biobabel: a unified interface for reading a plethora of file formats for biosignals such as cardiac, respiration, electrodermal data

Floris Tijmen van Vugt [1,2,3,4,5]

1 Department of Psychology, University of Montreal, Montreal, Canada **2** International Laboratory for Brain, Music and Sound Research (BRAMS), Montreal, Canada **3** Centre for Research on Brain, Language and Music (CRBLM), Montreal, QC, Canada **4** Centre Interdisciplinaire de Recherche sur le Cerveau et l'Apprentissage (CIRCA), Montreal, QC, Canada **5** Haskins Laboratories, Yale University, New-Haven CT, USA

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Marcos Vital

Reviewers:

- @sappelhoff
- @finsberg
- @ntolley

Submitted: 20 March 2024
Published: unpublished

Summary

Human biosignals such as breathing, cardiac rhythms or skin conductance contain a wealth of information about cognition, emotion and social connection. Measuring these biosignals is now possible using a range of open-source or commercial sensors. However, the software accompanying each of such sensors stores data in all manner of different file formats. This makes it difficult for researchers across the globe to exchange analysis scripts, which is needed for data reproducibility. Biobabel is an open-source software package that reads all the major biosignal file formats and allows programmers to access the data in a unified, straight-forward manner. It provides a handy set of tools for inspecting data and performing basic manipulations. Biobabel thus hopes to contribute to a unified, practical foundation allowing researchers interested in biosignal to focus on extracting meaningful insights from these data.

Statement of need

There is increasing interest on the part of the neuroscience and psychology research community in biosignals, that is, measurements of cardiac activity (typically from the electrocardiogram, ECG), electrodermal activity (EDA), respiration, and others (Horvers et al., 2021; Massaro & Pecchia, 2016; Posada-Quintero & Chon, 2020; Varga & Heck, 2017). There are now wonderful packages for biosignal preprocessing (e.g. neurokit (Makowski et al., 2020)) and analyzing (e.g. biopeaks (Brammer, 2020)). However, progress is hampered by the proliferation of a multitude of file formats (EDF, XDF, OpenSignals, BDF, CSV, Acknowledge ACQ, etc.). Existing software packages typically read only one or two of these formats, requiring researchers to convert between formats which is tedious and error-prone, or simply impossible when using read-only libraries. Furthermore, data in these different formats is typically organized differently, requiring researchers to reorganize their code to cater to different formats.

Individual Python packages exist that can each read single data formats (e.g. pyxdf or pyedflib). However, each makes the data available in a different structure. This means that pipelines have to be changed when switching from one data format to another, which is tedious and error-prone. The situation is further complicated by the fact that different file formats make different assumptions about the data structure: in some formats, multiple signals in a file are forced to have the same sampling rate (e.g. OpenSignals (Braga et al., 2019)) whereas in other formats sampling rates can vary (e.g. XDF). In some cases the signals are supposed to have the same onset time (e.g. EDF) whereas other formats allow different onset times requiring re-aligning (e.g. XDF). All this makes conversion cumbersome and errors can easily slip in.

This state of affairs also hampers the development of unified, reproducible pipelines that can be shared between research groups across the globe. Increasingly, the field calls for sharing of data analysis pipelines between research groups as an indispensible step to much-needed reproducibility (Wratten et al., 2021). In addition, sharing analysis pipelines rather than reinventing the wheel allows for more efficient use of scientists' time.

Further, it is becoming increasingly important for physiological software to accommodate data from multiple participants. There is increasing interest in neuroscience in collecting physiological data simultaneously from multiple participants interacting in real-time (Kelsen et al., 2022). Such *hyperscanning* studies place unique demands on file structures that classically were designed for data from single participants only.

Thus, what is needed is a software package that can read these diverse formats into a reasonably flexible data structure that abstracts away from differences. Such a package reads data from a variety of data formats, accommodating data streams from multiple participants and allowing it to be written in a sensible native open-standard format.

These challenges were already solved for neuroimaging data by the *nibabel* package (Brett et al., 2024) from which we draw inspiration here. But for the physiology data, surprisingly such a software suite has been missing until now.

Functionality

`biobabel` is a Python package whose main functionalities are:

- Seamless reading of a host of physiology data file formats.
- Data flows into an object with a flexible internal structure supporting multiple data streams, time point markers, various sampling rates and multiple participants.
- Basic data manipulation (cropping in time, selecting subsets of channels, etc.) and visualization (previewing) not typically implemented in existing software packages.
- A set of Swiss army knife command-line based tools for on-the-fly data inspection and manipulation.
- Streamlined modular code that allows the package to be easily extended to read file formats not yet included.
- Data can be written to an open standard file format based on HDF5.

For a full demonstration, see the basic documentation and illustration notebook.

Supported data formats

At the time of writing the following data formats are supported:

Format	File extension	Supported by
Extensible Data Format	.xdf	pyxdf
BIOSEMI 24-bit BDF	.bdf	pybdf
BioPAC Acknowledge	.acq	bioread
OpenSignals (r)evolution / BiTalino	.txt	opensignalsreader
European Data Format	.edf	pyedflib
Generic CSV	.csv	Custom developed code including sniffing and educated guesses
hdphysio5	.hdf5	Native format developed specifically for biobabel

The format of input files is guessed automatically at the time of reading, using clues such as file extension, but if these are insufficiently informative, guesses are made based on sniffing of the file. For some file formats, such as CSV, the way these formats are used varies between research groups: CSV data represents a table but the meaning and names of various columns in this table are not standardized. In those cases, `biobabel` will try to guess the meaning of the various columns, for example automatically guessing one column to be a time column if its values are increasing almost always by the same amount.

Within Python the following code is sufficient to read a data file:

```python
import biobabel as bb
bio = bb.load('tests/example.hdf5')
```

Then, we can view basic properties of the data file:

```python
bio.print()
```

This will produce an overview of the dataset indicating sampling frequencies and durations:

```
Summary of Simulated data
· date  07/20/2023 10:48:32 EDT-0400

Participant 'a'
└ channel a_ecg [ modality ecg ] 15000 samples @ 1000.0 Hz = 15.0 s
└ channel a_ppg [ modality ppg ] 15000 samples @ 1000.0 Hz = 15.0 s

Participant 'b'
└ channel b_ecg [ modality ecg ] 15000 samples @ 1000.0 Hz = 15.0 s
```

And easily inspect the data using a plot:

```python
bio.plot()
```

Which produces:

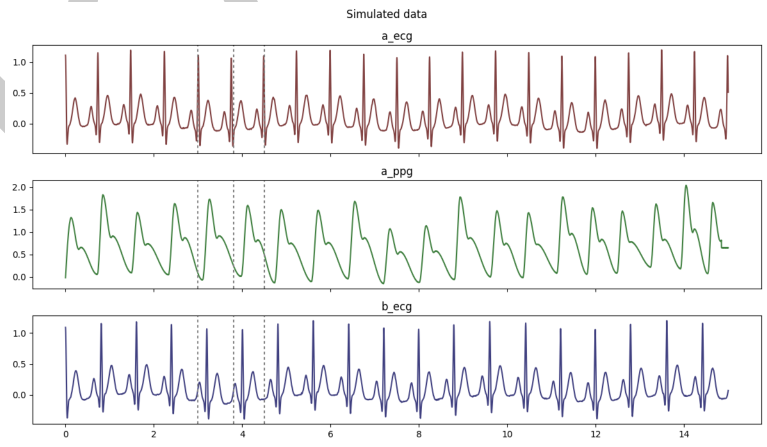

Figure 1: Overview plot of sample data file, indicating each channel as a separate panel. Vertical dashed lines are time markers.

Biobabel internal data structure

Interally, `biobabel` stores physiological datasets in a `Biodata` object (`bio` in the above example). Under the hood, this object contains a number of data streams, each of which is a single dimension data array with some associated key-value metadata, such as sampling frequency, participant ID, etc. Each data stream is identified with a unique ID.

The channel metadata allows us to easily find channels by data type:

```
bio.find_channels({'modality':'ecg'}) # find all channels containing ECG data
```

which returns a set of channel IDs: `['a_ecg', 'b_ecg']`.

The channel IDs can then be used to query the channel metadata (in dictionary format) and extract its data:

```
hdr,dat = bio.get('a_ecg')
hdr # find the associated metadata for this channel
```

which returns the metadata in `hdr`:

```
{'id': 'a_ecg',
 'participant': 'a',
 'sampling_frequency': 1000,
 'modality': 'ecg'}
```

In `biobabel`, each data stream can have its own sampling frequency, but all data streams are assumed to start at the same time. In my experience analyzing physiological data, this common starting time assumption was sensible, since it holds true in most applications and making this assumption simplifies subsequent data handling. For data formats in which this assumption does not necessarily hold true (e.g. XDF), data loaded into `biobabel` will be cropped by the software package to a common starting time.

`biobabel` also supports *markers*, which are points in time at which specific events are recorded to occur. This can be start/stop markers indicating separate recording segments (e.g. append-markers in BioPAC Acknowledge files). Markers are stored in the Biodata object and can be accessed using `bio.get_markers()` (to find the marker names) and `bio.get_marker(<NAME>)` (to extract the corresponding time points). In default plotting functions of `biobabel` they are indicated with dashed vertical lines (Figure 1).

`biobabel` allows a number of typical data management steps that most packages do not straightforwardly allow, such as cropping the data to a selected time range (`bio.crop(t_start,t_end)`) and dropping or selecting channels.

Finally, data can be saved in the `biobabel` native HDF5-based format (`bio.save`).

For labs engaging in hyperscanning, `biobabel` seamlessly accomodates support for data from multiple participants. Each data stream can be allocated to a specific participant, allowing the software to find all participants `bio.get_participants()` or get channels for a specific participant (`bio.find_channels({'participant':'b'})`).

Easy previewing and some manipulation from the command line

`biobabel` provides simple accessible previewing of data files directly from the command line. This functionality is inspired by AFNI (Cox, 1996), a toolbox of shell scripts for neuroimaging analysis.

The following shell scripts are currently included and available automatically if the package is installed via `pip`:

- `bioinfo <filename>` which reads the data file and prints a summary (a wrapper around `biodata.print()`)
- `biobabel <filename>` which reads the data file and produces a simple plot (a wrapper around `biodata.view()`)
- `tohdf5 <filename>` which converts a data file in any of the supported formats into biobabel's native HDF5 format.

- `biosplit <filename>` which splits the data along its integrated markers (which often correspond to different recording sessions) into multiple separate files (e.g. `<filename_001>`, `<filename_002>` etc.)
- `bioview <filename>` which launches a graphical user interface (GUI) reader allowing interactive inspection of data as shown below.

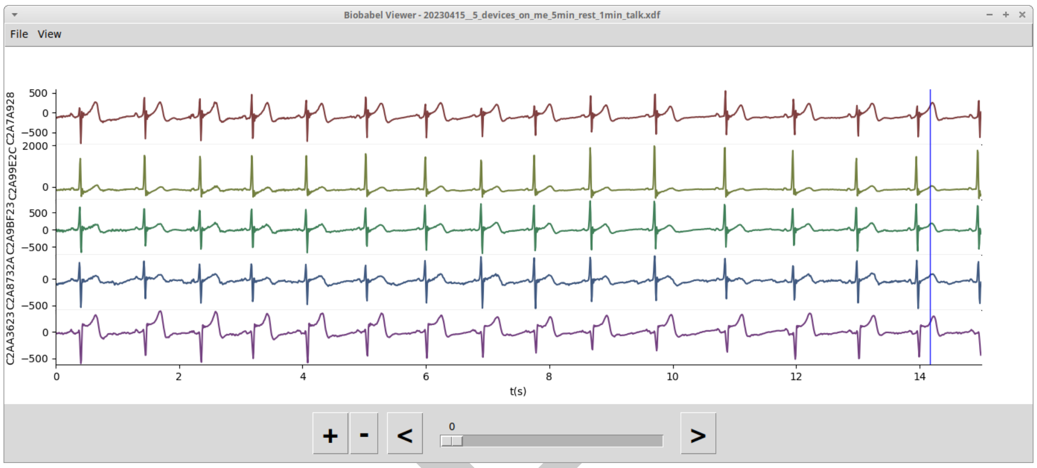

Figure 2: Bioview is a GUI allowing the user to inspect a data file by zooming and navigating the entire signal.

Integration with biosignals processing packages

Since `biobabel` takes care of all the peculiarities of data files, physiological processing pipelines can be substantially simplified. The following boilerplate code reads a data file and automatically finds the ECG columns and preprocesses the data using the excellent Python package `neurokit2` (Makowski et al., 2021):

```python
import neurokit2
import biobabel as bb
x = bb.load('dataset_copy.hdf5')
prep = {}
for hdr,signal in x.find({'modality':'ecg'}):
    prep[hdr['id']] = neurokit2.ecg_process(
        signal,sampling_rate=hdr['sampling_frequency'])
```

This code works without modifications for any of the supported data formats.

Conclusion

At the time of writing, `biobabel` is already being used at the Human Connection Science Lab and the International Laboratory for Brain, Music and Sound Research (BRAMS).

It is hoped that `biobabel` will simplify the lives of scientists by abstracting away from the specifics of physiology file formats. Using this package, data processing pipelines can be more easily shared across research groups that rely on different sensors, thus contributing towards greater reproducibility in our field.

Acknowledgements

This work was supported by a pilot project grant from the *Centre Interdisciplinaire de Recherche sur le Cerveau et l'Apprentissage* (CIRCA), Montreal, QC, Canada.

Mihaela Felezeu and Alex Nieva at BRAMS provided helpful tutorials on using all manners of biosignals. Inspiration for `biobabel` was taken from `nibabel` which is a Python library able to read virtually any neuroimaging file format in the known universe, and making it available in a unified Python interface (Brett et al., 2024). `biobabel` also builds on the strengths of a range of packages such as `matplotlib` (Hunter, 2007), `numpy` (Harris et al., 2020) and `pandas` (McKinney, 2010). I want to thank the contributors of all those packages for their excellent work.

References

Braga, L. C. M. F., Castro, M. C. F., & Avelino, V. F. (2019). Educational platform for physiological signal measurements. In R. Costa-Felix, J. C. Machado, & A. V. Alvarenga (Eds.), *XXVI brazilian congress on biomedical engineering* (pp. 671–677). Springer Singapore. ISBN: 978-981-13-2517-5

Brammer, J. C. (2020). Biopeaks: A graphical user interface for feature extraction from heart- and breathing biosignals. *Journal of Open Source Software*, *5*(54), 2621. https://doi.org/10.21105/joss.02621

Brett, M., Markiewicz, C. J., Hanke, M., Côté, M.-A., Cipollini, B., McCarthy, P., Jarecka, D., Cheng, C. P., Larson, E., Halchenko, Y. O., Cottaar, M., Ghosh, S., Wassermann, D., Gerhard, S., Lee, G. R., Baratz, Z., Wang, H.-T., Papadopoulos Orfanos, D., Kastman, E., … freec84. (2024). *Nipy/nibabel: 5.2.1* (Version 5.2.1). Zenodo. https://doi.org/10.5281/zenodo.10714563

Cox, R. W. (1996). AFNI: Software for analysis and visualization of functional magnetic resonance neuroimages. *Computers and Biomedical Research*, *29*(3), 162–173. https://doi.org/10.1006/cbmr.1996.0014

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, 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

Horvers, A., Tombeng, N., Bosse, T., Lazonder, A. W., & Molenaar, I. (2021). Detecting emotions through electrodermal activity in learning contexts: A systematic review. *Sensors*, *21*(23), 7869. https://doi.org/10.3390/s21237869

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Kelsen, B. A., Sumich, A., Kasabov, N., Liang, S. H. Y., & Wang, G. Y. (2022). What has social neuroscience learned from hyperscanning studies of spoken communication? A systematic review. *Neuroscience &Amp; Biobehavioral Reviews*, *132*, 1249–1262. https://doi.org/10.1016/j.neubiorev.2020.09.008

Makowski, D., Pham, T., Lau, Z. J., Brammer, J. C., Lespinasse, F., Pham, H., Schölzel, C., & Chen, S. H. A. (2021). NeuroKit2: A python toolbox for neurophysiological signal processing. *Behavior Research Methods*, *53*(4), 1689–1696. https://doi.org/10.3758/s13428-020-01516-y

Makowski, D., Pham, T., Lau, Z. J., Brammer, J. C., Lespinasse, F., Pham, H., Schölzel, C., & S H Chen, A. (2020). *NeuroKit2: A python toolbox for neurophysiological signal processing*. Zenodo. https://doi.org/10.5281/ZENODO.3597887

Massaro, S., & Pecchia, L. (2016). Heart rate variability (HRV) analysis: A methodology for organizational neuroscience. *Organizational Research Methods*, *22*(1), 354–393. https://doi.org/10.1177/1094428116681072

McKinney, W. (2010). Data structures for statistical computing in python. *Proceedings of the 9th Python in Science Conference*, 56–61. https://doi.org/10.25080/Majora-92bf1922-00a

Posada-Quintero, H. F., & Chon, K. H. (2020). Innovations in electrodermal activity data collection and signal processing: A systematic review. *Sensors*, *20*(2), 479. https://doi.org/10.3390/s20020479

Varga, S., & Heck, D. H. (2017). Rhythms of the body, rhythms of the brain: Respiration, neural oscillations, and embodied cognition. *Consciousness and Cognition*, *56*, 77–90. https://doi.org/10.1016/j.concog.2017.09.008

Wratten, L., Wilm, A., & Göke, J. (2021). Reproducible, scalable, and shareable analysis pipelines with bioinformatics workflow managers. *Nature Methods*, *18*(10), 1161–1168. https://doi.org/10.1038/s41592-021-01254-9