*ohun*: an R package for optimizing automatic acoustic signal detection

Marcelo Araya-Salas 1, 2, 3 \*, Grace Smith-Vidaurre 4, Gloriana Chaverri 3, 5, Fabiola Chirino 1 & Alejandro Rico-Guevara 6, 7

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 GRACE’S AFFILIATION

5 Smithsonian Tropical Research Institute, Panama City, Panamá

6 Department of Biology, University of Washington, Seattle, USA

7 Burke Museum of Natural History and Culture, University of Washington, Seattle, USA

\* *To whom correspondence should be addressed*

### 

### Keywords:

Word count: 3814 (including code chunks: 5445)

## Abstract

Animal acoustic signals are widely used in a diversity of research areas. This is partly due to the relative ease in which they can be registered for a wide range of taxonomic groups and research settings. However, bioacoustic research can easily generate large amounts of data which might prove challenging to analyze in a timely manner. Many tools for the automatic detection of acoustic signals are currently available. However, choosing the right approach for a specific task might turn difficult as there are no available tools that can provide a common framework for evaluating detection performance. Here we present the new R package *ohun*. This package is intended to facilitate the automatic detection of acoustic signals, by providing functions to diagnose and optimize detection routines as well as to compare the performance of different detection approaches. The package makes use of reference annotations containing the time position of target signals in a training data set to evaluate the performance of detection routines, using common signal detection theory indices. This can be done with routine outputs imported from other software and from detection run within the package itself. The package also provides a set of functions to organize acoustic data sets in a format amenable for detection analyses. It also includes an implementation of two automatic detection methods commonly used in bioacoustic analysis: energy-based detection and template-based detection. We included examples on how to evaluate detections from external software and showcase the package usage with study cases on zebra-finch (*Taenopygia gutata*) songs and Spix’s disc-winged bat (*Thyroptera tricolor*) ultrasonic social calls. Finally, we provide some general suggestions to improve signal detection performance.

## Introduction

Animal acoustic signals are widely used to investigate a variety of questions in highly diverse areas, ranging from neurobiology to community ecology and evolutionary biology. The profuse usage of animal sounds in research is partly due to the fact that they can be easily registered using non-intrusive methods, that they can be obtained in a variety of settings from laboratories to natural areas, and that the equipment required for registering an analyzing these signals has become increasingly inexpensive. In addition, the existence of online repositories and growing number has facilitated the study of these traits at larger taxonomic and geographic scales. However, the adoption of bioacoustic approaches in research may also imply large amounts of data (*i.e.* lots of recordings), which can be challenging to analyze manually. As a result, a growing number of computational tools for the analysis of acoustic features on those signals is increasingly available (reviewed by Stowell ([2022](#ref-Stowell2022))), reflecting the need of automated approaches for efficiently conducting these analyses.

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 tools also posits a challenge as it can be difficult to navigate. In this regard, the use of standard approaches for evaluating the performance of automatic detection tools might prove helpful to inform the decision about which method better fits a given study system. The performance of automatic acoustic signal detection routines has been evaluated using standard indices from signal detection theory (CITATIONS). 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 signals have been annotated. This comparison allows to quantify the number of correctly detected signals (true positives), wrongly detected signals (false positives) and missed signals (false negatives) as well as additional metrics derived from these indices (*e.g.*, recall, precision). However, the fact that acoustic signals are not being evaluated as discrete classification units (as opposed to, for instance, identifying species from pictures), demands additional information to fully diagnose detection performance. This is particularly relevant when the precise time position of signals is needed, as is common for research in which obtaining measurements of the acoustic structure of signals is the main goal. For instance, the same signal can be detected as several separated signals, the inferred time position can be far off from the target signal position, or several signals can be detected as a single one. Therefore, metrics that account for these additional performance dimensions are a valuable tool to properly diagnose automatic acoustic signal detection approaches.

Here we present the new R package *ohun*. The package is intended to facilitate the automatic detection of acoustic signals, providing functions to diagnose specific aspects of acoustic detection routines in order to simplify their optimization. The package makes use of reference annotations containing the time position of target signals, in a training data set, to evaluate the performance of detection routines. This can be done with routine outputs imported from a variety of acoustic processing programs, as well as with signal detection run within the package itself. The package also provides a set of functions to explore acoustic datasets 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. We explain how to explore and format acoustic data sets, how acoustic signal detection routines can be evaluated, and then we showcase the package usage with study cases on Zebra finch songs (*Taeniopygia guttata*) and Spix’s disc-winged bats (*Thyroptera tricolor*), which represent different recording settings (lab and flight cages) and signal types (sonic mating signals and ultrasonic social calls).

## Formatting acoustic data sets

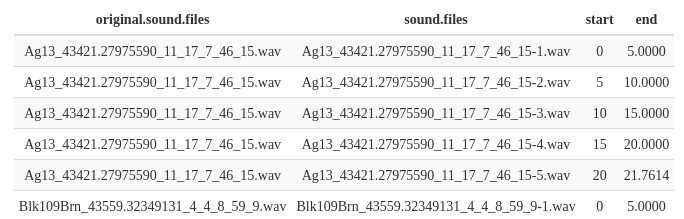
Having some sense of the format and size of the sound files to be analyzed is an important step for avoiding downstream errors and informing time performance expectations. Several functions can facilitate getting acoustic data sets in the right format prior to automatic detection. The function feature\_acoustic\_data 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 songs (suppl. mat):

# working directory  
path\_zebra\_finch <- "path\_to\_zebra\_finch\_files"  
  
feature\_acoustic\_data(path = path\_zebra\_finch)

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

In this case all recordings have 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, which is important as some tuning parameters of detection routines can behave differently depending on file format (*e.g.*, time window size is 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 regular 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)



The output shows to which time segments in the original sound files 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 signals 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 signal mean and range, gap duration (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, the duty cycle (fraction of a sound file corresponding to signals) and peak amplitude (highest amplitude in a detection) are also returned:

# read reference annotations  
manual\_ref\_tae <- read.csv(file.path(path\_zebra\_finch, "manual\_selections\_Taeniopygia.csv"))  
  
# explore annotations  
feature\_reference(reference = manual\_ref\_tae, path = path\_zebra\_finch)

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

## Diagnosing detection performance

The *ohun* package makes use of signal detection theory indices to evaluate detection performance. Signal detection theory deals with the process of recovering signals (*i.e.* target signals) from background noise (not necessarily acoustic noise), and it is widely used for optimizing this decision making process in the presence of uncertainty. During a detection routine, the detected ‘items’ can be classified into 4 classes: true positives (TPs, target signals correctly identified as signal), false positives (FPs, noise incorrectly identified as ‘signal’), false negatives (FNs, signals 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 acoustic signal detection as noise cannot always 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.

* Recall: correct detections relative to total detections (a.k.a. true positive rate or sensitivity; TPs / (TPs + FNs)).
* Precision: correct detections relative to total detections (TPs / (TPs + FPs)).
* F1 score: combines recall and precision as the harmonic mean of these two, providing a single value for evaluating performance (a.k.a. F-measure or Dice similarity coefficient).

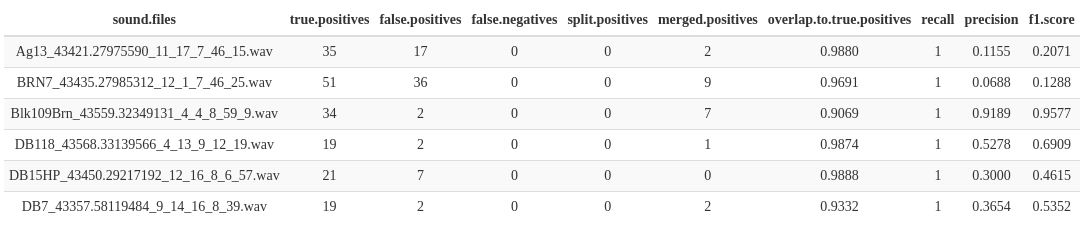
A perfect detection will have no false positives or false negatives, which will result in both recall and precision equal to 1. However, perfect detection cannot always be achieved and some compromise between detecting all target signals plus some noise (recall = 1 & precision < 1) and detecting only target signals but not all of them (recall < 1 & precision = 1) might be warranted. The right balance between these two extremes will be given by the relative costs of missing signals and mistaking noise for signals, given the specific goals of the study. These indices provide a useful framework for diagnosing and optimizing the performance of a detection routine.

This package offers tools to evaluate the performance of an acoustic signal detection 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 signals in the sound files. For instance, the following code evaluates a routine run in Raven Pro 1.6 (XXXX) using the “band limited energy detector” option (minimum frequency: 0.8 kHz; maximum frequency: 22 kHz; minimum duration: 0.54989 s; maximum duration: 0.54989s; minimum separation: 0.02268 s) on a subset of the Zebra finch recordings described below:

diagnose\_detection(reference = manual\_ref\_tae, detection = raven\_detec)

The output shows the indices described above. The function also allows to detail those indices by sound file. Here we show the output for the first six files:

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



Diagnostics from routines using different tuning parameters can be used to identify the parameter values that optimize detection. The 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 to *ohun*.

## Signal detection with *ohun*

This 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) signals, respectively. If the target signals 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 Barú Biological Station, in south-western Costa Rica in January 2020. Bats were captured at their roosting sites (furled leaves of Zingiberaceae 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 (cite) 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 (XXXX). Annotations were created by visual inspection of spectrograms, in which the start and end of signals were determined by the location of the continuous traces of power spectral entropy of the target signals. 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 (XXXX).

Inquiry calls of Spix’s disc-winged bats are structurally stereotyped. 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).

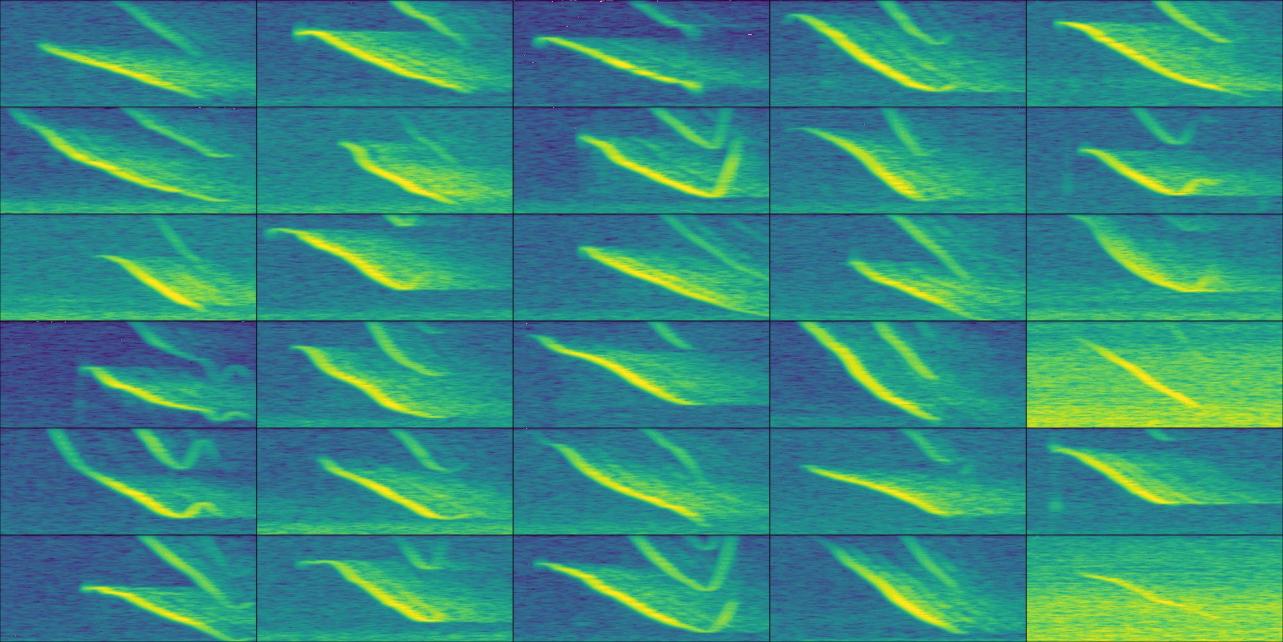


Fig. BAT-SPECTRO. Example spectrograms of Spix’s disc-winged social calls for each of the 30 recordings used in the analysis. The highest signal-to-noise ratio call by sound file are shown. The time scale range is 71 ms and the frequency range 10-44 kHz

Template-based detection is a useful approach when there are few structural differences in the target signals. 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. The function get\_templates finds several signals representative of the variation in signal structure. This function measures several spectrographic parameters which are then summarized using a Principal Component Analysis. The first two components are used to project the acoustic space. On this space the function defines sub-spaces as equal-size slices of a circle 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)

## The first 2 principal components explained 0.51 of the variance

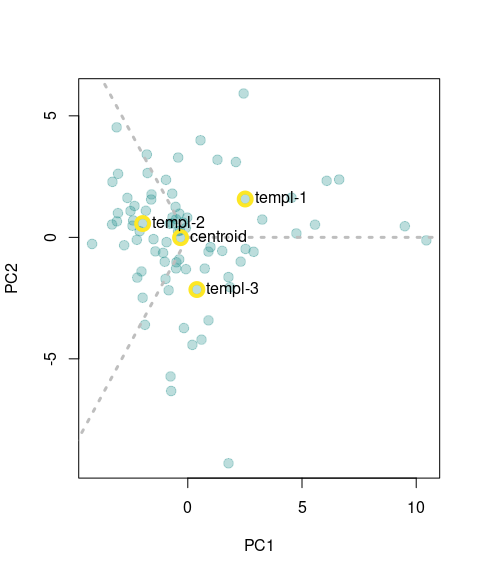


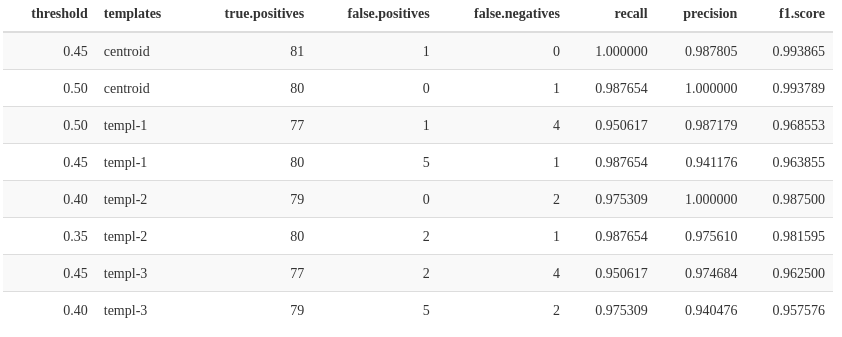
FIG. ACOUSTIC-SPACE. Acoustic space defined as the first two components of a Principal Component Analysis on spectrographic parameters. Templates are selected as those closer to the centroid for each of the sub-spaces. Gray dashed lines delimit the region of sub-spaces. Yellow circles around points highlight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



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).

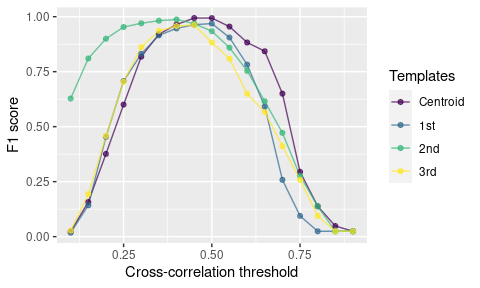


FIG. THRESHOLD vs F1.SCORE. Shows the changes in F1 score across the range of threshold values

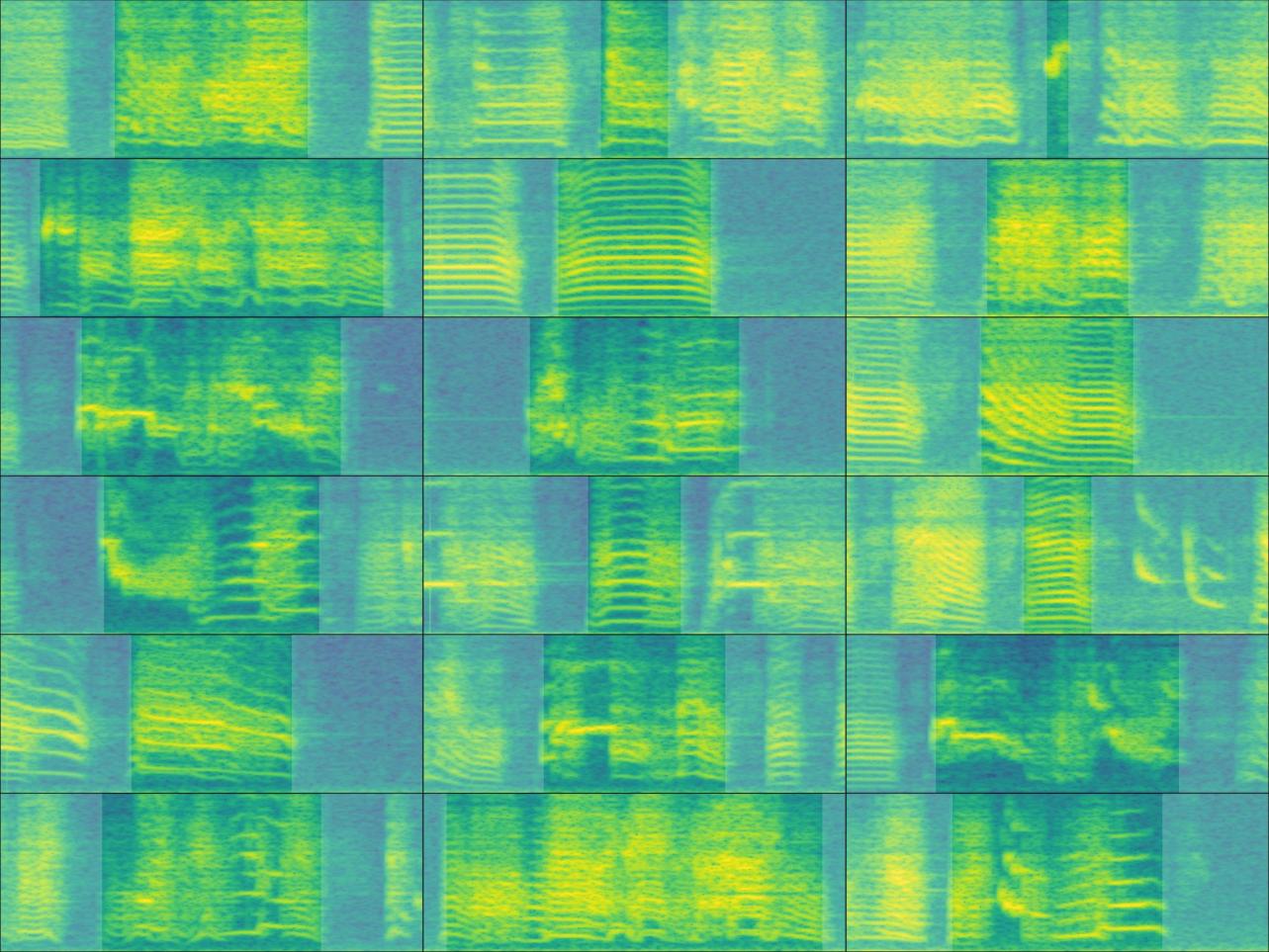
In this example the “centroid” template, produced the best performance (TABLE TEMPLATE PERFORMANCE; FIG. THRESHOLD vs F1.SCORE). Hence, we will use this template for detecting calls on the rest of the data. The following code extracts this template from the reference annotation table and use it to find 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)

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

### Energy-based detection on zebra finch songs

We used recordings from 18 Zebra finch males. These recordings were obtained ….

Zebra finch vocalizations are composed by 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). However, as the recorded signals show a good signal-to-noise ratio, they can potentially be detected using an energy-based approach.  


The following code loads the reference annotations and split them in two data set for training (3 sound files) and testing (15 sound files):

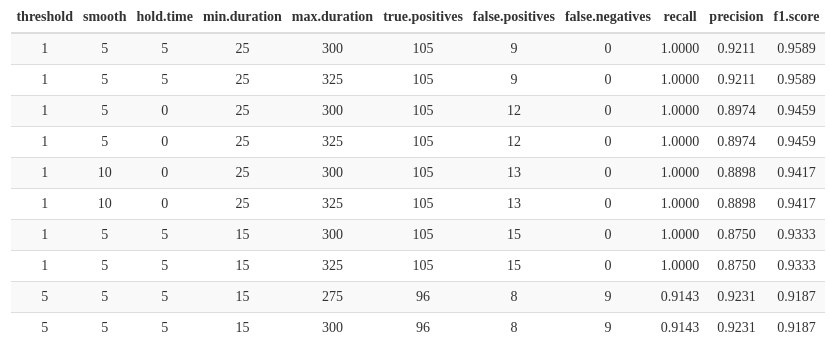
manual\_ref\_tae <- read.csv("manual\_selections\_Taeniopygia.csv")  
  
set.seed(450)   
train\_files <- sample(unique(manual\_ref\_tae$sound.files), 3)   
test\_files <- setdiff(manual\_ref\_tae$sound.files, train\_files)  
  
train\_ref <- manual\_ref\_tae[manual\_ref\_tae$sound.files %in% train\_files, ]  
test\_ref <- manual\_ref\_tae[manual\_ref\_tae$sound.files %in% test\_files, ]

Now we can optimize the detection parameters 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 signals 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. Here 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)



We now can use the tuning parameter values that yielded the best performance to detect signals 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)



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 signals or background noise levels.

### Additional tools

The *ohun* package offers additional tools to simplify acoustic signal detection. The function ‘feature\_reference’ allows to extract information about the duration and frequency range of acoustic signals than can then be used to define tuning parameter values. This can be particularly useful for energy-based detection in which time and frequency attributes of the target signals are required. Detected signals 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 avoid them. The function ‘filter\_detections’ can be used to remove ambiguous signals (*i.e.* those labeled as split or merged detection) keeping only those that maximize a specific criterium (*i.e.*the highest template correlation). Finally, note that several templates representing the range of variation in signal structure can be used to detect semi-stereotyped signals when running template-based detection (‘template\_detection’ function).

## Discussion

Here we have shown how to evaluate the performance of acoustic signal detection routines using the *ohun* package. 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 signal 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 acoustic signal detection evaluation can simplify comparing performances of different tools and the selection of those better suited to a given system and research question. Note that the tools offered by *ohun* for diagnosing signal detection may not be limited to acoustic data. Any type of automated detection in which the time of occurrence of discrete events needs to be found can be evaluated and optimized by comparing it to a reference annotation, for instance, detection of specific behaviors in video analysis of animal motor activity (*e.g.*, Sturman et al 2020; Hsu & Yttri 2021; DeepEthogram).

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 acoustic signals, the two native methods are relatively simple tools. However, these methods have been widely used by the bioacoustic community and, under the appropriate conditions, can reach adequate performance, as we have shown in our two study cases. Deep learning methods tend to require larger computational power and more complex training routines. This might bring unnecessary difficulties when dealing with less challenging detection tasks. Therefore, the availability of a wide range of approaches is desired in order to simplify finding the most appropriate tool for the intricacies of our study system and research goals. Note that the tools offered in ohun can be used in a pipeline in which detected signals are further classified using more elaborated discrimination algorithms. Acoustic parameters can be used to quantify the structure of the signals and distinguish target from non-target signals.

Detection routines can take a long time when working with large amounts of acoustic data (*e.g.*, large recordings and/or many files). Below we provide some tips that can help make a routine more time-efficient. 1) Always test procedures on small data subsets. Make sure that you are getting 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 signals on low sampling rate files is faster, so we must avoid having Nyquist frequencies much higher than the highest frequency of the target signals. 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 aspects to be considered when aiming to automatically detect acoustic signals. When running automated signal detection programs, try to use your knowledge about the signal structure to determine the initial range. This can be extremely helpful for narrowing down possible parameter values, particularly for energy-based detection. As a general rule, if human observers have a hard time figuring out where a target signal occurs, detection algorithms will also have a hard time. In cases in which occurrences are ambiguous, low performances are expected. Ensure that reference tables contain all target signals and only the desired target signals, otherwise performance optimization can be misleading, as the performance of detection cannot be better than the reference itself. Lastly, avoid having overlapping signals or several signals as a single one (like a multi-syllable vocalization) in the reference table when running an energy-based detector, as they are likely to be identified as separate units.

<https://cran.r-project.org/web/packages/bioacoustics/vignettes/tutorial.html>

## References

Sturman, O., von Ziegler, L., Schläppi, C. et al. Deep learning-based behavioral analysis reaches human accuracy and is capable of outperforming commercial solutions. Neuropsychopharmacol. 45, 1942–1952 (2020). <https://doi.org/10.1038>

Hsu, A.I., Yttri, E.A. B-SOiD, an open-source unsupervised algorithm for identification and fast prediction of behaviors. Nat Commun 12, 5188 (2021). <https://doi.org/10.1038/s41467-021-25420-x>

Stowell, D. (2022). Computational bioacoustics with deep learning: a review and roadmap. *PeerJ*, *10*, e13152. <https://doi.org/10.7717/PEERJ.13152/SUPP-1>

For acknowledgments or funding:

A.R-G. is supported by the Walt Halperin Endowed Professorship and the Washington Research Foundation as Distinguished Investigator.