SoftwareX 10 (2019) 100287

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original Software Publication

pyphysio: A physiological signal processing library for data science approaches in physiology

Andrea Bizzego [a,*], Alessandro Battisti [b], Giulio Gabrieli [c], Gianluca Esposito [a,c], Cesare Furlanello [b]

[a] Università degli Studi di Trento, Italy
[b] Fondazione Bruno Kessler, Italy
[c] Nanyang Technological University, Singapore

ARTICLE INFO

Article history:
Received 14 May 2019
Received in revised form 8 July 2019
Accepted 8 July 2019

Keywords:
Physiological signal processing
Psychophysiology
Autonomic indicators
Data science
Python

ABSTRACT

The lack of open-source tools for physiological signal processing hinders the development of standardized pipelines in physiology. Researchers usually must rely on commercial software that, by implementing black-box algorithms, undermines the control on the analysis and prevents the comparison of the results, ultimately affecting the scientific reproducibility. We introduce pyphysio as a step towards a data science approach oriented to compute physiological indicators, in particular of the Autonomic Nervous System activity. pyphysio serves as a basis for machine learning modules and it implements a suite of combinable algorithms for processing of signals from either by wearable or medical-grade quality devices.

Code metadata

Current code version	v2.1
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2019_180
Code Ocean compute capsule	https://doi.org/10.24433/CO.9224164.v1
Legal Code License	GNU GPL v3
Code versioning system used	git
Software code languages, tools, and services used	python v3.x
Compilation requirements, operating environments & dependencies	Requires the following Python packages: numpy, scipy, matplotlib
If available Link to developer documentation/manual	https://github.com/MPBA/pyphysio/tree/master/tutorials
Support email for questions	bizzego@fbk.eu

1. Motivation and significance

Physiological signals are extensively used in medical sciences for diagnostics and monitoring of a patient's health status. They also provide effective insight into the psychophysiological regulatory mechanisms that enable us to adapt to environmental changes and react to external stimuli [1,2].

The rise of Wearable Sensor (WS) technologies enabled the use of physiological signals as a core method in multiple research fields. The promise of WSs is to enable the acquisition of signals in the ecological context [3], based on advances in miniaturization, accuracy, sampling rates and energy efficiency. Unfortunately, such technological progress is not matched yet by a wide availability of scientific computing platforms for physiological data analysis.

The first hurdle to reproducibility of studies with physiological signals is the current preference for black-box algorithms, often centralizing the analysis in a proprietary cloud framework and preventing the access to raw data in the worst cases. With few exceptions (e.g. OpenANSLAB [4]), the scope of both commercial and open-source software is often focused on a single type

* Corresponding author.
E-mail address: andrea.bizzego@unitn.it (A. Bizzego).

https://doi.org/10.1016/j.softx.2019.100287

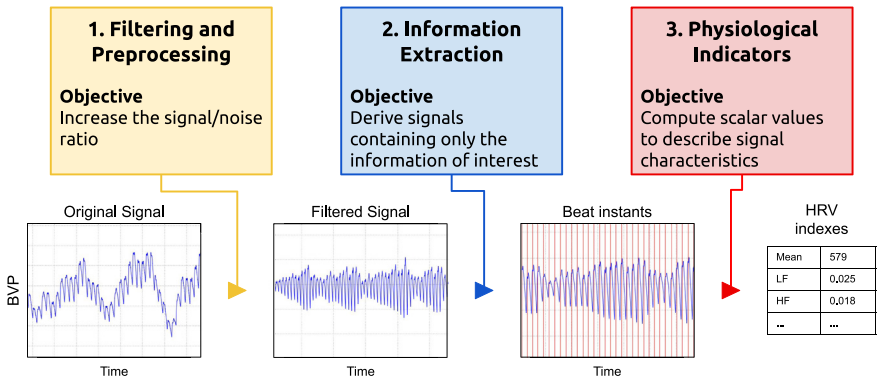

Fig. 1. Three steps of physiological signal processing (top) and an example on a Blood Volume Pulse (BVP) signal (bottom).

of signal, such as pyEEG [5] for electroencephalographic (EEG) signals; pyHRV [6], gHRV [7] and KUBIOS [8] for Heart Rate Variability (HRV) analysis; Ledalab [9], SCRalyze [10] and cvxEDA [11] for Electrodermal Activity (EDA). Lack of transparency and fragmentation of solutions in multiple software tools hinders reproducibility within and between studies that the Python library `pyphysio` aims to solve.

`pyphysio` is designed to offer a friendly interface to physiological signal processing. Its primary field of application is the study of the activity of the ANS activity; however, its use is not limited to psychophysiology but can be extended to any research based on physiological signals (e.g. sport and health, affective computing). `pyphysio` provides a wide range of choices and data analysis options, and is tailored to researchers with limited programming expertise. Although we acknowledge that each study has specific scientific questions and diverse experimental design, the sequence of data analysis modules should be fairly easy to concatenate in a standard modality and in a consistent framework of customized pipelines. The advantage of adopting a well-structured, yet general, framework is that state-of-the-art functions can be progressively made available to psychophysiology specialists who do not usually have the skills to efficiently code signal processing modules and may end up with using sub-optimal solutions.

In summary, the adoption of `pyphysio` is expected to increase the quality of studies based on the quantification of physiological signals: first, by growing the signal processing competences of researchers, without the need of having coding and programming skills; second, by allowing the development of reproducible pipelines and, third, by facilitating the implementation and adoption of novel algorithms.

2. Software description

We developed `pyphysio` using Python, in particular on top of well-established libraries for scientific computing, in particular *scipy*[1] and *numpy*[2] to ease the processing of multiple types of physiological signals. In particular, `pyphysio` provides algorithms and functions to accomplish a typical psychophysiology signal processing pipeline, which is composed of the following steps (see Fig. 1):

1. Filtering and preprocessing: for removing noise and improving the quality of the signal. These algorithms are generic and can be applied to different types of signals;
2. Information Extraction: steps that aim at extracting key information from the input signal. The algorithms are mostly specific to the type of signal (e.g. heart beat detection from an ECG signal); but they can also be used on multiple signal types by setting the algorithm's parameters accordingly;
3. Physiological Indicators: application of mathematical and statistical functions that compute metrics (i.e. scalar values) from the input signal (e.g. the energy in a specific frequency band). `pyphysio` provide a set of indicators directly associated with the activity of the Autonomic Nervous System (ANS) to support psychophysiology research. The main indicators proposed in literature are available in `pyphysio` for each type of supported signal, with default parameters specified (see Table 1).

Expected users: `pyphysio` is designed for users without extensive programming skills. Functions are defined at high levels to enable the complete development of experiments that investigate the physiological and psychological state of an individual. Further, the modular structure and the availability of `pyphysio` classes and functions enable the implementation of novel ad-hoc algorithms and of more complex signal processing pipelines.

Basic concepts: `pyphysio` has been developed for the off-line analysis of physiological signals, such as ECG, EDA, Respiration, Electromyography. `pyphysio` can be used without any restriction on the type of instrumentation used to collect the data. However, special functions have been provided to allow the researchers to deal with the lower quality of the data from the WSs. Its main intended use is to compute physiological indicators to study the activity of the ANS, which is the typical approach of psychophysiology research.

Main classes and functions: `pyphysio` introduces the class Signal (to represent a physiological signal) extending the numpy.array class of the Numpy package. All methods of numpy.array can thus be exploited to process an instance of the class Signal. Signals that are sampled with a fixed sampling frequency are represented with the subclass EvenlySignal, defined by the starting timestamp and the sampling frequency. Other types of signals (e.g. event triggers) are naturally associated to samples not equally spaced over time. The `pyphysio` subclass UnevenlySignal is available when the sampling frequency is not constant and thus the time reference of each sample needs to be stored. Both classes use default methods to access signal attributes, such as signal values or timestamps — get_values(), get_times(), or for fast signal manipulation (e.g. resampling, visualization or segmentation). The class Algorithm describes a computational function F with the set of parameters p that regulate its behavior. A signal processing pipeline is composed of multiple processing steps, each one modeled as an Algorithm instance. Different types of signal processing algorithms

[1] www.scipy.org/.
[2] www.numpy.org/.

Table 1
Main physiological indicators provided by default in pyphysio.

Signal	Indicators
Inter Beat Intervals	RRmean, RRSTD, RMSSD, pNN50, pNN25, pNN10, triang, TINN, VLF, LF, HF, SD1, SD2, SD12, Sell, ApEn, SampEn, DFAa1, DFAa2 [12]
Electrodermal Activity	Mean, Range, Standard deviation, Mean peaks amplitude, Maximal peak amplitude, Peak slope, Peak duration, Number of peaks, AUC [13]
Electroencephalogram	Energy in: delta, theta, alpha, beta bands [5]
Electromyogram	Maximum, Minimum, Mean, Range, Standard deviation, AUC, Energy in 4–40 Hz band [14]
Respiration	Energy Low (0–0.25 Hz band), Energy High (0.25–5 Hz band), Energy Ratio, Breath Rate [14]
Activity	Maximum, Minimum, Mean, Range, Standard deviation, AUC, Energy in 0–25 Hz bands [15]

are implemented, including simple wrappers of existing functions (e.g. scipy.filter) and more complex procedures (e.g. to estimate the component of an EDA signal associated with the sympathetic regulation [9]). The main algorithms implemented in pyphysio are listed in Table 2.

Furthermore, pyphysio provides functions to assess signal quality, the temporal alignment of multiple signals, the extraction of indicators from sequential segments and the output from post-processing analysis. In comparison to the scipy package and other general resources for scientific programming, pyphysio functions are indeed tailored for signal processing, mainly including parameters with a direct physical meaning (e.g. the use of seconds instead of indices of vectors, or Hertz instead of the relative frequency). To facilitate the creation of processing pipelines, the pyphysio Algorithm class is extended by three sub-classes (Filter, Estimator, and Indicator). Each sub-class is dedicated to a phase in a typical signal processing pipeline (see Fig. 1).

The class Tool collects additional pyphysio algorithms that can be used to develop complex processing steps, such as to identify local maxima/minima, and to estimate the power spectrum density. The class Segment and its sub-classes are used to generate segments of a signal (for instance with a shifting window) and manage the execution of operations on each segment, such as the computation of physiological indicators. Notably, the methods in Segment can include the experiment timeline as an additional signal in order to associate labels to computed indicators (e.g. the experimental session label to enable a consistent comparison between different experimental conditions). Additionally, the library provides diagnostic algorithms, such as validation of the input data with Signal Quality Indicators for preprocessing or for unsupervised machine learning.

3. Illustrative examples

With pyphysio, a typical signal processing pipeline to analyze the autonomic response to stimuli can be set in few lines of code (see Fig. 2). In this example, we process two signals: the Blood Volume Pulse (BVP) and the EDA [17], collected during an experiment in which the subject watched two 30-seconds length videos with different emotional content. Since the peaks in the BVP are associated to the cardiac activity, this signal is used to extract the distances between consecutive beats (Inter Beat Intervals, IBI) and then analyze the HRV, associated to the regulatory activity of the Autonomic Nervous System (ANS). The EDA signal is also associated to the ANS: the activity of the Sympathetic Nervous System (SNS), a branch of the ANS, increases the secretory activity of the sweat glands, thus causing increases of the electrical conductivity of the skin. The activity of the SNS, or phasic response, can be estimated from the EDA by a deconvolution process [13]. The first video, presented after 30 s of baseline elicited a calming response, the second elicited an aroused state, presented after 30 s of pause after the first video. We are interested in studying the different autonomic responses to the two videos.

The first step in the pipeline is creating the EvenlySignal instances that store the signal values and their temporal support (see Fig. 2A): it is sufficient to set the sampling frequency and the starting time of the acquisition. Note that more complex setups might use different sources of signals, thus allowing to separately specify the starting time is critical for synchronization of the different streams.

In the second step, we extract from the signals the information that is relevant for the autonomic response: the IBI from the BVP [16], and the phasic response from the EDA, associated with the SNS activity [9,13] (see Fig. 2B).

Then we define the parameters for the windowing and the type of autonomic indicators that should be computed on each signal. The windowing is a procedure in which the autonomic indicators are computed from portions of the signal and is a common practice to account for dynamics of the autonomic response, such as the response to a video stimulus. In this example (see Fig. 2C), we use partially overlapping (10 s overlap) windows of length 15 s and three time domain indicators of the HRV, and three of the peak patterns of the phasic response. Note that pyphysio includes a list of predefined indicators for each signal that can be used instead of the manually defined indicators (see Table 1).

Finally (see Fig. 2D), we use pyphysio to automatically compute the indicators from each segment. In this example, we observe (Fig. 2D) that during the second video the average length of the IBI (RRMean) decreases and the mean amplitude of the peaks (PKSMean) increases, suggesting that the second stimulus elicits a more aroused state.

A typical physiological signal processing pipeline includes steps for signal filtering and more complex use of the pyphysio algorithms, here omitted for the sake of simplicity. All parameters and their usage are described in the package Documentation.

pyphysio has already been used in research studies, for instance: to investigate the physiological response of adults to children crying [18,19] and social distress [20–22], to preprocess actigraphy signals for the automatic detection of stereotypical motor movements in Autism Spectrum Disorders [23] and to detect physiological response in dyadic interactions [24–26].

4. Impact and conclusions

The advent of data intensive experimental settings in psychophysiological studies is well supported by the availability of more compact devices, higher sampling rates, and the possibility of measuring multiple physiological signals. Big data analytics is thus now also a key task to consider, which applies to all data, from those produced from clinical instrument in lab settings to portable devices for flexible indoor studies up to wearable devices for in-the-wild applications. It is fair to expect that such scientific data assets will enable training Artificial Intelligence algorithms, and clinical applications later. It is thus urgent to fill the existing gap between better physiological sensing capabilities and the signal processing and analytics toolsets. As already acknowledged in bioinformatics, proprietary software is a limiting factor in the customization of algorithmic pipelines and a hurdle in supporting reproducibility, especially in a big data framework. On the other hand, pipelines that require the installation of multiple programming environments languages are hard to develop, maintain and reproduce.

Table 2
Selected algorithms implemented in `pyphysio`.

`pyphysio` function	Description	Notes
Filters.Normalize()	Rescale the signal according to various methods (e.g. standardize)	
Filters.IIRFilter()	Implementation of an Infinite Impulse Response Filter	Uses the nominal frequency instead of the relative frequency
Filters.FIRFilter()	Implementation of an Finite Impulse Response Filter	Uses the nominal frequency instead of the relative frequency
Filters.KalmanFilter()	Implementation of the Kalman's algorithm for signal filtering	Filter's parameters are inferred from the statistical properties of the input signal
Filters.ConvolutionalFilter()	Convolve a signal with a given Impulse Response Function	A pre-defined list of Impulse Response Functions for signal smoothing is provided
Filters.DeConvolutionalFilter()	Deconvolve a signal with a given Impulse Response Function	
Filters.ImputeNAN()	Impute missing values	Missing values are imputed with random values from the same distribution of the nearest values
Filters.RemoveSpikes()	Remove sudden changes in the signal values	Detects the spikes by analyzing the signal derivative
Estimators.BeatFromBP()	Detect the beat positions in a cardiac signal that measures the blood pulses	[16]
Estimators.BeatFromECG()	Detect the beat positions in an ECG	
Estimators.DriverEstim()	Deconvolve an EDA signal with a Bateman's function that represent the response of the sweat glands	[9]
Estimators.PhasicEstim()	Estimate the Phasic component from a signal of the sweat glands activity	[9]
Estimators.Energy()	Estimate the local energy of the signal	Uses a windowing with customizable window length to allow a localized estimation
SegmentsGenerators.FixedSegments()	Extract consecutive portions of signal with fixed length and overlap	
SegmentsGenerators.CustomSegments()	Extract consecutive portions of signal with custom start and stop timestamps	
SegmentsGenerators.LabelSegments()	Extract consecutive portions of signal from the values of an additional signal (e.g. stimulus presentation)	

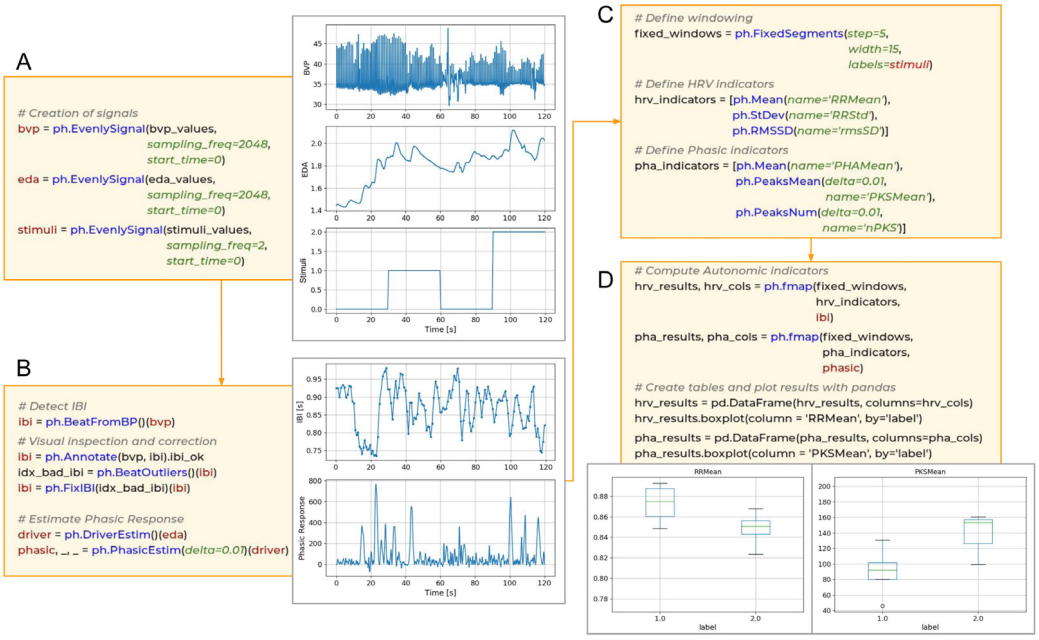

Fig. 2. Schematic representation of a typical physiological signal processing pipeline with `pyphysio`. (A) Definition of the signals with their temporal support; (B) Extraction of the information of interest from the signals (IBI from the BVP and phasic response from the EDA); (C) Definition of the windowing procedure and of the list of autonomic indicators; (D) Computation of the autonomic indicators and visualization of the results.

`pyphysio` aims at offering one common framework to work with different types of physiological signals. It requires minimal knowledge of Python and programming skills, and it is designed to facilitate scripting signal processing pipelines, where each step is implemented by an object of the class Algorithm. Further, `pyphysio` structure makes faster the integration of new algorithms, and of multiple physiological signals, such as EEG and functional Near Infrared Spectroscopy (fNIRS). Jupyter notebooks with additional examples for different signal types are available in the `pyphysio` repository.[3]

Notably, `pyphysio` is ready to use with the most used Machine Learning and Deep Learning environments, in particular

[3] https://github.com/MPBA/pyphysio

with scientific Python libraries, such as scikit-learn, keras, PyTorch. Researchers and developers are encouraged to adopt and contribute to the development of `pyphysio`, in particular by adding more algorithms and functions, for instance adding capabilities for real-time processing of streaming data.

Declaration of competing interest

The authors declare that there is no conflict of interest in this paper.

References

[1] Kreibig SD. Autonomic nervous system activity in emotion: A review. Biol. Psychol. 2010;84(3):394–421.
[2] Levenson RW. The autonomic nervous system and emotion. Emot. Rev. 2014;6(2):100–12.
[3] Pantelopoulos A, Bourbakis NG. A survey on wearable sensor-based systems for health monitoring and prognosis. IEEE Trans. Syst. Man Cybern C 2010;40(1):1–12.
[4] Blechert J, Peyk P, Liedlgruber M, Wilhelm FH. ANSLAB: Integrated multichannel peripheral biosignal processing in psychophysiological science. Behav Res Methods 2016;48(4):1528–45.
[5] Bao FS, Liu X, Zhang C. PyEEG: an open source python module for EEG/MEG feature extraction. Comput. Intell. Neurosci. 2011;2011.
[6] Bizzego A, Mina M, Zarbo C, Esposito G, Furlanello C. Physiolyze: a galaxy-based web service for heart rate variability analysis with online processing. In: 2014 8th Conference of the European Study Group on Cardiovascular Oscillations. IEEE; 2014, p. 97–8.
[7] Rodríguez-Liñares L, Lado M, Vila X, Méndez A, Cuesta P. GHRV: Heart rate variability analysis made easy. Comput Methods Programs Biomed 2014;116(1):26–38.
[8] Tarvainen MP, Niskanen J-P, Lipponen JA, Ranta-Aho PO, Karjalainen PA. Kubios HRV–heart rate variability analysis software. Comput Methods Programs Biomed 2014;113(1):210–20.
[9] Benedek M, Kaernbach C. A continuous measure of phasic electrodermal activity. J Neurosci Methods 2010;190(1):80–91.
[10] Bach DR. A head-to-head comparison of SCRalyze and Ledalab, two model-based methods for skin conductance analysis. Biol. Psychol. 2014;103:63–8.
[11] Greco A, Valenza G, Lanata A, Scilingo EP, Citi L. cvxEDA: A convex optimization approach to electrodermal activity processing. IEEE Trans Biomed Eng 2016;63(4):797–804.
[12] Malik M. Heart rate variability. Ann. Noninvasive Electrocardiol. 1996;1(2):151–81.
[13] Benedek M, Kaernbach C. Decomposition of skin conductance data by means of nonnegative deconvolution. Psychophysiology 2010;47(4):647–58.
[14] Koelstra S, Muhl C, Soleymani M, Lee J-S, Yazdani A, Ebrahimi T, Pun T, Nijholt A, Patras I. DEAP: A database for emotion analysis using physiological signals. IEEE Trans Affect Comput 2012;3(1):18–31.
[15] Casamassima F, Farella E, Benini L. Context aware power management for motion-sensing body area network nodes. In: Proceedings of the Conference on Design, Automation & Test in Europe. European Design and Automation Association; 2014, p. 170.
[16] Bizzego A, Furlanello C. DBD-RCO: Derivative based detection and reverse combinatorial optimization to improve heart beat detection for wearable devices, bioRxiv, 2017, p. 118943.
[17] Kushki A, Fairley J, Merja S, King G, Chau T. Comparison of blood volume pulse and skin conductance responses to mental and affective stimuli at different anatomical sites. Physiol Meas 2011;32(10):1529.
[18] Ozturk Y, Bizzego A, Esposito G, Furlanello C, Venuti P. Physiological and self-report responses of parents of children with autism spectrum disorder to children crying. Res. Dev. Disabil. 2018;73:31–9.
[19] Azhari A, Truzzi A, Rigo P, Bornstein MH, Esposito G. Putting salient vocalizations in context: Adults' physiological arousal to emotive cues in domestic and external environments. Physiol. Behav. 2018;196:25–32.
[20] Truzzi A, Poquérusse J, Setoh P, Shinohara K, Bornstein MH, Esposito G. Oxytocin receptor gene polymorphisms (rs53576) and early paternal care sensitize males to distressing female vocalizations. Dev. Psychobiol. 2018;60(3):333–9.
[21] Truzzi A, Bornstein MH, Senese VP, Shinohara K, Setoh P, Esposito G. Serotonin transporter gene polymorphisms and early parent-infant interactions are related to adult male heart rate response to female crying. Front. Physiol. 2017;8:111.
[22] Esposito G, Truzzi A, Setoh P, Putnick DL, Shinohara K, Bornstein MH. Genetic predispositions and parental bonding interact to shape adults' physiological responses to social distress. Behav. Brain Res. 2017;325:156–62.
[23] Rad NM, Bizzego A, Kia SM, Jurman G, Venuti P, Furlanello C. Convolutional neural network for stereotypical motor movement detection in autism. 2015, arXiv preprint arXiv:1511.01865.
[24] Truzzi A, Setoh P, Shinohara K, Esposito G. Physiological responses to dyadic interactions are influenced by neurotypical adults' levels of autistic and empathy traits. Physiol. Behav. 2016;165:7–14.
[25] Bizzego A, Gabrieli G, Azhari A, Setoh P, Esposito G. Computational methods for the assessment of empathic synchrony. In: 2019 29th Italian Workshop on Neural Networks. Smart Innovation Systems and Technologies, Springer; 2019.
[26] Calabrò G, Bizzego A, Cainelli S, Furlanello C, Venuti P. M-MS: a multimodal synchrony dataset to explore dyadic interaction in ASD. In: 2019 29th Italian Workshop on Neural Networks. Smart Innovation Systems and Technologies, Springer; 2019.