The Journal of Open Source Software

HRV: a Pythonic package for Heart Rate Variability Analysis

Rhenan Bartels[1] **and Tiago Peçanha**[2]

1 Pulmonary Engineering Laboratory, Biomedical Engineering Program, Federal University of Rio de Janeiro, Rio de Janeiro, Brazil **2** Faculty of Medicine, University of São Paulo, Brazil

DOI: 10.21105/joss.01867

Software

- Review
- Repository
- Archive

Editor: Kevin M. Moerman

Reviewers:

- @paulvangentcom
- @Kevin-Mattheus-Moerman

Submitted: 29 October 2019
Published: 25 July 2020

Summary

Heart rate variability (HRV) is a non-invasive tool to assess the cardiac autonomic integrity and cardiovascular homeostasis (Electrophysiology, 1996). HRV quantifies the instantaneous variations in the RR intervals (RRi), which is produced by the balanced action of the parasympathetic and sympathetic branches of the autonomic nervous system (ANS) over the sinoatrial (SA) node, and modulated by different physiological inputs (e.g., respiration, blood pressure, temperature, emotions, etc) (Malik & Camm, 1990). Specifically, the parasympathetic activation produces fast and short-lasting bradycardia, which results in high-frequency oscillations (i.e., 0.15 - 0.40 Hz) in heart rate. On the other hand, sympathetic activation produces slower and longer-lasting variations in heart rate, which results in low-frequency (i.e., ~ 0.1 Hz) oscillations in heart rate.

Increased HRV indicates a predominance of the parasympathetic over the sympathetic activation in the SA node, and indicates enhanced cardiac autonomic flexibility and improved overall health (Malik & Camm, 1990). Conversely, a reduced HRV is usually accompanied by sympathetic dominance and suggests increased rigidity and loss of control of the ANS to the cardiovascular system, which is a sign of disease vulnerability (Malik & Camm, 1990). For instance, a reduced SDNN (i.e., standard deviation of RR intervals, a simple statistical-based time domain index of HRV) has been shown to overperform traditional cardiovascular risk parameters in predicting mortality in a cohort of heart failure patients (Nolan et al., 1998). A reduced HRV has also been linked with metabolic dysfunction (Weissman, Lowenstein, Peleg, Thaler, & Zimmer, 2006), increased inflammation (Sajadieh et al., 2004), depression (Sgoifo, Carnevali, Pico Alfonso, & Amore, 2015), psychiatric disorders (DeGiorgio et al., 2010), sleep disturbance (Burton, Rahman, Kadota, Lloyd, & Vollmer-Conna, 2010), among others. Based on this wide prognostic utility, the interest in approaches to evaluate HRV has shown an exponential growth in different medicine specialties and research fields in the recent years.

HRV is routinely assessed using linear methods, through the calculation of different indices either in time- or frequency-domain. Time-domain consists of a collection of statistical metrics, such as the average value of RRi (mRRi), the standard deviation of RRi (SDNN; the NN stands for natural or sinusal intervals), the standard deviation of the successive differences (SDSD), the number or percentage of RRi longer than 50ms (NN50 and pNN50) and the root mean squared of successive difference in adjacent RRi (RMSSD - equation 1) (Electrophysiology, 1996). Each of these indices quantifies different facets of the HRV, which are promoted by different autonomic sources. SDNN quantifies overall variability behind HRV, which is produced by both parasympathetic and sympathetic branches. NN50, pNN50, and RMSSD quantify beat-to-beat HRV, which is produced predominantly by the parasympathetic action in the heart.

$$RMSSD = \sqrt{\frac{1}{N-1}\sum_{j-1}^{N-1}(RRi_j - RRi_{j+1})^2}$$

- Equation 1

where N is the count of RRi values and RRi_j is the jth RRi value.

The frequency-domain analysis quantifies the extent of contribution of each frequency component to the overall heart rate fluctuation (Figure 2). The main frequency components are the VLF (i.e., very low frequency; $<$ 0.04 Hz); LF (low frequency; 0.04-0.15 Hz) and the HF (i.e., high frequency; 0.15-0.40 Hz). The HF component is coupled with the respiratory fluctuation (i.e., respiratory sinus arrhythmia) and is produced by the parasympathetic modulation on the heart. The LF is mainly coupled with variations in the blood pressure (i.e., Mayer waves), and is thought to represent the modulation of both parasympathetic and sympathetic branches on the heart. The VLF does not have a defined physiological source, but it may involve alterations in heart rate produced by hormones and body temperature (Electrophysiology, 1996).

Roughly, frequency domain analysis involves the calculation of the spectral energy content of each frequency component through a power spectral density (PSD) estimation. Several methods have been developed to perform the PSD estimation and they are generally divided into two categories that provide comparable results: non-parametric and parametric methods, each with respective pros and cons (Electrophysiology, 1996). The Welch periodogram (Welch, 1967) is a non-parametric approach based on the Fourier Transform and consists of the average of several PSD estimations on different segments of the same RRi series, which is an important approach to reduce the spectral estimation variability (Welch, 1967). On the other hand, the autoregressive technique is the most widely used parametric method to estimate the spectral components of the HRV signal (Berntson et al., 1997). The PSD estimation with the autoregressive method consists of a parametric representation of the RRi series and the frequency response of the estimated model. From the estimated PSD, generally, the following indices presented in Table 1 are calculated.

Variable	Units	Frequency Band
Total Power	ms2	0 - 0.4 Hz
VLF	ms2	$<$ 0.04 Hz
LF	ms2	0.04 - 0.15 Hz
HF	ms2	0.15 - 0.4 Hz
LF/HF		
LFn.u	normalized units $\frac{LF}{TotalPower-VLF}$	
HFn.u	normalized units $\frac{HF}{TotalPower-VLF}$	

Non-linear indices are also frequently used to extract information from the ANS based on the heart rate fluctuations patterns. The Poincaré ellipse plot belongs to the non-linear methods and consists of a diagram in which each RRi is plotted as a function of the previous RRi value (Berntson et al., 1997). In addition to the visual information about the RRi scatter given by the plot, two indices are extracted from this diagram: SD1 and SD2. The former reflects the short term fluctuations of the heart rate, and for this reason is highly correlated with the RMSSD, pNN50 and HF indices, while the latter reflects both short and long terms of the fluctuation of the heart rate and correlates with SDNN and LF indices. Additionaly, the SD1 index represents the standard deviation spread orthogonally to the identity line (y=x) and it is the ellipse width, whereas the SD2 index represents the standard deviation spread along the identity line and specifies the length of the ellipse. At the end of the following section, the Poincaré plot of a given RRi series is depicted using the module presented in the current article.

The calculation of the SD1 and SD2 can be derived from SDSD and SDNN values as shown by equations 2 and 3 below:

$$SD1 = \sqrt{2SDNN^2 - 2SD2^2}$$

- Equation 2

$$SD2 = \sqrt{2SDNN^2 - \frac{1}{2}SDSD^2}$$

- Equation 3

There are several software packages written in many different programming languages that offer functions to work with RRi signals. Some of them have a command-line interface (Rodríguez-Liñares, Vila, Mendez, Lado, & Olivieri, 2008) and others offer a user's interface to improve the interaction with the RRi series and the analyses (Tarvainen, Niskanen, Lipponen, Ranta-Aho, & Karjalainen, 2014, p. @bartels2017sinuscor). Specifically for Python, there is also open-source packages available and ready to work on HRV analisys, such as hrvanalysis, pyhrv (Gomes, Margaritoff, & Silva, 2019), and heartpy (Gent, Farah, Nes, & Arem, 2019). Although these modules do a great work offering many of the most widely used techniques to deal with tachograms and to extract relevant information from HRV signals, their functions interface (API) relies on RRi signals stored as Python iterable or numpy arrays and is based mostly on the procedural programming paradigm.

The `hrv` is a simple and open-source Python module that comes with the most common techniques for filtering, detrending and extracting information about the ANS from the RRi signals without losing the power and flexibility of a native Python object and a numpy arrays (Oliphant, 2006). It brings the necessary methods to work with a tachogram encapsulated in a Python class. In other words, once an RRi class is instantiated there are several methods available for visualization, descriptive statistics, slicing the signal in shorter segments, and displaying the metadata of the series.

With many software available to work with HRV analysis, the main reason why the `hrv` module is being developed is to improve and simplify the interaction with an RRi series with idiomatic Python code, closer to the native objects of this language. The object-oriented approach offered by the present module allows a strong relation between the RRi series and its methods, especially regarding time information. With a class representing and encapsulating the RRi object, each RRi value is bound to its respective time information, and therefore, after actions like slicing and filtering, the RRi series still keeps track of its information. Additionally, the instance's properties help to keep the state of the RRi series, informing, for instance, if it is already detrended and/or resampled.

The following sections present the basic workflow with an RRi series and gives a better overview of the functionalities available in the `hrv` module, starting with reading a file containing a tachogram, visualizing the given RRi series, dealing with noise filtering and detrending and, finally, calculating the time/frequency domain and non-linear HRV indices.

Basic Usage

This section presents a non-exhaustive walkthrough of the features offered by the `hrv` module. To have access to the source code and more usage examples, please refer to the software repository, the complete documentation or the notebooks with some use cases of the `hrv` module.

Once the RRi series is created in Python using the `hrv.io` submodule, which supports text, CSV and hrm (PolarTM) files, or from any Python iterable (i.e lists, tuples, etc), an RRi

instance with the necessary methods to implement the Python iterable pattern is created. With the RRi object it is possible to iterate (i.e `[r for r in rri_series]`), search for a value at a given index (i.e `rri_series[0]`), and slice the tachogram (i.e `rri_series[5:10]`). As the RRi class also implements some of the behaviors of the numpy array (Oliphant, 2006), it is possible to perform math operations with the tachogram, i.e: `rri_series / 1000`.

The RRi class also has methods for basic statistical metrics calculation, such as average, standard deviation, min and max, and others. In order to access a complete Python dictionary containing all available statistical metrics of an RRi instance, it is possible to call the `describe()` method. Features for visualization are also present in the RRi class. In order to visualize the time series represented by the RRi series, the `plot()` method can be called. The visualization of the histograms showing the distributions of RRi or heart rate time series is also possible with the method `hist()`.

Read a file containing RRi values and visualising it

The following code snippet shows how to read a RRi series from a single column CSV file and plot the respective series with black lines.

```
from hrv.io import read_from_csv

rri = read_from_csv('path/to/file.csv')

fig, ax = rri.plot(color='k')
```

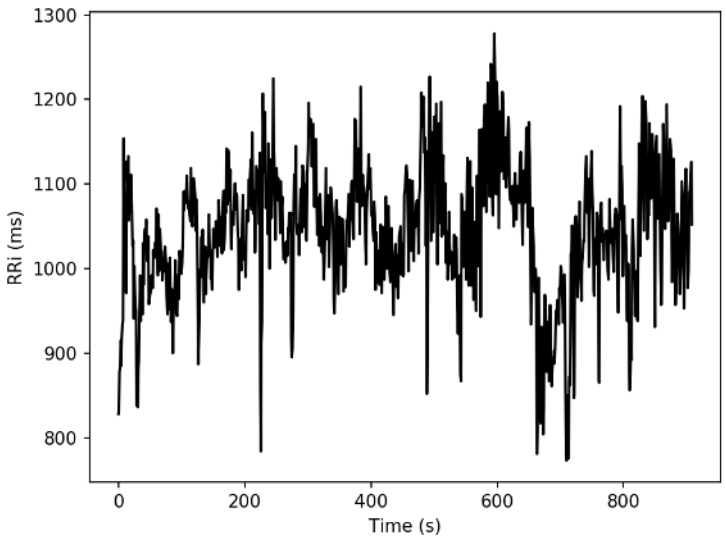

Figure 1: RR intervals of a young subject at rest condition produced with the `plot()` method from the RRi class.

To retrieve statistical properties of a RRi series the method `describe()` can be invoked:

```
desc = rri.describe()

desc
```

```
--------------------------------------
                   rri          hr
--------------------------------------
min             750.00       66.30
max             905.00       80.00
mean            805.50       74.78
var            2646.25       20.85
std              51.44        4.57
median          805.00       74.54
amplitude       155.00       13.70
```

```
print(desc['std'])
{'rri': 51.44171459039833, 'hr': 4.5662272355549725}
```

Pre-processing

Filtering the RRi series

In some cases and for many different reasons, the tachogram may present with movement artifacts or undesired RRi values, which may jeopardize the analysis results. One way to deal with this scenario is to apply filters to the RRi series. For this reason, the `hrv` package offers four lowpass filters for noise removal: moving average, which given an order value N, replaces every RRi value by the average of its N neighbors values; the moving median, which works similarly to the moving average filter, but apply the median function; the quotient filter (Piskorski & Guzik, 2005), that removes the RRi values which the ratio with its adjacent RRi is greater than 1.2 or smaller than 0.8; and finally, the threshold filter, which is inspired in Kubios (Tarvainen et al., 2014) threshold-based artifact correction algorithm: each RRi is compared to a local value consisting of the median of adjacent RRi. If the difference between a given RRi and the local median is greater than the threshold in milliseconds this RRi is considered an ectopic beat. Ectopic RRi values are replaced with cubic spline interpolation of the entire tachogram.

```
from hrv.filters import moving_median, quotient

filt_rri_median = moving_median(rri, order=3)
filt_rri_quotient = quotient(rri)

filt_rri_median.plot(ax=ax)
filt_rri_quotient.plot(ax=ax)
```

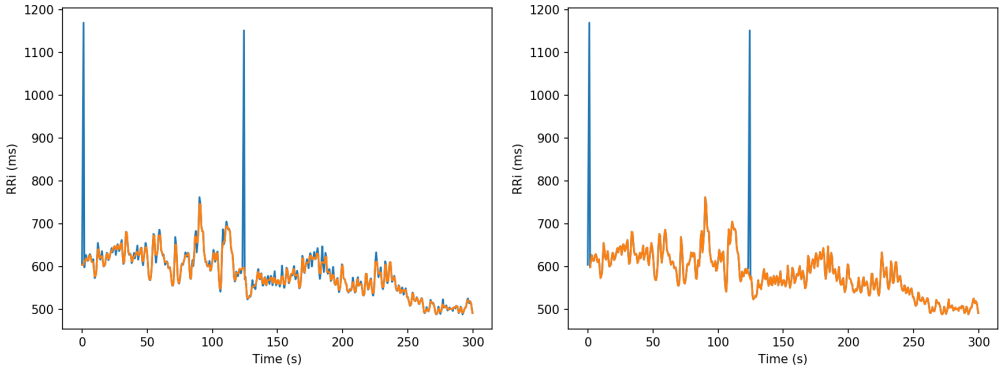

Figure 2: The left panel shows the original RRi (blue line) and after filtering with a moving median filter with order equal to 3 (orange line). The left panel depicts the original RRi (blue line) and after filtering with a quotient filter (orange line). This picture was created using the `plot()` method from the `RRi` instance.

Detrending the RRi series

Although the very-low-frequency components of the PSD function might have useful information, they are generally removed from the RRi signals before the frequency-domain analysis is performed. This pre-processing step before the frequency-domain analysis is important to remove intrinsic slow trends that are present in the HR fluctuation. This non-stationary behavior may contaminate the overall dynamic of the RRi series and influence the results, especially the VLF and LF measures (Tarvainen, Ranta-Aho, & Karjalainen, 2002). For this reason, several methods have been developed to extract the frequency components responsible for the non-stationary behavior of the RRi series.

Among the methods available in the literature for detrending the RRi series, the `hrv` module offers the polynomial detrend, which consists of the subtraction of a Nth degree polynomial from the RRi signal, where N is smaller than the length of the tachogram. It also offers the Smoothness Priors method (Tarvainen et al., 2002), which is widely used in HRV analyses and acts as a lowpass filter to remove complex trends from the RRi series. Finally, the `hrv` module also offers a detrending method that uses the Savitsky-Golay lowpass filter to remove low-frequency trends from the RRi series. The following code fragment applies the polynomial detrend with a degree equal to 1 and the Savitsky-Golay filter to remove the slow frequency components from an RRi recorded during rest.

```
from hrv.detrend import polynomial_detrend, sg_detrend
from hrv.sampledata import load_rest_rri

rri = load_rest_rri()

detrended_rri_poly = polynomial_detrend(rri, degree=1)
detrended_rri_sg = sg_detrend(rri, window_length=51, polyorder=3)

detrended_rri_poly.plot()
detrended_rri_sg.plot()
```

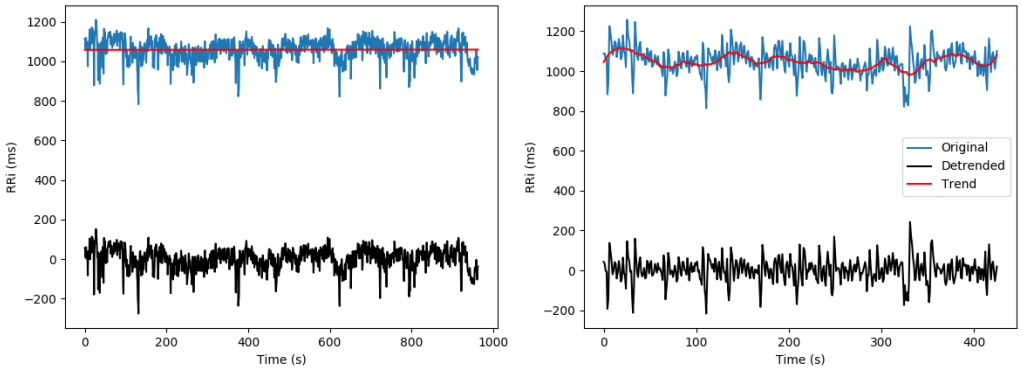

Figure 3: The left panel shows the original RRi (blue line) and after detrending with polynomial function with degree equal to 1 (black line). The left panel depicts the original RRi (blue line) and after detreding with a Savitsky-Golay lowpass filter (black line).

Analyses

Time Domain

In order to calculate the time-domain indices, the function `time_domain` can be imported from the submodule `hrv.classical` and applied to any Python iterable containing the RRi series including the RRi instance from the module presented in this article.

```
from hrv.classical import time_domain

results = time_domain(rri)
print(results)
```

```
{'mhr': 66.528130159638053,
 'mrri': 912.50302419354841,
 'nn50': 337,
 'pnn50': 33.971774193548384,
 'rmssd': 72.849900286450023,
 'sdnn': 96.990569261440797
 'sdsd': 46.233829821038042}
```

Frequency Domain

Similarly to the `time_domain` function, to calculate the frequency-domain indices, the `frequency_domain`, which is also placed in the `hrv.classical` submodule, can be used. The `frequency_domain` function present in the `hrv` module takes care of the pre-processing steps: the detrending of the RRi series (which the default is a linear function, but can be any custom Python function), interpolation using cubic splines (also accepts linear interpolation) and resampling at a given frequency, the default is 4Hz.

When Welch's method is selected, a window function (default: hanning), the number of RRi values per segment and the length of superposition between adjacent segments can be chosen. When the AR method is selected, the order of the model (default 16) can be set.

The area under the curve of each frequency range in the estimated PSD is calculated using the trapezoidal method. As a default, the hrv module uses the frequencies cutoffs shown in Table 1 to limit the integration range of each frequency domain indices, however, it is possible to set the frequency range of VLF, LF, and HF in the `frequency_domain` function call.

```
from hrv.classical import frequency_domain

results = frequency_domain(
    rri=rri,
    fs=4.0,
    method='welch',
    interp_method='cubic',
    detrend='linear'
)
print(results)
```

```
{'hf': 1874.6342520920668,
 'hfnu': 27.692517001462079,
 'lf': 4894.8271587038234,
 'lf_hf': 2.6110838171452708,
 'lfnu': 72.307482998537921,
 'total_power': 7396.0879278950533,
 'vlf': 626.62651709916258}
```

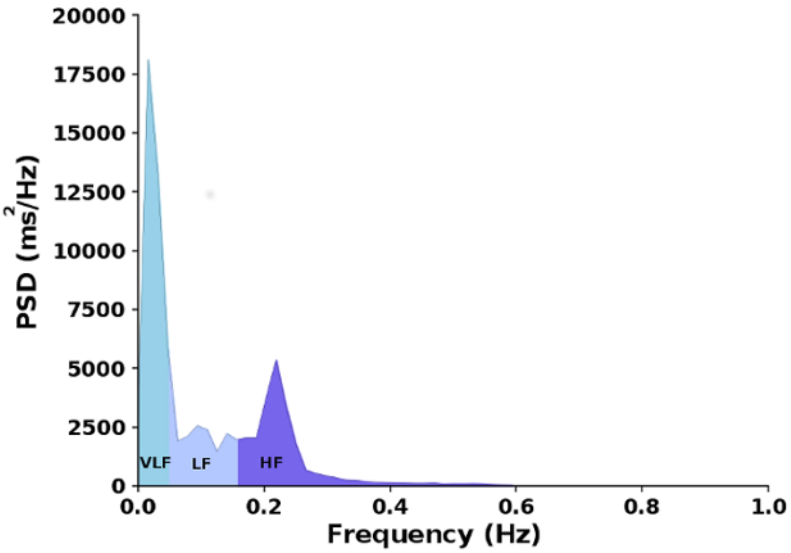

Figure 4: Power Spectral Density of a RRi series estimated with the Welch's method.

Non-linear

Finally, among the non linear metrics, `hrv` module offers SD1 and SD2, which can be calculated with the `non_linear` function from the `hrv.classical` submodule.

```
from hrv.classical import non_linear

results = non_linear(rri)
print(results)
```

```
{'sd1': 51.538501037146382,
 'sd2': 127.11460955437322}
```

The respective Poincaré plot of a given RRi series can be depicted with the `poincare_plot()` method, as follows:

```
rri.poincare_plot()
```

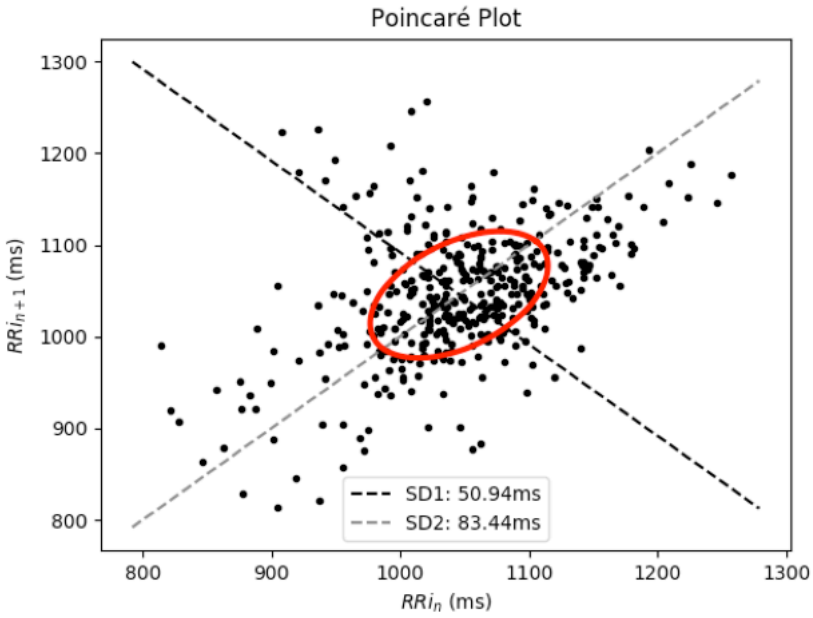

Figure 5: Poincaré plot of a given RRi series.

Dependencies

The `hrv` package depends on the following modules: numpy (Oliphant, 2006), matplotlib (Hunter, 2007), scipy (Jones, Oliphant, Peterson, & others, 2001) and spectrum (Cokelaer & Hasch, 2017).

Acknowledgements

The authors are grateful to CAPES (Coordenadoria de Aperfeiçoamento de Pessoal de Nível Superior), FAPERJ (Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro), FAPESP (2016/23319-0 - Fundação de Amparo à Pesquisa do Estado do São Paulo) and CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) for financial support. We are also grateful to Wilson Mello a.k.a Bakudas for creating the nice logo of the `hrv` module.

References

Bartels, R., Neumamm, L., Peçanha, T., & Carvalho, A. R. S. (2017). SinusCor: An advanced tool for heart rate variability analysis. *Biomedical engineering online*, *16*(1), 110. doi:10.1186/s12938-017-0401-4

Berntson, G. G., Thomas Bigger Jr, J., Eckberg, D. L., Grossman, P., Kaufmann, P. G., Malik, M., Nagaraja, H. N., et al. (1997). Heart rate variability: Origins, methods, and

interpretive caveats. *Psychophysiology*, *34*(6), 623–648. doi:10.1111/j.1469-8986.1997.tb02140.x

Burton, A., Rahman, K., Kadota, Y., Lloyd, A., & Vollmer-Conna, U. (2010). Reduced heart rate variability predicts poor sleep quality in a case–control study of chronic fatigue syndrome. *Experimental brain research*, *204*(1), 71–78. doi:10.1007/s00221-010-2296-1

Cokelaer, T., & Hasch, J. (2017). 'Spectrum': Spectral analysis in python. *J. Open Source Software*, *2*(18), 348. doi:10.21105/joss.00348

DeGiorgio, C. M., Miller, P., Meymandi, S., Chin, A., Epps, J., Gordon, S., Gornbein, J., et al. (2010). RMSSD, a measure of vagus-mediated heart rate variability, is associated with risk factors for sudep: The sudep-7 inventory. *Epilepsy & Behavior*, *19*(1), 78–81. doi:10.1016/j.yebeh.2010.06.011

Electrophysiology, T. F. of the E. S. of C. the N. A. S. of P. (1996). Heart rate variability: Standards of measurement, physiological interpretation, and clinical use. *Circulation*, *93*(5), 1043–1065. doi:10.1161/01.CIR.93.5.1043

Gent, P. van, Farah, H., Nes, N. van, & Arem, B. van. (2019). Analysing noisy driver physiology real-time using off-the-shelf sensors: Heart rate analysis software from the taking the fast lane project. *Journal of Open Research Software*, *7*(1). doi:10.5334/jors.241

Gomes, P., Margaritoff, P., & Silva, H. (2019). pyHRV: Development and evaluation of an open-source python toolbox for heart rate variability (HRV). In *Proc. Int'l conf. On electrical, electronic and computing engineering (icetran)* (pp. 822–828).

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. doi:10.1109/MCSE.2007.55

Jones, E., Oliphant, T., Peterson, P., & others. (2001). SciPy: Open source scientific tools for Python. Retrieved from http://www.scipy.org/

Malik, M., & Camm, A. J. (1990). Heart rate variability. *Clinical cardiology*, *13*(8), 570–576. doi:10.1002/clc.4960130811

Nolan, J., Batin, P. D., Andrews, R., Lindsay, S. J., Brooksby, P., Mullen, M., Baig, W., et al. (1998). Prospective study of heart rate variability and mortality in chronic heart failure: Results of the united kingdom heart failure evaluation and assessment of risk trial (uk-heart). *Circulation*, *98*(15), 1510–1516. doi:10.1161/01.cir.98.15.1510

Oliphant, T. (2006). NumPy: A guide to NumPy. USA: Trelgol Publishing. Retrieved from http://www.numpy.org/

Piskorski, J., & Guzik, P. (2005). Filtering poincare plots. *Computational methods in science and technology*, *11*(1), 39–48. doi:10.12921/cmst.2005.11.01.39-48

Rodŕiguez-Liñares, L., Vila, X., Mendez, A., Lado, M., & Olivieri, D. (2008). RHRV: An r-based software package for heart rate variability analysis of ecg recordings. In *3rd iberian conference in systems and information technologies (cisti 2008)* (pp. 565–574).

Sajadieh, A., Nielsen, O. W., Rasmussen, V., Hein, H. O., Abedini, S., & Hansen, J. F. (2004). Increased heart rate and reduced heart-rate variability are associated with subclinical inflammation in middle-aged and elderly subjects with no apparent heart disease. *European heart journal*, *25*(5), 363–370. doi:10.1016/j.ehj.2003.12.003

Sgoifo, A., Carnevali, L., Pico Alfonso, M. de los A., & Amore, M. (2015). Autonomic dysfunction and heart rate variability in depression. *Stress*, *18*(3), 343–352. doi:10.3109/10253890.2015.1045868

Tarvainen, M. P., Niskanen, J.-P., Lipponen, J. A., Ranta-Aho, P. O., & Karjalainen, P. A. (2014). Kubios hrv–heart rate variability analysis software. *Computer methods and programs in biomedicine*, *113*(1), 210–220. doi:10.1016/j.cmpb.2013.07.024

Tarvainen, M. P., Ranta-Aho, P. O., & Karjalainen, P. A. (2002). An advanced detrending method with application to hrv analysis. *IEEE Transactions on Biomedical Engineering*, *49*(2), 172–175. doi:10.1109/10.979357

Weissman, A., Lowenstein, L., Peleg, A., Thaler, I., & Zimmer, E. Z. (2006). Power spectral analysis of heart rate variability during the 100-g oral glucose tolerance test in pregnant women. *Diabetes care*, *29*(3), 571–574. doi:10.2337/diacare.29.03.06.dc05-2009

Welch, P. (1967). The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on audio and electroacoustics*, *15*(2), 70–73. doi:10.1109/TAU.1967.1161901