Journal of
open research software

A Python Package for Heart Rate Variability Analysis and Signal Preprocessing

SOFTWARE METAPAPER

ROBIN CHAMPSEIX
LAURENT RIBIERE
CLÉMENT LE COUEDIC

*Author affiliations can be found in the back matter of this article

ubiquity press

ABSTRACT

This paper presents *hrv-analysis*, a Python package for Heart Rate Variability (HRV) analysis. *hrv-analysis* is an open-source package for the Python statistical computing environment, which supports a wide variety of time, frequency and non-linear HRV analysis methods. It also includes several functions for signal preprocessing like outliers and ectopic beat removal.

This package is suitable for researchers, professionals working in healthcare as well as for developers and more widely for anyone willing to perform detailed analysis on heart rate variability.

hrv-analysis was developed as part of the Aura project. Aura is a non-profit project aiming to develop an open-source forecasting system to help people suffering from epilepsy regain autonomy in their everyday lives.

This package was developed using Python 3.

CORRESPONDING AUTHOR:
Robin Champseix
Lead author, Data scientist, OCTO Technology, Paris, France
robin.champseix@gmail.com

KEYWORDS:
Python; Heart-rate Variability; RR-Interval

TO CITE THIS ARTICLE:
Champseix R, Ribiere L, Le Couedic C 2021 A Python Package for Heart Rate Variability Analysis and Signal Preprocessing. *Journal of Open Research Software*, 9: 28. DOI: *https://doi.org/10.5334/jors.305*

(1) OVERVIEW

INTRODUCTION

Heart rate variability (HRV) describes the change in the beat-to-beat heart rate calculated from the electrocardiogram (ECG) signal. It is considered a key indicator of an individual's cardiovascular condition.

Since establishing that a significant relationship exists between the autonomic nervous system and cardiovascular mortality, researchers have focused on finding useful features derived from raw RR-interval signal to characterize autonomic nervous system state.

Softwares providing HRV analytic capabilities already exist but are under proprietary licenses and do not provide access to how results are calculated. In order to make HRV analytics freely available and allow for some transparency, we decided to open-source our package. *hrv-analysis* is an open-source package for the Python environment that provides a complete set of tools for performing HRV analysis.

In this paper, we shall only present the core aspects of the package. Further information can be found in the documentation [1]. The two main parts are the filtering function and feature extraction functions.

COMPARISON TO EXISTING SOFTWARE

To ensure the accuracy of all implemented methods, a careful comparison has been made with an existing standard software: Kubios HRV (v3.1) [2]. We compared all the features that are provided by both our package and Kubios software. All the methods detailed by domain are:

- for time-domain features: Mean RR, SDNN, Mean HR, RMSSD, NN50, pNN50, HRV triangular index
- for frequency domain features: VLF, LF, HF, LF/HR ratio, LFnu, HFnu
- for non-linear domain features: SD1, SD2, SD2/SD1, SampEn

We won't present each feature in this paper. All relevant information is available in the documentation [1].

To collect some test data, we relied on the Polar H10 chest belt [3]. We chose this sensor because it is reliable and provides very accurate RR-intervals [4, 5]. In addition, an API is available to allow the extraction of the successive RR-interval values for custom analysis.

One candidate was equipped with the device for several days and we selected segments of data with no obvious artifact. The segments' time range has been set to 5 minutes in line with the standard [6], to the best of our knowledge, for proper hrv analysis. No preprocessing has been applied to have an unbiased comparison.

In order to compare the calculation of the features in different situations, we focused on 5 periods with a distinct pattern of cardiac rhythm: one with an acceleration/deceleration, one with a high heart rate (physical activity situation, running) and several with normal heart rate activity.

For the sake of readability, we will only show 2 of those comparisons but the results are similar for all the segments. The results of the comparison for the time-domain and non-linear domain features are shown in the table below (***Figure 1***).

Activity pattern	**Moderate physical activity**			**Intense physical activity**		
	Kubios	**hrv-analysis**	**Relative error (%)**	**Kubios**	**hrv-analysis**	**Relative error (%)**
Time domain						
Mean RR (ms)	710.32	710.32	0	524,72	524,718	2.8x10e-4
SDNN (ms)	64.108	64.1082	3.8x10e-4	65.118	65.118	0
Triangular index	11.1	11.41	2.76	12.711	13.119	3.2
RMSSD (ms)	32.143	32.142	1.4x10e-3	15.305	15.305	0
NNxx	44	44	0	9	9	0
pNNxx	10.45	10.42	2.3x10e-1	1.576	1.573	1.7x10e-1
Non-linear domain						
SD1	22.755	22.755	0	10.832	10.832	4.7x10e-5
SD2	87.868	87.76	1.2x10e-1	91.418	91.452	3.7x10e-2
SD2/SD1	3.8614	3.8567	1.2x10e-1	8.4395	8.4427	3.8x10e-2
SampEn	1.2157	1.2101	4.6x10e-1	0.3523	0.3545	6.2x10e-1

Figure 1 Comparison of Time-domain and non- linear domain features between hrv-analysis and Kubios.

All the calculated features are quite close. Some small differences observed can be explained by mathematical approximations.

Regarding the frequency domain, we made the comparison using Welch's Fast Fourier Transform to estimate the power spectral density (PSD). Welch's periodogram is a premium functionality in Kubios so we were not able to compare the results of this method. We made sure to have the same settings and parameters as Kubios. For instance, we set the sampling frequency to 4 Hz (it is the default parameter in Kubios) and the segment size was fixed to 300s with 50% overlap.

Some differences are shown in the table below (***Figure 2***).

These differences might be considered significant, but it only highlights the importance of some other parameters like the window function, the method to detrend the segments or how integrals are calculated for which Kubios software provides no information on. It might also be due to the library implementation of the Scipy package [7] that we use in order to calculate the PSD. Another explanation could be that Kubios is using a different precision for computed variables (e.g float64 bits versus float32 bit) or relying on different FFT's implementations which might lead to minor variations in the computations.

However, because the order of magnitude of both features is very close, we think those measures are relevant medical indicators.

We notice that the over and underestimation of the LF and HF features vanished when we look at the normalized value or at the ratio, which tends to confirm the accuracy of our functions.

IMPLEMENTATION AND ARCHITECTURE

Language

hrv-analysis was written in pure Python. Thanks to the rise of analytics and Machine Learning projects, Python is becoming more and more popular among the field of researchers. It is an open-source language, development is simple, code is concise and there are lots of efficient scientific libraries available. For these reasons, we hope that it will help the package's adoption among the research community.

API organisation

The heart rate variability analysis rely on 2 sequential steps: the first being the preprocessing, the second being the feature calculations.

The full Python API reference is accessible on the Aura association github (*https://aura-healthcare.github.io/hrv-analysis/hrvanalysis.html*).

- ***hrvanalysis.preprocessing***: contains all methods to pre-process the RR-intervals
- ***hrvanalysis.extract_features***: contains all methods to compute features based on RR-intervals
- ***hrvanalysis.plot***: contain all methods to plot graph and features for RR-intervals (***Figure 3***)

It is possible, of course, to add new features, preprocessing methods or plot functions in each block to improve the library.

Step 1 – preprocessing

Outliers and ectopic beats are known to have a huge impact [8] on calculation of heart rate variability features. In a clinical setting, the best practice is to manually check for artifacts in ECG recordings and to include only artifact-free sections in the analysis [1]. This process is extremely time-consuming and not feasible for long term recordings. Alternatively, it is a common practice to automatically adjust RR-intervals and remove aberrant beats [9], but there is no consensus, to our knowledge, on the best

Activity pattern	Normal			Intense		
	Kubios	hrv-analysis	Relative error (%)	Kubios	hrv-analysis	Relative error (%)
Frequency domain						
LF	3214.3	3032.17	5.66	1515.9	1593.04	5.1
HF	198.99	209.43	5.24	147.38	123.45	16.2
LF / HF ratio	16.153	14.47	10.37	10.285	12.904	25.5
Lfnu	94.165	93.539	6.7x10e-1	91.123	92.808	1.85
Hfnu	5.8296	6.46	10.82	8.859	7.192	18.8

Figure 2 Comparison of frequency domain features between hrv-analysis and Kubios.

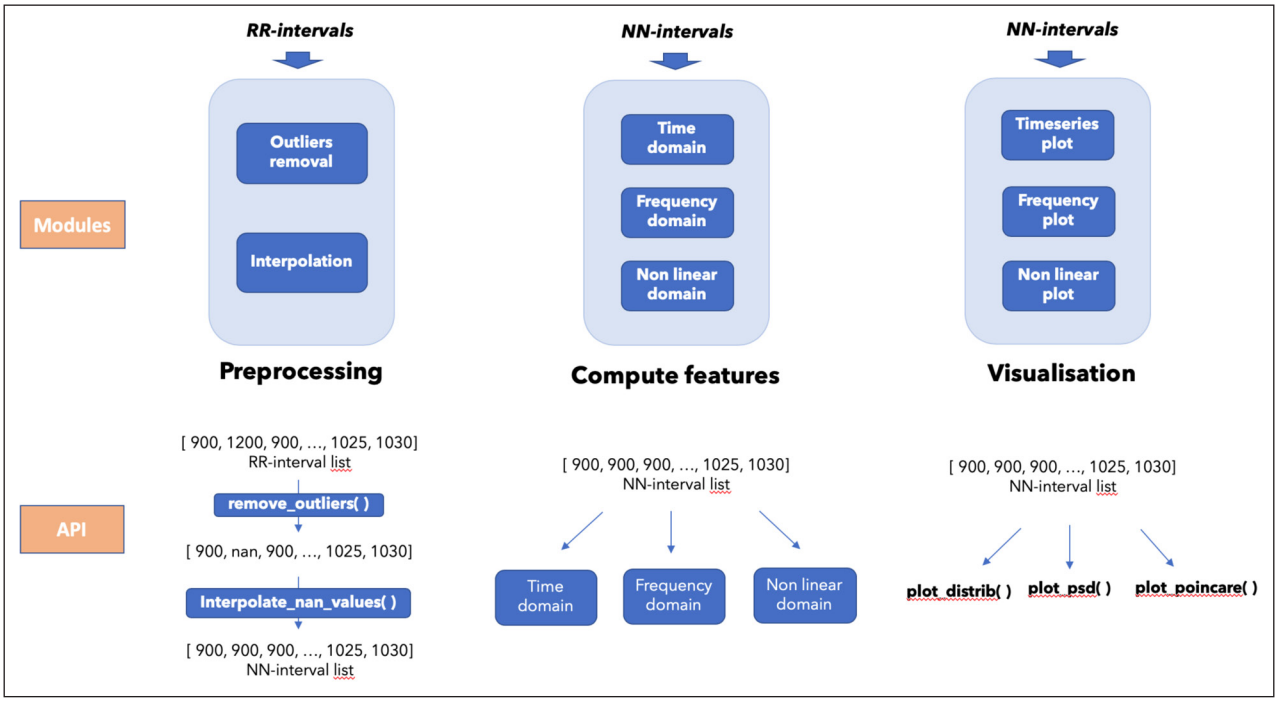

Figure 3 Building blocks of the package architecture.

method to use. The *hrv-analysis* package provides a two-step preprocessing system. First, two methods provide both outlier and ectopic beat detection and replacement with *NaN* values (Not a Number in Python). The second method interpolates the previously deleted values.

The Python API is used as shown in the example below:

```
from hrvanalysis.preprocessing import (remove_outliers, interpolate_nan_values)

rr_intervals_list = [1000, 150, 1050, 2600, 1000]
rr_intervals_list = remove_outliers(rr_intervals_list, low_rri=300, high_rri=2000)
rr_intervals_list
>>> [1000, nan, 1050, nan, 1000]

interpolate_nan_values(rr_intervals_list, interpolation_method=« linear »)
>>> [1000, 1025, 1050, 1000, 900]
```

Step 2 – Feature calculations

As previously stated, researchers and physiologists have been tracking and utilizing HRV for decades because it is considered a useful indicator for several health-related issues. There are several ways to evaluate variations in Heart rate: Time domain, Frequency domain and Non-linear domain analysis. They all have been implemented in the package and are very easy to use as they all take the same input RR-interval data.

The Python API can be used as follows:

```
from hrvanalysis import (get_time_domain_features,
                         get_frequency_domain_features)

rr_intervals_list = [1010, 987, …, 950, 929]

get_time_domain_features(rr_intervals_list)
>>> {'mean_nni': 718.248, 'sdnn': 43.11, 'sdsd': 19.51,'nni_50': 24, 'pnni_50': 2.4,
    'nni_20': 225, 'pnni_20': 22.5, 'rmssd': 19.52, 'median_nni': 722.5,
    'range_nni': 249, 'cvsd': 0.027, 'cvnni': 0.060, 'mean_hr': 83.85,
    'max_hr': 101.69, 'min_hr': 71.51, 'std_hr': 5.197}

get_frequency_domain_features(rr_intervals_list)
>>> {'lf': 299.72, 'hf': 436.15, 'lf_hr_ratio': 0.69, 'lfnu': 40.73,
'hfnu': 59.27, 'total_power': 833.46, 'vlf': 97.58}
```

Our implementation supports a large range of input parameters (sampling frequency, frequency bands, etc.) and Fourier decomposition methods that are commonly described in the literature. All information for that is available in the documentation of the package.

Plotting module

In parallel, there are some functions to plot standard graphics (with associated features on it) that are typically used by researchers for hrv analysis.

Name	Produced plot
plot_timeseries	Plot the RR-intervals time series.
plot_distrib	Plot histogram distribution of the NN Intervals.
plot_psd	Plot the power spectral density of the NN Intervals.
plot_poincare	Pointcaré / Lorentz Plot of the NN Intervals.

The following is an example of the *plot_psd()* method using two distinct plotting method (Welch vs Lomb) of the package (*Figure 4*).

Dependencies

The statistical analysis and preprocessing methods rely on Numpy [10] and Pandas [11] which provide optimized and efficient methods for arrays. Scipy [7] and Astropy [12] were used for frequency domain analysis, especially those involving Fourier transforms. The plot module depends entirely on Matplotlib library.

QUALITY CONTROL

Code quality has also been an important topic. Currently, 82% of the code is covered by tests using the Python unittest framework. We used codecov [13], a code coverage reporting tool to track this statistic. The tests are run through Travis CI, a continuous integration tool. All the latest dependencies are loaded into a Docker container thanks to github CI module. The package was developed in Python 3.5 but is available and can be used in all Python versions ranging from 3.5 to the latest (currently Python 3.9).

The package also follows PEP 484 by using "types hints" for all function parameters and outputs. We hope this will improve code readability and help detect bugs more easily.

There are two main tests files (all are available on github) corresponding to each high level functionality of the library:

- Preprocessing tests: checks that each preprocessing method returns the correct list of modified RR-intervals
- Feature calculations tests: checks that each feature calculation is similar to Kubios features.

All tests can also be run separately and instructions for running them locally are given in the provided documentation.

The library also follows PEP8 coding standards and a linter, pylint, has been used on all files to insure consistency across different modules.

We hope that it will ease maintenance and further improvements.

CONCLUSION

This first version of the library integrates what we think are the most important features for a robust and adequate basic HRV analysis. It doesn't depend on any additional software other than the Python dependencies listed. Its architecture is flexible and, we hope, easy to integrate in custom systems (like an API). Robustness has been assured by appropriate test coverage and comparison with existing state of the art tools used to compute HRV features.

The first usage of this library will be for epilepsy monitoring through AURA Association's project and a lot of future use cases might be considered. For instance, it could allow the follow up of various chronic cardiac diseases.

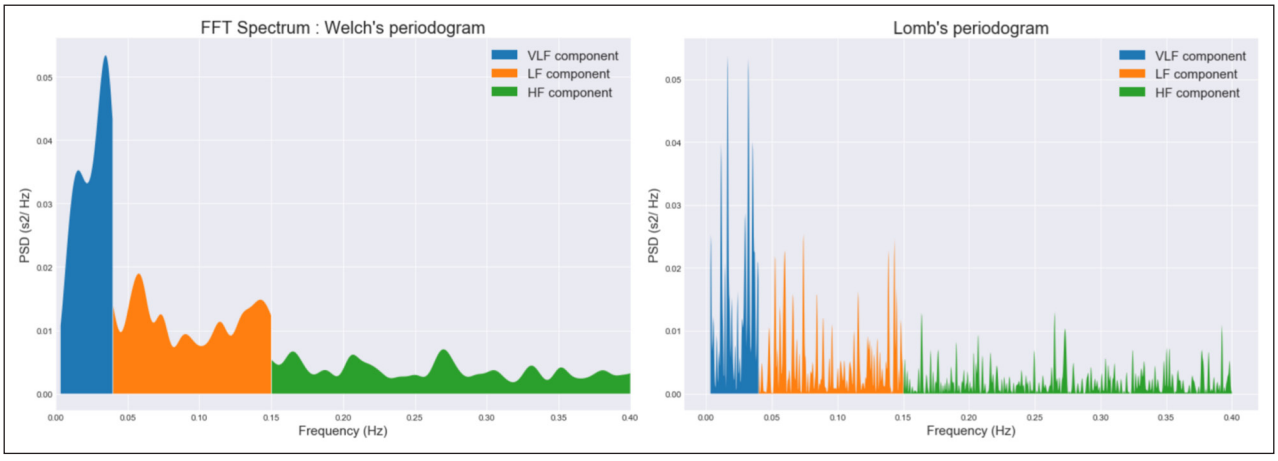

Figure 4 Example of plot with plot_psd() method.

The package does not pretend to be exhaustive though. Some next steps could be adding some support for raw ECG data (RR-interval extraction through QRS complex detection), or advanced time frequency analysis (multitaper filters) to name a few. By open-sourcing this package (available at *https://github.com/Aura-healthcare/hrv-analysis*), we hope other researchers or users will make good use of it and that the community will help us enrich this library.

(2) AVAILABILITY

OPERATING SYSTEM

GNU/Linux, Mac OS X, Windows and on any platform supported by Docker, such as Azure and AWS.

PROGRAMMING LANGUAGE

hrv-analysis has been written in Python 3.

DEPENDENCIES

E.g. libraries, frameworks, incl. minimum version compatibility.

Python version 3.5 minimum
Numpy>=1.15.1
SciPy ≥1.1.0
Pandas>=0.23.4
Matplotlib>=2.2.2
Astropy>=3.0.4
Nolds>=0.4.1

SOFTWARE LOCATION

Archive

Name: hrv-analysis
Persistent identifier: *https://pypi.org/project/hrv-analysis/#files*
Licence: GNU General Purpose License version 3.0
Publisher: Robin Champseix
Version published: v1.0.0
Date published: 11 October, 2018

Code repository

Name: GitHub
Identifier: *https://github.com/Aura-Healthcare/hrv-analysis*
Licence: GNU General Purpose License version 3.0
Date published: September, 2018

LANGUAGE

English

(3) REUSE POTENTIAL

The Python language and the dependent library packages used are all open-source. Hrv-analysis can be used by clinical and public health researchers, professionals working on human or animal well-being, or sports enthusiasts; for anybody who wants to perform detailed analysis on heart rate variability and to examine autonomic nervous system function.

FUNDING STATEMENT

This work was supported by OCTO Technology and Aura association.

COMPETING INTERESTS

The authors have no competing interests to declare.

AUTHOR AFFILIATIONS

Robin Champseix *orcid.org/0000-0002-9407-6780*
Data scientist, OCTO Technology, Paris, France
Laurent Ribiere *orcid.org/0000-0002-9427-4752*
Data Engineer, former OCTO Technology employee, Paris, France
Clément Le Couedic
Researcher, AURA association, Paris, France

REFERENCES

1. *https://aura-healthcare.github.io/hrv-analysis/*.
2. *https://www.kubios.com/*.
3. *https://www.polar.com/uk-en/products/accessories/polar_h10_heart_rate_sensor*.
4. **Gamelin FX, Berthoin S, Bosquet L.** Validity of the Polar S810 Heart Rate Monitor to Measure R–R Intervals at Rest. *Med Sci Sports Exerc*; 2006. DOI: *https://doi.org/10.1249/01.mss.0000218135.79476.9c*
5. **Vanderlei LCM, Silva RA, Pastre CM, Azevedo FM, Godoy MF.** Comparison of the Polar S810i monitor and the ECG for the analysis of heart rate variability in the time and frequency domains; 2008. DOI: *https://doi.org/10.1590/S0100-879X2008005000039*
6. **Task force of the European society of cardiology and the North American society of pacing and electrophysiology.** Heart rate variability – standards of measurement, physiological interpretation, and clinical use.
7. *https://docs.scipy.org/doc/scipy-1.2.1/reference/generated/scipy.signal.welch.html*.
8. **Choi A, Shin H.** Quantitative Analysis of the Effect of an Ectopic Beat on the Heart Rate Variability in the Resting Condition. DOI: *https://doi.org/10.3389/fphys.2018.00922*
9. **Kamath MV, Fallen EL.** Correction of the Heart Rate Variability Signal for Ectopics and Missing Beats.
10. *https://numpy.org/*.
11. *https://pandas.pydata.org*.
12. *https://www.astropy.org/*.
13. *https://docs.codecov.io/docs/python*.

TO CITE THIS ARTICLE:
Champseix R, Ribiere L, Le Couedic C 2021 A Python Package for Heart Rate Variability Analysis and Signal Preprocessing. *Journal of Open Research Software*, 9: 28. DOI: *https://doi.org/10.5334/jors.305*

Submitted: 15 October 2019 Accepted: 03 September 2021 Published: 06 October 2021

Journal of Open Research Software is a peer-reviewed open access journal published by Ubiquity Press.