PhysioZoo Documentation

PhysioZoo Team

Feb 14, 2021

TUTORIALS

INTRODUCTION

PhysioZoo is a collaborative platform dedicated to the study of the heart rate variability (HRV) from Humans and other mammals' electrophysiological recordings. The main components of the platform are:

- *Software*

  - An open-source algorithmic toolbox for matlab (`mhrv`), which implements all standard HRV analysis algorithms, a selection of peak detection algorithms and prefiltering routines. This can be used within your own data analysis code using the `mhrv` API.

  - An open-source graphical user interface (`PZ-UI`) that provides a user friendly interface for advanced HRV analysis of RR-intervals time series and data visualization tools. This enables easy access to HRV analysis without writing any code.

- *Databases*

  - A set of annotated databases (`PZ-DB`) of electrophysiological signals from different mammals (dog, rabbit and mouse). Available here.

  - Manually audited peak locations and signal quality annotations for each of the recordings.

- *Configuration*

  - A set of configuration files that adapt the HRV measures and `mhrv` algorithms to work with data from different mammals.

  - All HRV measures can be further adapted for the analysis of other mammals by creating simple human-readable mammal-specific configuration files.

Note: The `PZ-UI` user interface has two modules: a `Peak detection` (used to process electrophysiological signals and obtain the RR time series) module and a `HRV analysis` module (used to process the RR time series and compute HRV measures).

The **PhysioZoo** mission is to standardize and enable the reproducibility of HRV analysis in mammals' electrophysiological data. This is achieved through its open source code, freely available user interface and open access databases. It also aims to encourage the scientific community to contribute their electrophysiological databases and novel HRV algorithms/analysis tools for advancing the research in the field.

Feedback on how to improve the **PhysioZoo** platform is welcomed. Do not hesitate to drop us an email at:

physiozoolab@gmail.com

Source code, data or interface enhancement contributions are welcome. Look here on how to contribute to PhysioZoo.

Please include the standard citation to **PhysioZoo** when using the resources available on the platform:

Joachim A. Behar*, Aviv A. Rosenberg*, Ido Weiser-Bitoun, Ori Shemla, Alexandra Alexandrovich, Evgene Konyukhov, Yael Yaniv. 2018. 'PhysioZoo: a novel open access platform for heart rate variability analysis of mammalian electrocardiographic data.' Accepted for publication in Frontiers in Physiology.

*Equal contribution.

1.1 Installing PhysioZoo

The PhysioZoo user interface (`PZ-UI`) can be run from Matlab or be installed as a standalone application. If you do not need the graphic user interface then you can directly download the HRV source code (`mhrv` library). `PZ-UI` and `mhrv` were tested for Matlab 2016a and above running on Windows.

1.1.1 PhysioZoo `mhrv` matlab toolbox

The PhysioZoo platform includes the `mhrv` matlab toolbox which implements all the algorithmic functions required by PhysioZoo (and more). The toolbox is aimed at researchers or developers that wish to write their own code for ECG signal processing, HRV analysis or other applications by using the *mhrv* toolbox as a library.

See the toolbox's *Getting Started* page for installation and initial setup instructions. The API reference for the toolbox is available on this site.

The development version of the `mhrv` toolbox is available here.

1.1.2 PhysioZoo `PZ-UI` user interface

The PhysioZoo user interface (`PZ-UI`) is aimed mainly at researchers working with physiologic data who do not wish to write any code for data analysis. It provides many tools out of the box, all accessible from an interactive UI.

The UI can be run from Matlab or be installed as a standalone application. `PZ-UI` was tested for Matlab 2016a and above running on Windows.

Running `PZ-UI` from within Matlab

Versions supported: MATLAB 2016a and above.

1. Download or clone the source code from the repository.
2. From MATLAB launch PhysioZoo by running the script `PhysioZoo.m` from the root of the repo.

Note: The `PZ-UI` user interface has two modules: a `Peak detection` (used to process electrophysiological signals and obtain the RR time series) module and a `HRV analysis` module (used to process the RR time series and compute HRV measures).


Running `PZ-UI` as standalone software (.exe)

Operating system: Windows 10 (and above), 64 bits.

After downloading the software from the project's main page, run the `physiozoo.exe` file. This will start the installation process. You can then go through the following steps.

Note: Some of the screens might be slightly different depending on the version of Windows you are using and whether you already have the MATLAB runtime compiler installed. If Matlab runtime compiler is not installed then it will be done through this process but the installation might take some time.

---

1. If you get the following screen displayed then click "Run"
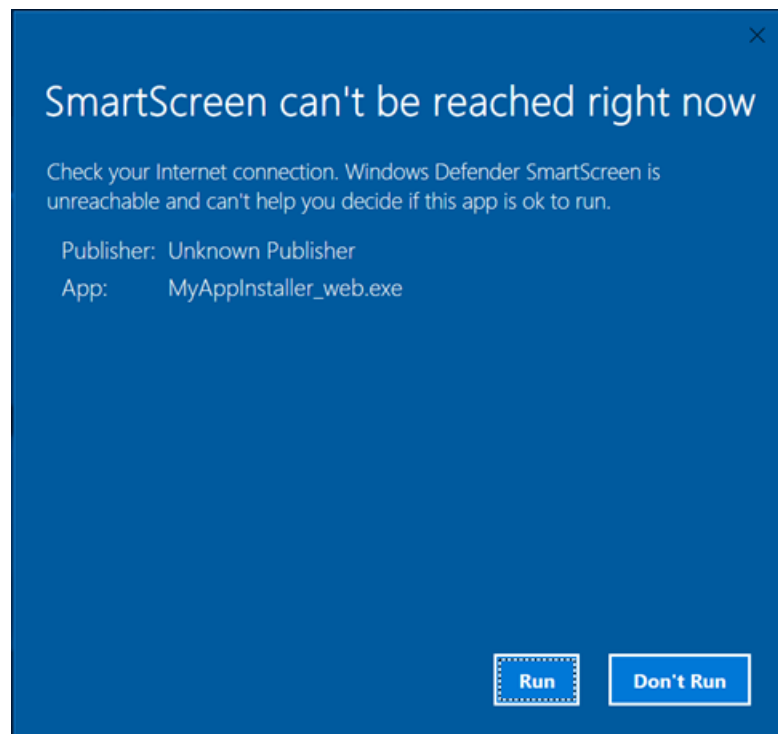
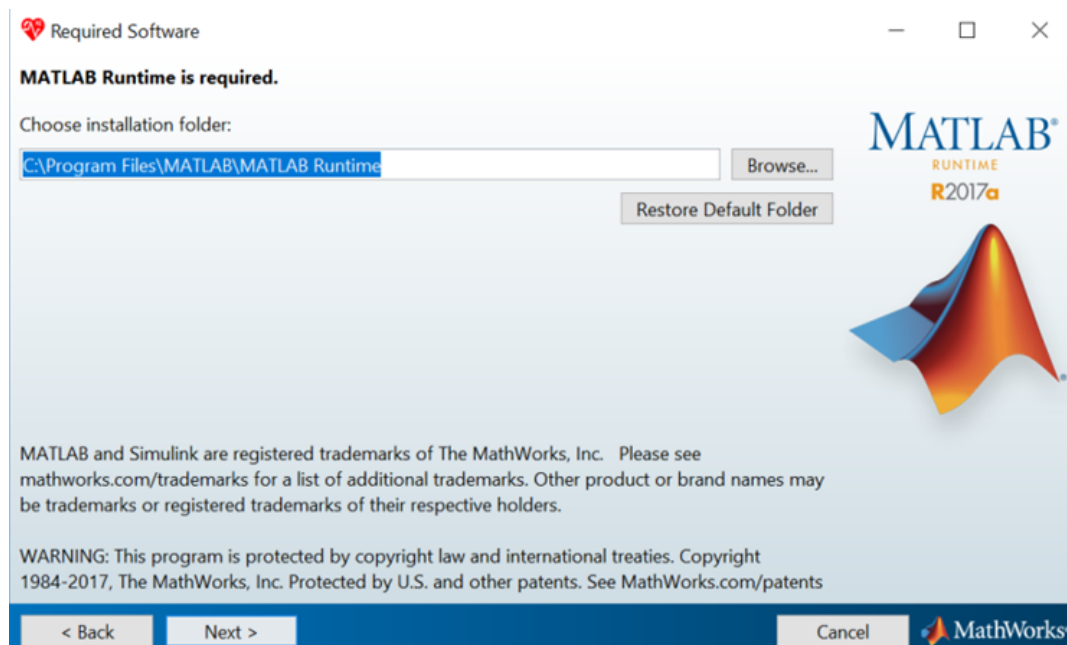2. When the following screen is prompted then click "Next"
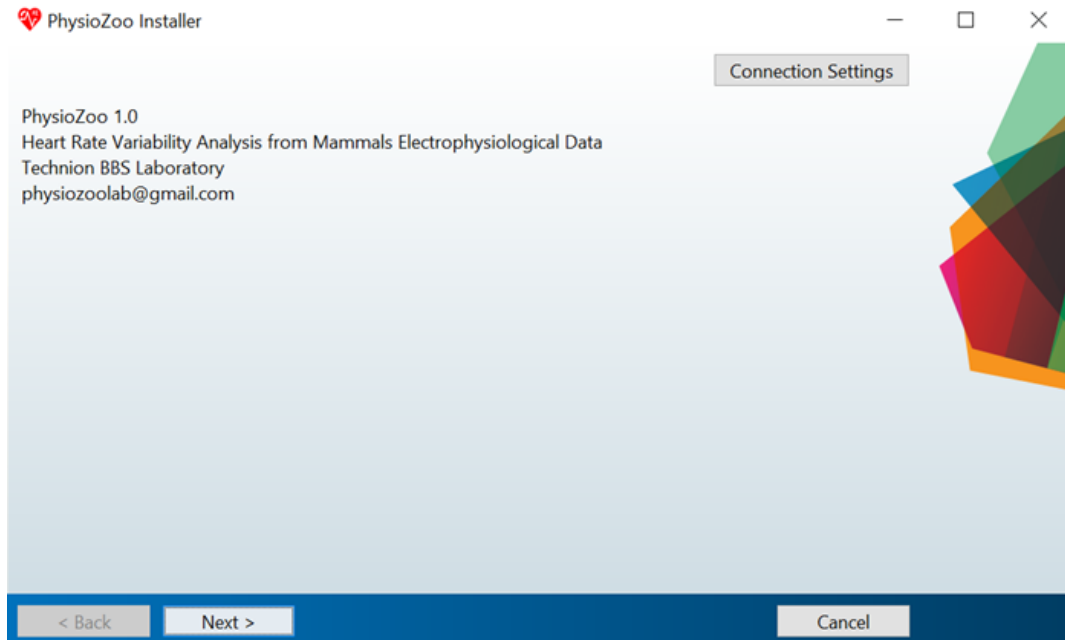
3. When the following screen is prompted then click "Next"

4. When the following screen is prompted check the "Yes" and then click "Next"
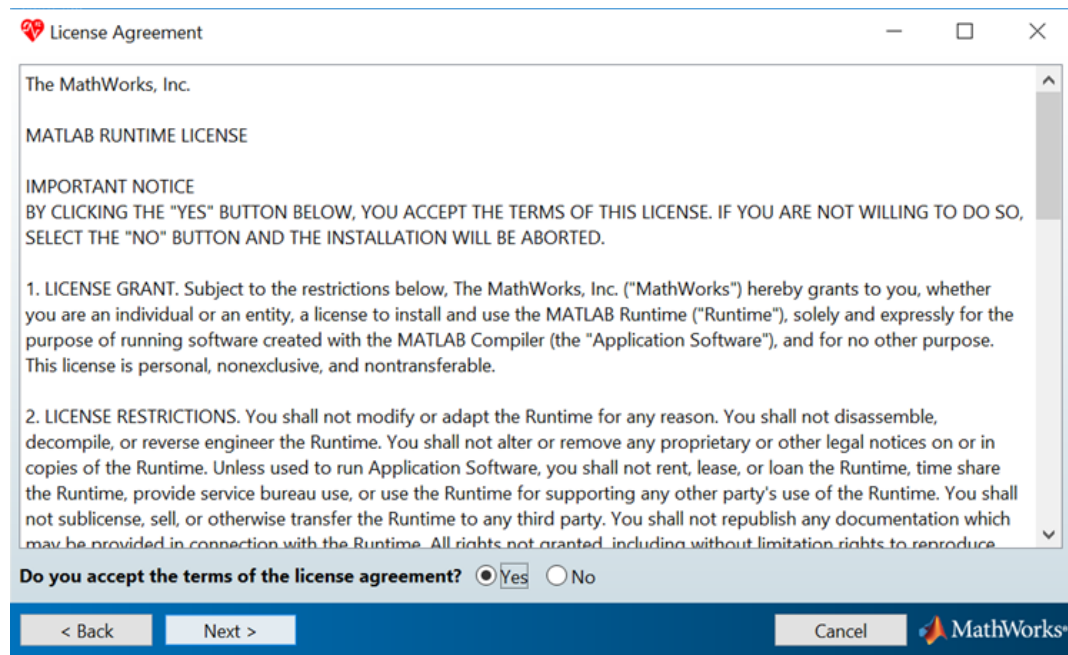
5. When the following screen is prompted then click "Install"
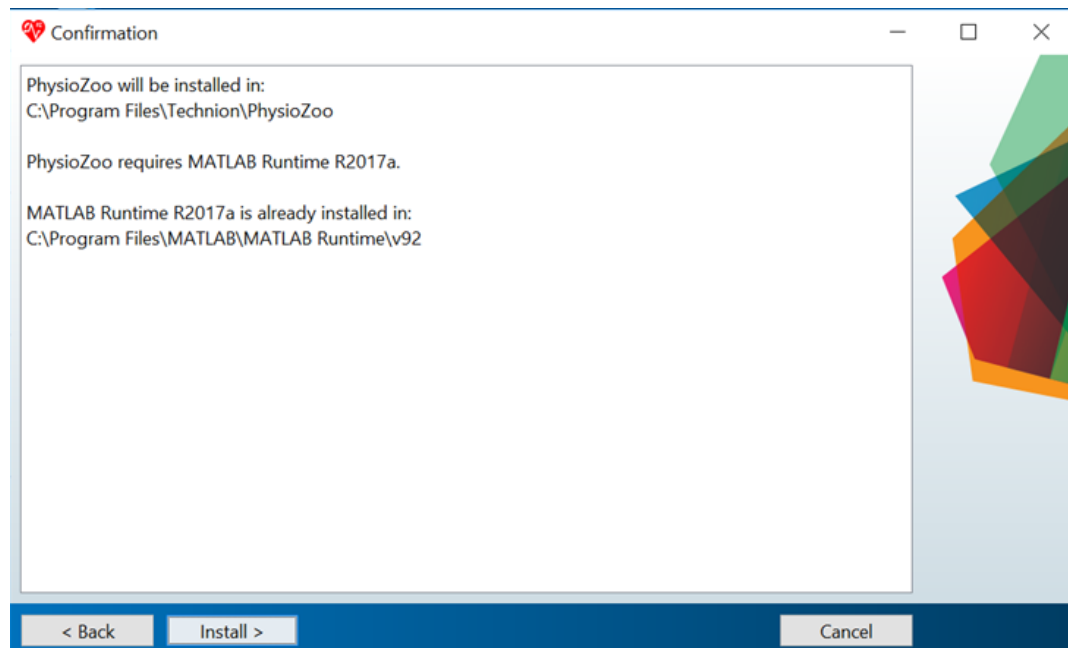
6. The installation will start. Wait until it is finished. Note that this might take quite some time.

7. When the installation is finished it will show the following screen.

You can now click on the **PhysioZoo** logo located on your desktop or in the list of programs.

1.2 Electrocardiographic recordings

Electrophysiological signals need to be digitized before they can be worked with. For that purpose both the range and domain need to be digitized. Important attention should be given to the sampling frequency and quantization level in order to ensure the integrity of the data and a meaningful HRV analysis.
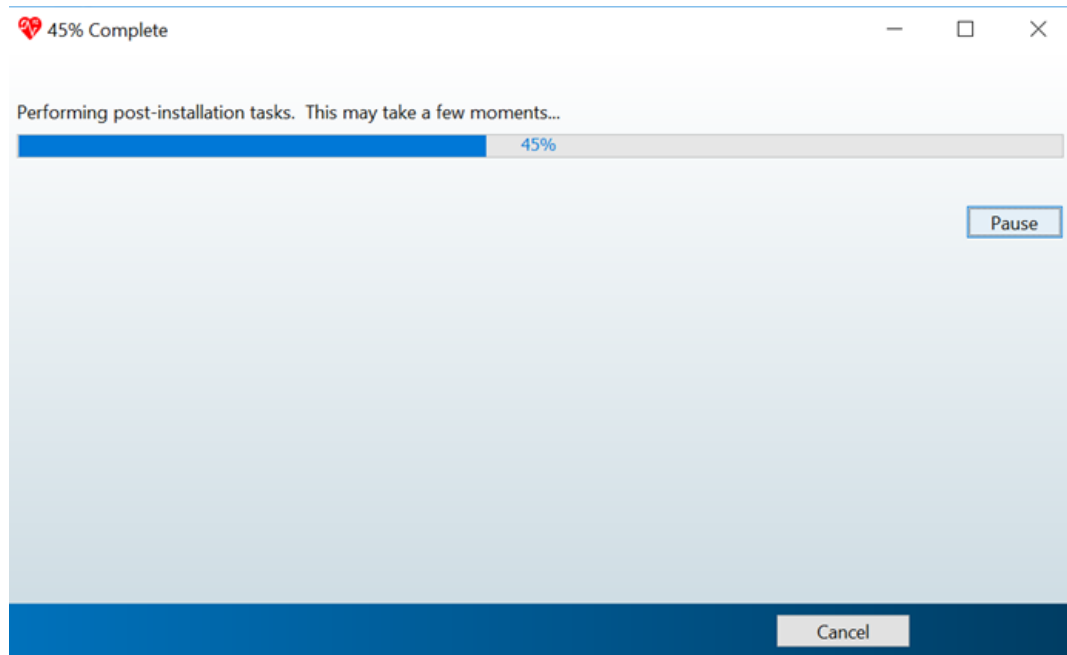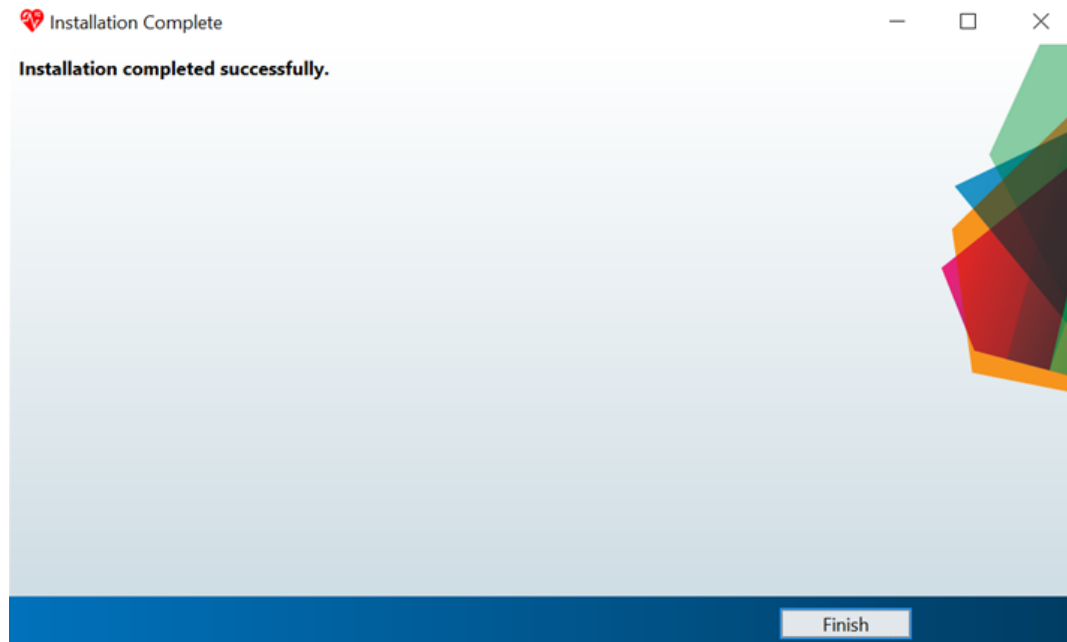
1.2.1 Sampling frequency and quantization level

Sampling refers to the process of discretizing the time domain. The sampled time domain will be described by the sampling frequency (fs) corresponding to the number of samples per second. The resulting digital signal will have a discrete time domain.

Quantization refers to the process of constraining an input from a continuous signal to a discrete set of values. The resulting digital signal will have a discrete range.

The sampling frequency and the quantization that are used for recording data will affect the quality of the HRV analysis and thus it is important to inspect the raw data that are being recorded and eventually, when necessary, adjust the sampling frequency and quantization level of the recording device.

1.2.2 Examples of typical issues

The following example shows what happens when a mouse ECG recording is downsampled from 1000 Hz to 500 Hz and 250 Hz, thus simulating the type of signal you would record if your original sampling rate was below 1000 Hz. As you will observe, the exact peak location is significantly affected by the sampling frequency. It is degraded at 500 Hz and at 250 Hz it will be very poor. Thus for mouse data, 1000 Hz is really the minimal sampling frequency to use in order to ensure a meaningful HRV analysis. However, note that the adequate sampling frequency will be mammal specific.
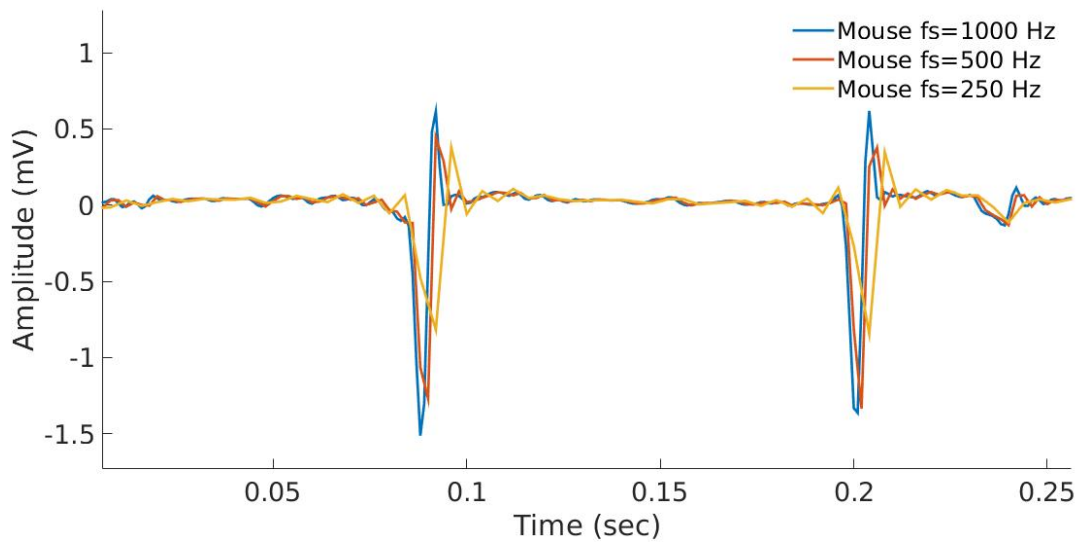
The following example shows a rabbit recording where the quantization level was poorly set. This can be observed by the low discretization levels / high amplitude `jumps' between consecutive samples. A poor quantization can affect the HRV analysis (as well as any morphological analysis).
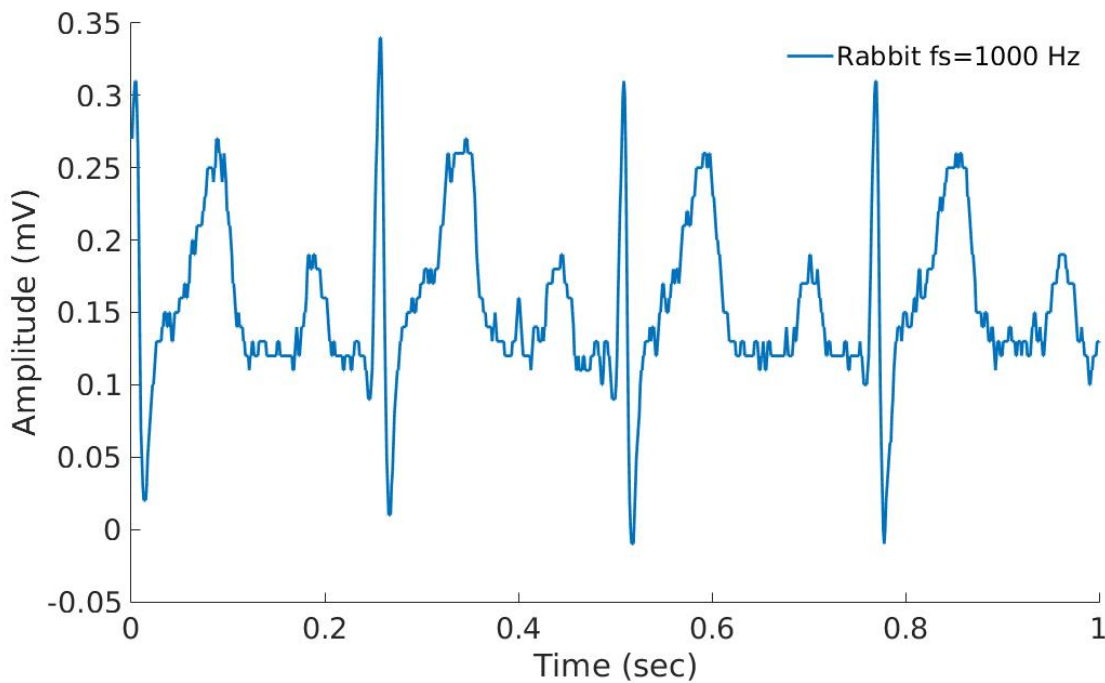
1.3 Peak detection

In this tutorial you will learn how to load an electrocardiogram (ECG) recording and perform R-peak detection. You will also learn how to manually correct misdetected peaks. Peak detection is performed in the `Peak detection` module.

1.3.1 Introduction

Accurate peak detection is critical for performing meaningfull HRV analysis. **PhysioZoo** provides a module to perform accurate peak detection from different mammalian species and the option to manually correct misdetected peaks to ensure the accuracy of the RR time series that will be analysed in the HRV analysis module.

1.3.2 Why mammal specific peak detection?

Numerous algorithms for finding R-peaks in human ECGs have been developped. However, these need to be adapted to the different dynamics accross mammalian species. **PhysioZoo** provides a set of peak detection algorithms (`rqrs`, `jqrs` and `wjqrs`) with parameters preset for human, dog, rabbit and mouse. Suitable parameters for other mammals can be specified by the user from the configuration panel.

Representative example of the R-peaks detected (A) using the original `gqrs` human peak detector and (B) using the `rqrs` **PhysioZoo** detector with parameters adapted for the different mammals. This figure illustrates the need for mammal-specific R-peak detectors to ensure a correct evaluation of the RR time series.
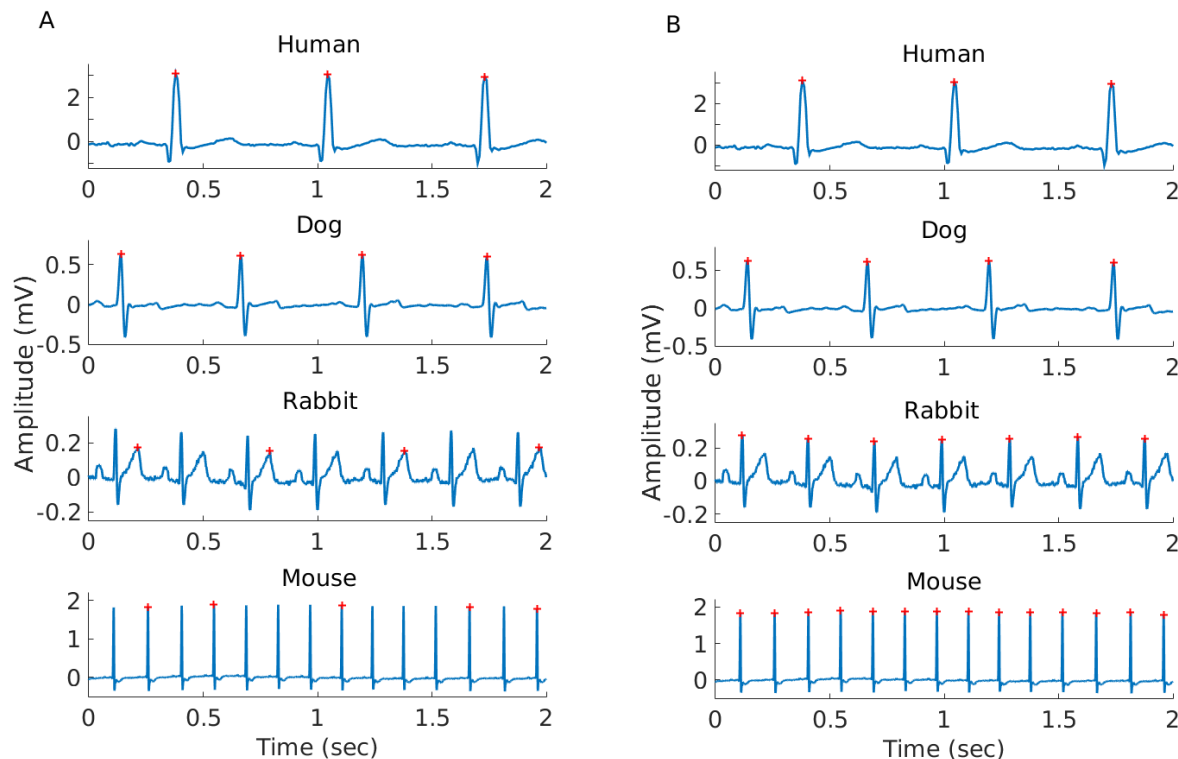
1.3.3 Performing peak detection in PhysioZoo

1. Open the PhysioZoo user interface `PZ-UI`.

2. Click "Peak Detection" on the Menu bar to open the peak detection interface.

3. Load an ECG recording (e.g. File -> Open File -> Rabbit_example.txt). The ECG will be displayed. The default R-peak detector (`rqrs`) will run automatically. You will see some red crosses appearing on the ECG signal at the locations that have been detected (panel A).

4. You can browse through the recording by stretching and moving the red window displayed in panel B. You can also move through the recording by using the arrows circled in red on the right hand side. Finally, you can also use the mouse scroll wheel and Ctrl scroll wheel.
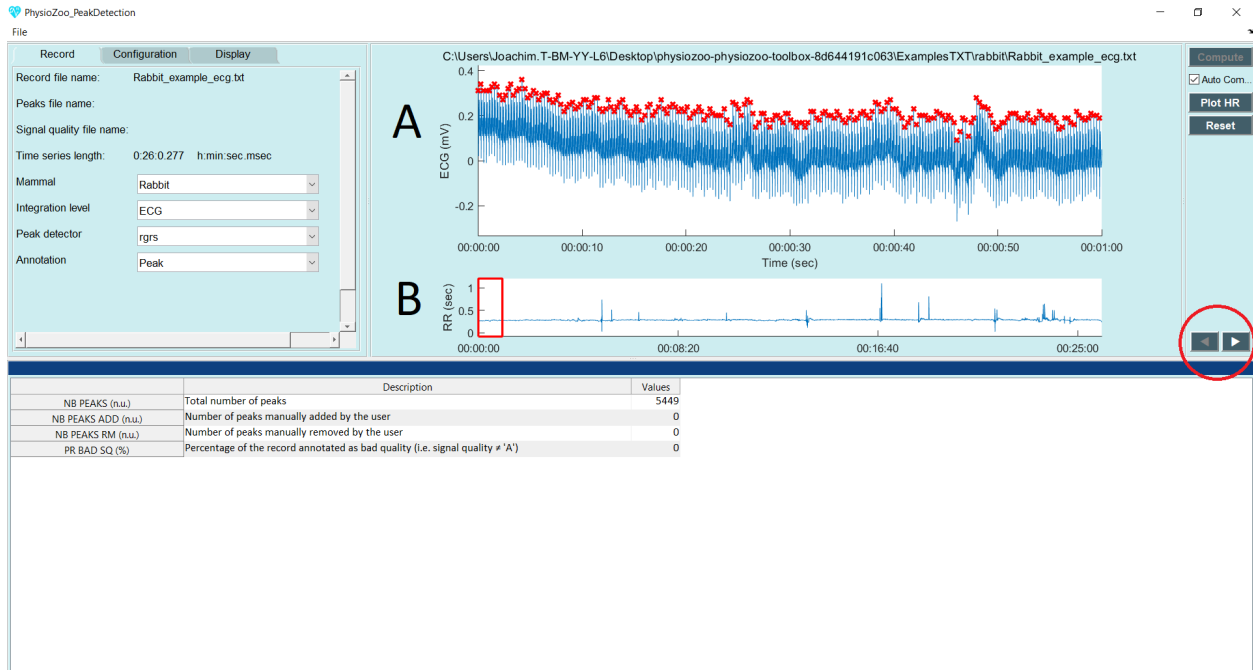
Warning: The input ECG data imported in **PhysioZoo** MUST be in mV (i.e. physiological units). The R-peak detector might not run appropriately if the data are not correctly scaled.

1.3.4 Manual peak correction

Sometimes the R-peak detector might make mistakes and miss some R-peaks/detect some points which are not peaks. You can fix these mistakes manually using the **PhysioZoo** interface by using the following functionalities:

1. Select a part of the ECG which is noisy and where some peaks were misdetected.

2. Point your cursor on a misdetected peak or at the location where a peak is missing. When you click, **PhysioZoo** will automatically remove/add a peak at this location.

3. If a whole segment contains misdetection and you need to clear all the peaks within this section, draw a rectangle on the area where you want the peaks to be deleted (see dotted rectangle figure below). When you 'drop the rectangle', all the peaks contained within it will be deleted.

4. Save your R-peak time series: File -> Save Peaks.
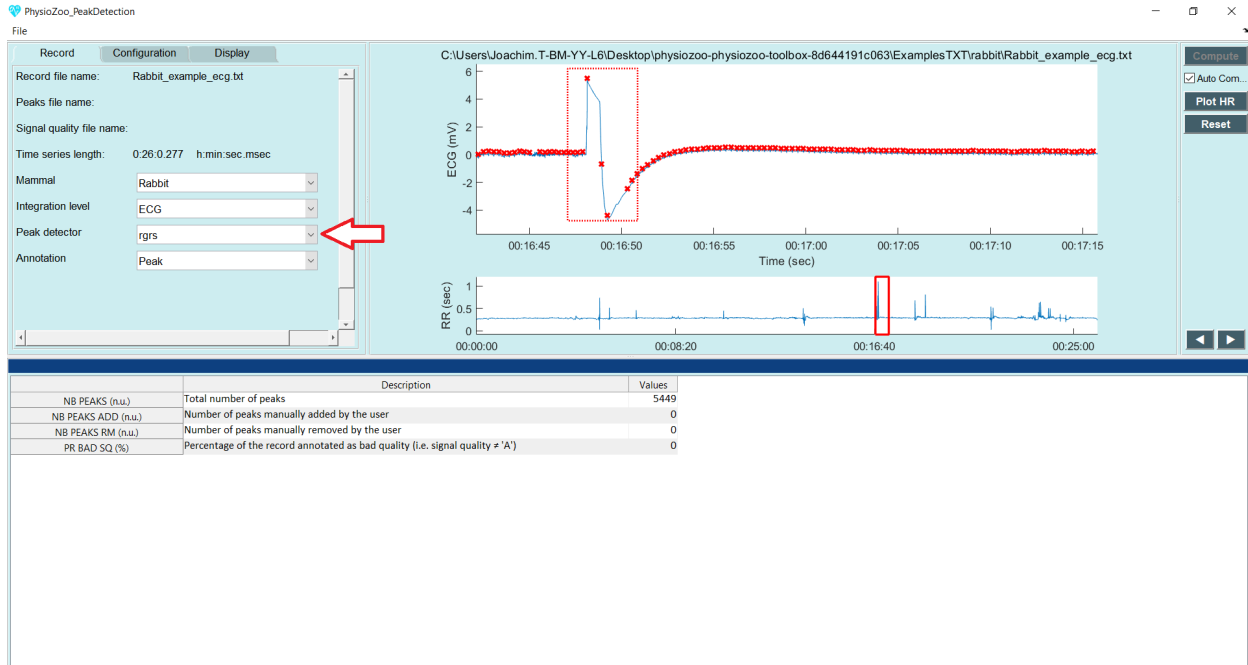
1.3.5 Configuration for other mammals

The **PhysioZoo** R-peak detectors are readily adapted for human, dog, rabbit and mouse ECG data. If you need to use them on another species then you will need to adapt their parameters accordingly. The parameters for the peak detectors can be found under the "Config Params" tab. You can save the configuration of the peak detectors in order to re-use it by going to File -> Save configuration file.

For `rqrs`:

- HR: Typical heart rate (beats/min)

- QS: Typical QRS duration (sec)

- QT: Typical QT interval duration (sec)

- QRSa: Typical QRS peak-to-peak amplitude (microVolt)

- QRSamin: Minimum QRS peak-to-peak amplitude (microVolt)

- RRmin: Minimum RR interval ("refractory period", sec)

- RRmax: Maximum RR interval (sec)

For `jqrs` and `wjqrs`:

- Lower cutoff frequency: the lower cutoff frequency of the bandpass filter used to prefilter the ECG (Hz)

- Upper cutoff frequency: the upper cutoff frequency of the bandpass filter used to prefilter the ECG (Hz)

- Threshold: the energy threshold (nu)

- Refractory period: the minimal time interval tolerated between two consecutive peaks (sec)

- Window size: this parameter is only used with `wjqrs`. This peak detector is applied on consecutive (non-overlapping) windows of size 'Window size' (sec).

Other:

- Peaks window: when manually correcting peaks, this parameter corresponds to the window size for which to look for a local maximum / minimum around the click location.

1.3.6 Frequently asked questions

What is a suitable sampling frequency for my data?

In order to locate the peaks accurately from the electrophysiological signal it is important to work with data sampled at a sufficiently high frequency. For example, for animal data with a high heart rate such as the mouse ECG, the QRS is only a few milliseconds long (~ 7 ms). Thus even at a relatively high sampling rate such as 1000 Hz the QRS will only be described by very few (~7) samples. See this tutorial for a visual example.

What's the difference between peak detectors?

Sometime a peak detector will fail to detect the R-peaks. This is due to the fact that these detectors were originally built for human ECG analysis and extended to work with mammalian data. In animal ECGs the position and type (e.g. subcutaneous) of the electrodes are not as standardized as for the uman. For that reason we included a set of three R-peak detectors so that it is possible to change to the one that is performing best for your specific dataset and electrodes configuration.

How best to deal with long recordings?

If you deal with long recordings (i.e. hours long) then use `rqrs` or `wjqrs`.

1.4 Preprocessing

In this tutorial you will learn how to preprocess your RR time series within the `HRV analysis` module. The resulting preprocessed time series is traditionally called the 'NN' time series.

1.4.1 Introduction

When loading an RR time series in the HRV module, one of the parameters you need to set is the preprocessing method.

We implemented in **PhysioZoo** three methods (and one combination of methods) for pre-filtering the RR-interval time series:

- Range,
- Moving average,
- Quotient,
- Combined (the combination of the Range and the Moving average filters).

1.4.2 Why is preprocessing important?

A preprocessing step is usually performed in HRV analysis to filter out suspected ectopic beats, missed beats and artifacts.

The example below shows an ECG segment with some transient noise. Note the miss-detected R-peaks. Leaving these miss-detected R-peaks could lead to the evaluation of incorrect HRV measures.
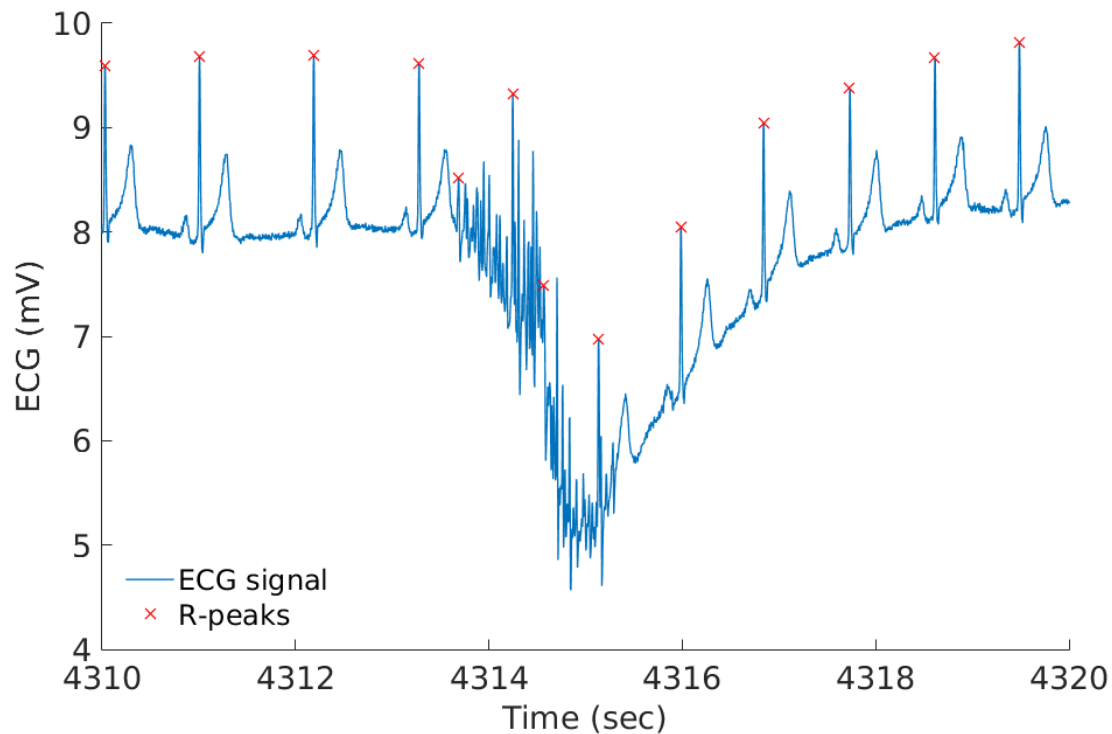
Below, the corresponding RR interval time series that is filtered. The filtering allows to remove the sudden increase and decrease in the RR time series that are caused by the miss-detected beats.
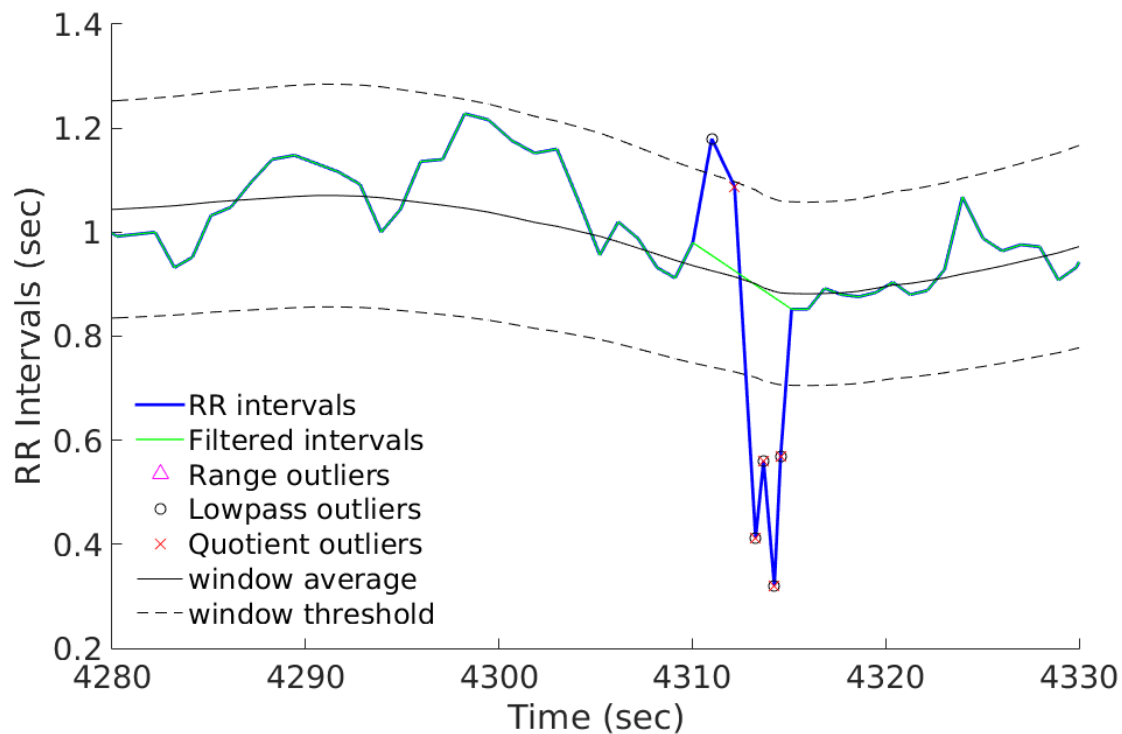
1.4.3 Preprocessing methods in PhysioZoo

You can select the preprocessing filter with the dropdown "Preprocessing" located under the Main tab. You can vary the level of preprocessing by using the dropdown "Preprocessing level" located under the Main tab. Three levels of preprocessing are available: "Weak", "Moderate" and "Strong". If you want to customize the preprocessing filters further then you can access each of the parameters of each filter by going to Options -> Filtering.

The image below shows an example of a rabbit RR time (blue time series) series which has been filtered over the selected window (green time series) using the Moving average filter.

1.4.4 Frequently asked questions

Why preprocessing the RR time series?

Preprocessing methods are useful to remove local miss-labelled beats caused by noise or sudden drop/increase in the RR time series due to ectopic beats.

What is the limitation of preprocessing routines?

In the presence of large segments of noise it will not help. Rather such large segments of bad quality data should be discarded from the analysis. You can annotate the quality of the electrophysiological time series in order to have this contextual information when performing HRV analysis.

Is there instances when not to preprocess the RR time series?

If your aim is to use HRV measures for identifying certain arrythmic episodes for example then you might want to avoid preprocessing the RR time series or to use a weak preprocessing in order to only remove outliers beats. Indeed, in this use case you will want to capture the important irregularities (e.g. atrial fibrillation) in the RR intervals of the time series.

1.5 Heart Rate Variability analysis

In this tutorial you will learn how to compute the Heart Rate Variability (HRV) measures and export the computed values within the **PhysioZoo** `HRV analysis` module. You will learn how to analyse a single window of defined length and a series of consecutive windows.

1.5.1 Introduction

HRV analysis is particularly interesting because it provides a non-invasive way to monitor cardiac function and the autonomic system activity. In particular, in the context of animal studies, HRV tools can be used to study the effects of mutations and pharmacological treatments. **PhysioZoo** provides the framework and tools for performing HRV analysis from human and animal model data.

1.5.2 Why mammal specific HRV?

Because the heart rate range, dynamics and autonomic control can vary significantly between mammals, some HRV measures need to be adapted to the mammal whose data are being analysed. As an example, the following figure illustrates the important differences in the location of the characteristic power spectral peaks in analysing RR time series from different mammals using power spectral analysis. This highlights the need to re-define the spectral bands (VLF, LF and HF) for each mammal.
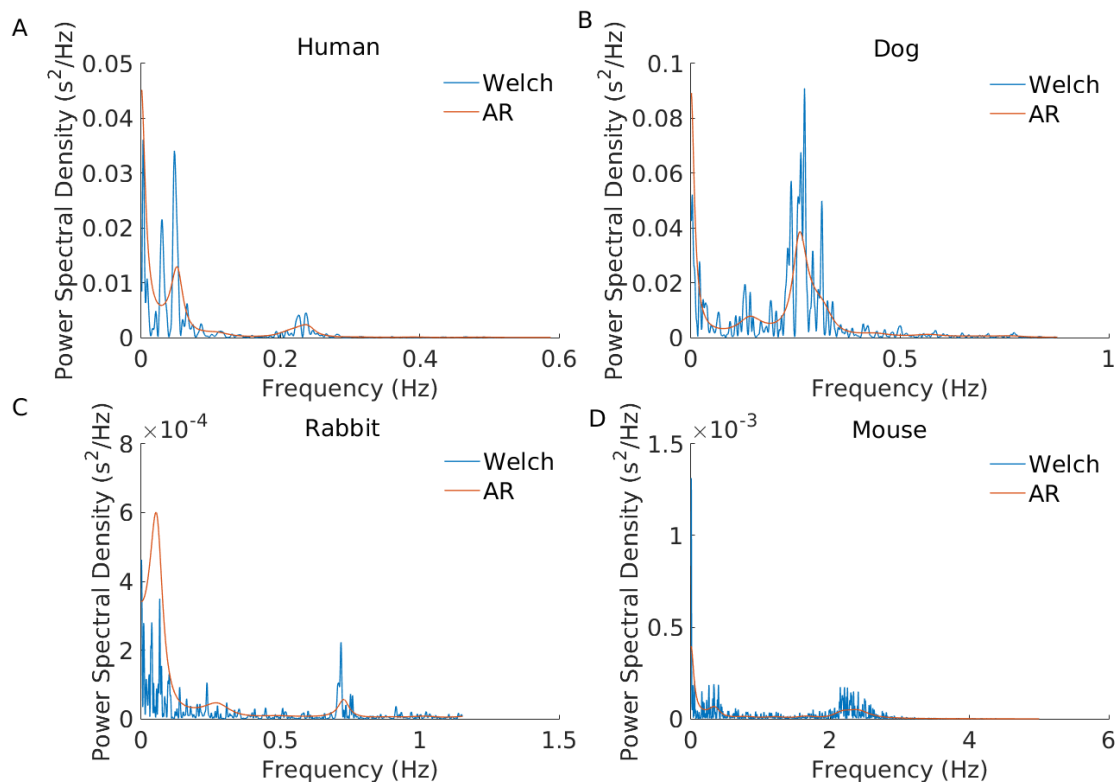
1.5.3 Performing HRV analysis

Start by loading some example data by clicking File -> Open data file -> mouse/Mouse_example_qrs.txt. The program will start the analysis automatically and display the following window:
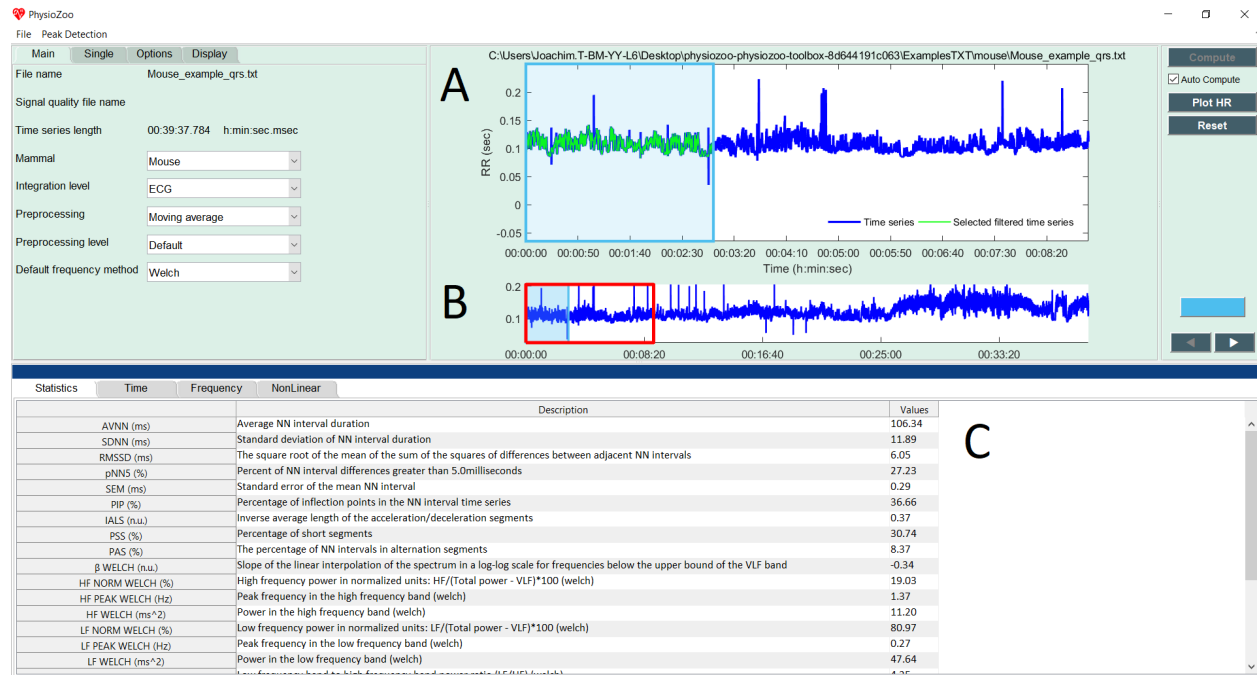
In the upper figure (A) the selected window (colored in blue) defines the time interval for which the HRV measures are computed. In the lower panel (B), the RR time series is plotted. Two windows are drawn on it: one window with a red frame and one with a blue frame (and alpha color from within.) The red window defines the part of the RR time series plotted in the larger upper figure (A). The blue frame defines the part of the RR time series for which the HRV measures will be computed. The window can be modified (extended/shrunk/moved) using the mouse. Pannel (C) shows all the HRV measures that have been computed.

Congrats! You have made your first HRV analysis with **PhysioZoo**!

Note: Every time you move the analysis window to another location the newly selected segment will be automatically analyzed. You can disable this by deselecting the checkbox "Auto Compute" located under the "Compute" button.

Note: The length of the selected window is important. Some HRV measures assume that the RR time series is stationary over the selected window. In our context stationary means that the statistical properties of the RR time series (such as mean and standard deviation) are about constant. Other measures such as the detrended fluctuation analysis ones do not assume stationarity and thus a long window may be used.

1.5.4 Exporting HRV measures

You can export the HRV measures and prefiltered NN intervals generated by **PhysioZoo**. Go to File -> Save HRV measures. The following window will display:
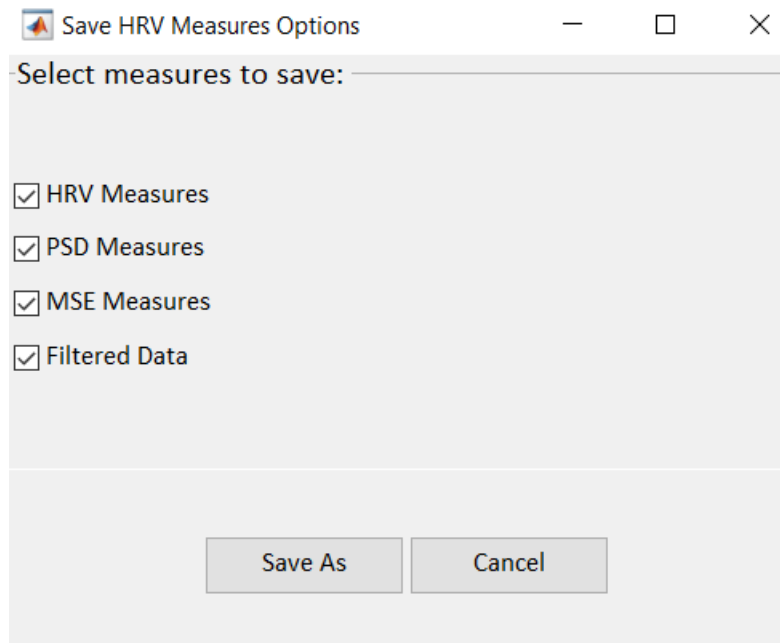
You can select the computation(s) you want to save with the checkboxes:

- **HRV Measures**: The HRV measures as displayed in the 'Statistics section' of the user interface.

- **Power Spectral Density**: the measure of signal's power content versus frequency. This corresponds to saving the power content for each frequency bin of the power spectral density displayed under the 'Frequency tab'.

- **Multi Scale Entropy**: the value of sample entropy for a range of scales as specified under Options->NonLinear and displayed under the 'NonLinear tab'.

- **Preprocessed RR intervals**: the RR interval time series after it was preprocessed by the filter you specified under Main-> Preprocessing. The resulting preprocessed time series is traditionally called the 'NN' time series.

Select the location where you want to save the HRV measures. Open the HRV Measures file and you will see the list of HRV measures together with their definitions and values for the window that was selected. For that purpose open the text file (Mouse_example_qrs_hrv.txt) you just saved.

```
Mouse_example_qrs_hrv - Notepad
File  Edit  Format  View  Help
AVNN (ms)               :    Average NN interval duration                                                                              106.34
SDNN (ms)               :    Standard deviation of NN interval duration                                                                11.89
RMSSD (ms)              :    The square root of the mean of the sum of the squares of differences between adjacent NN intervals        6.05
pNN5 (%)                :    Percent of NN interval differences greater than 5.0milliseconds                                           27.23
SEM (ms)                :    Standard error of the mean NN interval                                                                    0.29
PIP (%)                 :    Percentage of inflection points in the NN interval time series                                            36.66
IALS (n.u.)             :    Inverse average length of the acceleration/deceleration segments                                          0.37
PSS (%)                 :    Percentage of short segments                                                                              30.74
PAS (%)                 :    The percentage of NN intervals in alternation segments                                                    8.37
BETA_WELCH (n.u.)       :    Slope of the linear interpolation of the spectrum in a log-log scale for frequencies below the upper bound of the VLF band  -0.34
HF_NORM_WELCH (%)       :    High frequency power in normalized units: HF/(Total power - VLF)*100 (welch)                               19.03
HF_PEAK_WELCH (Hz)      :    Peak frequency in the high frequency band (welch)                                                         1.37
HF_POWER_WELCH (ms^2)   :    Power in the high frequency band (welch)                                                                  11.20
LF_NORM_WELCH (%)       :    Low frequency power in normalized units: LF/(Total power - VLF)*100 (welch)                                80.97
LF_PEAK_WELCH (Hz)      :    Peak frequency in the low frequency band (welch)                                                          0.27
LF_POWER_WELCH (ms^2)   :    Power in the low frequency band (welch)                                                                   47.64
LF_TO_HF_WELCH (n.u.)   :    Low frequency band to high frequency band power ratio (LF/HF) (welch)                                     4.25
TOTAL_POWER_WELCH (ms^2):    Total power (welch)                                                                                       117.94
VLF_NORM_WELCH (%)      :    Very low frequency power in normalized units: (welch)                                                     45.54
VLF_POWER_WELCH (ms^2)  :    Power in the very low frequency band (welch)                                                              53.71
BETA_AR (n.u.)          :    Slope of the linear interpolation of the spectrum in a log-log scale for frequencies below the upper bound of the VLF band  -0.85
HF_NORM_AR (%)          :    High frequency power in normalized units: HF/(Total power - VLF)*100 (ar)                                  20.95
HF_PEAK_AR (Hz)         :    Peak frequency in the high frequency band (ar)                                                            1.45
HF_POWER_AR (ms^2)      :    Power in the high frequency band (ar)                                                                     7.11
LF_NORM_AR (%)          :    Low frequency power in normalized units: LF/(Total power - VLF)*100 (ar)                                   79.05
LF_PEAK_AR (Hz)         :    Peak frequency in the low frequency band (ar)                                                             0.36
LF_POWER_AR (ms^2)      :    Power in the low frequency band (ar)                                                                      26.82
LF_TO_HF_AR (n.u.)      :    Low frequency band to high frequency band power ratio (LF/HF) (ar)                                        3.77
TOTAL_POWER_AR (ms^2)   :    Total power (ar)                                                                                          69.77
VLF_NORM_AR (%)         :    Very low frequency power in normalized units: (ar)                                                        47.74
VLF_POWER_AR (ms^2)     :    Power in the very low frequency band (ar)                                                                 33.31
SD1 (ms)                :    NN interval standard deviation along the perpendicular to the line-of-identity                            4.28
SD2 (ms)                :    NN interval standard deviation along the line-of-identity                                                 16.19
alpha1 (n.u.)           :    DFA low-scale slope                                                                                       1.32
alpha2 (n.u.)           :    DFA high-scale slope                                                                                      0.96
SampEn (n.u.)           :    Sample entropy                                                                                            0.88
```
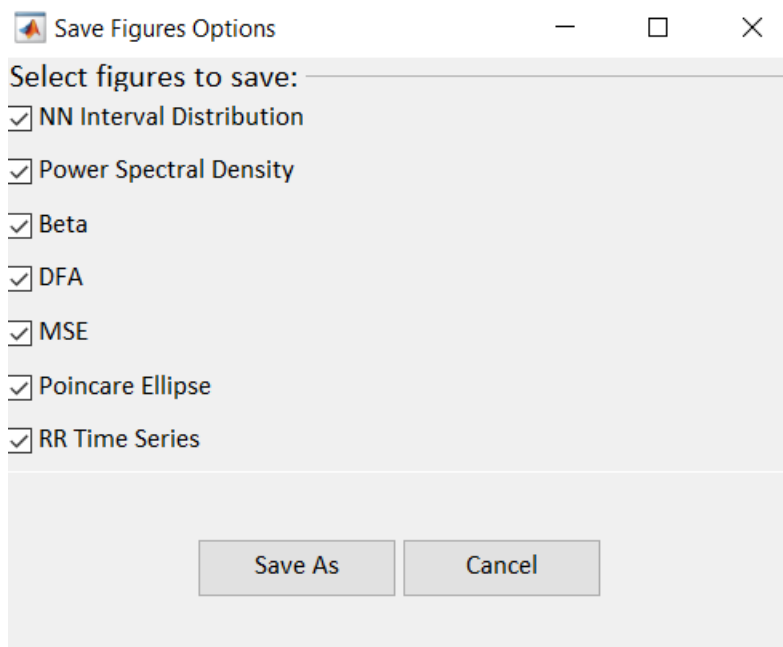
1.5.5 Exporting figures

Figures can be exported in high quality format and thus easily included in your research reports or papers. For exporting figures click File -> Export figures. You will be prompted with the following window on which you can choose what figures you want to export. After clicking 'Save As' you will be able to choose the format of the figure.

1.5.6 Consecutive windows analysis

You might want to track the evolution of the HRV measures over time: for example, if you are injecting some drug to the mammal and want to observe the resulting changes on the HRV measures over time. For that purpose **PhysioZoo** enables the analysis to be performed on consecutive segments.

Let's use a long RR time series: File -> Open data file -> Mouse_example_qrs

Click the "Single" Submenu in the left panel and click on the "Use full length" button. Then press the "Compute" button located on the top right of the interface. You will see the following screen, which contains all the HRV measures for each consecutive window over the whole recording.
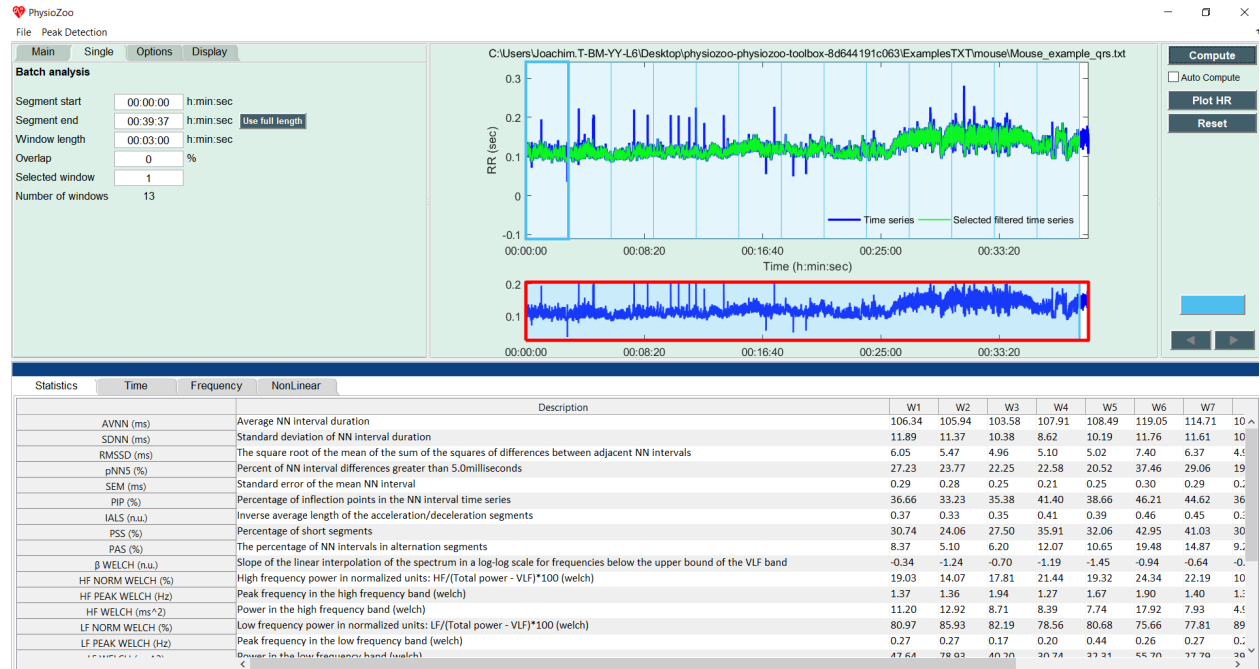
You can also enable the analysis window to be overlapping by using the "Overlap" entry in the Analysis menu. By default the value is 0% (i.e. no overlap between successive windows). Change it to 50% and re-run the analysis.
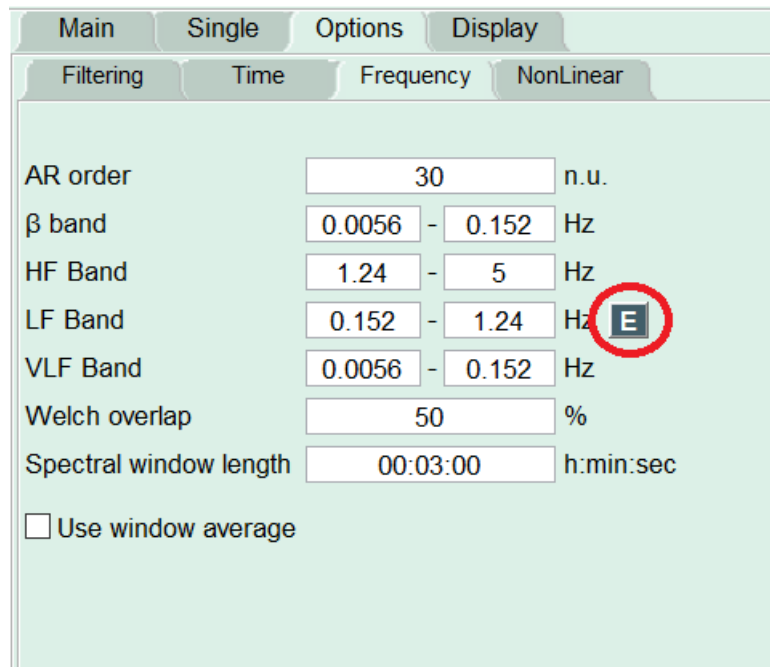
You can export all HRV measures from all consecutive windows (Main -> Save HRV measures as).

If you want to export figures, then select with the mouse the window you want to save the figures for.

Note: While using the successive segment analysis feature, it is possible to export the HRV measures and PSD from all the analyzed consecutive windows together. However, you can only export the figures from one single window at the time i.e. from the window that you select with the mouse (i.e. the 'selected window').

1.5.7 Adapting HRV measures to alternative mammals

If you want to adapt the HRV measures to other mammals you can create a new 'Configuration file' (see here) with parameters suitable to the specie you are studying. For some of the HRV measures (power frequency bands and pNNxx) you will find a button 'E' next to their parameters value under Options->Time/Frequency. You can use click on this button to adapt the HRV measure to the alternative mammalian model you are using.

1.5.8 Video Example

1.6 Signal quality annotations

In this tutorial you will learn how to perform signal quality annotations within the **PhysioZoo** `Peak detection` module and how to use these for your analysis in order to provide contextual information.

1.6.1 Introduction

PhysioZoo allows you to annotate and load signal quality annotations, which contain time intervals with quality annotations made on the original electrophysiological signal time series. The annotations highlight parts of the RR time series which are not to be trusted because the underlying ECG was of bad quality and the peak detector might have failed in estimating the heart rate within these intervals or just because you want to exclude these segments because they correspond to some transition period during drug injection and so you decide to label these time periods as 'bad quality'.

1.6.2 Annotating signal quality

To annotate the quality of an electrophysiological signal time series, follow these steps:

1. Select the rabbit ECG example: File-> Open data file-> Rabbit_example.txt

2. Under the record panel, select Annotation -> Signal quality. See the red rectangle on the figure below.

3. Look for a segment with some bad quality data and draw a rectangle around it with the mouse. The background of the area you have selected will become red.
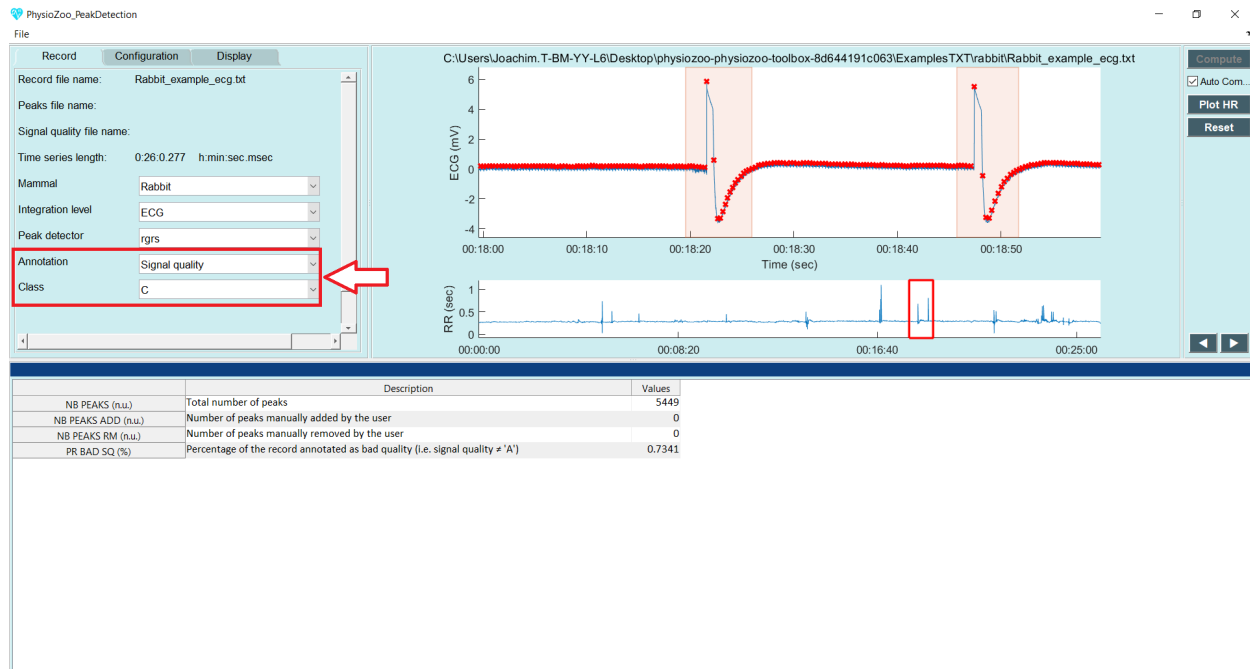
In **PhysioZoo** we define three levels of signal quality:

- "**A**": good quality signal where HRV and morphological analysis of the electrophysiological signal are possible. This means that the waveform is clean enough to detect the beats and to also study its morphology.

- "**B**": average quality signal where the beats can be detected and thus HRV analysis is possible but the signal quality is not good enough for morphological analysis.

- "**C**": bad quality signal. Neither HRV or morphological analysis is possible.

1.6.3 Loading signal quality

When using a recording for which you have performed signal quality annotations, you can load the signal quality annotations: Open -> Open signal quality file.

After the quality annotations are loaded, you will see a bar with green and red parts appearing on the top of the RR interval time series figure. The part in green corresponds to good quality data (i.e. data which analysis can be trusted) and the part in red corresponds to bad quality data (i.e. data which should not be trusted.) In addition, the RR time series is highlighted in red for the intervals which are indicated as bad quality.

1.6.4 Frequently asked questions

When do we need to make signal quality annotations?

Small bad quality segments (few of beats) will be 'cleaned up' by the prefiltering step and so one should not worry too much about it. The signal quality annotations are more useful when a 'large' section of the recording is of bad quality in which case the prefiltering step will be useless or even misleading. Analysing such sections with large segments of bad quality data will provide meaningless results - the signal quality annotations are useful to prevent that.

1.7 Importing data

In this tutorial you will learn how to import data in **PhysioZoo**.

1.7.1 Introduction

Data loading in **PhysioZoo** is centralized and done through the `PZ Loader` tool. Three types of input formats are supported by **PhysioZoo** - see formats supported.

1.7.2 Concepts

- `PZ Loader`: three types of formats can be loaded in **PhysioZoo**. This is done through the `PZ Loader`. The `PZ Loader` will ensure that the data that enter the `PZ-UI` are properly formatted.

- Header: it is possible to include a header at the beginning of a record. This header will specify all the information that **PhysioZoo** needs to open a record. If all the information is specified in the header, then you will not need to go through populating the `PZ Loader` User Interface and the record will be directly opened in **PhysioZoo**.

See here for the formats supported and how to create **PhysioZoo** headers.

The figure below illustrates the pathway for loading a recording: the `PZ Loader` is used to load a recording from one of the supported formats. If no header is available, then the `PZ Loader` user interface is opened and the user is prompted to enter the necessary information. If the header is available, then the record is directly loaded in **PhysioZoo**.
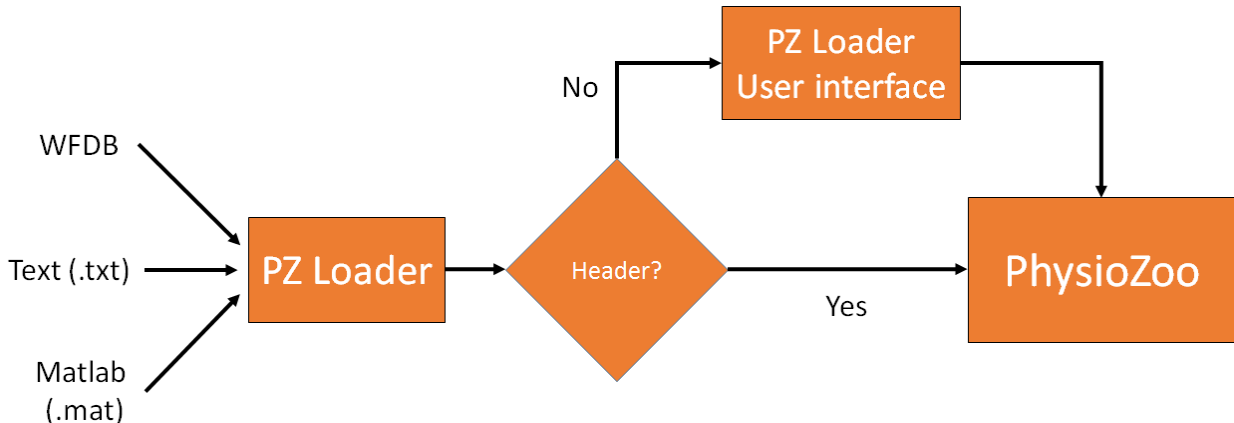
In this tutorial we will see how to make use of the `PZ Loader` user interface (UI) when no header is available.

1.7.3 PZ Loader User Interface

Open an example annotation file:

Download the following example file and load it in **PhysioZoo**.

In case some information is missing from the file (e.g. mammal type is not available in the header) then the `PZ Loader` will request the corresponding fields to be filled by the user. When all the fields have been filled then click OK and the data will be opened in **PhysioZoo**.

For the example provided, fill the fields as in the screenshot below and click OK:
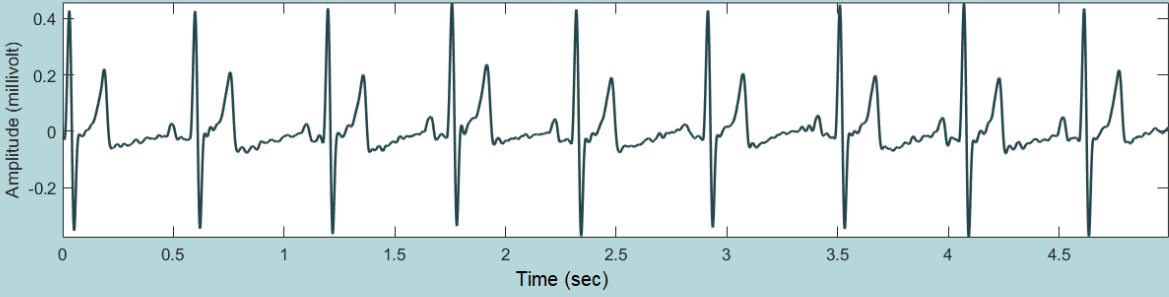
1.8 Formats supported

In this tutorial you will learn about how to prepare your data for importing them in **PhysioZoo**. Data samples are available here.

1.8.1 Introduction

PhysioZoo supports three types of formats:

- WFDB,

- text (.txt),

- MATLAB (.mat).

The .txt and .mat files structures are specific to **PhysioZoo** whereas the WFDB follows the standard structure of WFDB powered by Physionet (physionet.org) with some additional fields in the WFDB header files. If you are experienced with WFDB then we recommend you use this format.

For any of the three formats a file consists of a "header" and the "data":

```
---
Mammal:            dog
Fs:                500
Integration_level: electrocardiogram

Channels:

    - type:    time
      name:    time
      unit:    second
      enable:  yes

    - type:    electrography
      name:    data
      unit:    millivolt
      enable:  yes


---

0.000000    0.060057
0.002000    0.064323
0.004000    0.068589
0.006000    0.073524
0.008000    0.077790
0.010000    0.080550
0.012000    0.083561
0.014000    0.085987
0.016000    0.088747
0.018000    0.090002
0.020000    0.090002
```

Header

Data

Note: When downloading **PhysioZoo** a set of examples (WFDB,.txt and .mat) are provided with the structure that is needed in **PhysioZoo**. You can use these as examples for formatting your own recordings.

1.8.2 General structure of the Header

For each of the files the following information can be specified in the header:

- Mammal: the name of the mammal (e.g. 'dog', 'mouse', 'rabbit', 'human'),

- Fs: the sampling frequency in Hertz,

- Integration_Level: 'electrocardiogram', 'electrogram' or 'action potential'.

As part of the header, information characterizing each channel available can be entered:

- type: 'peak', 'signal quality', 'time', 'interval', 'beating rate' and 'electrography'. See definitions of these in the next section,

- name: a name you want to give to the channel (of note this information is not used by the PZ Loader to load the data),

- unit: 'millisecond', 'second', 'index', 'bpm', 'millivolt', 'microvolt' and 'volt',

- enable: 'yes' or 'no'. If you specify 'no' then this channel will be ignored when the file is loaded by the PZ Loader. Only specify 'yes' for the channels you want to use.

In the case all or some of this information is missing from the Header then you will be prompted to enter it through the PZ Loader User Interface.

1.8.3 Channels Type

There are two categories of Channels: 'Annotations' and 'Time series'. Within these two categories the Channel can be one of the following type's (specified in the header):

Annotations

- 'peak': the location of the peaks (e.g. R-peak from an ECG time series). The peak locations can be specified in millisecond/second or index.

- 'signal quality': annotation on the signal quality. The signal quality annotations can be specified in millisecond/second or index.

Time series

- 'time': a time vector giving the position of each sample of another time series. An entry of type 'time' needs to be associated with another time series and must be in units of seconds or milliseconds.

- 'interval': the time interval between consecutive beats (e.g. RR time series). The interval length can be specified in second/millisecond or index.

- 'beating rate': the reciprocal of the interval in units of beats per minute (e.g. heart rate).

- 'electrography': the amplitude of an electrography time series (e.g. ECG). The electrography amplitude is given in microvolt, millivolt or volt. Thus only physiological units are allowed.

1.8.4 Headers in the different formats

Explanation of the structure of the different formats supported is provided below. Examples in all formats are also available in the 'Examples' folder provided with the **PhysioZoo** download.

Text format (.txt)

```
---
Mammal:             dog
Fs:                 500
Integration_level: electrocardiogram

Channels:

    - type:    time
      name:    time
      unit:    second
      enable: yes

    - type:    electrography
      name:    data
      unit:    millivolt
      enable: yes

---

0.000000    0.060057
0.002000    0.064323
0.004000    0.068589
0.006000    0.073524
0.008000    0.077790
0.010000    0.080550
0.012000    0.083561
0.014000    0.085987
0.016000    0.088747
0.018000    0.090002
0.020000    0.090002
```
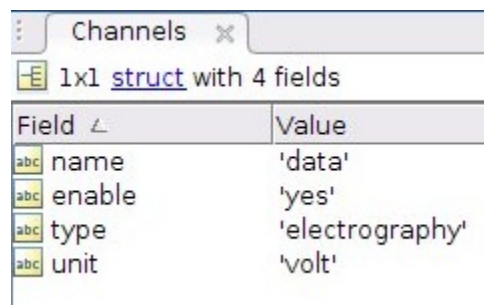
Header

Data

Matlab format (.mat)

A .mat file needs to contain the following fields:

```
Channels  ✕
⊟ 1x1 struct with 4 fields

Field ∠          Value
abc name         'data'
abc enable       'yes'
abc type         'electrography'
abc unit         'volt'
```

The Channels field is a Cell. Each element of the Channel Cell will contain the following fields : 'type', 'name', 'unit' and 'enable'.

WFDB (.hea)

WFDB files (annotation or data) will be accompanied by a header (.hea) file specifying the relevant information for reading an annotation or data file. Refer to the WFDB documentation for that. To the standard WFDB format of the header, you will need to add one comment line at the end of the header and starting '#' then followed by the mammal type and the integration level.

```
Dog_01 1 500 134258
Dog_01.dat 16 51300.0027(-1159)/mV 0 0 -1487 25886 0 Data
(electrography)
#Mammal:dog,Integration_level:electrocardiogram
```

1.8.5 Frequently asked questions

What if I am used to another format?

For now **PhysioZoo** only supports the WFDB, text and Matlab formats. Most commercial softwares enable you to export your data in text format which you can then import in **PhysioZoo** using the PZ Loader (see here).

Is there a way to import data without a formatted header?

Yes you can use the PZ Loader (see here) to import data which do not have a Header. You will use the PZ Loader User Interface to fill in the information needed and it will be opened in **PhysioZoo**.

1.9 Configuration files

In this tutorial you will learn how to create your custom configuration files for R-peak detection and for performing HRV analysis.

Note: In the next version of PhysioZoo the two configuration files will be merged in one unique configuration file.

1.9.1 Introduction

PhysioZoo enables you to work with HRV data from different mammals. Since the beating rate and its variability patterns vary across species then some parameters of the peak detectors and of the time, frequency and nonlinear HRV measures must be adjusted.

In the current version of the software these adaptations are readily available for the processing of Human, dog, rabbit and mouse electrocardiographic data. For processing data from other species or if you are using other data than ECG (e.g. electrogram, action potential) you will need to define your own configuration files. There are two types of configuration files in **PhysioZoo**: the peak detection configuration file and the HRV configuration file.
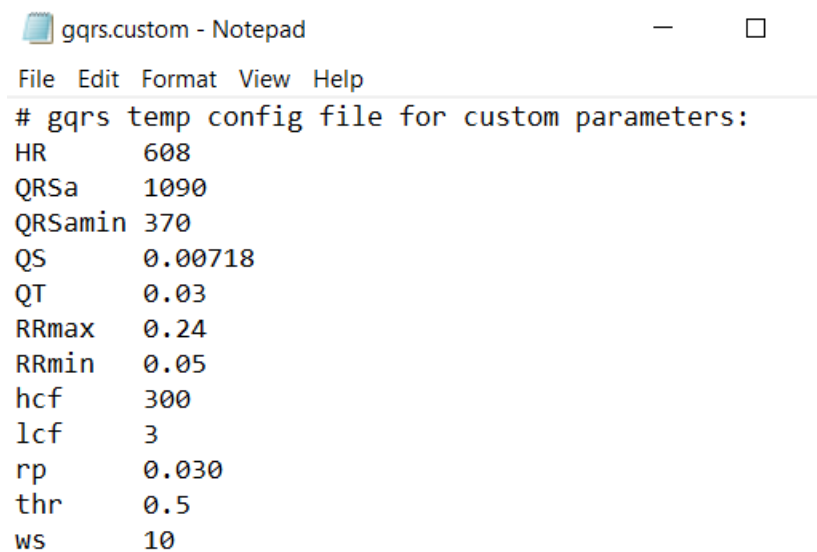
1.9.2 PhysioZoo software

Defining a peak detector configuration file

In the Peak detection module:

1. By choosing Mammal->Custom this will prompt you with a selection window. Select the configuration file of the closest mammal to the one you want to use. This will set the default HRV configuration parameters. Follow the following steps:

2. Modify the configuration of the peak detector you choose to use uder Configuration.

---

3. Save the peak detector configuration by clicking File -> Save configuration file.

4. To use this configuration file when you next use the **PhysioZoo** peak detection module: (1) File->Load custom config file or (2) Main->Mammal->Custom and choose your configuration file.

Defining an HRV configuration file

In the HRV module: under the "Main" tab you can change the Mammal type to one of the following options: Human, Dog, Rabbit, Mouse and Custom. In built in the software are configuration files for human, dog, rabbit and mouse electrocardiographic data. By choosing thee "Custom" option under `Mammal' you can define an HRV configuration file of your own. Follow the following steps:

1. By choosing Mammal->Custom this will prompt you with a selection window. Select the configuration file of the closest mammal to the one you want to use. This will set the default HRV configuration parameters.

2. Modify some of the HRV measures configuration by going to the "Options" menu (see screenshot below.) There you will find all the parameters of the HRV measures sorted with respect to their category: Time, Frequency and Nonlinear as well as the Preprocessing parameters. If, as an example, you want to change the cut-off frequency between the low frequency and high frequency bands ("LF Band" and "HF Band") in the frequency based methods then you can do so by changing the 0.341 Hz (see screen below) value to something else more suited to your data.

3. Go to File->Save config file and save the configuration file under the name of the mammal you are using.

4. To use this configuration file when you next use **PhysioZoo** HRV module: (1) File->Load custom config file or (2) Main->Mammal->Custom and choose your configuration file.

1.9.3 Frequently asked questions

Why configuration files?

The beating rate and its variability pattern vary across species thus some parameters of the peak detectors and of the time, frequency and nonlinear HRV measures must be adjusted.

1.10 Rhythm annotations

In this tutorial you will learn how to perform rhythms annotations from raw ECG recording within the **PhysioZoo** `Peak detection` module.

1.10.1 Introduction

PhysioZoo enables you to annotate and load rhythm annotations i.e. timestamped rhythm annotations marking the beginning and the end of a given rhythm. For example, rhythm annotations may have been produced by an original automated analysis machine (e.g. Holter ECG software) that you which to review and edit. The heart rhythms for wish annotations exist in **PhysioZoo** are:

Table 1: Available rhythms in **PhysioZoo**

Abbreviation	Condition
AFIB	Atrial fibrillation
AB	Atrial bigeminy
AFL	Atrial flutter
B	Ventricular bigeminy
BII	2° heart block
IVR	Idioventricular rhythm
NOD	Nodal (A-V junctional) rhythm
P	Paced rhythm
PREX	Pre-excitation (WPW)
SBR	Sinus bradycardia
SVTA	Supraventricular tachyarrhythmia
T	Ventricular trigeminy
VFL	Ventricular flutter
VT	Ventricular tachycardia
J	Junctional rhythm
PAT	Paroxysmal atrial tachycardia
AT	Atrial tachycardia
IVR	Idioventricular rhythm
AIVR	Accelerated idioventricular rhythm

1.10.2 Annotating rhythms

To annotate the quality of an electrophysiological signal time series, follow these steps:

1. Select the following ECG example: File-> Open data file-> physiozoo\ExamplesTXT\rhythms\ECG.txt.

2. Under the record panel, select Annotation -> Rhythms. See the red rectangle on the figure below. The different options will be displayed below in the same panel.

3. Draw a rectangle with your mouse on the desired window. The background behind the selected window will be displayed in color.

4. The rhythms can be exported under a .txt file: File -> Save Rhythms file (Ctrl+T).
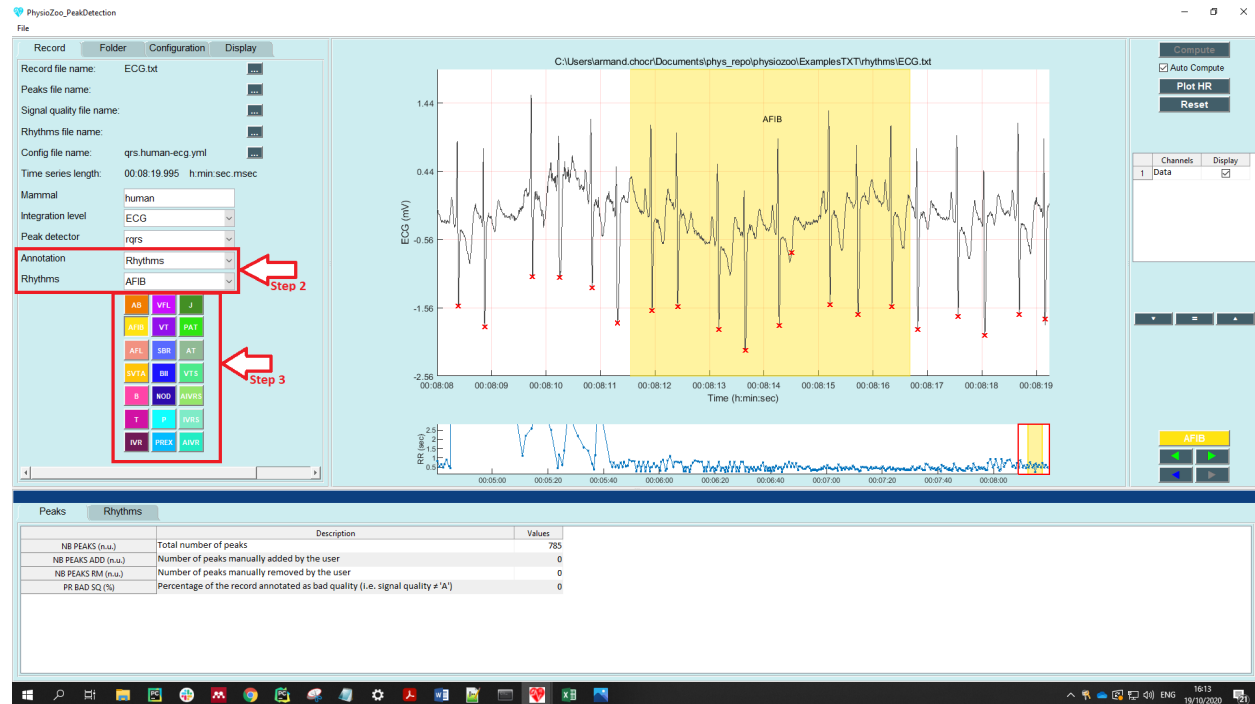
1.10.3 Loading rhythms

When using a recording for which you have performed rhythms annotations, you can load the rhythms annotations: Open -> Open rhythms file.

After the rhythms annotations are loaded, you will see a set of shaded areas with different colors appearing on the top of the RR interval time series figure as well as on the ECG raw data. These areas correspond to the rhythms annotations and each color is used to represent a different rhythm. These annotations may be further edited/saved.

On the bottom panel, you can find summary statistics for each rhythm: burden, minimal interval length, maximal interval length and 5 summary statistics.
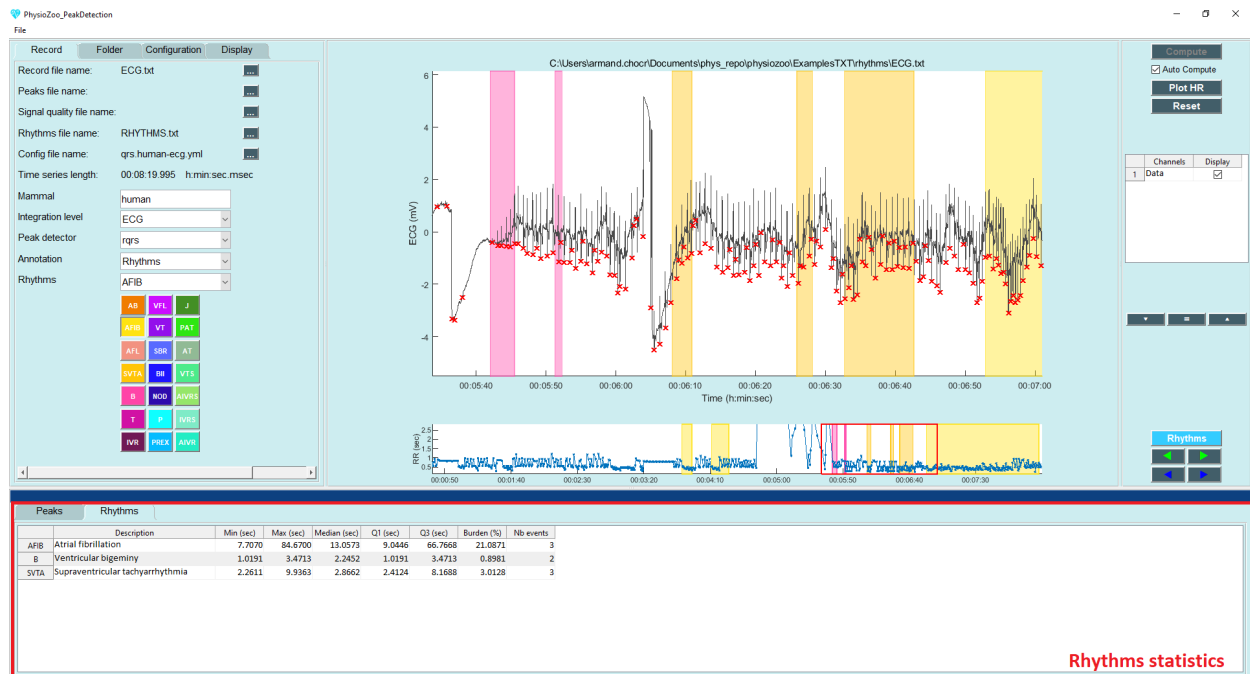
1.11 Multi-channel visualization

In this tutorial you will learn how to visualize several ECG channels within the **PhysioZoo** `Peak detection` module.

1.11.1 Introduction

PhysioZoo allows you to visualize several ECG channels in parallel. This can be use to facilitate the review of Holter recordings for example. This tutorial will show how to display several channels in parallel.

1.11.2 Displaying several channels

To display several channels of an ECG, follow these steps:

1. Select the multi-channel ECG example: File-> Open data file-> physiozoo\ExamplesTXT\multi-lead\multi-lead.txt.

2. On the right panel, select the desired leads to be displayed. By default, only channel 1 is selected.
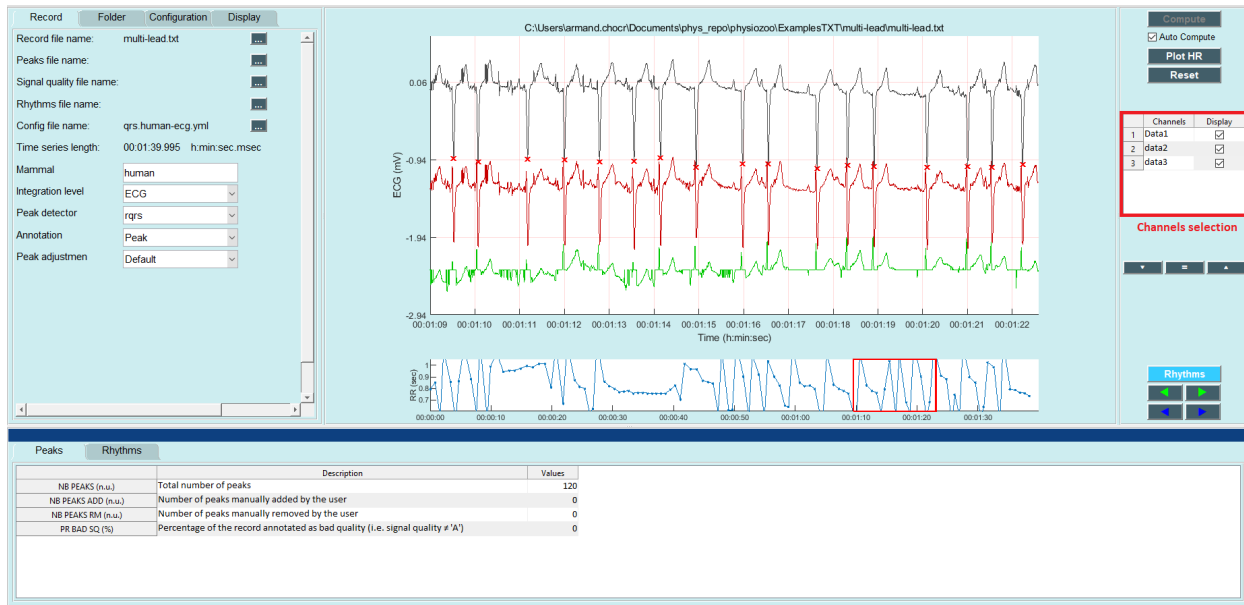
1.11.3 Frequently asked questions

What is the format of the files to be loaded to PhysioZoo to display several channels ?

If recordings are provided in .mat format then each column will represent a different channel.

An other option is to use a .txt file which will contain the following header:

Mammal: **Mammal type (human, rabbit...)**
Fs: **Sampling frequency**
Integration_level: electrocardiogram

Channels:

> - type: electrography
>> name: **Channel name**
>> unit: **Units (usually mV)**
>> enable: yes

> - type: electrography
>> name: **Channel name**
>> unit: **Units (usually mV)**
>> enable: yes

> - type: electrography
>> name: **Lead name**
>> unit: **Units (usually mV)**
>> enable: yes

—

Following this header, the user shall place the samples in 3 distinct columns, separated by a space. The user is invited to open the 'multi-lead.txt' example to visualize the file structure.

MHRV TOOLBOX DOCUMENTATION

`mhrv` is a matlab toolbox for calculating Heart-Rate Variability (HRV) metrics from both ECG signals and RR-interval time series. The toolbox works with ECG data in the PhysioNet WFDB data format.

2.1 Features

- **WFDB wrappers and helpers**. A small subset of the PhysioNet WFDB tools are wrapped with matlab functions, to allow using them directly from matlab.

- **ECG signal processing**. Peak detection and RR interval extraction from ECG data in PhysioNet format.

- **RR-intervals signal processing**. Ectopic beat rejection, frequency filtering, nonlinear dynamic and fractal analysis.

- **HRV Metrics**. Calculating quantitative measures that indicate the activity of the heart based on RR intervals using all standard HRV metrics defined in the literature.

- **Configuration**. The toolbox is fully configurable with many user-adjustable parameters. Everything can be configured either globally with human readable YAML config files, or when calling the toolbox functions via matlab style key-value pair arguments.

- **Plotting**. All toolbox functions support plotting their output for data visualization. The plotting code is separated from the algorithmic code in order to simplify embedding this toolbox in other matlab applications.

- **Top-level analysis functions**. These functions work with PhysioNet records and allow streamlined HRV analysis by composing the functions of this toolbox. Supports multi-record batch analysis for calculating HRV features of large datasets.

2.2 Getting Started

2.2.1 Requirements

- Matlab with Signal Processing toolbox. Should work on Matlab R2014b or newer.

- The PhysioNet WFDB tools. The toolbox can install this for you.

2.2.2 Installation

1. Clone the repo or download the source code.

2. From matlab, run the `mhrv_init` function from the root of the repo. This function will:

   - Check for the presence of the WFDB tools in your system `PATH`. If WFDB tools are not detected, it will attempt to automatically download them for you into the folder `bin/wfdb` under the repository root.

   - Set up your MATLAB path to include the code from this toolbox.

Notes about matlab's `pwd` and `path`

Matlab maintains a PWD, or "present working directory". It's the folder you see at the top of the interface, containing the files you see in the file explorer pane. Type `pwd` at the matlab command prompt to see it's value.

Additionally, matlab maintains a PATH variable, containing a list of folders in which it searches for function definitions (similar to the shell PATH concept). Type `path` at the matlab command prompt to see it's value.

You don't need to change your `pwd` to the root of the repo folder for the toolbox to work. You can simple run the `mhrv_init` function from your current `pwd`, and it will take care of updating matlab's path. For example, if you cloned or downloaded the toolbox in the folder `/Users/myname/mhrv/`, you can run the following command from the matlab prompt:

```
run /Users/myname/mhrv/mhrv_init.m
```

After this the toolbox will be ready to use, regardless of your `pwd`.

Manual WFDB Installation (Optional)

The above steps should be enough to get most users started. If however you don't want `mhrv_init` to download the WFDB tools for you, or the automatic installation fails for some reason, you can install them yourself.

- On OSX, you can use homebrew to install it easily with `brew install wfdb`.

- On Windows and Linux, you should either download the WFDB binaries for your OS or compile them from source using the instructions on their website.

Once you have the binaries, place them in some folder on your `$PATH` or somewere under the repo's root folder (`bin/wfdb` would be a good choice as it's `.gitignore`d) and they will be found and used automatically. Or, if you would like to manually specify a path outside the repo which contains the WFDB binaries (e.g. `/usr/local/bin` for a homebrew install), you can edit defaults.yml and set the `mhrv.paths.wfdb_path` variable to the desired path.

For linux users it's recommended to install from source as the binaries provided on the PhysioNet website are very outdated.

2.3 `mhrv`

The top level analysis functions in `mhrv` can be used as a command-based user interface to the toolbox. The `mhrv()` and `mhrv_batch()` functions allow analysis of both ECG and R-peak annotation files in WFDB format and return all HRV metrics supported by the toolbox.

mhrv(*rec_name*, *varargin*)

Analyzes an ECG signal, detects and filters R-peaks and calculates various heart-rate variability (HRV) metrics on them.

Parameters

- **rec_name** – Path and name of a wfdb record's files e.g. db/mitdb/100 if the record files (both 100.dat and 100.hea) are in a folder named 'db/mitdb' relative to MATLABs pwd.

- **varargin** – Pass in name-value pairs to configure advanced options.

  - ecg_channel: The channel number to use (in case the record has more than one). If not provided, mhrv will attempt to use the first channel that has ECG data.

  - ann_ext: Specify an annotation file extention to use instead of loading the record itself (.dat file). If provided, RR intervals will be loaded from the annotation file instead of from the ECG. Default: empty (don't use annotation).

  - window_minutes: Split ECG signal into windows of the specified length (in minutes) and perform the analysis on each window separately.

  - window_index_offset: Number of windows to skip from the beginning.

  - window_index_limit: Maximal number of windows to process. Combined with the above, this allows control of which window to start from and how many windows to process from there.

  - params: Name of mhrv defaults file to use (e.g. 'canine'). Default '', i.e. no parameters file will be loaded. Alternatively, can also be a cell array containing the exact arguments to pass to mhrv_load_params. This allows overriding parameters from a script.

  - transform_fn: A function handle to apply to the NN intervals before calculating metrics. The function handle should accept one argument only, the NN interval lengths.

  - plot: true/false whether to generate plots. Defaults to true if no output arguments were specified.

  Returns

  - hrv_metrics: A table where each row is a window and each column is an HRV metrics that was calculated in that window.

  - hrv_stats: A table containing various statistics about each metric, calculated over all windows.

  - plot_datas: Cell array containing the plot_data structs for each window.

mhrv_batch(*rec_dir*, *varargin*)

Performs batch processing of multiple records with mhrv.

This function analyzes multiple physionet records with mhrv and outputs tables containting the results. The records to analyze can be subdivided into record types, in which case output tables will be generated for each record type. Optionally, an Excel file can be generated containing the results of the analysis, including a comparison of the results per group.

  Parameters

  - **rec_dir** – Directory to scan for input files.

  - **varargin** – Optional key-value parameter pairs.

    - rec_types: A cell array containing the names of the type of record to analyze.

    - rec_filenames: A cell array with identical length as 'rec_names', containing patterns to match against the files in 'rec_dir' for each 'rec_type'.

    - rec_transforms: A cell array of transform functions to apply to each file in each record (one transform for each rec_type).

    - ann_ext: Specify an annotation file extention to use instead of loading the record itself (.dat file). If provided, RR intervals will be loaded from the annotation file instead of

---

from the ECG. Can also be a cell-array of strings the same length as rec types. This
allows using a different annotator extension for each rec type. Default: empty string
(don't use annotation).

– min_nn: Set a minumum number of NN intervals so that windows with less will be dis-
carded. Default is 0 (don't discard anything).

– rr_dist_max: Sanity threshold for RR interval distribution. Windows where
mean(RR)+2std(RR) > rr_dist_max will be discarded.

– rr_dist_min: Similar to above, but will discard windows where mean(RR)-2std(RR) <
rr_dist_min.

– mhrv_params: Parameters pass into mhrv when processing each record. Can be either a
string specifying the parameters file name, or it can be a cell array where the first entry
is the parameters file name and the subsequent entires are key-value pairs that override
parameters from the file.

– skip_plot_data: Whether to skip saving the plot data for each record. This can reduce
memory consumption significantly for large batches. Default: false.

– writexls: true/false whether to write the output to an Excel file.

– output_dir: Directory to write output file to.

– output_filename: Desired name of the output file.

Returns

A structure, batch_data, containing the following fields:

- rec_types: A cell of strings of the names of the record types that were analyzed.

- rec_transforms: A cell array of the RR transformation functis used on each record type.

- mhrv_window_minutes: Number of minutes in each analysis windows that each record was
split into.

- mhrv_params: A cell array containing the value of the *params* argument passed to mhrv for
the analysis (see mhrv documentation).

- hrv_tables: A map from each value in 'rec_types' to the table of HRV values for that type.
- stats_tables: A map with keys as above, whose values are summary tables for each type.

- plot_datas: A map with keys as above, whose values are also maps, mapping from an indi-
vidual record filename to the matching plot data object (which can be used for generating
plots).

2.4 `mhrv.ecg`

Functions in this package work on raw ECG data passed in as matlab vectors.

bpfilt(*ecg*, *fs*, *lcf*, *hcf*, *nt*, *debug*)

This function [1]_ is made for prefiltering an ecg time series before it is passed through a peak detector. Of
important note: the upper cut off (hcf) and lower cutoff (lcf) of the bandpass filter that are applied, highly
depends on the mammal that is being considered. This is particularly true for the hcf which will be higher for a
mouse ECG file versus a Human ecg. This is because the QRS is `sharper' for a mouse ecg than for a Human
one. Of note, NaN should be represented by the value -32768 in the ecg (WFDB standard).

Parameters

- **ecg** – electrocardiogram (mV)

- **fs** – sampling frequency (Hz)

- **lcf** – low cut-off frequency (Hz)

- **hcf** – high cut-off frequency (Hz)

- **debug** – plot output filtered signal (boolean)

- **nt** – frequency to cut with a Notch filter (Hz). Leave the field empty ('[]') if you do not want the Notch filter to be applied.

Returns

- bpfecg: band pass filtered ecg signal

Example: preprocess an ecg recording from the mitdb (physionet.com) by applying a bandpass filter (0.5-100 Hz)

```
download_wfdb_records('mitdb', '105', '.');
[~,ecg,Fs]=rdsamp('mitdb/105',1);

bpfecg = bpfilt(ecg,Fs,0.5,100,[],1);
```

bsqi (*refqrs*, *testqrs*, *agw*, *fs*)
This algorithm can be used to estimate the signal quality of a single channel electrocardiogram **[3]_ [4]_**. It compares the agreement between the R-peak annotations (ann1 and ann2) made by two different R-peak detectors. If the two detectors agree locally then the signal quality (sqi) is good and if they disagree then it is likely because of some uderlying artifacts/noise in the signal. The limitation of this method is that when one of the two detectors fails for whatever reason other than the presence of noise then the quality will be zero.

Parameters

- **refqrs** – vector of QRS annotations from a first peak detector (in seconds)

- **testqrs** – vector of QRS annotations from a secon peak detector (in seconds)

- **agw** – agreement window (sec)

- **fs** – sampling frequency (Hz)

Returns

- F1: signal quality measure (nu)

- IndMatch: indices of matching peaks (nu)

- meanDist: mean distance between matching refqrs and testqrs peaks (sec)

Example: Comparing gqrs to wjqrs on a record from mitdb.

```
download_wfdb_records('mitdb', '105', '.');
[tm,ecg,Fs]=rdsamp('mitdb/105',1,'from',450000,'to',480000);
bpfecg = bpfilt(ecg,Fs,5,45,[],0); % prefilter in range [5 - 45] Hz
anns_jqrs = wjqrs(bpfecg,Fs,0.3,0.250,10); % wjqrs peak detector
anns_gqrs = gqrs('mitdb/105','from',450000,'to',480000); % gqrs peak detector
anns_gqrs = double(anns_gqrs);

[F1,~] = bsqi( anns_jqrs', anns_gqrs,0.050,Fs);
plot(tm,ecg);
hold on; plot(tm(anns_jqrs),ecg(anns_jqrs),'+r');
hold on; plot(tm(anns_gqrs),ecg(anns_gqrs),'+k');
legend(['ecg with quality ' num2str(F1)],'wjqrs','gqrs');
```

ecgsqi (*ann1*, *ann2*, *ecg*, *fs*, *varargin*)

> Computes bsqi [3]_ [4]_ to estimate the signal quality of an ECG signal. The bsqi index is computed over the whole ecg recording with a granularity specified by the user.

> **Parameters**
>> - **ann1** – vector of QRS annotations from a first peak detector (sample)
>>
>> - **ann2** – vector of QRS annotations from a secon peak detector (sample)
>>
>> - **ecg** – electrocardiogram time series (mV)
>>
>> - **fs** – sampling frequency of the ecg (Hz)
>>
>> - **varargin** – pass in name-value pairs to configure advanced options:
>>> - agw: agreement window tolerated between annotations from the two peak detectors in order for the two annotations to be considered in agreement (in seconds)
>>>
>>> - sw: the size of the window on which the signal quality is computed (in seconds). Default is 5 sec.
>>>
>>> - rw: granularity at which the sqi is computed (in seconds). By default the sqi is computed at every second.
>>>
>>> - mw: window for a post-processing median smoothing (in number of samples). By default there is no median post-processing.
>>>
>>> - thrsqi: final sqi threshold. The signal will be considered of good quality for all sqi value above this threshold (output sqi will be one for these). Conversely the signal will be considered of bad quality for all sqi value below this threshold (output sqi will be zero for these).
>>>
>>> - plot: true/false whether to generate a plot. Defaults to true if no output arguments were specified.

> **Returns**
>> - sqi: signal quality time series (nu, in range 0-1)
>>
>> - tsqi: time vector for the sqi vector (sec)

> Example: an ecg sample from the mitdb (physionet.org) is downloaded, preprocessed and two peak detectors are ran (wjqrs and gqrs). The two sets of R-peak annotations are used to compare the signal quality using the ecgsqi function.

```
download_wfdb_records('mitdb', '105', '.');
recordName = 'mitdb/105';
[~,ecg,Fs]=rdsamp(recordName,1);
bpfecg = bpfilt(ecg,Fs,5,45,[],0); % prefilter in range [5 – 45] Hz
anns_jqrs = wjqrs(bpfecg,Fs,0.3,0.250,10); % wjqrs peak detector
anns_gqrs = gqrs(recordName); % gqrs peak detector
anns_gqrs = double(anns_gqrs);

[ sqi, tsqi ] = ecgsqi( anns_jqrs', anns_gqrs, ecg, Fs, 'plot', true, 'mw', 1, 'sw
↪', 5, 'rw', 1, 'agw', 0.050);
```

jqrs (*ecg*, *fs*, *thr*, *rp*, *debug*)

> Implementation of an energy based qrs detector [1]_. The algorithm is an adaptation of the popular Pan & Tompkins algorithm [2]_. The function assumes the input ecg is already pre-filtered i.e. bandpass filtered and that the power-line interference was removed. Of note, NaN should be represented by the value -32768 in the ecg (WFDB standard).

Parameters

- **ecg** – vector of ecg signal amplitude (mV)
- **fs** – sampling frequency (Hz)
- **thr** – threshold (nu)
- **rp** – refractory period (sec)
- **debug** – plot results (boolean)

Returns

- qrs_pos: position of the qrs (sample)

Example: process an ecg recording from the mitdb (physionet.org). A bandpass filter is used to preprocess the ECG in the range [4 - 45]Hz. Then the peak detector is ran with a refractory period of 250 ms (a standard value for Human ECG analysis) is specified.

```
download_wfdb_records('mitdb', '105', '.');
[~,ecg,Fs]=rdsamp('mitdb/105',1);

bpfecg = bpfilt(ecg,Fs,4,45,[],0); % prefilter in range [4-45] Hz

qrs_pos = jqrs(bpfecg,Fs,0.4,0.250,1); % peak detection and plot
```

qrs_adjust (*ecg*, *qrs*, *fs*, *inputsign*, *tol*, *debug*)

This function **[1]_** is used to adjust the qrs location by looking for a local min or max around the input qrs points. For example, this is useful when parts of the qrs are located on a positive part of the R-wave and parts on a negative part of the R-wave in order to make them all positive or negative - this can be useful for heart rate variability analysis. It is also useful for template substraction in the context of non-invasive fetal ECG source separation in order to ensure the maternal template is well aligned with each beat.

Parameters

- **ecg** – vector of ecg signal amplitude (mV)
- **qrs** – peak position (sample number)
- **fs** – sampling frequency (Hz)
- **inputsign** – sign of the peak to look for (-1 for negative and 1 for positive)
- **tol** – tolerance window (sec)
- **debug** – (boolean)

Returns

- cqrs: adjusted (corrected) qrs positions (sample number)

Example:

```
download_wfdb_records('mitdb', '105', '.');
[~,ecg,Fs]=rdsamp('mitdb/105',1);
bpfecg = bpfilt(ecg,Fs,4,45,[],0); % prefilter in range [4-45] Hz
anns_jqrs = wjqrs(bpfecg,Fs,0.3,0.250,10); % jqrs running on each segment of 10↵
↪sec length

cqrs = qrs_adjust(ecg,anns_jqrs,Fs,-1,0.050,1);
```

wjqrs (*ecg*, *fs*, *thres*, *rp*, *ws*)

This function **[1]_** is used to run the jqrs peak detector using a sliding (non-overlapping) window. This is usefull

in the cases where the signal contains important artefacts which could bias the jqrs threshold evaluation or if the amplitude of the ecg is changing substantially over long recordings because of the position of the electrodes move for example. In these instances the adaptation of the energy threshold is useful.

> **Parameters**
>
> - **ecg** – ecg signal (mV)
> - **fs** – sampling frequency (Hz)
> - **thres** – threshold to be used in the P&T algorithm(nu)
> - **rp** – refractory period (sec)
> - **ws** – window size (sec)
>
> **Returns**
>
> - qrs: qrs location in nb samples (ms)

Example: perform peak detection on an ecg recording from the mitdb (physionet.org), A refractory period of 250 ms (a standard value for Human ECG) and a threshold of 0.3 are used.

```
download_wfdb_records('mitdb', '105', '.');
[~,ecg,Fs]=rdsamp('mitdb/105',1);
bpfecg = bpfilt(ecg,Fs,4,45,[],0); % prefilter in range [4-45] Hz

anns_jqrs = wjqrs(bpfecg,Fs,0.3,0.250,10); % jqrs running on each segment of 10
↪sec length
```

2.5 `mhrv.rri`

Functions in this package work on RR-interval data passed in as matlab vectors.

ansrr(*rri*, *freqs*, *varargin*)

> Add Noised Sines to RR interval time series. This function generates a signal of sines embedded in gaussian white noise and adds it to an RR interval time series.
>
> **Parameters**
>
> - **rri** – RR-intervals values in seconds.
> - **freqs** – Vector containing desired sine frequencies, in Hz.
> - **varargin** – Pass in name-value pairs to configure advanced options:
>   - var_r: Desired variance of the RR intervals. Can be used the scale the intervals before adding the noised sines. Leave empty to forgo scaling.
>   - mix: Mixture ratio between the given RR intervals and the generated sines. Default: 0.5 i.e. sines will have alf the variance of the RR intervals (after scaling).
>   - snr: Signal to Noise ratio of the white gaussian noise that will be added to the sines. Default: 2.
>
> **Returns**
>
> - rri_out: RR intervals after adding noised sines.

detrendrr(*rri*, *lambda*, *Fs*, *varargin*)

> A detrending method [4]_ for the RR intervals. This is used when analysing RR interval time series over a long window thus where the stationarity assumption is not valid anymore. Of note: a number of HRV methods such

---

as the fragmentation measures and the DFA measures do not assume the intervals to be stationary. Thus usage of this detrending tool is specific to what HRV measures are being studied. Detrending will also likely affect the low frequency fluctuation information contained in the VLF band.

Parameters

- **rri** – Vector of RR-interval dirations (seconds)

- **lambda** – lambda (lambda=10 for Human)

- **Fs** – the sampling frequency of the original ECG signal (Hz)

- **varargin** – Pass in name-value pairs to configure advanced options:

  - plot: true/false whether to generate a plot. Defaults to true if no output arguments were specified (boolean).

Returns

- rri_detrended: detrended rri interval (seconds)

Example: process an ecg recording sampled at 1000 Hz and specifying a refractory period of 250 ms (a standard value for Human ECG) and with a threshold of 0.5.

```
recordName = 'mitdb/105';
[ecg,fs,~] = rdsamp(recordName,1);
bpfecg = bpfilt(ecg,fs,5,45,[],0); % prefilter in range [5-45] Hz
anns_jqrs = wjqrs(bpfecg,fs,0.3,0.250,10); % jqrs running on each segment of 10␣
↪sec length
z = diff(anns_jqrs)/fs; % get the RR intervals

nni_detrend = detrend_nn(z',10,'Fs',Fs ,'plot',true); % detrend and plot
```

dfa (*t*, *sig*, *varargin*)

Detrended fluctuation analysis, DFA [1]_. Calculates the DFA of a signal and it's scaling exponents α_1 and α_2.

Parameters

- **t** – time (or x values of signal)

- **sig** – signal data (or y values of signal)

- **varargin** – Pass in name-value pairs to configure advanced options:

  - n_min: Minimal DFA block-size (default 4)

  - n_max: Maximal DFA block-size (default 64)

  - n_incr: Increment value for n (default 2). Can also be less than 1, in which case we interpret it as the ratio of a geometric series on box sizes (n). This should produce box size values identical to the PhysioNet DFA implmentation.

  - alpha1_range: Range of block size values to use for calculating the α_1 scaling exponent. Default: [4, 15].

  - alpha2_range: Range of block size values to use for calculating the α_2 scaling exponent. Default: [16, 64].

Returns

- n: block sizes (x-axis of DFA)

- fn: DFA value for each block size n

- alpha1: Exponential scaling factor

- alpha2: Exponential scaling factor

filtrr(*rri*, *trr*, *varargin*)

Calculate an NN-interval time series (filtered RR intervals). Performs outlier detection and removal on RR interval data. This function can perform three types of different outlier detection, based on user input: Range based detection, moving-average filter-based detection and quotient filter based detection.

Parameters

- **rri** – RR-intervals values in seconds.

- **trr** – RR-interval times in seconds.

- **varargin** – Pass in name-value pairs to configure advanced options:

  – filter_range: true/false whether to use range filtering (remove intervals smaller or larger than 'rr_min' and 'rr_max').

  – filter_ma: true/false whether to use a moving-average filter to detect potential outliers in the rr-intervals. If an interval is greater (abs) than 'win_percent' percent of the average in a window of size 'win_samples' around it, excludes the interval.

  – filter_quotient: true/false whether to use the quotient filter. This fliter checks the quotient between an interval and it's predecessor and successors, and only allows a configured change percentage ('rr_max_change') between them.

  – win_samples: Number of samples in the filter window on each side of the current sample (total window size will be 2*win_samples+1). Default: 10.

  – win_percent: The percentage above/below the average to use for filtering. Default: 20.

  – rr_min: Min physiological RR interval, in seconds. Intervals shorter than this will be removed prior to poincare plotting. Default: 0.32 sec.

  – rr_max: Max physiological RR interval, in seconds. Intervals longer than this will be removed prior to poincare plotting. Default: 1.5 sec.

  – rr_max_change: Maximal change, in percent, allowed between adjacent RR intervals. Intervals violating this will be removed prior to poincare plotting. Default: 25.

  – plot: true/false whether to generate a plot. Defaults to true if no output arguments were specified.

Returns

- nni: NN-intervals (RR intervals after removing outliers) values in seconds

- tnn: NN-interval times in seconds

freqband_detect(*pxx_dataset*, *varargin*)

FREQBAND_DETECT Detect frequency bands from multiple specrums of RR intervals This function uses an automated clustering algorithm to attempt to detect bands with high frequency power from a dataset containing multiple power spectrums obtained from RR intervals.

freqfiltrr(*rri*, *fc*, *varargin*)

Performs frequency-band filtering of RR intervals. This function can apply a low-pass or high-pass filter to an RR interval time series.

Parameters

- **rri** – RR-intervals values in seconds.

- **fc** – Filter cutoff frequency, in Hz.

- **varargin** – Pass in name-value pairs to configure advanced options:

- resamp_freq: Frequency to resample the RR-intervals at before filtering. Must be at least twich the maximal frequency in the signal. Default: 5 Hz.

- forder: Order (length in samples) of the filter to use. Default: 100.

- ftype: A string, either 'low' or 'high', specifying the type of filter to apply.

- plot: true/false whether to generate a plot. Defaults to true if no output arguments were specified.

Returns

- rri_out: RR intervals after filtering.

- trr_out: Times of filtered RR intervals, in seconds.

mse (*sig*, *varargin*)

Calculates the Multiscale Entropy [2]_, MSE, of a signal, a measure of the signals complexity. The algorithms calculates the Sample Entropy of the signal at various 'scales' from 1 to max_scale. At each scale, the signal is downsampled by averaging 'scale' samples and the Sample Entropy is calculated for the downsampled signal.

Parameters

- **sig** – Signal to calculate MSE for.

- **varargin** – Pass in name-value pairs to configure advanced options:

  - mse_max_scale: Maximal scale to calculate up to. Default: 20.

  - sampen_r: Value of 'r' parameter to use when calculating sample entropy (max distance between two points that's considered a match). Default: 0.2.

  - sampen_m: Value of 'm' parameter to use when calculating sample entropy (length of templates to match). Default: 2.

  - normalize_std: Whether or not to normalize the signal to std=1 before calculating the MSE. This affects the meaning of r.

  - plot: true/false whether to generate a plot. Defaults to true if no output arguments were specified.

Returns

- mse_result: The sample entropy value at each scale.

- scale_axis: The corresponding scale values that were used.

poincare (*rri*, *varargin*)

Calculates HRV metrics from a Poincaré plot of the input data.

Parameters

- **rri** – Row vector of RR-interval lengths in seconds.

- **varargin** – Pass in name-value pairs to configure advanced options:

  - sd1_factor: Factor to multiply the standard devation along the perpendicular line (SD1) to get the radius of the ellipse along that axis. Default is 2 (so over 95% of points will be inside the ellipse).

  - sd2_factor: As above, but for the standard deviation along the line of identity (SD2). Default: 3.

  - plot: true/false whether to generate a plot. Defaults to true if no output arguments were specified.

Returns

- sd1: Standard deviation of RR intervals along the axis perpendicular to the line of identity.

- sd2: Standard deviation of RR intervals along the line of identity.

sample_entropy(*sig*, *m*, *r*)
    Calculate sample entropy [3]_ (SampEn) of a signal. Sample entropy is a measure of the irregularity of a signal.

        Parameters

- **sig** – The input signal.

- **m** – Template length in samples.

- **r** – Threshold for matching sample values.

        Returns The sample entropy value of the input signal.

2.6 `mhrv.hrv`

This package provides high-level HRV analysis functionality.

hrv_fragmentation(*nni*, *varargin*)
    Computes HRV fragmentation indices [1]_ of a NN interval time series.

        Parameters **nni** – Vector of NN-interval dirations (in seconds)

        Returns

        Table containing the following fragmentation metrics:

- PIP: Percentage of inflection points.

- IALS: Inverse average length of segments.

- PSS: Percentage of NN intervals that are in short segments.

- PAS: Percentage of NN intervals that are in alternation segments of at least 4 intervals.

hrv_freq(*nni*, *varargin*)
    NN interval spectrum and frequency-domain HRV metrics This function estimates the PSD (power spectral density) of a given nn-interval sequence, and calculates the power in various frequency bands.

        Parameters

- **nni** – RR/NN intervals, in seconds.

- **varargin** – Pass in name-value pairs to configure advanced options:

    – methods: A cell array of strings containing names of methods to use to estimate the spectrum. Supported methods are:

      * `lomb`: Lomb-scargle periodogram.

      * `ar`: Yule-Walker autoregressive model. Data will be resampled. No windowing will be performed for this method.

      * `welch`: Welch's method (overlapping windows).

      * `fft`: Simple fft-based periodogram, no overlap (also known as Bartlett's method).

    In all cases, a window will be used on the samples according to the `win_func` parameter. Data will be resampled for all methods except `lomb`. Default value is `{'lomb', 'ar', 'welch'}`.

– time_intervals: specify the time interval vector tnn. If it is not specified then it will be computed from the nni time series.

– power_methods: The method(s) to use for calculating the power in each band. A cell array where each element can be any one of the methods given in 'methods'. This also determines the spectrum that will be returned from this function (pxx). Default: First value in `methods`.

– norm_method: A string, either `total` or `lf_hf`. If `total`, then the power in each band will be normalized by the total power in the entire frequency spectrum. If `lf_hf`, then only for the LF and HF bands, the normalization will be performed by the (LF+HF) power. This is the standard method used in many papers to normalize these bands. In any case, VLF and user-defined custom bands are not affected by this parameter.

– band_factor: A factor that will be applied to the frequency bands. Useful for shifting them linearly to adapt to non-human data. Default: 1.0 (no shift).

– vlf_band: 2-element vector of frequencies in Hz defining the VLF band. Default: [0.003, 0.04].

– lf_band: 2-element vector of frequencies in Hz defining the LF band. Default: [0.04, 0.15].

– hf_band: 2-element vector of frequencies in Hz defining the HF band. Default: [0.15, 0.4].

– extra_bands: A cell array of frequency pairs, for example `{[f_start,f_end], ...}`. Each pair defines a custom band for which the power and normalized power will be calculated.

– window_minutes: Split intervals into windows of this length, calcualte the spectrum in each window, and average them. A window funciton will be also be applied to each window after breaking the intervals into windows. Set to [] if you want to disable windowing. Default: 5 minutes.

– detrend_order: Order of polynomial to fit to the data for detrending. Default: 1 (i.e. linear detrending).

– ar_order: Order of the autoregressive model to use if `ar` method is specific. Default: 24.

– welch_overlap: Percentage of overlap between windows when using Welch's method. Default: 50 percent.

– win_func: The window function to apply to each segment. Should be a function that accepts one parameter (length in samples) and returns a window of that length. Default: @hamming.

– plot: true/false whether to generate plots. Defaults to true if no output arguments were specified.

Returns

- hrv_fd: Table containing the following HRV metrics:

    – TOTAL_POWER: Total power in all three bands combined.

    – VLF_POWER: Power in the VLF band.

    – LF_POWER: Power in the LF band.

    – HF_POWER: Power in the HF band.

    – VLF_NORM: 100 * Ratio between VLF power and total power.

– LF_NORM: 100 * Ratio between LF power and total power or the sum of LF and HF
power (see 'norm_method'). - HF_NORM: 100 * Ratio between HF power and total
power or the sum of LF and HF power (see `norm_method`).

– LF_TO_HF: Ratio between LF and HF power.

– LF_PEAK: Frequency of highest peak in the LF band.

– HF_PEAK: Frequency of highest peak in the HF band.

– BETA: Slope of log-log frequency plot in the VLF band.

Note that each of the above metrics will be calculated for each value given in
`power_methods`, and their names will be suffixed with the method name (e.g.
LF_PEAK_LOMB).

• pxx: Power spectrum. It's type is determined by the first value in `power_methods`.

• f_axis: Frequencies, in Hz, at which pxx was calculated.

hrv_nonlinear(*nni*, *varargin*)

Calcualtes non-linear HRV metrics based on Poincar?? plots, detrended fluctuation analysis (DFA) **[2]_** and
Multiscale Entropy (MSE) **[3]_**.

Parameters

• **nni** – RR/NN intervals, in seconds.

• **varargin** – Pass in name-value pairs to configure advanced options:

– mse_max_scale: Maximal scale value that the MSE will be calculated up to.

– mse_metrics: Whether to output MSE at each scale as a separate metric.

– sampen_r: `r` value used to calculate Sample Entropy

– sampen_m: `m` value used to calculate Sample Entropy

– plot: true/false whether to generate plots. Defaults to true if no output arguments were
specified.

Returns

• hrv_nl: Table containing the following HRV metrics:

– SD1: Poincare plot SD1 descriptor (std. dev. of intervals along the line perpendicular to
the line of identity).

– SD2: Poincare plot SD2 descriptor (std. dev. of intervals along the line of identity).

– alpha1: Log-log slope of DFA in the low-scale region.

– alpha2: Log-log slope of DFA in the high-scale region.

– SampEn: The sample entropy.

hrv_time(*nni*, *varargin*)

Calculates time-domain HRV mertics from NN intervals.

Parameters

• **nni** – Vector of NN-interval dirations (in seconds)

• **varargin** – Pass in name-value pairs to configure advanced options:

– pnn_thresh_ms: Optional. Threshold NN interval time difference in milliseconds (for the
pNNx HRV measure).

– plot: true/false whether to generate a plot. Defaults to true if no output arguments were specified.

Returns

Table containing the following HRV metrics:

- AVNN: Average NN interval duration.

- SDNN: Standard deviation of NN interval durations.

- RMSSD: Square root of mean summed squares of NN interval differences.

- pNNx: The percentage of NN intervals which differ by at least x (ms) (default 50) from their preceding interval. The value of x in milliseconds can be set with the optional parameter 'pnn_thresh_ms'.

- SEM: Standard error of the mean NN interval length.

2.7 `mhrv.wfdb`

This package provides wrappers for PhysioNet *wfdb* tools and custom functions which work with data in the PhysioNet format.

download_wfdb(*dest_base_dir*)
   Downloads the WFDB binaries for this OS. This function detects the current OS and attempts to download the approprate WFDB binaries. By default they will be downloaded into the folder 'bin/wfdb' under the current MATLAB directory.

   Parameters dest_base_dir – Optional. Directory to download into. Will be created if it doesn't exist. A folder name 'wfdb' will be created inside. If this is not provided, defaults to the folder 'bin/' under the current MATLAB directory.

   Returns

   - bin_path: Path the the directory containing the WFDB binaries that were downloaded.

download_wfdb_records(*db_name*, *rec_names*, *outdir*, *varargin*)
   Downloads records from the PhysioBank database on PhysioNet.

   Parameters

   - **db_name** – Name of database on physiobank, e.g. `mitdb`.

   - **rec_names** – A string or cell of strings containing a regex pattern to match against the record names from the specified database. Matching will be performed against the entire record name (as if regex is delimited by ^$). Empty array or string will match all records. See examples below.

   - **outdir** – The root directory to download to. A folder with the specified `db_name` will be created within it.

   - **varargin** – Pass in name-value pairs to configure advanced options:

     – base_url: Specify an alternative physiobank URL to download from.

   Returns

   - dl_recs: Names of downloaded records.

   - dl_ann: Names of annotators for downloaded records.

   - dl_files: Paths to all downloaded files.

Examples:

1. Download single record, `mitdb/100` to folder `db/mitdb`:

```
download_wfdb_records('mitdb', '100', 'db');
```

2. Download three specific records from `mitdb`:

```
download_wfdb_records('mitdb', {'100','200','222'}, 'db');
```

3. Download all records starting with '1' (e.g. 100, 101, 122...) from `mitdb`:

```
download_wfdb_records('mitdb', '1\d+', 'db');
```

4. Download records '123', '124' and any record ending with 0 from `mitdb`:

```
download_wfdb_records('mitdb', {'12[3,4]', '\d+0'}, 'db');
```

5. Download all records from `mitdb`:

```
download_wfdb_records('mitdb', [], 'db');
```

ecgrr (*rec_name*, *varargin*)

Calculate an RR-interval time series from PhysioNet ECG data. Detects QRS in a given sigal and calculates the RR intervals.

Parameters

- **rec_name** – Path and name of a wfdb record's files e.g. db/mitdb/100 if the record files (both 100.dat and 100.hea) are in a folder named 'db/mitdb' relative to MATLABs pwd.

- **varargin** – Pass in name-value pairs to configure advanced options:

  - ann_ext: Specify an annotation file extention to use instead of loading the record itself (.dat file). If provided, RR intervals will be loaded from the annotation file instead of from the ECG. Default: empty (don't use annotation).

  - ecg_channel: Number of ecg signal in the record (default [], i.e. auto-detect signal).

  - from: Number of first sample to start detecting from (default 1)

  - to: Number of last sample to detect until (default [], i.e. end of signal)

  - plot: true/false whether to generate a plot. Defaults to true if no output arguments were specified.

Returns

- rri: RR-intervals values in seconds.

- trr: RR-interval times in seconds.

get_signal_channel (*rec_name*, *varargin*)

Find the channel of a signal in the record matching a description. By default, if no description is specified it looks for ECG signal channels.

Parameters

- **rec_name** – Path and name of a wfdb record's files e.g. db/mitdb/100 if the record files (both 100.dat and 100.hea) are in a folder named 'db/mitdb' relative to MATLABs pwd.

- **varargin** – Pass in name-value pairs to configure advanced options:

- sig_regex: A regular expression that should match the desired signal's description in the header file.

- comment_regex: A regular expression that matches the comment format in the header file.

Returns

- chan: Number of the first channel in the signal that matches the description regex, or an empty array if no signals match.

- Fs: Sampling frequency

- N: Number of samples

get_wfdb_tool_path(*tool_name*, *base_search_path*)

Returns the path to a wfdb tool, takes OS into account Looks for the given tool recursively under the current MATLAB directory (or a given directory), and then recursively under the folders in the `$PATH` environment variable. In case the tool is found, the containing directory path will be persisted to speed up the next search.

Parameters

- **tool_name** – A string containing the name of the wfdb tool, e.g. 'gqrs', 'wfdb-config', 'rdsamp' etc. Should not include a file extension.

- **base_search_path** – Optional. An absolute path to use as a base for searching for the tool. If not provided defaults to pwd().

Returns

- tool_path - The path of the wfdb tool, including it's os-specific file extension (e.g. .exe). In case the tool wasn't found, an error will be raised.

gqrs(*rec_name*, *varargin*)

Wrapper for WFDB's 'gqrs' and 'gqpost' tools. Finds the onset of QRS complexes in ECG signals given in PhysioNet format and returns them as a MATLAB vector.

Parameters

- **rec_name** – Path and name of a wfdb record's files e.g. db/mitdb/100 if the record files (both 100.dat and 100.hea) are in a folder named 'db/mitdb' relative to MATLABs pwd.

- **varargin** – Pass in name-value pairs to configure advanced options:

  - ecg_channel: Number of ecg signal in the record (default [], i.e. auto-detect signal).

  - gqconf: Filename or Path to a gqrs config file to use. This allows adapting the algorithm for different signal and/or animal types (default is '', i.e. no config file). Note that if only a filename is provided, 'gqrs' will attempt to find the gqconf file on the MATLAB path.

  - gqpost: Whether to run the 'gqpost' tool to find erroneous detections (default false).

  - from: Number of first sample to start detecting from (default 1)

  - to: Number of last sample to detect until (default [], i.e. end of signal)

Returns

- qrs: Vector of sample numbers where the an onset of a QRS complex was found.

- outliers: Vector of sample numbers which were marked by gqpost as suspected false detections.

Note: If no output variables are given to the function call, the detected ECG signal and QRS complexes will be plotted in a new figure.

---

isrecord(*rec_name*, *data_ext*)

Checks if the given WFDB record name exists locally.

    Parameters

- **rec_name** – Path and name of a wfdb record's files e.g. 'db/mitdb/100'. Can be an absolute or relative path (relative to the MATLAB pwd).

- **data_ext** – Optional. The extension of the data file to look for. Defaults to 'dat' if not specfied.

    Returns

- isrecord: True if the given record path is valid, otherwise false. Valid means that e.g. both the files db/mitdb/100.dat (or another extension as specified in 'data_ext') and db/mitdb/100.hea exist if rec_name was 'db/mitdb/100').

qrs_compare(*rec_name*, *varargin*)

Compare R-peak detection algorithm to annotations. qrs_compare can run a r-peak detector on a given physionet record and compare the detections to an annotation file. The function assumes that the annotation file has the same record name, with a user-configurable file extension. The function supports both wfdb format and matlab's 'mat' format for the annotation files. The comparison of the detected QRS locations to the reference annotations is performed by calculating a joint-accuracy measure (F1), based on the Sensitivity (SE) and Positive-predictivity (PPV) of the test detector's annotations [1]_.

    Parameters

- **rec_name** – Path and name of a wfdb record's files e.g. db/mitdb/100 if the record files (both 100.dat and 100.hea) are in a folder named 'db/mitdb' relative to MATLABs pwd.

- **varargin** – Pass in name-value pairs to configure advanced options:

  - tolerance: Threshold tolerance time, in seconds, for two peak detections to be considered equal.

  - ann_ext: Extension of annotation file.

  - ann_format: Format of annotation file. Can be `wfdb` or `mat`.

  - ecg_channel: Channel number of ecg signal in the record (default [], i.e. auto-detect signal).

  - qrs_detector: Name of qrs detector to use. Can be `rqrs` or `gqrs`.

  - plot: true/false whether to generate a plot. Defaults to true if no output arguments were specified.

    Returns

- sqi: Signal quality indices for the comparison between the detector and annotations.

qrs_compare_set(*set_dir*, *ann_ext*, *varargin*)

Compares reference QRS detections to test detections on a set of wfdb records.

    Parameters

- **set_dir** – directory path containing the wfdb files and annotations

- **ann_ext** – file extension of the annotation files.

- **varargin** – Pass in name-value pairs to configure advanced options:

  - any parameter supported by qrs_compare() can be passed to this function.

  - plot: true/false whether to generate a plot. Defaults to true if no output arguments were specified.

> **Returns**

- sqis: A table containing quality indices for each of the input files.

- stats: A table containing the Mean and Gross values for the quality indices over all files.

rdann (*rec_name*, *ann_ext*, *varargin*)

> Wrapper for WFDB's 'rdann' tool. Reads annotation files in PhysioNet format and returns them as a MATLAB vector.

> **Parameters**

- **rec_name** – Path and name of a wfdb record's files e.g. db/mitdb/100 if the record files (both 100.dat and 100.hea) are in a folder named 'db/mitdb' relative to MATLABs pwd.

- **ann_ext** – Extension of annotation file. E.g. use 'qrs' is the annotation file is mitdb/100.qrs.

- **varargin** – Pass in name-value pairs to configure advanced options:

  - 'ann_types': A double-quoted string of PhysioNet annotation types that should be read, e.g. '"N|"' to read both annotations of type 'N' and type '|'. Default is empty, i.e. return annotations of any type.

  - from: Number of first sample to start detecting from (default 1)

  - to: Number of last sample to detect until (default [], i.e. end of signal)

  - plot: Whether to plot the all the channels and annotations in the file. Useful for debugging.

> **Returns**

- ann: A Nx1 vector with the sample numbers that have annotations.

- ann_types: A Nx1 cell array with annotation types (strings, see PhysioNet documentation).

rdsamp (*rec_name*, *varargin*)

> Wrapper for WFDB's 'rdsamp' tool. Reads channels in PhysioNet data files and returns them in a MATLAB matrix.

> **Parameters**

- **rec_name** – Path and name of a wfdb record's files e.g. db/mitdb/100 if the record files (both 100.dat and 100.hea) are in a folder named 'db/mitdb' relative to MATLABs pwd.

- **varargin** – Pass in name-value pairs to configure advanced options:

  - chan_list: A list of channel numbers (starting from 1) to read from the record, e.g. to read the first three channels use [1, 2, 3]. Default is [], i.e. read all channels from the record.

  - from: Number of first sample to start detecting from (default 1)

  - to: Number of last sample to detect until (default [], i.e. end of signal)

> **Returns**

- t: A vector with the sample times in seconds.

- sig: A matrix where is column is a different channel from the signal.

- Fs: The sampling frequency of the data.

rqrs(*rec_name*, *varargin*)

R-peak detection in ECG signals, based on gqrs and gqpost. rqrs Finds R-peaks in PhysioNet-format ECG records. It uses the gqrs and gqpost programs from the PhysioNet WFDB toolbox, to find the QRS complexes. Then, it searches forward in a small window to find the R-peak.

> **Parameters**
>
> - **rec_name** – Path and name of a wfdb record's files e.g. db/mitdb/100 if the record files (both 100.dat and 100.hea) are in a folder named 'db/mitdb' relative to MATLABs pwd.
>
> - **varargin** – Pass in name-value pairs to configure advanced options:
>
>   - ecg_channel: Number of ecg signal in the record (default [], i.e. auto-detect signal).
>
>   - gqconf: Path to a gqrs config file to use. This allows adapting the algorithm for different signal and/or animal types (default is '', i.e. no config file).
>
>   - gqpost: Whether to run the 'gqpost' tool to find and remove possibly erroneous detections.
>
>   - from: Number of first sample to start detecting from (default 1)
>
>   - to: Number of last sample to detect until (default [], i.e. end of signal)
>
>   - window_size_sec: Size of the forward-search window, in seconds.
>
>   - plot: true/false whether to generate a plot. Defaults to true if no output arguments were specified.
>
> **Returns**
>
> - qrs: Vector of sample numbers where the an onset of a QRS complex was found.
>
> - tm: Time vector (x-axis) of the input signal.
>
> - sig: The input signal values.
>
> - Fs: The input signals sampling frequency.

signal_duration(*N*, *Fs*)

SIGNAL_DURATION Calculates the duration of a signal. Input

> N: Number of samples Fs: Sampling frequency

Output: t_max: total duration in seconds. h: Hours component m: Minutes component s: Seconds component ms: Milliseconds component

wfdb_header(*rec_name*)

Returns metadata about a WFDB record based on it's header file.

> Parameters **rec_name** – Path and name of a wfdb record's files e.g. db/mitdb/100 if the record files (both 100.dat and 100.hea) are in a folder named 'db/mitdb' relative to MATLABs pwd.
>
> **Returns**
>
> A struct with the following fields:
>
> - rec_name: The record name
>
> - Fs: Sampling frequency
>
> - N_samples: Number of samples
>
> - N_channels: Number of channels (different signals) in the record
>
> - channel_info: A cell array of length N_channels with the metadata about each channel

- duration: A struct with the fields h,m,s,ms corresponding to duration fields - hours, miutes, seconds, milliseconds.

- total_seconds: Records total duration in seconds.

Note: If no output arguments are given, prints record and channel info to console.

---

wrann (*rec_name*, *ann_ext*, *ann_idx*, *varargin*)

Wrapper for WFDB's 'wrann' tool. Write annotation files in PhysioNet format given a MATLAB vector.

Parameters

- **rec_name** – Path and name of a wfdb record's files e.g. db/mitdb/100. If a header file for the record doesn't exist one will be created (but fs must be specified in varargin).

- **ann_ext** – Extension of annotation file to write. E.g. use 'qrs' to write the annotation file mitdb/100.qrs.

- **ann_idx** – A column vector of integer type containing sample indices of the annotations.

- **varargin** – Pass in name-value pairs to configure %advanced options:

  - fs: The sampling frequency of the signal which is being annotated. Pass this in if writing annotations for a record which doesn't exist (i.e. a header file should be created).

  - comments: A cell array of strings which will bea written to the header file as comments (one per line). Will only be written when a new header file is craeted by this function.

  - type: Either a single character that will be used as the type for all annotations, or a cell array the same size 'ann_idx' containing a different annotation type per sample.

  - sub: Either a single number (-128 ~ 128) that will be used as the subtype attribute for all annotations, or a column vector the same size as 'ann_idx' containing a different subtype per sample.

  - chan: Either a single number (-128 ~ 128) that will be used as the chan attribute for all annotations, or a column vector the same size as 'ann_idx' containing a different chan per sample.

  - num: Either a single number (-128 ~ 128) that will be used as the num attribute for all annotations, or a column vector the same size as 'ann_idx' containing a different num per sample.

  - aux: Either a single string that will be used as the aux attribute for all annotations, or a string cell array the same size as 'ann_idx' containing a different aux per sample.

Returns A cell array with the paths of files that were created.

2.8 `mhrv.util`

This package provides utility functions used by the toolbox.

fig_print (*fig_handle*, *out_filename*, *varargin*)

FIG_PRINT Prints a figure to file. This function sets various figure properties and then prints it to a specified format. Default format is EPS which can then be imported into a LaTeX document and will maintain it's properties. Note that A4 size is 21.0 x 29.7 cm.

freqband_power (*pxx*, *f_axis*, *f_band*)

FREQBAND_POWER Calculates the power in a frequency band

---

table_stats(*input_table*)
> TABLE_STATS Calcultes statistics of each variable (column) in a table. Calculates Mean, Standard Error and Median values for each variable (column) in a table.
>
> input_table: A table containing numeric data. stats_table: A new table with the same variables as the input, but with a row for each of
>
>> the calculated statistics.

table_transpose(*input_table*)
> TABLE_TRANSPOSE Transposes a table Exchanges between a table's rows and columns, while preserving their names.
>
> input_table: Table to transpose. transposed_table: Transposed table.

vercmp(*ver1*, *ver2*)
> VERCMP Compares two version strings Compares two strings of version numbers, e.g. '1.2.3' and '1.23.4'. Returns an integer similar to strcmp:
>
>> **-1 if ver1 < ver2** 0 if ver1 == ver2 1 if ver1 > ver2

2.9 `mhrv.defaults`

This package provides functionality to get and get toolbox default values.

mhrv_get_all_defaults(*params_struct*)
> Returns all parameter default values of the mhrv toolbox in a map.
>
>> **Parameters params_struct** – Optional. The parameter structure to work on. If not provided, this function will search traverse the globally defined toolbox parameters.
>
> The returned map contains keys that correspond the the unique id's of parameters, e.g `dfa.n_max`. The map's values are structures containing the value of the parameter and metadata fields.

mhrv_get_default(*param_id*, *meta*)
> Returns the default value configured for a parameter. This function attemps to get the value of a parameter as configured by a user config file. If it can't find a user-specified default for the parameter, it throws an error.
>
>> **Parameters**
>>
>>> • **param_id** – Unique id of the parameter This is made up of the keys in the defaults file leading to the parameter (not including the 'value' key), joined by a '.' character (example: 'hrv_freq.methods').
>>>
>>> • **meta** – Optional. A fieldname to return from the parameter structure (instead of the structure itself). Can be value/description/name/units.
>>
>> **Returns**
>>
>>> A structure containing the following fields:
>>>
>>> • value: The user-configured parameter value, if exists, otherwise returns the 'default_value'.
>>>
>>> • name: The user-friendly/display name of the parameter.
>>>
>>> • description: A description about the parameter.
>>>
>>> • units: The units the parameter is specified in.

> **Note:** If a value for 'meta' was specified, only the corresponding field will be returned.

---

mhrv_load_defaults (*varargin*)

Loads an mhrv defaults file, setting it's values as default for all toolbox functions. Optionally, all current parameter defaults will be cleared.

Usage:

```
mhrv_load_defaults [--clear]
mhrv_load_defaults [--clear] <defaults_filename>
mhrv_load_defaults(defaults_filename, 'param1', value1, 'param2', value2, ...)
```

This function loads the parameter values from the default mhrv parameters file and sets them as the default value for the various toolbox functions.

The second usage form loads the parameter values from an arbitrary mhrv parameters file (.yml). The given file can be a name of a file on the matlab path, or, if it's not found there, it will be interpreted as a path (absolute or relative to pwd).

The third usage form also allows overriding or adding specific parameters with custom values given to the function. In this form, the filename is optional; the function will also accept just key-value pairs.

Note: This function clears all current default parameters if the '–clear' option is given. Otherwise, it merges the loaded values with the previously existing parameter defaults.

mhrv_parameter (*value*, *description*, *name*, *units*)

Creates a parameter structure for use with the mhrv toolbox.

> **Parameters**
>
> - **value** – The parameter's value. Can be any matlab object.
>
> - **description** – Informative description of the parameter.
>
> - **name** – User friendly/display name of the parameter.
>
> - **units** – Parameters units.

Note: All inputs are optional and default to an empty string if not provided.

> **Returns** An object representing the parameter. Can be added to the defaults with the mhrv_set_default or mhrv_load_defaults functions.

mhrv_save_defaults (*output_filename*)

Save the current default values of all parameters defined in the toolbox to a file.

Usage:

```
mhrv_save_defaults <output_filename>
```

This function saves the current default values of all parameters in the toolbox to a specified output file. The file will be in YAML format. If the output_filename parameters doesn't specify a `.yml` extension, it will be added.

mhrv_set_default (*varargin*)

Sets (overrides) current default parameter value(s) with new values.

Usage:

```
mhrv_set_default('param1', value1, 'param2', value2, ...)
```

This function allows overriding specific parameters with custom values given to the function. The input should consist of key-value pairs, where the keys use the '.' character to separate heirarchy levels (e.g. 'rqrs.gqconf').

2.10 `mhrv.plots`

This package provides plotting capabilities.

`default_axes_tag` (*func_name*)
> DEFAULT_AXES_TAG Default Tag for axes. func_name: Name of the plotting function.

`plot_dfa_fn` (*ax*, *plot_data*, *varargin*)
> PLOT_DFA_FN Plots the DFA F(n) function. ax: axes handle to plot to. plot_data: struct returned from dfa.

`plot_ecgrr` (*ax*, *plot_data*, *varargin*)
> PLOT_ECGRR Plots filtered RR intervals from ecgrr. ax: axes handle to plot to. plot_data: struct returned from ecgrr.

`plot_filtrr` (*ax*, *plot_data*, *varargin*)
> PLOT_FILTRR Plots filtered RR intervals from filtrr. ax: axes handle to plot to. plot_data: struct returned from filtrr.

`plot_hrv_freq_beta` (*ax*, *plot_data*, *varargin*)
> PLOT_HRV_NL_BETA Plots the slope of the log-log spectrum in the VLF range. ax: axes handle to plot to. plot_data: struct returned from hrv_freq.

`plot_hrv_freq_spectrum` (*ax*, *plot_data*, *varargin*)
> PLOT_HRV_FREQ_SPECTRUM Plots the spectrums generated by hrv_freq. ax: axes handle to plot to. plot_data: struct returned from hrv_freq.

`plot_hrv_time_hist` (*ax*, *plot_data*, *varargin*)
> PLOT_HRV_TIME_HIST Plots the hrv_time intervals histogram. ax: axes handle to plot to. plot_data: struct returned from hrv_time.

`plot_mse` (*ax*, *plot_data*, *varargin*)
> PLOT_MSE Plots the MSE function. ax: axes handle to plot to. plot_data: struct returned from dfa.

`plot_poincare_ellipse` (*ax*, *plot_data*, *varargin*)
> PLOT_POINCARE_ELLIPSE Plots a poincare plot and an ellipse fit. ax: axes handle to plot to. plot_data: struct returned from poincare.

INDICES AND TABLES

- genindex
- modindex
- search