1   **NeuroKit2: A Python Toolbox for Neurophysiological Signal Processing**

2   Dominique Makowski [1,*], Tam Pham [1], Zen J. Lau [1], Jan C. Brammer [2], Francois

3   Lespinasse [3], Hung Pham [4], Christopher Schölzel [5], & S.H. Annabel Chen [1, 6, 7]

4   [1] School of Social Sciences, Nanyang Technological University, Singapore

5   [2] Behavioural Science Institute, Radboud University, Netherlands

6   [3] Departement de psychologie, Universite de Montreal, Canada

7   [4] Eureka Robotics, Singapore

8   [5] Life Science Informatics, THM University of Applied Sciences, Germany

9   [6] Centre for Research and Development in Learning, Nanyang Technological University,

10   Singapore

11   [7] Lee Kong Chian School of Medicine, Nanyang Technological University, Singapore

12   Author Note

15   Abstract

16   NeuroKit2 is an open-source, community-driven, and user-friendly Python package

17 dedicated to neurophysiological signal processing (e.g., ECG, EDA, EMG, …). Its design

18 philosophy is centred on user-experience and accessibility to both novice and advanced

19 users. The package provides general functions that enable data processing in a few lines of

20 code using validated pipelines, as well as tools dedicated to specific processing steps,

21 offering a trade-off between efficiency and fine-tuned control. NeuroKit2 aims at improving

22 transparency and reproducibility in neurophysiological research, as well as fostering

23 exploration and innovation.

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

25   Word count: 2270

NeuroKit2: A Python Toolbox for Neurophysiological Signal Processing

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 & 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, Zeng, Chen, & Fan, 2016). Moreover, the extraction of meaningful information from neurophysiological signals is facilitated by current advances in signal processing algorithms (Clifton, Gibbons, Davies, & Tarassenko, 2012; Roy et al., 2019).

Unfortunately, these algorithms are mostly inaccessible to researchers without experience in programming and signal processing. Moreover, many software tools for neurophysiological analyses are limited to one type of signal (for instance, focused on ECG). This makes it inconvenient for researchers who might have to learn and concurrently rely on different software to process multimodal data.

NeuroKit2 addresses these challenges by offering a free, user-friendly, and comprehensive solution for neurophysiological data processing. It is an open-source Python package, developed by a multi-disciplinary team that actively invites new collaborators. It aims at being accessible, well-documented, well-tested, cutting-edge, flexible and efficient. Historically, *NeuroKit2* is the re-forged successor *NeuroKit1*, taking on its most successful features and design choices, and re-implementing them in a professional and well-thought way.

The package is implemented in Python 3 (Van Rossum & Drake, 2009), which means that *NeuroKit2's* users benefit from an large amount of learning resources and a vibrant community. The package depends on relatively few, well established and robust packages from the Python data analysis ecosystem (Virtanen et al., 2020) such as *NumPy*, *pandas*, *SciPy*,

52 *scikit-learn* and *MatplotLib* (with an additional system of optional dependencies), making
53 *NeuroKit2* itself a viable dependency in other software.

54 *NeuroKit2's* source code is available under the permissive MIT license on GitHub (*https://*
55 *github.com/neuropsychology/NeuroKit*). Its documentation (*https://neurokit2.readthedocs.*
56 *io/*) is automatically built and rendered from the code and includes guides for installation
57 and contribution, a description of the package's functions, as well as several "hands-on"
58 examples and tutorials (e.g., how to extract and visualize individual heartbeats, how to ana-
59 lyze event-related data etc.). Importantly, users can add new examples by simply uploading
60 a Python notebook (Kluyver et al., 2016) to the GitHub repository. The notebook will
61 automatically be displayed on the website, ensuring easily accessible and evolving documen-
62 tation. Moreover, users can try out the example notebooks directly in their browser via a
63 cloud-based *Binder* environment (Jupyter et al., 2018). Finally, the issue tracker on GitHub
64 offers a convenient and public forum that allows newcomers and potential collaborators to
65 report bugs, get help and gain insight into the development of the package.

66 *NeuroKit2* aims at being reliable and trustworthy, including peer-reviewed processing pipelines
67 and functions tested against established software such as *BioSPPy* (Carreiras et al., 2015),
68 *hrv under review*, *PySiology* (Gabrieli, Azhari, & Esposito, 2019), *HeartPy* (Gent, Farah,
69 Nes, & Arem, 2019), *systole* (Legrand & Allen, 2020) or *nolds* (Schölzel, 2020). The repos-
70 itory leverages a comprehensive test suite and continuous integration to ensure stability
71 and prevent errors. Thanks to its collaborative and open development, *NeuroKit2* can re-
72 main cutting-edge and continuously evolve, adapt, and integrate new methods as they are
73 emerging.

74 Finally, we believe that the design philosophy of *NeuroKit2* contributes to an efficient (i.e.,
75 allowing to achieve a lot with few functions) yet flexible (i.e., enabling fine control and
76 precision over what is done) user interface (API). We will illustrate these claims with two
77 examples of common use-cases (the analysis of event-related and resting state data), and will

78 conclude by discussing how *NeuroKit2* contributes to neurophysiological research by raising

79 the standards for validity, reproducibility and accessibility.

Design Philosophy

81 As stated above, *NeuroKit2* aims at being accessible to beginners and, at the same time,

82 offering a maximal level of control to experienced users. This is achieved by allowing begin-

83 ning users to implement complex processing and analyses pipelines with very few functions,

84 while still enabling fine-tuned control and precision over arguments and parameters to more

85 experienced users. In concrete terms, this trade-off is allowed by a API structure organized

86 in three three layers of abstraction.

Low-level: Base Utilities for Signal Processing

88 The basic building blocks are functions for general signal processing, i.e., filtering, resam-

89 pling, interpolation, peak detection, etc. These functions are signal-agnostic, and include

90 a lot of parameters (e.g., one can change the filtering method, frequencies, and order, by

91 overwriting the default arguments). Most of these functions are based on established algo-

92 rithms implemented in *scipy* (Virtanen et al., 2020). Examples of such functions include

93 `signal_filter()`, `signal_interpolate()`, `signal_resample()`, `signal_detrend()`, and

94 `signal_findpeaks()`.

Mid-level: Neurophysiological Processing Steps

96 The base utilities are used by mid-level functions specific to the different physiological modal-

97 ities (i.e., ECG, RSP, EDA, EMG, PPG). These functions carry out modality-specific signal

98 processing steps, such as cleaning, peak detection, phase classification or rate computa-

99 tion. Critically, for each type of signal, the same function names are called (in the form

100 `signaltype_functiongoal()`) to achieve equivalent goals, e.g., `*_clean()`, `*_findpeaks()`,

101 `*_process()`, `*_plot()`, making the implementation intuitive and consistent across different
102 modalities.

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

High-level Wrappers for Processing and Analysis

111 The mid-level functions are assembled in high-level "master" functions, that are convenient
112 entry points for new users. For instance, the `ecg_process()` function internally chains
113 the mid-level functions `ecg_clean()`, `ecg_findpeaks()`, `ecg_rate()`. A specific processing
114 pipeline can be selected with the `method` argument, that is then propagated throughout the
115 internal functions. Easily switching between processing pipelines allows for the compari-
116 son of different methods, and streamlines critical but time-consuming steps in reproducible
117 research, such as the validation of data preparation and quality control (Quintana, Al-
118 vares, & Heathers, 2016). Finally, the package includes convenience meta-functions (e.g.,
119 `bio_process`) that enable the combined processing of multiple types of signals at once (e.g.,
120 `bio_process(ecg=ecg_signal, eda=eda_signal)`).

121 Performing an entire set of operations with sensible default parameters in one function can
122 be rewarding, especially for beginners, allowing them to perform cutting-edge processing or
123 replication of research steps without requiring much programming expertise. Moreover, it
124 contributes to the demystification of the usage of "pure" programming tools (as opposed to
125 GUI-based software such as *SPSS*, *Kubios*, or *Acqknowledge*), providing a welcoming frame-

work to further explore the complexities of physiological data processing. Importantly, more advanced users can easily build custom analysis pipelines by using the mid-level functions, allowing for a finer control over the processing parameters. We believe that this implementation is a well-calibrated trade-off between flexibility and user-friendliness.

Examples

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 short-term physiological changes related to specific events. The second example shows how to extract the characteristics 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 who was presented with 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 with a photosensor (which signal decreases when a stimulus appeared on the screen).

```
# Load the package
import neurokit2 as nk
```

```python
# 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)
```
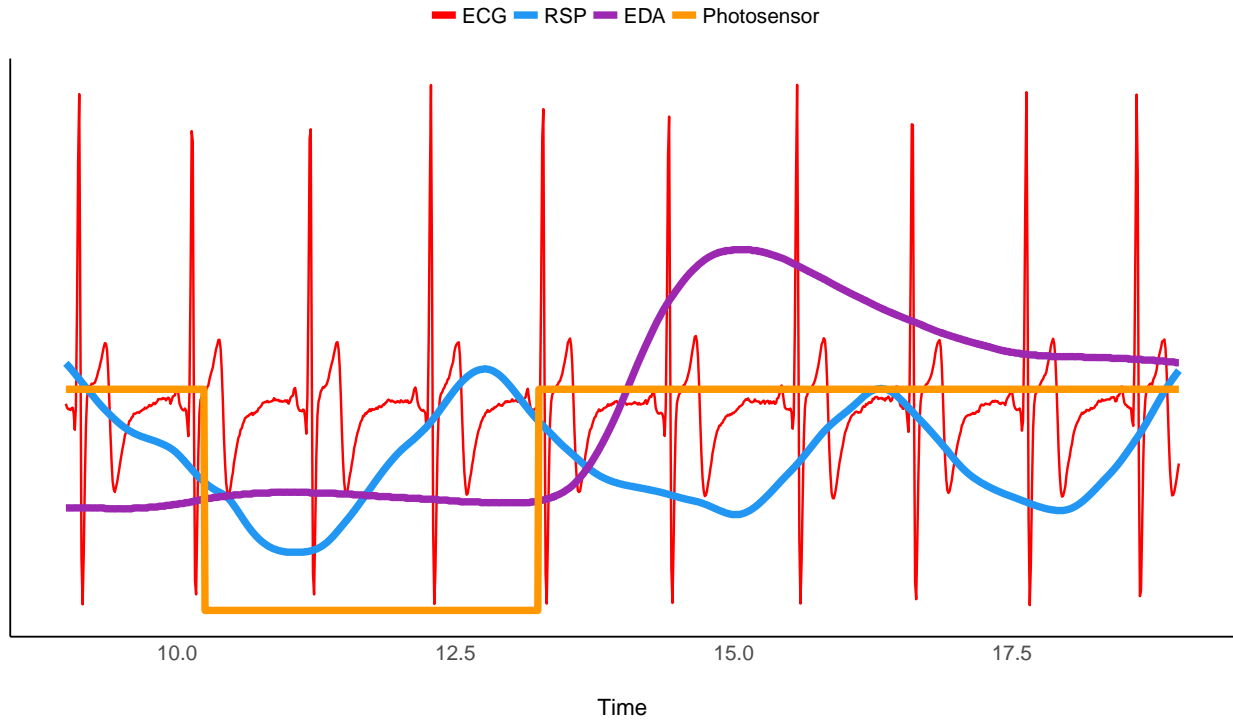
```
# Show subset of results

results[["Condition", "ECG_Rate_Mean", "RSP_Rate_Mean", "EDA_Peak_Amplitude"]]
```


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

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

NEUROKIT2   10

In this example, after loading the package and the example dataset, each physiological signal is processed using `bio_process()`. Stimulus onsets in the photosensor are detected separately with `events_find()`. Once we have the 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()`. Finally, relevant features are computed for each epoch (i.e., each stimulus) by passing them to `bio_analyze()`.

The features include for example 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).

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 participants. 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 descriptive interpretation is given solely for illustrative purposes).

Resting-state Features

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

172 three channels (ECG, PPG and RSP) sampled at a frequency of 100Hz.

```python
# 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

173 In this example, the steps of the analysis are identical to the previous example, including
174 loading the package, the dataset and processing the data. The difference is that there is
175 no epoching, as we want to compute features related to the whole dataset. Thus, we can

176 directly pass the dataframe to `bio_analyze()`, which will detect that these are not epochs,

177 and compute the appropriate features accordingly. These include for instance the average

178 heart and breathing rate, as well as indices of heart rate variability (HRV) and respiratory

179 sinus arrhythmia (RSA).

180 This example illustrates a second type of physiological analysis, that we refer to as interval-

181 related (as opposed to event-related). Interval-related analyses compute features of signal

182 variability and activation patterns over a longer-term period of time (typically minutes).

183 *NeuroKit2* allows for the fast creation of a standardized and reproducible pipeline to describe

184 this kind of physiological activity, which can be beneficial for a wide variety of applications.

Discussion

186 *NeuroKit2* is a neurophysiological signal processing software accessible to people with all

187 levels of programming experience and background. Its development is focused on creating an

188 intuitive user-experience, as well as building a collaborative community. It is also a pragmatic

189 answer to the broader need for transparent and reproducible methods in neurophysiology.

190 Its modular structure and organization not only facilitates the use of existing and validated

191 processing pipelines, but also creates a fertile ground for experimentation and innovation.

192 We expect the package's future evolution to be driven by the communities' needs and the

193 advances in related fields. Possible directions include extending the support for other types

194 of bodily signals (e.g., electrogastrography - EGG, electrooculography - EOG) and achieving

195 performance gains for large datasets by using efficient algorithms. Further validation of the

196 available processing pipelines could be made through the (re)analysis of public databases.

197 In line with this objective, the support of standardized data structure formats (e.g. WFDB,

198 BIDS, …) could be extended.

199 In conclusion, we believe that *NeuroKit2* provides useful tools for anyone who is interested

200 in analyzing physiological data from research-grade hardware as well as wearable "smart

₂₀₁ health devices". By increasing the autonomy of researchers and practitioners, and by short-

₂₀₂ ening the delay between data collection and results acquisition, *NeuroKit2* could be useful

₂₀₃ beyond academic research in neuroscience and psychology, including applications such as

₂₀₄ biofeedback, personal physiological monitoring and exercise science. Finally, we hope that

₂₀₅ *NeuroKit2* encourages users to become part of a supportive open-science community with

₂₀₆ diverse areas of expertise rather than relying on closed-source and proprietary software, thus

₂₀₇ shaping the future of neurophysiology and its related fields.

Conflict of Interest

₂₀₉ The authors declare that the research was conducted in the absence of commercial or financial

₂₁₀ relationships that could constitute a conflict of interest.

Acknowledgements

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

239   workflows. In *ELPUB* (pp. 87–90).

240 Legrand, N., & Allen, M. (2020). Systole: A python toolbox for preprocessing, analyz-

241   ing, and synchronizing cardiac data. Retrieved from https://github.com/embodied-

242   computation-group/systole

243 Marchewka, A., Żurawski, Ł., Jednoróg, K., & Grabowska, A. (2014). The nencki affec-

244   tive picture system (naps): Introduction to a novel, standardized, wide-range, high-

245   quality, realistic picture database. *Behavior Research Methods*, *46*(2), 596–610.

246 Quintana, D., Alvares, G. A., & Heathers, J. (2016). Guidelines for reporting articles on

247   psychiatry and heart rate variability (graph): Recommendations to advance research

248   communication. *Translational Psychiatry*, *6*(5), e803–e803.

249 Roy, Y., Banville, H., Albuquerque, I., Gramfort, A., Falk, T. H., & Faubert, J. (2019).

250   Deep learning-based electroencephalography analysis: A systematic review. *Journal*

251   *of Neural Engineering*, *16*(5), 051001.

252 Schölzel, C. (2020). NOnLinear measures for dynamical systems (nolds). Retrieved from

253   https://github.com/CSchoel/nolds

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

255   CreateSpace.

256 Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D.,

257   … Contributors, S. 1. 0. (2020). SciPy 1.0: Fundamental Algorithms for Scientific

258   Computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/https://doi.

259   org/10.1038/s41592-019-0686-2

260 Yuehong, Y., Zeng, Y., Chen, X., & Fan, Y. (2016). The internet of things in healthcare:

261   An overview. *Journal of Industrial Information Integration*, *1*, 3–13.