Title: 'Sound-Extraction: A Python package for subsampling audio files'

Authors:

- Name: Prayag N. Shah
- Name: Douglas P. Hynes

Date: 26 July 2023

Summary

Ecologists use digital audio recordings of ambient sounds to monitor changes in habitats, biodiversity, and ecosystem health. Recording devices can produce massive data sets consisting of multitudes of long duration audio files. Machine learning tools can aid in the anlaysis of this big data, but the ground truthing of sounds (e.g., validating species-specific vocalizations) must be done by listening to, and visualizing, a much smaller fraction of audio. The `Sound-Extraction` Python package enables user-friendly batch subsampling of long duration audio files (Figure 1). The program can segment (aka "clip") and extract (aka "copy") new recordings of a defined duration, using simple command-line arguments. Options include stratified sampling, with subgroups that can be defined by the user, or by the `recording_times_generator` program, which extracts audio bounded by start and end dates, a given location, and the corresponding sunlight phase. The package supports both WAV and FLAC formats. Functionality in `Sound_Extraction` and `recording_times_generator` is primarily handled using two key open source python packages: `soundfile` (Bechtold, 2013) and `astral` (Kennedy, 2009).

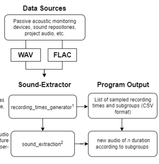

Figure 1. The general workflow of the Python program `Sound-Extraction`.

Statement of need

The availability of low-cost hardware and innovative software has facilitated long-term field deployments of acoustic devices for ecological research and monitoring (Morgan and Braasch, 2021; Roe et al. 2021). Ecologists can extract ecologically significant information from the resulting–often massive–volumes of audio data produced by recording devices (often referred to as "autonomous recording units", or ARUs), through the computation of acoustic indices (Campos et al., 2021) or by discriminating features with machine learning techniques (Prince et al., 2019), among others. But teasing out relavant biological information is a laborious process, requiring many, often repetitive, tasks that can hinder analyses. For example, many algorithm-aided analyses still require expert validation of the sound content itself. In this case, the acoustic analyst must navigate through a myraid of recordings, creating sound clips and sprectrograms to manually skim, scan, and listen to (Wimmer et al., 2013; Budka et al., 2023). Preceding validation, audio recordings must often be sampled, which includes several intermediate tasks, such as the running of sampling scripts to assign recordings to various subgroupings. Recordings are then selected, segmented according the sampling regime, and finally extracted (i.e., copied and moved) with standardized durations to new locations.

The computational limits of software and hardware presents other challenges. In certain bioacoustic tools, analyses may be optimzed with audio of specific durations (e.g., SonoBat; Szewczak, 2010), hence presenting the need for one to segment those audio files that do not fit within a computational optimum. `Sound-Extraction` enables one to slice long-duration audio recordings into shorter durations of uniform length, while parsing and preserving the datetimes in file names.

To help facilitate manual processing tasks, and comparisons of audio data sets with standardized samples, we present `Sound-Extraction`, an audio extraction program that enables users to sample, clip, copy, and move recordings to some pre-defined folder structure. Written in Python, users may define and import lists of subgroups and their corresponding audio files, the latter of which are then clipped and copied in batches on the basis of their datetimes and subgroupings. The program utilizes basic data structures like dictionaries, strings and lists, and supports those working with FLAC or WAV audio formats, and frequencies up to 256,000 hertz. `Sound-Extraction` efficently manages datetime objects; both sample and input recording datetimes are stored in data dictionaries, with the original recordings serving as keys and the corresponding sample files as values in the `Sound-Extraction` tool. In this way, sample files are assigned correctly to the original audio files, and filtered (i.e., sampled) files are passed through the `soundfile` library for the final extraction. The `soundfile` library enables the storage of audio data as NumPy arrays (Harris et al., 2020), facilitating seamless extraction of audio clips according to user-defined subgroups, or with groups created via the `recording_time_generator`, which samples recordings based on the sunrise and sunset times of given location and date.

While existing public libraries like FFmpeg (Bellard, 2006) and SoX (Norskog, 1991) can segment and copy recordings, `Sound-Extraction` will appeal to those with limited programming experience. Hence, the purpose of `Sound-Extraction` is to address this gap by offering a Python package that implements customized batched sampling and segmenting of audio.

Usage Scenarios

1. Generate sample times based on sunlight phase using start and end dates, and a location:

```
pip install sound-extraction
recording_times_generator
--start_date "2021-07-01"
--end_date "2021-07-05"
--latitude "42.12547"
--longitude "-50.55426"
--timezone "Canada/Atlantic"
--sample_size 1
```

The `recording_times_generator` works by passing datetimes (from 1-5 July 2021, in this case) through `astral`, and produces a sample recording time every half an hour (n = 48 maximum possible samples per day; Table 1) relative to sunlight phase.

Table 1: Rules defining the creation of the daily sample recording time list from which audio extractions occur.

Category	Time chunk	Maximum no. of samples
Nocturnal	Sunset+1.5H 0M 0S until next day Sunrise-1.5H 0M 0S	11
Sunrise	Sunrise-1H 0M 0S until Sunrise+5H 0M 0S	12
Daytime	Sunrise+5.5H 0M 0S until Sunset-1H 0M 0S	18
Dusk	Sunset-1H 0M 0S until Sunset+1H 24M 0S	6

This dataframe of datetimes represents a pool of samples that are implicitly passed to the `n sample_size` argument, which in turn takes a `n` random sample (without replacement) from six categories: daytime, dusk, nocturnal, and three morning times–early, mid and late sunrise (Table 2).

Table 2: Rules defining the sunlight phase based categorization of extracted recordings.

Categories	Assigned time range
Nocturnal	Sunset+0H 10M 0S until next day Sunrise-1H 4M 54S
Early Sunrise	Sunrise-1H 5M 0S until Sunrise+1H 59M 54S
Mid Sunrise	Sunrise+1H 59M 55S until Sunrise+2H 21M 23S
Late Sunrise	Sunrise+2H 21M 24S until Sunrise+4H 59M 53S
Daytime	Sunrise+4H 59M 54S until Sunset-59M 59S
Dusk	Sunset-1H 0M 0S until Sunset+1H 30M 0S

Categories can be edited by manipulating the assigned time chunks and time ranges in the source code.

2. Provide `Sound-Extraction` with the list of recordings and subgroupings

```
sound-extraction
--csv_file_path "G:/acousticData/SandPond192450/SandPond192450_RecordingDraw.c
--root_directory "G:/acousticData/FrontierLabs/SandPond192450/UneditedRecordin
--output_directory "G:/acousticData/FrontierLabs/SandPond192450/ExtractedRecor
--site_name "SandPond192450_"
--extension ".wav"
```

Shorter recordings can now be extracted by passing the generated list of recording times (either created via `recording_times_generator`, or by supplying a list of your own), to the `sound_extraction` command. The command is designed to recursively search directories for audio files with filenames in the datetime format {'yyyyMMddTHHmmss", e.g., 20220611T202300.wav or 20220611T202300.flac). To facilitate the standardized renaming of recording names, we employed the `Ecoacoutsics Metadata Utility` (Truskinger et al., 2023). The list of times to be extracted, within the CSV, must follow the naming convention such as "20220611_202300.wav" or "20220611_202300.flac", and fall under the heading `sampleFile`.

Recordings are then extracted and written to eponymous folders. If a user supplies their own subgroupings (i.e., category names), the program creates folders with those names instead. In cases where the user does not populate the `category` feld of the input CSV, the program will simply extract the audio recordings and store them in the current directory from where the program is being run.

3. Audio slicing for bioacoustic analysis

```
sound-extraction
--root_directory "G:/acousticData/FrontierLabs/SandPond192450/UneditedRecordin
--output_directory "G:/acousticData/FrontierLabs/SandPond192450/ExtractedRecor
--slice 15
```

In some bioacoustic analyses, large audio recordings can present computational challenges. In this example, the `Sound-Extraction` program is used to batch segment long-duration recordings into shorter, contiguous, clips. The option `--slice` defines the output duration, in seconds, of each segmented audio file. The function segments recordings but maintains the temporal continuity of the data, by parsing and writing corresponding datetimes (or timestamps) into the segmented recordings. For example, a 3600 second-long recording (e.g., "20230622_020000.wav"), sliced by 15 seconds, would produce 240 15-second long recordings with file names as follows: "20230622_020000.wav", "20230622_020015.wav", "20230622_020030.wav", and so on.

Acknowledgements

The authors are grateful for funding provided by Environment and Climate Change Canada. Thanks to Greg Campbell and Peter Thomas for providing sampling scripts, written in the R programming language, that inspired the creation of `Sound-Extraction`. We also extend a warm thank you to Harsil Patel and Shivam Patel for reviewing the code.

References

Bechtold, B. (2013). SoundFile can read and write sound files and manipulate their data. https://pysoundfile.readthedocs.io/en/latest/

Bellard, F. (2006). A complete, cross-platform solution to record, convert and stream audio and video. https://ffmpeg.org/

Bishop, S. (2004). Current and historical timezone database for Python. https://pypi.org/project/pytz/

Budka, M., Sokołowska E., Muszyńska, A., & Staniewicz, A. (2023). Acoustic indices estimate breeding bird species richness with daily and seasonally variable effectiveness in lowland temperate Białowieża forest. Ecological Indicators, 148, 110027. https://doi.org/10.1016/j.ecolind.2023.110027.

Campos, I. B., Fewster, R., Truskinger, A., Towsey, M., Roe, P., Filho, D. V., Lee, W., & Gaskett, A. (2021). Assessing the potential of acoustic indices for protected area monitoring in the Serra do Cipó National Park, Brazil. Ecological Indicators, 120, 106953. https://doi.org/10.1016/j.ecolind.2020.106953

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., & Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Kennedy, S. (2009). Package for calculating the times of various aspects of the sun and moon. https://astral.readthedocs.io/en/latest/

Morgan, M. M., & Braasch, J. (2021). Long-term deep learning-facilitated environmental acoustic monitoring in the Capital Region of New York State. Ecological Informatics, 61, 101242. https://doi.org/10.1016/j.ecoinf.2021.101242.

Norskog, L. (1991). SoX – Sound eXchange, the Swiss Army knife of audio manipulation. http://sox.sourceforge.net/

Prince, P., Hill, A., Piña Covarrubias, E., Doncaster, P., Snaddon, J. L., & Rogers, A. (2019). Deploying Acoustic Detection Algorithms on Low-Cost, Open-Source Acoustic Sensors for Environmental Monitoring. Sensors (Basel, Switzerland), 19(3), 553. https://doi.org/10.3390/s19030553

Roe, P., Eichinski, P., Fuller, R. A., McDonald, P. G., Schwarzkopf, L., Towsey, M., Truskinger, A., Tucker, D., & Watson, D. M. (2021). The Australian Acoustic Observatory. Methods in Ecology and Evolution, 12, 1802– 1808. https://doi.org/10.1111/2041-210X.13660

Szewczak, J. M. (2023). The benefits of full-spectrum data for analyzing bat echolocation calls. http://www.sonobat.com

Truskinger, A., MacAskill, N., Mercer, J., & Scarpelli, M. D. A. (2023). QutEcoacoustics/emu: Support for AudioMoth CONFIG.TXT files improved https://github.com/QutEcoacoustics/emu

Wimmer, J., Towsey, M., Roe, P., & Williamson, I. (2013). Sampling environmental acoustic recordings to determine bird species richness. Ecological Applications, 23(6), 1419–1428. http://www.jstor.org/stable/23596835?[1]