APPLICATION

Analysis of backpack microphone recordings using *callsync*

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

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

[†] Co-first author

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

Summary

1. To better understand how vocalisations are used during interactions of multiple individuals captive studies with microphones on the animal are often performed. The resulting recordings are challenging to analyse, since microphones 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, 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 829 calls across two days of 3.5 hours of recording each. We also use function that trace the fundamental frequency and apply spectrographic cross correlation to show that calls coming from the same individual sound more similar.
4. The `callsync` package can be used to go from raw recordings to a cleaned dataset of features. The package is designed to be modular and allow 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 segmentation

Introduction

The study of vocal signals in animals is a critical tool for understanding the evolution of vocal communication (Endler 1993). Recent innovations in on-animal recording technologies have allowed for a dramatic increase of fine scale bioacoustic data collection and researchGill et al. (2016). Studying the ways that animals communicate in 'real time' allows us to untangle the complicated dynamics of how group members signal one another (Gill et al. 2015). These communication networks can help us understand how animals coordinate call response with movement (Demartsev et al. 2022) as well as how group signatures form Knörnschild et al. (2012). However, as the capability of placing small recording devices on animals increases, so too 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. The critical issue is that multiple microphones recording simultaneously can drift apart in time (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 them, as well as take measurements of the given calls (see Figure 1). However, `callsync` takes a modular approach to aligning, segmenting, and analysing audio tracks so that researchers can use only the components of the package that suit their needs.

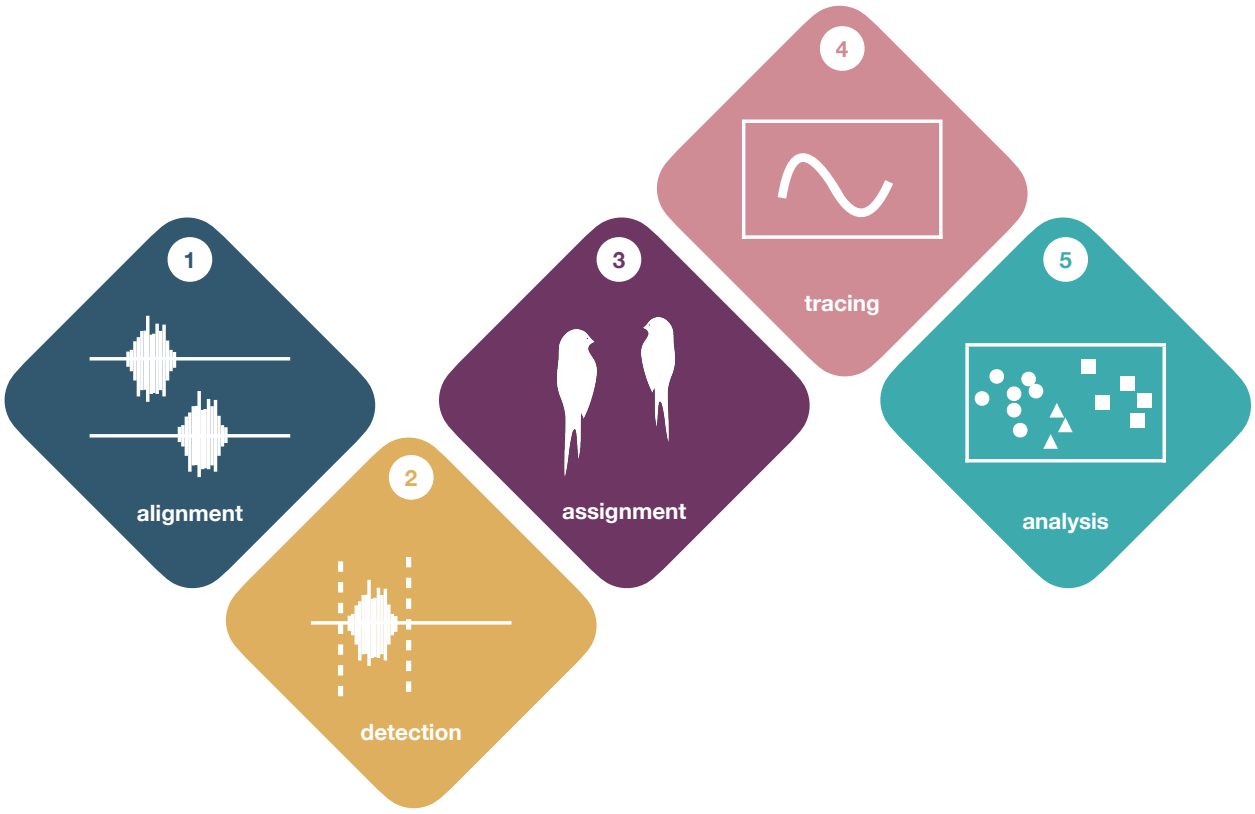

Figure 1: Flowchart from the `callsync` package. The *alignment* module can be used to align multiple microphones that have non-linear temporal drift. The *detection* module can be used to detect vocalisations in each recording. The *assignment* module can be used to assign a vocalisation to the focal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

Current research packages that implement call alignment strategies are either used in matlabAnisimov et al. (2014) or c++ (Gill et al. 2015). However these tools have not, up to now, been adapted for the R environment. Many of these tools are not documented publicly nor open source, and can require high licensing fees (i.e Matlab). While the design of this 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 more difficult contexts. This package is publicly available on 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. We will provide a case study and workflow that will demonstrate the use of this package.

Case study: cockatiel contact calls

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 a part of an ongoing study at the Max Planck Institute of Animal Behavior. Birds were housed in several groups of six individuals in a 4x3x2.7m aviary facility. We equipped six cockatiels 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. Audio recordings were scheduled to record for a maximum of 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 days of recordings (4 hours each) were selected for processing where the microphones were scheduled to record from 6:30 until 10:30 in the morning. After several days of deployment, microphone recorders are removed and downloaded as .wav files directly onto the computer from the tag. These tags are placed into the appropriate folder (see workflow instructions) and processed in accordance with our package workflow.

Installation and set-up

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

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

All required packages are also automatically installed and loaded for the case study when running the `00_set_up.R` script.

Alignment of raw recordings

Raw recordings consist of two days with 4 hour long wav files for six cockatiels. The backpack microphones have internal clocks that automatically turn them 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 during the recording period, creating additional variable drift up to a minute between recordings. The function `align` can be used as a first step in order to align the audio recordings. In order to accurately align these tracks, the full audio files are automatically split into shorter 15 minute chunks of recording to ensure that drift is reduced to mere seconds. This value can be adjusted depending on the amount of drift. The function selects one 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, in our case 0.5 seconds.This value can also be adjusted.

```
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('bird_', '_tag'),             # how to recognise the individiual/microphone in the path
      blank = 15,                               # how much should be discarded before and after in minutes
      wing = 10,                                # how much extra should be loaded for alignment in minutes
      save_pdf = TRUE)                          # should a pdf 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 (see 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 option `chunk_seq` in order to avoid re-running the entire dataset.

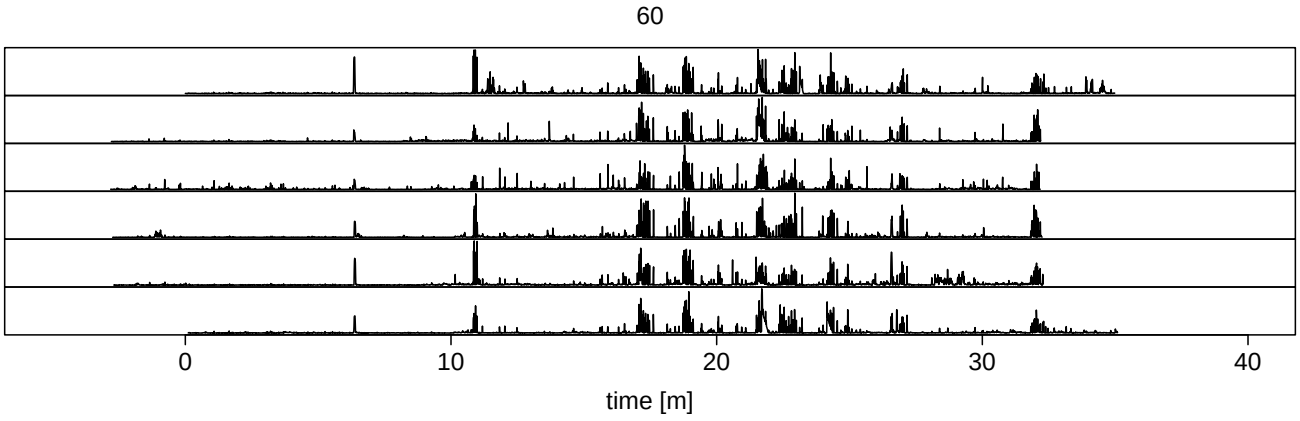

Figure 2: Example of the alignment output. Black lines represent the summed absolute amplitude per bin (= 0.5 seconds). Recordings are aligned relative to the first recording (which starts at 0). Note that recording 2-5 start ~2 minutes earlier, but are still aligned. The title displays the start time 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 individual. The `detect.and.assign` function 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 it calls the function `call.detect.multiple`, which can detect multiple calls in an R wave object. It first applies the `env` function from the *seewave* 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 or longer than a set maximum it returns all the start and end times as a data frame. Because the microphones on non-focal individuals are very likely to record the calls of the vocalising individual as well, we implemented a step that assigns the detected calls to the correct individual. For this `detect.and.assign` calls the function `call.assign`, which runs through all the detections in a given chunk for a given individual 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 is corrected by rerunning the `simple.cc` function. After alignment it calculates the summed absolute energy content on all recordings for the time frame when the call was detected and compares this to the focal recording. If the focal recording is louder by a set percentage 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. The function also allows the user to create a pdf with all the detections (see Figure 3 for a short example) to manually inspect the results.

```
detect.and.assign(ffilter_from = 1100,           # from where to filter in Hz
                  threshold = 0.4,               # 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
                  keys_rec = c('_\\(', '\\)_'),  # how to recognise the recording in the path
                  keys_id = c('bird_', '_tag'))  # how to recognise the individiual/microphone in the path
```

For the cockatiel dataset the function detected and assigned 1088 calls, 829 of which were retained after filtering. We manually assigned 174 calls in three chunks with a lot of activity and compared the performance of the `detect.and.assign` function to manually labelled data. The ground truth was performed using the function `calc.perf`. The false positive rate was 1% (single false detection) and the true positive rate was 53%.

Analysis of single calls and call comparison

To analyse the calls, the short wav clips 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. We ran the latter step in parallel using the function `mclapply` from the package *parallel* (see Figure 4a for an example).

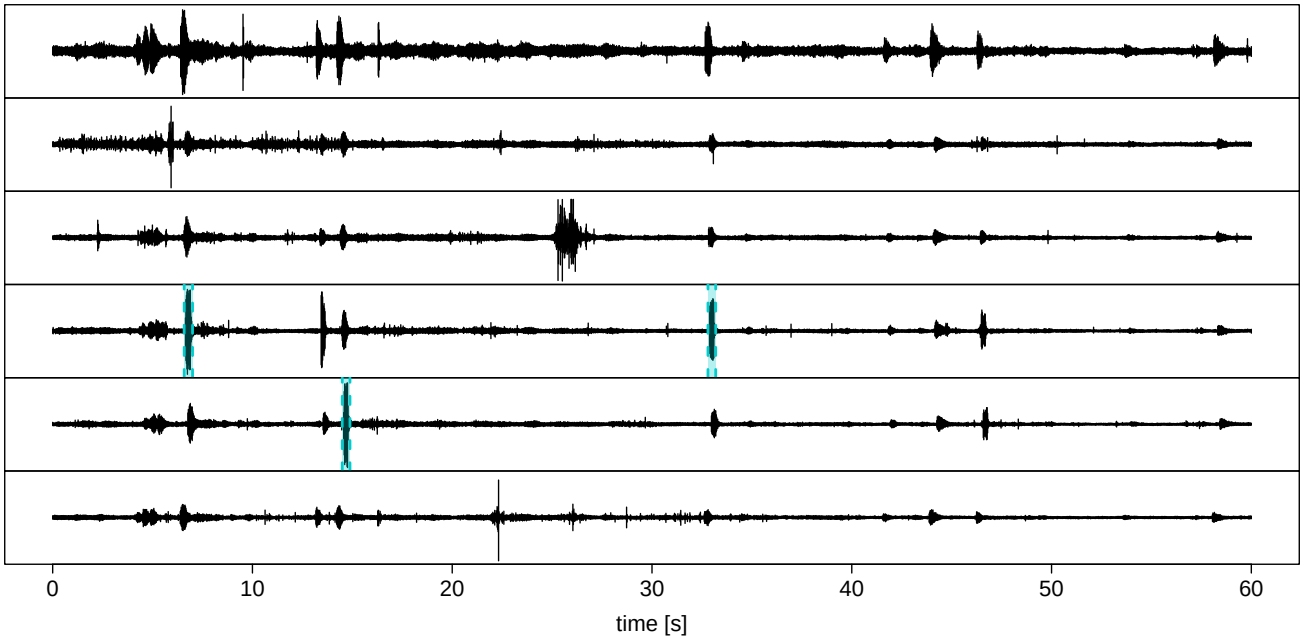

Figure 3: Example of the detection output. Black lines are the wave forms. Cyan dashed lines with shaded area in between are the detected calls.

```
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
```

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

```
keep = measurements$prop_missing_trace < 0.1 &   # max 10% missing points
  measurements$signal_to_noise > 5 &             # signal to noise at least 5
  measurements$band_hz > 600 &                   # bandwidth at least 600 Hz
measurements = measurements[keep,]               # keep only these measurements
traces = traces[keep]                            # and these traces
```

Another way to analyse calls is to measure their similarity directly. A frequently used method is SPCC - spectrographic cross correlation (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 principal coordinate analysis, and plotted the first two coordinates in. Calls clearly cluster by individual, but there is also a lot of overlap between individuals (see Figure 5).

Discussion

We present a case study and workflow to demonstrate `callsync`. Each of the modular components (alignment, detection, assignment, tracing, and analysis) successfully achieved the stated goals in the cockatiels 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

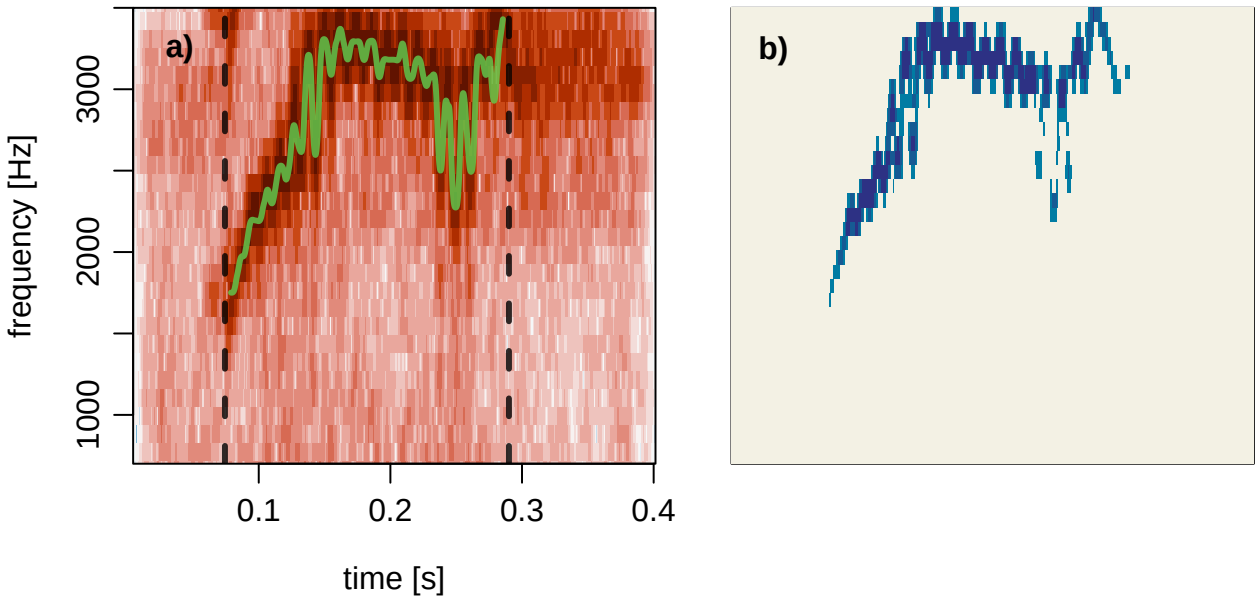

Figure 4: a) Spectrogram of a cockatiel call with start and end (black dashed lines) and the fundamental frequency trace (green solid line). b) Noise reduced spectrogram where darker colours indicate higher intensity.

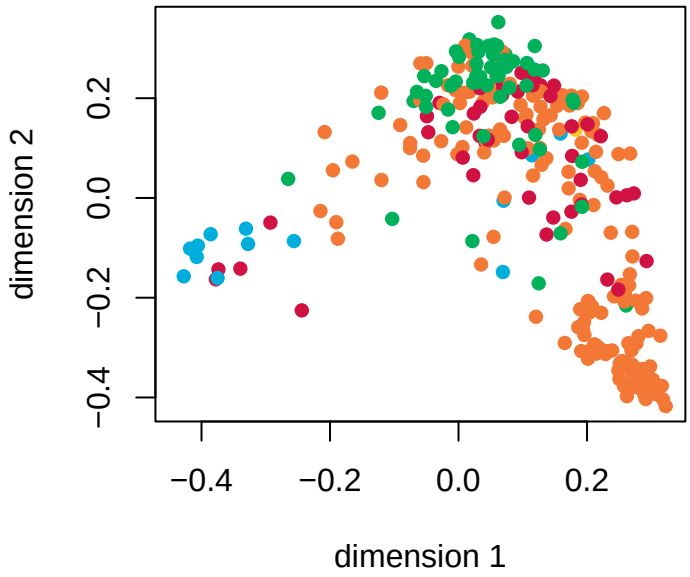

Figure 5: Call distribution in principle coordinate space. Dots represents calls and are coloured by individual.

focal individual making the call was selected (see Figure 3), and downstream data analysis was performed (figure 4). callsync can perform alignment even on drift that lasts at least 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 mislabelled recording (1% of total ground truth dataset), we are confident that `callsync` is a robust and useful tool for bio acoustics research. The true positive rate of our results was 53%, meaning that about half of the manually selected calls for ground truthing were not detected by the `detect.calls` function. However, it should be noted that the manual selection procedure included all call types while detect.and.assign were isolated to the typical cockatiel 'contact call'. Thus, one would need to run several detect.and.assign iterations to capture every type of vocalisation that were selected manually. Also settings were such that very little noise was detected and all noise was filtered out. Researchers themselves can decide the importance of whether obtaining every call type is needed or just particular types and apply the function accordingly.

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 that 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 Balsby and Bradbury (2009), or in situations where animals constantly move (i.e., flying) or do independent behaviours that other group members do not (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.

One possible challenge with the `call.detect` function could be that certain call types are not easily distinguishable from background noise (i.e. broadband signals; seals, monk parakeets, cockatoos). In these situations, `call.detect` is likely to pick up a significant amount of background noise instead of calls. Function parameters can be adapted and should function on most call-types, as can post-processing thresholding. For example, machine learning approaches Stowell et al. (2019) or image recognition tools Valletta et al. (2017) can be later applied to separate additionally detected noise. As well, in particularly difficult cases, once the align function is performed, the entire `call.detect` function can be swapped out for programs such as ANIMAL-SPOT (in review). However, this function should perform very well in most cases.

One critical issue when assigning a focal bird can be the proximity to other group members. If the group members spend time too close to one another (within roosts; Boughman and Wilkinson (1998),Kloepper and Kinniry (2018)), 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. In addition, other tools such as accelerometry data Anisimov et al. (2014), and video footage (Forrester 2008) as well as discriminant analysis (McIlraith and Card 1997) or cepstral coefficients (Lee, Lee, and Huang 2006) can be used when separating individuals is particularly difficult.

Lastly, both fundamental frequency tracing and SPCC work well in certain contexts. For example, automatic fundamental frequency traces works 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 instead such as Luscinia (Lachlan, Ratmann, and Nowicki 2018), manual tracing (Araya-Salas and Smith-Vidaurre 2017), or the researchers preferred analytical methodology for their system. 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 like PAM systems (Thode et al. 2006) and microphone arrays (Blumstein et al. 2011) should also find benefit within this package depending on the setup 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 package is publicly available on github and open source. We welcome all continued suggestions and believe that our package will result in an increase in the possibilities and amount of bio acoustic research. Our package provides functions that allow alignment, detection, assignment, tracing and analysis of calls in a multi-recorder captive setting. 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 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 contribute to understanding the dynamics of animal communication.

Acknowledgements

We thank Dr. Ariana Strandburg-Peshkin for early feedback. SQS and SAT received 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. The authors declare no conflicts of interest.

Data accessibility

All code is publicly available on GitHub (https://github.com/simeonqs/methods_paper). All data and code is also publicly available on Edmond (**link**) 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).

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, 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.

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.

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.

Forrester, Gillian Sebestyen. 2008. "A Multidimensional Approach to Investigations of Behaviour: Revealing Structure in Animal Communication Signals." *Animal Behaviour* 76 (5): 1749–60.

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.

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.

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*.