Step-by-step guide for compiling an example machine learning algorithm to detect target species sounds using R

Version 1.0, Aug 22nd 2024

This guide is part of the supplementary material of Landler et al. 2024 – Information, which contains all the specific R files and example files that allow you to set up and test the Machine Learning Algorithm (MLA) for detecting green toad breeding calls. This should allow you to modify the scripts to fit your particular target species sound or study questions.

Software used

Working installations of

- R [version 4.2.2 used]
- RStudio [version 2023.06.0+421 used] (both free, open-source software)

The session info obtained by the function SESSIONINFO for a successful run of the scripts:

R version 4.2.2 (2022-10-31)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Ventura 13.4.1

Matrix products: default
LAPACK: /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRlapack.dylib

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:

[1] base	datasets	graphics	grDevices
[5] methods	parallel	stats	utils

other attached packages:

[1] av_0.8.6	caret_6.0-94	dplyr_1.1.3	forcats_1.0.0
[5] ggplot2_3.5.1	knitr_1.45	lattice_0.22-5	lubridate_1.9.3
[9] NatureSounds_1.0.4	purrr_1.0.2	readr_2.1.4	seewave_2.2.3
[13] stringr_1.5.0	tibble_3.2.1	tidyr_1.3.0	tidyverse_2.0.0
[17] tuneR_1.4.5	warbleR_1.1.29	xgboost_1.7.5.1	

loaded via a namespace (and not attached):

[1] bitops_1.0-7	brio_1.1.3	class_7.3-22	cli_3.6.1
[5] codetools_0.2-19	colorspace_2.1-0	compiler_4.2.2	data.table_1.14.8
[9] digest_0.6.33	dtw_1.23-1	e1071_1.7-13	fansi_1.0.5
[13] farver_2.1.1	fftw_1.0-7	foreach_1.5.2	future.apply_1.11.0
[17] future_1.33.0	generics_0.1.3	globals_0.16.2	glue_1.6.2
[21] gower_1.0.1	grid_4.2.2	gtable_0.3.4	hardhat_1.3.0
[25] hms_1.1.3	ipred_0.9-14	iterators_1.0.14	jsonlite_1.8.7
[29] lava_1.7.3	lifecycle_1.0.3	listenv_0.9.0	magrittr_2.0.3
[33] MASS_7.3-60	Matrix_1.5-4.1	ModelMetrics_1.2.2.2	munsell_0.5.0
[37] nlme_3.1-163	nnet_7.3-19	parallelly_1.36.0	pbapply_1.7-2
[41] pillar_1.9.0	pkgconfig_2.0.3	plyr_1.8.9	pROC_1.18.5
[45] prodlim_2023.08.28	proxy_0.4-27	R6_2.5.1	Rcpp_1.0.11

[49] RCurl_1.98-1.13	recipes_1.0.8	reshape2_1.4.4	rjson_0.2.21
[53] rlang_1.1.2	rpart_4.1.21	rstudioapi_0.15.0	scales_1.3.0
[57] signal_0.7-7	splines_4.2.2	stats4_4.2.2	stringi_1.7.12
[61] survival_3.5-7	testthat_3.2.0	tidyselect_1.2.0	timechange_0.2.0
[65] timeDate_4022.108	tools_4.2.2	tzdb_0.4.0	utf8_1.2.4
[69] vctrs_0.6.4	withr_2.5.2	xfun_0.41	

- Audio editing software of your choice (e.g. Audacity, which is free, open-source, and multi-platform; https://www.audacityteam.org/).

File and folder structure

Structure	Contents
Root folder	• Your chosen project folder name
├── Audio_files_training	• Audio files for MLA training
│ ├── non-target_species_source	• Original files (in non-WAV audio formats)
│ ├── non-target_species_WAV	• Original files (in WAV audio format); to use in the MLA training
│ ├── target_species_source	• Original files (in non-WAV audio formats)
│ ├── target_species_WAV	• Original files (in WAV audio format)
│ └── target_species_WAV_trimmed	• Trimmed target sounds to use in MLA training
├── Matrix_temporary_RDS_files	• Temporary storage of matrices as RDS
├── MLA	• Location of the MLA (MLA_xgboost_green-toad_example) and the corresponding confusion matrix results (MLA_confusion_matrix.txt)
├── MLA_output	• Location for output files of the MLA when analysing sound files (RDS file and a txt file)
├── R_scripts	• The R scripts that create and use the MLA
│ ├── 0_overview.R	• An overview file with the steps
│ ├── 1_global_constants.R	• Sets up the global constants for easier setup
│ ├── 2_convert_audio_to_WAV.R	• Converts audio to WAV (in this example, from MP3)
│ ├── 3_target_species_sound_cleaning.R	• Trims the target sounds from input files and saves them for MLA training
│ ├── 4_create_sound_matrices.R	• Creates temporary RDS matrices for MLA training
│ ├── 5_MLA_training.R	• Trains the MLA
│ ├── functions_sound_identification.R	• R functions used throughout
│ └── animal_sound_identification_MLA.Rproj	• R project file
├── Sounds_to_identify	• Location of files to be identified by the MLA
└── Compiling a machine learning algorithm to detect target animal sounds.docx	• This guide

Notations used within the guide

'R FUNCTION' – R function name

"folder/*file name*" – folder/file name; folder names are preceded with a "/"

Step-by-step instructions

Below we provide the steps to replicate the process of generating the MLA provided in the publication. You can adapt the scripts to the particular sound characteristics you are interested in by modifying parameters such as the frequency ranges (parameter 'flim' (const_flim_low and const_flim_high) and also function 'FFILTER') and also time interval (parameter 'offset' (const_offset). Additionally, the scripts have specific comments and notes within.

Setup

1. Check that you have all the needed R and RStudio components installed and the folder/file structure is correct (see above). Once you load the project you can use the INSTALL.PACKAGES() provided in "*/R_scripts/0_overview.R*".
2. Load the R project file ("*/R_scripts/animal_sound_identification_MLA.Rproj*") – this will allow you to load all the necessary scripts without setting a new working directory.
3. Open the file "*/R_scripts/0_overview.R*" in RStudio. "*0_overview.R*" guides you through the whole process, by calling the respective scripts using the function 'SOURCE'. Running the entire "*0_overview.R*" as is would result in a target species' sound detection MLA generated and saved in "*/MLA*".
4. Check the global constants (variables that need to be identical between the scripts) defined in "*1_global_constants.R"*. Likely you do not need to change anything to run the example.

Generating the MLA

5. The script works with audio files in WAV format, so in case you have source files in other formats (e.g. MP3), convert them all to WAV using specialized audio editing software or follow the approach shown in "*2_convert_audio_to_WAV.R*".

   Note: The need to use WAV files is commented out in the scripts. This step was already done in this example.
6. We need to make sure that we train the MLA using samples that include only the target species sound we want to recognize. Thus, the target species' sound files require preliminary work that needs to be carefully done. For each file, we manually select the section containing the target species' sound. The 'sound' might be a single animal calling continuously or several individuals calls completely or partially overlapping. However, if there are notable breaks (i.e., pauses) between calls these need to be removed (i.e., split in separate instances, each containing a continuous call). Listen to each audio file using the specialized audio editing software and write down (e. g., in a text file) the approximate time interval containing the target species' sound. You may use the audio software to visualize the spectrogram to aid in trimming the file accurately. However, you can also visualize and trim the files with the R script we provide ("*3_target_species_sound_cleaning.R*", see

the comments inside this script for more information) – this way the process is stored and you can always go back, double-check your decisions, and make corrections if need be.

Note: For "poor-quality" audio (e.g. with a lot of background noise or barely audible target species' sound) we recommend using time intervals where it is certain that the target sound is present. It is best for the MLA to use shorter duration sounds containing only the correct target sound. In contrast, adding audio without the target sounds will train the algorithm on non-target sounds, leading to increased rate of misidentifications. In general, we recommend including some low-quality training files, which will increase the robustness of the MLA.

7. After the target species' sound files are trimmed to the relevant parts (and are thereby sometimes split into multiple smaller files), we convert both the trimmed target sounds and the non-target sounds into the matrix used for model training ("*4_create_sound_matrices.R*"). The core function to transform the audio files into the matrix is sourced (loaded via SOURCE) within the script and provided in the given folder structure (function 'REARRANGING_BOTH' within the background script: "*functions_sound_identification.R*").

   Note: For different types of sounds and/or species you might want to change the parameters for 'flim' for function 'FFILTER' (const_flim_low and const_flim_high) and parameter 'offset' (const_offset) (check "*1_global_constants.R"* and "*4_create_sound_matrices.R*"). However, we advise against making major changes, as the default values appear to work very well. However, there are two necessary adaptations when applying this script to other species than the green toad.

   First, the frequency range ('flim'::'FFILTER') should be set wide enough to include the base frequency of the target species and specific enough to exclude as much background noise as possible.

   Second, the minimum time interval that the MLA needs to classify the call should be considered (related to the parameter 'offset' (const_offset), set at +/- 0.2 s, using a value of 18; see "*1_global_constants.R"* and "*4_create_sound_matrices.R*"). Each line in the matrix, representing the information on which the MLA decides if this time point is part of the desired sound or not, includes the spectrogram information of the preceding and following 0.2 s in specific columns. Hence, the MLA not only gets trained on the momentary amplitude per frequency pattern, but also on the timing of the previous and following spectrogram patterns. Therefore, you should think about a time interval that is small enough to contain the information characteristic for the target species' sound, without it being unnecessarily long to add noise at the start and the end of target sound (and thus decrease the accuracy of the MLA) and also increase the matrix sizes (and thereby increase the time needed for calculations).

8. The resultant matrices are then used to train the extreme gradient boosting algorithm ('xgboost') ("*5_MLA_training.R*") in a two-step process. First, the cleaned audio files (in "*/Audio_files_training/target_species_WAV_trimmed*" and "*/Audio_files_training/non-target_species_WAV*") are automatically split into randomly selected training (70%) and testing (30%) data sets. This is used to obtain accuracy measures within the given set of audio files. After this is calculated (saved in "*/MLA/MLA_confusion_matrix.txt*"), the MLA is trained on the entire available data set to provide the best detection possible with these exemplary files.
9. Now the obtained MLA can be called through the target species' detection function 'TARGET_SPECIES_SOUND_FUNCTION()' loaded in the script "*6_batch_sound_identification.R*".

   Note: You can specify when the MLA considers an audio file as containing the sound of the target species or not. Inside the function we have set the 'positive detection' as '>=10' consecutive time intervals categorized as "Target species detected" by the xgboost algorithm (this can be changed on line 309 of "functions_sound_identification.R"). In addition, here you could change the appearance of the resultant spectrogram and the video output: for instance, you can change the color and shape of the markings indicating the moments of detection (line 223 onwards of "functions_sound_identification.R").

   We have turned the video output off in the default parameter settings in the TARGET_SPECIES_SOUND_FUNCTION()' and recommend leaving it off in general as it significantly slows down obtaining the results and producing videos for each analyzed audio file could take up a lot of storage space. The movie can be generated by setting "save_movie=TRUE", the plot can be shown by setting "only_text=FALSE".

   Important Note: For the MLA to function, key values (e.g. the frequency range, offset) or any other parameters for manipulating the audio files must be identical between the different scripts. The MLA only works if the matrices are derived the same way along with the training of the MLA and running it to identify sounds. Therefore, we have placed them as global constants set in "*1_global_constants.R".*

Application of the MLA

10. To demonstrate how the TARGET_SPECIES_SOUND_FUNCTION()' operates, load (using line 104-109 in the 0_overview.R file) the eight audio files in the folder "*../Sounds_to_identify*" that were not included in the training procedure and let it be classified by the MLA. The evaluation is printed directly in the RStudio console in the following way:

```
[1] "File 1 of 8 ../Sounds_to_identify/ID-983366_240602_Be Rehor.wav|Non-definitive result; double check"
[1] "File 2 of 8 ../Sounds_to_identify/ID-984295_240604_Quetzalcoatl.wav|Target species not detected"
[1] "File 3 of 8 ../Sounds_to_identify/ID-985566_240607_Quetzalcoatl.wav|Non-definitive result; double check"
[1] "File 4 of 8 ../Sounds_to_identify/ID-990095_240616_Jasl.wav|Target species not detected"
[1] "File 5 of 8 ../Sounds_to_identify/WK_ID-985807_240607_Sarah Moser.wav|Target species detected"
[1] "File 6 of 8 ../Sounds_to_identify/WK_ID-986160_240607_Bettina Seidl.wav|Target species detected"
```

[1] "File 7 of 8 ../Sounds_to_identify/WK_ID-993918_240624_Huiza.wav|Target species detected"

[1] "File 8 of 8 ../Sounds_to_identify/WK_ID-999265_240706_Amphibi Buffay.wav|Target species detected"

Evaluation results are printed after "|" and could be either "Target species detected", "Target species not detected", "Non-definitive result; double check" or "All time points identified as target species, check potential false positives!".

Note: If your goal is to classify green toad calls from the nominate subspecies (*Bufotes v. viridis*) use the MLA provided in the supplement ("GREEN_DETECTION()" with the MLA file "*xgbst_green_toad_calls_maxdepth100_nrounds100_Final*", updated versions will be provided at this GitHub link: LINK_WILL_BE_PROVIDED_UPON_ACCEPTANCE) which was trained on much more data. This working example just demonstrates the procedure on a small data set.

Notes and potential improvements, issues, etc.

The provided method works for identifying the presence of a single target sound within an audio file. However, we presume it could be relatively easily adapted if the target species may produce more than one specific sound, you could run the process described here for every specific sound and create multiple MLAs (remember to use different copies of the root folders for each sound as not to overwrite MLAs). At the end, in a new instance of the root folder you can copy all the MLAs using unique file names in "*/MLA*" and then use the detection function specifying the different algorithms.

Additionally, this specific method for training the MLA should work best when the target sounds have repeating elements (e. g. such as in typical amphibian breeding calls); the applicability to single long-duration sounds with increasing/decreasing frequency (e. g. some bird calls) has not been tested.