NeuroKit2: A Python Toolbox for Neurophysiological Signal Processing

Dominique Makowski [1,*], Tam Pham [1], Zen J. Lau [1], Jan C. Brammer [2], Francois Lespinasse [3], Hung Pham [4], Christopher Schölzel [5], & S.H. Annabel Chen [1, 6, 7]

[1] School of Social Sciences, Nanyang Technological University, Singapore

[2] Donders Institute and Behavioural Science Institute, Radboud University, Netherlands

[3] Departement de psychologie, Universite de Montreal, Canada

[4] Eureka Robotics, Singapore

[5] Life Science Informatics, THM University of Applied Sciences, Germany

[6] Centre for Research and Development in Learning, Nanyang Technological University, Singapore

[7] Lee Kong Chian School of Medicine, Nanyang Technological University, Singapore

Author Note

* Correspondence concerning this article should be addressed to Dominique Makowski (HSS 04-18, 48 Nanyang Avenue, Singapore; dmakowski@ntu.edu.sg).

Abstract

NeuroKit2 is an open-source, community-driven and user-friendly Python package dedicated to neurophysiological signal processing (e.g., ECG, EDA, EMG, …). Its design philosophy, which is centred on user-experience and a collaborative environment, makes it accessible to both novice and advanced users. The package provides general functions allowing for data processing and analysis in a few lines of code using validated pipelines, as well as tools dedicated at specific processing steps, offering flexibility and fine-tuned control for advanced users. NeuroKit2 aims at improving transparency and reproducibility in neurophysiological research, as well as being a scaffolding for exploration and innovation.

Keywords: Neurophysiology, Biosignals, Python, ECG, EDA, EMG

Word count:

NeuroKit2: A Python Toolbox for Neurophysiological Signal Processing

Cognitive neuroscience and psychology are increasingly relying on neurophysiological methods to assess brain and bodily activity. These approaches include electroencephalography (EEG), electrocardiography (ECG), electromyography (EMG) and electrodermal activity (EDA) signals. This trend was driven not only by theoretical motivations (e.g., the growth of embodied or affective neuroscience; Kiverstein & Miller, 2015), but also by practical reasons including low monetary cost (especially compared with other imaging techniques, such as MRI or MEG), high user convenience (e.g., portability, setup speed), and the increasing availability of recording devices (e.g., in smart watches; Yuehong, Zeng, Chen, & Fan, 2016). Together with the development of recording tools, advancements in the fields of signal processing and computational data science bolstered the emergence of new processing algorithms (Clifton, Gibbons, Davies, & Tarassenko, 2012; Roy et al., 2019), in turn offering a myriad of novel methods for users to process and analyze neurophysiological signals.

Unfortunately, because most of the algorithms are implemented as code, neurophysiological data processing remains a challenge for many researchers without a formal training or experience in programming. Moreover, many existing implementations are also limited to one type of signals (for instance, focused on ECG or EDA), which makes it inconvenient for researchers who might have to learn and concurrently rely on different software to process multimodal data.

NeuroKit2 aims at addressing these challenges by offering a free, user-friendly, and comprehensive solution for neurophysiological data processing. It is an open-source Python package, developed in a collaborative environment that continues to welcome contributors from different countries and fields. Historically, *NeuroKit2* is the re-forged successor *NeuroKit1* (*https://github.com/neuropsychology/NeuroKit.py*), a PhD side project that ended up benefiting an important community of users (252 GitHub stars as of 03-05-2020). The new version takes on its best features and design choices, and re-implements them in a

professional and well-thought way. It aims at being 1) accessible and well-documented, 2) reliable and cutting-edge, and 3) flexible and efficient.

Being available for Python 3 (Van Rossum & Drake, 2009), one of the most popular programming languages, *NeuroKit2's* users benefit from an important amount of existing tutorials and a large online community. The package is also relatively lightweight, using mainly standard dependencies (Virtanen et al., 2020) such as *NumPy*, *pandas*, *SciPy*, *scikit-learn* and *MatplotLib* (with an additional system of optional dependencies), enabling its use as a dependency in other software. The package source code is available under a permissive license on GitHub (*https://github.com/neuropsychology/NeuroKit*). Its documentation, automatically built and rendered from the code, is hosted at *https://neurokit2.readthedocs.io/*. Apart from guides for installation and contribution, and a description of the package's functions, the documentation also includes several "hands-on" examples and tutorials providing a walk-through on how to address specific issues (e.g., how to extract and visualize individual heartbeats, how to analyze event-related data). New examples can be easily added by users simply by uploading a Python notebook file (Kluyver et al., 2016) to the repository. This file will be automatically transformed into a webpage and displayed on the website, ensuring a transparent and evolutive documentation. Moreover, these examples can be used interactively via a cloud-based *Binder* environment (Jupyter et al., 2018), allowing users to try out the features directly in their browser. Finally, the accessibility for newcomers is reinforced by the issue tracker of GitHub, where users can create public issues to inquire for help.

The package aims at being reliable and trustworthy, including peer-reviewed processing pipelines and functions tested against existing implementations of established reference software such as *BioSPPy* (Carreiras et al., 2015), *hrv under review*, *PySiology* (Gabrieli, Azhari, & Esposito, 2019), *HeartPy* (Gent, Farah, Nes, & Arem, 2019), *systole* (Legrand & Allen, 2020) or *nolds* (Schölzel, 2020). The code itself includes a comprehensive test suite using continuous integration tools [e.g., travis CI] to ensure stability and prevent errors. Moreover,

users are able to easily report any bugs and be notified of their fixes via the issue tracker. Thanks to its collaborative and open development as well as its modular organization, *NeuroKit2* can easily follow the latest developments and remain cutting-edge through its ability to evolve, adapt, and integrate new methods as they are emerging.

Finally, we believe that the design philosophy of *NeuroKit2* contributes to an efficient (i.e., allowing to achieve a lot with few functions) yet flexible (i.e., enabling fine control and precision over what is done) user interface (API). We will illustrate these claims with two examples of common use-cases (the analysis of event-related and resting state data), and will conclude by discussing how *NeuroKit2* contributes to neurophysiological research by raising the standards for validity, reproducibility and accessibility.

Design Philosophy

As stated above, *NeuroKit2* aims at being accessible to beginners and, at the same time, offering a maximal level of control to experienced users. This is achieved by allowing beginning users to implement complex processing and analyses pipelines with very few functions, while still enabling fine-tuned control and precision to more experienced users. In concrete terms, this trade-off is allowed by the implementation of three abstract levels of functions.

Low-level: Base Utilities for Signal Processing

The basic building blocks are functions to facilitate general signal processing, i.e., to do filtering, resampling, interpolating, peak detection, etc. These functions are signal-agnostic, and include a lot of tweakable parameters. For instance, one can change the filtering method, frequencies, order and such. Most of these functions are based on validated algorithms present in *scipy* (Virtanen et al., 2020). Examples of such functions include `signal_filter()`, `signal_interpolate()`, `signal_resample()`, `signal_detrend()`, and `signal_findpeaks()`.

Mid-level: Neurophysiological Processing Steps

The signal processing utilities are then used by functions specific to the different types of physiological signals (i.e., ECG, RSP, EDA, EMG, PPG). These functions aim at taking care of specific steps of physiological data processing, such as cleaning, peak detection, phase classification or rate computation. Critically, for each type of signals, the same function names are called (in the form `signaltype_functiongoal()`) to achieve equivalent goals, e.g., `*_clean()`, `*_findpeaks()`, `*_process()`, `*_plot()`, making it intuitive and consistent across different modalities.

For example, the `rsp_clean()` function uses `signal_filter()` and `signal_detrend()`, with different possible sets of default parameters that can be switched via a "method" argument (corresponding to different published or validated pipelines). For instance, setting `method="khodadad2018"` will use the cleaning workflow described in Khodadad et al. (2018). However, if a user wants to build its own custom cleaning pipeline, she or he can use the cleaning function as a template, and directly tweak the parameters in the low-level signal processing operations.

High-level Wrappers for Processing and Analysis

These steps are then assembled in "master" functions, that are usually the entry point for new users. For instance, the `ecg_process()` function uses `ecg_clean()`, `ecg_findpeaks()`, `ecg_rate()`. A general processing pipeline can be selected via the `method` argument, that is then propagated throughout its lower-level functions. Easily switching between processing pipelines allows for the comparison of different methods, and streamlines critical but time-consuming steps in reproducible research, such as the validation of data preparation and quality control (Quintana, Alvares, & Heathers, 2016). Finally, the package includes convenience meta-functions (e.g., `bio_process`) that enable the combined processing of multiple types of signals at once (e.g., `bio_process(ecg=ecg_signal, eda=eda_signal)`).

Performing a set of operations with sensible default parameters can be rewarding, especially for beginners, allowing them to perform cutting-edge processing or replication of research steps without requiring a programming expertise. Moreover, it contributes to the demystification of the usage of "pure" programming tools (as opposed to GUI-based software such as *SPSS*, *Kubios*, or *Acqknowledge*), providing a welcoming framework to further understand the frontend, backend and the in-betweens of physiological data processing. Importantly, more advanced users can again very easily build their own custom analysis pipeline by using the mid-level functions, allowing for a finer control over the processing parameters.

Overall, we believe that this code structure offers a calibrated trade-off between flexibility and user-friendliness, with functions that are easy to memorize and implement. We hope that it may further encourage researchers to become part of a supportive open-science community construed of many expertises - rather than relying on closed and proprietary software - to achieve their goals.

Example

In this section, we present two examples that illustrate the most common use-cases. The first example is an event-related paradigm, in which the interest lies in the momentarily short-term physiological changes related to specific stimuli, whereas the second shows how to extract the characteristics (features) of physiological activity during a longer period of time (not necessarily tied to a specific and sudden event). The example datasets are made available with the package and can be downloaded using the `data()` function.

Event-related Paradigm

This example dataset contains ECG, RSP and EDA signals of one participant to whom were presented four emotional images (from the NAPS database; Marchewka, Żurawski, Jednoróg, & Grabowska, 2014), in a typical (albeit highly shortened) experimental psychology

paradigm.

Signals are 2.5 minutes long and are recorded at a frequency of 100Hz (note that the sampling rate is low for storage purposes and should be higher in actual recordings, see Quintana et al., 2016). It has 4 channels including three physiological signals, and one corresponding to the marking of events via a photosensor (which signal decreased when a stimulus appeared on the screen).

```
# Load the package
import neurokit2 as nk


# Download the example dataset
data = nk.data("bio_eventrelated_100hz")


# Figure 1. Visualize 10 seconds of data (on the same scale)
nk.signal_plot(data[900:1900], standardize=True)


# Process the data
df, info = nk.bio_process(ecg=data["ECG"],
                          rsp=data["RSP"],
                          eda=data["EDA"],
                          sampling_rate=100)


# Find events
conditions = ["Negative", "Neutral", "Neutral", "Negative"]
events = nk.events_find(event_channel=data["Photosensor"],
                        threshold_keep='below',
                        event_conditions=conditions)
```

```
# Epoch the data
epochs = nk.epochs_create(data=df,
                          events=events,
                          sampling_rate=100,
                          epochs_start=-0.1,
                          epochs_end=4)

# Extract event related features
results = nk.bio_analyze(epochs)

# Show subset of results
results[["Condition", "ECG_Rate_Mean", "RSP_Rate_Mean", "EDA_Peak_Amplitude"]]
```

Table 1

Subset of the ouput related to event-related analysis characterizing the pattern of physiological changes related to specific stimuli.

Condition	ECG_Rate_Mean	RSP_Rate_Mean	EDA_Peak_Amplitude
Negative	-1.94	-0.22	None
Neutral	-4.36	1.57	None
Neutral	1.02	-0.30	None
Negative	-3.61	2.22	1.68

In this example, after loading the package and the example dataset, each physiological signal is processed using `bio_process()`. Data from the photosensor is processed separately with `events_find()`, that locates the stimuli onsets in the signal. Once we have preprocessed signals and the location of events, we can slice the data into segments corresponding to a time window (ranging from -0.1 to 4 seconds) around each stimulus with `epochs_create()`.

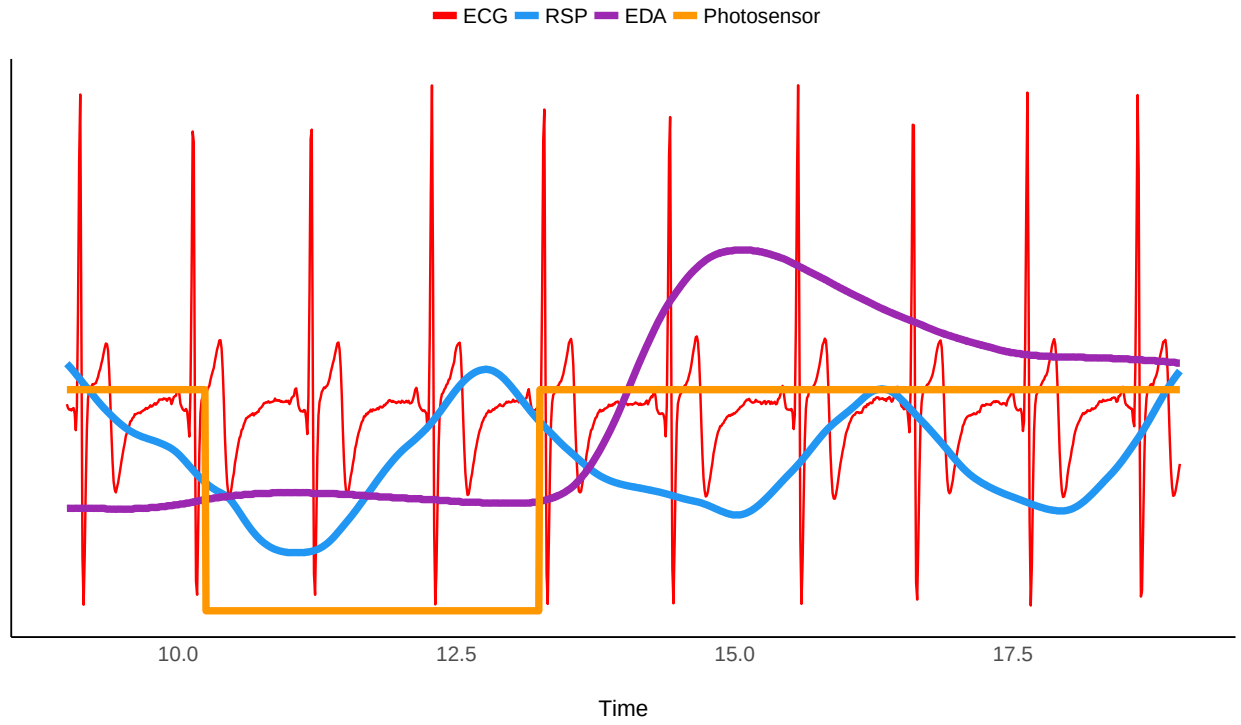

Figure 1. Subset of the dataset showing one event (in orange) and the other physiological signals.

Finally, relevant features are computed for each epoch (i.e., each stimulus) by providing them to `bio_analyze()`.

The features include for example the changes in rates of ECG and RSP signals (e.g. maximum, minimum and mean rate after stimulus onset, and the time at which they occur), and the peak characteristics of EDA signal (e.g., occurrence of skin conductance response (SCR), and if SCR is present, its corresponding peak amplitude, time of peak, rise and recovery time). In addition, respiration and cardiac cycle phases are also extracted (i.e., the respiration phase - inspiration/expiration - and cardiac phase - systole/diastole - occurring at the onset of event).

This example shows the straightforward process of extracting features of physiological responses. This pipeline can easily scale up to group-level analysis by aggregating the average of features across subjects. In addition to streamlining data analyses, *NeuroKit2* aims to

provide researchers an extensive suite of signal features, allowing for precise interpretations in terms of relationship between physiological activity and neurocognitive processes. In this example (see **Table 1**), exposure to negative stimuli, as compared to neutral stimuli, is related to stronger cardiac deceleration, higher skin conductance response, and accelerated breathing rate (note that this illustrative interpretation is purely descriptive).

Resting-state Features

The second dataset corresponds to 5 minutes physiological activity of a human participant at rest (eyes-closed in a seated position), under no specific set of instructions. It contains three channels (ECG, PPG and RSP) sampled at a frequency of 100Hz.

```
# Load the package
import neurokit2 as nk


# Download the example dataset
data = nk.data("bio_resting_5min_100hz")


# Process the data
df, info = nk.bio_process(ecg=data["ECG"],
                          rsp=data["RSP"],
                          sampling_rate=100)


# Extract features
results = nk.bio_analyze(df)


# Show subset of results
results[["ECG_Rate_Mean", "ECG_HRV_RMSSD", "RSP_Rate_Mean", "RSA_P2T_Mean"]]
```

Table 2

Subset of properties characterizing the physiological activity over a period of 5 minutes of resting-state.

ECG_Rate_Mean	ECG_HRV_RMSSD	RSP_Rate_Mean	RSA_P2T_Mean
86.42	4.28	15.86	0.01

In this example, the steps of the analysis are in fact identical to the previous example, including loading the package, the dataset and processing the data. The difference is that here there is no epoching, as we want to compute features related to the whole dataset. Thus, we can directly pass the dataframe to `bio_analyze()`, which will detect that these are not epochs, and compute the appropriate features accordingly. These include for instance the average heart and breathing rate, as well as indices of heart rate variability (HRV) and respiratory sinus arrhythmia (RSA).

This example illustrates a second type of physiological analysis, that we could refer to as interval-related (as opposed to event-related), where one is typically interested in computing features of signal variability and activation patterns over a longer-term period or interval of time (typically more than a few seconds, as is generally the case in event-related paradigms). The simplicity of usage of *NeuroKit2* allows for the fast creation of a standardized and reproducible pipeline to describe physiological activity.

Discussion

NeuroKit2 is a neurophysiological signal processing software accessible to people with very little knowledge of programming, due to its design choices focusing on user-experience and collaborative community. It is also a pragmatic answer to the broader need for transparent and reproducible methods in neurophysiology. Its modular organization not only facilitates the use of existing and validated processing pipelines, but is also an viable platform for

experimentation and innovation.

We expect future evolution to be mostly driven by the community and the advances in related fields. Possible directions include extending the support for other types of bodily signals (e.g., electrogastrography - EGG, electrooculography - EOG) and strengthening the efficiency of the code to obtain performance gains for large datasets. Further validation of the available processing pipelines could be made through the (re)analysis of public databases. In line with this objective, the support of standardized data structure formats (e.g. WFDB, BIDS, …) could be extended.

In conclusion, we think *NeuroKit2* provides useful tools not only for novice and senior researchers, but amateurs and tech-enthusiasts interested in health or embodied aspects of (neuro)psychology. Whether the data is produced by "smart health devices" or academic research-grade equipment, the package foster reproducible science. Critically, it is a community-oriented project. By increasing the autonomy of researchers and practitioners, and by shortening the delay between data collection and results acquisition, *NeuroKit2* could be useful beyond fundamental research in neuroscience and psychology, including applications such as biofeedback, personal physiological monitoring and sport research.

Conflict of Interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Acknowledgements

We would like to thank Prof. C. F. Xavier for inspiration, all the contributors (https://neurokit2.readthedocs.io/en/latest/authors.html), and the users for their support.

References

Carreiras, C., Alves, A. P., Lourenço, A., Canento, F., Silva, H., Fred, A., & others. (2015). BioSPPy: Biosignal processing in Python. Retrieved from https://github.com/PIA-Group/BioSPPy/

Clifton, D. A., Gibbons, J., Davies, J., & Tarassenko, L. (2012). Machine learning and software engineering in health informatics. In *2012 first international workshop on realizing ai synergies in software engineering (raise)* (pp. 37–41). IEEE.

Gabrieli, G., Azhari, A., & Esposito, G. (2019). PySiology: A python package for physiological feature extraction. In *Neural approaches to dynamics of signal exchanges* (pp. 395–402). Springer Singapore. https://doi.org/10.1007/978-981-13-8950-4_35

Gent, P. van, Farah, H., Nes, N. van, & Arem, B. van. (2019). HeartPy: A novel heart rate algorithm for the analysis of noisy signals. *Transportation Research Part F: Traffic Psychology and Behaviour*, *66*, 368–378. https://doi.org/10.1016/j.trf.2019.09.015

Jupyter, Bussonnier, Forde, Freeman, Granger, Head, … Willing. (2018). Binder 2.0 - Reproducible, interactive, sharable environments for science at scale. In Fatih Akici, David Lippa, Dillon Niederhut, & M. Pacer (Eds.), *Proceedings of the 17th Python in Science Conference* (pp. 113–120). https://doi.org/%2010.25080/Majora-4af1f417-011%20

Khodadad, D., Nordebo, S., Mueller, B., Waldmann, A., Yerworth, R., Becher, T., … others. (2018). Optimized breath detection algorithm in electrical impedance tomography. *Physiological Measurement*, *39*(9), 094001.

Kiverstein, J., & Miller, M. (2015). The embodied brain: Towards a radical embodied cognitive neuroscience. *Frontiers in Human Neuroscience*, *9*, 237.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B. E., Bussonnier, M., Frederic, J., … others. (2016). Jupyter notebooks-a publishing format for reproducible computational

workflows. In *ELPUB* (pp. 87–90).

Legrand, N., & Allen, M. (2020). Systole: A python toolbox for preprocessing, analyzing, and synchronizing cardiac data. Retrieved from https://github.com/embodied-computation-group/systole

Marchewka, A., Żurawski, Ł., Jednoróg, K., & Grabowska, A. (2014). The nencki affective picture system (naps): Introduction to a novel, standardized, wide-range, high-quality, realistic picture database. *Behavior Research Methods*, *46*(2), 596–610.

Quintana, D., Alvares, G. A., & Heathers, J. (2016). Guidelines for reporting articles on psychiatry and heart rate variability (graph): Recommendations to advance research communication. *Translational Psychiatry*, *6*(5), e803–e803.

Roy, Y., Banville, H., Albuquerque, I., Gramfort, A., Falk, T. H., & Faubert, J. (2019). Deep learning-based electroencephalography analysis: A systematic review. *Journal of Neural Engineering*, *16*(5), 051001.

Schölzel, C. (2020). NOnLinear measures for dynamical systems (nolds). Retrieved from https://github.com/CSchoel/nolds

Van Rossum, G., & Drake, F. L. (2009). *Python 3 reference manual*. Scotts Valley, CA: CreateSpace.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., … Contributors, S. 1. 0. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/https://doi.org/10.1038/s41592-019-0686-2

Yuehong, Y., Zeng, Y., Chen, X., & Fan, Y. (2016). The internet of things in healthcare: An overview. *Journal of Industrial Information Integration*, *1*, 3–13.