The Journal of Open Source Software

pybreathe: a python package for respiratory airflow rates analysis

Thibaut Coustillet [1]

1 Sorbonne Université, CNRS, INSERM, Development, Adaptation and Ageing, Dev2A, F-75005 Paris, France **2** Sorbonne Université, CNRS, Inserm, Institut de Biologie Paris-Seine, IBPS, F75005 Paris, France

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

Breathing is the vital function that enables air to flow into the lungs during inhalation and out during exhalation. Inhalation delivers (di)oxygen to the tissues, while exhalation flushes out carbon dioxide. In physiology, breathing is widely studied to investigate a whole range of respiratory diseases as well as physical abilities. Consistent and robust analysis is essential to cope with the large volume of data and the sometimes time-consuming routine analysis procedures. pybreathe is a python package that allows breathing to be formally analysed. For each of the two respiratory phases, pybreathe lets users to extract essential features such as the volume inhaled and exhaled, the inspiratory and expiratory times. It also provides the breathing frequency. The package has been designed to be solely based on the air flow rate. A set of methods specially designed to make analysis user-friendly requires only a user-supplied discretised air flow to be functional. The package comes with example scripts based on simulated breathing signals that are representative of the data that can be collected experimentally. They explain the milestones involved in carrying out an analysis (feature extraction). The input breathing signal can be obtained from any type of specialised software.

Statement of need

In 2017, half a billion people worldwide lived with a chronic respiratory disease (Soriano et al., 2020). Concurrently, the role of breathing has garnered growing attention in research focused on both athletic/sport performance (Contreras-Briceño et al., 2024; Harbour et al., 2022) and overall well-being (Fincham et al., 2023). Due to the large amount of respiratory data acquired over the last few years, numerous algorithms have been developed to structure analysis and open up new insights. Such tools complement proprietary analysis software used historically (Lusk et al., 2023). Most of the open source software available relies on peak detection (Bishop et al., 2022; Brammer, 2020) to extract signal features such as amplitude and breathing period. However, such detection methods often require human correction, manual curation or advanced algorithms to guarantee the accuracy of the results (Vanegas et al., 2020). On the other hand, some advanced algorithms use cutting-edge clustering methods to detect respiratory patterns that go beyond the features mentioned above (Germain et al., 2023). Although this kind of approach is particularly valuable for revealing different responses and adaptations to a wide range of experimental conditions and for providing a deeper understanding of respiratory physiology, it requires advanced programming skills and knowledge and may be complicated to set up in practice for non-computer users.

In this paper, we sought to implement an easy-to-use framework specially designed to faciliate respiratory analyses derived from air flow. pybreathe is a python package with an API designed to be used by both experimenters and developers. Users can acquire their respiratory data using any standard software. A necessary and sufficient condition for using pybreathe is to

export the data (instantaneous air flow rate) and discretise it into a classic text format (.txt, .csv, etc.). Most software used to measure respiratory air flow rates includes this functionality and outputs a two-column output file: the first column represents the discretised time, and the second contains the corresponding flow values at each time point.

The main difference with other analysis algorithms is that `pybreathe` is based on ventilatory flow and not on volume. Although the latter can be deduced from the former, it is generally flow rates themselves derived from pressure differences that are supplied (Criée et al., 2011). To our knowledge, there is no open-source algorithm that simply extracts elementary but essential features from air flow recordings. In the case of air flow rates, and as the main feature of a respiratory signal is the tidal volume (i.e., the volume passing through the lungs during a single breath), peak analysis cannot be applied because the amplitude (i.e., height) depends on the 'speed' at which the air flows in and out, i.e., for the same volume exhaled (or inhaled), the faster the exhalation (or inhalation), the greater the amplitude (Figure 1). In this situation, to really grasp the tidal volume, we need to get the Area Under the Curve (AUC) instead of the amplitude.

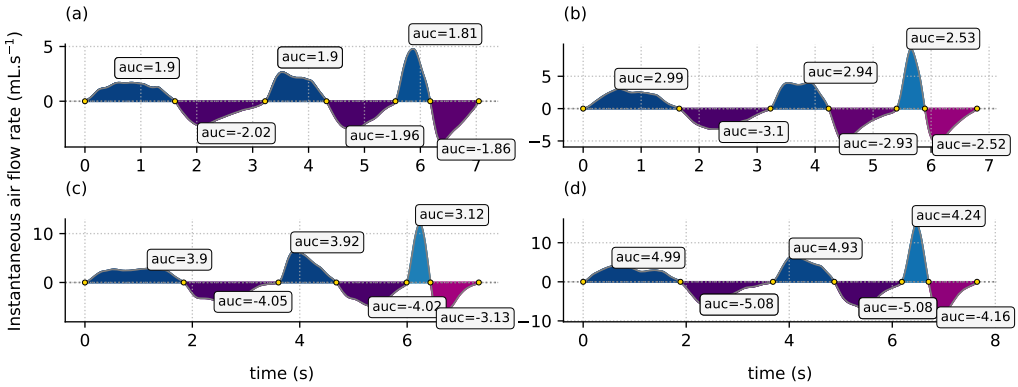

Figure 1: Manual injection/suction of different quantities of air into a chamber at three different speeds: slow, moderate and fast. (a) injection/suction of 2 mL; (b) injection/suction of 3 mL; (c) injection/suction of 4 mL; (d) injection/suction of 5 mL. Injection corresponds to the positive parts (blue) while suction corresponds to the negative parts (purple).

For a given respiratory signal, `pybreathe` detects zero-crossings and each positive segment will be either inhalation or exhalation (depending on the configuration of the primary acquisition software) and each negative segment will be the other phase. AUC (integral) of these segments therefore corresponds to the volume inhaled or exhaled. The duration of these segments (time between two zeros) corresponds to the inspiratory or expiratory time. Breathing frequency is also provided bases either on the frequency of peaks or hollows, or using a more sophisticaded spectral analysis.

The quantitative data for manually injected and suctioned volumes highlighting the relevance of focusing on volumes rather than ampliutdes (Figure 1) are shown in (Table 1).

Table 1: Comparison of the integral (Area Under the Curve) and amplitude (height) of several volumes of air manually injected/suctioned into a chamber.

actual volume	speed	positive integral	negative integral	positive amplitude	negative amplitude
≈ 2 mL	slow	1.90	- 2.02	1.70	- 2.26
≈ 2 mL	moderate	1.90	- 1.96	2.66	- 2.54
≈ 2 mL	fast	1.81	- 1.86	4.79	- 3.75

actual volume	speed	positive integral	negative integral	positive amplitude	negative amplitude
≈ 3 mL	slow	2.99	- 3.10	3.04	- 3.19
≈ 3 mL	moderate	2.94	- 2.93	3.98	- 5.07
≈ 3 mL	fast	2.53	- 2.52	9.24	- 5.2
≈ 4 mL	slow	3.90	- 4.05	2.91	- 3.71
≈ 4 mL	moderate	3.92	- 4.02	6.46	- 4.81
≈ 4 mL	fast	3.12	- 3.13	12.03	- 7.09
≈ 5 mL	slow	4.99	- 5.08	4.21	- 5.76
≈ 5 mL	moderate	4.93	- 5.08	6.58	- 6.70
≈ 5 mL	fast	4.24	- 4.16	14.88	- 9.04

For each quantity of air, the integral faithfully represents the volume injected and suctioned. The amplitude is not representative of the volume injected or suctioned. `pybreathe` is therefore capable of detecting the right volumes. Interestingly, regardless of the volume of air injected, high injection speeds consistently compromise measurement precision. This issue arises solely because the air is injected manually by the experimenter

pybreathe fundamentals

The `pybreathe` package is accompagnied by a 'BreathingFlow' object which is the core of the algorithm. Users should instantiate a BreathingFlow- object with their own data and use the associated methods.

```
from pybreathe import BreathingFlow

# Loading a discretised respiratory flow rate
my_signal = BreathingFlow.from_file(
  filename="path_to_your_discretised_flow",
  identifier="my data"
)
```

where `path_to_your_discretised_flow` is a two-column discretised file representing instantaneous airflow as a function of time (e.g., see Table 2).

Table 2: Discretised instantaneous airflow two-columns file used as input to instantiate an object of type 'BreathingFlow'. The file shows the first values of the sine function. User files should have the same configuration.

time	values
0.0	0.0650
0.004	0.0660
0.008	0.0671
0.012	0.0681
...	...

Example files are also supplied with the package. Users can either use the sinus function or two artificial breathing signals by instantiating them using the class method provided for this purpose:

```python
from pybreathe import BreathingFlow

# Sinus function
sinus = BreathingFlow.load_sinus()

# Artificial signal #1
example_01 = BreathingFlow.load_breathing_like_signal_01()

# Artificial signal #2
example_02 = BreathingFlow.load_breathing_like_signal_02()
```

Then, the five main methods for extracting features are:

```python
# Average duration of segments where the flow rate is positive.
my_signal.get_positive_time()

# Average duration of segments where the flow rate is negative.
my_signal.get_negative_time()

# Average area under the curve of segments where the flow rate is positive.
my_signal.get_positive_auc()

# Average area under the curve of segments where the flow rate is negative.
my_signal.get_negative_auc()

# Breathing frequency.
my_signal.get_frequency()
```

Proof

To ensure that the package worked correctly and that the methods returned the desired output, we checked the volumes extracted by pybreathe when we injected/suctioned known quantities of air (Figure 1 and Table 1). Because the manual injection/suction of air into a chamber can be imprecise due to the experimenter and the precision of the syringes, we also created an artifical respiratory signal corresponding to the sine function (Figure 2). Based on the sine, we checked that the signal features obtained with pybreathe were indeed the same as those obtained *mathematically*.

Let f the sinus function.

$$\begin{aligned} f : \mathbb{R} &\longrightarrow \mathbb{R} \\ x &\longmapsto \sin(x) \end{aligned}$$

The $\sin$ function is 2π-periodic, i.e.,

$$\forall x \in \mathbb{R}, \quad \sin(x + 2\pi) = \sin(x)$$

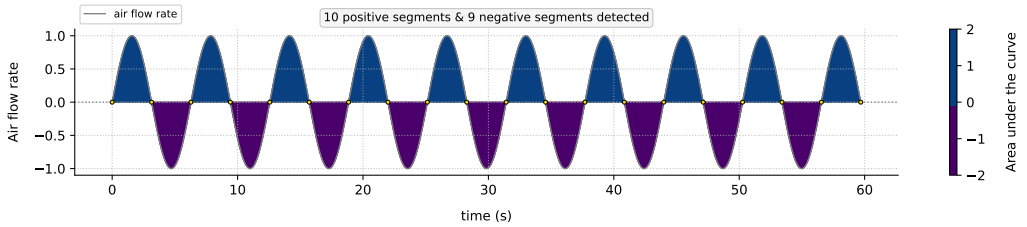

Figure 2: Graph of the sine function on the interval $[0, 10\pi]$.

Positive time (~ expiratory time)

Statement: the mean length of the interval where the sine function is positive is exactly equal to π.

Mathematical proof

Let $x \in \mathbb{R}$.

$\sin(x) \geq 0 \iff x \bmod 2\pi \in [0, \pi]$

Thus, the duration (interval length) of all positive segments is $\pi - 0$, which is equal to π.

pybreathe validation

```
>>> from pybreathe import BreathingFlow
>>> sinus = BreathingFlow.load_sinus()
>>> sinus.get_positive_time()
mean = 3.14 ± 9.19e-10 (n = 10).
(3.14, 9.19e-10, 10)
```

Negative time (~ inspiratory time)

In the same way, we can demonstrate that the average duration of negative segments is also π, which is also found with pybreathe.

```
>>> from pybreathe import BreathingFlow
>>> sinus = BreathingFlow.load_sinus()
>>> sinus.get_negative_time()
mean = 3.14 ± 7.43e-10 (n = 9).
(3.14, 7.43e-10, 9)
```

Positive Area Under the Curve (~ exhaled volume)

Statement: the mean AUC (integral) of the positive segments of the sinus function is exactly 2.

Mathematical proof

Let $x \in \mathbb{R}$.

$\sin(x) \geq 0 \iff x \bmod 2\pi \in [0, \pi]$

$$\begin{aligned}\int_0^{\pi} \sin x \, dx &= [-\cos x]_0^{\pi} \\ &= -\cos(\pi) - (-\cos(0)) \\ &= -\cos(\pi) + \cos(0) \\ &= -(-1) + 1 \\ &= 1 + 1 \\ &= 2\end{aligned}$$

Thus, the mean area under the curve of all positive segments is 2.

pybreathe validation

```
>>> from pybreathe import BreathingFlow
>>> sinus = BreathingFlow.load_sinus()
>>> sinus.get_positive_auc()
mean = 2.0 ± 8.4e-12 (n = 10).
(2.0, 8.4e-12, 10)
```

Negative Area Under the Curve (~ inhaled volume)

In the same way, we can demonstrate that the average area under the curve of negative segments is exactly -2, which is also found with pybreathe.

```
>>> from pybreathe import BreathingFlow
>>> sinus = BreathingFlow.load_sinus()
>>> sinus.get_negative_auc()
mean = -2.0 ± 7.68e-12 (n = 9).
(-2.0, 7.68e-12, 9)
```

Acknowledgements

This work was supported by SATT Lutech. Special thanks are due to Eugénie Faure and Alexandre Palazzi for their valuable feedback throughout the development of pybreathe.

References

Bishop, M., Weinhold, M., Turk, A. Z., Adeck, A., & SheikhBahaei, S. (2022). An open-source tool for automated analysis of breathing behaviors in common marmosets and rodents. *eLife*, *11*, e71647. https://doi.org/10.7554/eLife.71647

Brammer, J. (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

Contreras-Briceño, F., Cancino, J., Espinosa-Ramírez, M., Fernández, G., Johnson, V., & Hurtado, D. E. (2024). Estimation of ventilatory thresholds during exercise using respiratory wearable sensors. *Npj Digital Medicine*, *7*(1). https://doi.org/10.1038/s41746-024-01191-9

Criée, C. P., Sorichter, S., Smith, H. J., Kardos, P., Merget, R., Heise, D., Berdel, D., Köhler, D., Magnussen, H., Marek, W., Mitfessel, H., Rasche, K., Rolke, M., Worth, H., & Jörres, R. A. (2011). Body plethysmography – its principles and clinical use. *Respiratory Medicine*, *105*(7), 959–971. https://doi.org/10.1016/j.rmed.2011.02.006

Fincham, G. W., Strauss, C., Montero-Marin, J., & Cavanagh, K. (2023). Effect of breathwork on stress and mental health: A meta-analysis of randomised-controlled trials. *Scientific Reports*, *13*(1). https://doi.org/10.1038/s41598-022-27247-y

Germain, T., Truong, C., Oudre, L., & Krejci, E. (2023). Unsupervised classification of plethysmography signals with advanced visual representations. *Frontiers in Physiology*, *14*, 1154328. https://doi.org/10.3389/fphys.2023.1154328

Harbour, E., Stöggl, T., Schwameder, H., & Finkenzeller, T. (2022). Breath tools: A synthesis of evidence-based breathing strategies to enhance human running. *Frontiers in Physiology*, *13*. https://doi.org/10.3389/fphys.2022.813243

Lusk, S., Ward, C. S., Chang, A., Twitchell-Heyne, A., Fattig, S., Allen, G., Jankowsky, J. L., & Ray, R. S. (2023). An automated respiratory data pipeline for waveform characteristic analysis. *The Journal of Physiology*, *601*(21), 4767–4806. https://doi.org/10.1113/JP284363

Soriano, J. B., Kendrick, P. J., Paulson, K. R., Gupta, V., Abrams, E. M., Adedoyin, R. A., Adhikari, T. B., Advani, S. M., Agrawal, A., Ahmadian, E., Alahdab, F., Aljunid, S. M., Altirkawi, K. A., Alvis-Guzman, N., Anber, N. H., Andrei, C. L., Anjomshoa, M., Ansari, F., Antó, J. M., … Vos, T. (2020). Prevalence and attributable health burden of chronic respiratory diseases, 1990–2017: A systematic analysis for the global burden of disease study 2017. *The Lancet Respiratory Medicine*, *8*(6), 585–596. https://doi.org/10.1016/S2213-2600(20)30105-3

Vanegas, E., Igual, R., & Plaza, I. (2020). Sensing Systems for Respiration Monitoring: A Technical Systematic Review. *Sensors (Basel, Switzerland)*, *20*(18), 5446. https://doi.org/10.3390/s20185446