APPLICATION

callsync: an R package for alignment and analysis of multi-microphone animal recordings

Simeon Q. Smeele [1,2,3,†,*], Stephen A. Tyndel [1,3,†], Barbara C. Klump [1], Gustavo Alarcón-Nieto [1,3,4] & Lucy M. Aplin [1,5,6]

[1] *Cognitive & Cultural Ecology Research Group, Max Planck Institute of Animal Behavior, Radolfzell, Germany*

[2] *Department of Human Behavior, Ecology and Culture, Max Planck Institute for Evolutionary Anthropology, Leipzig, Germany*

[3] *Department of Biology, University of Konstanz, Konstanz, Germany*

[4] *Department of Migration, Max Planck Institute of Animal Behavior, Radolfzell, Germany*

[5] *Division of Ecology and Evolution, Research School of Biology, The Australian National University, Canberra, Australia*

[6] *Department of Evolutionary Biology and Environmental Studies, University of Zurich, Zurich, Switzerland*

[†] Co-first author

[*] Corresponding author. E-mail: ssmeele@ab.mpg.de

Summary

1. To better understand how vocalisations are used during interactions of multiple individuals, studies are increasingly deploying on-board devices with a microphone on each animal. The resulting recordings are challenging to analyse, since microphone clocks drift non-linearly and record the vocalisations of non-focal individuals as well as noise.

2. Here we present `callsync`, an R package designed to align recordings, detect and assign vocalisations to the caller, trace the fundamental frequency, filter out noise and perform basic analysis on the resulting clips.

3. We present a case study where the pipeline is used on a new dataset of six captive cockatiels (*Nymphicus hollandicus*) wearing backpack microphones. Recordings initially had drift of ~2 minutes, but were aligned up to ~2 seconds with our package. We detected and assigned 970 calls across two 3.5 hours recording sessions. We then used a function that traces the fundamental frequency and applied spectrographic cross correlation to show a possible analytical pipeline where vocal similarity is visually assessed.

4. The `callsync` package can be used to go from raw recordings to a clean dataset of features. The package is designed to be modular and allows users to replace functions as they wish. We also discuss the challenges that might be faced in each step and how the available literature can provide alternatives for each step.

Keywords: communication networks, bio acoustics, microphone alignment, recording processing

Introduction

The study of vocal signals in animals is a critical tool for understanding the evolution of vocal communication (Endler 1993). Vocalisations commonly occur in group contexts, where they may function to signal movement or identity, and mediate interactions. However the problem of assigning identity has led most previous work to focus on long-range calls or on vocalisations made when alone, e.g., territorial calls and song, and to focus on recording one individual at a time. Yet only by studying the ways that animals communicate in 'real time' can allow us to untangle the complicated dynamics of how group members signal one another (Gill et al. 2015). For example, the context of how individuals address members of their social network (Cheney and Seyfarth 2018), and the ensuing call and response-dynamics (Araya-Salas et al. 2020), can only be studied by recording multiple individuals simultaneously. These communication networks can help us understand how animals coordinate call-response with movement (Demartsev et al. 2022) as well the function of vocal imitation (Dahlin et al. 2014; Knörnschild et al. 2012; Nousek et al. 2006).

Recent innovations in recording technologies have allowed for a dramatic increase of fine scale on-animal bioacoustic data collection (Wild et al. 2022; Bravo Sanchez et al. 2021; Gill et al. 2016), allowing multiple individuals to be recorded simultaneously. However, as the capability of placing small recording devices on animals increases, so does the need for tools to process the resulting data streams. Several publicly available R packages exist that measure acoustic parameters from single audio tracks (seewave: Sueur, Aubin, and Simonis (2008), tuneR: Ligges et al. (2022), WarbleR: Araya-Salas and Smith-Vidaurre (2017)), but to our knowledge, none address the critical issue of microphone clock drift and the ability to align and process multiple recordings. This poses a serious issue for those studying communication networks of multiple tagged individuals. In this paper, we apply a new R package, `callsync`, that aligns multiple misaligned audio files, detects vocalisations, assigns these to the focal individual and provides an analytical pipeline for the resulting synchronised data.

The primary target for use of this package are researchers that study animal communication systems within groups. As researchers deploy multiple microphones that record simultaneously, the resulting clock drift can prove to be a barrier in further data processing steps (Schmid et al. 2010). To make matters worse this drift is often non-linear (Anisimov et al. 2014). Thus, if several microphone recorders (whales; Miller and Dawson (2009), Hayes et al. (2000), bats; Stidsholt et al. (2019)) are placed on animals, it is critical for researchers to be able to line up all tracks so that calls can be assigned correctly to the focal individual (loudest track). The main functionality of `callsync` is to align audio tracks, detect calls from each track, determine which individual (even ones in relatively close proximity to one another) is vocalising and segment (detects start and end of signal as in detection module) them, as well as take measurements of the given calls (see Figure 1). As `callsync` takes a modular approach to aligning, detecting, and analysing audio tracks, researchers can use only the components of the package that suit their needs.

Current research packages that implement call alignment strategies are either used in matlab (Malinka et al. 2020; Anisimov et al. 2014) or c++ (Gill et al. 2015). However these tools have not, up to now, been adapted for the R environment, a popular programming language among many animal behaviour and bioacoustic researchers. Many of these tools are not documented publicly nor open source, and can require high licensing fees (i.e., Matlab). While the design of our package is best suited to contexts where all microphones exist in the same spatial area, it is the goal that it can be adapted to other contexts. 'callsync' is publicly available on CRAN and GitHub, is beginner friendly with strong documentation, and does not require extensive programming background. This open source tool will allow researchers to expand the study of bioacoustics and solve an issue that impedes detailed analysis of group-level calls.

Case study: cockatiel vocalisations

We present a case study to show how `callsync` functions can be included in a workflow (see Figure 1). We used a dataset of domestic cockatiels (*Nymphicus hollandicus*). These birds are part of an ongoing study at the Max Planck Institute of Animal Behavior in Radolfzell, Germany. 30 birds were housed in five groups of six individuals, with each group of six housed separately in a 4x3x2.7m aviary facility. Each bird was fitted with a TS-systems EDIC-Mini E77 tag inside a sewn nylon backpack fitted via Teflon harness around the wings, with the total weight of all components under 7% of body weight (weight range of birds 85-120g). Audio recordings were scheduled to record for 4 hours per day. Each microphone was automatically programmed to turn on and off daily at the same time. For the purposes of demonstration, two full recording sessions (4 hours each) were selected for processing where the microphones were scheduled to record from 6:30 until 10:30 in the morning (July 15th and 16th 2021). Seven days after deployment, microphone recorders were removed and downloaded as .wav files directly onto a computer from the tag, according to the manufacturer protocols. Data from each microphone were placed into the appropriate folder (see workflow instructions) and processed in accordance with our package workflow.

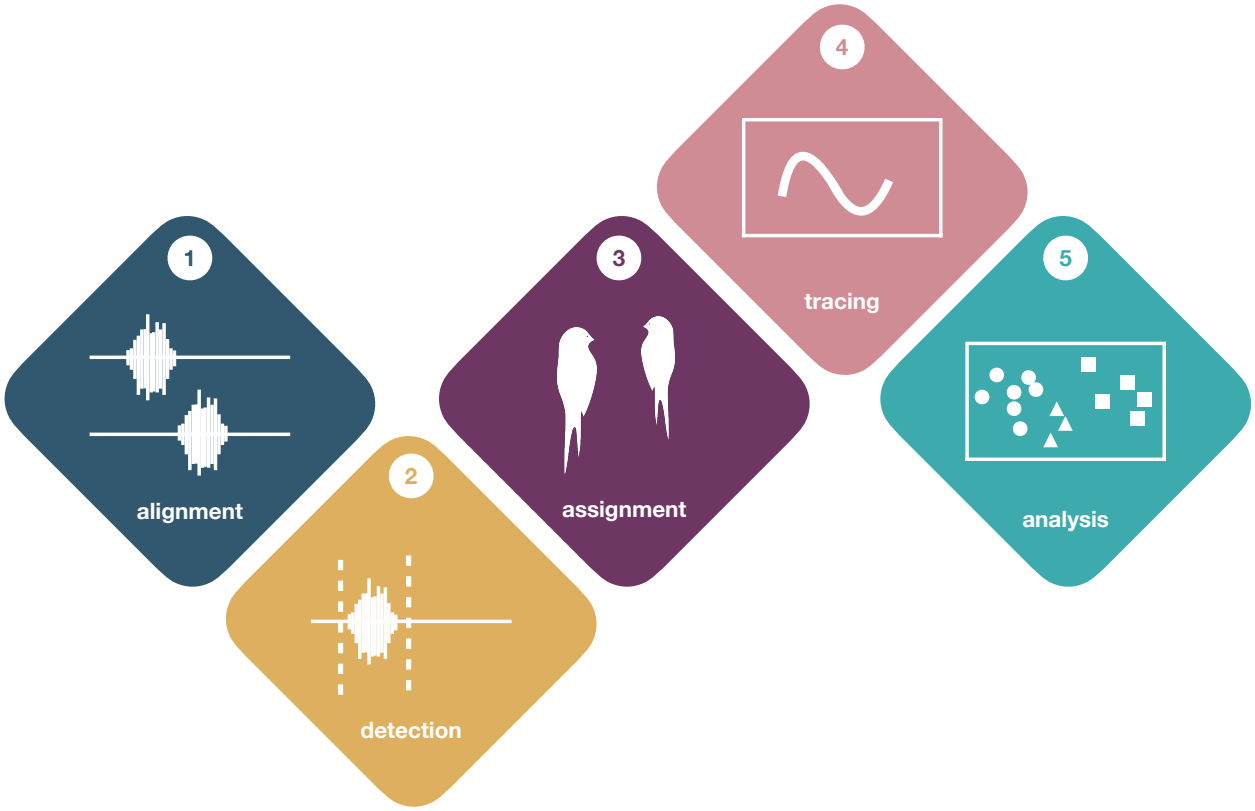

Figure 1: Flowchart from the `callsync` package. The *alignment* module can be used to align multiple microphones that have non-linear temporal drift. The *detection* module can be used to detect vocalisations in each recording. Filters can be applied to remove false alarms in the detection module. The *assignment* module can be used to assign a vocalisation to the focal individual, making sure that vocalisations from conspecifics are excluded from the focal recording. The *tracing* module can be used to trace and analyse the fundamental frequency for each vocalisation. The final *analysis* module can be used to run spectrographic cross correlation and create a feature vector to compare across recordings.

Installation and set-up

The package call be installed from CRAN running `install.packages(callsync)` or a developmental version can be installed from GitHub (https://github.com/simeonqs/callsync):

```
install.packages('devtools')
library(devtools)
devtools::install_github('simeonqs/callsync')
```

While all underlying code and functions from the CRAN repository can be found in the aforementioned github repository, we have created a seperate repository from which readers can follow the code from the case study: https://github.com/simeonqs/callsync_an_R_package_for_alignment_and_analysis_of_multi-microphone_animal_recordings. All required packages are also automatically installed and loaded for the case study when running the `00_set_up.R` script from this repository.

Alignment of raw recordings

The general goal of the alignment step is to shift unaligned recordings to a degree that will allow vocalisations to be processed further. While researchers' requirements on alignment precision will vary, the output of this function should allow researchers to be able to identify that vocalisations picked up across all microphones are identified as the same call off of every microphone. It should be noted that recordings are still not perfectly aligned, since clock-drift occurs within the chunks.

The raw recordings consisted of two four hour long wav files for six cockatiels (i.e., 12 files total). The backpack microphones have internal clocks that sync with the computer clock when connected via USB and automatically turn recordings on and off.

However, these clocks drift in time both during the off period, creating start times that differ up to a few minutes, and also during the recording period, causing additional variable drift up to a minute in four hours of recording for the microphone model we used. The function `align` can be used as a first step to align the audio recordings. To accurately align these tracks, the full audio files were automatically split into 15 minute chunks of recording to ensure that drift was reduced to mere seconds. This value can be adjusted depending on the amount of drift. The function selects one recording (focal recording) and aligns all the other recordings relative to the selected one recording using cross correlation on the energy content (summed absolute amplitude) per time bin (function step_size), in our case 0.5 seconds was used.This value can also be adjusted, lower values allow for greater precision but higher computational load. The call detection and assignment step includes fine alignment and we therefore accepted a potential error of 0.5 seconds. There is also an option to store a log file that records the offset for each chunk.

```
align(chunk_size = 15,                          # how long should the chunks be in minutes
      step_size = 0.5,                          # bin size for summing in seconds
      path_recordings = 'ANALYSIS/DATA',        # where raw data is stored
      path_chunks = 'ANALYSIS/RESULTS/chunks',  # where to store the chunks
      keys_rec = c('_\\(', '\\)_'),             # how to recognise the recording in the path
      keys_id = c('ASWMUX', '.wav'),            # how to recognise the individual/microphone in the path
      blank = 15,                               # how many minutes should be discarded before/after detection
      wing = 10,                                # how much extra should be loaded for alignment in minutes
      save_pdf = TRUE,                          # should a pdf be saved
      save_log = TRUE)                          # should a csv file with alignment times be saved
```

For cross correlation, `align` loads the chunks with additional minutes before and after (option `wing`) to ensure that overlap can be found. The cross correlation is performed using the function `simple.cc`, which takes two vectors (the binned energy content of two recordings) and calculates the absolute difference while sliding the two vectors over each other. It returns the position of minimum summed difference, or in other words, the position of maximal overlap. This position is then used to align the recordings relative to the first recording and save chunks that are maximally aligned. Note that due to drift during the recording, the start and end times might still be seconds off; it is the overall alignment of the chunks that is optimised. The function also allows the user to create a pdf with waveforms of each individual recording and a single page per chunk (Figure 2), to visually verify if alignment was successful. For our dataset all chunks aligned correctly without a filter. If this is not the case the option `ffilter_from` can be set to apply a high-pass filter to improve alignment. Mis-aligned chunks can also be rerun individually using the chunk size argument (function chunk_seq) to avoid re-running the entire dataset.

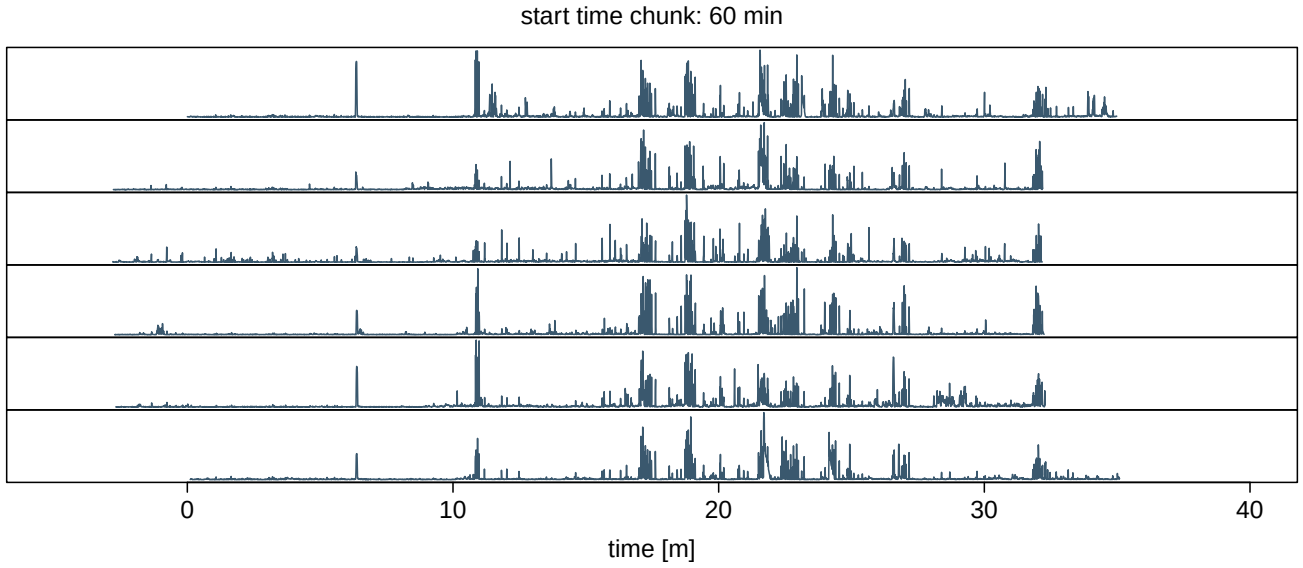

Figure 2: Example of the alignment output. Olive-coloured lines represent the summed absolute amplitude per bin (= 0.5 seconds). Recordings are aligned relative to the first recording (which starts at 0). Note that recordings 2-5 initially started ~2 minutes earlier (purple arrow), but are now aligned. The title displays the start time of the chunk in the raw recording.

Call detection and assignment

The next step is to detect calls in each set of chunks and assign them to the correct source individual. The `detect.and.assign` function wrapper loads the chunks using the function `load.wave` where it optionally applies a high-pass filter to reduce the amount of low frequency noise. To detect calls, 'detect.and.assign' first calls the function `call.detect.multiple`, which is used to detect multiple calls in an R wave object. It first applies the `env` function from the *seewave* (Sueur, Aubin, and Simonis 2008) package to create a smooth Hilbert amplitude envelope. It then detects all the points on the envelope which are above a certain threshold relative to the maximum of the envelope. After removing detections that are shorter than a set minimum duration (function min_dur) or longer than a set maximum (function max_dur) it returns all the start and end times as a data frame.

Because the microphones on focal individuals are very likely to record the calls of the non-focal individuals as well, we implemented a step that assigns the detected calls to the individual emitting the sound, based on amplitude. For this, `detect.and.assign` subsequently calls the function `call.assign`, which runs through all the detections in a given chunk for a given individual (i.e., output of 'call.detect.multiple') and runs the `call.detect` function to more precisely determine the start and end time of the call. It then ensures that minor temporal drift after the align function is corrected by rerunning the `simple.cc` function pairwise between the focal recording and all others. After alignment it calculates the summed absolute energy content on all recordings for the time frame when the call was detected and compares the recording where the detection was made to all other recordings. If the loudest recording is louder by a set percentage (this value can be adjustable according to the researchers needs) than the second loudest recording, the detection is saved as a separate wav file. If not, this means it's not possible to determine the focal individual and the detection is discarded (there is an option to save all detections before the assignment step). The function also allows the user to create a pdf file with all the detections (see Figure 3 for a short example) to manually inspect the results.

```
detect.and.assign(path_chunks = 'ANALYSIS/RESULTS/chunks', # where to read the chunks
                  path_calls = 'ANALYSIS/RESULTS/calls',   # where to store the calls
                  ffilter_from = 1100,                     # from where to filter in Hz
                  threshold = 0.18,                        # fraction of maximum of envelope for detection
                  msmooth = c(1000, 95),                   # smoothening argument for `env`
                  min_dur = 0.1,                           # minimum duration in seconds for acceptance
                  max_dur = 0.3,                           # maximum duration in seconds for acceptance
                  step_size = 1/50,                        # bin size for summing in seconds
                  wing = 10,                               # how many extra seconds to load for alignment
                  save_files = TRUE,                       # should the files be stored in path_calls
                  save_extra = 0.05)                       # save 0.05 seconds before and after detection
```

For the 8 hour long cockatiel dataset, the function detected and assigned 1780 events, 970 of which were retained as vocalisations (see next step). The removed events were primarily noise and noisy calls (e.g. overlapping calls and calls with echo). To estimate the accuracy of the function we manually detected and assigned IDs for 192 calls in three chunks (chosen due to high vocal activity), ran the same filtering step as `callsync` functions described earlier to get rid of 41 noisy examples, and compared the performance of the `detect.and.assign` function to manually labelled data. The ground truth was performed using the function `calc.perf`. This function simply determines whether the detected calls (and subsequent labelled ids) of two datasets match. If sufficient temporal overlap exists between the two datasets, they are determined to match. The false positive rate was <1% (single false detection) and the true positive rate was 89%. The remaining 11% (false negatives) were mostly calls that were too quiet to be picked up by our set amplitude threshold.

Analysis of single calls and call comparison

To analyse the calls, the chunks were loaded and the function `call.detect` was rerun to determine the start and end times of the call. The wave objects were then resized to only include the call (`new_wave`). To trace the fundamental frequency we applied the `trace.fund` function to the resized wave objects. This function detects the fundamental frequency in a sliding window based on the spectrum and a relative threshold. We ran the latter step in parallel using the function `mclapply` from the base R package *parallel* (see Figure 4a for an example).

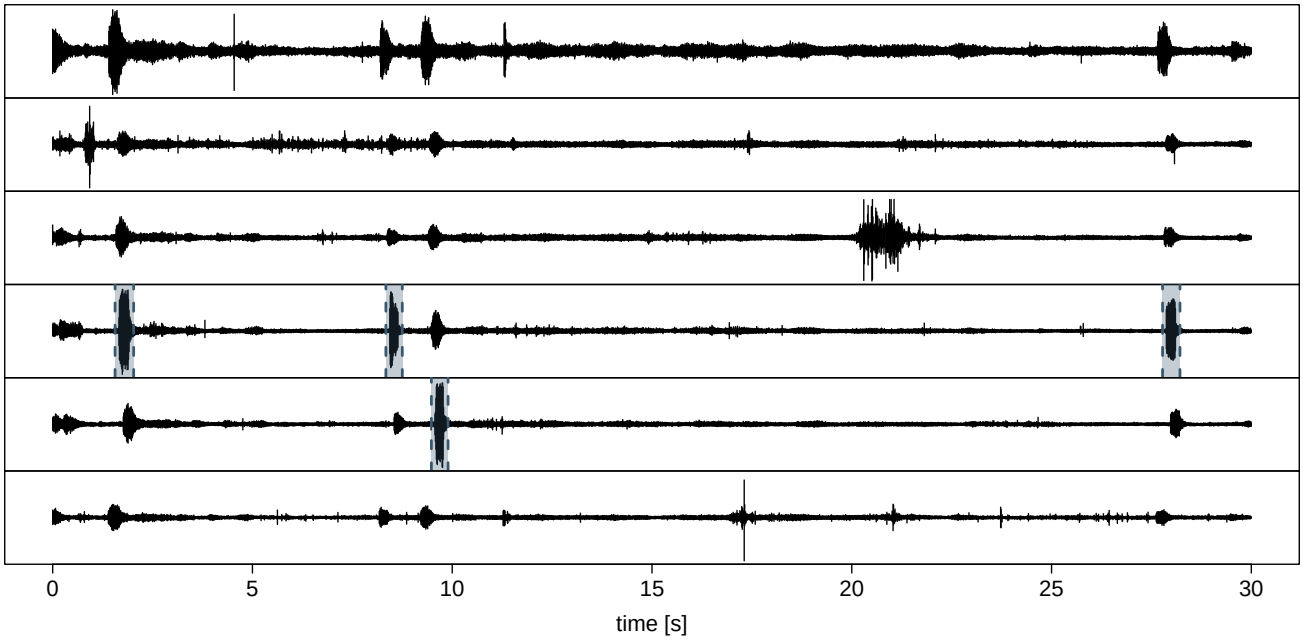

Figure 3: Example of the detection output. Black lines are the waveforms. Olive-coloured dashed lines with shaded areas in between are the detected calls. Note that only the loudest version of each call is selected. The first call is also loud in the first channel (top-row), but contains less energy throughout the whole call (channel four is loud for a longer duration). Animals with individual specific-sounds (scratching, etc) are seen clearly on only single microphones.

```
traces = mclapply(new_waves, function(new_wave)  # apply the function to each new_wave
  trace.fund(wave = new_wave,                    # use the new_wave
             spar = 0.3,                         # smoothing argument for the `smooth.spline` function
             freq_lim = c(1.2, 3.5),             # only consider trace between 1.2 and 3.5 Hz
             thr = 0.15,                         # threshold for detection, fraction of max of spectrum
             hop = 5,                            # skip five samples per step
             noise_factor = 1.5),                # only accept if trace is 1.5 times greater than noise
  mc.cores = 4)                                  # run on four threads, has to be 1 on Windows
```

Since the call detection step also picks up on a lot of noise (birds scratching, flying, walking around) as well as calls, we ran a final step to filter the measurements and traces before these were saved.

```
measurements = measure.trace.multiple(traces = traces,              # object containing traces
                                       new_waves = new_waves,       # object containing adjusted waves
                                       waves = waves,               # object containing waves
                                       detections = detections,     # object containing detections
                                       path_pdf = path_pdf_traces)  # where to store pdf
keep = measurements$prop_missing_trace < 0.1 &                      # max 10% missing points
  measurements$signal_to_noise > 6 &                                # signal to noise at least 6
  measurements$band_hz > 400 &                                      # bandwidth at least 400 Hz
measurements = measurements[keep,]                                 # keep only these measurements
traces = traces[keep]                                               # and these traces
```

A frequently used method to compare calls is to measure their similarity using spectrographic cross correlation (SPCC) (Cortopassi and Bradbury 2000), where two spectrograms are slid over each other and the pixelwise difference is computed for each step. At the point where the signals maximally overlap one will find the minimal difference. This score is then used as a measure of acoustic distance between two calls. The function `run.spcc` runs SPCC and includes several methods to reduce noise in the spectrogram before running cross correlation (for an example see Figure 4b). To visualise the resulting feature vector from running SPCC on the cockatiel calls we used uniform manifold approximation and projection (Konopka 2022) which projects

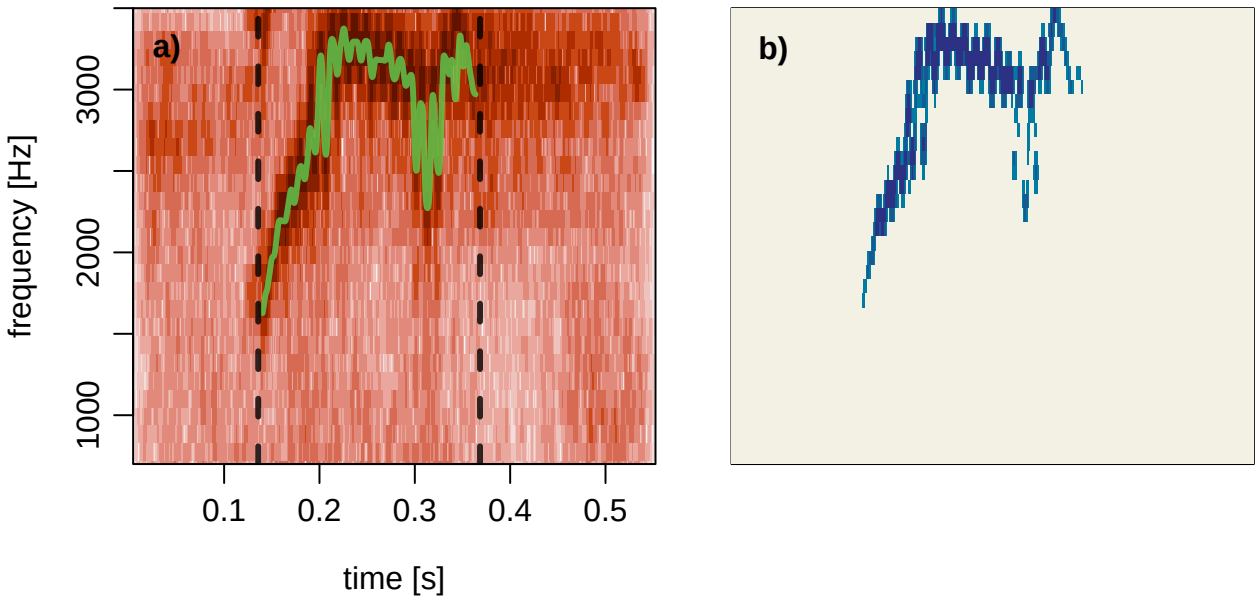

Figure 4: a) Spectrogram of a cockatiel call with start and end (black dashed lines) and the fundamental frequency trace (green solid line). b) Noise reduced spectrogram where darker colours indicate higher intensity.

the results in two dimensional space. Specific details on umap can be found in [Konopka (2022)}. Calls cluster somewhat by individual, but there is also a lot of overlap between individuals (see Figure 5). Further exploration of the results is beyond the scope of this paper, but generally overlapping clusters is common in parrot literature (citation). We demonstrate the results, to show how one might further explore the resulting distance matrices from step 4.

Discussion

Here we detail our R package 'callsync', designed to take raw microphone recordings collected simultaneously from multiple individuals and align, extract and analyse their calls. We present a case study and workflow to demonstrate `callsync` in action on a dataset of 8 hours of recordings from 6 communally housed captive cockatiels. Each of the modular components (alignment, detection, assignment, tracing, and analysis) successfully achieved the stated goals in the cockatiel system. Misaligned audio tracks were accurately aligned in a first step (see Figure 2), calls were correctly identified in the aligned recordings (see Figure 3), the individual making the call was selected (see Figure 3), and downstream data analysis was performed (figures 4 and 5). `callsync` can perform alignment even on inter-microphone drift that lasts minutes as well as handle unpredictable and non-linear drift patterns on different microphones.

With tracks aligned to only a single second and only one false positive, we are confident that `callsync` is a robust and useful tool for bioacoustics research. Additionally, the one false positive was actually a true positive; in this case because of a tiny difference in start and end time between the ground truth and automatic detection, which led the ground truth to be filtered out, but the automatic detection to stay. Overall, the true positive rate of our results was 89%, meaning that only 11% of the manually selected calls for ground truthing were not detected by the `call.detect` function. Where call rate across call types is important, researchers can set the threshold very low and manually remove false positives. Alternatively a deep neural network can be used to sort signal from noise (Bergler et al. 2022).

A further possible challenge with the `call.detect` function could be that certain call types are never easily distinguishable from background noise. In these situations, `call.detect` is likely to pick up a significant amount of background noise in addition to calls. Function parameters can be adapted and should function on most call-types, as can post-processing thresholding. For example, machine learning approaches (Bergler et al. 2022; Cohen et al. 2022; Stowell et al. 2019) or image recognition tools (Smith-Vidaurre, Araya-Salas, and Wright 2020; Valletta et al. 2017) can be later applied to separate additionally detected noise. As well, in specific cases (e.g., low amplitude calls), once the align function is performed, the entire `call.detect` function can be swapped out for deep learning call detection algorithms such as ANIMAL-SPOT (Bergler et al. 2022).

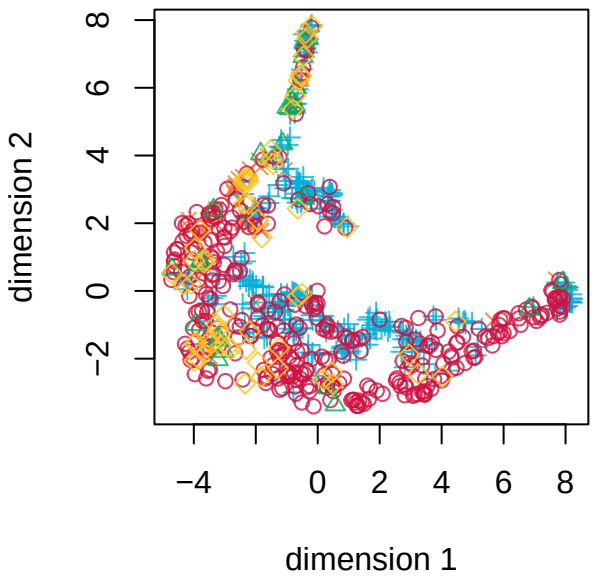

Figure 5: Call distribution in uniform manifold approximation and projection space. Dots represents calls and are coloured by individual.

The microphones used in this case study were implemented in a captive setting where all calls were within hearing range of each other and each microphone. Thus, all microphones contained a partially shared noisescape. Despite their proximity to one another, it should be noted that each microphone still contained unique vocal attributes, such as wing beats and scratching, and yet the major alignment step still aligned all chunks correctly. It is possible that researchers find that the noise differences between microphones is too high for the first alignment step to perform adequately. This would be particularly salient in field settings where individuals within fission-fusion groups find themselves in the proximity of other group members only some of the time (Furmankiewicz et al. 2011; Balsby and Bradbury 2009; Buhrman-Deever, Hobson, and Hobson 2008), or in situations where animals constantly move (i.e., flying) or do independent behaviours that other group members do not (e.g., preening) (Demartsev et al. 2022). Researchers will have to assess their own dataset and test this package to determine whether the first step will perform well on their dataset. If it does not, `callsync` is a modular package and other approaches, such as deep learning (O'shea and West 2016) can be used instead of the first align function, while still using other components of the pipeline.

Alternatively, assigning a focal bird may be more difficult if individuals tend to be in very close proximity to other group members. If the group members spend time too close to one another (in our case less than 5-10 cm, the distance between the backpack and head of the focal individual) (Kloepper and Kinniry 2018; Boughman and Wilkinson 1998), it will become more difficult to distinguish the calling bird from other surrounding individuals. In the context of this case study, it was sufficient to only select a focal bird when the second loudest call was at least 5% quieter than the focal call. This threshold can be adapted if needed, or the assignment part can be removed entirely if so chosen. More generally, the issue of variation in spatial proximity could be solved by including other on-board or group-level tools, such as accelerometers (Gill et al. 2015; Anisimov et al. 2014), or proximity tags (Wild et al. 2022). This adds extra weight and cost to on-board devices, and so if not possible, the other analytical tools could be considered to distinguish the focal calling birds. The specific tool will depend heavily on use-cases, but include usingdiscriminant analysis (McIlraith and Card 1997) or cepstral coefficients (Lee, Lee, and Huang 2006) to predict the calling ID.

Lastly, both fundamental frequency tracing and SPCC work well in certain contexts. For example, automatic fundamental frequency traces work best for tonal calls while SPCC works best when the signal to noise ratio is sufficiently low (Cortopassi and Bradbury 2000). However, if these criteria are not met, several other tools can be used to manually trace fundamental frequency instead, such as Luscinia (Lachlan, Ratmann, and Nowicki 2018) or manual tracing (Araya-Salas and Smith-Vidaurre 2017). While it is important to consider all possible limitations of `callsync` it should also be noted that there are few tools that exist that perform this much needed task. Indeed, the fine scale alignment step of `callsync` allows for call and response

dynamics to be measured regardless of how close the calls are to one another. While this paper has thus far only addressed on-board microphones, other systems that implement passive acoustic monitoring systems (Thode et al. 2006) and microphone arrays (Blumstein et al. 2011) should also find benefit within this package depending on the set-up and degree of drift. Possible future research opportunities include trying to incorporate machine learning and noise reduction techniques so that the major alignment can perform in all contexts.

Conclusion

This open source package is publicly available on GitHub and CRAN. We welcome all continued suggestions and believe that our package will result in an increase in the possibilities and amount of bioacoustic research. Our package provides functions that allow alignment, detection, assignment, tracing and analysis of calls in a multi-recorder setting where all microphones are within acoustic distance. The package can be used to generate a fully automated pipeline from raw recordings to the final feature vectors. We show that such a pipeline works well on a captive dataset with 4 hour long recordings from backpack microphones on six cockatiels which experience non-linear time drift up to several minutes. Each module can also be replaced with alternatives and can be further developed. This package is, to our knowledge, the first R package that performs this task. We hope this package expands the amount of data researchers can process and contributes to understanding the dynamics of animal communication.

Acknowledgements

We thank Dr. Ariana Strandburg-Peshkin for early feedback. SQS and SAT received funding from the International Max Planck Research School for Organismal Biology and the International Max Planck Research School for Quantitative Behaviour, Ecology and Evolution. SAT received additional funding from a DAAD PhD fellowship. LMA was funded by a Max Planck Group Leader Fellowship from the Max Planck Society.

Ethics statement

Data from cockatiels were provided by LMA, SAT, and BCK under ethical permission from Regierungspräsidium Freiburg Az. 35-9185.81/G-18/009 within a larger study investigating social learning in cockatiels.

Data accessibility

All code associated with this paper is publicly available on GitHub (https://github.com/simeonqs/callsync_an_R_package_for_alignment_and_analysis_of_multi-microphone_animal_recordings). All data and code is also publicly available on Edmond (https://edmond.mpdl.mpg.de/privateurl.xhtml?token=05b4aa17-a9bd-44ac-b62b-bdf620aceebb). A DOI for the Edmond repository will be added when the manuscript is accepted and assigned a DOI. The `callsync` package can be installed from CRAN and a developmental version can be found on GitHub (https://github.com/simeonqs/callsync).

Competing interests

All authors declare they have no competing interests.

Author contributions

SQS: Conceptualization, Data Curation, Formal Analysis, Methodology, Project Administration, Software, Validation, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing; SAT: Conceptualization, Data Curation, Investigation,

Methodology, Project Administration, Validation, Writing – Original Draft Preparation, Writing – Review & Editing; BCK: Investigation, Supervision, Writing – Review & Editing; GAN: Investigation, Methodology, Writing – Review & Editing; LMA: Funding Acquisition, Resources, Supervision, Writing – Review & Editing.

References

Anisimov, Victor N, Joshua A Herbst, Andrei N Abramchuk, Alexander V Latanov, Richard HR Hahnloser, and Alexei L Vyssotski. 2014. "Reconstruction of Vocal Interactions in a Group of Small Songbirds." *Nature Methods* 11 (11): 1135–37.

Araya-Salas, Marcelo, H Andrés Hernández-Pinsón, Nazareth Rojas, and Gloriana Chaverri. 2020. "Ontogeny of an Interactive Call-and-Response System in Spix's Disc-Winged Bats." *Animal Behaviour* 166: 233–45.

Araya-Salas, Marcelo, and Grace Smith-Vidaurre. 2017. "warbleR: An r Package to Streamline Analysis of Animal Acoustic Signals." *Methods in Ecology and Evolution* 8 (2): 184–91.

Balsby, Thorsten JS, and Jack W Bradbury. 2009. "Vocal Matching by Orange-Fronted Conures (Aratinga Canicularis)." *Behavioural Processes* 82 (2): 133–39.

Bergler, Christian, Simeon Q Smeele, Stephen A Tyndel, Alexander Barnhill, Sara T Ortiz, Ammie K Kalan, Rachael Xi Cheng, et al. 2022. "ANIMAL-SPOT Enables Animal-Independent Signal Detection and Classification Using Deep Learning." *Scientific Reports* 12 (1): 1–16.

Blumstein, Daniel T, Daniel J Mennill, Patrick Clemins, Lewis Girod, Kung Yao, Gail Patricelli, Jill L Deppe, et al. 2011. "Acoustic Monitoring in Terrestrial Environments Using Microphone Arrays: Applications, Technological Considerations and Prospectus." *Journal of Applied Ecology* 48 (3): 758–67.

Boughman, Janette Wenrick, and Gerald S Wilkinson. 1998. "Greater Spear-Nosed Bats Discriminate Group Mates by Vocalizations." *Animal Behaviour* 55 (6): 1717–32.

Bravo Sanchez, Francisco J, Md Rahat Hossain, Nathan B English, and Steven T Moore. 2021. "Bioacoustic Classification of Avian Calls from Raw Sound Waveforms with an Open-Source Deep Learning Architecture." *Scientific Reports* 11 (1): 1–12.

Buhrman-Deever, Susannah C, Elizabeth A Hobson, and Aaron D Hobson. 2008. "Individual Recognition and Selective Response to Contact Calls in Foraging Brown-Throated Conures, Aratinga Pertinax." *Animal Behaviour* 76 (5): 1715–25.

Cheney, Dorothy L, and Robert M Seyfarth. 2018. "Flexible Usage and Social Function in Primate Vocalizations." *Proceedings of the National Academy of Sciences* 115 (9): 1974–79.

Cohen, Yarden, David Aaron Nicholson, Alexa Sanchioni, Emily K Mallaber, Viktoriya Skidanova, and Timothy J Gardner. 2022. "Automated Annotation of Birdsong with a Neural Network That Segments Spectrograms." *Elife* 11: e63853.

Cortopassi, Kathryn A, and Jack W Bradbury. 2000. "The Comparison of Harmonically Rich Sounds Using Spectrographic Cross-Correlation and Principal Coordinates Analysis." *Bioacoustics* 11 (2): 89–127.

Dahlin, Christine R, Anna M Young, Breanne Cordier, Roger Mundry, and Timothy F Wright. 2014. "A Test of Multiple Hypotheses for the Function of Call Sharing in Female Budgerigars, Melopsittacus Undulatus." *Behavioral Ecology and Sociobiology* 68 (1): 145–61.

Demartsev, Vlad, Andrew S Gersick, Frants H Jensen, Mara Thomas, Marie A Roch, Marta B Manser, and Ariana Strandburg-Peshkin. 2022. "Signalling in Groups: New Tools for the Integration of Animal Communication and Collective Movement." *Methods in Ecology and Evolution*.

Endler, John A. 1993. "Some General Comments on the Evolution and Design of Animal Communication Systems." *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences* 340 (1292): 215–25.

Furmankiewicz, Joanna, Ireneusz Ruczyński, Radosław Urban, and Gareth Jones. 2011. "Social Calls Provide Tree-Dwelling Bats with Information about the Location of Conspecifics at Roosts." *Ethology* 117 (6): 480–89.

Gill, Lisa F, Pietro B D'Amelio, Nicolas M Adreani, Hannes Sagunsky, Manfred C Gahr, and Andries Ter Maat. 2016. "A Minimum-Impact, Flexible Tool to Study Vocal Communication of Small Animals with Precise Individual-Level Resolution." *Methods in Ecology and Evolution* 7 (11): 1349–58.

Gill, Lisa F, Wolfgang Goymann, Andries Ter Maat, and Manfred Gahr. 2015. "Patterns of Call Communication Between Group-Housed Zebra Finches Change During the Breeding Cycle." *Elife* 4: e07770.

Hayes, Sean A, David K Mellinger, Donald A Croll, Daniel P Costa, and J Fabrizio Borsani. 2000. "An Inexpensive Passive Acoustic System for Recording and Localizing Wild Animal Sounds." *The Journal of the Acoustical Society of America* 107 (6): 3552–55.

Kloepper, Laura N, and Morgan Kinniry. 2018. "Recording Animal Vocalizations from a UAV: Bat Echolocation During Roost Re-Entry." *Scientific Reports* 8 (1): 1–6.

Knörnschild, Mirjam, Martina Nagy, Markus Metz, Frieder Mayer, and Otto von Helversen. 2012. "Learned Vocal Group Signatures in the Polygynous Bat Saccopteryx Bilineata." *Animal Behaviour* 84 (4): 761–69.

Konopka, Tomasz. 2022. *Umap: Uniform Manifold Approximation and Projection*. https://CRAN.R-project.org/package=umap.

Lachlan, Robert F, Oliver Ratmann, and Stephen Nowicki. 2018. “Cultural Conformity Generates Extremely Stable Traditions in Bird Song.” *Nature Communications* 9 (1): 1–9.

Lee, Chang-Hsing, Yeuan-Kuen Lee, and Ren-Zhuang Huang. 2006. “Automatic Recognition of Bird Songs Using Cepstral Coefficients.” *Journal of Information Technology and Applications (□ □ □ □ □ □ □ □ □)* 1 (1): 17–23.

Ligges, Uwe, Andrea Preusser, Anita Thieler, Johanna Mielke, Claus Weihs, et al. 2022. “Package ‘tuneR’.”

Malinka, Chloe E, John Atkins, Mark P Johnson, Pernille Tønnesen, Charlotte A Dunn, Diane E Claridge, Natacha Aguilar de Soto, and Peter Teglberg Madsen. 2020. “An Autonomous Hydrophone Array to Study the Acoustic Ecology of Deep-Water Toothed Whales.” *Deep Sea Research Part I: Oceanographic Research Papers* 158: 103233.

McIlraith, Alex L, and Howard C Card. 1997. “Birdsong Recognition Using Backpropagation and Multivariate Statistics.” *IEEE Transactions on Signal Processing* 45 (11): 2740–48.

Miller, B, and S Dawson. 2009. “A Large-Aperture Low-Cost Hydrophone Array for Tracking Whales from Small Boats.” *The Journal of the Acoustical Society of America* 126 (5): 2248–56.

Nousek, Anna E, Peter JB Slater, Chao Wang, and Patrick JO Miller. 2006. “The Influence of Social Affiliation on Individual Vocal Signatures of Northern Resident Killer Whales (Orcinus Orca).” *Biology Letters* 2 (4): 481–84.

O’shea, Timothy J, and Nathan West. 2016. “Radio Machine Learning Dataset Generation with Gnu Radio.” In *Proceedings of the GNU Radio Conference*. Vol. 1. 1.

Schmid, Thomas, Roy Shea, Zainul Charbiwala, Jonathan Friedman, Mani B Srivastava, and Young H Cho. 2010. “On the Interaction of Clocks, Power, and Synchronization in Duty-Cycled Embedded Sensor Nodes.” *ACM Transactions on Sensor Networks (TOSN)* 7 (3): 1–19.

Smith-Vidaurre, Grace, Marcelo Araya-Salas, and Timothy F Wright. 2020. “Individual Signatures Outweigh Social Group Identity in Contact Calls of a Communally Nesting Parrot.” *Behavioral Ecology* 31 (2): 448–58.

Stidsholt, Laura, Mark Johnson, Kristian Beedholm, Lasse Jakobsen, Kathrin Kugler, Signe Brinkløv, Angeles Salles, Cynthia F Moss, and Peter Teglberg Madsen. 2019. “A 2.6-g Sound and Movement Tag for Studying the Acoustic Scene and Kinematics of Echolocating Bats.” *Methods in Ecology and Evolution* 10 (1): 48–58.

Stowell, Dan, Michael D Wood, Hanna Pamuła, Yannis Stylianou, and Hervé Glotin. 2019. “Automatic Acoustic Detection of Birds Through Deep Learning: The First Bird Audio Detection Challenge.” *Methods in Ecology and Evolution* 10 (3): 368–80.

Sueur, Jérôme, Thierry Aubin, and Caroline Simonis. 2008. “Seewave, a Free Modular Tool for Sound Analysis and Synthesis.” *Bioacoustics* 18 (2): 213–26.

Thode, Aaron M, Peter Gerstoft, William C Burgess, Karim G Sabra, Melania Guerra, M Dale Stokes, Michael Noad, and Douglas H Cato. 2006. “A Portable Matched-Field Processing System Using Passive Acoustic Time Synchronization.” *IEEE Journal of Oceanic Engineering* 31 (3): 696–710.

Valletta, John Joseph, Colin Torney, Michael Kings, Alex Thornton, and Joah Madden. 2017. “Applications of Machine Learning in Animal Behaviour Studies.” *Animal Behaviour* 124: 203–20.

Wild, Timm A, Martin Wikelski, Stephen Tyndel, Gustavo Alarcón-Nieto, Barbara C Klump, Lucy M Aplin, Mirko Meboldt, and Hannah J Williams. 2022. “Internet on Animals: Wi-Fi-Enabled Devices Provide a Solution for Big Data Transmission in Biologging.” *Methods in Ecology and Evolution*.