Behavior Research Methods (2021) 53:1689–1696
https://doi.org/10.3758/s13428-020-01516-y

NeuroKit2: A Python toolbox for neurophysiological signal processing

Dominique Makowski[1] · Tam Pham[1] · Zen J. Lau[1] · Jan C. Brammer[2] · François Lespinasse[3,4] · Hung Pham[5] · Christopher Schölzel[6] · S. H. Annabel Chen[1,7,8]

Accepted: 19 November 2020 / Published online: 2 February 2021

Abstract

NeuroKit2 is an open-source, community-driven, and user-centered Python package for neurophysiological signal processing. It provides a comprehensive suite of processing routines for a variety of bodily signals (e.g., ECG, PPG, EDA, EMG, RSP). These processing routines include high-level functions that enable data processing in a few lines of code using validated pipelines, which we illustrate in two examples covering the most typical scenarios, such as an event-related paradigm and an interval-related analysis. The package also includes tools for specific processing steps such as rate extraction and filtering methods, offering a trade-off between high-level convenience and fine-tuned control. Its goal is to improve transparency and reproducibility in neurophysiological research, as well as foster exploration and innovation. Its design philosophy is centred on user-experience and accessibility to both novice and advanced users.

Keywords Neurophysiology · Biosignals · Python · ECG · EDA · EMG

Neurophysiological measurements increasingly gain popularity in the study of cognition and behavior. These measurements include electroencephalography (EEG), electrocardiography (ECG), electromyography (EMG) and electrodermal activity (EDA). Their popularity is driven by theoretical motivations (e.g., the growth of embodied or affective neuroscience; Kiverstein and Miller,: 2015) as well as practical reasons. The latter include low costs (especially compared with other imaging techniques, such as MRI or MEG), ease of use (e.g., portability, setup speed), and the increasing availability of recording devices (e.g., wearables; Yuehong et al., 2016). Moreover, the extraction of meaningful information from neurophysiological signals is facilitated by current advances in signal processing algorithms (Clifton et al., 2012; Roy et al., 2019). Unfortunately, these algorithms are often not distributed in a usable way (i.e., in the form of packaged code) which makes them inaccessible to researchers who do not have the time or experience to implement them. Moreover, many software tools for neurophysiological analyses are limited to a single type of signal (for instance ECG). This makes it inconvenient for researchers who might have to concurrently rely on a number of software packages to process and analyze multimodal data.

Additionally, psychology and neuroscience face a "reproducibility crisis" (Maizey & Tzavella, 2019; Miłkowski et al., 2018; Nosek et al., 2015; Topalidou et al., 2015) which has lead to a profound reassessment of research practices (by researchers, publishers, funding agencies, etc.). The opacity of data processing, such as ill-specified, or inaccessible analysis pipelines, plays a major role in the crisis. This issue could in part be alleviated by making analyses code an integral part of scientific publications, rather than

✉ Dominique Makowski
dmakowski@ntu.edu.sg

1 School of Social Sciences, Nanyang Technological University, HSS 04-19, 48 Nanyang Avenue, Singapore, Singapore

2 Behavioural Science Institute, Radboud University, Nijmegen, Netherlands

3 Département de psychologie, Université de Montréal, Montréal, Canada

4 Centre de Recherche de l'Institut Universitaire Geriatrique de Montréal, Montréal, Canada

5 Eureka Robotics, Singapore, Singapore

6 Life Science Informatics, THM University of Applied Sciences, Giessen, Germany

7 Centre for Research and Development in Learning, Nanyang Technological University, Singapore, Singapore

8 Lee Kong Chian School of Medicine, Nanyang Technological University, Singapore, Singapore

treating a paper as the sole and most important part of the research project. However, distributing the analysis script alongside the paper poses new challenges: Scripts must be shareable (not always feasible with closed-source and proprietary software or programming languages), accessible (well-documented and organized scripts) and reproducible (which is difficult for software relying on graphical user interfaces - GUI - in which the manual point-and-click sequence can be hard to automate).

NeuroKit2, addresses these challenges by offering a free, user-centered, and comprehensive solution for neurophysiological data processing, with an initial focus on bodily signals including ECG (electrocardiography is used to measure cardiac activity), PPG (photoplethysmogram is an optical measurement of blood flow), RSP (respiration measures), EDA (electrodermal activity measuring the electrical conductance of the skin), EMG (electromyography measuring muscular activity) and EOG (electrooculography measuring eye movements). It also provides modality-independent functions that can be used for other modalities such as EEG (electroencephalography measuring electrical activity of the brain), for which more specific support is in development.

The open-source Python package is developed by a multi-disciplinary team that actively invites new collaborators. The target audience of *NeuroKit2* includes both experienced and novice programmers. Although being a programming-based tool, users not familiar with Python or other languages can start using the software (and improve their programming skills along the way) by following our step-by-step examples. Moreover, we also include a thorough tutorial on Python installation, as well as a "10 minutes introduction to Python" in the documentation. While many of the existing software caters to a single signal modality (e.g., *KUBIOS*©(Tarvainen et al., 2014), *HeartPy* (van Gent et al., 2019) and *pyHRV* (Gomes et al., 2019) for ECG, *cvxEDA* (Greco et al., 2015), *Ledalab* (Benedek & Kaernbach, 2010), and *SCRalyze* (Bach, 2014) for EDA), *NeuroKit2* provides support for various signals and allows its users to process signals from multiple physiological modalities with a uniform application programming interface (API). It aims at being accessible, well-documented, well-tested, cutting-edge, flexible and efficient. The library allows users to select from a wide range of validated analysis pipelines and to create custom pipelines tailored to specific analyses requirements. Historically, *NeuroKit2* is the re-forged successor of *NeuroKit "1"* (Makowski, 2020), taking over its most successful features and design choices, and re-implementing them in a way that adheres to current best practices in open source software development.

NeuroKit2 offers a breadth of functionalities which includes, but is not limited to, signal simulation; data management (e.g., downloading existing datasets, reading and formatting files into a dataframe); event extraction from signals; epoch extraction, signal processing (e.g., filtering, resampling, rate computation using different published algorithms detailed in the package's documentation); spectral analyses; complexity and entropy analyses; and convenient statistical methods (e.g., K-means clustering, ICA or PCA). A variety of plotting functions allow for quick and expressive visualization of the signal processing and the resulting features.

The package is implemented in Python 3 (Van Rossum & Drake, 2009), which means that *NeuroKit2's* users benefit from an large number of learning resources and a vibrant community. The package depends on relatively few, well established and robust packages from the numeric Python ecosystem such as *NumPy* (Harris et al., 2020), *pandas* (McKinney & et al. 2010), *SciPy* (Virtanen et al., 2020), *scikit-learn* (Pedregosa et al., 2011) and *MatplotLib* (Hunter, 2007) (with an additional system of optional dependencies), making *NeuroKit2* a viable dependency for other packages.

NeuroKit2's source code is available under the MIT license on GitHub (https://github.com/neuropsychology/NeuroKit). Its documentation (https://neurokit2.readthedocs.io/) is automatically built and rendered from the code and includes guides for installation and contribution, a description of the package's functions, as well as several "hands-on" examples and tutorials (e.g., how to extract and visualize individual heartbeats, how to analyze event-related data etc.). Importantly, users can add new examples by simply submitting a Jupyter notebook (Kluyver et al., 2016) to the GitHub repository. The notebook will automatically be displayed on the website, ensuring easily accessible and evolving documentation. Moreover, users can try out the example notebooks directly in their browser via a cloud-based *Binder* environment (Jupyter et al., 2018). Finally, the issue tracker on GitHub offers a convenient and public forum that allows users to report bugs, get help and gain insight into the development of the package. Our active collaborators range from academics, professionals and practitioners in the life sciences and engineering fields (See the "authors" section on the package's documentation). Based on community feedback that we received (social networks, GitHub issues), *NeuroKit2* has attracted users of different profiles, ranging from those who are new to signal processing and programming to more experienced users.

NeuroKit2 aims at being reliable and trustworthy, including implementations of processing pipelines that have been described in peer-reviewed publications. Details and references regarding those pipelines are available in the package's documentation. Many pipelines have been tested against established software such as *BioSPPy* (Carreiras et al., 2015), *hrv* (Bartels & Pecanha, 2020), *PySiology* (Gabrieli et al., 2019), *HeartPy* (van Gent et al., 2019),

systole (Legrand & Allen, 2020) or *nolds* (Schölzel, 2019). Additionally, the repository leverages a comprehensive test suite (using *pytest*) and continuous integration (using Travis-CI and GitHub actions) to ensure software stability and quality. The test coverage and build status can transparently be tracked via the GitHub repository. Thanks to its collaborative and open development, *NeuroKit2* can remain cutting-edge and continuously 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) UI. We will illustrate these claims with two examples of common use-cases (the interval-related analysis on resting state data and the event-related analysis), and will conclude by discussing how *NeuroKit2* contributes to neurophysiological research by raising the standards for validity, reproducibility and accessibility.

Design philosophy

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 pipelines with a few functions, while still providing experienced users with fine-tuned control over arguments and parameters. In concrete terms, this trade-off is enabled by an API structure organized in three layers.

Low-level: Base utilities for signal processing

The basic building blocks are functions for general signal processing, i.e., filtering, resampling, interpolation, peak detection, etc. These functions are modality-independent, and include several parameters (e.g., one can change the filtering method, frequencies, and order, by overwriting the default arguments). Most of these functions are based on established algorithms implemented 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 base utilities are used by mid-level functions specific to the different physiological modalities (i.e., ECG, RSP, EDA, EMG, PPG). These functions carry out modality-specific signal processing steps, such as cleaning, peak detection, phase classification or rate computation. Critically, for each type of signal, uniform function names are used (in the form `signaltype_functiongoal()`) to achieve equivalent goals, e.g., `*_clean()`, `*_findpeaks()`, `*_process()`, `*_plot()`, making the implementation intuitive and consistent across different modalities.

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

High-level wrappers for processing and analysis

The mid-level functions are assembled in high-level wrappers, that are convenient entry points for new users. For instance, the `ecg_process()` function internally chains the mid-level functions `ecg_clean()`, `ecg_peaks()`,`ecg_quality()`,`ecg_delineate()`, and `ecg_phase()`, as shown in Fig. 1. A specific processing pipeline can be selected with the `method` argument that is then propagated throughout the internal 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 et al., 2016). Finally, the package includes convenience-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 an entire set of operations with sensible default parameters in one function can be rewarding, especially for beginners, allowing them to perform cutting-edge processing or replication of research steps without requiring much programming expertise. Moreover, it contributes to the demystification of the usage of programming tools (as opposed to GUI-based software such as *SPSS*, *Kubios*, or *Acqknowledge*), providing a welcoming framework to further explore physiological data processing. Importantly, more advanced users can build custom analysis pipelines by using the low- and mid-level functions, allowing for finer control over the processing parameters. We believe that this implementation is a well-calibrated trade-off between flexibility and user-friendliness.

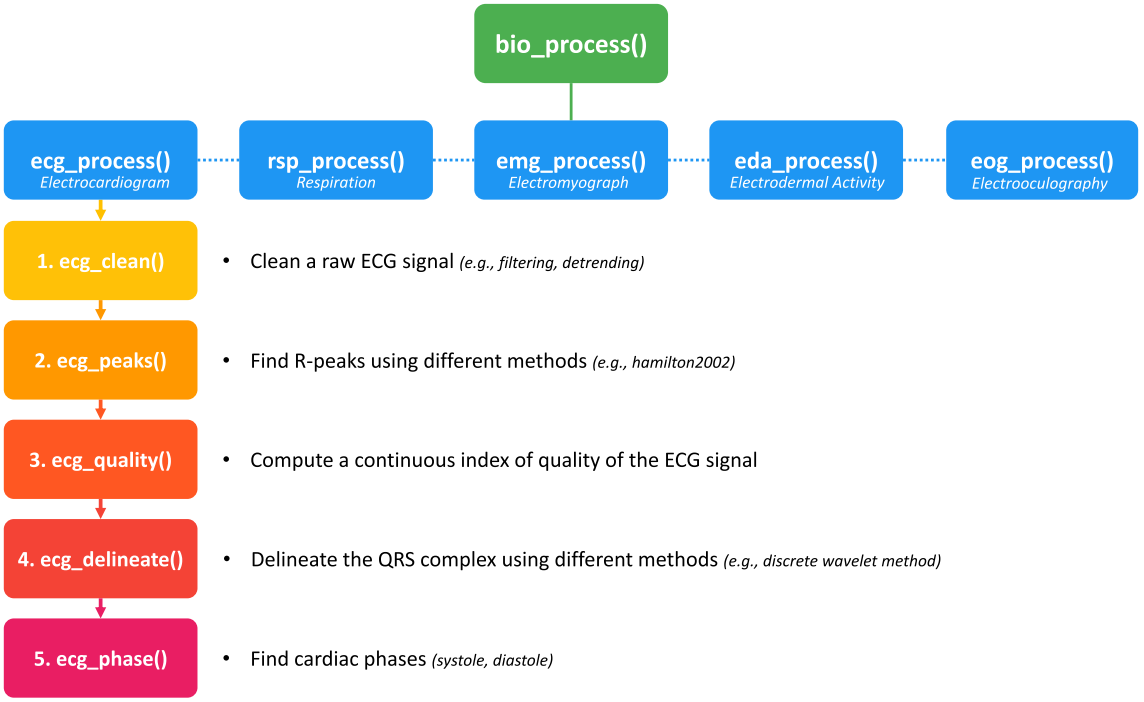

Fig. 1 Illustration of the NeuroKit2 package architecture, in the case of ECG signal processing

Installing NeuroKit2

NeuroKit2 is available on PyPI, a repository of software for the Python programming language. and can be installed using pip (via "*pip install neurokit2*" command). Detailed instructions on how to install Python are also available in the *installation* section of the package's documentation.

Examples

In this section, we present two examples that illustrate the most common use-cases (Fig. 2). Both examples can be accessed in an interactive format (without any prior installation) via a Binder environment. The first example illustrates an interval-related paradigm where characteristics of physiological activity during a certain time interval (not necessarily tied to a specific and sudden event) are extracted. The second example presents an event-related paradigm, in which the interest lies in shorter-term physiological changes related to specific events (see Fig. 1 and Table 1). The example datasets are available with the package and can be downloaded using the `data()` function. This utility reads comma separated values files (*.csv*) with the Pandas function *pd.read_csv()*, where each column is a different biosignal. Each row is a sample that correspond to signals' value at a given point in time. All examples use the 0.0.41 version release of *NeuroKit2*.

Interval-related paradigm

The first dataset corresponds to 5 minutes of physiological activity of a human participant at rest (eyes-closed in a seated position), with no specific instructions. It contains

Table 1 Examples of features computed in different domains

Interval-related Features	Event-related Features
ECG Rate Characteristics (Mean, Amplitude)	ECG Rate Changes (Min, Mean, Max, Time of Min, Max, Trend)
Heart Rate Variability (HRV) indices	RSP Rate Changes (Min, Mean, Max, Time of Min, Max)
Respiratory Rate Variability (RRV) indices	RSP Amplitude Measures (Min, Mean, Max)
Respiratory Sinus Arrhythmia (RSA) indices	ECG and RSP Phase (Inspiration/Expiration, Systole/Diastole, Completion)
Number of SCR Peaks and mean amplitude	SCR peak and its characteristics (amplitude, rise time, recovery time)

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, sampling_rate=100)

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

Here, the aim was to illustrate a type of physiological analysis that we refer to as interval-related (or resting-state paradigm, as opposed to an event-related paradigm). After loading the package and the example dataset, each physiological signal is processed using `bio_process()`. As we want to compute features related to the entire dataset (see Table 2), we can directly pass the whole dataframe to `bio_analyze()`, and compute the interval-related features. Users can choose a specific time interval from their dataset.

Interval-related analyses compute features of signal variability and activation patterns over a given period of time, including average heart and breathing rate, as well as indices of heart rate variability (HRV) and respiratory sinus arrhythmia (RSA). *NeuroKit2* allows for the fast creation of standardized and reproducible pipelines to describe this kind of physiological activity.

Event-related Paradigm

This example dataset contains ECG, RSP and EDA signals of one participant who was presented with four emotional images (from the NAPS database; Marchewka et al., 2014) in a typical (albeit shortened) experimental psychology paradigm.

The signals are 2.5 minutes (150 seconds) long and are recorded at a frequency of 100Hz (note that the sampling rate is lower than usually required, see Quintana et al. (2016), in order to be able to include the example data in the *NeuroKit2* distribution). It has 4 channels including three physiological signals, and one corresponding to events marked with a photosensor (signal strength decreases when a stimulus appears on the screen).

```
# Load the package
import neurokit2 as nk

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

# 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, sampling_rate=100)

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

In this example, the steps of the analysis are identical to the previous example, including loading the package, the dataset and processing the data. The difference is that stimulus onsets in the photosensor are detected separately with `events_find()`. Once we have the preprocessed signals and the location of events, we use `epochs_create()` to slice the data into segments corresponding to a time window (ranging from -0.1 to 4 seconds) around each stimulus. Finally, relevant features are computed for each epoch (i.e., each stimulus) by passing them to `bio_analyze()`.

Table 2 Subset of properties characterizing the physiological activity over a period of 5 minutes of resting-state

ECG_Rate_Mean	HRV_RMSSD	RSP_Rate_Mean	RSA_P2T_Mean
86.39	38.84	15.74	0.07

Table 3 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	−2.01	−0.15	0.93
Neutral	−3.13	1.40	0.41
Neutral	1.34	−0.34	0.02
Negative	−3.55	1.97	1.06

Notably, the features include the changes in rate 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 the 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 extracted (i.e., the respiration phase - inspiration/expiration - and cardiac phase - systole/diastole - occurring at the onset of event).

We hope that these examples demonstrate how straightforward the process of extracting features of physiological responses can be with *NeuroKit2*. This pipeline can easily scale up to group-level analyses by aggregating the average of features across participants. In addition to streamlining data analyses, *NeuroKit2* aims to allow researchers to extract an extensive suite of features that can be linked to neurocognitive processes. In this example (see Table 3), 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 descriptive interpretation is given solely for illustrative purposes).

Discussion

NeuroKit2 is a neurophysiological signal processing library accessible to people across different levels of programming experience and backgrounds. For users who are novice programmers or are new to neurophysiology, the package presents an ideal opportunity for exploration and learning. The experienced programmer is encouraged to choose and validate the preprocessing and analysis pipelines most appropriate for their data. Suggestions for improvements or additions to the library are welcome and openly discussed in the community. Overall, the development of *NeuroKit2* is focused on creating an intuitive user-experience, as well

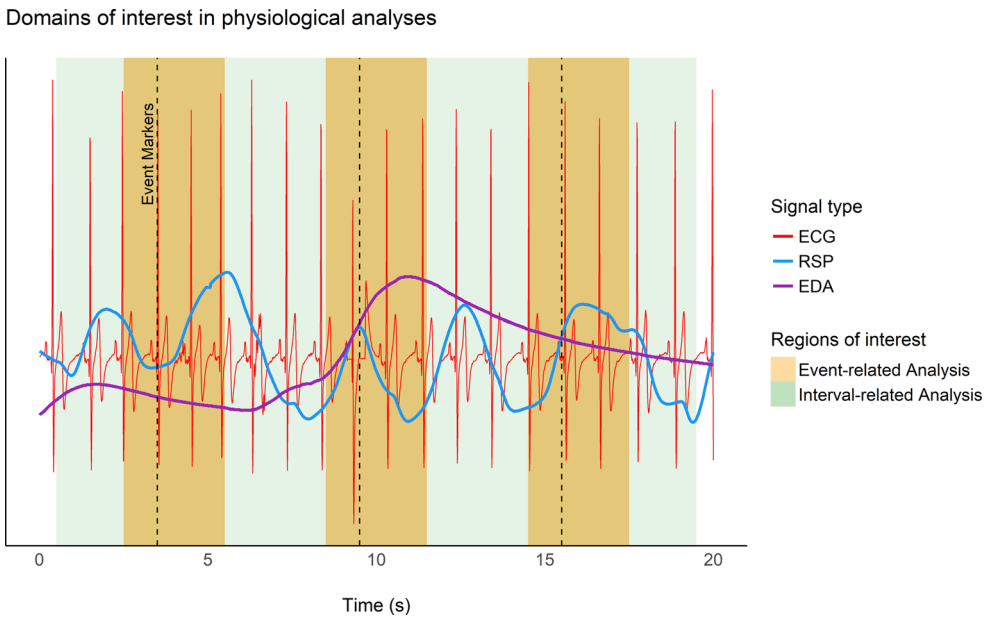

Fig. 2 Plot window displaying a period of raw electrocardiogram (ECG in red), respiration (RSP in blue) and electrodermal activity (EDA in purple) data. The green highlighted section, spanning from 0 to 20s, represents the periodic region of interest during interval-related analysis. The 3 event markers are indicated by dotted lines, and the orange highlighted sections spanning 0.1s before the onset of each event and ending 4s after the event, represent periodic regions of interest during event-related analysis. The link for generating the figure can be found on NeuroKit2's GitHub repository (https://github.com/neuropsychology/NeuroKit/blob/master/paper/make_figures.Rmd)

as building a collaborative community. Its modular structure and organization not only facilitate the use of existing and validated processing pipelines, but also create a fertile ground for experimentation and innovation.

The library is also a pragmatic answer to the broader need for transparent and reproducible methods in neurophysiology. The impact of our package on reproducibility in research is two-fold: firstly, while black-box software can be easy and convenient to use, users do not have access to the source code, making processing results subject to unknown idiosyncrasies of the underlying implementation of processing routines. This makes it difficult to identify the source of potential discrepancies in results obtained with other software and can lead to irreproducible findings. In contrast, *NeuroKit2* documents each step of the implementation along with the analysis method, allowing users to pin-point the analysis steps where differences might arise. While maintaining a focus on overall user-experience, the open-source nature of *NeuroKit2* encourages independent researchers to cross-validate research findings. Secondly, not only does *NeuroKit2* implement several methods for analysis, it also allows for the comparison of different algorithms. For instance, using a suite of open-source databases, different algorithms for ECG R-peak detection have been compared for their robustness (number of errors encountered), efficiency (computation time) and accuracy (absolute distance from true R-peak location), documented in the "Studies" section of the package's documentation. As *NeuroKit2* continues to work on benchmarking, we hope to support users in making more informed decisions regarding which method is most suited for their specific requirements.

NeuroKit2 also prioritizes a high standard of quality control during code development. This is done through automated testing using continuous integration, as well as striving for code simplicity and readability. The API is thoroughly documented, including working examples. We ensure that the documentation evolves alongside the code by including it in our continuous integration. While *NeuroKit2* currently has a fairly comprehensive documentation, more examples and tutorials will be added as the package grows and expands. Additionally, we provide thorough guidelines for new contributors who wish to contribute code or documentation.

We expect the package's future evolution to be driven by the communities' needs and the advances in related fields. For instance, although *NeuroKit2* already implements a lot of useful functions for EEG processing (such as entropy and fractal dimensions quantification), its support could be further improved (for example with high-level functions built on top of utilities provided by the leading EEG Python software, namely *MNE*, Gramfort et al. (2013). Additionally, in the future we strive to support other types of bodily signals (e.g., electrogastrography - EGG, electrooculography - EOG) and plan to optimize computational efficiency on large datasets. We also plan to further validate the available processing pipelines using public databases. In line with this objective, the support of standardized data structure formats (e.g. WFDB, BIDS, . . .) could be extended.

In conclusion, we believe that *NeuroKit2* provides useful tools for anyone who is interested in analyzing physiological data collected with research-grade hardware or wearable "smart health devices". By increasing the autonomy of researchers and practitioners, and by shortening the delay between data collection and results acquisition, *NeuroKit2* could be useful beyond academic research in neuroscience and psychology, including applications such as personal physiological monitoring and exercise science. Finally, we hope that *NeuroKit2* encourages users to become part of a supportive open-science community with diverse areas of expertise rather than relying on closed-source and proprietary software, thus shaping the future of neurophysiology and its related fields.

Acknowledgments We would like to thank Prof. C. F. Xavier for inspiration, all the current and future contributors (https://neurokit2.readthedocs.io/en/latest/authors.html), and the users for their support. Additionally, François Lespinasse would like to thank the Courtois Foundation for its support through the Courtois-NeuroMod project (https://cneuromod.ca)

Compliance with Ethical Standards

Conflict of interests The authors declare that the research was conducted in the absence of commercial or financial relationships that could constitute a conflict of interest.

References

Bach, D. R. (2014). A head-to-head comparison of scralyze and ledalab, two model-based methods for skin conductance analysis. *Biological Psychology*, *103*, 63–68.

Bartels, R., & Pecanha, T. (2020). HRV: A pythonic package for heart rate variability analysis. *Journal of Open Source Software*, *5*(51), 1867. https://doi.org/10.21105/joss.01867

Benedek, M., & Kaernbach, C. (2010). A continuous measure of phasic electrodermal activity. *Journal of Neuroscience Methods*, *190*(1), 80–91.

Carreiras, C., Alves, A. P., Lourenço, A., Canento, F., Silva, H., Fred, A., & et al (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. *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

Gomes, P., Margaritoff, P., & Silva, H. (2019). pyHRV: Development and evaluation of an open-source python toolbox for heart rate

variability (hrv). *Proc. Int'l conf On electrical, electronic and computing engineering (icetran)*, 822–828.

Gramfort, A., Luessi, M., Larson, E., Engemann, D. A., Strohmeier, D., Brodbeck, C., & et al. (2013). MEG and eeg data analysis with mne-python. *Frontiers in Neuroscience*, *7*, 267.

Greco, A., Valenza, G., Lanata, A., Scilingo, E. P., & Citi, L. (2015). CvxEDA: A convex optimization approach to electrodermal activity processing. *IEEE Transactions on Biomedical Engineering*, *63*(4), 797–804.

Harris, C. R., Millman, K. J., Van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., & et al. (2020). Array programming with numpy. *Nature*, *585*(7825), 357–362.

Hunter, J. D. (2007). Matplotlib: a 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95.

Jupyter, B., Forde, F., Granger, H. W., Akici, F., Lippa, D., Niederhut, D., & Pacer, M. (2018). Binder 2.0 - Reproducible, interactive, sharable environments for science at scale. In *Proceedings of the 17th Python in Science Conference*. https://doi.org/%2010.25080/Majora-4af1f417-011%20, (pp. 113–120).

Khodadad, D., Nordebo, S., Mueller, B., Waldmann, A., Yerworth, R., Becher, T., & et al. (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., & et al. (2016). Jupyter notebooks-a publishing format for reproducible computational workflows. *ELPUB*, 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

Maizey, L., & Tzavella, L. (2019). Barriers and solutions for early career researchers in tackling the reproducibility crisis in cognitive neuroscience. *Cortex*, *113*, 357–359.

Makowski, D. (2020). Neurokit: A python toolbox for statistics and neurophysiological signal processing (eeg, eda, ecg, emg...). Retrieved from https://github.com/neuropsychology/NeuroKit.py

Marchewka, A., Żurawski, J. 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.

McKinney, W., et al. (2010). Data structures for statistical computing in python. *Proceedings of the 9th python in science conference* (vol. 445, pp. 51–56). Austin.

Miłkowski, M., Hensel, W. M., & Hohol, M. (2018). Replicability or reproducibility? on the replication crisis in computational neuroscience and sharing only relevant detail. *Journal of Computational Neuroscience*, *45*(3), 163–172.

Nosek, B. A., Cohoon, J., Kidwell, M., & Spies, J. R. (2015). Estimating the reproducibility of psychological science. *Science*, *349*(6251), aac4716.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

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. (2019). *Nonlinear measures for dynamical systems*. Zenodo. https://doi.org/10.5281/zenodo.3814723

Tarvainen, M. P., Niskanen, J.-P., Lipponen, J. A., Ranta-Aho, P. O., & Karjalainen, P. (2014). A Kubios hrv–heart rate variability analysis software. *Computer Methods and Programs in Biomedicine*, *113*(1), 210–220.

Topalidou, M., Leblois, A., Boraud, T., & Rougier, N. (2015). P A long journey into reproducible computational neuroscience. *Frontiers in Computational Neuroscience*, *9*, 30.

van Gent, P., Farah, H., van Nes, N., & van Arem, B. (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

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

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., & Contributors, S. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods, 17*, 261–272. 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.