inexpensive and broadly accesible ([Blumstein *et al.* 2011](#ref-Blumstein2011); [Sugai *et al.* 2019](#ref-sugai2019)). In addition, the existence of online repositories has facilitated the study of these communication signals at larger taxonomic and geographic scales. However, the adoption of bioacustic approaches in research may also imply large amounts of data (i.e. lots of recordings), which can be challenging to analyzed manually ([Gibb *et al.* 2019](#ref-gibb2019)). As a result, a growing number of computational tools for the automatic detection of animal sounds on is increasingly available (reviewed by [Stowell 2022](#ref-Stowell2022)), reflecting the need of automated approaches for efficiently conducting these analyses ([Gibb *et al.* 2019](#ref-gibb2019)).

The growing availability of tools for automatic detection of acoustic events, particularly as free software, is expected to further simplify acoustic data processing, making it 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](#ref-Stowell2022)). In this regard, the use of standard approaches for evaluating the performance of automatic detection tools might prove helpful to inform researchers’ decisions about which method better fits a given question and study system ([Knight *et al.* 2017](#ref-knight2017)). The performance of automated sound event detection routines have typically been evaluated using standard indices from signal detection theory ([Macmillan & Creelman 2004](#ref-macmillan2004); [Knight *et al.* 2017](#ref-knight2017); [Balantic & Donovan 2020](#ref-balantic2020)). 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 correctly detected sounds (true positives), wrongly detected sounds (false positives) and missed sounds (false negatives) as well as additional metrics derived from these indices (e.g. recall, precision). However, the fact that sound events are not being evaluated as discrete classification units (as opposed to for instance identifying species from pictures), but are rather embedded within a continuous string of sound, demand additional information to fully diagnose the temporal precision detection performance. This is particularly relevant when the precise time position of sounds is needed as is common when measuring the acoustic structure of sounds is the main goal ([Araya-Salas & Smith-Vidaurre 2017](#ref-Araya-Salas2017f)). For instance, the same signal can be detected as several separated sounds, the inferred time position can be offset from the the target signal position, or several sounds can be detected as a single one. Therefore, metrics that account for these additional performance dimensions are a valuable tool to properly diagnose 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 in order to simplify their optimization. The package makes use of reference annotations containing the time position of target sounds that along with the corresponding sound files, serves as a training data set in which to evaluate the performance of detection routines. This can be done with routine outputs imported from other software as well as detection routines run within the *ohun* package itself. The package also provides a set of functions to explore acoustic data sets and organize them in a format amenable 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](#ref-mellinger2000); [Charif *et al.* 2010](#ref-Charif2010); [Aide *et al.* 2013](#ref-aide2013); [Bioacoustics 2014](#ref-avi2014); [Hafner & Katz 2015](#ref-Hafner2015)). We explain how to explore and format acoustic data sets, how sound event detection routines can be evaluated and then showcase the package usage with study cases on male Zebra-finch songs (*Taenopygia gutata*) and Spix’s disc-winged bat calls (*Thyroptera tricolor*) which represent different recording settings (lab and flight cages) and signal types (sonic mating sounds and ultrasonic social calls).

## Formatting acoustic data sets

The format and size of the acoustic data to be analyzed is important to validate and standardize in order to avoid downstream errors, as well as 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)

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 duration and size of files. 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 hard to analyze in 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   
## 1 "Ag13\_43421.27975590\_11\_17\_7\_46\_15.wav"   
## 2 "Ag13\_43421.27975590\_11\_17\_7\_46\_15.wav"   
## 3 "Ag13\_43421.27975590\_11\_17\_7\_46\_15.wav"   
## 4 "Ag13\_43421.27975590\_11\_17\_7\_46\_15.wav"   
## 5 "Ag13\_43421.27975590\_11\_17\_7\_46\_15.wav"   
## 6 "Blk109Brn\_43559.32349131\_4\_4\_8\_59\_9.wav"  
## sound.files start end   
## 1 "Ag13\_43421.27975590\_11\_17\_7\_46\_15-1.wav" " 0" " 5.00000"  
## 2 "Ag13\_43421.27975590\_11\_17\_7\_46\_15-2.wav" " 5" "10.00000"  
## 3 "Ag13\_43421.27975590\_11\_17\_7\_46\_15-3.wav" "10" "15.00000"  
## 4 "Ag13\_43421.27975590\_11\_17\_7\_46\_15-4.wav" "15" "20.00000"  
## 5 "Ag13\_43421.27975590\_11\_17\_7\_46\_15-5.wav" "20" "21.76138"  
## 6 "Blk109Brn\_43559.32349131\_4\_4\_8\_59\_9-1.wav" " 0" " 5.00000"

The output shows the time segments in the original sound files to which the clips belong to. 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 sound files.

Annotations can also be explored using the function feature\_reference, which returns the mean and range of signal duration and gap duration (time intervals between selections), bottom and top frequency and of the number of annotations by sound file. If the path to the sound files is supplied, then the duty cycle (the fraction of a sound file corresponding to sounds) and peak amplitude (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 43.28 65.19 88.69  
## bottom.freq 0.50 0.50 0.50  
## top.freq 10.00 10.00 10.00

## Diagnosing detection performance

The *ohun* package makes use of 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’s widely used for optimizing this decision making process in the presence of uncertainty ([Hossin & Sulaiman 2015](#ref-hossin2015)). During a detection routine, the detected ‘items’ can be classified into 4 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 might not be partitioned in 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 the context of sound event detection ([Knight *et al.* 2017](#ref-knight2017)): recall (correct detections relative to total detections), precision (proportion of target sounds that were correctly detected) and F1 score (combined recall and precision as the harmonic mean of these two, providing 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 false negatives, which will result 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 will be given 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](#ref-Charif2010)) 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")

## tuning\_parameters detections true.positives  
## [1,] "band: 1-10 kHz; sep: 0.005 s" " 574" "448"   
## [2,] "band: 1-10 kHz; sep: 0.02322 s" " 155" " 30"   
## [3,] "band: 1-15 kHz; sep: 0.02322 s" " 632" "375"   
## [4,] "band: 1-22 kHz; sep: 0.02322.txt s" "1020" "423"   
## false.positives false.negatives splits merges overlap recall   
## [1,] "126" "143" "0" "0" "0.8323344" "0.75803723"  
## [2,] "125" "561" "0" "0" "0.7434591" "0.05076142"  
## [3,] "257" "216" "0" "0" "0.8331330" "0.63451777"  
## [4,] "597" "168" "0" "0" "0.8326257" "0.71573604"  
## precision f1.score   
## [1,] "0.7804878" "0.76909871"  
## [2,] "0.1935484" "0.08042895"  
## [3,] "0.5933544" "0.61324612"  
## [4,] "0.4147059" "0.52513966"

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’ 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 to detail those indices by sound file. Here we show the first 10 files:

diag\_raven <- diagnose\_detection(reference = manual\_ref\_tae, detection = raven\_detec,  
 by = "tuning\_parameters", by.sound.file = TRUE)  
  
head(diag\_raven, 10)

diag\_raven <- diagnose\_detection(reference = manual\_ref\_tae, detection = raven\_detec,  
 by = "tuning\_parameters", by.sound.file = TRUE)  
  
as.matrix(diag\_raven[1:10, ])

## tuning\_parameters sound.files   
## 1 "band: 1-10 kHz; sep: 0.005 s" "Ag13\_43421.27975590\_11\_17\_7\_46\_15.wav"   
## 2 "band: 1-10 kHz; sep: 0.005 s" "BRN7\_43435.27985312\_12\_1\_7\_46\_25.wav"   
## 3 "band: 1-10 kHz; sep: 0.005 s" "Blk109Brn\_43559.32349131\_4\_4\_8\_59\_9.wav"   
## 4 "band: 1-10 kHz; sep: 0.005 s" "DB118\_43568.33139566\_4\_13\_9\_12\_19.wav"   
## 5 "band: 1-10 kHz; sep: 0.005 s" "DB15HP\_43450.29217192\_12\_16\_8\_6\_57.wav"   
## 6 "band: 1-10 kHz; sep: 0.005 s" "DB7\_43357.58119484\_9\_14\_16\_8\_39.wav"   
## 7 "band: 1-10 kHz; sep: 0.005 s" "DG124DB\_43559.30160960\_4\_4\_8\_22\_40.wav"   
## 8 "band: 1-10 kHz; sep: 0.005 s" "GRY37HP\_43442.27431670\_12\_8\_7\_37\_11.wav"   
## 9 "band: 1-10 kHz; sep: 0.005 s" "Gold183\_43555.5549813\_3\_31\_1\_32\_29.wav"   
## 10 "band: 1-10 kHz; sep: 0.005 s" "Gry35HP\_43455.29800260\_12\_21\_8\_16\_40.wav"  
## detections true.positives false.positives false.negatives splits merges  
## 1 "40" "35" " 5" " 0" "0" "0"   
## 2 "34" "21" "13" "30" "0" "0"   
## 3 "30" "27" " 3" " 7" "0" "0"   
## 4 "18" "18" " 0" " 1" "0" "0"   
## 5 "23" "21" " 2" " 0" "0" "0"   
## 6 "18" "18" " 0" " 1" "0" "0"   
## 7 "39" "29" "10" " 0" "0" "0"   
## 8 "54" "45" " 9" " 9" "0" "0"   
## 9 " 2" " 2" " 0" " 0" "0" "0"   
## 10 "18" "15" " 3" "11" "0" "0"   
## overlap recall precision f1.score   
## 1 "0.8557378" "1.0000000" "0.8750000" "0.9333333"  
## 2 "0.7473391" "0.4117647" "0.6176471" "0.4941176"  
## 3 "0.8521391" "0.7941176" "0.9000000" "0.8437500"  
## 4 "0.8080231" "0.9473684" "1.0000000" "0.9729730"  
## 5 "0.9019064" "1.0000000" "0.9130435" "0.9545455"  
## 6 "0.8816899" "0.9473684" "1.0000000" "0.9729730"  
## 7 "0.8148303" "1.0000000" "0.7435897" "0.8529412"  
## 8 "0.8745580" "0.8333333" "0.8333333" "0.8333333"  
## 9 "0.8266727" "1.0000000" "1.0000000" "1.0000000"  
## 10 "0.8185580" "0.5769231" "0.8333333" "0.6818182"

Diagnostics from routines using different tuning parameters can be used to identify the parameter values that optimize detection. These 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 on 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 highly stereotyped or good signal-to-noise ratio (SNR) 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 south-western Costa Rica in January 2020. Bats were captured at their roosting sites (furled leaves of Zingeberaceae plants). Each individual bat was released in a large flight cage (9 x 4 x 3 m) for a 5-minute period 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](#ref-Charif2010)). 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% of overlap, and were then imported into R using the package Rraven ([Araya-Salas 2020](#ref-Araya-Salas2017e)).

Inquiry calls of Spix’s disc-winged bats are structurally stereotyped ([Chaverri *et al.* 2010](#ref-Chaverri2010b)). Most variation is found among individuals, although the basic form of a short, downward broadband frequency modulation is always shared (Fig. BAT-SPECTRO, Araya-Salas et al 2021 ontogeny).

## Error in knitr::include\_graphics(file.path(rootdir, "data/raw/thyroptera/Catalog\_p1.jpeg")): Cannot find the file(s): "../data/raw/thyroptera/Catalog\_p1.jpeg"

Template-based detection is a useful approach when there are minimal structural differences in the target sounds ([Knight *et al.* 2017](#ref-knight2017); e.g. when signals are produced in a highly stereotyped manner, [Balantic & Donovan 2020](#ref-balantic2020)). 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 parameters which are then 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-size slices of a sphere centered at the centroid of the acoustic space. Templates are then selected as those closer to the centroid within each of the 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.spaces = 3)

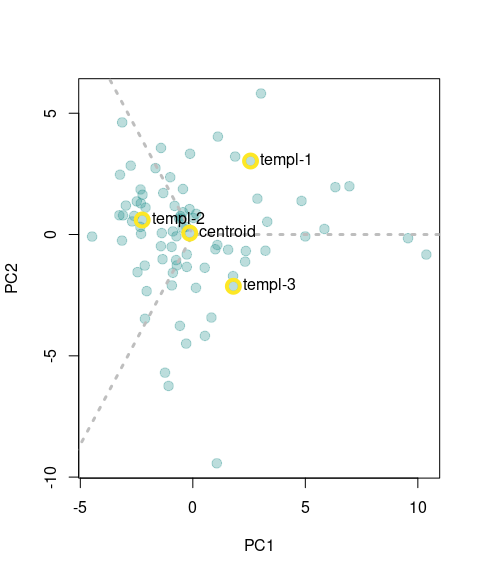


FIG. ACOUSTIC-SPACE. Acoustic space defined as the the first two components of a Principal Component Analysis on spectrographic parameters. Templates are selected as those closer to the centroid for each of the sub-spaces. Gray dashed lines delimit the region of sub-spaces. Yellow circles around points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

The output of the get\_templates function includes an acoustic space plot (FIG. ACOUSTIC-SPACE) 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 those templates for detecting bat social calls. The code iterates a template-based detection on the training data set across a range of correlation thresholds, 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 TEMPLATE PERFORMANCE shows the two highest performance runs for each template.

TABLE TEMPLATE PERFORMANCE. 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. THRESHOLD vs F1.SCORE).

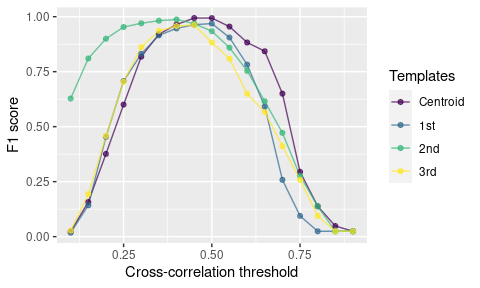


FIG. THRESHOLD vs F1.SCORE. Shows the changes in F1 score across the range of threshold values

In this example the “centroid” template, produced the best performance (TABLE TEMPLATE PERFORMANCE; FIG. THRESHOLD vs F1.SCORE). 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 use 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)

## detections true.positives false.positives false.negatives splits merges  
## 1 532 522 10 41 0 14  
## overlap recall precision f1.score  
## 1 0.8345997 0.9271758 0.981203 0.9534247

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

### Energy-based detection on zebra finch vocalizations

We used recordings from 18 zebra finch males recorded at obtained from the Rockefeller University Field Research Center Song Library (<http://ofer.sci.ccny.cuny.edu/songs>, [Tchernichovski *et al.* 2021](#ref-Tchernichovski2021)). Recordings contain undirected vocalizations (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 importantly in key features as duration and frequency range (ZEBRAFINCH-SPECTRO) and are clearly not as stereotyped as the Spix’s disc-winged bats. However, as the recorded sounds show a good signal-to-noise ratio, such that 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.

## Error in knitr::include\_graphics(file.path(rootdir, "data/raw/taeniopygia/same.time-Catalog\_p1.jpeg")): Cannot find the file(s): "../data/raw/taeniopygia/same.time-Catalog\_p1.jpeg"

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):

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):

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. Hhere we show the 10 combinations with the highest F1 score:

# 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  
## 45 1 5 5 25 300 105  
## 69 1 5 5 25 325 105  
## 41 1 5 0 25 300 105  
## 65 1 5 0 25 325 105  
## 43 1 10 0 25 300 105  
## 67 1 10 0 25 325 105  
## 37 1 5 5 15 300 105  
## 61 1 5 5 15 325 105  
## 14 5 5 5 15 275 96  
## 38 5 5 5 15 300 96  
## false.positives false.negatives recall precision f1.score  
## 45 9 0 1.0000000 0.9210526 0.9589041  
## 69 9 0 1.0000000 0.9210526 0.9589041  
## 41 12 0 1.0000000 0.8974359 0.9459459  
## 65 12 0 1.0000000 0.8974359 0.9459459  
## 43 13 0 1.0000000 0.8898305 0.9417040  
## 67 13 0 1.0000000 0.8898305 0.9417040  
## 37 15 0 1.0000000 0.8750000 0.9333333  
## 61 15 0 1.0000000 0.8750000 0.9333333  
## 14 8 9 0.9142857 0.9230769 0.9186603  
## 38 8 9 0.9142857 0.9230769 0.9186603

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 sound files 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)

detections

true.positives

false.positives

false.negatives

splits

merges

overlap

recall

precision

f1.score

490

449

41

37

0

0

0.933

0.924

0.916

0.92

The performance on the test data set was also acceptable with a 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 be used to remove ambiguous sounds (i.e. those labeled 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 when running template-based detection (‘template\_detection’ function).

## Discusion

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 in order to find those values that optimize detection. Despite signal detection indices being commonly reported when presenting new automatic detection methods, to our knowledge widely applicable performance evaluating routines have not been made available in a free, open source platform. Providing a common framework for the evaluation of sound event detection can simplify comparing the performance of different tools and the selection of those tools better suited to a given research question and study system. The tools offered by *ohun* for diagnosing detection performance should not be necessarily 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](#ref-sturman2020); [Hsu & Yttri 2021](#ref-hsu2021); [Bohnslav *et al.* 2021](#ref-bohnslav2021)). The detection of such motor events in video can also be evaluated and optimized in comparison to a reference annotation, as we have shown above for acoustic signals.

The *ohun* package provides two native 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](#ref-mellinger2000); [Charif *et al.* 2010](#ref-Charif2010); [Aide *et al.* 2013](#ref-aide2013); [Bioacoustics 2014](#ref-avi2014); [Hafner & Katz 2015](#ref-Hafner2015)) and can reach adequate performance under the appropriate conditions, as we have shown in our two study cases and has been shown elsewhere ([Knight *et al.* 2017](#ref-knight2017)). Deep learning methods tend to require greater computational power, larger training data sets and, in some cases, more complex training routines. This might bring unnecessary difficulties when dealing with when 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 having tools accessible for a broader community of researchers. The tools offered in *ohun* can also be used in a subsequent pipeline in which detected sounds are further classified using more elaborated discrimination algorithms ([Balantic & Donovan 2020](#ref-balantic2020)). In addition, detection performance can be improved by using acoustic structure measurements that can be used 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. large recordings and/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 that you obtain decent results on a small subset of recordings before trying to scale 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 functions for automatic detection and performance evaluation allow users to run analysis in parallel (see parallel argument in those functions). Hence a computer with several cores can be helpful for improving efficiency. 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. 5. Try using a computer with lots of RAM memory or a computer cluster for working on large amounts of data. Lastly, we underscore that these tips are not restricted to *ohun* and can also be helpful to speed up routines in other software packages.

There are some additional things to be considered when aiming to automatically detect sound events. When running energy-based detection routines try to use your knowledge about the signal structure to determine the initial range of tunning parameters. This can be extremely helpful for narrowing down possible parameter values. As a general rule, if human observers have a hard time 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 (like a multi-syllable vocalization) in the reference annotation when running an energy-based detector, as they are likely to be identified as separated units.

### Acknowledgements

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 and Jorge Elizondo for his help on manually annotating recordings. We also thank the Centro Biológico Hacienda Barú for their continuous support of our research and Mijail Rojas and Andrey Sequeira for support on 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 capture, handling and care of mammals ([Sikes *et al.* 2016](#ref-Sikes2016)) 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 XXXXXXXXXXXXXXXXX

## 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](https://doi.org/10.7717/PEERJ.103). *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. Retrieved from <http://onlinelibrary.wiley.com/doi/10.1111/2041-210X.12624/abstract>

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. Retrieved from <https://www.tandfonline.com/action/journalInformation?journalCode=tbio20>

Bioacoustics, A. (2014). Avisoft SasLab pro.

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](https://doi.org/10.1111/j.1365-2664.2011.01993.x). *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](https://doi.org/10.7554/ELIFE.63377). *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](https://doi.org/10.1016/j.neubiorev.2010.11.011). *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](https://doi.org/10.1098/rsbl.2009.0964). *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. Retrieved from <https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/2041-210X.13101>

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**. Retrieved from <https://academic.oup.com/zoolinnean/article-abstract/186/3/725/5372980>

Hafner, S. & Katz, J. (2015). monitoR: Acoustic template detection in r. *R package version 1.0.3*. Retrieved from <http://www.uvm.edu/rsenr/vtcfwru/R/?Page=monitoR/monitoR.htmURL: http://www.uvm.edu/rsenr/vtcfwru/R/?Page=monitoR/monitoR.htmURL: http://www.uvm.edu/rsenr/vtcfwru/R/?Page=monitoR/monitoR.htm>

Hossin, M. & Sulaiman, M. (2015). A review on evaluation metrics for data classification evaluations. *International journal of data mining*. Retrieved from <https://www.academia.edu/download/37219940/5215ijdkp01.pdf>

Hsu, A. & Yttri, E. (2021). B-SOiD, an open-source unsupervised algorithm for identification and fast prediction of behaviors. *Nature Communications*. Retrieved from <https://www.nature.com/articles/s41467-021-25420-x>

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](https://doi.org/10.5751/ACE-01114-120214). *Avian Conservation and Ecology, Published online: Nov 27, 2017 | doi:10.5751/ACE-01114-120214*, **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*](https://doi.org/10.11646/zootaxa.4251.1.1).

Macmillan, N.A. & Creelman, C.D. (2004). [Detection theory: A user’s guide: 2nd edition](https://doi.org/10.4324/9781410611147/DETECTION-THEORY-NEIL-MACMILLAN-DOUGLAS-CREELMAN). *Detection Theory: A User’s Guide: 2nd edition*, 1–445.

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. Retrieved from <http://doi.wiley.com/10.1111/jeb.12694>

Mellinger, D.K. & Clark, C.W. (2000). [Recognizing transient low-frequency whale sounds by spectrogram correlation](https://doi.org/10.1121/1.429434). *The Journal of the Acoustical Society of America*, **107**, 3518–3529.

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, A.N. (2021). [Comparative bioacoustics: A roadmap for quantifying and comparing animal sounds across diverse taxa](https://doi.org/10.1111/brv.12695). *Biological Reviews*, **96**, 1135–1159.

S, S. (2020). Neuroethology of acoustic communication in field crickets-from signal generation to song recognition in an insect brain. *Progress in Neurobiology*. Retrieved from <https://www.sciencedirect.com/science/article/pii/S0301008220301374>

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. Retrieved from <https://academic.oup.com/jmammal/article/97/3/663/2459909 http://www.mammalsociety.org/committees/animal-care-and-use#tab3>

Stowell, D. (2022). Computational bioacoustics with deep learning: A review and roadmap. *PeerJ*, **10**, e13152. Retrieved from <https://peerj.com/articles/13152>

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*. Retrieved from <https://www.nature.com/articles/s41386-020-0776-y>

Sugai, L., Silva, T., Jr, J.R. & Llusia, D. (2019). Terrestrial passive acoustic monitoring: Review and perspectives. *academic.oup.com*. Retrieved from <https://academic.oup.com/bioscience/article-abstract/69/1/15/5193506>

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. Retrieved from <https://www.nature.com/articles/s41467-021-22852-3>

Tiwari, C. & Diwakar, S. (2022). [The katydid country: Bioacoustics and ecology of tettigoniid communities from the indian subcontinent](https://doi.org/10.1080/09524622.2022.2064330). *Bioacoustics*.

Zsebők, S., Schmera, D., Miklós Laczi, |, Nagy, G., Vaskuti, É., János Török, |, László, | & Garamszegi, Z. (2021). A practical approach to measuring the acoustic diversity by community ecology methods. *Wiley Online Library*, **12**, 874–884. Retrieved from <https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/2041-210X.13558>