

PhysioData Toolbox

Version 0.5

User Guide



**Universiteit
Leiden**
Sociale Wetenschappen

Leiden University
Research Support Department
Faculty of Social and Behavioural Sciences

Autumn 2019.

Elío E. Sjak-Shie,
Research Engineer & Scientific Programmer

Faculty of Social and Behavioural Sciences, Leiden University.
Wassenaarseweg 52, 2333 AK Leiden, The Netherlands.

Contact: E.E.Sjak-Shie@fsw.leidenuniv.nl

Contributor:
Iris M. Spruit (I.M.Spruit@fsw.leidenuniv.nl)



Contents

1 Introduction	5
1.1 Toolbox Installation	5
2 File Converter	6
2.1 Supported Raw File Formats.....	7
2.2 Custom Converters (Beta).....	10
3 Session Manager	11
3.1 PhysioData Files Panel.....	12
3.2 PhysioAnalyzer Configurator Panel.....	13
3.3 Data Export Panel.....	15
4 Data Viewers	17
4.1 Raw Data Viewer.....	18
4.2 PhysioAnalyzer Viewer	20
5 Generic Signal Analyzer	23
5.1 Settings	23
5.2 Metrics.....	25
5.3 Resampled Signals.....	25
5.4 Data Correction	25
6 ECG Signal Analyzer	27
6.1 Settings	27
6.2 Metrics.....	29
6.3 Resampled Signals.....	30
6.4 Data Correction	30
7 IBI Sequence Analyzer	33
7.1 Settings	33
7.2 Metrics.....	35
7.3 Resampled Signals.....	35
7.4 Data Correction	35
8 HRV Analyzer	36
8.1 Analysis Types.....	36
8.2 Settings	38
8.3 Metrics.....	40
8.4 Resampled Signals.....	42
8.5 User Interaction and Data Correction.....	42
8.6 Data Pipeline	42
9 ICG Ensemble Analyzer (Manual Scoring)	44
9.1 Settings	45

9.2 Metrics 46

9.3 Resampled Signals..... 46

9.4 User Interaction and Data Correction 46

10 Blood Pressure Analyzer..... 48

10.1 Settings 48

10.2 Metrics..... 49

10.3 Resampled Signals..... 50

10.4 Data Correction 50

11 Respiration Analyzer (Beta Version)..... 52

11.1 Settings 52

11.2 Metrics..... 54

11.3 Resampled Signals..... 55

11.4 User Interaction and Data Correction..... 55

11.5 Data Pipeline 55

12 EMG Signal Analyzer 57

12.1 Settings 57

12.2 Metrics..... 58

12.3 Resampled Signals..... 59

12.4 Data Correction 59

13 Skin Conductance Analyzer..... 60

13.1 Settings 60

13.2 Metrics..... 61

13.3 Resampled Signals..... 61

13.4 Data Correction 62

14 Pupil Diameter Analyzer (Beta Version) 63

14.1 Preprocessing Pipeline and Settings..... 64

14.2 Metrics..... 72

14.3 Resampled Signals..... 72

15 Area of Interest Hit Analyzer..... 73

15.1 Settings 73

15.2 Data Correction 73

15.3 Metrics..... 73

15.4 Resampled Signals..... 74

16 Epochs and Events 75

16.1 Defining Epochs..... 77

16.2 The Epoch Builder 78

16.3 Defining Events..... 79

17 Tips and Advice	80
18 PhysioData Format	82
18.1 PhysioData File Format	82
19 Toolbox User Analytics	88

1 Introduction

The PhysioData Toolbox is an easy-to-use graphical environment for visualizing, segmenting and processing physiological data. It is maintained by the Social and Behavioural Sciences Faculty of the Leiden University and is made freely available to researchers as a compiled MATLAB-based application.

The Toolbox and all its documentation can be downloaded from its website:

<https://PhysioDataToolbox.leidenuniv.nl>

Questions, comments and suggestions can be sent to the email below. Please also consider sending in your articles if you used this Toolbox:

PhysioDataToolbox@fsw.leidenuniv.nl

If you use the Toolbox, please cite it using the reference below, or refer to the ‘Tips and Advice’ section for further author information:

Sjak-Shie, E. E. (2019). PhysioData Toolbox (Version 0.5) [Computer software]. Retrieved from <https://PhysioDataToolbox.leidenuniv.nl>

In addition to this User Guide, also included in the toolbox package is a short **walkthrough**. Use that document to quickly get acquainted with the Toolbox and its features.

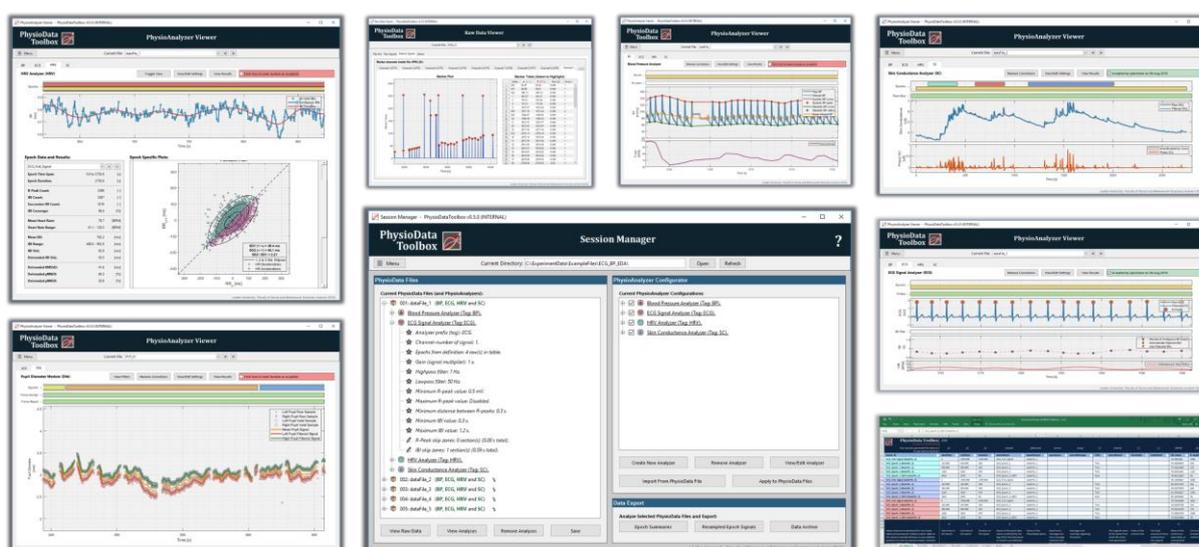


Figure 1: PhysioData toolbox impression.

1.1 Toolbox Installation

The toolbox is delivered as a zip file containing the executables and their ancillary files. To run, **unzip the zip file** and run ‘**PhysioDataToolbox.exe**’ in the Toolbox folder. When launched for the first time, it may take **a few minutes** before the toolbox is completely loaded and ready for use, even after the splash screen disappears.

To use the toolbox, the MATLAB R2018b (9.5) runtime must be installed; this can be downloaded free of charge but requires administrator privileges to install. If you do not have sufficient privileges, ask your IT department to install it for you.

See the website (mentioned above) for additional installation and troubleshooting tips.

2 File Converter

The PhysioData toolbox is designed to only analyze standardized **PhysioData files**, which are specially formatted MATLAB files with the physioData extension. To facilitate the batch conversion of raw physiological data to the PhysioData format, the toolbox is bundled with a separate ‘**File Converter**’ application. The File Converter can be launched via the ‘Open File Converter’ option in the Session Manager’s menu, or by running the FileConverter.exe file directly.

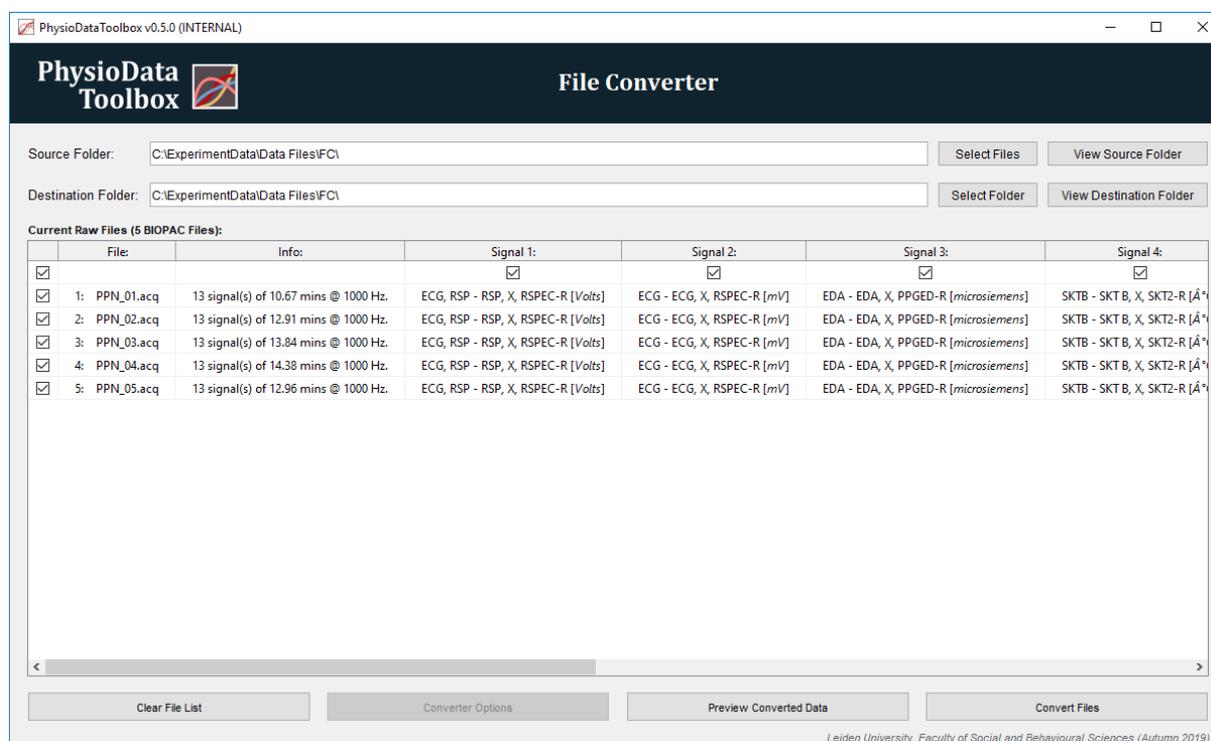


Figure 2: The File Converter interface, showing 5 imported BIOPAC files (PPN_01.acq – PPN_05.acq) with 13 channels each.

User Interface

Files can be imported into the File Converter by clicking the ‘**Select Files**’ button and selecting the correct file type and import options. Once the raw files have been imported, information about their contents is presented in the File Converter’s main table, labeled ‘**Current Raw Files**’, with each row below the top row representing a single file.

The first column in the Current Raw Files table contains checkboxes used for enabling and disabling individual files for conversion. The ‘**File**’ and ‘**Info**’ columns to the right of that show the file names and a summary of the file contents, respectively.

In the case that the raw files contain signals, the subsequent columns will show their names and units. Similarly, if a file contains pupil diameter data, columns representing the left and right pupil size data will appear. The checkboxes in the top row can be used to enable and disable individual channels for conversion. **It is strongly recommended to not convert data that are not required for analysis.**

Below the table, the ‘**Clear File List**’ button resets the converter to its default empty state, and the ‘**Converter Options**’ button launches a menu showing custom conversion options for the selected file type. This button is disabled if the file type does not have any custom options, as is the case for BIOPAC files.

Data Previewer

Clicking the **'Preview Converted Data'** button converts the first enabled raw file and displays the converted data in a new window called **'Data Previewer'**, which is similar to the Toolbox's **Raw Data Viewer**. For more information about viewing and navigating through raw data, see the 'Data Viewers' chapter.

The Data Previewer only converts the signals currently enabled for conversion in the Raw Files table (i.e., data in columns where the first row has a checked checkbox). Additionally, if available, the Data Previewer uses the currently set custom converter options when generating data for previewing. These options can be viewed by clicking the custom options button in the main File Converter window (below the table), or the same button in the Data Previewer (to the left of the current file menu). Note that previewing does not actually create a PhysioData file.

File Conversion

The **'Convert Files'** button in the main File Converter window starts the file conversion process, which, depending on the file type, may take a few minutes to complete. The resulting PhysioData files are always given the same name as their raw file counterparts, with the exception of the extension. Before conversion, **make sure that all files are similar and compatible with the selected conversion options**.

If the File Converter detects imminent filename collisions, i.e. that similarly named PhysioData files already exist in the destination folder, the following options become available:

- **Skip files that already exist (default).**
 - If this option is left selected, the raw files that would produce a PhysioData files that already exist are not converted.
- **Only overwrite the raw data inside the existing PhysioData files.**
 - This option causes the File Converter to only rewrite the raw data inside the PhysioData file, if it already exists, leaving any modules settings, states and corrections intact.
- **Completely overwrite existing PhysioData files.**
 - When selected, any preexisting PhysioData file with a conflicting name will be completely overwritten.

2.1 Supported Raw File Formats

The current version of the File Converter features built-in support for the following raw file formats:

- **BIOPAC**
- **VU-AMS**
- **Philips Achieva MRI 'PhysLog' files**
- **Biosemi**
- **E-Prime Extension for Tobii (EET)**
- **EyeLink**

Other raw file types can be converted to the PhysioData format by either importing a 'Custom Converter' into the File Converter application, or by running a script inside MATLAB that handles the conversion. For information about the former approach, see the 'Custom Converters' section later in this chapter. For details about the required PhysioData file specification, see the 'PhysioData Format' chapter.

BIOPAC

The built-in BIOPAC converter supports AcqKnowledge (v3.9 – v5.0.2) data saved as .acq files or exported as .mat files. Due to a limitation in the BIOPAC File API, only AcqKnowledge .acq files that comply with the following requirements can be converted: **all channels must have the same sample rate**

and all channels must have the same length. However, AcqKnowledge data that don't conform to this requirement can still be exported as .mat files and converted.

When using digital markers, and depending on the experimental design, the BIOPAC AcqKnowledge data may contain 8 digital channels that are only used to calculate a separate 8-bit decimal marker channel. If this is the case, these 8 single-bit channels, usually labeled 'Digital Input', should be omitted from conversion as they are not necessary for data analysis.

The BIOPAC converter does not feature any custom options.

VU-AMS

The VU-AMS converter supports converting raw 5FS files recorded using the VU-AMS system. All sub-sampled signals are up-sampled to the master sampling rate through linear interpolation and nearest neighbor extrapolation.

If a channel labeled Z0 is present, a -dZdt channel is generated using a differentiating 256 order FIR filter with a high-pass cutoff frequency of 1 Hz, and a low-pass cutoff frequency of 10 Hz.

The VU-AMS converter does not feature any custom options.

LIBC Achieva MRI

PhysLog files generated by the LIBC's Philips Achieva 3T MRI scanner can be converted to the PhysioData format using the LIBC Achieva MRI converter. Note that this converter is designed for use with data from the Leiden University scanner, and may not work as intended on data from other scanners, even those of the same make and model.

Biosemi

The Biosemi converter can be used to extract physiological signals from .bdf files and save them in the PhysioData format. Since the PhysioData Toolbox cannot analyze EEG data, only other physiological signals present in the file, such as ECG, EMG, skin conductance, etc., should be extracted.

For Biosemi files, the following Biosemi Options can be set (see Figure 3):

- **Calculation Channels:** This field can be used to generate new data from signals in the file. This can for example be useful for generating an ECG signal. The Biosemi Options window provides a calculation example.
- **Markers and Button Presses:** The user can choose to convert Markers and/or Button Presses to markers and or labels in the PhysioData file, respectively. By default, both Markers and Button Presses are converted. Note, however, that button presses are not debounced.

EET Output

The E-Prime Extensions for Tobii (EET) output converter supports EET files with the .gazedata (EET 2.x – 3.1) and the .txt (EET 3.2) extensions.

This converter features the following custom options (see Figure 4):

- **Eye-tracking Event Generation:** In this field, one or more column names of the EET files can be specified. These columns will then be used to generate eye-tracking events by finding the start and end of each contiguous section of values, or a combination of values. The default value for this field is CurrentObject, which is automatically created in EET 3.2 files and holds the E-Prime object that was currently running.
- **AOI Analysis:** In this field, one column name of the EET files can be specified. This column should hold the current area of interest (AOI) hit data. It thus holds the AOI name that is currently looked at (if any). In EET 3.2 files, the ComponentName column can be used for AOI

Analysis. The ComponentName column is automatically created in EET 3.2 files. This column holds the (sub)object or slide state that is currently looked at.

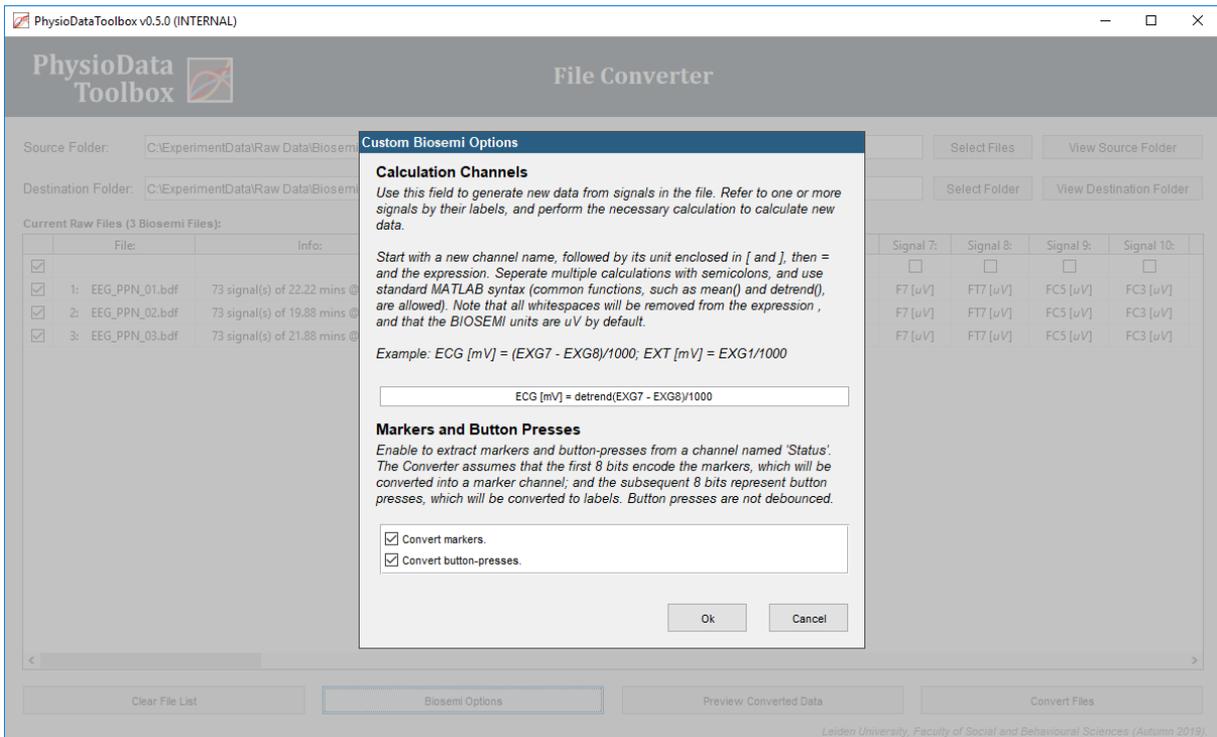


Figure 3: Biosemi Options.

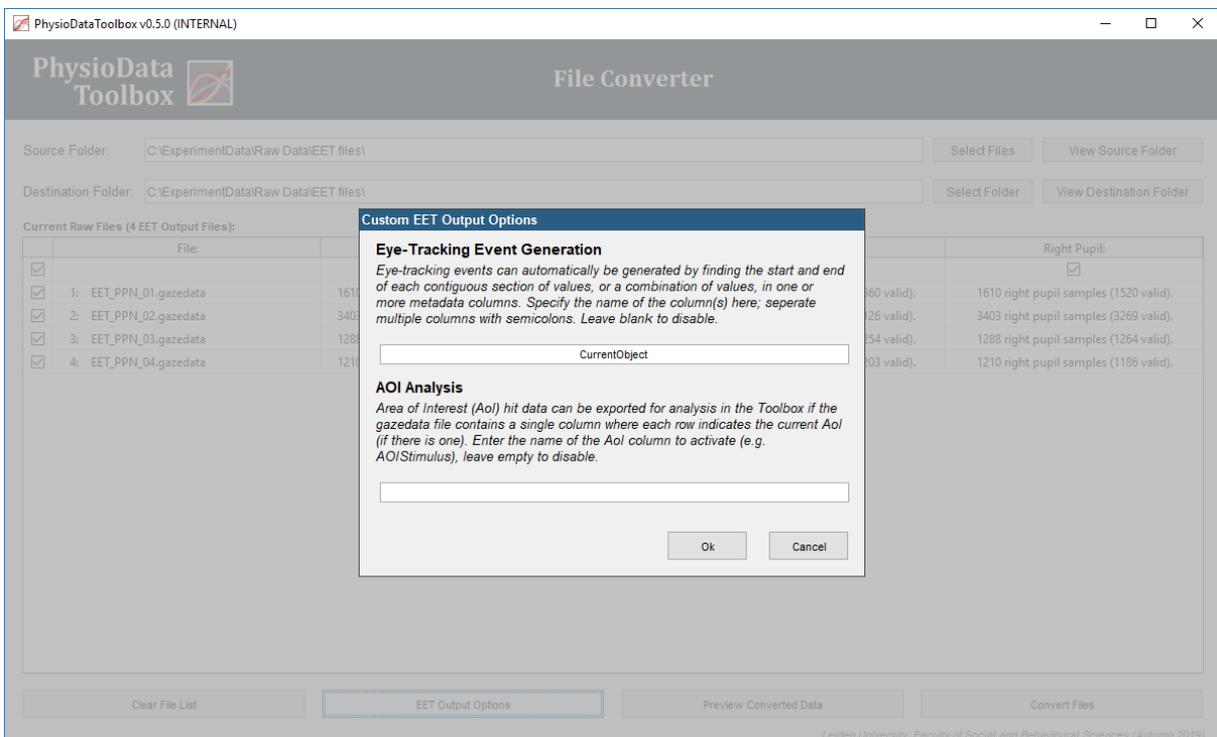


Figure 4: EET Output Options.

EyeLink

The EyeLink converter can be used to extract raw pupil-size data from SR research's EyeLink .edf files, and features the following custom options (see Figure 5):

- **Remove EyeLink System Events:** The File Converter converts all messages available in the .edf file to eye-tracking events, except the events that match the regular expression specified in this field. Leaving the field blank converts all messages. Many EyeLink system-events are not actually used by the PhysioData Toolbox and can therefore be omitted from conversion. The default value removes these system-events.
- **Eyelink via E-Prime Options:** In the case that the onset delay correction method outlined in the simple.es2 E-Prime example task was used, the converter can automatically extract the offset from the message and use it to correct its timestamp. Additionally, the offset is removed from the message string, allowing it to be referenced in a generalized manner in the Toolbox (see the 'Epochs and Events' section).

Note, the data timestamps are zeroed relative to the first event, which means that the first pupil size sample may not appear until a few seconds or minutes into the recording. All events are considered, even those subsequently removed.

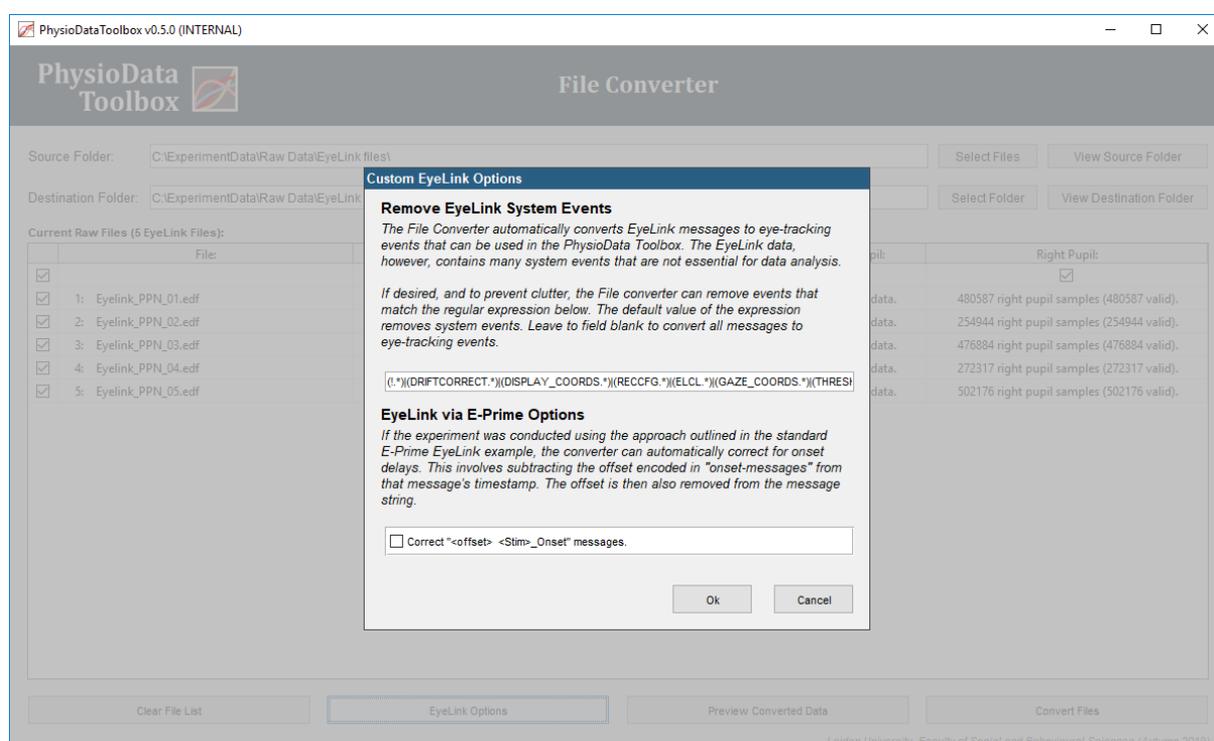


Figure 5: EyeLink Options.

2.2 Custom Converters (Beta)

When other raw data formats need to be converted, or a customized conversion pipeline is required, a 'Custom Converter' can be imported into the File Converter. These converters are essentially MATLAB functions that adhere to a strict convention, which is described in the DESC.m file inside the supplementary code folder (\supplementaryCode\) included in the Toolbox zip file. See also the included `convert_CUSTOM_GAZEDATA.m` and `convert_CUSTOM_GAZEDATA.m` examples in the same folder. Additionally, the code for the built-in converters can be found in the \sharedCode\converters\ folder inside the supplementary code folder

This functionality is still experimental and requires a valid installation of MATLAB (2017b or newer).

3 Session Manager

The ‘**Session Manager**’ is the main window of the PhysioData Toolbox, and is used to manage PhysioData files and the analyses performed thereon.

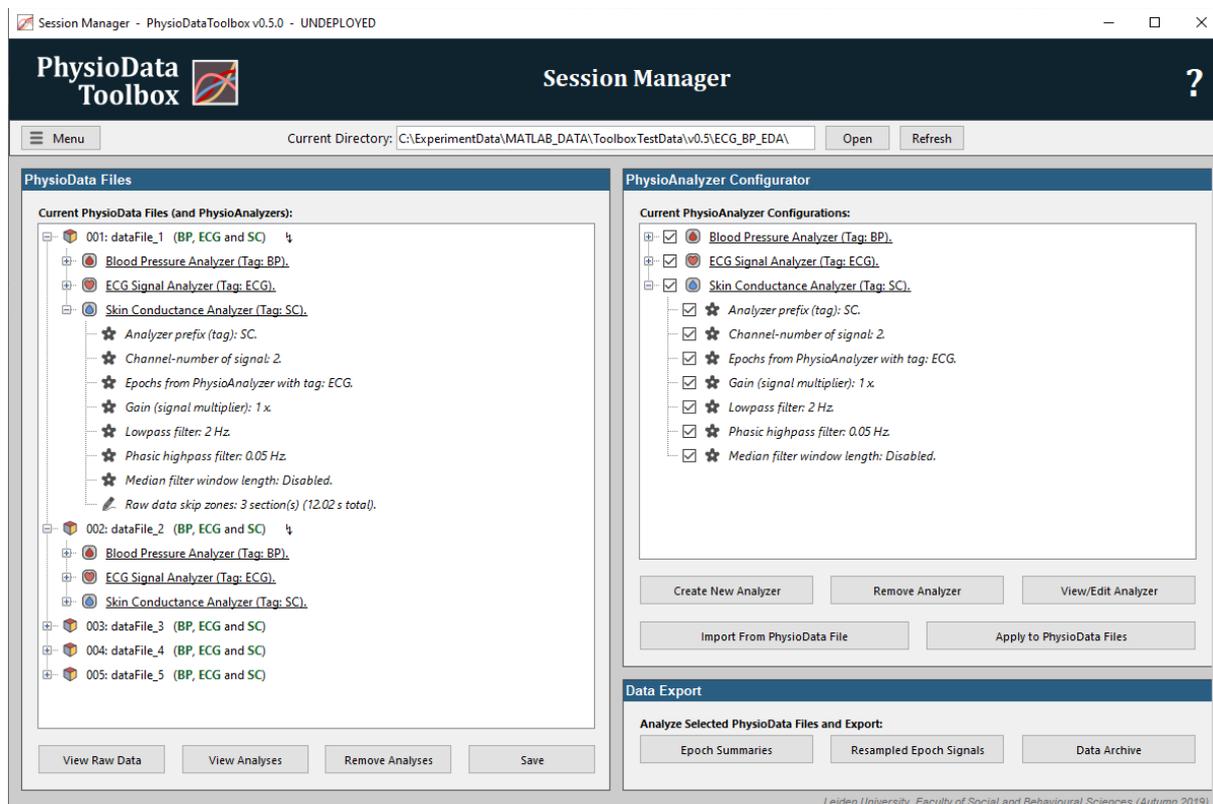


Figure 6: The Session Manager, which is the main toolbox window. It manages the PhysioData files, the PhysioAnalyzers and the data export.

From the Session Manager, users can: import files; create ‘PhysioAnalyzer’ modules and propagate them to the files; launch viewers to review and interact with the data in the files; batch analyze the data inside the files; and save the current session. These concepts and terminologies, which are paramount to understanding the PhysioData Toolbox and its user interaction mechanics, are laid out below:

- **PhysioData files:** specifically formatted MATLAB files that contain **raw data**, and all settings and user-corrections necessary to analyze those data. By design, the toolbox saves all information necessary to analyze an individual file inside that same file, which obviates the need for separate ‘analysis definition’ files. In the Session Manager, the raw files are visualized in the file tree in the ‘PhysioData Files’ panel, see Figure 6.
- **Raw Data:** data that have been converted to the PhysioData specification but has not been altered by the Toolbox. The current version of the toolbox supports two types of raw data: **signals** and **eye-tracking datasets**. Signals are a collection of uniformly sampled time-series (e.g. a concurrent recording of ECG and EMG data) and may contain ‘**labels**’, which are named moments inside the recording (e.g. ‘Start Recording’ or ‘End Trial 1’). A signal inside a collection thereof may be referred to as ‘**channel**’ and referenced by means of a ‘**channel number**’ (e.g. the ECG and EMG data are in channels 1 and 2, respectively). A special type of signal is a ‘**Marker**’ channel, which is a digital signal with activity that reflects sections and/or events inside the recording. An eye-tracking dataset may contain a variety of pupil size and gaze information, as well as named moments (‘**eye-tracking events**’) and named segments (‘**eye-tracking sections**’). The raw data can be inspected using the ‘**Raw Data Viewer**’, which is launched by clicking the ‘**View Raw Data**’ button in the Session Manager.

- **Epochs:** useful time-segments inside the recording. Analyzing continuous recordings usually involves segmenting the signals into sections of interest, aka epochs, and analyzing them independently. These epochs are generally defined relative to **Markers, Labels, or Events**. A fundamental aim of the toolbox is to automate the epoch-based batch-analyses of signals by providing a flexible non-ambiguous rule-based method for defining how signals must be sectioned. A collection of user-defined rules by which a signal is to be segmented is referred to as an **Epoch Definition** table. These user-created definitions are used by the toolbox to automatically identify, isolate, name, and analyze special sections of recorded data. See the 'Epochs and Events' section for further information. Additionally, PhysioData files can contain **'File Epochs'**, which are pre-generated custom epochs. This allows users to create and analyze custom epochs that cannot be generated using the an epoch definition table
- **PhysioAnalyzer modules:** components that define the analysis pipeline, user-interaction and visualization scheme for a specific type of data. For example, the ECG Signal Analyzer defines how a raw ECG signal is processed and which settings are used in this procedure, as well as how the data is visualized and which user-corrections are possible (e.g. rejecting faulty sections). These modules can be created, modified and applied to the desired PhysioData files using the **PhysioAnalyzer Configurator** panel in the Session Manager. Note that the signals generated by the PhysioAnalyzers (e.g. the Heartrate and Phasic Skin Conductance signals) are not saved to disk, only the necessary information to generate these signals from the raw data is. This information consists of the **Settings** (channel number, gain, tag, filter parameters, etc.) and the **State** (user corrections) of the PhysioAnalyzer. Using the Raw Data, the Settings, and the State, the toolbox can dynamically generate the PhysioAnalyzer data when it is requested.
- **Saving:** committing the settings and state to the PhysioData files so that they can be reloaded in the future. When users import files, a new **'Session'** is started, during which all changes to the files are kept in memory and persist only for the current session. It is not until the user saves the data that the changes are stored to disk. Closing the toolbox or starting a new session without saving the current session will destroy all modules that were created during that session, including their settings and states. A session can be saved by clicking the 'Save' button in the Session Manager. It is strongly advised to save often if the changes made are to be kept. Once saved, a session can be resumed by once again pointing the toolbox to the folder containing the PhysioData files.

In addition to the 'PhysioData Files', 'PhysioAnalyzer Configurator' 'Data Export' panels described later in this chapter, the Session Manager features a directory bar showing the **'Current Directory'**, which is the folder from which the current PhysioData files have been imported. This directory is set using the **'Open'** button, and can be rescanned by clicking the **'Refresh'** button.

Clicking the **'Menu'** button reveals options, including a shortcut to the File Converter application and to the PhysioData Toolbox website. In addition, the question mark in the top right corner of the Session Manager can be clicked to view applications and version information.

3.1 PhysioData Files Panel

The 'PhysioData Files' panel is used for managing the imported PhysioData files.

The imported PhysioData files and their included PhysioAnalyzers are displayed in the file tree. Green and red tag colors indicate that those PhysioAnalyzers have or have not been marked as 'accepted' by the user, respectively. A lightning bolt indicates that the file's raw data is loaded into memory.

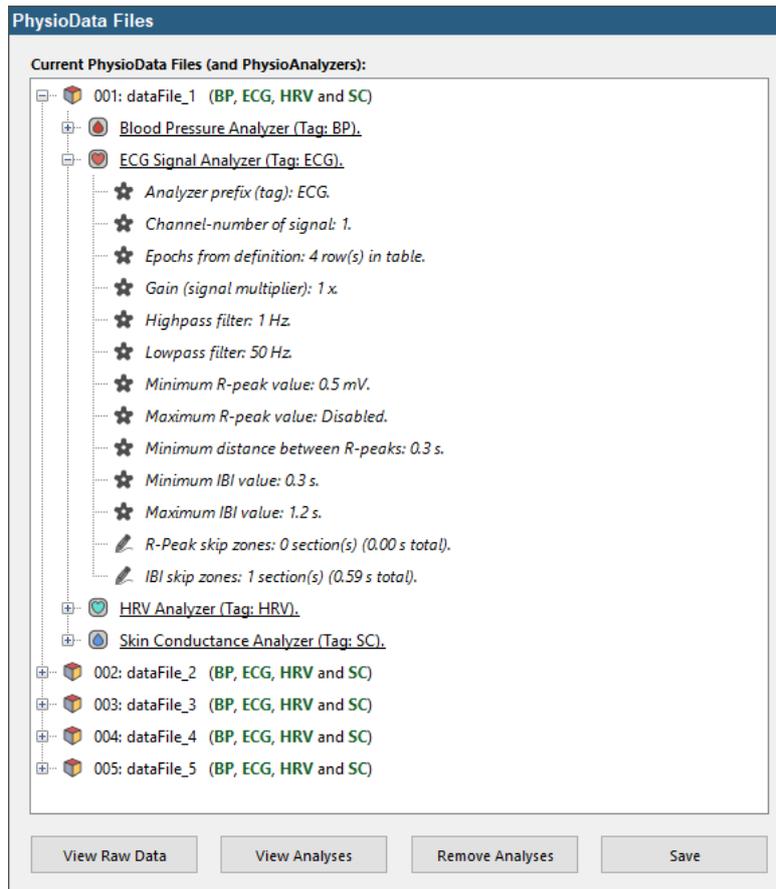


Figure 7: Current PhysioData Files tree, showing the imported PhysioData files and their PhysioAnalyzers. File 001 is expanded, showing it has a Blood Pressure Analyzer tagged BP, an ECG analyzer tagged ECG, an HRV analyzer tagged HRV, and a Skin Conductance Analyzer tagged SC. Additionally, the ECG analyzer is expanded showing its settings and state.

In the file tree, each PhysioData file can be expanded to reveal the PhysioAnalyzers it contains, which in turn can be expanded to reveal its settings and state, as shown in Figure 7.

Once a PhysioData file has been in the tree, the ‘**View Raw Data**’ button opens the Raw Data Viewer, which visualizes all raw data inside the selected file. Similarly, the ‘**View Analyses**’ button opens the PhysioAnalyzer Viewer, which shows the PhysioAnalyzer modules inside each file. The ‘**Remove Analyses**’ button launches a selection window allowing the user to choose which PhysioAnalyzers to remove from the selected files.

The current session can be saved using the ‘**Save**’ button, which stores the PhysioAnalyzers, including all user corrections, inside their corresponding PhysioData files. It is important to note that the toolbox does not automatically save anything unless explicitly instructed to by the user.

3.2 PhysioAnalyzer Configurator Panel

The ‘PhysioAnalyzer Configurator’ panel is used for defining or modifying the PhysioAnalyzers, and then propagating them to selected PhysioData files.

The ‘Current PhysioAnalyzer Configurations’ tree shows the currently defined PhysioAnalyzers and their settings. These PhysioAnalyzers are not linked to any files, and do not contain any state (user corrections). It is not until the user applies the PhysioAnalyzer to the PhysioData files using the ‘**Apply to PhysioData Files**’ button that the PhysioAnalyzer settings are propagated to the selected files, and linked to data.

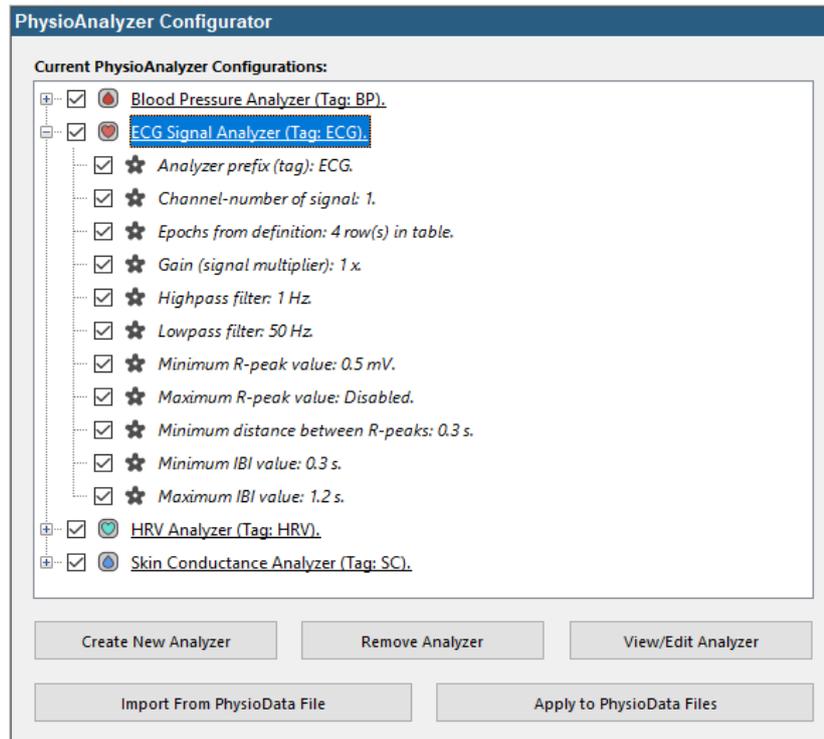


Figure 8: The PhysioAnalyzer Configurator panel in the Session Manager. This panel features a tree showing the defined PhysioAnalyzers and their settings, and buttons for creating and applying these PhysioAnalyzers to the files. The checkboxes can be used to select only a subset of the settings for applications.

New PhysioAnalyzers can be created using the ‘**Create New Analyzer**’ button, which opens a list of available PhysioAnalyzers. Once the desired PhysioAnalyzer is selected, a window opens allowing the user to modify its settings. This window can be reopened at any time by selecting the PhysioAnalyzer in the tree and clicking ‘**View/Edit Analyzer**’. Analyzers can be removed by selecting them and clicking the ‘**Remove Analyzer**’ button.

PhysioAnalyzers can be imported from a loaded PhysioData file by first selecting the desired file in the PhysioAnalyzer file tree, then clicking the ‘**Import From PhysioData File**’ button. This copies the settings of all the PhysioAnalyzers inside the selected PhysioData file to the PhysioAnalyzer Configurator.

The PhysioAnalyzer tree allows the user to check and uncheck individual PhysioAnalyzers and any of its settings. If a PhysioData file already contains a PhysioAnalyzer of the same type and with the same tag, only the checked setting(s) within that PhysioAnalyzer will be updated. This allows the user to change only certain settings across selected files, leaving other customized settings intact.

Once the ‘**Apply to PhysioData Files**’ button is clicked, the user is presented with a summary of the settings that are to be pushed to the files, and some additional options, see Figure 9. Checking the ‘**Mark overwritten PhysioAnalyzers as ‘Unaccepted’**’ box will reset the accepted status of each PhysioAnalyzer that new settings were pushed to, while the ‘**Clear State**’ option clears the state (all user corrections and scored items) from the PhysioAnalyzer.

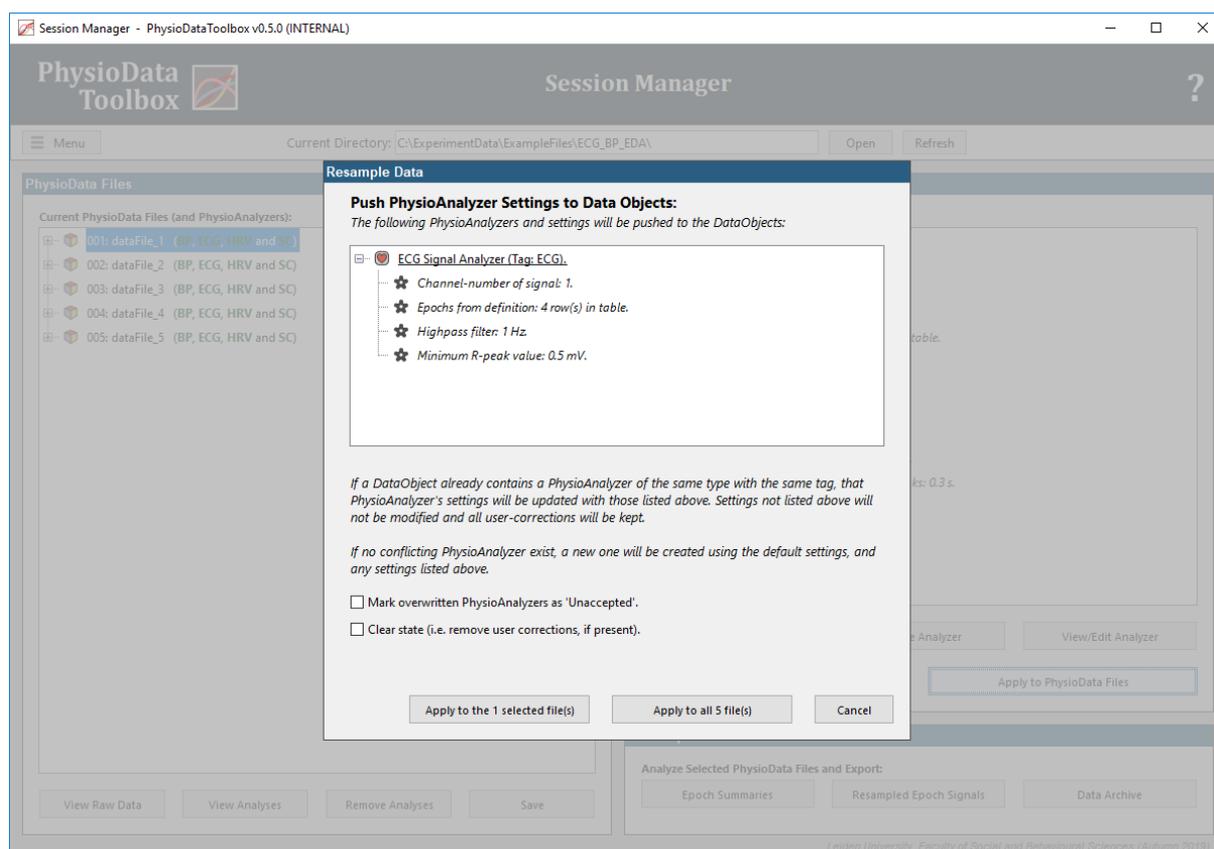


Figure 9: The summary and options window that pops up when applying PhysioAnalyzers to PhysioData files. Note that in this case, only the following settings will be pushed to the files: channel number, epochs definitions, high-pass filter settings, and minimum R-peak value. All other settings and all PhysioAnalyzers with different tags will be left unaffected.

Note that the ‘**Remove Analyzer**’ and ‘**View/Edit Analyzer**’ buttons act on the selected PhysioAnalyzers (highlighted in blue), while the ‘**Apply to PhysioData Files**’ button acts on the checked PhysioAnalyzers and their checked settings, regardless of selection.

3.3 Data Export Panel

The ‘Data Export’ panel contains options for batch analyzing and exporting the raw and processed data for further analysis.



Figure 10: The Data Export panel.

Generating Epoch Summaries

The PhysioData Toolbox is primarily meant for performing epoch-based descriptive statistical analysis; i.e., for each PhysioAnalyzer, the relevant metrics are calculated per epoch. When the ‘**Epoch Summaries**’ button is clicked, all epochs inside all PhysioAnalyzers inside the selected PhysioData files are analyzed and the results are exported as Excel file, MATLAB file, or tsv (tab separated) files.

Epoch_ID	endTime	duration	epochName	epochSource	epochError	epochMessage	TYPE	HR_mean	R_peakCount	IBI_mean
ECG_Epoch_1 (dataFile_1)	419.998	300	ECG_Epoch_1	dataFile_1			Trial	91.64296925	458	0.6554
ECG_Epoch_2 (dataFile_1)	899.998	300	ECG_Epoch_2	dataFile_1			Trial	77.52911444	378	0.7737
ECG_Epoch_1 (dataFile_2)	419.998	300	ECG_Epoch_1	dataFile_2			Trial	98.15571107	491	0.6111
ECG_Epoch_2 (dataFile_2)	899.998	300	ECG_Epoch_2	dataFile_2			Trial	89.25873026	446	0.6721
ECG_Epoch_1 (dataFile_3)	419.998	300	ECG_Epoch_1	dataFile_3			Trial	76.55287798	382	0.7841
ECG_Epoch_2 (dataFile_3)	899.998	300	ECG_Epoch_2	dataFile_3			Trial	75.03102739	375	0.7999
ECG_Epoch_1 (dataFile_4)	419.998	300	ECG_Epoch_1	dataFile_4			Trial	92.92438012	465	0.6451
ECG_Epoch_2 (dataFile_4)	899.998	300	ECG_Epoch_2	dataFile_4			Trial	90.40773543	452	0.6631
ECG_Epoch_1 (dataFile_5)	419.998	300	ECG_Epoch_1	dataFile_5			Trial	89.67406963	447	0.6669
ECG_Epoch_2 (dataFile_5)	899.998	300	ECG_Epoch_2	dataFile_5			Trial	88.18381232	440	0.6801

Please read and understand the User Guide before processing the Toolbox output. Refer to the relevant scientific literature in your field for guidance on how to properly analyze, interpret and report these data.

Elio Sjak-Shie
Faculty of Social and Behavioral Sciences,
Leiden University, The Netherlands.
Autumn 2019.

↑ End time of the epoch.
↑ Duration of the epoch.
↑ Name of the epoch (the name is prefixed by the tag of the PhysioAnalyzer in which it was defined).
↑ Name of the PhysioData object.
↑ Epoch error messages (an error message indicates that the epoch could not be constructed).
↑ Messages and warnings regarding the Epoch.
↑ Mean of the continuous Heart Rate, as interpolated from the accepted IBI data points.
↑ Count of accepted R-peaks inside current epoch.
↑ Arithmetic of the discrete values defined IBI(n) R(n-1) times using: R(n).

Figure 11: Excel data export format. The first column shows the unique Epoch identifier, and its cells' background colors are determined by the file; i.e., cells with the same 'epochSource' have the same color.

Generating Resampled Epoch Signals

Instead of generating summaries of data per epoch, it is also possible to export resampled data of each epoch. When the 'Resampled Epoch Signals' button is clicked, the user is asked which module (PhysioAnalyzer) to resample and the desired sampling rate (ranging from 50 Hz to 1 Hz) can be selected. The data of all epochs of the selected PhysioAnalyzer and of the selected PhysioData files is then resampled and saved as Excel file, MATLAB file, or TSV (tab separated) files.

Generating a Data Archive

Next to epoch summaries and resampled epoch signals, the toolbox can also export all PhysioAnalyzer data as MATLAB files. This is useful for when the user wants to have access to the data generated by the PhysioAnalyzers, before it is condensed into epoch summaries.

It is advisable to create a data archive of MATLAB files once everything has been satisfactorily corrected and/or reviewed. Since the PhysioData files do not actually contain the signals generated by the PhysioAnalyzers, it will be impossible to regenerate these signals and hence recreate the analysis without the version of the PhysioData Toolbox originally used.

4 Data Viewers

Data Viewers are graphical user interfaces that display data, such as graphs and tables, and allow the user to navigate and interact with the displayed data.

The PhysioData Toolbox contains two types of Viewers:

- **The Raw Data Viewer:**
 - This viewer displays information about the raw data inside the PhysioData. To open the Raw Data Viewer, select a file in the Session Manager and click the ‘**View Raw Data**’ button.
- **The PhysioAnalyzer Viewer:**
 - The PhysioAnalyzer Viewer is a window for inspecting and interacting with the PhysioAnalyzers inside each PhysioData file. The data of each PhysioAnalyzer is visualized in a separate tab. To open the PhysioData Viewer, select a file in the Session Manager and click the ‘**View Analyses**’ button.

The Data Viewers feature a navigation bar just below the banner, which can be used to change the file currently displayed in the Viewer.

Zooming

The graphs shown in the PhysioData and PhysioAnalyzer Viewer windows all exhibit the same general zooming behavior, which is controlled using the scroll wheel, and the middle and right mouse buttons.

The following zooming actions can be performed when interacting with axes:

- **Selection Zooming:**
 - Right-clicking inside a graph and moving the mouse while holding down the right mouse button creates a transparent blue selection rectangle, releasing the mouse button then zooms in to that rectangle.
- **Horizontal Zooming:**
 - Zooming the horizontal axis is done by scrolling the mouse scroll-wheel; scrolling up zooms in and scrolling down zooms out.
- **Vertical Zooming:**
 - Zooming the vertical axis is done by pressing the mouse scroll-wheel, then scrolling it while holding it down; scrolling up zooms in and scrolling down zooms out.
- **Dragging:**
 - You can drag the graph left, right, up and down by clicking the mouse scroll-wheel button while the cursor is positioned inside the graph, then holding the button down and moving the mouse.
- **Horizontal Panning:**
 - You can pan horizontally—i.e., shift the graphs left or right—by scrolling up or down while holding down the right mouse button.
- **Reset:**
 - Double-click on a graph using the left mouse button to reset the zoom. This will zoom out to a level where all data points fit inside the graph.

Additionally, when a graph features a legend, the legend entries can be clicked to hide the corresponding lines.

4.1 Raw Data Viewer

The Raw Data Viewer displays information about the raw data inside a PhysioData file.

The viewer consists of the following tabs:

- **File Info:**
 - Information about the PhysioData file are displayed in the 'File Info' tab.
- **Raw Signals:**
 - All signals are plotted in their raw form in the 'Raw Signals' tab. It may be necessary to scroll down to see all channels. Each graph can be collapsed or re-expanded by clicking the button with the arrow in the top right corner of the graphs panel.
- **Marker Signals:**
 - The toolbox automatically classifies a raw signal as a 'marker signal' if it only contains non-negative integers with a max value of 65535. Signals classified as a marker signal are automatically analyzed and the detected markers are visualized and tabulated.
- **Labels:**
 - Labels, if present, are plotted and tabulated in the 'Labels' tab.
- **Eye Tracking Dataset #:**
 - Each eye-tracking dataset, if present, is visualized in its own tab, with each type of eye-tracking data occupying a subtab:
 - **Pupil Diameters:**
 - The raw pupil diameters are visualized as a scatter plot, with each dot representing a sample point.
 - **Gaze & Fixation Data:**
 - Both the gaze and fixation coordinates are visualized in an XY map, per snapshot. The snapshot image, if present, is also displayed in the graph. Note that gaze and fixation data is currently not included in the conversion to PhysioData files when using the File Converter to convert EyeLink .edf files or EET .gazedata or .txt files.
 - **Areas of Interest:**
 - The AOIs are plotted as a series of horizontal bars, with each bar signifying the duration of a single hit. Like gaze and fixation data, AOIs are visualized per snapshot.
 - **Events:**
 - Each eye-tracking dataset can contain its own set of events. These events are visualized in a similar manner to Labels.
 - **Scored Sections:**
 - Scored sections are visualized in a similar manner to AOIs, but are independent of snapshots. Note that scored sections are currently not included in the conversion to PhysioData files when using the File Converter to convert EyeLink .edf files or EET .gazedata or .txt files.
- **Epochs in File:**
 - Pre-generated epochs inside the PhysioData files, if present, are plotted and tabulated in the 'Epochs in File' tab.



Figure 12: Visualizations of Signals in the Raw Data Viewer. Left: the ‘Raw Signals’ tab shows all channels inside the file as collapsible vertically stacked axes panels; note the ongoing box-zoom action (right click + drag). Right: the ‘Marker Signals’ tab shows all channels that the toolbox classified as a ‘Marker Channel’, and visualizes their events.

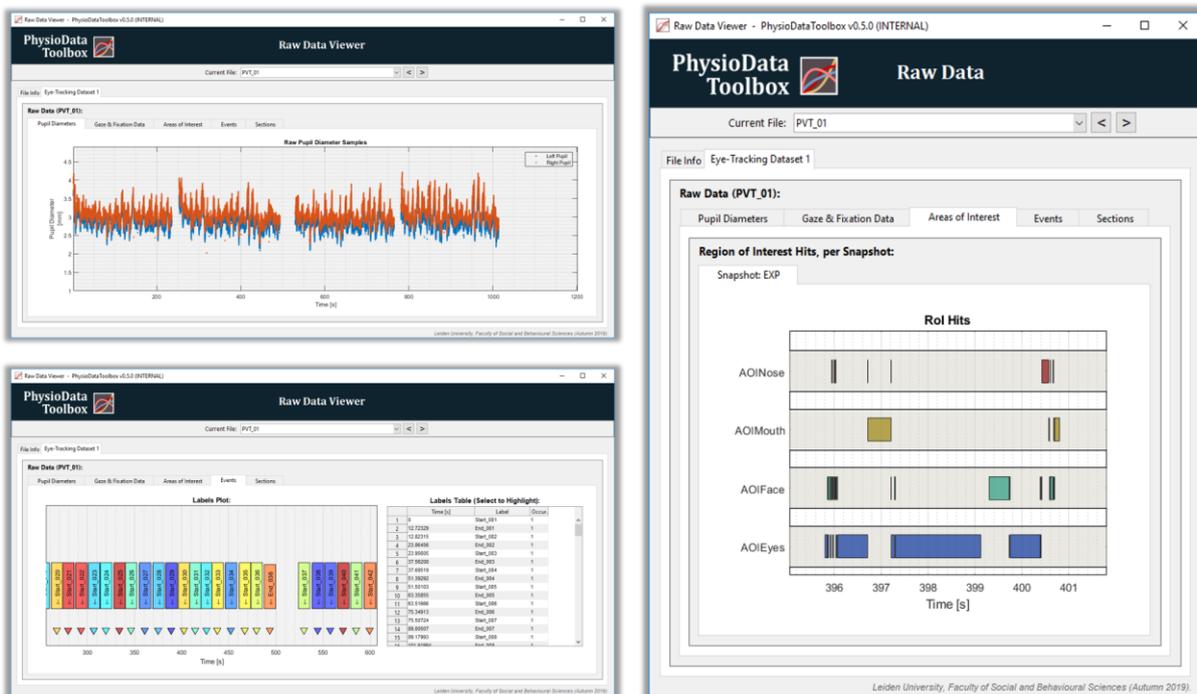


Figure 13: Visualizations of the raw eye-tracking data. Top left: the pupil diameters. Bottom left: events inside the eye-tracking dataset. Right: Region of Interest hits.

4.2 PhysioAnalyzer Viewer

The PhysioAnalyzer Viewer generates one tab for each PhysioAnalyzer inside the current PhysioData file. The contents of the tabs are then custom generated by the PhysioAnalyzers. Standard components featured in the tabs are: **Tab Toolbars**; **Accepted Checkboxes**; **Epoch or Event Bars**; and, **Corrections Graphs**, for displaying the ‘correction zones’.

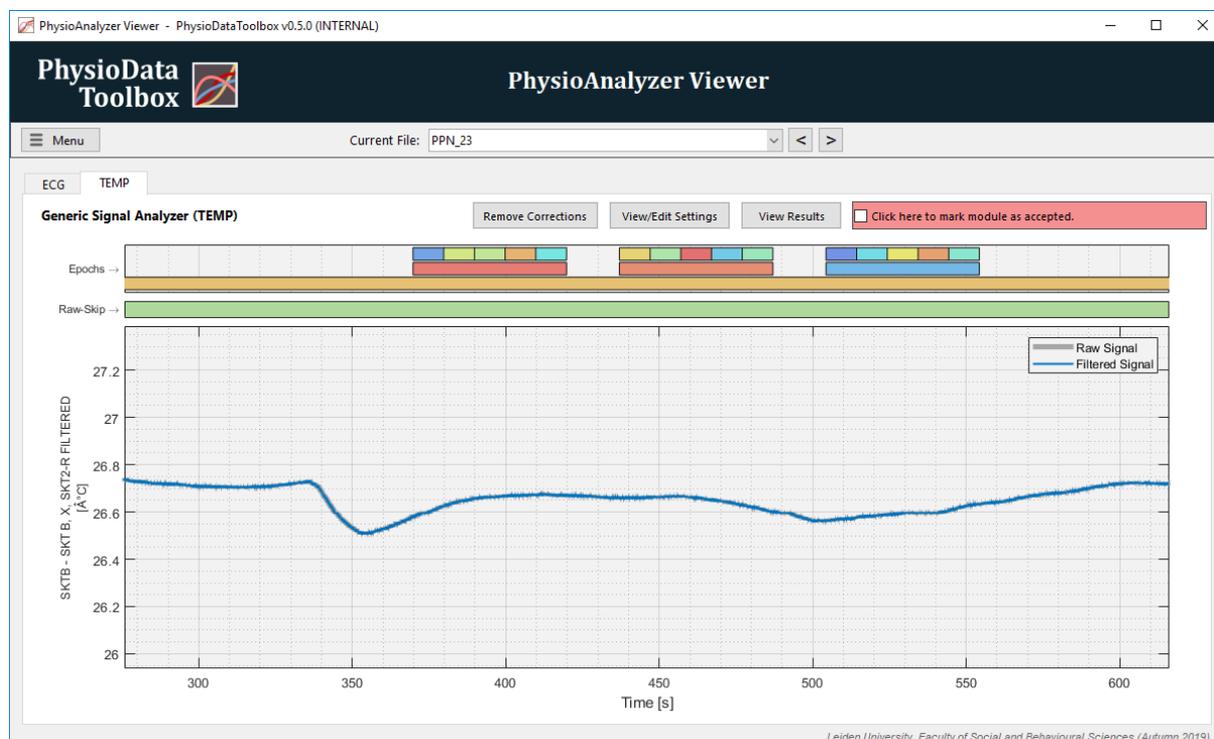


Figure 14: A PhysioAnalyzer Viewer, with each tab representing one PhysioAnalyzer inside the file ‘PPN_23’. The foreground tab shows a Generic Signal Analyzer tagged ‘TEMP’. The Epochs are plotted in the topmost axes, followed by the ‘Raw-Skip’ correction graph, and the main axes.

User Correction Zones

The standard method for correcting data in the toolbox involves the user defining special ‘correction zones’, within which the toolbox treats the data points differently; e.g., by disregarding detected peaks, or skipping the raw data and interpolating over the section instead.

Defining zones rather than having corrections be associated with specific features allows the **State** of the PhysioAnalyzer to be independent of its **Settings**, which allows the settings to be changed without invalidating the corrections.

User correction zones are plotted as bars in one or more ‘Corrections Graphs’. These thin axes display the sections where the user inserted a correction zone. Each PhysioAnalyzer determines what corrections the user may apply, and how these corrections are used. A light green colored corrections-graph indicates that the user did not enter any correction zones in that particular axes, as shown in the Raw-Skip bar in Figure 14. A green-colored bar indicates that there are correction zones present, which are usually plotted using red bars, as in Figure 15.

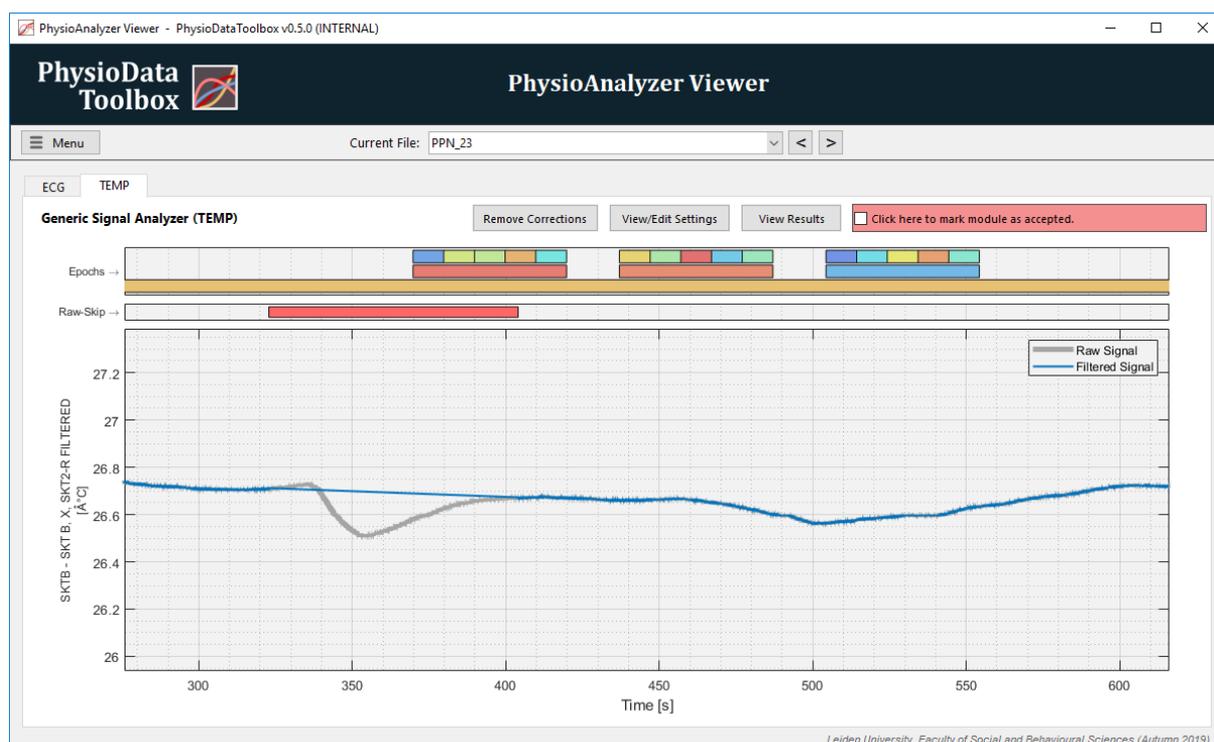


Figure 15: The Generic Signal Analyzer allows the user to mark zones in which the raw data are not used, but is instead linearly interpolated before filtering. The red bar in the Raw-Skip shows the marked section, notice how the filtered signal differs from the raw signal.

Menu

The 'Menu' button in the PhysioAnalyzer Viewer offers the following options:

- **Automatically plot all tabs:**
 - When this option is checked, data in all tabs are plotted when the PhysioAnalyzer Viewer is opened as opposed to when the tab is clicked on.
- **Use Fast-Mode selection when possible:**
 - When this option is checked, Fast-Mode is enabled. In Fast-Mode, the Section Editor does not appear, instead, the correction zone is immediately created thereby allowing faster data correction. Left-click in the graph, then drag to the right to 'Disregard Data' or drag to the left to 'Use Data'. Note that this mode cannot be activated in all PhysioAnalyzers, such as the Pupil Diameter Analyzer.

Tab Toolbar

Each tab has a toolbar displaying the PhysioAnalyzer type and its tag, and features an 'Accepted Checkbox', and the following buttons:

- **Remove Corrections:**
 - Removes all user corrections in the current PhysioAnalyzer; i.e., it resets the PhysioAnalyzer's state. The settings, however, keep their current parameters.
- **View/Edit Settings:**
 - Opens the settings window for the current PhysioAnalyzer. Only the settings in the current PhysioAnalyzer are affected by changes made here. Not all settings can be changed in the individual PhysioAnalyzers; the tag, and Epoch or Event Definitions, for instance, must be changed using the PhysioAnalyzer Creator in the Session Manager.
- **View Analysis:**
 - Opens a window containing a table showing the Epoch or Event analysis of the current PhysioAnalyzer. Multiple of these windows can be opened, and the tabulated results are 'analyses snapshots'; i.e., they are not updated when the PhysioAnalyzer is changed.

Accepted Checkbox

The Accepted Checkbox can be used to mark PhysioAnalyzers as ‘reviewed’ or ‘accepted’. This feature exists purely for user reference as it does not influence how, or if, the PhysioAnalyzer is analyzed. Changing the settings, or adding or removing corrections, does not reset the accepted status.



Figure 16: The Accepted Checkbox, in its default (top) and activated (bottom) states.

Epoch and Event Graphs

Epochs, which are sections within a signal, are plotted as non-overlapping rectangles. Their colors are randomly chosen, and their height and thickness do not carry any information.

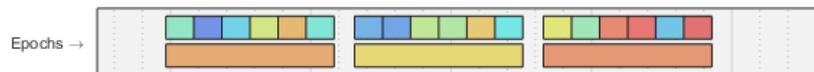


Figure 17: An Epoch plot, showing three minute-long epochs that have been sliced into 10 second segments.

Certain analyzers make use of Events rather than Epochs. Events are discrete—i.e., they have no duration—and are visualized with colored triangles.



Figure 18: An Event plot showing nine events.

The Epoch and Event graphs can themselves not be zoomed or panned, but are instead linked to the horizontal limits of the other axes inside the tab. Clicking on an epoch rectangle or an event triangle opens a window displaying its information.

Graphs

Each module creates its own customized set of graphs and plots. Generally, right-clicking inside a graph without moving the mouse opens a menu with zooming and PhysioAnalyzer-specific analysis and interaction options.

5 Generic Signal Analyzer

The Generic Signal Analyzer is the ‘standard’ PhysioAnalyzer module, which all other modules are based on. This PhysioAnalyzer type performs basic descriptive analyses on miscellaneous signals, and allows the user to indicate noisy sections that the toolbox can then interpolate over.

Unlike the other PhysioAnalyzers, the units and labels in the Generic Signal Analyzer are taken directly from the raw data channels. These cannot be edited in this version of the toolbox. This module can be used for signals such as: body temperature, heartrate, etc.

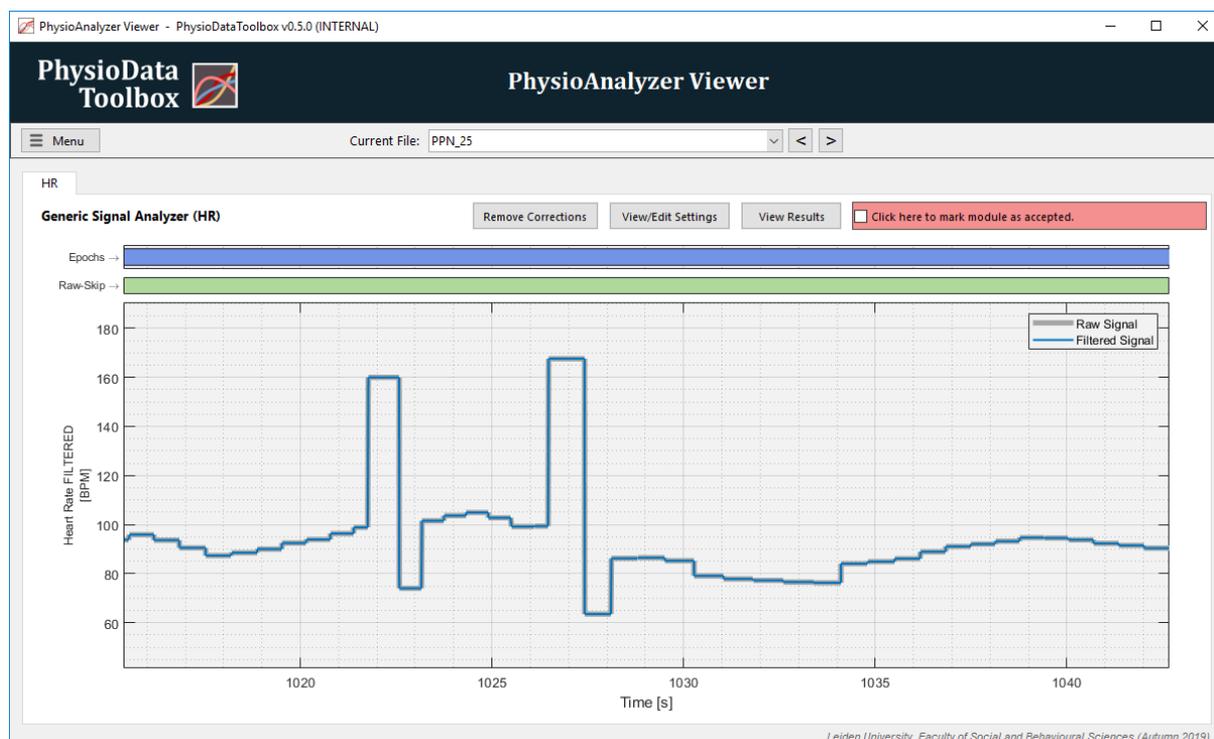


Figure 19: A Generic Signal Analyzer tagged HR, showing a preprocessed heartrate signal.

5.1 Settings

The Generic Signal Analyzer features standard PhysioAnalyzer settings also available in several other modules.

General Settings

The general settings determine the name, data-source and epoch settings for a module. The tag is hardcoded at module creation and cannot subsequently be modified.

The General Settings for the Generic Signal Analyzer:

- **Analyzer prefix (tag):**
 - The tag will be the name of the PhysioAnalyzer. It must: be unique; contain only numbers, letters and underscores; and, start with a letter.
- **Channel-number of signal:**
 - The channel-number (index) of the signal to analyze. Use the ‘See Channels’ button to determine what the channel-number of the desired signal is. This value must be a valid number.
- **Generate Epochs from:**
 - This allows the user to select two epoch generation methods:
 - **PhysioAnalyzer with tag:**

- This option causes the PhysioAnalyzer to use the epochs defined in another analyzer, the tag of which must be entered in the text-field.
- **Epoch Definition Table:**
 - Select this option to have the PhysioAnalyzer generate epochs based on epoch definition rules. Click the button to open the Epoch Builder. See the ‘Epochs and Events’ section for details. An epoch spanning the whole signal is always created by default.

Preprocessing Settings

The preprocessing settings determine how the raw data signal is transformed into the filtered signal, which is what is ultimately analyzed—or further processed, in the case of other PhysioAnalyzer.

The following preprocessing settings are available:

- **Gain (Signal Multiplier):**
 - The gain is the value by which the raw signal is multiplied before it is further processed and analyzed.
- **Highpass Filter [Hz]:**
 - If the checkbox is enabled, the signal will be filtered using a 2nd order Butterworth zero-phase forward and reverse high-pass IIR filter, with the entered cutoff frequency.
- **Lowpass Filter [Hz]:**
 - If the checkbox is enabled, the signal will be filtered using a 2nd order Butterworth zero-phase forward and reverse low-pass IIR filter, with the entered cutoff frequency.

Before applying the digital filters, the cutoff frequencies are normalized using:

$$\omega_0 = \frac{2 \cdot f_{cutoff}}{f_{sampling}}$$

The normalized cutoff frequencies (ω_0) must be between 0 and 1, otherwise the signal will not be filtered, nor analyzed.

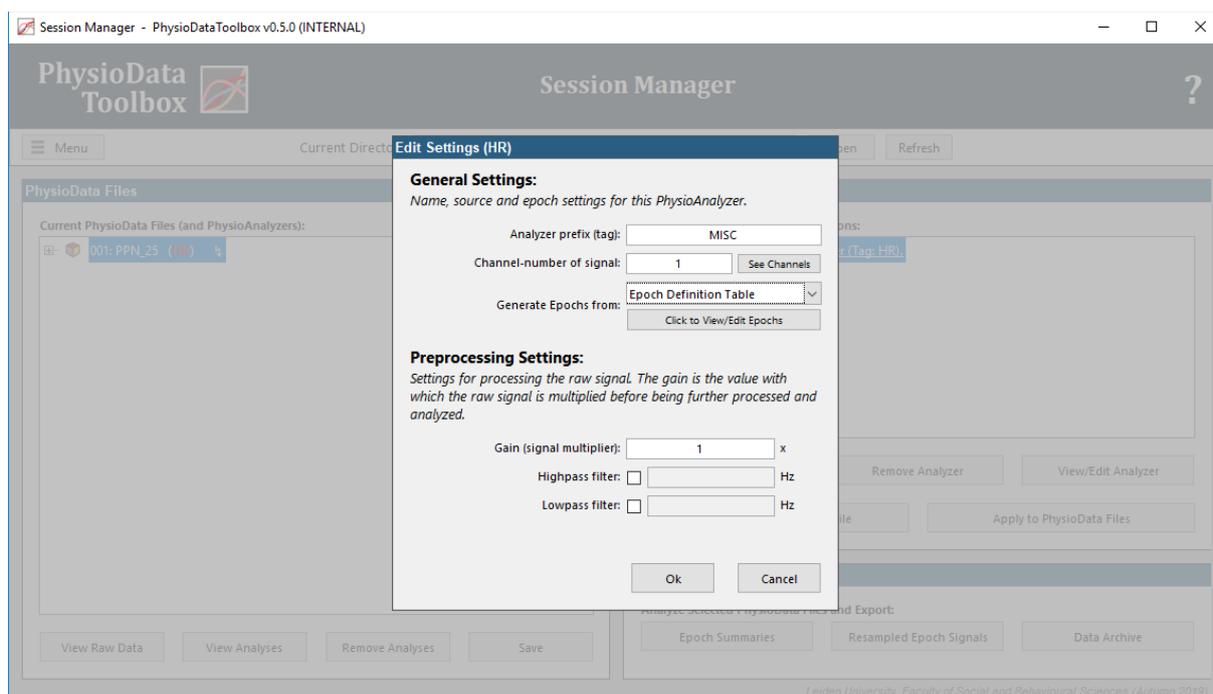


Figure 20: The settings dialog of the Generic Signal Analyzer.

5.2 Metrics

The Generic Signal Analyzer performs basic epoch-based analyses on signals, and extracts descriptive statistical metrics per epoch. One possible application is to calculate the means per epoch of a body temperature signal.

Table 1: Generic Signal Analyzer Metrics

Code:	Unit:	Description
mean	*	The mean value of the valid samples inside an epoch.
Min	*	The minimum value of the valid samples inside an epoch.
minOnset	s	The time between the onset of an epoch and the occurrence of the minimum value.
Max	*	The maximum value of the valid samples inside an epoch.
maxOnset	s	The time between the onset of an epoch and the occurrence of the maximum value.
Std	*	The standard deviation of the valid samples inside an epoch.
validSampleCount	count	The number of valid samples inside an epoch.

* The unit is taken from the channel metadata.

Unless otherwise specified, the toolbox uses the following conventions:

- **mean:**
 - The arithmetic mean of all valid values (inside a specific epoch or within a specific set).
 - The mean is calculated using MATLABs `nanmean` command.
- **std:**
 - The standard deviation of all valid values (inside a specific epoch or within a specific set), normalized by $(N-1)$, where N is the sample size.
 - The standard deviation is calculated using MATLABs `nanstd` command.

5.3 Resampled Signals

When exporting the resampled epoch signals, the Generic Signal Analyzer resamples and extracts the filtered signal.

5.4 Data Correction

Artifact removal in the Generic Signal Analyzer occurs through the user marking sections of the signal as invalid. These sections, aka ‘Raw-Skip’ zones, are not extracted from the raw data, but are instead linearly interpolated before filtration. In the case that a section extends beyond the start or end of the signal, extrapolation in that section occurs via the nearest-neighbor method.

Data correction options in the Generic Signal Analyzer:

- **Removing raw data artifacts:**
 - To mark or unmark a section of the raw signal for interpolation, left-click in the graph at the start-time of the desired section, drag the mouse, and release at the end-time of the desired section. Choose the desired action from the pop-up menu.
 - ‘Disregard Data’ tells the toolbox to reject the raw data samples within the marked section, and replace them with linearly interpolated values.

- ‘Use Data’ tells the toolbox to use the raw data inside the section. This option is the opposite of the ‘Disregard’ option, and can be used to remove a section marked as ‘Disregard Data’.

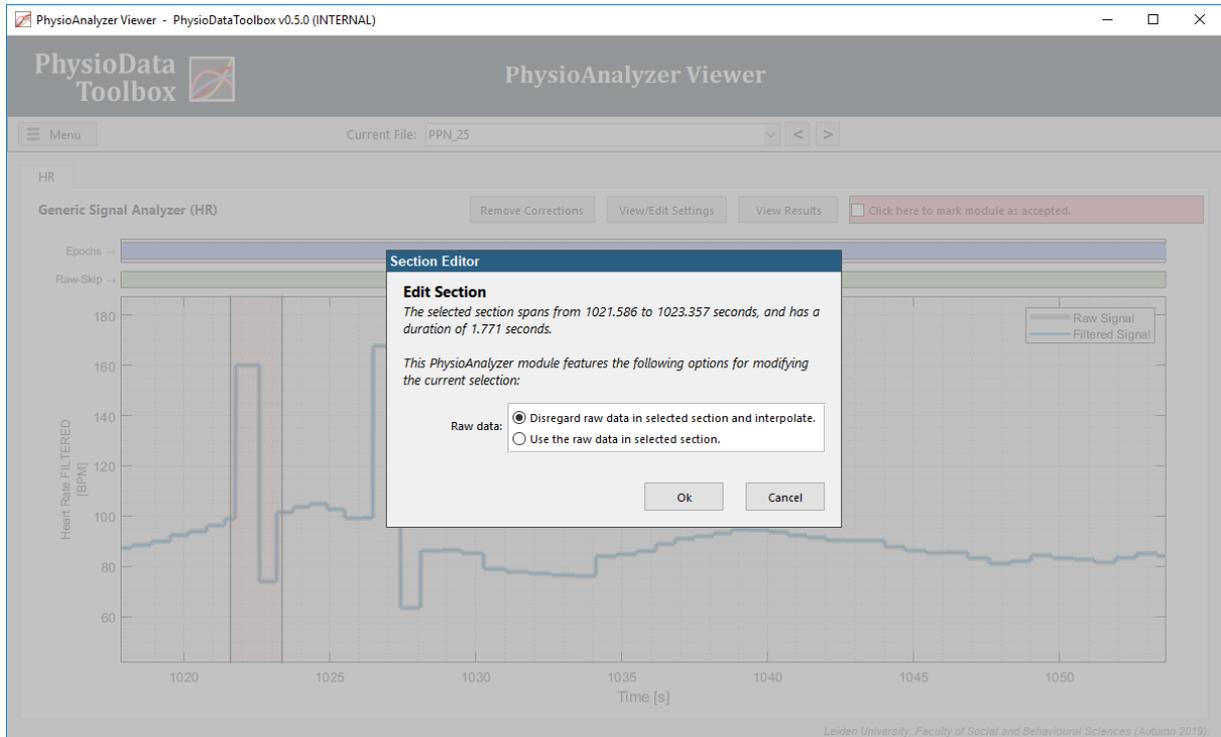


Figure 21: Marking a correction zone for interpolation: after the red section has been selected by dragging the left mouse button, the pop-up menu presents two options: disregard the raw data and interpolate; or, use the raw data. Note that when Fast-Mode is checked in the Menu, this popup menu will not appear and the raw data is immediately disregarded.

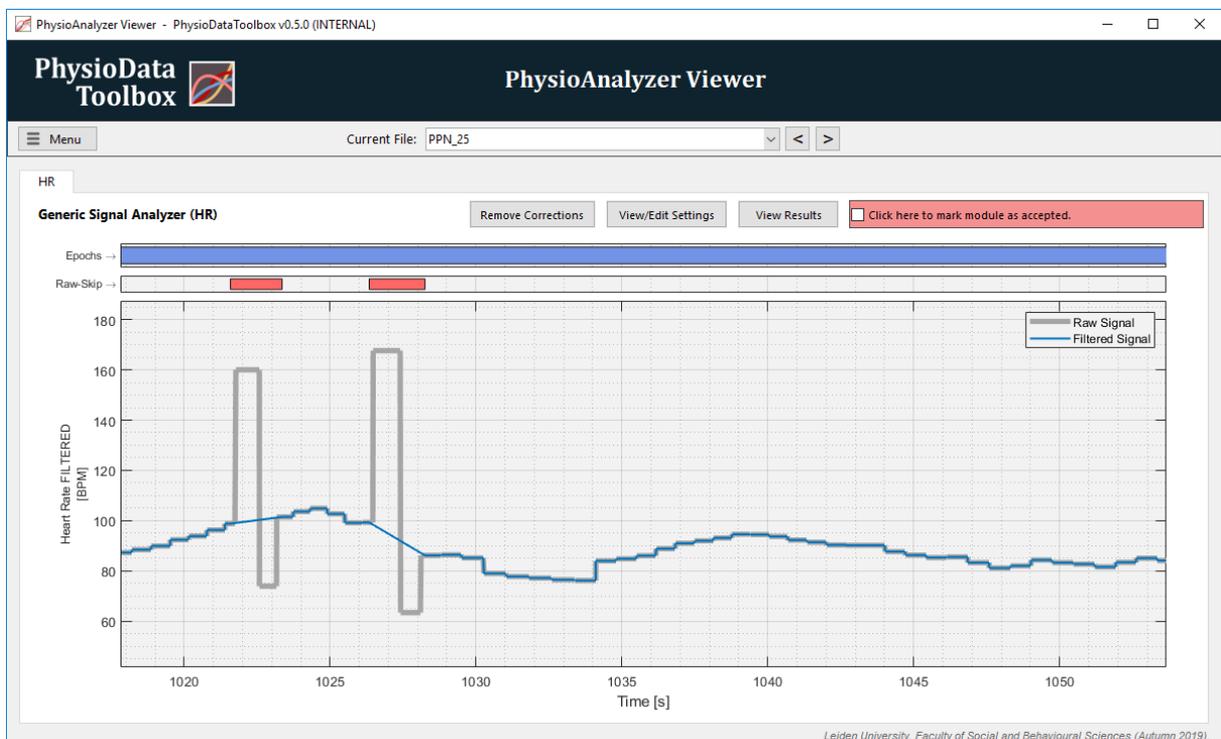


Figure 22: The heartrate signal, which featured two artifacts, has been corrected by inserting two interpolation zones.

6 ECG Signal Analyzer

The ECG Signal Analyzer performs epoch-based time-domain analysis of ECG signals and allows the user to perform basic R-peak and IBI correction.

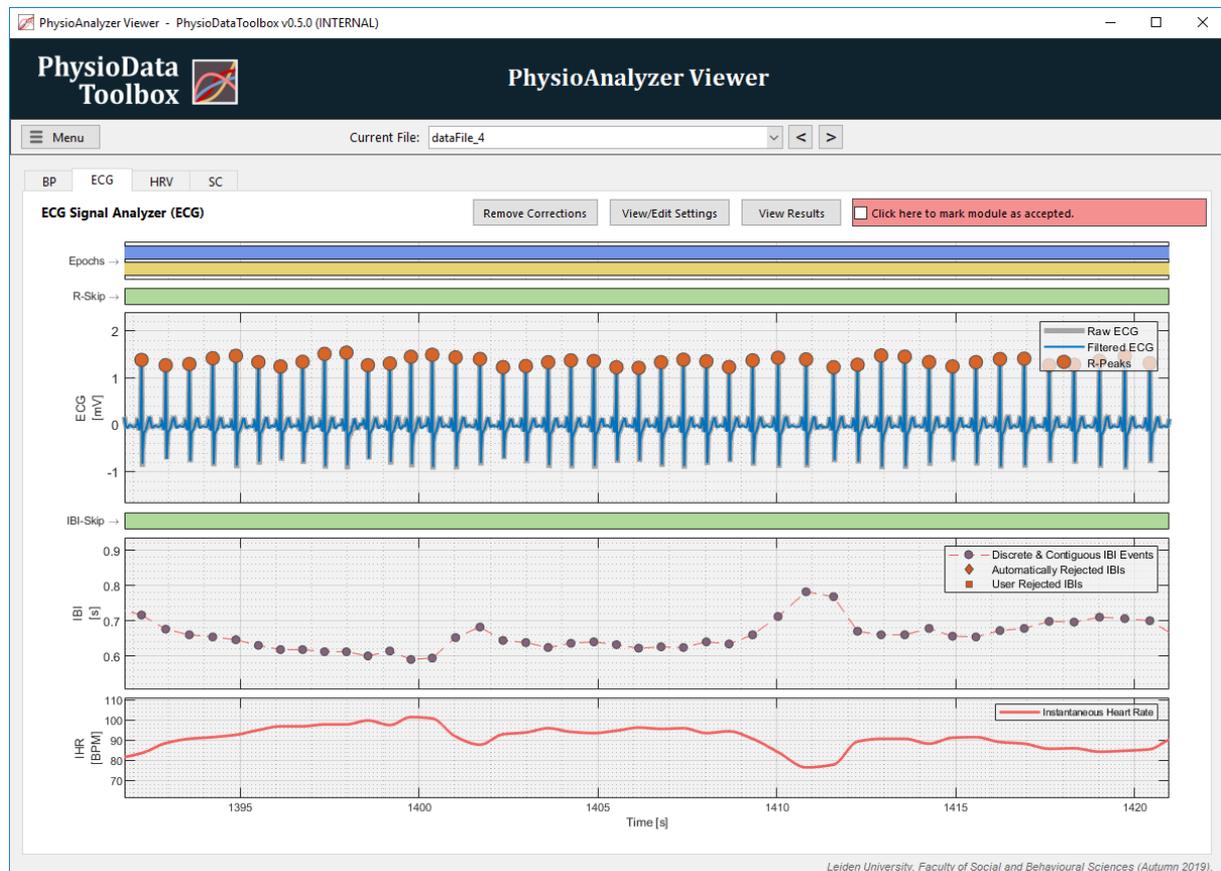


Figure 23: The ECG Signal Analyzer interface, showing the raw and filtered ECG signals, the detected R-peaks, the IBIs and the instantaneous heart rate.

The ECG Signal Analyzer module features a GUI containing the following plots:

- **ECG (top):**
 - Shows the raw and the filtered ECG signals.
- **IBI (middle):**
 - Shows the IBI events, and draws a dashed line between them for easier visual detection of outliers. Note, however, that the IBIs are purely discrete events, and not a continuous signal.
- **IHR (bottom):**
 - Shows the continuous Instantaneous Heart Rate, as calculated from the IBIs.

Read the provided walkthrough document for a guide on how to create an ECG module and use it to correct and analyze ECG data.

6.1 Settings

See the description in the Generic Signal Analyzer section for details on the General and Preprocessing Settings.

The ECG Signal Analyzer assumes that the unit of the ECG signal is mV. If this is not the case, fill in an appropriate gain value (multiplier) to transform the raw signal into mV; e.g., if the signal is in V, the gain should be 1000. Additionally, the gain can be used to flip the ECG signal if necessary. Since the

toolbox cannot detect downwards pointing R-waves, 'upside-down' ECG signals must be flipped by applying a negative gain.

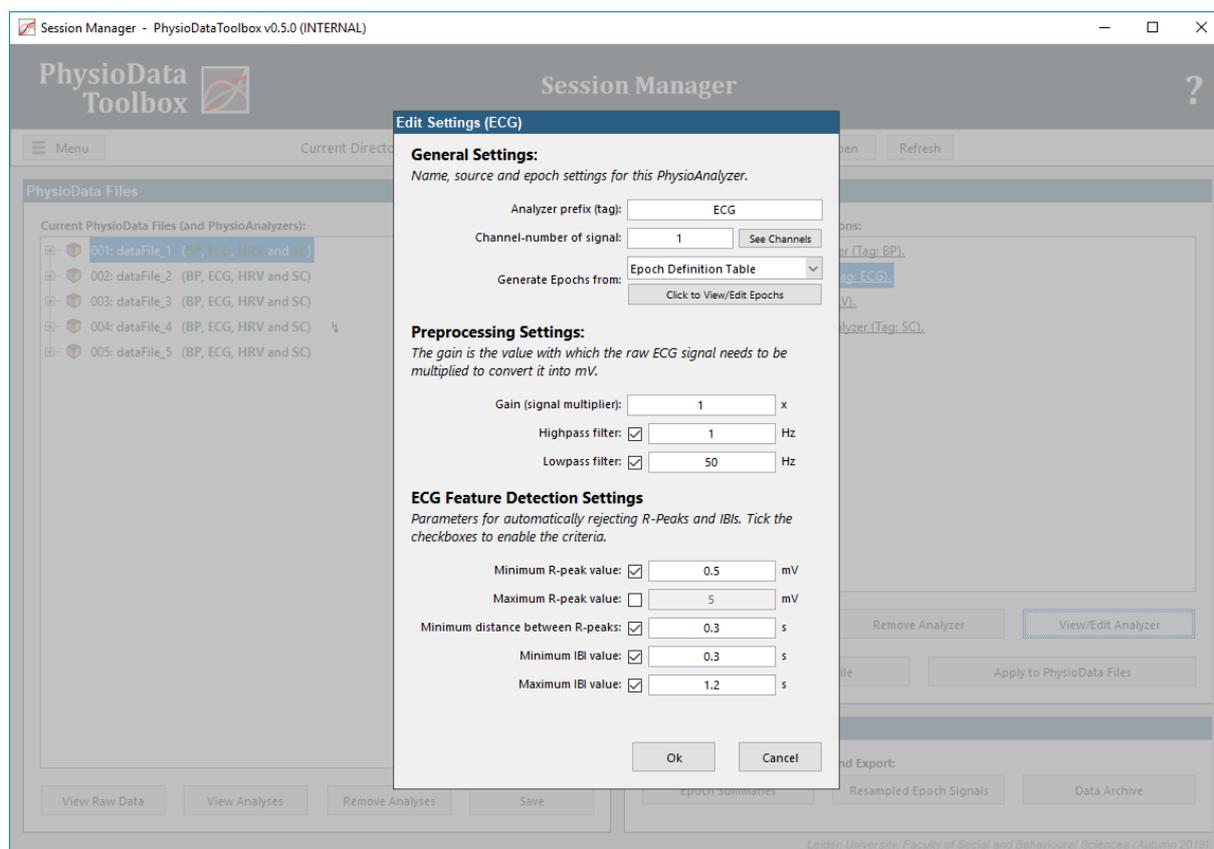


Figure 24: The settings dialog of the ECG Signal Analyzer.

Feature Detection Settings

Automatically detected R-peaks and the subsequently extracted IBIs can both be automatically corrected using predefined rules.

Feature detection settings of the ECG Signal Analyzer:

- **Minimum R-peak value [mV]:**
 - R-peaks located lower than this value on the filtered ECG curve will not be detected.
- **Maximum R-peak value [mV]:**
 - R-peaks located higher than this value on the filtered ECG curve will be rejected.
- **Minimum distance between R-Peaks [s]:**
 - Only the highest R-peak within this distance from itself will be accepted.
- **Minimum IBI value [s]:**
 - IBIs shorter than this duration will be automatically rejected.
- **Maximum IBI value [s]:**
 - IBIs longer than this duration will be automatically rejected.

6.2 Metrics

The ECG Signal Analyzer extracts standard cardiac metrics from user-corrected ECG data.

Table 2: ECG Signal Analyzer Metrics

Code:	Unit:	Description
HR_mean	BPM	The mean of the continuous Heart Rate signal, as generated through shape-preserving piecewise cubic interpolation at 100 Hz of the accepted IBI data points. The mean is calculated per epoch.
R_peakCount	count	The number of accepted R-peaks inside an epoch.
IBI_mean	s	The arithmetic mean of the accepted discrete IBIs inside an epoch*.
IBI_weighted_mean	s	Self-weighted mean of the accepted discrete IBIs inside n epoch*. The self-weighted mean represents the value of the IBI, weight by its duration, which is also its value: $\overline{IBI}_{Weighted} = \frac{\sum IBI^2}{\sum IBI}$
IBI_min	s	The minimum value of the accepted discrete IBIs data points inside an epoch*.
IBI_max	s	The maximum value of the accepted discrete IBIs data points inside an epoch*.
IBI_std**	s	The standard deviation of the accepted discrete IBIs data points inside an epoch*.
IBI_count	count	Count of accepted IBI data points inside an epoch*.
IBI_coverage	%	The percentage-wise IBI coverage of an epoch, calculated as the sum of the IBIs inside an epoch divided by the duration of that epoch, multiplied by 100. IBI coverage may exceed 100% due to the way IBIs are categorized*.
HRV_RMSSD**	ms	The Root Mean Squared of the Successive Differences, of the IBIs inside an epoch. Nonadjacent IBIs are not used to calculate the successive differences (SSD).
HRV_ssdCount	count	The number of SSDs inside an epoch.
HRV_pNN20**	%	Percentage of absolute SSDs that are greater than 20 ms, per epoch.
HRV_pNN50**	%	Percentage of absolute SSDs that are greater than 50 ms, per epoch.

* The IBIs are temporally located at the time of the R-peak that ends that IBI; as such, IBIs that start outside of the epoch but end inside it are still considered part of that epoch.

** These metrics are calculated on the non-detrended IBI signal, as opposed to the HRV module, which calculates these metrics on the detrended IBI signal.

6.3 Resampled Signals

When exporting the resampled epoch signals, the ECG Signal Analyzer resamples and extracts the instantaneous HR signal.

6.4 Data Correction

Next to specifying automatic feature rejection parameters in the settings, the ECG Signal Analyzer allows the user to correct the generated data by marking two types of rejection zones: R-peak rejection zones; and, IBI rejection zones. The raw data cannot be interpolated in this version of the toolbox, nor can R-peaks be manually added.

Data correction options of the ECG Signal Analyzer:

- **Rejecting faulty R-peaks:**
 - To mark or unmark a section of the filtered ECG signal as an R-Peak exclusion zone, left-click in the ECG graph at the start-time of the desired section, drag the mouse, and release at the end-time of the desired section. Choose the desired action from the menu.
- **Rejecting faulty IBIs:**
 - To mark or unmark a section of the IBI time series as an IBI exclusion zone, left click in the IBI graph at the start-time of the desired section, drag the mouse, and release at the end-time of the desired section. Choose the desired action from the menu.

Correction Cases and Workflow

Firstly, if the ECG Signal Analyzer consistently fails to properly detect R-peaks, consider changing the automatic rejection criteria in the Settings window. This can be done per file via the PhysioAnalyzer Viewer; or, if the issue persists across files, via the PhysioAnalyzer Configurator, and then applied to selected files.

In the case that either noise or a secondary ECG wave is erroneously detected as an R-Peak, two distinctly shorter consecutive IBIs will be visible in the plot. To correct this, insert an R-peak exclusion zone that covers the wave producing the erroneous R-peak. This will cause the faulty R-peak to be rejected, and will merge the two short IBIs into a single valid IBI.

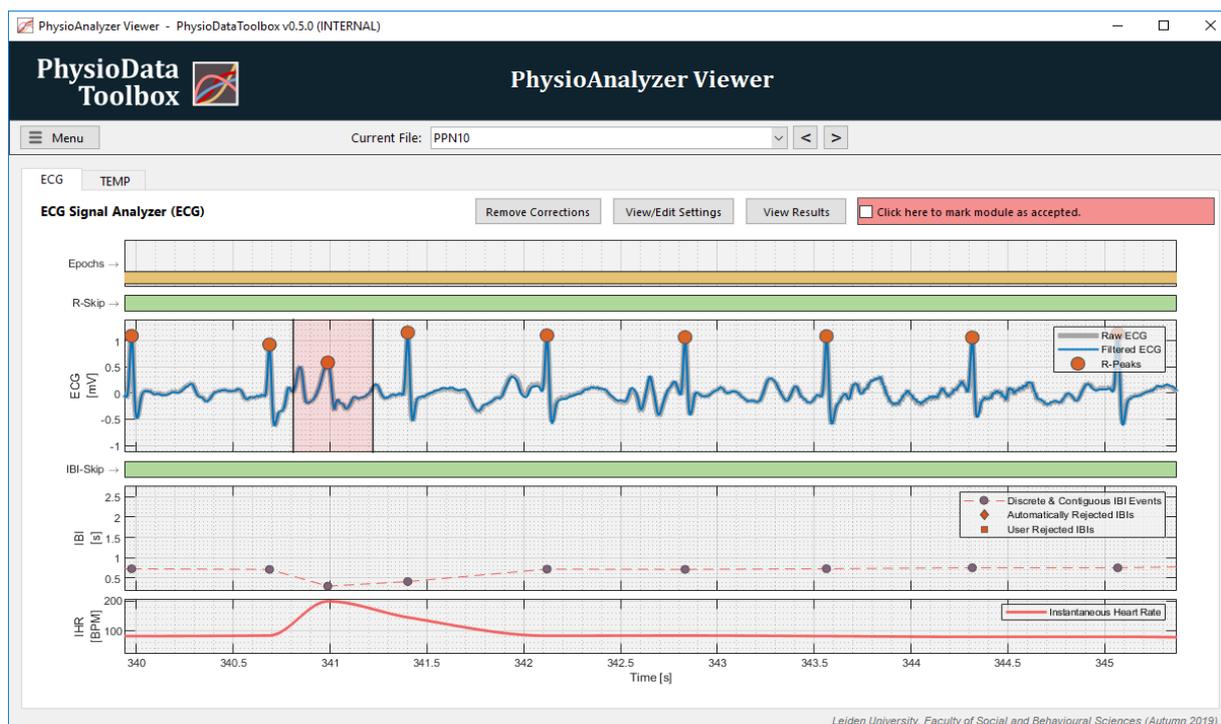


Figure 25: Excluding an erroneous R-peak, located at approximately 341 seconds: first select the section in the ECG axes using the left mouse button, then select the disregard option from the popup menu.

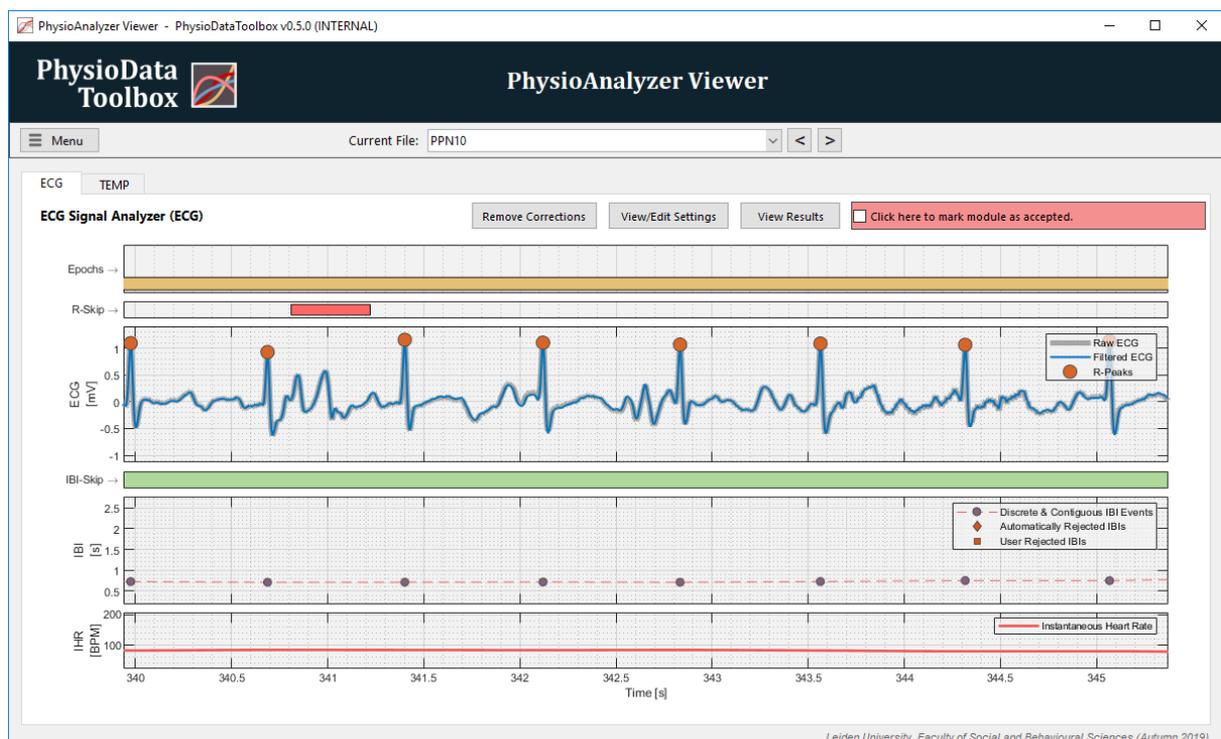


Figure 26: An R-peak exclusion zone has now been created, and the erroneous R-peak removed. Notice how the IBIs and IHR lines are updated.

Conversely, a missed R-peak results in a single distinctly long IBI. This can be corrected by inserting an IBI exclusion zone over the long IBI. Missed R-peaks can be caused by temporary loss of transmission when using wireless systems. In those cases, the loss of transmission produces a flat-line in the raw signal, which can sometimes be seen at the spot where the missing R-peak is expected to have been.



Figure 27: Excluding an erroneous IBI, located at approximately 335.7 seconds: first select the section in the IBI axes by clicking the left mouse button and dragging the cursor; then, select the 'disregard' option from the menu.

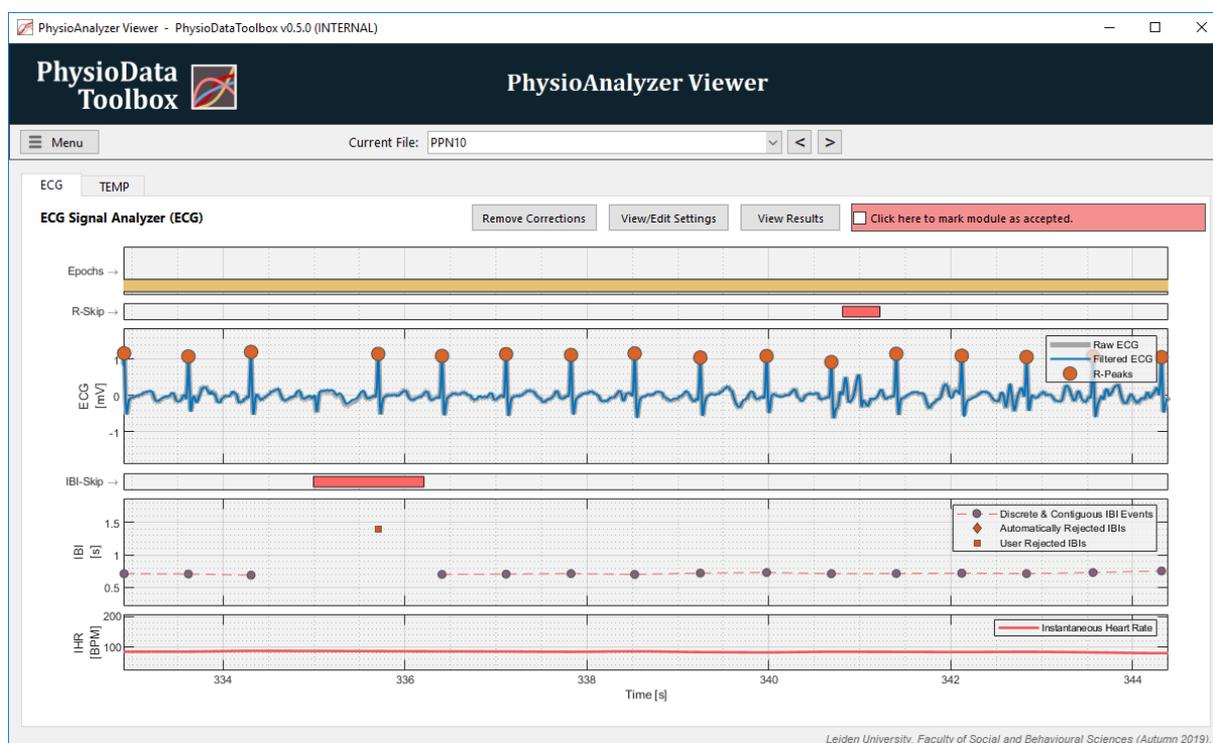


Figure 28: An IBI exclusion zone has now been created, and the erroneous IBI removed. Notice how the IHR line is also updated. IBIs that are rejected based on the criteria defined in the Feature Detection Settings are marked using a diamond, whereas IBIs that are rejected based on the exclusion zones are marked using squares, as in the figure above.

7 IBI Sequence Analyzer

The IBI Sequence Analyzer performs event-based IBI sequence extraction using the IBIs generated and corrected in an ECG Signal Analyzer, and events defined using Event Definition Rules. Per event, the neighboring IBIs are detected and indexed.

This module requires that an ECG Signal Analyzer be present in the PhysioAnalyzer file. The IBIs used in the IBI Sequence Analyzer are automatically updated when the linked ECG Signal Analyzer changes.

The output of this module consists of the IBIs surrounding events, and their delays relative to those events. As such, with some post processing, this module allows anticipatory and evoked IBIs to be analyzed both as a function of their beat index (i.e., beat -1, beat 0, beat 1, beat 2, etc. around a stimulus); and, through interpolation, as a function of relative time (i.e., IBI at -1, 0, 1, 2 etc. seconds around a stimulus).

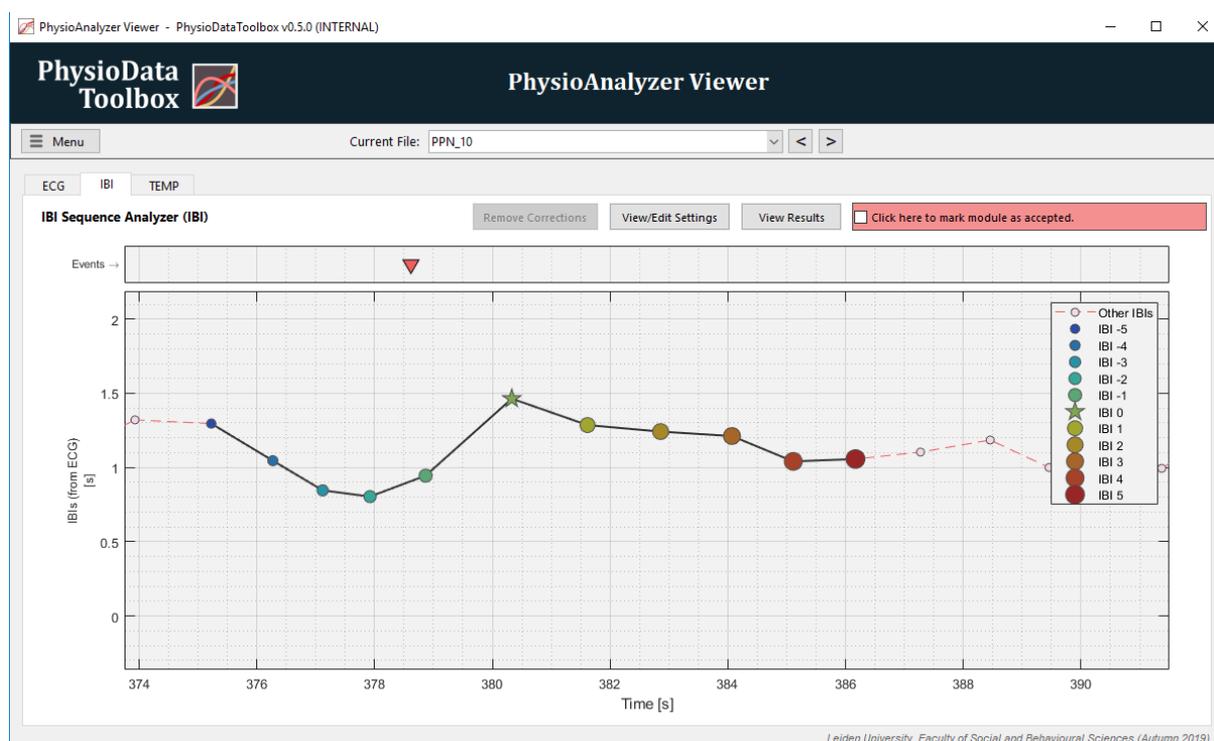


Figure 29: The IBI Sequence Analyzer, showing IBIs -5 to 5 surrounding an event (yellow triangle). The size of the IBI circles increase with their index, which allows the user to detect when IBIs from one event overlap IBIs from another event. Note that IBI(-1) is the first IBI to occur after the event, however, it is not characterized as IBI(0) due to the threshold being set at 0.5.

7.1 Settings

General Settings

The IBI Sequence Analyzer requires its own tag, and the tag of the ECG Signal Analyzer from which the IBIs are to be used. Unlike other PhysioAnalyzers, the IBI module analyzes data around Events rather than inside Epochs. Events can be defined using the Event Builder, see the Epochs and Events section for details.

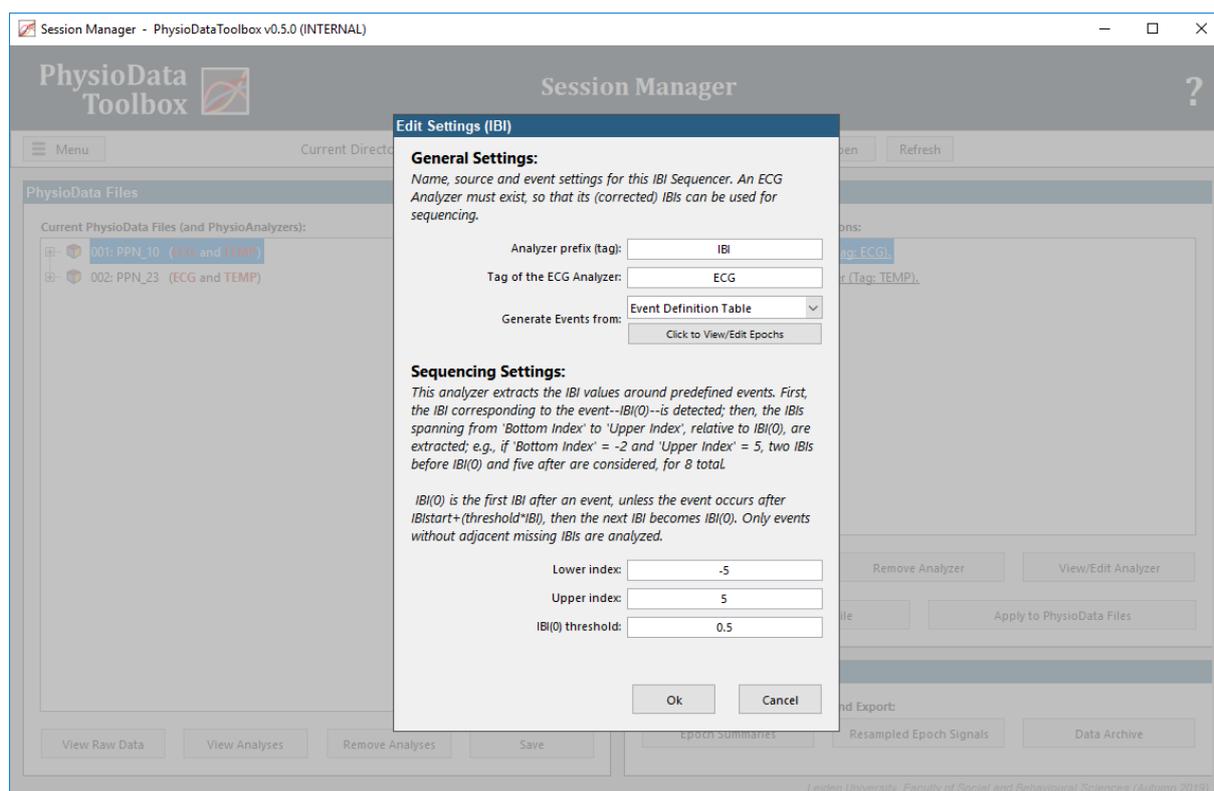


Figure 30: The IBI Sequence Analyzer settings dialog.

Sequencing Settings

When indexing the IBIs around events, the IBI corresponding to the event itself, IBI(0), is first identified. IBI(0) is the first IBI after an event, unless the event occurs after a certain fraction of that IBI, in which case the next IBI becomes IBI(0). The time before which the event must occur so that a given IBI remains IBI(0) is calculated using:

$$t_{cutoff} = t_{IBIstart} + threshold \cdot IBI$$

After IBI(0) is defined, the IBIs spanning from 'Bottom Index' to 'Upper Index', relative to IBI(0), are extracted; e.g., if 'Bottom Index' = -2 and 'Upper Index' = 5, two IBIs before IBI(0) and five after are considered, for 8 in total. **NOTE: Only events where all neighboring IBIs are present are analyzed.**

Sequencing options of the IBI Sequence Analyzer:

- **Lower Index:**
 - The lower index of the IBIs to be analyzed.
- **Upper Index:**
 - The upper index of the IBIs to be analyzed. This index must be greater than the lower index.
- **IBI(0) threshold:**
 - The threshold for classifying IBI(0).

7.2 Metrics

The output of the IBI Sequence Analyzer is determined by the settings; for each IBI index (e.g., IBI(-1) or IBI(5)), two columns of data are produced. Empty rows may indicate that that event features missing IBIs.

Table 3: IBI Sequence Analyzer Metrics

Code:	Unit:	Description
IBI_<index>	s	The mean value of the IBI with index <index>, per event.
Delay_IBI_<index>	s	The time of the IBI, relative to the event, per event.

7.3 Resampled Signals

No resampled signal can be extracted. The resampled instantaneous HR signal can be extracted using the ECG Signal Analyzer.

7.4 Data Correction

This PhysioAnalyzer itself does not support data correction; however, the IBIs can be corrected in the linked ECG PhysioAnalyzer.

8 HRV Analyzer

The Heart Rate Variability (HRV) analyzer retrieves corrected IBI data from a linked ECG module, detrends it, and performs epoch-based analyses to extract several time-domain, frequency-domain and non-linear HRV measures. Basic usage of this module is also outlined in the Walk-through document that accompanies this User Guide.

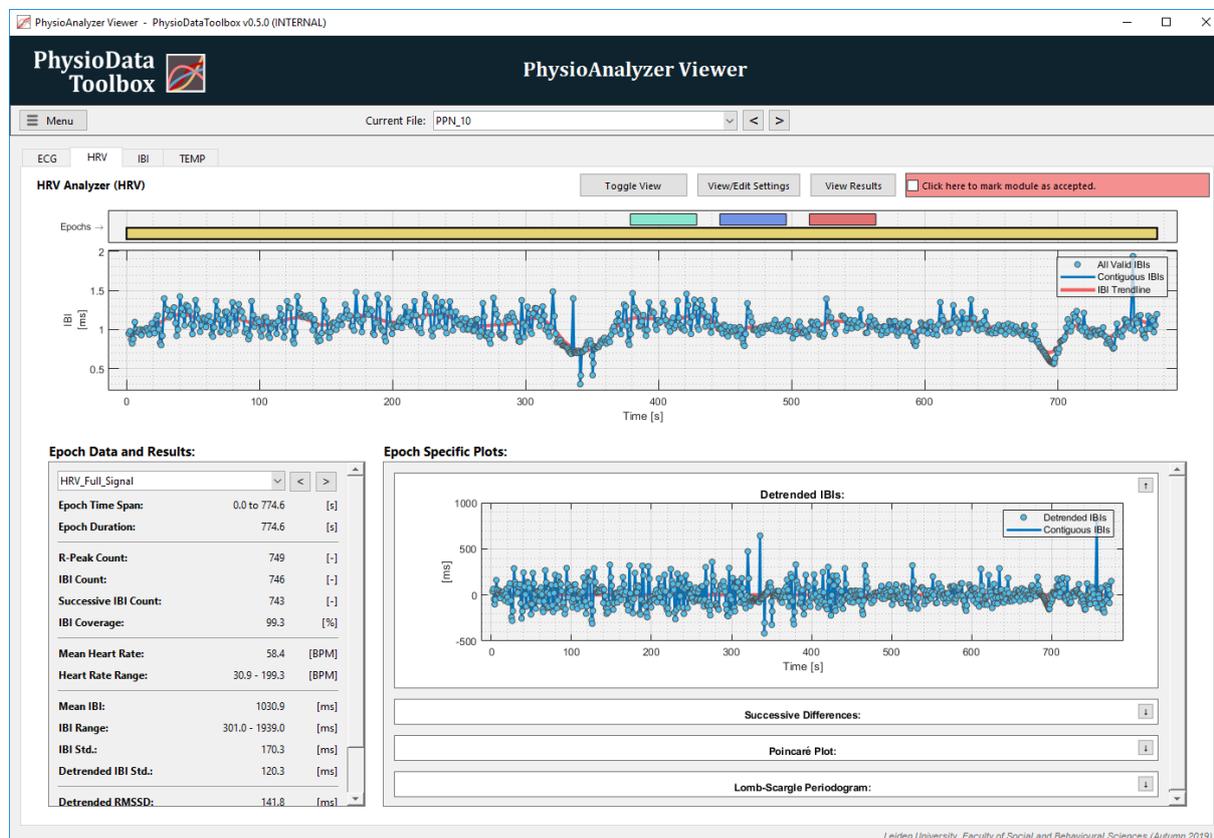


Figure 31: The HRV module’s GUI. The top graph shows the epochs (green, blue and red rectangles), and the graph below that shows the IBI events (blue circles), the contiguous IBIs (blue line) and the global trend baseline (red line). The bottom left panel shows the name of the current epoch in the dropdown menu, and under it various metrics for that epoch. The bottom right panel shows collapsible graphs that visualize the: detrended IBIs; IBI successive differences; Poincaré plot; and Lomb-Scargle periodogram (for power analysis).

The two topmost graphs of the HRV module visualize the epochs, the IBI data that were retrieved from the ECG module and—if enabled—the global trend baseline. Below those are two panels labeled ‘Epoch Data and Results’ and ‘Epoch Specific Plots’, which show the analysis results and plots of the selected epoch.

8.1 Analysis Types

The HRV module performs a collection of time-domain, frequency-domain and non-linear analyses on detrended IBIs located within a specific epoch. See the ‘Metrics’ section for more information about the output, and the ‘Data Pipeline’ section for information about the processing.

Time-Domain Analysis

The HRV module extracts the following standard time-domain metrics from the detrended IBI data:

- Percentage of absolute differences between successive IBIs that exceed 20 ms (pNN20) and 50 ms (pNN50).
- IBI standard deviation.
- Root Mean Squared of Successive Differences (RMSSD).

The RMSSD and the successive IBI differences from which it was calculated are visualized in the module’s ‘Successive Differences’ graph in the ‘Epoch Specific Plots’ panel. Also see Figure 32.

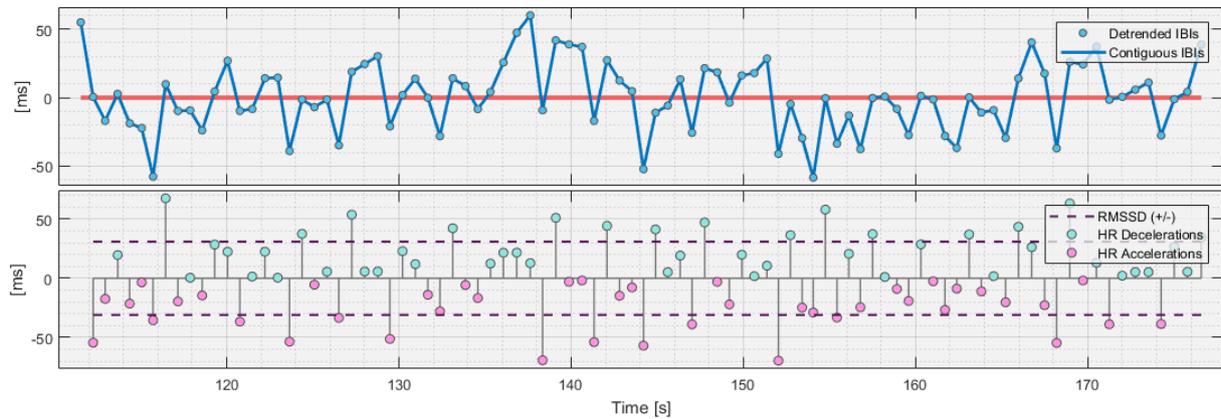


Figure 32: The detrended epoch-specific IBIs (top) and the differences between them (bottom). For visualization purposes, the heart rate acceleration events are plotted in pink and the heart rate deceleration events in aquamarine. Note that a heartrate acceleration is defined as an increase in heartrate, which equals a decrease in interbeat interval.

Frequency-Domain Analysis

The Lomb-Scargle method is used to estimate the IBI time-series’ Power Spectral Density (PSD), from which the Very Low Frequency (VLF), Low Frequency (LF), and High Frequency (HF) powers are computed.

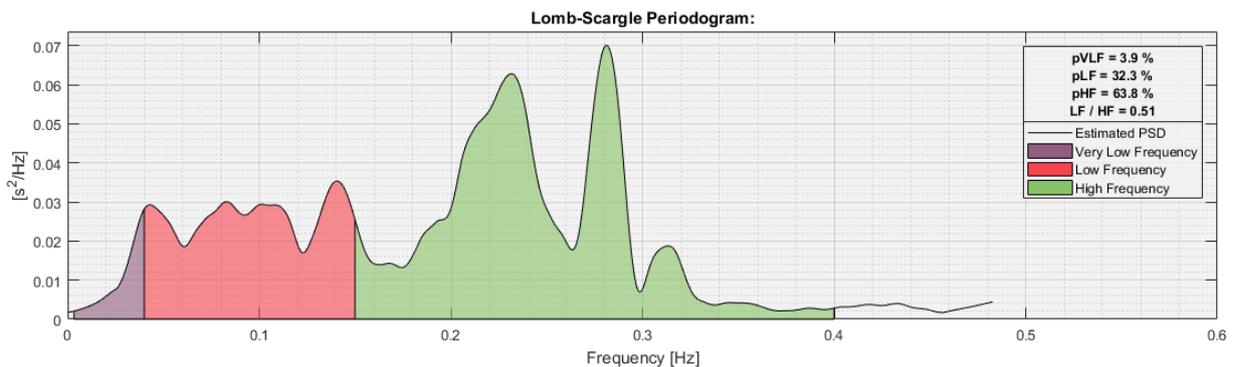


Figure 33: A Power Spectral Density estimate computed using the Lomb-Scargle periodogram method. The Very Low, Low and High Frequency bands are visualized in purple, red and green, respectively.

Non-Linear Analysis

The HRV module generates a Poincaré plot for the non-linear analysis of HRV. A Poincaré plot is a scatter plot where each IBI (IBI_n) is plotted against the subsequent IBI (IBI_{n+1}), with the former and latter representing the horizontal and vertical axes, respectively. The Poincaré plot refers to IBI_n and IBI_{n+1} as RR_n and RR_{n+1} in order to better comply with convention.

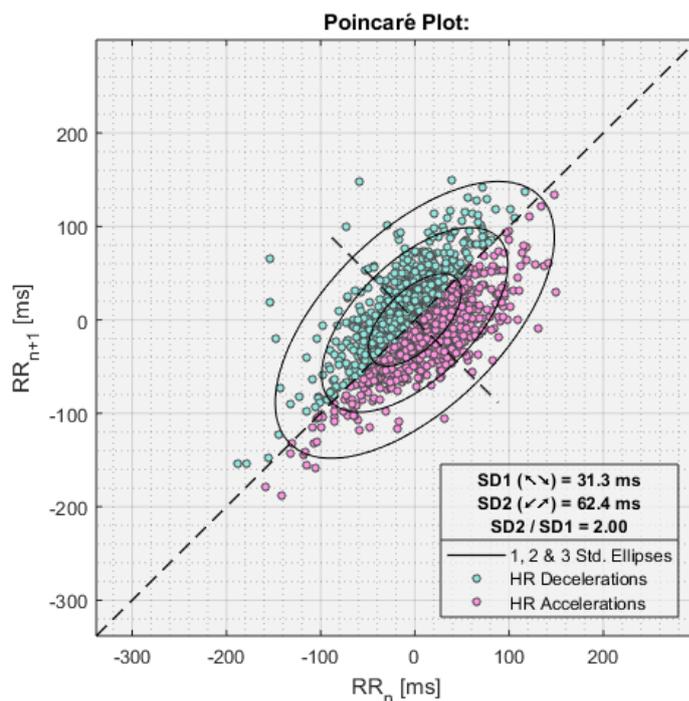


Figure 34: A Poincaré plot showing IBI_n vs IBI_{n+1} points. Heart rate accelerations are plotted in pink, while decelerations are plotted in aquamarine. The ellipses represent 1, 2 and 3 times SD1 and SD2; i.e., the spread perpendicular to and along the identity line.

8.2 Settings

See the description in the Generic Signal Analyzer section for details on the ‘Analyzer prefix (tag)’ and ‘Generate Epochs from’ settings.

The HRV must be linked to an ECG module in order for it to function. This is done by filling in the tag of the ECG module in the ‘Tag of the ECG Analyzer’ field.

Global Detrending Settings

The IBI time series can be detrended using the smoothing priors approach, which requires a lambda parameter (see Footnote 1 on page **Error! Bookmark not defined.** and Figure 36). The higher the lambda the smoother the trend baseline will appear.

- **Smoothing priors detrending:**
 - Enables the smoothness priors detrending and sets its lambda parameter.

PSD Estimation Settings

Once the PSD is estimated using the Lomb-Scargle method, the VLF, LF and HF power bands are isolated. The module will give an error if the epoch does not contain enough IBIs to compute the PSD across all bands.

The settings below indicate the start and end frequencies of the three bands:

- **VLF power band [Hz]:**
 - The start of the band must be greater than 0 Hz.
- **LF power band [Hz]:**
 - The start of the band must be greater than or equal to the end of the VLF band.
- **HF power band [Hz]:**
 - The start of the band must be greater than or equal to the end of the LF band.

Additionally, the module can smooth the PSD using a local quadratic polynomial fit, with a user-adjustable span, specified in Hertz.

- **LS Periodogram smoothing [Hz]:**
 - Enables PSD smoothing and sets the span.

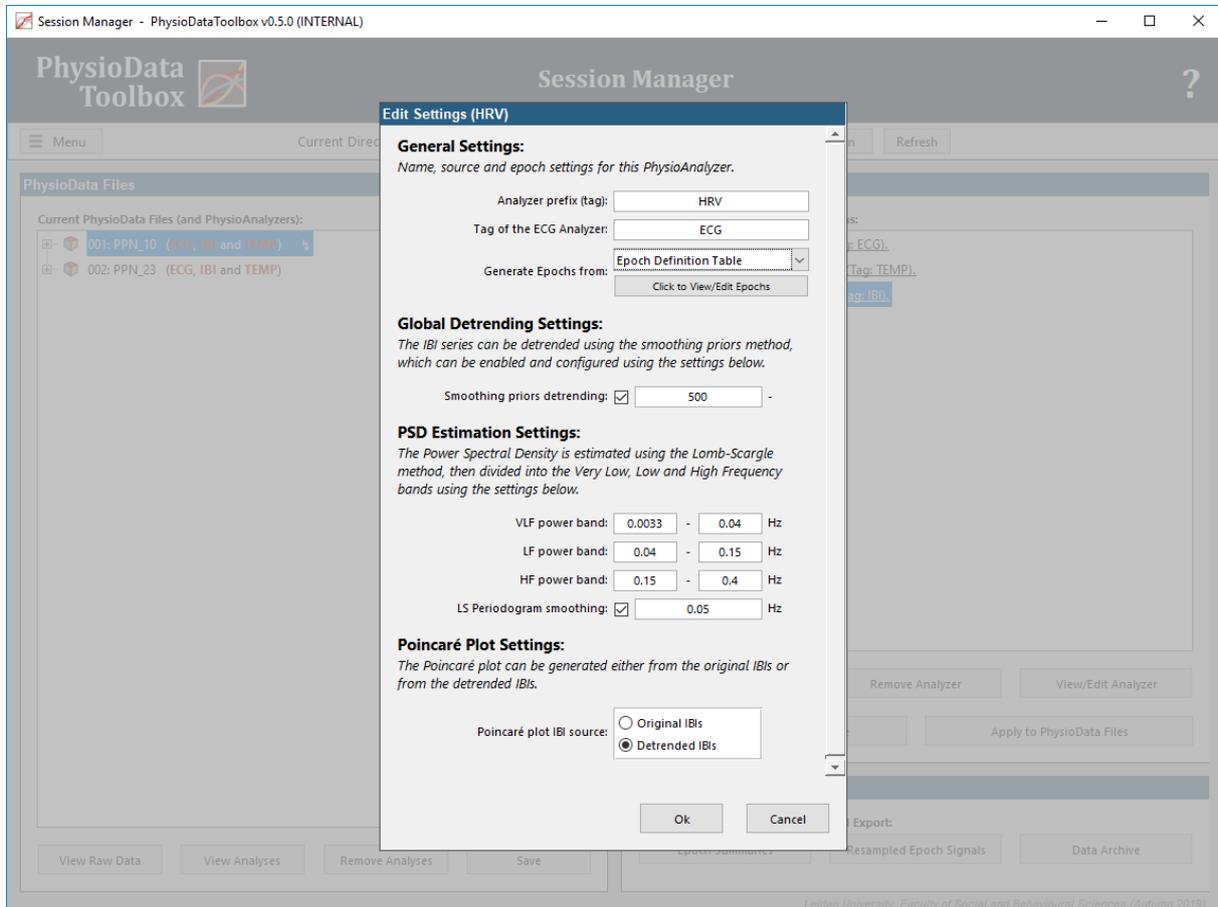


Figure 35: The HRV module's settings.

Poincaré Plot Settings

The Poincaré plot can be generated from either the original IBIs or the detrended IBIs.

- **Poincaré plot IBI source:**
 - Sets which IBIs to use when generating the Poincaré plot.

8.3 Metrics

The HRV module outputs the time-domain, frequency-domain and non-linear metrics listed in Table 4.

Table 4: The HRV Analyzer Metrics

Code:	Unit:	Description
timeDomain_error	-	Time domain analysis errors.
RMSSD_detrended	ms	The Root Mean Squared of the Successive Differences between contiguous IBIs, as computed from the detrended IBIs.
IBI_std_detrended	ms	Standard Deviation of the accepted discrete IBI data points (SD normalized by (N-1), where N is the sample size), as computed from the detrended IBIs
pNN20_detrended	%	Percentage of absolute differences between successive IBIs that are greater than 20 ms, as computed from the detrended IBIs.
pNN50_detrended	%	Percentage of absolute differences between successive IBIs that are greater than 50 ms, as computed from the detrended IBIs.
LS_error	-	Lomb-Scargle periodogram calculation errors.
VLF_powerPercent	%	The power of the Very Low Frequency band, calculated using the Lomb-Scargle method, as a percentage of the sum of the VLF, LF and HF powers.
VLF_power	ms ²	The absolute power of the Very Low Frequency band, as calculated using the Lomb-Scargle method.
VLF_psdPeak	s ² /Hz	The highest power spectral density peak in the Very Low Frequency band. Empty values may indicate that the band did not feature any peaks.
VLF_psdPeakFreq	Hz	The frequency at the highest power spectral density peak inside the Very Low Frequency band. Empty values may indicate that the band did not feature any peaks.
LF_powerPercent	%	The power of the Low Frequency band, calculated using the Lomb-Scargle method, as a percentage of the sum of the VLF, LF and HF powers.
LF_power	ms ²	The absolute power of the Low Frequency band, as calculated using the Lomb-Scargle method.
LF_psdPeak	s ² /Hz	The highest power spectral density peak in the Low Frequency band. Empty values may indicate that the band did not feature any peaks.
LF_psdPeakFreq	Hz	The frequency at the highest power spectral density peak inside the Low Frequency band. Empty values may indicate that the band did not feature any peaks.

HF_powerPercent	%	The power of the High Frequency band, calculated using the Lomb-Scargle method, as a percentage of the sum of the VLF, LF and HF powers.
HF_power	ms ²	The absolute power of the High Frequency band, as calculated using the Lomb-Scargle method.
HF_psdPeak	s ² /Hz	The highest power spectral density peak in the High Frequency band. Empty values may indicate that the band did not feature any peaks.
HF_psdPeakFreq	Hz	The frequency at the highest power spectral density peak inside the High Frequency band. Empty values may indicate that the band did not feature any peaks.
LF_HF_ratio	-	The Low Frequency power divided by the High Frequency power.
poincare_error	-	Poincare calculation errors.
poincare_SD1	ms	The poincare SD1: the standard deviation of RR _n against RR _{n+1} in the direction perpendicular to the diagonal RR _n = RR _{n+1} line.
poincare_SD2	ms	The poincare SD2: the standard deviation of RR _n against RR _{n+1} , in the direction of the diagonal RR _n = RR _{n+1} line.
poincare_SD2_SD1_ratio	-	The poincare SD2 divided by the poincare SD1.
HR_mean	BPM	Mean of the continuous Heart Rate, as interpolated from the accepted IBI data points.
R_peakCount	count	Count of accepted R-peaks inside current epoch.
IBI_mean	s	Arithmetic mean of the accepted discrete IBIs. IBI values are defined as: IBI(n) = Rt(n)-Rt(n-1), and are timestamped using: IBI _t (n) = Rt(n).
IBI_min	s	Min value of the accepted discrete IBI data points
IBI_max	s	Max value of the accepted discrete IBI data points
IBI_std	s	Standard Deviation of the accepted discrete IBI data points (SD normalized by (N-1) , where N is the sample size)
IBI_count	count	Count of accepted IBI data points inside current epoch.
IBI_coverage	%	The percentage-wise IBI coverage of the epoch; i.e., 100*(sum of the IBIs)/(Epoch Duration).
HRV_ssdCount	count	Count of successive IBIs inside current epoch. Nonadjacent IBIs are not considered successive.

8.4 Resampled Signals

No resampled signal can be extracted. The resampled instantaneous HR signal can be extracted using the ECG Signal Analyzer.

8.5 User Interaction and Data Correction

The time-domain, frequency-domain, and non-linear results are visualized for the current epoch in the GUI's two bottom panels. Users can navigate to another epoch by selecting it from the dropdown menu in the 'Epoch Data and Results' panel, or by clicking the buttons next to it. Additionally, epochs can be selected by clicking on the epoch rectangles in the epoch graph.

The HRV module itself does not allow users to correct artifacts, this must instead be done in the linked ECG module. Any changes made to the IBIs in the linked ECG module are automatically transmitted to the HRV module, and the graphs and results table are automatically updated.

8.6 Data Pipeline

The data processing and analysis pipeline used by the HRV module is described below:

Step 1: IBI sourcing and Global Detrending

The corrected IBIs from the linked ECG module are retrieved. If enabled by the user, the IBI trend is then calculated by first resampling the IBIs at 4 Hz, and then using the smoothness priors approach¹ and the user-specified lambda value to compute the trend baseline.

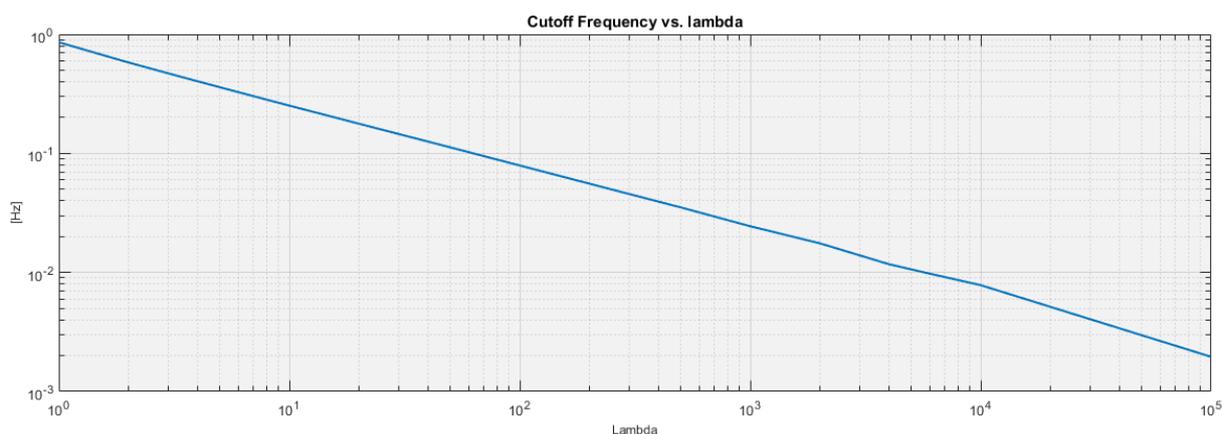


Figure 36: The high-pass cutoff frequency of the smoothness priors detrending filter as a function of lambda (λ), as computed for a 10 minute epoch.

Step 2: Epoch Segmentation and Detrending

The epoch-specific IBIs are isolated by copying all IBIs and rejecting those located outside of the current epoch, with the 'IBI location' for a given IBI event being defined as the timestamp of the R-peak that ends that interbeat interval.

If smoothness priors detrending is enabled by the user, the epoch-specific IBIs are detrended by subtracting from each IBI its corresponding trend baseline value, as visualized by the red line in the IBI graphs in Figure 31. The epoch-specific IBIs are then also linearly detrended. This is done subsequent to the smoothness priors detrending, or in absence of it if that is disabled.

All subsequent steps are performed on the epoch-specific detrended IBIs unless otherwise specified.

¹ Tarvainen, M. P., Ranta-Aho, P. O., & Karjalainen, P. A. (2002). An advanced detrending method with application to HRV analysis. *IEEE Transactions on Biomedical Engineering*, 49, 172-175. doi: 10.1109/10.979357

Step 3: Calculating Time-Domain Metrics

First, the standard deviation of the IBIs is calculated, which gets reported as the 'IBI_std_detrended' metric. Because the detrended IBIs are used, this standard deviation should be less than the standard deviation calculated by the ECG module, which uses the actual non-detrended IBIs.

Subsequently, the detrended IBIs that are 'contiguous'—i.e., directly adjacent to each other—are identified and used to calculate the successive differences. The root mean squared of these differences is then calculated and reported as 'RMSSD_detrended'. This metric is also calculated by the ECG Signal Analyzer module, but from the non-detrended IBIs. However, since RMSSD is not strongly influenced by slower IBI oscillations, the RMSSDs reported by both modules should not differ substantially.

Additionally, the fraction of absolute successive differences that exceed 20 and 50 ms are calculated and reported as a percentage in 'pNN20_detrended' and 'pNN50_detrended' respectively.

Step 4: Calculating Frequency-Domain Metrics

First, the PSD of the detrended IBI time-series is estimated using MATLAB's implementation of the Lomb-Scargle periodogram and a frequency resolution of 0.0001 Hz. An error is thrown if the epoch does not feature enough IBIs to resolve the required frequency range. If enabled in the module's settings, the PSD is then smoothed using a 'loess' filter and the specified span. Any negative PSD values that may result from the smoothing are set to 0.

The band powers are then calculated from the smoothed PSD for the following ranges, adjustable in the module's settings:

- **Very Low Frequency (VLF):** 0.0033 – 0.04 Hz.²
- **Low Frequency (LF):** 0.04 – 0.15 Hz.
- **High Frequency (HF):** 0.15 – 0.4 Hz.

The absolute powers are reported in ms^2 as the metric 'XF_power', where XF is either VLF, LF or HF. Additionally, the percentages of each power band as a fraction of the total power is reported in the XF_powerPercent metric.

For each of the VLF, LF, and HF bands, the highest peak is detected and its value and frequency are logged. These data are exported as XF_psdPeak and XF_psdPeakFreq, respectively. No data is exported for a certain band if that band does not contain a peak.

Step 5: Poincaré Analysis

To generate the Poincaré plot, all the epoch-specific IBIs that have a directly subsequent IBI are identified. These IBIs are labeled IBI_n and their subsequent IBIs are labeled IBI_{n+1} . The Poincaré plot is then generated by plotting IBI_n against IBI_{n+1} , with the former and latter representing the horizontal (X) and vertical (Y) axes, respectively.

A diagonal identity line ($X = Y$) is then generated, and the standard deviations of the Poincaré data-points is calculated in the direction of the identity line (SD2) and in the direction perpendicular to it (SD1).

Depending on the module's 'Poincaré plot IBI source' setting, either the original IBIs or the detrended IBIs are used.

² By default, the detrending lambda is set to 500, which equals an approximate high-pass cutoff frequency of 0.04 Hz given the resampling frequency of 4 Hz. This strongly attenuates the VLF power so, analyzing sub 0.04 Hz frequencies may require that either the smoothness priors detrending be disabled or that the lambda be increased.

9 ICG Ensemble Analyzer (Manual Scoring)

The ICG Ensemble Analyzer (Manual Scoring) module performs epoch-based analysis of ensemble ICG and ECG signals, and allows the user to manually score the Q, B, C, and X points on the ensembles. In addition, the user can exclude deviant waveforms from the ensemble. **Waveforms** are sections of the ICG and ECG signals that straddle each R-peak, and the **ensemble** is the average curve of the individual waveforms, see Figure 37.

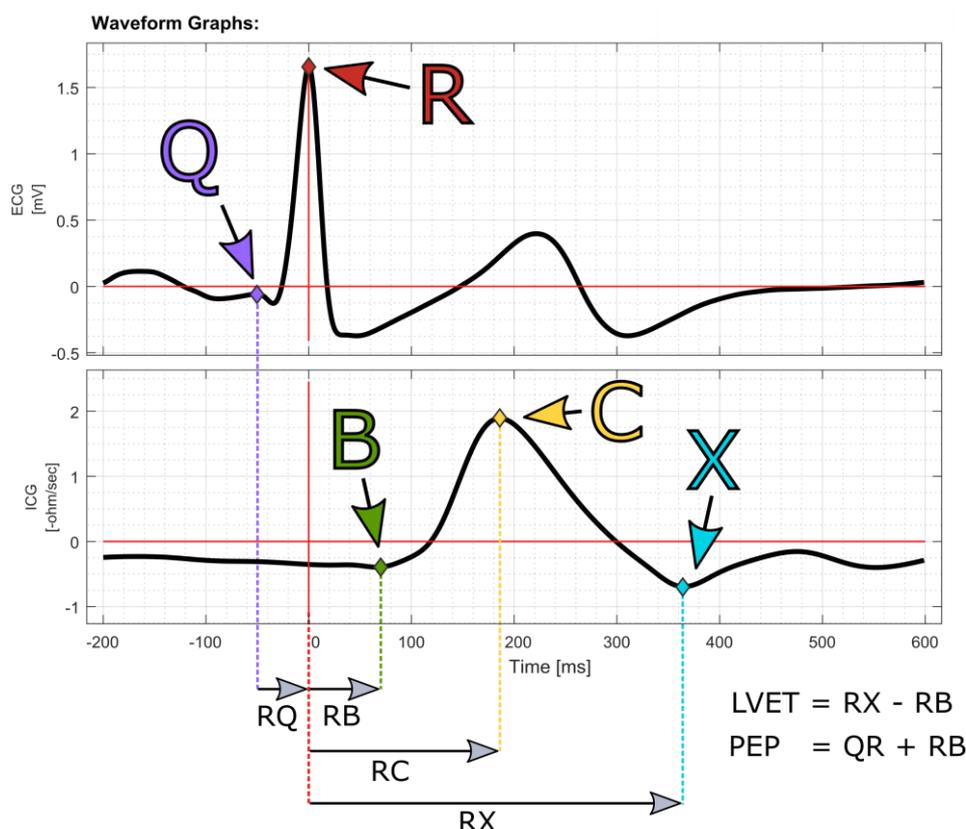


Figure 37: The ECG (top) and ICG (bottom) ensembles, and their landmarks. The R and C points are automatically detected, whereas the Q, B and X points must be manually scored by the user.

To ensemble the ICG and ECG signals, a valid ECG Signal Analyzer module must be present in the PhysioData file. The R-peak times from this ECG module, as well as the user specified waveform start and end offsets, are used to slice the ICG and ECG signals into individual waveforms. Only waveforms that completely fit inside one or more epochs are further processed. The filtered ECG signal from the ECG module is used for the ECG waveforms, and the ICG signal is taken from the user specified channel, and optionally pre-processed.

The ICG Ensemble Analyzer only analyzes the ICG ($-dZ/dt$) signal, the Generic Signal Analyzer module can be used to calculate descriptive statistics from the Z-signal; e.g., mean Z value per epoch. Similarly, ECG based metrics are only calculated in the ECG module, not the ICG module itself. For convenience and compatibility, consider defining the necessary epochs in the ECG module, then referring to them in the ICG and Generic Signal Analyzers. The toolbox cannot generate a $-dZ/dt$ signal from a Z signal, but this can be done beforehand using a custom MATLAB script.

The waveform rejection criteria, and the placement strategy for the Q, B, C and X points are at the researchers' own discretion, and fall outside of the scope of this document. Because the Q-point is used to mark the onset of the PEP, 'R-onset PEP' can technically be calculated by placing the Q-point at the R onset.



Figure 38: The ICG Ensemble Analyzer (Manual Scoring) GUI, showing: the ICG signal and the R-peak times (top graph, labeled dZ/dt); the current epoch data (leftmost table); the ensemble ECG and ICG curves, as well as the individual waveforms (two center graphs); and, the individual waveform data (rightmost table). Note that the first waveform in the table can be excluded from the ensemble by unchecking the ‘Use Waveform’ box. The rejected waveform is visualized by a triangle in the Wave-Skip graph, above its R-peak location.

9.1 Settings

The ICG Ensemble Analyzer requires its own tag, and the tag of the ECG Signal Analyzer from which the R-peak times and the filtered ECG signal are to be used. Epochs can be copied from another module, or defined using the Epoch Builder, see the ‘Epochs and Events’ section for details.

See the ‘Generic Signal Analyzer’ section for details on the other General and Preprocessing Settings.

NOTE: The ICG Ensemble Analyzer assumes that the unit of the ICG signal is $-\Omega/s$. If this is not the case, fill in an appropriate gain value (multiplier) to transform the signal into $-\Omega/s$.

Epoch Ensembling Settings

The ICG Ensemble Analyzer features the following module specific settings, which in this version of the toolbox are hardcoded and cannot be changed:

- **Waveform Start [s]:**
 - The start of the waveforms, relative to the R-peak times. This is also the start time of the ensemble signal. This value is currently set at -0.2 s.
- **Waveform End [s]:**
 - The end of the waveforms, relative to the R-peak times. This is also the end time of the ensemble signal. This value is currently set at 0.6 s.

9.2 Metrics

The ICG Ensemble Analyzer extracts metrics from the ensemble ECG and ICG curves, and the user scored landmarks.

Table 5: ICG Ensemble Analyzer Metrics

Code:	Unit:	Description
RQ	ms	The location of the ensemble Q-point in time, relative to the R-peak, which is located at $t=0$. This metric is only available if the user scored the Q-point.
Q_Amp	mV	The amplitude of the ensemble ECG signal at the Q-point. This metric is only available if the user scored the Q-point.
RB	ms	The location of the ensemble B-point in time, relative to the R-peak. This metric is only available if the user scored the B-point.
B_Amp	$-\Omega/s$	The amplitude of the ensemble ICG signal at the B-point. This metric is only available if the user scored the B-point.
RC	ms	The location of the ensemble C-point in time, relative to the R-peak.
C_Amp	$-\Omega/s$	The amplitude of the ensemble ICG signal at the C-point.
RX	ms	The location of the ensemble X-point in time, relative to the R-peak. This metric is only available if the user scored the X-point.
X_Amp	$-\Omega/s$	The amplitude of the ensemble ICG signal at the X-point. This metric is only available if the user scored the X-point.
PEP	ms	Pre-ejection period; the time interval between the Q-point and the B-point.
LVET	ms	Left Ventricular Ejection Time; the time interval between the B-point and the X-point.
nWaveforms	count	The total number of complete waveforms inside the current epoch. This is equal to the number of R-peaks in the epoch, excluding those that are too close to the epoch-edges to produce completely enclosed waveforms.
nAcceptedWaveforms	count	The number of complete waveforms in the epoch that have not been rejected.

9.3 Resampled Signals

No resampled signal can be extracted. The resampled instantaneous HR signal can be extracted using the ECG Signal Analyzer.

9.4 User Interaction and Data Correction

With the exception of basic pre-filtering, the ICG Ensemble Analyzer module does not allow the user to modify the ICG signal. It is, however, possible to include or exclude individual waveforms from the ensembles, and to custom-place the ICG and ECG landmarks (except the R-peaks).

The ‘Epoch Data’ and ‘Waveform Data’ tables, and the ‘Waveform Graphs’ display information about the currently selected epoch, and allow that epoch’s ensemble to be scored. To change the currently selected epoch, choose another one from the dropdown menu in the ‘Epoch Data’ table, or click on the desired epoch in the epoch graph.

The filtered ECG signal and the R-peak times, and hence also the ensembles, are dependent on the linked ECG module, and are automatically updated whenever the ECG module’s data is changed.

Rejecting or Accepting Waveforms

Once a waveform is rejected, neither the ICG nor ECG waveforms corresponding to that R-peak will be used to calculate the ensemble averages of any epoch that that R-peak may be in. Rejected waves are visualized in the wave-skip graph as triangles above their R-peak times, as in Figure 37. Additionally, rejected waveforms are plotted as dashed lines in the ‘Waveform Graphs’. Individual waveforms can be rejected and included one at a time by checking or unchecking their ‘Use Waveform’ box in the ‘Waveform Data’ table.

The analyzer module calculates a deviation metric for each ICG waveform to facilitate the identification and rejection of abnormal waveforms. This metrics, the Normalized Root Mean Squared Deviation (NRMSD) is calculated as root mean squared of the deviation between each sample of a ICG waveform (ICG_i) and the corresponding sample of the ICG ensemble (\widehat{ICG})_{*i*}. The RMSD is then normalized using the range of the ensemble:

$$NRMSD = \sqrt{\frac{\sum_{i=1}^n (\widehat{ICG}_i - ICG_i)^2}{n}} \cdot (\widehat{ICG}_{max} - \widehat{ICG}_{min})$$

The “Sort” button toggles the table sort between increasing R-peak time and decreasing NRMSD.

Scoring

The Q-point can be scored by right-clicking inside the ECG waveform graph axes at the desired Q-point time, and selecting the ‘Place Q-Point at this time’ from the context menu. Similarly, the B and X points can be scored by right-clicking inside the ICG waveform graph axes. All manually scored points are automatically placed on their respective ensemble curves at the sample that is closest to the user-specified time; i.e., a sampling rate of 1000 Hz allows scoring with 1 ms accuracy.

To facilitate the scoring of the ECG and ICG landmarks, the first time derivative, acceleration, and the second time derivative, jerk, of both signals are calculated. Clicking the ‘Ensemble | Accel. + Jerk | Waves’ button toggles between showing the waveforms or the derivatives. As the derivatives are arbitrarily scaled to fit the ensemble curve range, they should only be used for visual aid when scoring.

10 Blood Pressure Analyzer

The Blood Pressure (BP) Analyzer performs standard epoch-based analyses on a BP signal. The module features automatic calibration event detection, pulse estimation, and user-correction of the systolic peak detection.

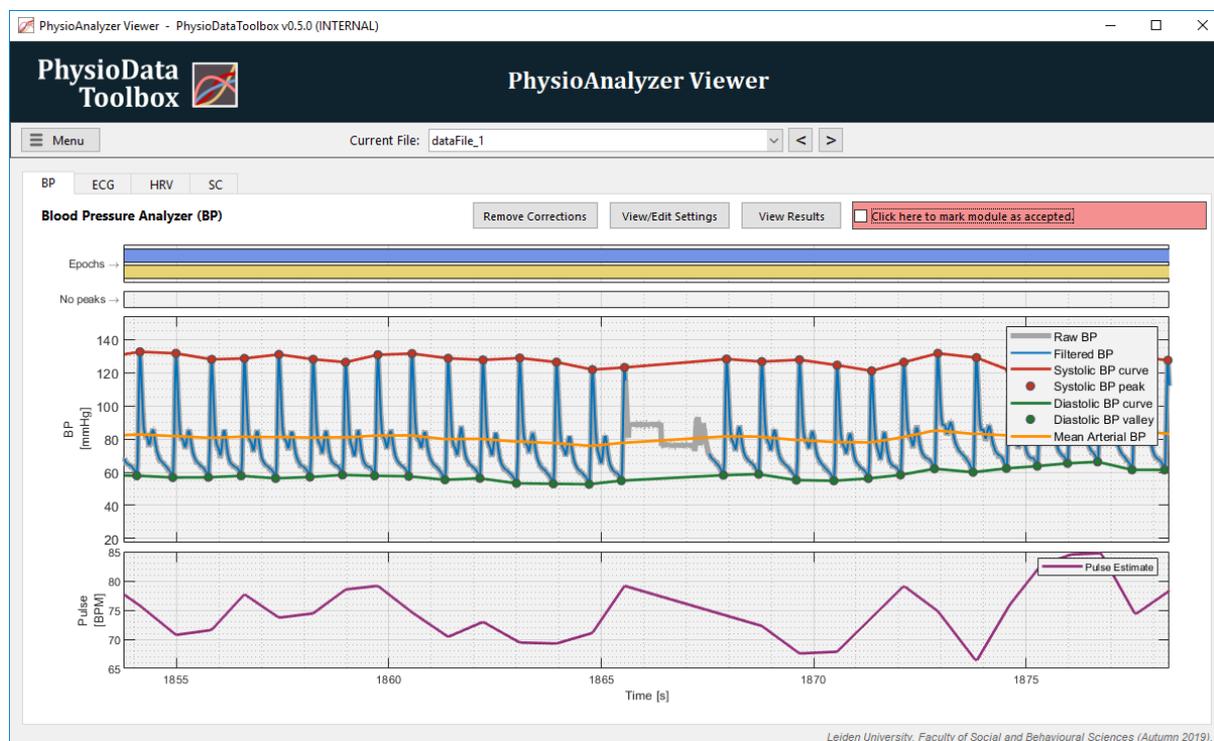


Figure 39: The BP Analyzer GUI, showing the BP signals and the estimated pulse. Note the automatically detected calibration event around 1865 seconds.

10.1 Settings

General Settings

See the description in the Generic Signal Analyzer section for details on the General and Preprocessing Settings. Note that the toolbox assumes that the BP signal has the unit mmHg, if this is not the case, use a gain to correct it.

Calibration Detection Settings

The toolbox automatically detects calibration segments in the raw signal, and removes them from the filtered signal. The user cannot modify the segments that were classified as calibration segments. In the case that the calibration fails, the user may tweak the sensitivity, or completely disable the calibration detection by setting the sensitivity to zero. A calibration sensitivity values below have been found to work well:

- **BMEYE Nexfin and Finapres systems:** 0.02 (default).
- **BIOPAC NIBP100D ('CNAP Monitor' by CN systems):** 0.002.

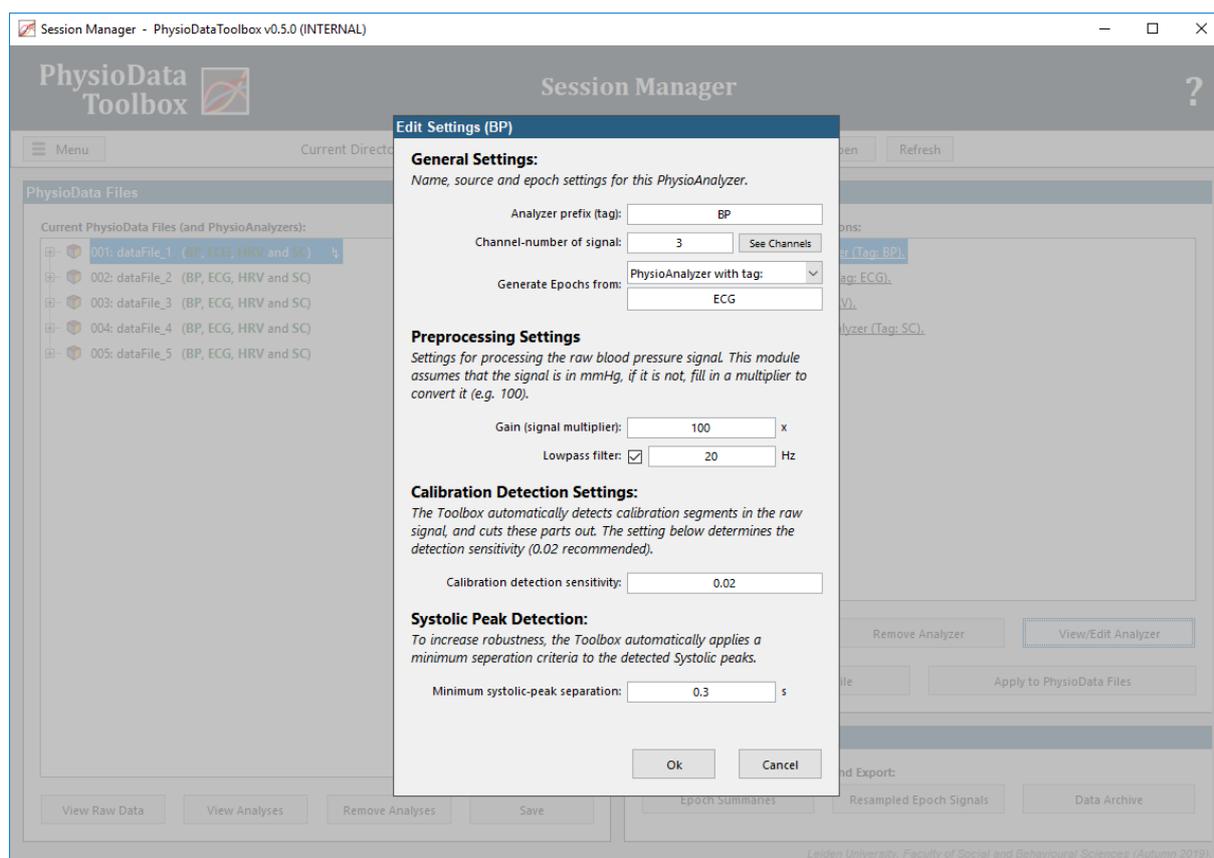


Figure 40: The BP Analyzer Settings.

10.2 Metrics

The BP analyzer detects the diastolic valleys and systolic peaks, and generates continuous diastolic blood pressure, systolic blood pressure and mean arterial pressure signals. Standard descriptive statistical analysis is then performed on these signals.

Table 6: BP Analyzer Metrics

Code:	Unit:	Description
Min_Sys_BP	mmHg	The minimum systolic blood pressure, per epoch.
Max_Sys_BP	mmHg	The maximum systolic blood pressure, per epoch.
Mean_Sys_BP	mmHg	The mean systolic blood pressure, per epoch.
Count_of_Sys_Points	-	The number of systolic peaks detected inside the epoch.
Min_Dia_BP	mmHg	The minimum diastolic blood pressure, per epoch.
Max_Dia_BP	mmHg	The maximum diastolic blood pressure, per epoch.
Mean_Dia_BP	mmHg	The mean diastolic blood pressure, per epoch.
Count_of_Dia_Points	-	The number of diastolic valleys detected inside the epoch.
Min_MAP	mmHg	The minimum MAP (Mean Arterial Pressure), per epoch.
Max_MAP	mmHg	The maximum MAP (Mean Arterial Pressure), per epoch.

Mean_MAP	mmHg	The mean MAP (Mean Arterial Pressure), per epoch.
Mean_Pulse	BPM	The mean of the interpolated pulse, as estimated from the intervals between the systolic peaks, per epoch.
Pulse_Coverage	%	The percentage of the epoch for which there is (interpolated) pulse data.
Missing_Data	%	Percentage of missing data (either due to calibrations or user rejection), per epoch.

10.3 Resampled Signals

When exporting the resampled epoch signals, the Blood Pressure Analyzer resamples and extracts the filtered BP signal, the systolic BP signal, the diastolic BP signal, and the MAP signal.

10.4 Data Correction

The Blood Pressure Analyzer allows the user to define zones in which the systolic and diastolic landmarks are disregarded, see the section for details about the peak detection pipeline.

The module estimates the pulse, in BPM, using the detected systolic peaks. The pulse signal itself cannot be corrected, and is instead automatically filtered using MATLAB's hampel filter ($k = 10$, $n\text{Sigma} = 4$). Artifacts in the pulse signal must thus be corrected indirectly by inserting correction zones in the blood pressure signal.

When a noisy section is detected, the user can insert a correction zone by left-clicking in the graph and dragging the mouse. Once a section is selected, landmarks can either be disabled or enabled. It is advisable to select a segment that is as big as possible, maximizing the interval in which no peaks should be detected.

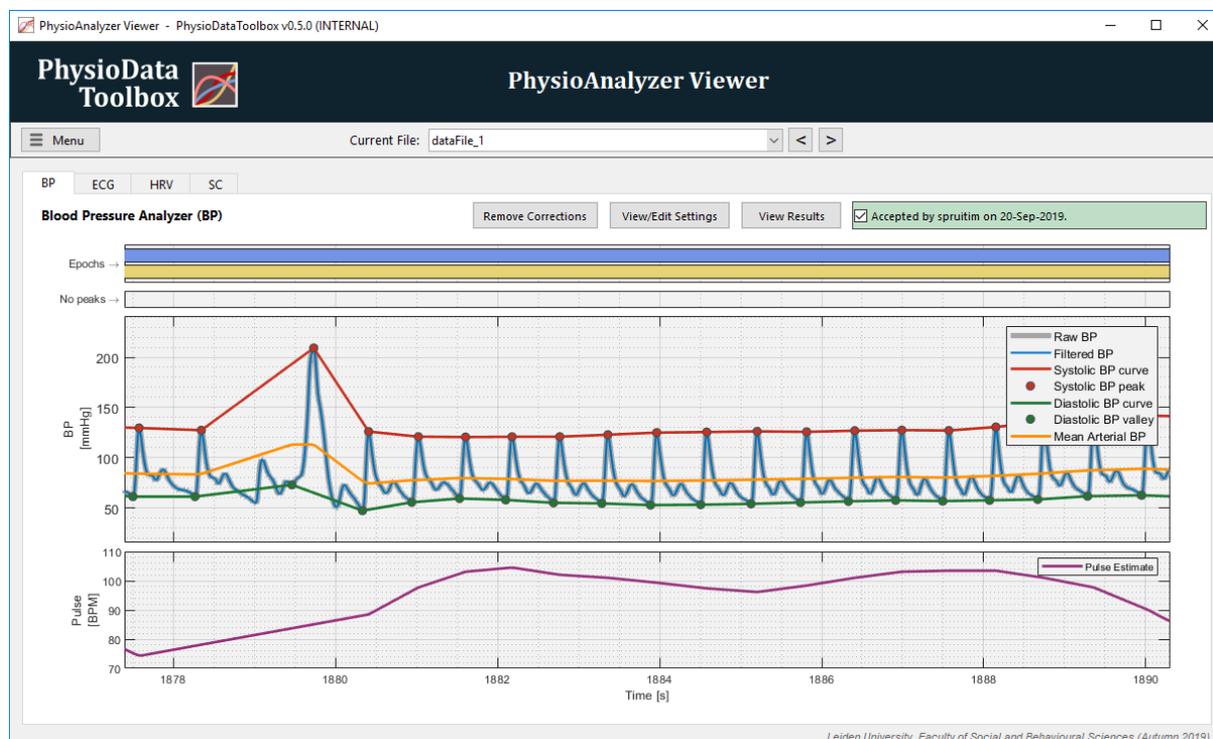


Figure 41: A blood pressure plot showing some deviating waveforms around 1880 seconds.

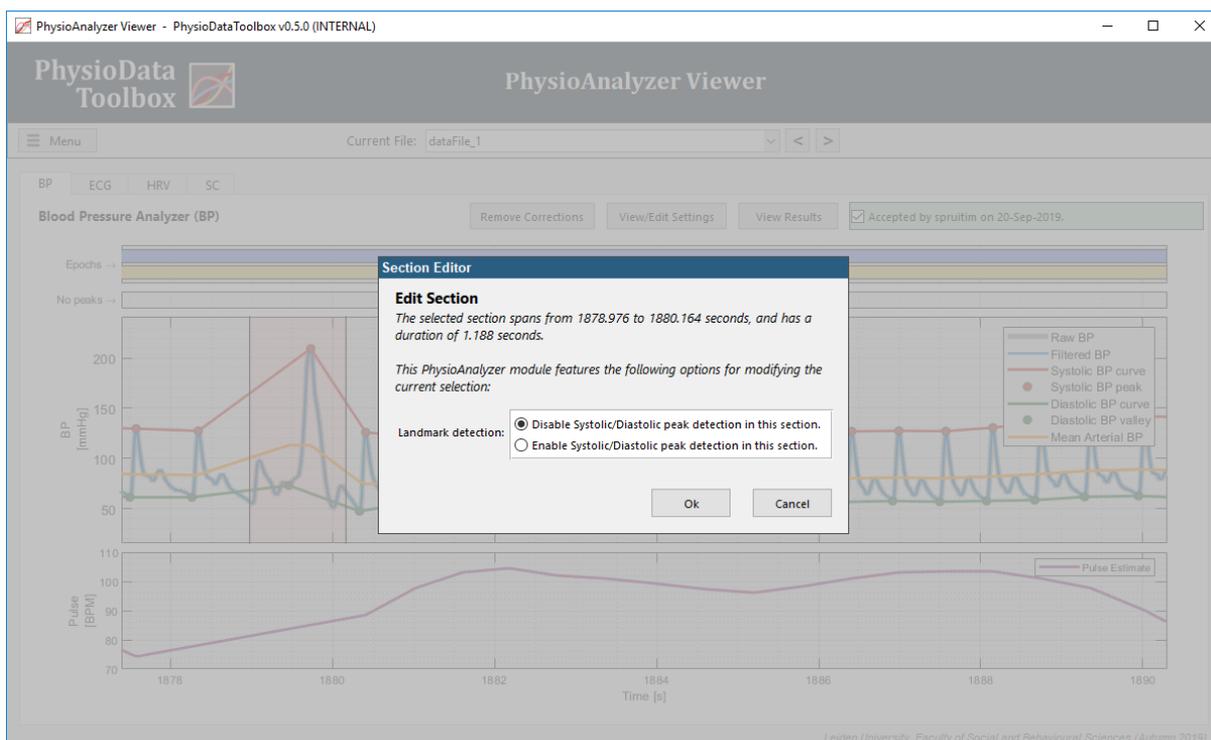


Figure 42: Once the noisy section is selected, systolic and diastolic peak detection can be enabled or disabled within that section.

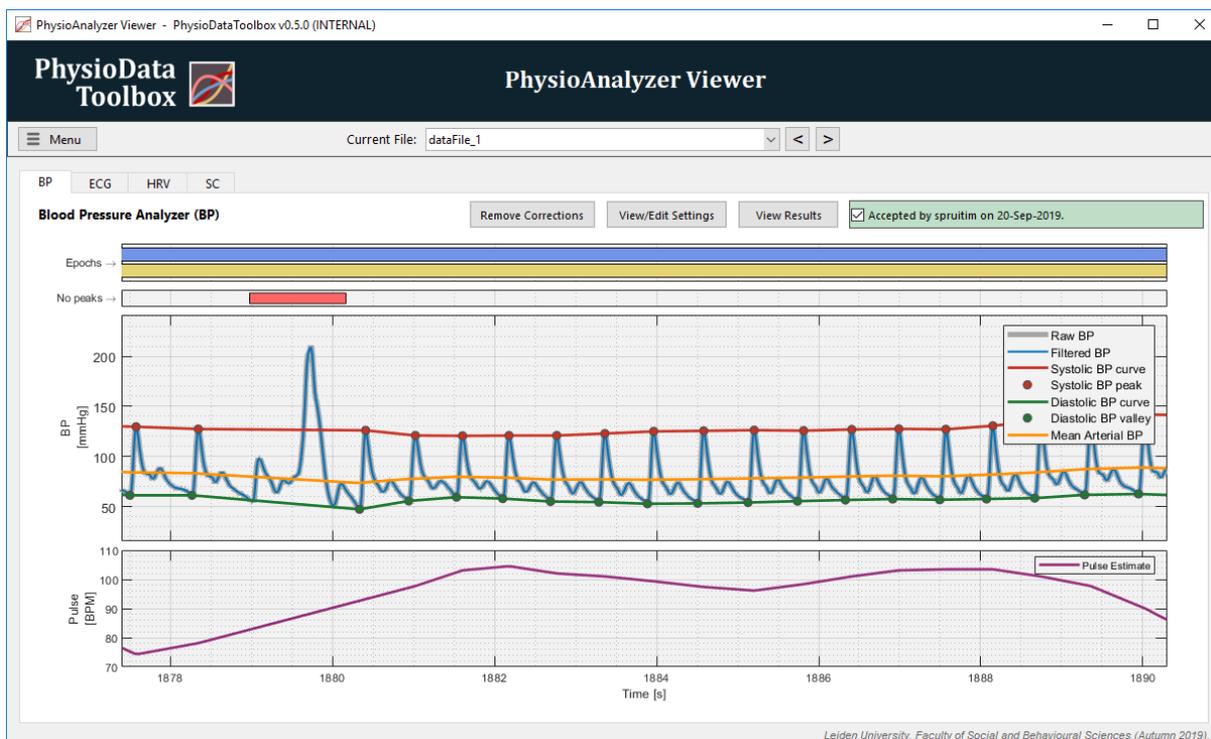


Figure 43: Disabling the peak detection within the previously selected section removes the deviation from the interpolated blood pressure signals.

11 Respiration Analyzer (Beta Version)

The Respiration Analyzer extracts respiration metrics from a chest or abdomen expansion signal generated by a force-transducer.

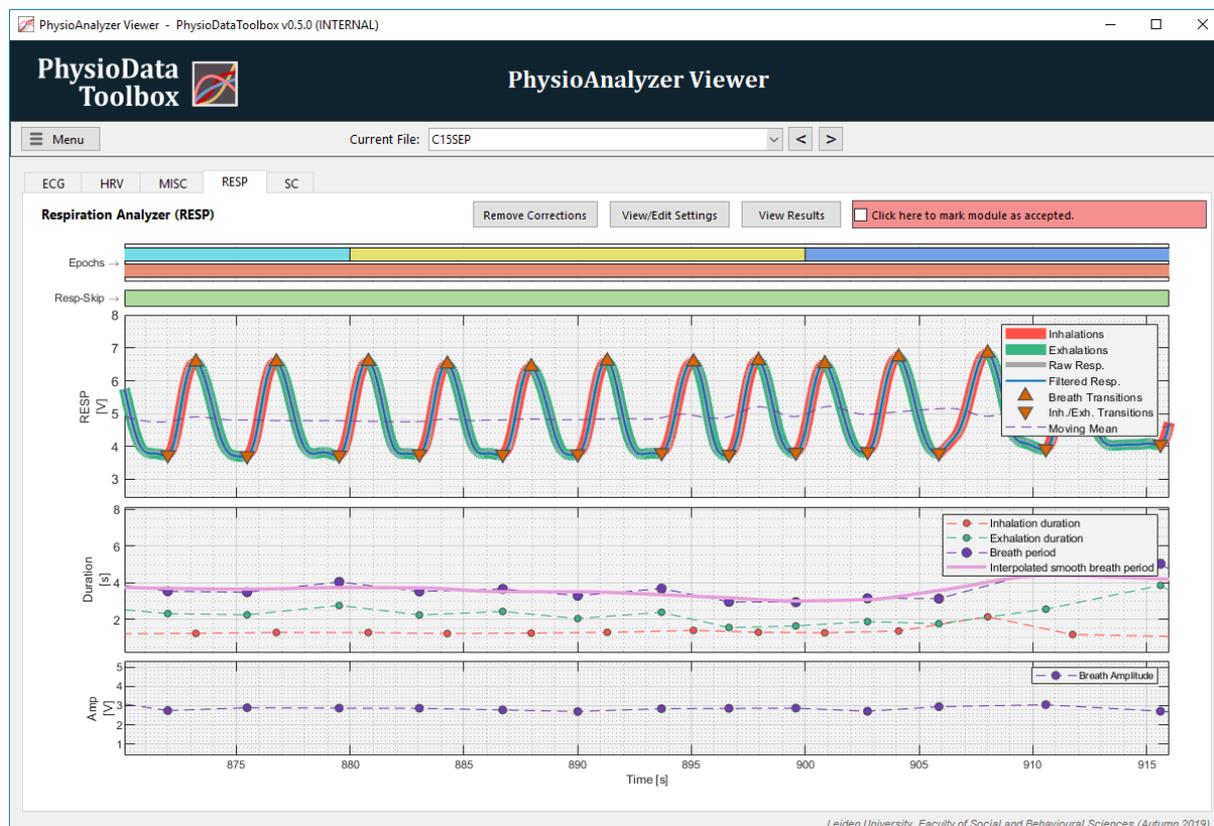


Figure 44: The Respiration Module GUI.

The respiration module defines a single breath as an inhalation followed directly by an exhalation. Breaths cannot exist without both an inhalation and an exhalation phase. Breaths, inhalations, and exhalations are timestamped at the end of their cycle.

11.1 Settings

For the General and Preprocessing Settings, see the description in the General Signal Analyzer section.

Inhalation and Exhalation Detection

Inhalation and exhalation scoring occurs by detecting intercepts between the filtered respiration signal, and its moving-mean signal. The latter is generated using a moving-mean filter with a window width that is the ‘moving-mean multiplier’ times the local breath rate, the latter of which is estimated using a Fast Fourier Transform (FFT).

Once inhalations and exhalations have been detected, outliers are removed using a moving mean and standard deviation filter, which removes inhalations and exhalations with durations and amplitudes outside of a moving mean +/- a multiple of the local standard deviation. See the Data Pipeline section for details about the data processing pipeline.

- **Moving-mean multiplier [n]:**
 - The estimated breath-rate multiplier to use when sizing the moving-mean filter window.
- **Min. and max. inhalation duration [s]:**
 - The minimum and maximum allowable inhalation duration.
- **Min. and max. exhalation duration [s]:**

- The minimum and maximum allowable exhalation duration.
- **Outlier filter threshold [-] and span [s]:**
 - The threshold, in standard deviation multiples, and the moving window span, in seconds, of the outlier filter. Set the threshold to info to turn off the filtering.
- **Inhalation/Exhalation detection:**
 - The method used to determine where an inhalation or exhalation phase starts. See ‘Data Pipeline’ section.
- **Slope gradient threshold [%]:**
 - The percentage of the maximum slope at which the onset of the inhalation or exhalation phase is set.
- **Smoothing window widths [s]:**
 - The window span of the smoothing filter used to smooth the interpolated breath duration signal.

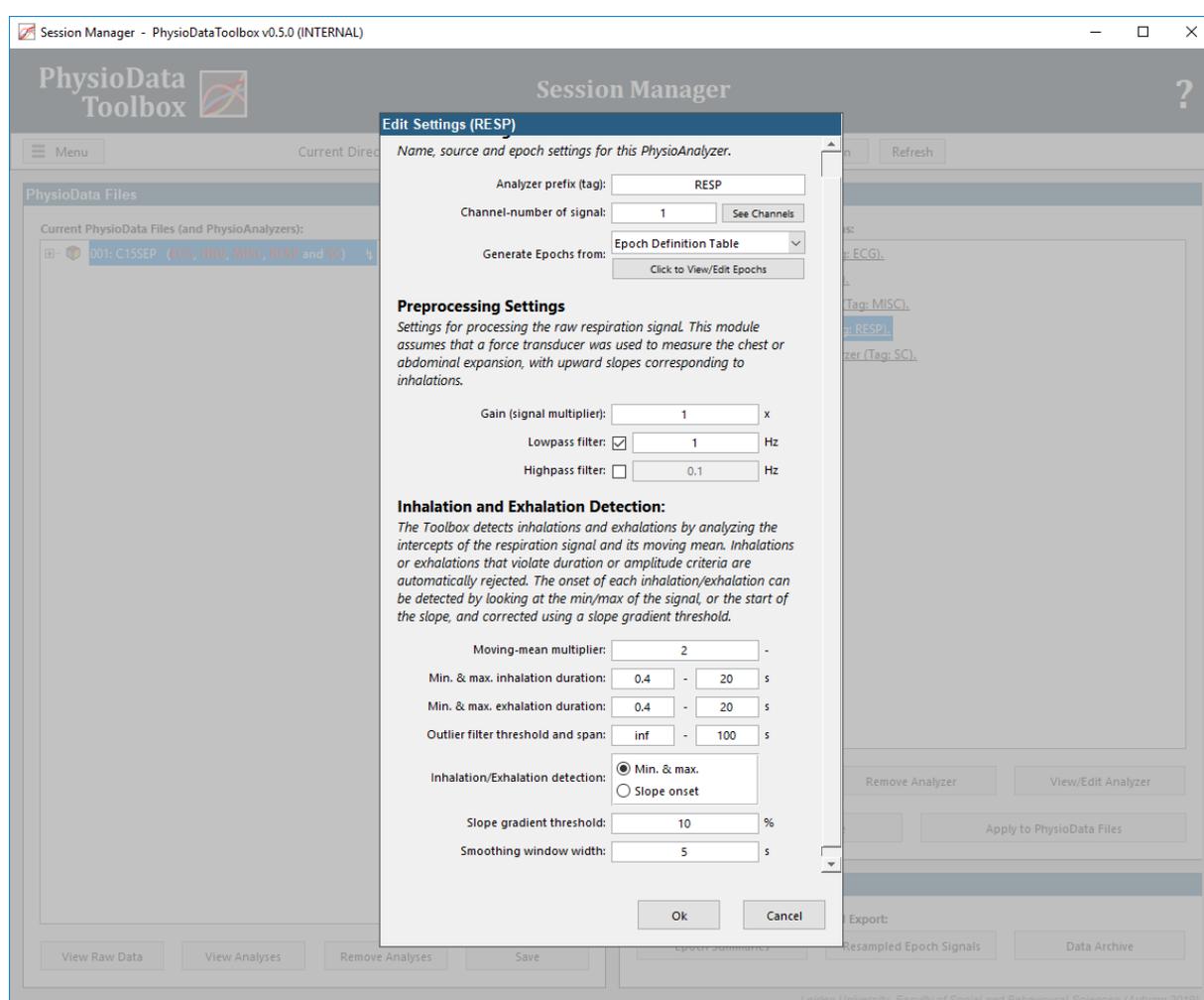


Figure 45: The Respiration Analyzer module’s settings window.

11.2 Metrics

The Respiration module calculates basic descriptive statistics for the inhalation, exhalation and breath phases, per epoch.

Table 7: The Respiration module's output metrics.

Code:	Unit:	Description
Min_Inhalation_dur	s	The minimum inhalation duration, per epoch.
Mean_Inhalation_dur	s	The mean inhalation duration, per epoch.
Median_Inhalation_dur	s	The median inhalation duration, per epoch.
Max_Inhalation_dur	s	The maximum inhalation duration, per epoch.
Std_Inhalation_dur	s	The standard deviation of the inhalation durations, per epoch.
Min_Exhalation_dur	s	The minimum exhalation duration, per epoch.
Mean_Exhalation_dur	s	The mean exhalation duration, per epoch.
Median_Exhalation_dur	s	The median exhalation duration, per epoch.
Max_Exhalation_dur	s	The maximum exhalation duration, per epoch.
Std_Exhalation_dur	s	The standard deviation of the exhalation durations, per epoch.
Min_Breath_dur	s	The minimum breath duration, per epoch.
Mean_Breath_dur	s	The mean breath duration, per epoch.
Median_Breath_dur	s	The median breath duration, per epoch.
Max_Breath_dur	s	The maximum breath duration, per epoch.
Std_Breath_dur	s	The standard deviation of the breath durations, per epoch.
Min_Breath_amp	V	The minimum breath amplitude, per epoch.
Mean_Breath_amp	V	The mean breath amplitude, per epoch.
Median_Breath_amp	V	The median breath amplitude, per epoch.
Max_Breath_amp	V	The maximum breath amplitude, per epoch.
Std_Breath_amp	V	The standard deviation of the breath amplitudes, per epoch.
fullBreathsCount	-	The number of complete breaths detected inside the epoch.
breathCoverage	%	Percentage of the epoch covered by full breaths; i.e. the total breath duration over the epoch duration, multiplied by 100.

mean_InterpBreathPeriod	s	The mean of the breath period, calculated as the mean of the smooth interpolated breath period (i.e. breath duration) signal, per epoch.
-------------------------	---	--

11.3 Resampled Signals

No resampled signal can be extracted.

11.4 User Interaction and Data Correction

The Respiration Analyzer module allows users to remove complete breath cycles by selecting a section in the main graph and selecting the ‘disregard’ option, which adds a ‘Resp-skip’ zone to the module. Any breath that overlaps any skip zones, even partially, are completely removed.

11.5 Data Pipeline

The data processing and analysis pipeline used by the Respiration module is described below. The method is based on the approach described by Lu et al. (2006)³.

Step 1: Filtering the signal

The raw respiration transducer signal is fetched from the specified channel, and multiplied by the specified gain. If enabled, the signal is then low and/or highpass filtered at the specified cutoff frequencies.

Step 2: Generating the Moving Mean Signal

Article [1] calculates a moving average signal from the respiration signal, then uses the locations of the intercepts between these signals to search for the inhalation and exhalation transitions. The article uses an FFT to estimate the respiration period (T) of the signal, then uses this to dimension the moving average window, which they give a width of 2T. Since the Toolbox does not assume steady state free-breathing, the respiration rate is instead estimated using a half-overlap moving window, across the whole signal, in order to take varying T values into account.

The signal is first resampled to 100 Hz, high-pass filtered at 0.05 Hz, and split into windows of 60 seconds, with 15 second overlaps on each side. Then, using FFT, the T is estimated per window, and that window is filtered using a moving mean filter with a window length equal to ‘T_multiplier’ times the estimated T of that window, with ‘T_multiplier’ being a user-specifiable setting.

The windows are then reassembled into a single signal, smoothed using a 1-second moving mean filter, and resampled to the original respiration signal frequency.

Step 2: Detecting the Intercepts

First, the Toolbox searches for the up and down-intercepts between of the moving mean signal produced by step 2, and the filtered respiration signal produced by step 1.

Secondly, all up-intercepts that do not follow a down-intercept, and all down-intercepts that do not follow up-intercepts are iteratively deleted until the intercept series alternates between up and down intercepts.

Thirdly, all intercept intervals that are shorter than 100 ms are removed, with intercept intervals being defined as down-intercepts times minus the corresponding up-intercept times.

Step 3: Finding Inhalation and Exhalation Onsets

Inhalation onsets are detected by searching backwards from each up-intercept, and looking for the first sample that features a positive gradient; i.e., searching for the start of the slope, on the filtered respiration

³ Lu, W., Nystrom, M. M., Parikh, P. J., Fooshee, D. R., Hubenschmidt, J. P., Bradley, J.D., & Low, D. A. (2006). A semi-automatic method for peak and valley detection in free-breathing respiratory waveforms. *Medical Physics*, 33, 3634-3636.

signal, which intercepts the moving mean signal. The onset is then placed on the first sample after the start of the slope that features a gradient larger than the specified percentage of the maximum gradient of the slope. Note that the end of the slope is defined as the last sample with a positive gradient.

If the 'max/min' method for inhalation/exhalation onset detection was chosen, the start and end of the 'slopes' are instead defined as the minimum and maximum values of the filtered respiration signals between the down-intercepts surrounding the up-intercept in question.

The exhalations are detected in the same way, albeit in inverse.

The end of each inhalation and exhalation phase is defined as occurring on the sample before the next inhalation or exhalation start.

Step 4: Defining Breaths

Using the inhalation and exhalation onsets and offsets detected in step 3, breaths are defined as starting at an inhalation onset, and ending at the subsequent exhalation offset, with an inhalation/exhalation transaction in between.

Subsequently, inhalations with a duration shorter than the specified minimum threshold are detected and merged into the surrounding exhalation. Similarly, short exhalations are merged into the surrounding inhalations.

Inhalations with amplitudes that are smaller than the local moving mean minus a multiplier times the local moving standard deviation are then also merged into the surrounding exhalations. The moving mean and moving standard deviation are calculated using the user-specified window span, and the rejection threshold is calculated using the user-specified multiplier. Inhalation and exhalation amplitudes are defined as the value of the filtered respiration signal at the end of the phase, minus that at the start of the phase. Breaths that feature inhalations that are larger than the local moving mean plus the multiplier times the local moving standard deviation are completely removed.

Next, the exhalation amplitude outliers are processed in the same way.

Breaths containing either inhalations or exhalation with durations that are larger than the specified maximum are then completely removed.

Next, breaths containing either inhalations or exhalation with durations that are larger than the local moving mean plus the multiplier times the local moving standard deviation are also completely removed.

Lastly, all breaths that overlap the user-specified 'resp-skip' zones are removed, even if they only partly overlap the skip zone.

Step 5: Generating the Smooth Breath Period Signal

A breath period is defined as the duration from the start of a breath to the end, with the breath being timestamps at the end of that breath. A smooth breath period signal is generated by smoothing the breath timeseries using the 'lowess' method, with a span being the user-specified smoothing window width.

The smoothed breath period time-series are then interpolated at 50 Hz using the 'pchip' method to form the smooth continuous breath period signal.

Step 6: Data Analysis

When analyzing the data, the minimum, mean, median, maximum and standard deviation of the inhalation, exhalation and breath durations, and the breath amplitude, are calculated for each epoch. Inhalations, exhalations and breaths are only included inside an epoch if they start and end inside the epoch.

Additionally, the total number of full breaths, and the breath coverage—which is the sum of all breath durations as a percentage of the epoch duration—are calculated.

12 EMG Signal Analyzer

The EMG Signal Analyzer rectifies and filters a raw EMG signal, and performs standard epoch-based analyses. The module allows the user to remove sections from the signal, and features two types of configurable EMG smoothing filters: a boxcar filter and a lowpass filter. The default settings are based on A. van Boxtel (2010)⁴, which tested the efficacy of various filters when applied to blink reflex research.

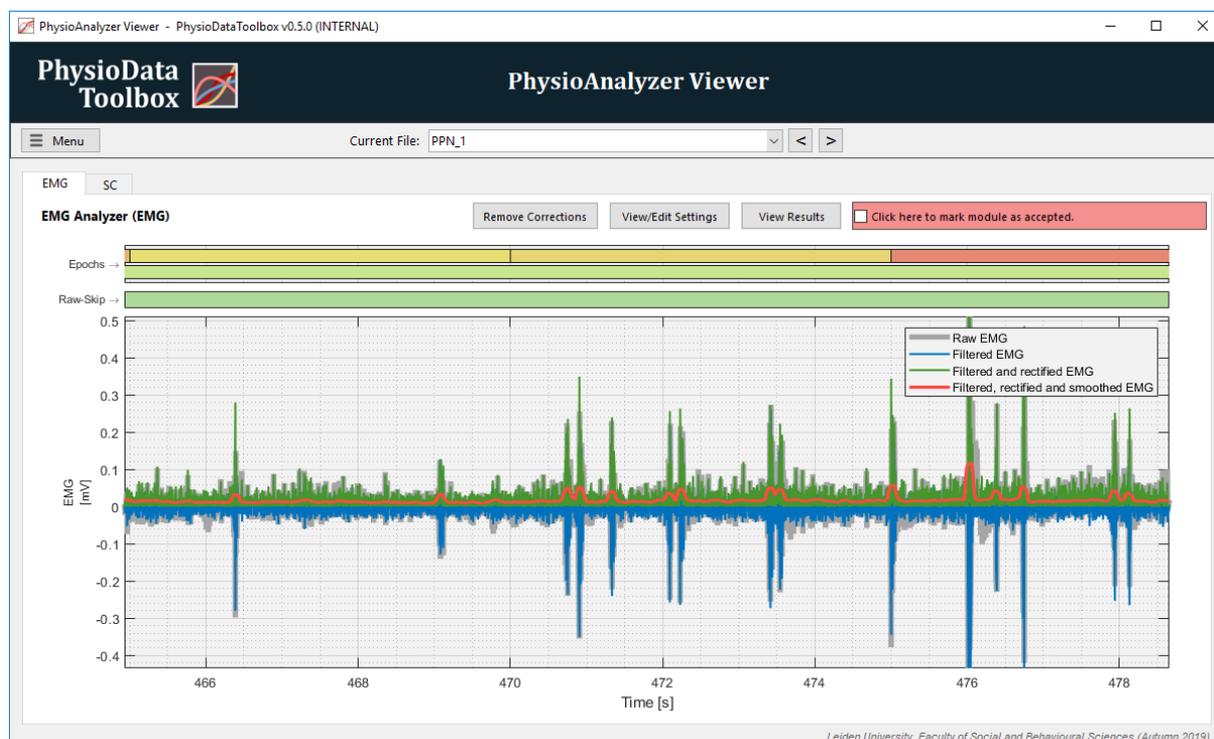


Figure 46: The EMG signal analyzer.

12.1 Settings

General Settings and Preprocessing Settings

See the description in the Generic Signal Analyzer section for details on the General and Preprocessing Settings. Note that the toolbox assumes that the EMG signal has the unit mV, if this is not the case, use a gain to correct it.

The EMG Analyzer additionally features a notch filter (`iirnotch` with $Q = 35$) that can be used to filter out line noise.

Smoothing Settings

The 'Filter-type' setting allows the user to set the smoothing filter to either a boxcar or lowpass filter. Depending on the filter choice, the 'Boxcar-size/filter-frequency' settings determines the size of the boxcar, in ms, or the lowpass cutoff frequency, in Hz.

⁴ Van Boxtel, A. (2010). Filter for optimal smoothing of acoustic and electric blink reflex EMG responses to determine blink response magnitude. *Biological Psychology*, 85, 299-305.

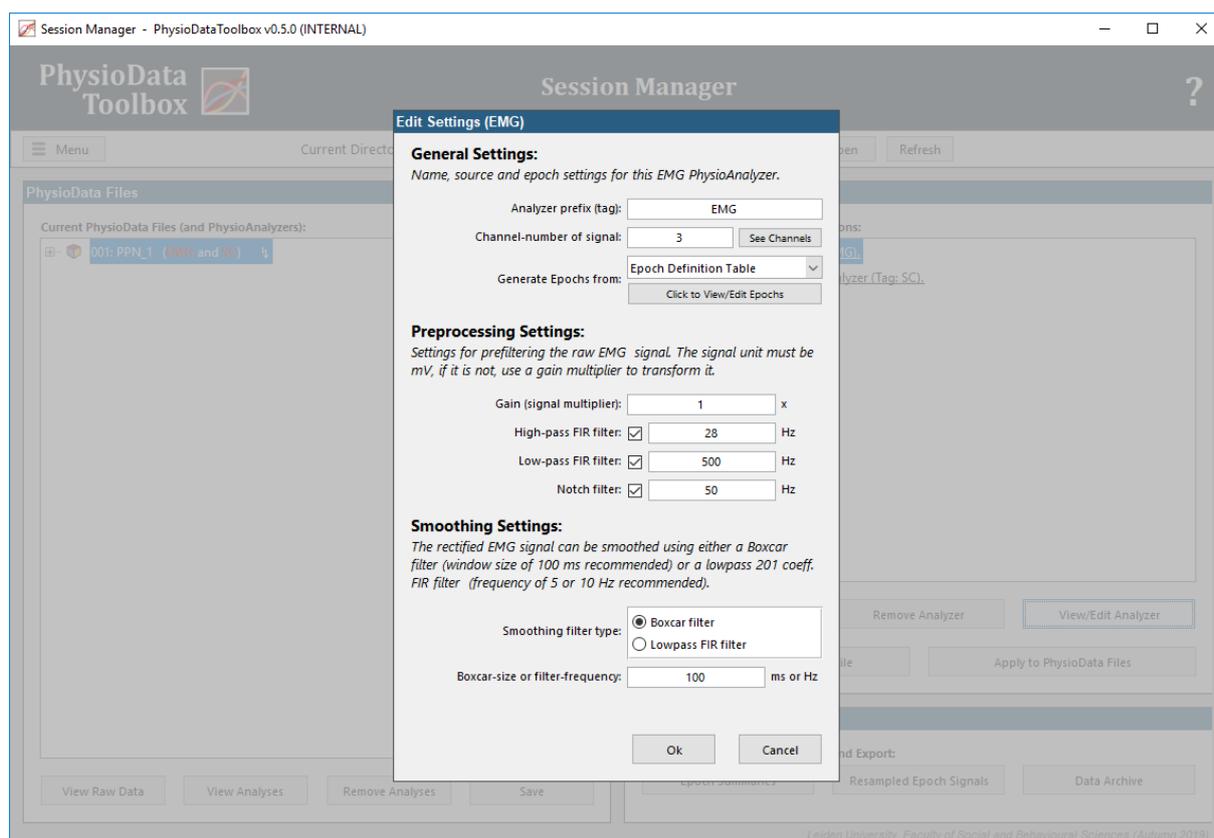


Figure 47: The BP Analyzer Settings.

12.2 Metrics

The EMG analyzer filters the raw EMG signal to create the ‘filtered EMG signal’, then rectifies and smooths this signal to create the ‘filtered, rectified and smoothed EMG signal’. Descriptive statistical analysis is performed on both these signals, per epoch.

Table 8: EMG Analyzer Metrics

Code:	Unit:	Description
Min_Smooth_EMG	mV	The minimum value of the filtered, rectified and smoothed EMG signal, per epoch.
Max_Smooth_EMG	mV	The maximum value of the filtered, rectified and smoothed EMG signal, per epoch.
Mean_Smooth_EMG	mV	The mean value of the filtered, rectified and smoothed EMG signal, per epoch.
BoundedArea_Smooth_EMG	mV	The area bounded by the filtered, rectified and smoothed EMG signal and the abscissa, per epoch.
TimeToMax_Smooth_EMG	s	The time between the start of the epoch and the point where the filtered, rectified and smoothed EMG signal reaches its maximum point in that epoch.
Min_Filt_EMG	mV	The minimum value of the filtered EMG signal, per epoch.
Max_Filt_EMG	mV	The maximum value of the filtered EMG signal, per epoch.

Mean_Filt_EMG	mV	The mean value of the filtered EMG signal, per epoch.
Min_FiltRect_EMG	mV	The minimum value of the filtered and rectified EMG signal, per epoch.
Max_FiltRect_EMG	mV	The maximum value of the filtered and rectified EMG signal, per epoch.
Mean_FiltRect_EMG	mV	The mean value of the filtered and rectified EMG signal, per epoch.
Skipped_Section_Raw_EMG	%	The percentage of the epoch that the user chose to skip; i.e. the size of the raw signal section(s) that was/were interpolate over, relative to epoch size.

12.3 Resampled Signals

When exporting the resampled epoch signals, the EMG Analyzer resamples and extracts the filtered EMG signal, the smoothed EMG signal, and the rectified EMG signal.

12.4 Data Correction

The EMG Signal Analyzer allows the user to define zones in which the raw EMG signal is not used, but is instead interpolated over. This is similar to the Generic Signal Analyzer module, see its 'Data Correction' section for details.

13 Skin Conductance Analyzer

The Skin Conductance Analyzer performs epoch-based analysis of a Skin Conductance EDA Signal. Firstly, the raw signal is low-pass filtered to extract a smooth SCL channel; then, this SCL channel is highpass-filtered to extract the phasic signal. Standard descriptive analysis is performed on the SCL signal, and the area bounded by the curve is calculated for the phasic channel. This version of the toolbox does not feature any SCR detection algorithms.

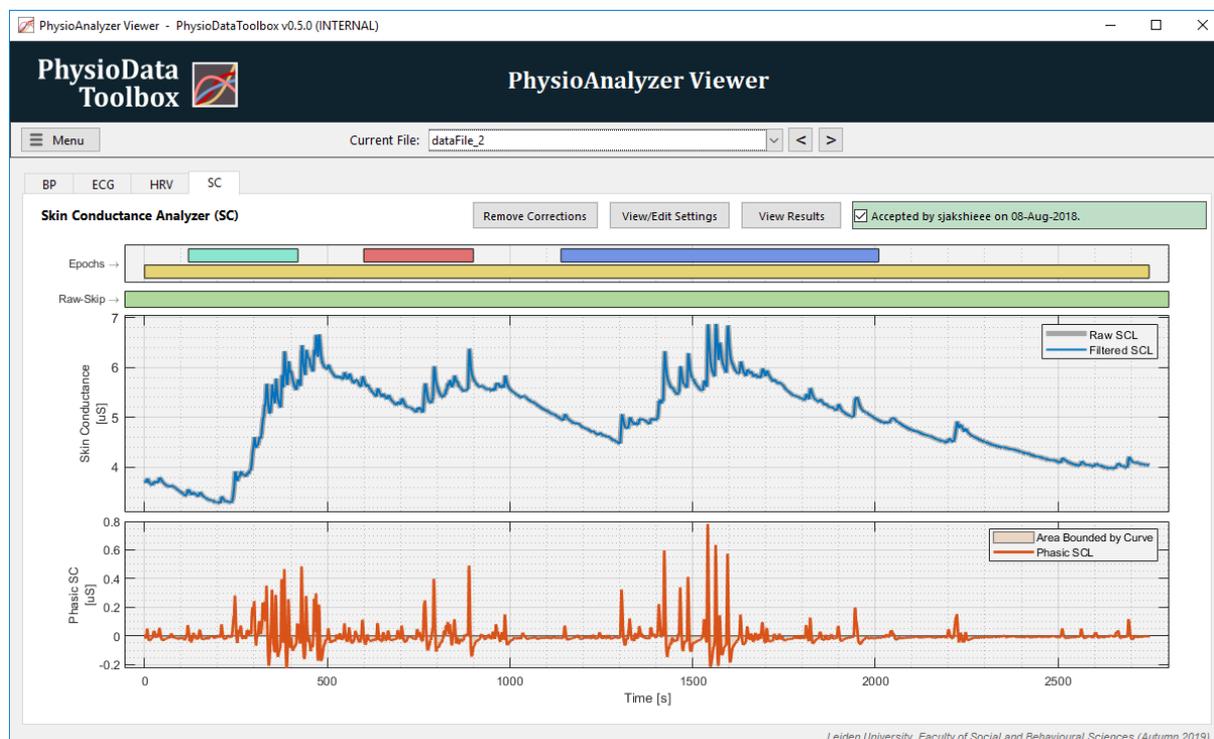


Figure 48: The Skin Conductance Analyzer GUI, showing the SCL (in axes labeled Skin Conductance) and the phasic channel (in the axes labeled Phasic SC).

13.1 Settings

For the General Settings, see the description in the General Signal Analyzer section.

Preprocessing and Filter Settings

The Skin Conductance Analyzer features the following settings:

- **Gain:**
 - The gain is the value with which the raw signal must be multiplied to transform it into μS . Because the analyzer performs amplitude analysis assuming that the unit is μS , it is imperative that the gain be correctly entered.
- **SC Lowpass filter [Hz]:**
 - The cutoff frequency used to lowpass-filter the raw signal. Uncheck to disable.
- **Phasic high-pass filter [Hz]:**
 - The cutoff frequency used to highpass-filter the low-pass filtered signal, and thereby creating the phasic channel.
- **Median filter window length [ms]:**
 - The window length of the median filter applied to the raw SCL signal before lowpass filtering. If participants underwent electrical stimulation during the data acquisition, electrical spike artifacts that cannot be removed by the lowpass filter alone may be present in the skin conductance signal. The median filter can be used to remove these types of artifacts.

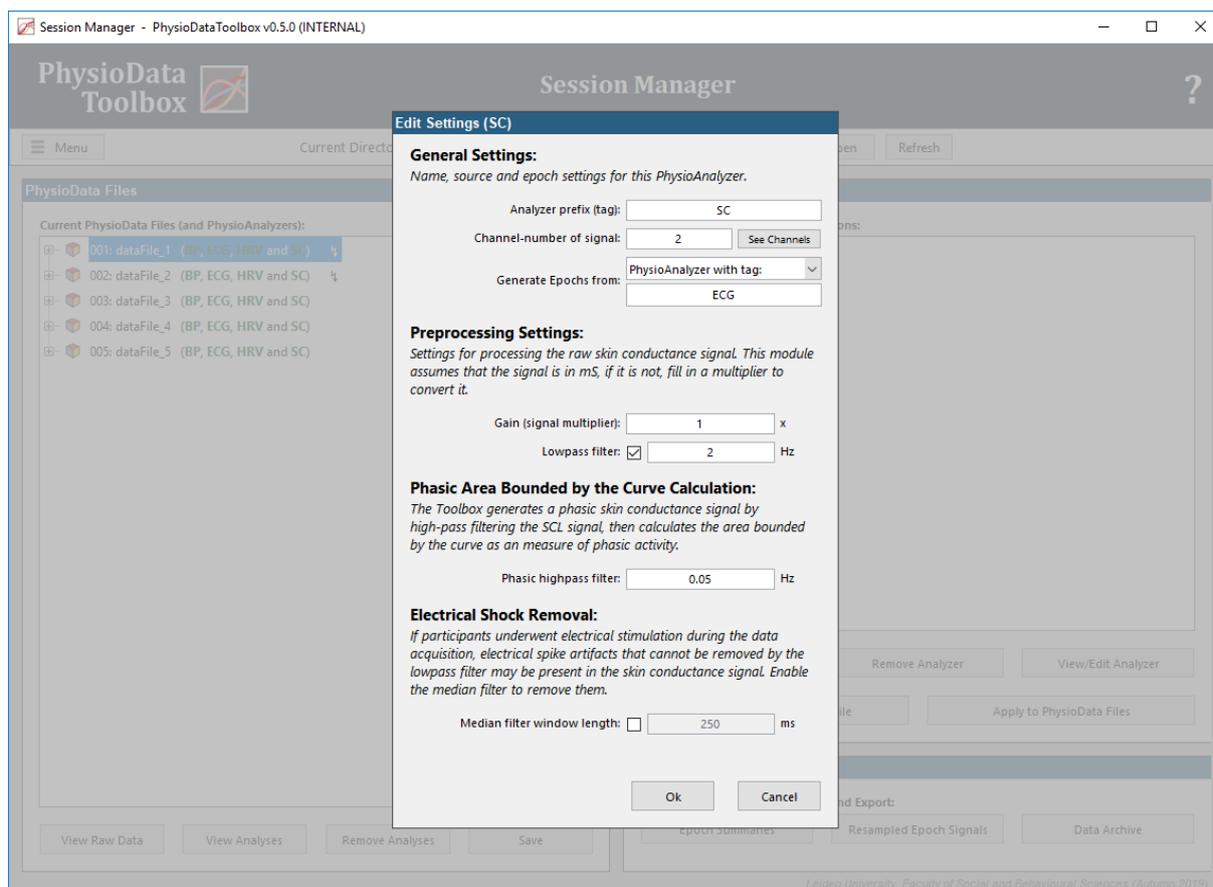


Figure 49: Skin Conductance Analyzer settings.

13.2 Metrics

The SC Signal Analyzer extracts descriptive analysis from the SCL signal, and the area bounded by the curve from the phasic channel.

Table 9: Skin Conductance Analyzer Metrics

Code:	Unit:	Description
SCL_mean	uS	The mean of the filtered SCL signal, per epoch.
SCL_min	uS	The minimum of the filtered SCL signal, per epoch.
SCL_max	uS	The maximum of the filtered SCL signal, per epoch.
SCL_std	uS	The standard deviation of the filtered SCL signal, per epoch.
Phasic_BoundedArea	uS/s	The area inside the epoch bounded by the phasic signal and abscissa, divided by the epoch duration. ⁵

13.3 Resampled Signals

When exporting the resampled epoch signals, the Skin Conductance Analyzer resamples and extracts the filtered SCL signal and the phasic SCL signal.

⁵ Figner, B., & Murphy, R. O. (2011). Using skin conductance in judgment and decision making research. In M. Schulte-Mecklenbeck, A. Kuehberger, & R. Ranyard (Eds.), A handbook of process tracing methods for decision research. New York, NY: Psychology Press.

13.4 Data Correction

The Skin Conductance Analyzer features the same correction method as the Generic Signal Analyzer; i.e., sections of the raw signal can be marked for exclusion and interpolation. Refer to the General Signal Analyzer section for details.

14 Pupil Diameter Analyzer (Beta Version)

The Pupil Diameter Analyzer performs epoch-based analysis of pupil diameter data. The module performs outlier rejection and filters the data using user-specified settings and criteria. In addition, the module allows the user to amend or override the filter's rejections. The purpose of the module is to generate smooth pupil diameter signals that are suitable for epoch-based analyses, or further processing⁶.

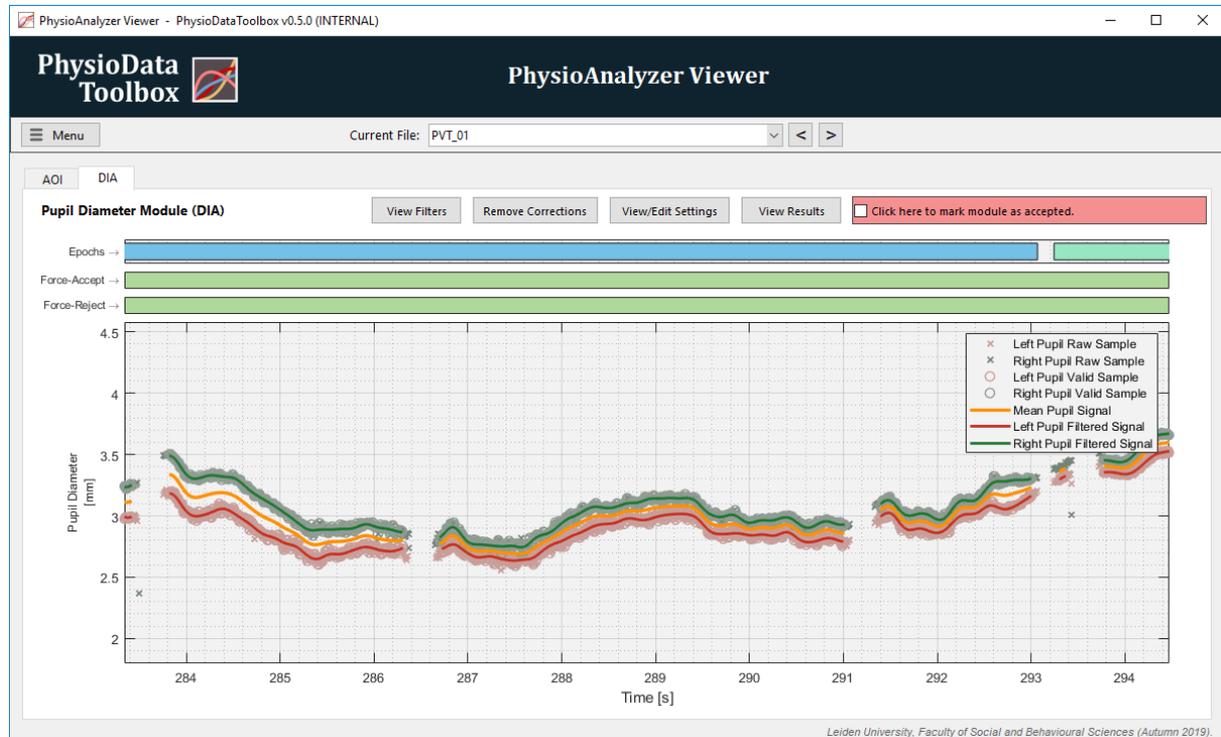


Figure 50: The Pupil Diameter GUI.

Next to the standard epoch plot, the GUI consists of the following graphs:

- **Force Reject:**
 - The 'Force Reject' correction graph shows the zones in which the user overrides the filter's rejection of raw samples; i.e., the user forces the inclusion of the samples within the marked zone. Both the 'Force Reject' and 'Force Accept' graphs feature two bars, one for the right pupil (green-colored bottom bar) and one for the left (red-colored top bar) pupil.
- **Force Accept:**
 - The 'Force Accept' correction graph shows the zones in which the user overrides the filter's inclusion of raw samples; i.e., the user forces the exclusion of samples.
- **Pupil Diameter:**
 - The pupil diameter graph shows the raw and valid pupil diameter samples, as well as the interpolated and filtered pupil diameter signals.

⁶ The filtering pipeline described in this section is largely taken from: Kret, E., & Sjak-Shie, E. E. (2018). Preprocessing pupil size data: Guidelines and code. *Behavioral Research Methods*, 1-7. Available at: <https://rdcu.be/2QCd>.

14.1 Preprocessing Pipeline and Settings

The toolbox features a robust and customizable preprocessing pipeline, which can be broken down into two steps: (1) filtering the raw data to extract the subset of valid samples; and, (2) upsampling and smoothing the valid samples. Before the filtering commences, the raw data can be multiplied by a constant value, the gain:

- **Gain [-]:**
 - A constant value by which the raw data are multiplied before preprocessing. Use this value to transform pupil size data of arbitrary diameter units into millimeters.

Step 1: Filtering the Raw Data

Raw eye-tracking data often contain samples that are purely the result of noise or artefacts, and therefore carry no useful information for pupil size analysis. Identifying and removing these samples, however, is not a trivial task. This filtering pipeline is aimed at identifying three types of often occurring invalid pupil size samples (see Figure 51): (1) dilation speed outliers and edge artefacts; (2) trend-line deviation outliers; and (3), temporally isolated samples.

In addition, pupil size samples that are simply outside of a predefined feasible range can be rejected:

- **Minimum allowable pupil size [mm]:**
 - The smallest allowable pupil diameter. All samples with values below this criterion will be rejected.
- **Maximum allowable pupil size [mm]:**
 - The largest allowable pupil diameter. All samples with values above this criterion will be rejected.

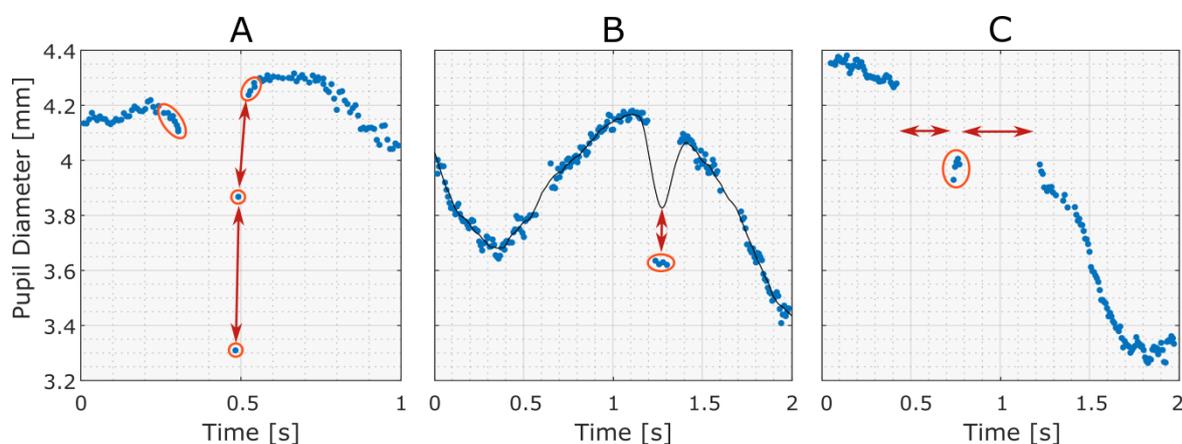


Figure 51: Raw pupil diameter data showing the different kinds of artefacts that are targeted by the raw data filter presented in this paper. The invalid samples targeted for rejection are shown circled. A. Certain artefacts, especially those caused by blinks, are characterized by large inter-sample changes in pupil size; i.e., disproportionately large dilation speeds, as visualized by the arrows. Additionally, the edges of eye-blink gaps may show slopes caused by the onset of eyelid occlusion. B. Outlying clusters of erroneous data-points can be identified by their abnormally large deviation from a smooth trend-line (solid black line). C. Small islands of spurious samples can be identified by their temporal isolation from other samples, as visualized by the horizontal arrows.

Removing Dilation-Speed Outliers and Edge-Artifacts

Dilation speed outliers are samples that feature a disproportionately large absolute pupil size change relative to their adjacent samples. Because the changes between samples due to actual pupil dilation and constriction are generally less than those resulting from artefacts, such as system errors or blinks, detecting outliers in these changes is an effective way of spotting and rejecting invalid samples.

To detect dilation speed outliers, the Median Absolute Deviation (MAD), which is a robust and outlier resilient data dispersion metric, is calculated from the dilation speed series, multiplied by a constant (n), and summed with the median dilation speed. Samples with dilation speeds above the threshold can now be marked as outliers and rejected.

Dilation speed filter settings:

- **Speed-filter MAD threshold [-]:**
 - The number (n) of MADs used as a threshold to reject outliers.
- **Absolute speed-filter threshold [mm/ms]:**
 - The absolute speed filter threshold. Any samples with dilation speeds above this value are rejected. This criteria can be used if the rejection should occur according to some metric other than MADs, such as standard deviation. The absolute threshold, such as the standard deviation of the dilation speeds of all pupil diameter samples inside epochs, can be calculated using a custom MATLAB script.

After the dilation speed outliers have been removed, artefacts around gaps in the data may still remain, especially if these gaps are the result of eye blinks, which may cause pupil size underestimation due to eye-lid occlusion. Therefore, it is sensible to reject the samples that border certain gaps in the data. In order to identify 'gaps', the toolbox looks for sections of missing samples with durations within a specifiable range: the gap lower and upper bounds. Once these gaps have been found, samples that form its leading and trailing edges are removed.

Gap detections and edge-artifact removal settings:

- **Gap lower-bounds [ms]:**
 - The minimum duration a gap needs to have to be classified as such.
- **Gap upper-bounds [ms]:**
 - The maximum duration a gap may have in order to still be classified as such.
- **Backward gap-padding [ms]:**
 - The duration of the section before the start of the gap in which samples are rejected.
- **Forward gap-padding [ms]:**
 - The duration of the section after the end of the gap in which samples are rejected.

Removing Trend-line Outliers

Certain eye-trackers, especially those with higher sampling rates, may produce small groups of clearly invalid samples which, since they are clustered together, are resistant to dilation speed filtering. Instead, these invalid samples can be identified by their strong departure from the signal's trend-line, which can be generated by interpolating and smoothing the data that remain after the previous filtering steps.

Outliers in absolute trend-line deviations can then be identified and rejected in a similar manner to dilation speed outliers by feeding the absolute trend-line deviations into the MAD filter. Subsequently, a new trend-line can be generated using the now remaining samples, and the outlier detection process repeated on all samples considered in the first deviation filter pass. This multi-pass approach allows for the reintroduction of valid samples that were previously rejected due to the invalid samples 'pulling away' the trend-line.

These trend-line outliers can be removed in the toolbox using the 'Residuals filter', which features the following settings:

- **Residuals-filter passes [-]:**
 - The number of passes that the filter will make.
- **Residuals-filter MAD threshold [ms]:**
 - The MAD multiplier; i.e., the number of MADs to use as the rejection threshold.
- **Residuals filter low-pass filter [ms]:**
 - The cutoff frequency of the low-pass filter used to generate the smooth trend-line from the non-rejected samples.

Removing Isolated and Sparse Samples

Another feature of raw pupil size samples that may indicate invalidity is their sparsity. Since a proper pupil size signal is fairly solid, with continuous gaps during blinks and look-away moments, secluded samples are likely to be the result of noise or a momentary eye-tracker glitch, such as erroneous pupil detection during shut eyes.

The toolbox can look for these ‘sample-islands’, which are clusters of samples that are temporally separated from other samples, and reject them. The sparsity filter includes the following settings:

- **Island isolation criteria [ms]:**
 - The minimum distance used to consider samples ‘separated’. The toolbox scans raw non-missing pupil diameter samples for gaps greater than this distance, and identifies the islands—sample clusters—upon which the criterion below is applied.
- **Minimum allowable island size [ms]:**
 - Once islands have been identified, those smaller than the minimum allowable size are removed.

Step 2: Processing the Valid Samples

At this point, depending on whether monocular or binocular data was collected, one or two valid subsets of the original raw samples remain. If data from both eyes are available, the toolbox uses them to generate a third time-series: the ‘mean pupil size signal’. When doing so for the time-points where one pupil’s data is missing, the dynamic offset between the sizes of the two pupils is taken into account.

To increase the temporal resolution and smoothness of the data, the data-points are interpolated at 1000 Hz, then low-pass filtered at the specified cutoff frequency. Subsequently, the toolbox can set sections that were interpolated over unacceptably large gaps to ‘missing’.

Valid samples processing settings:

- **Smoothing low-pass filter frequency [Hz]:**
 - The cutoff frequency of the low-pass smoothing filter applied to the interpolated signals generated from the valid samples (the samples that were not rejected). The recommended cutoff frequency is 4 Hz⁷.
- **Maximum interpolation gap [ms]:**
 - The maximum allowable interpolation gap. Sections that were generated by interpolating over gaps larger than this criterion are set to missing, and not used in the analysis.

⁷ Sirois, S., & Brisson, J. (2014). Pupillometry. *Wiley Interdisciplinary Reviews: Cognitive Science*, 5, 679-692. doi:10.1002/wcs.1323.

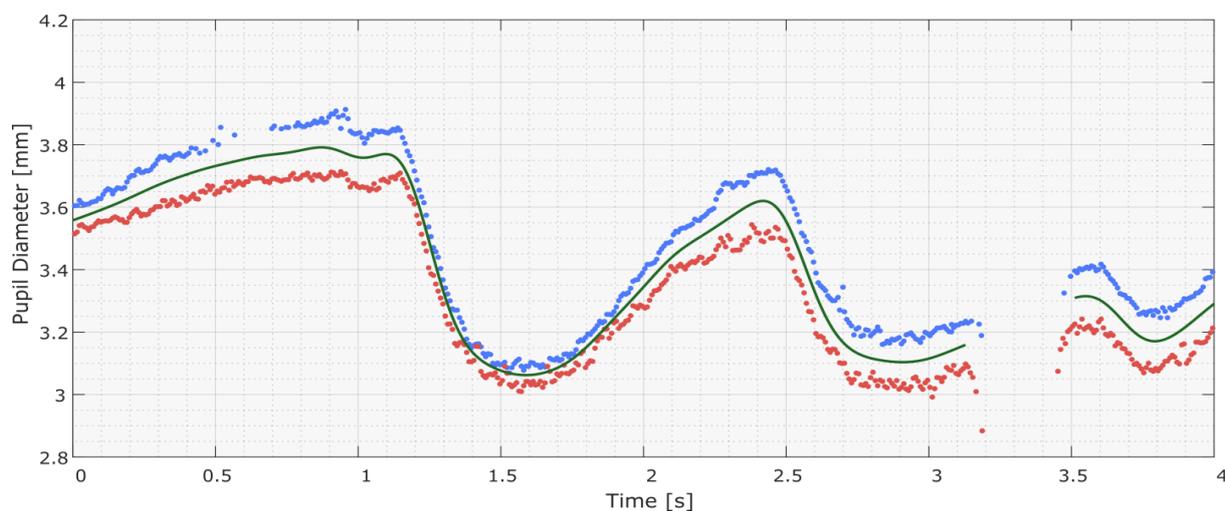


Figure 52: A non-toolbox plot showing the result of the preprocessing pipeline. Shown are: the raw pupil diameter samples of the right and left eyes (blue and red dots, respectively), and the interpolated and lowpass filtered 'mean pupil diameter' signal (green line). The mean pupil diameter signal was generated from the valid raw samples of both pupils, including during the absence of one pupil's data, in which case the local pupil size difference was estimated and used to generate the 'mean pupil size' value (as can be seen at 0.6 seconds). The settings used stipulated that signals were not to be interpolated over gaps larger than 250 ms, hence the missing data around 3.3 seconds.

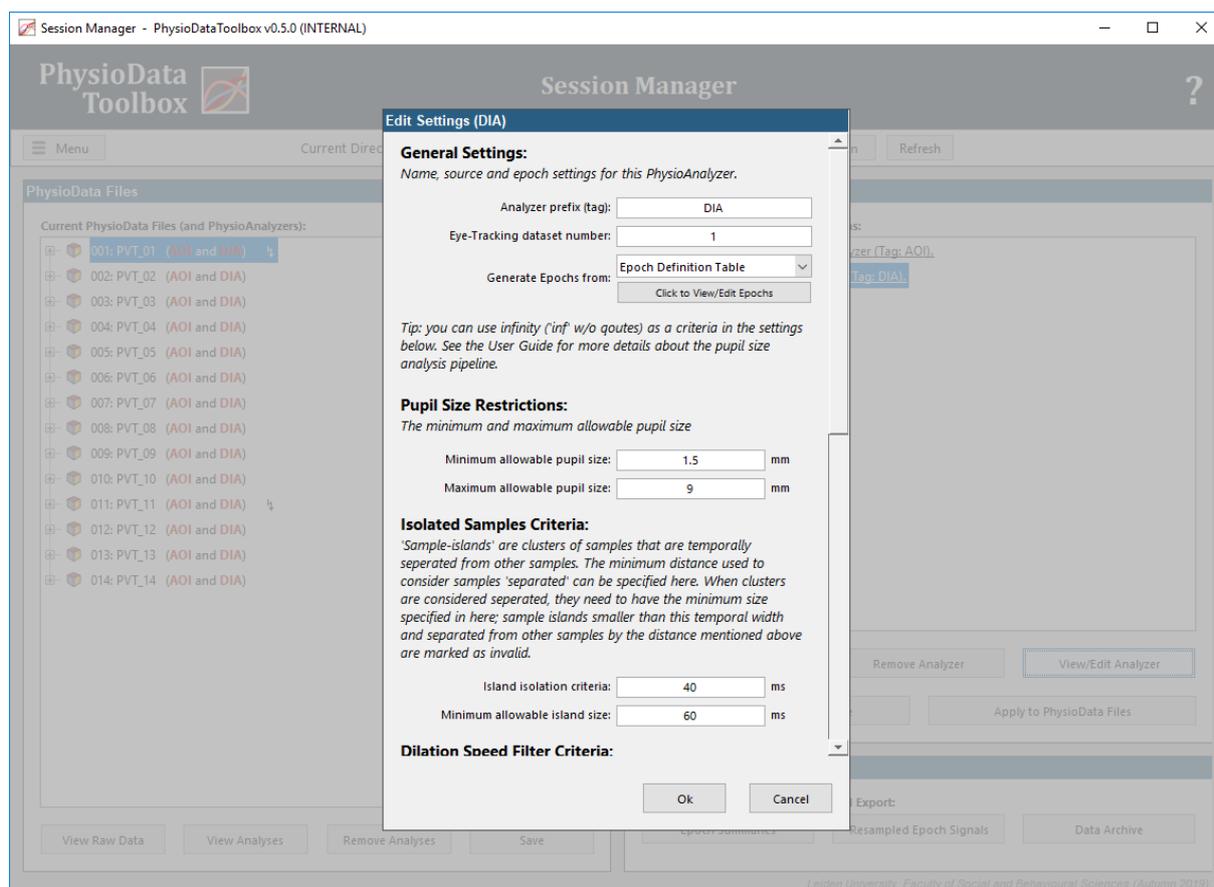


Figure 44: The Pupil Diameter module's settings dialog. A short explanation of the settings is included in the settings dialog.

Data Correction and Workflow

The toolbox allows the user to override the previously described filters by forcing the acceptance or rejection of raw data samples, regardless of the filter and threshold violation. It is advisable to first determine the optimal automatic filtering settings for all files, then to review, and, if necessary, correct the filtering algorithm.

The Pupil Diameter module includes a popup window for visualizing the filter thresholds and intermediate steps, see Figure 53. To show, click the 'Show Filters' button in the PhysioAnalyzer viewer. The information in this window can be used to review the efficacy of the filter settings on the current data.

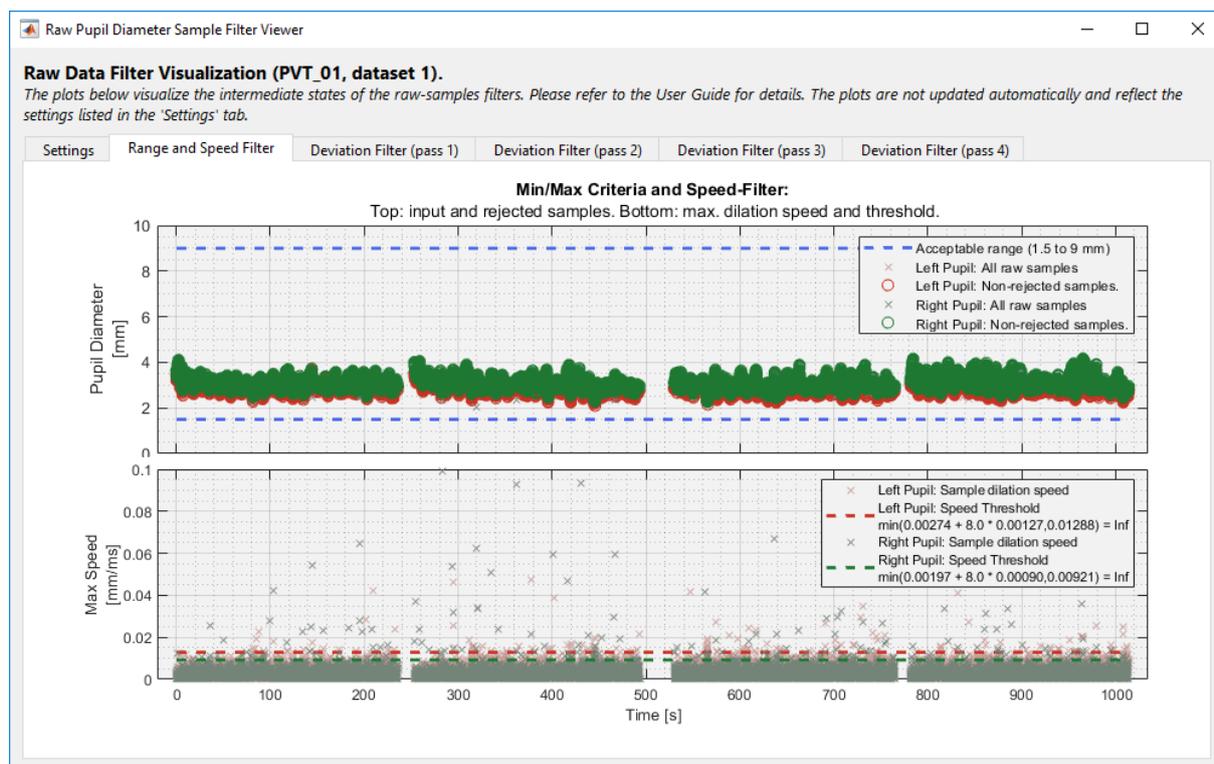


Figure 53: The Pupil Diameter module's filter visualization window.

As described at the start of this section, the user can overrule the automatic sample rejection algorithm. This can be done by selecting a time section of the data by right-clicking and dragging inside the graph, then selecting one of the following actions, per eye:

- **Keep as is:**
 - Nothing is changed for the corresponding eye.
- **Force acceptance of samples in selected range:**
 - Force that the module uses the raw samples inside the selected range, for the given eye.
- **Force rejection of samples in selected range:**
 - Force that the module discards the raw samples inside the selected range, for the given eye.
- **Remove user overrides in selected range:**
 - Remove any force-accept or force-reject zones inside the selected range, for the given eye.

The filter settings and data correction approach are the responsibility of the researcher, and are currently out of the scope of this document.

The module automatically ensures that a force-accept zone and a force-reject zone do not overlap for a given eye. As such, adding one zone where the other exists removes the overlap from the pre-existing zone. As with other modules, the user correction graphs are light green when no zones are present.

Figure 54 shows a section of pupil diameter data obtained using a Tobii Glasses II eye-tracker. It is noticeable that the system had trouble detecting the right pupil diameter between 524 and 562 seconds, resulting in a cloud of erroneous samples, which is in stark contrast to the smooth and coherent pupil diameter signals seen in the rest of the recording.

Due to the nature of the noise and the filter settings used, the erroneous right-pupil samples were not rejected by the raw data filters. Consequently, the 'mean pupil diameter signal' was distorted by the contribution of the incorrect right-pupil filtered signal, which was generated by interpolating the erroneous right-pupil samples.

To remedy this, the user can insert a force-reject zone for the right-pupil samples covering the noisy section, see Figure 56. This will prevent the toolbox from using any right-pupil samples within that zone. Additionally, given that the section exceeds the maximum interpolation distance, the toolbox will not generate the interpolated smooth signal for that section, for the right pupil diameter. As a result, the 'mean pupil diameter' signal in the force-reject section will be generated solely from the left pupil data, which contains far less noise. See Figure 57 for the results.

Similarly, in cases where the filter rejected valid samples, and the issue cannot be easily corrected by tweaking the settings, the user can force the acceptance of rejected samples by inserting a 'force acceptance' zone.

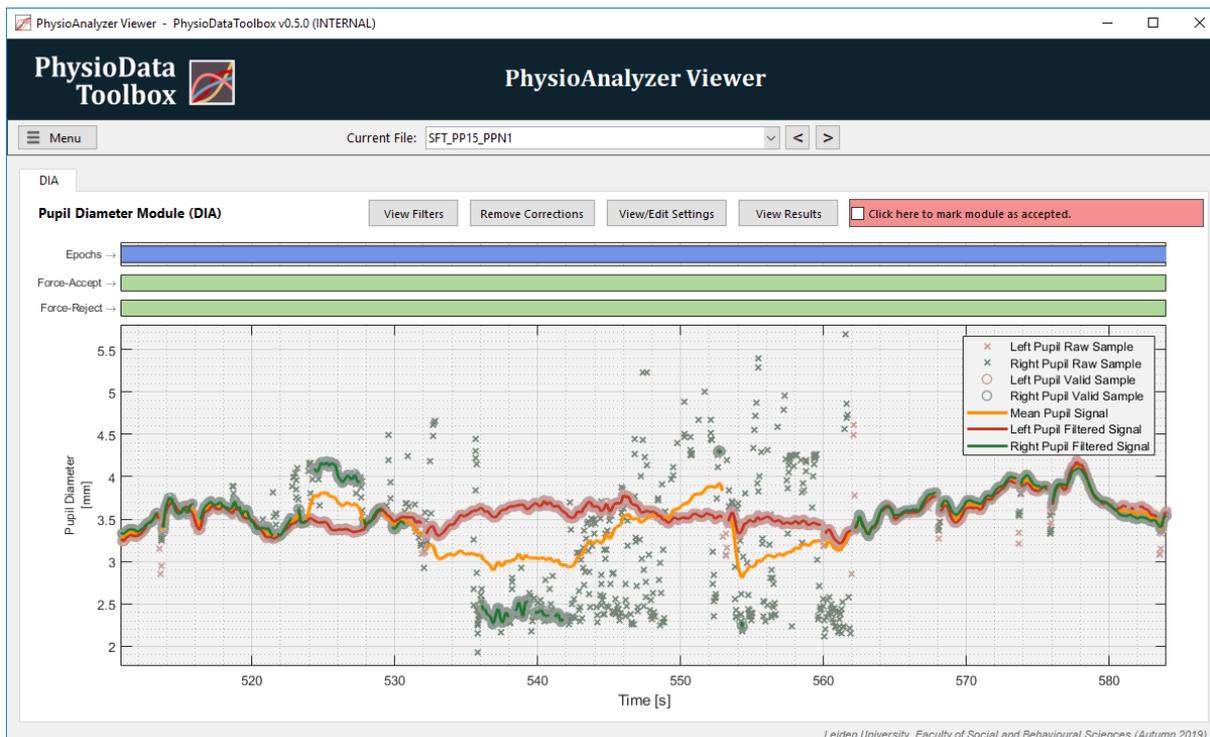


Figure 54: A Pupil diameter dataset showing measurement artifacts in the right pupil diameter samples. To correct this, the user can complement the raw data filter by manually rejecting the erroneous samples, see Figure 56.

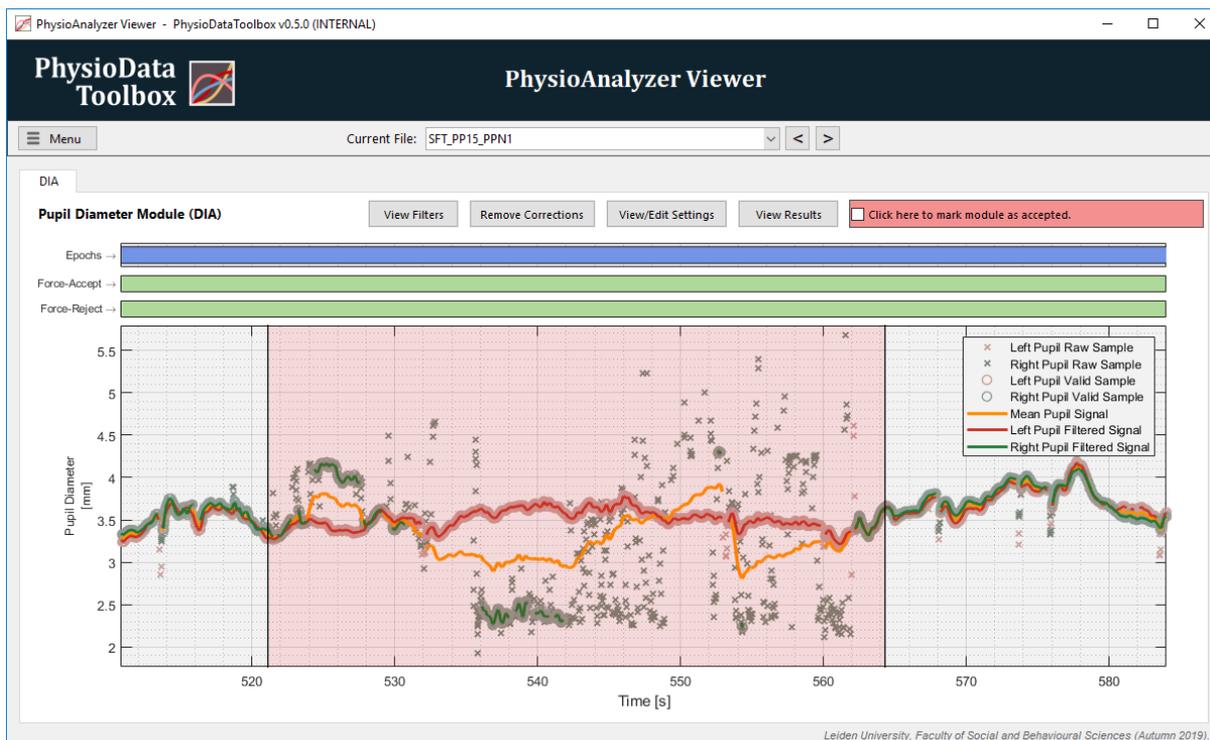


Figure 55: Inserting a rejection zone. The desired section is selected by left-clicking in the pupil diameter graph and dragging the highlighted section while holding down the mouse button. Once the desired section has been selected, releasing the mouse button opens a popup menu with several correction options.

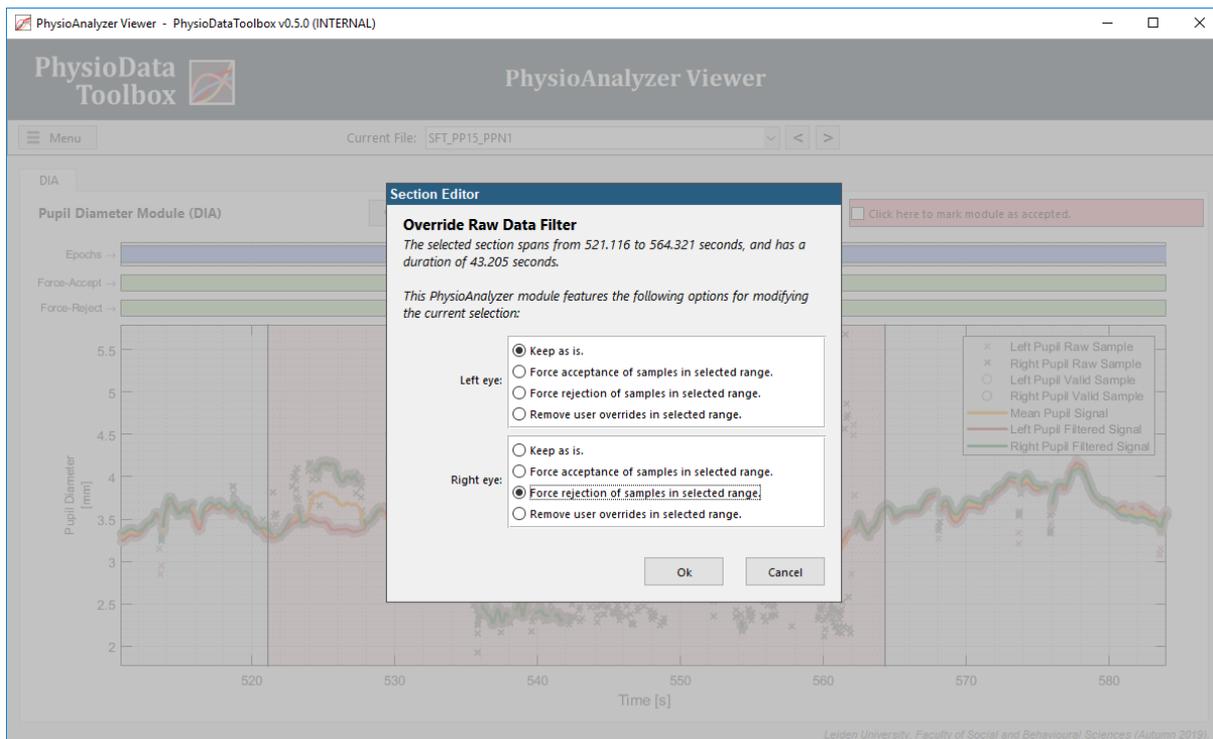


Figure 56: Force reject samples. In the current example, the ‘Force rejection’ option is chosen for the right eye to manually remove the erroneous samples shown in Figure 54 .

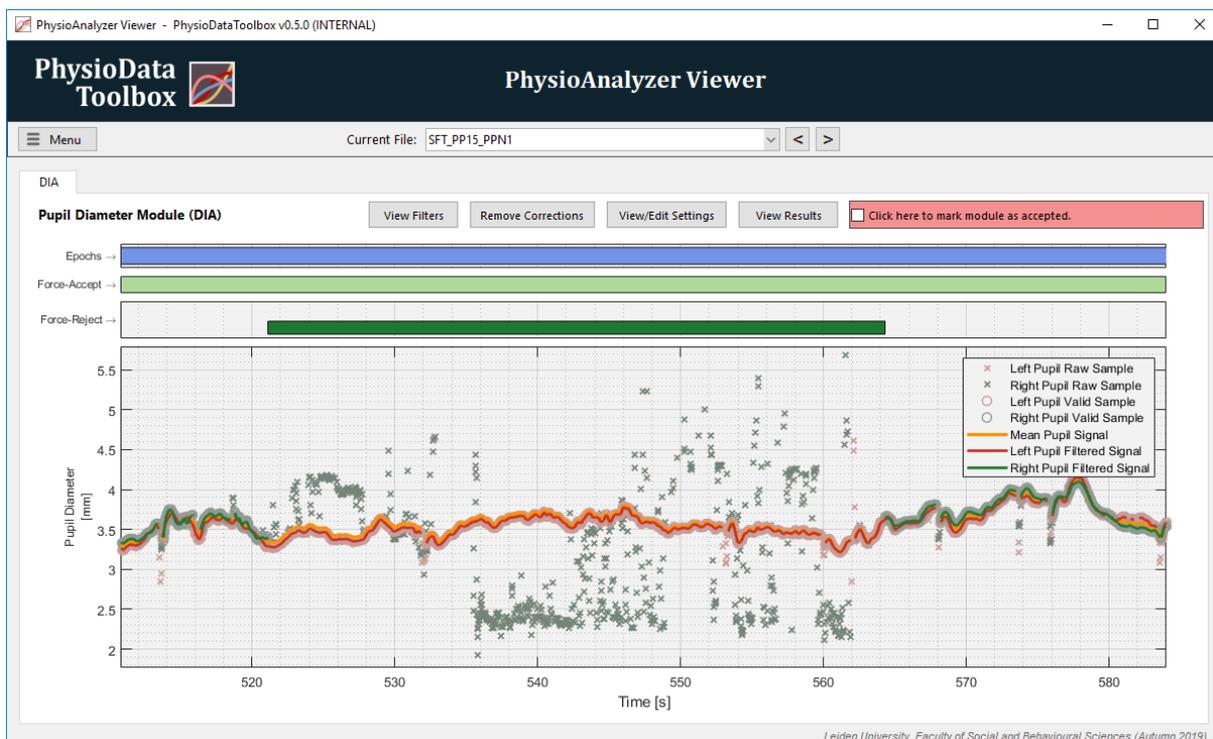


Figure 57: Once a ‘force-rejection’ zone has been inserted, no raw samples for the eye it was defined for are used. In this case, the erroneous contribution of the noisy right pupil samples is removed, leaving a smooth ‘mean pupil diameter’ signal generated solely from the left pupil data (in the force-reject zone).

14.2 Metrics

The Pupil Diameter Analyzer extracts standard descriptive statistics from the filtered and interpolated pupil diameter data. The metrics described in Table 10 are calculated for raw and processed data of the left and right pupils. In the case that data from both pupils are present, the ‘mean pupil diameter’ signal is also analyzed, and given the suffix ‘BothEyes’.

Table 10: The metrics calculated by the Pupil Diameter module. In the output, the <eye> specifier indicates which pupil the metric belongs to (Both eyes, Left eye or Right eye).

Code:	Unit:	Description
countRawSamples_<eye>	-	The total number of raw pupil diameter samples inside the epoch, for the current eye. Only Non-NaN samples are counted.
countValidSamples_<eye>	-	The total number of valid samples inside the epoch, for the current eye. Only samples that were not rejected by the user or the filters are counted.
meanValidSamples_<eye>	mm	The mean value of the valid samples inside the epoch, for the current eye.
minValidSamples_<eye>	mm	The minimum value of the valid samples inside the epoch, for the current eye.
maxValidSamples_<eye>	mm	The maximum value of the valid samples inside the epoch, for the current eye.
stdValidSamples_<eye>	mm	The standard deviation of the values of the valid samples inside the epoch, for the current eye.
mean_dia_<eye>	mm	The mean pupil diameter, calculated per epoch from the filtered signal generated from the data of the current eye(s).
min_dia_<eye>	mm	The minimum pupil diameter, calculated per epoch from the filtered signal generated from the data of the current eye(s).
max_dia_<eye>	mm	The maximum pupil diameter, calculated per epoch from the filtered signal generated from the data of the current eye(s).
signalCoverage_<eye>	%	The approximate percentage of the epoch in which data from the interpolated signal is available, for the current eye(s).

In addition, the Pupil Diameter Analyzer exports metadata regarding the parameters used to process the data. These parameters are exported to the ‘INFO’ or ‘metadata’ tab in the Excel output.

14.3 Resampled Signals

When exporting the resampled epoch signals, the Pupil Diameter Analyzer resamples the filtered pupil size signals.

15 Area of Interest Hit Analyzer

The Area of Interest Hit Analyzer performs epoch-based analysis of AOI hit data. AOI hit data should consist the AOI name that was hit, per sample. The Analyzer outputs several generic hit data.

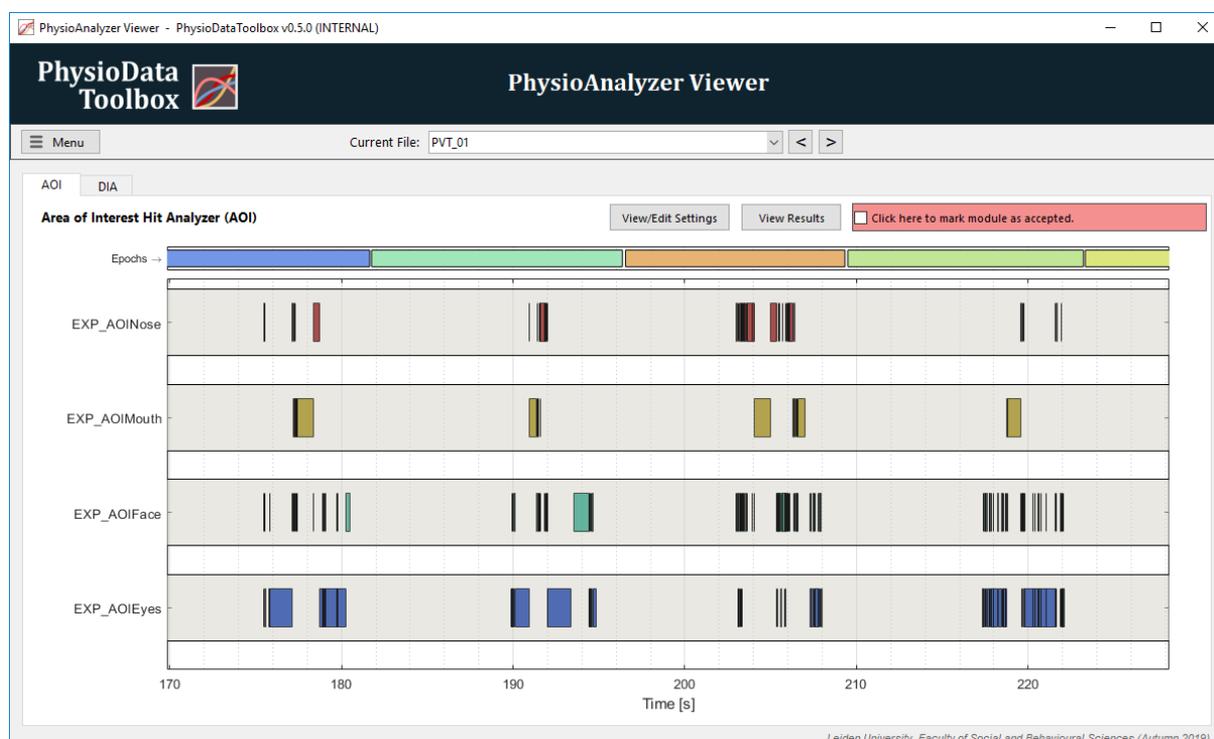


Figure 58: The Area of Interest Hit GUI, showing the hit data per AOI. In the current file there were four AOIs. Notice that the EXP_AOIEyes AOI was hit most frequently.

15.1 Settings

The AOI Hit Analyzer only has General Settings. For the General Settings, see the description in the General Signal Analyzer section.

15.2 Data Correction

The Area of Interest Hit Analyzer does not allow for corrections to be made, i.e. it is not possible to exclude sections of the raw AOI hit data for further analyses.

15.3 Metrics

The AOI Hit Analyzer extracts descriptive analysis from the AOI hit signal. It outputs the AOI hit metrics described in Table X, per AOI.

Table 11: Are of Interest Hit Analyzer Metrics

Code:	Unit:	Description
<AOI>Total_Hit_Time	s	The total time the AOI was hit, per epoch.
<AOI>Time_to_First_Hit	s	The time to the first hit on the AOI, per epoch.
<AOI>First_Hit_Duration	s	The duration of the first hit on the AOI, per epoch.
<AOI>Hit_Count	n	The number of hits on the AOI, per epoch.

15.4 Resampled Signals

When exporting the resampled epoch signals, the AOI Hit Analyzer resamples and extracts the AOI hits. This results in a hit fraction per bin indicating the fraction of the bin the AOI was hit (0 = no hit, 1 = complete hit).

16 Epochs and Events

The PhysioData Toolbox can analyze subsections of signals based on two methods: **Epochs** and **Events**. These can be defined using rules that reference moments in the recorded data, such as **Markers** and **Labels**.

Data can either be analyzed inside an epoch, or around an event:

- **Epochs:**
 - When a PhysioAnalyzer uses Epochs, sections of the signal are identified using Epoch Definition Rules, and those sections are analyzed.
 - Epochs are used when segment demarcation can occur using only markers and/or labels, in combination with predetermined rules.
- **Events:**
 - When a PhysioAnalyzer uses Events, discrete moments in the Signal are identified using Event Definition Rules, and areas around those moments are analyzed.
 - Events are used when analyzing segments that cannot be demarked based on markers or labels alone; such as is the case with IBIs, for which the analyzed area is partly determined by the IBIs themselves.
 - Events are currently only used by the IBI Sequence Analyzer module.

Referenceable moments in signals:

- **Markers:**
 - Markers are referenceable features in a special marker channel. They can be short duration pulses or long duration plateaus, and are usually sent via a TTL or LPT signal.
 - Markers have the following properties:
 - **Channel** (number): The channel that contains the markers.
 - **Value** (positive non-zero integer): this is the value—i.e., height—of the marker, be it a pulse or plateau.
 - **Occurrence** (number): Marker occurrence is defined as the sequence index of the desired marker within all markers with the entered value, in that specific channel; i.e., the first, second, third occurrence of marker with value *X*.
 - **At** (on or off): The time when a marker assumes its value is referred to as the 'on' event, and when it releases that value is called the 'off' event. Hence, one can refer to the moment 'At' the 'on' event or at the 'off' event of a marker.
 - **Delay** (number in seconds): delays can be used when an offset to the on or off events in a marker is necessary when defining an epoch or event.
- **Labels:**
 - Labels are referenceable named moments inside a recording; e.g., AcqKnowledge allows the user to add Labels to a recording. These labels are purely discrete; i.e., they contain a single location in time. Therefore, labels do not have on or off events. When the toolbox looks for label matches, leading and trailing whitespaces in the raw-data labels are ignored.
 - **Channel** ('label'): Enter 'label' without quotation marks as the channel to refer to labels.
 - **Value** (word or phrase): this is the contents of the label.
 - **Occurrence** (number): Similar to Marker occurrence, Label occurrence is the sequence index of the desired label, within the set of all labels with the entered value.
 - **Delay** (number in seconds): delays can be used when an offset to the label time is necessary when defining an epoch or event.

Referenceable moments in Eye-Tracking datasets:

- **Eye-Tracker Events:**
 - Eye-tracker events can be referenced by filling in 'ET.events' in the channel columns; or, when the file contains multiple eye-tracking datasets: 'ET(n).events' where 'n' refers to the dataset number. The other columns can be filled in a similar manner as the Labels. Event occurrences are calculated only within their dataset.
- **Eye-Tracker Sections:**
 - Eye-tracker sections are segments defined in an eye-tracking dataset. These sections, similar to predefined epochs, can be referred to by filling in ET(n).sections in the channel columns ('n' being the dataset number). The other columns can be filled in a similar manner as the Labels, but with the addition of specifying either the on or off of the section.

In addition, the toolbox supports referencing the start or end of the recording. This can be done by entering 'file' as the channel (this only works if the PhysioData file contains signals):

- **File Events:**
 - File events refer to the start or end of the recording.
 - **Channel** ('file'): Enter 'file' without quotation marks as the channel to refer to the file events.
 - **Value** ('SOF' or 'EOF'): SOF and EOF refer to the start of the recording (t=0) and the end of the recording, respectively. Enter without quotation marks.
 - **Delay** (number in seconds): The delay relative to the SOF or EOF.

An example of how to refer to specific features in a marker channel is given in Figure 59.

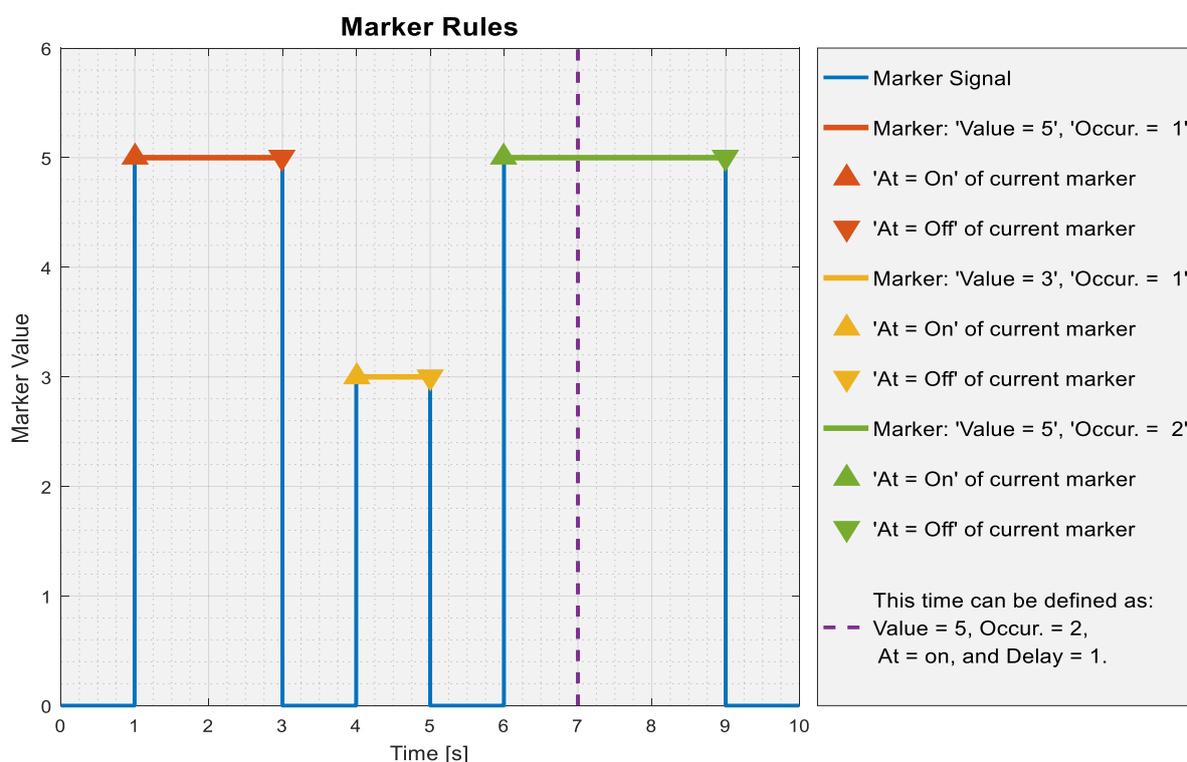


Figure 59: Marker referencing example. The marker signal is 10 seconds long and contains three markers: the first and third have value 5, and the second has value 3. The legend shows the properties of the specific marker features; and lastly, how to define a time based on a specific marker reference. Note that the third marker is the second occurrence of the marker with value 5.

Epochs are created by defining two time points, the Epoch Start and the Epoch End, whereas Events only contain one time point, the event time itself.

16.1 Defining Epochs

The PhysioData Toolbox creates Epochs based on Epoch Definition Rules, which has the form of a table with the following columns: **StartChannel**, **StartValue**, **StartAt**, **StartDelay**, **StartOccur**, **EndChannel**, **EndValue**, **EndAt**, **EndDelay**, **EndOccur**, **EpochModify**, and **EpochName**. The EpochName column is the name of the Epoch, which must: be unique; contain only numbers, letters and underscores; and, start with a letter. The EpochModify column is explained in the section ‘Extra Options’.

In addition, extra metadata columns can be added to the table. These extra columns are copied to the output, and may be useful for refactoring the data.

Four examples of untypically short Epochs are given in Table 12 and Table 13. Assume that the markers are located in channel 4, and that the marker signal is equal to the one shown in Figure 59.

Table 12: Columns 1 through 7 of an Epoch Definition example.

StartChannel	StartValue	StartAt	StartDelay	StartOccur	EndChannel	EndValue
4	5	on	0	1	4	5
4	5	on	0	2	4	5
4	3	on	-3	1	4	3
file	SOF		0		file	EOF

Table 13: Columns 8 through 13 of an Epoch Definition example. The last column is an optional metadata column.

EndAt	EndDelay	EndOccur	EpochModify	EpochName	extra1
off	0	1		Epoch1	Condition 5
off	0	2		Epoch2	Condition 5
on	3	1		Epoch3	Condition 3
	0		timeSlice(3)	FullSignal	

Epochs defined in Table 12 and Table 13:

- **Row 1:**
 - This epoch covers the area spanned by the first marker with value 5.
 - Epoch name: Epoch1
 - Time: from 1 to 3 seconds.
 - Extra info: Condition 5
- **Row 2:**
 - This epoch covers the area spanned by the second marker with value 5.
 - Epoch name: Epoch2
 - Time: from 6 to 9 seconds.
 - Extra info: Condition 5

Row 3:

- This epoch starts 3 seconds before the onset of the first marker with value 3, and ends 3 seconds after that same onset.
 - Epoch name: Epoch2
 - Time: from 1 to 7 seconds.
 - Extra info: Condition 3
- **Row 4:**
 - This epoch spans the whole file, from start to finish. Subsequently, the epoch is sliced into 3-second segments. See the section for details.

The Epoch Starts and Epoch Ends are defined independently, and can use different referencing methods, as long as the resulting Epoch has a positive finite duration; i.e., Epoch End > Epoch Start.

16.2 The Epoch Builder

The PhysioData Toolbox contains an Epoch Builder GUI to facilitate the creation of the Epoch Definition Rules. The GUI can be launched when modifying the epoch settings of a newly created or already existing PhysioAnalyzer: in the settings dialog window of the PhysioAnalyzer, click the button labeled ‘Click to View/Edit Epochs’.

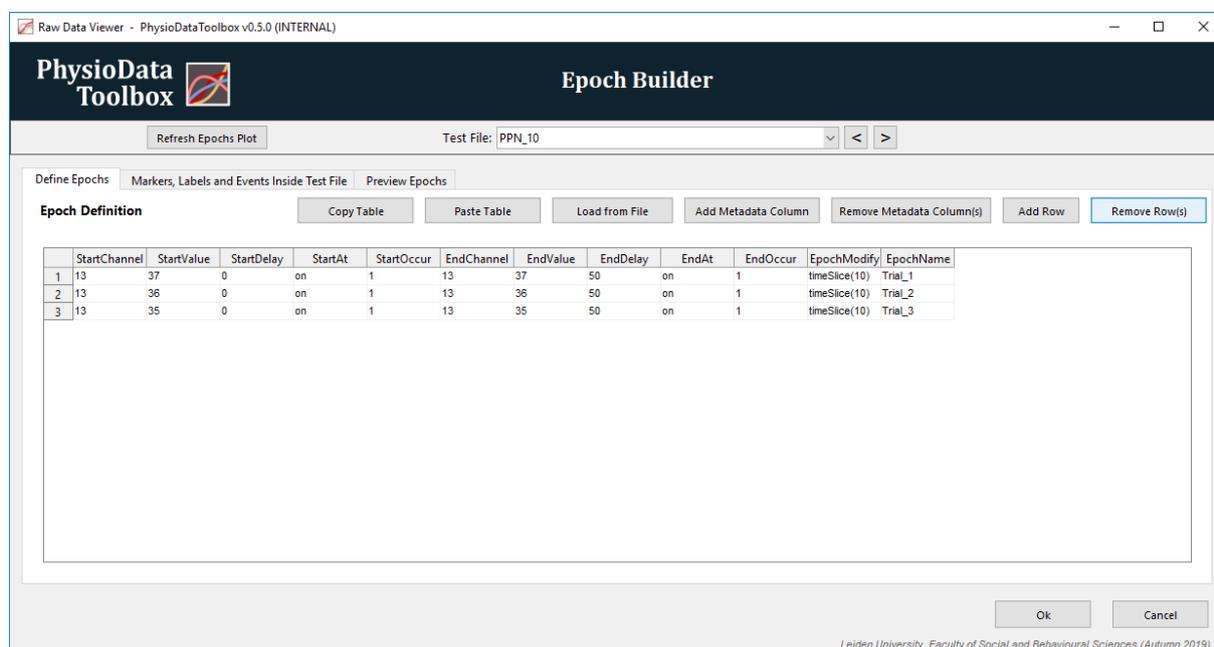


Figure 60: The Epoch Builder GUI, showing the definitions of three epochs, all of which are sliced into 10 second segments.

Use the table in the ‘Define Epochs’ tab to create a set of Epoch Definition rules. Premade tables can also be loaded from an Excel file, pasted from the clipboard. In both cases, the first row is assumed to contain only headers. When loading an Excel file, only the first (leftmost) worksheet is read.

The ‘Test Epoch on Test File’ tab allows the user to test the epoch rules entered in the ‘Define Epochs’ tab, on any file currently imported, see Figure 61. Use the navigation bar and the ‘Refresh Epochs Plot’ button to test the rules.

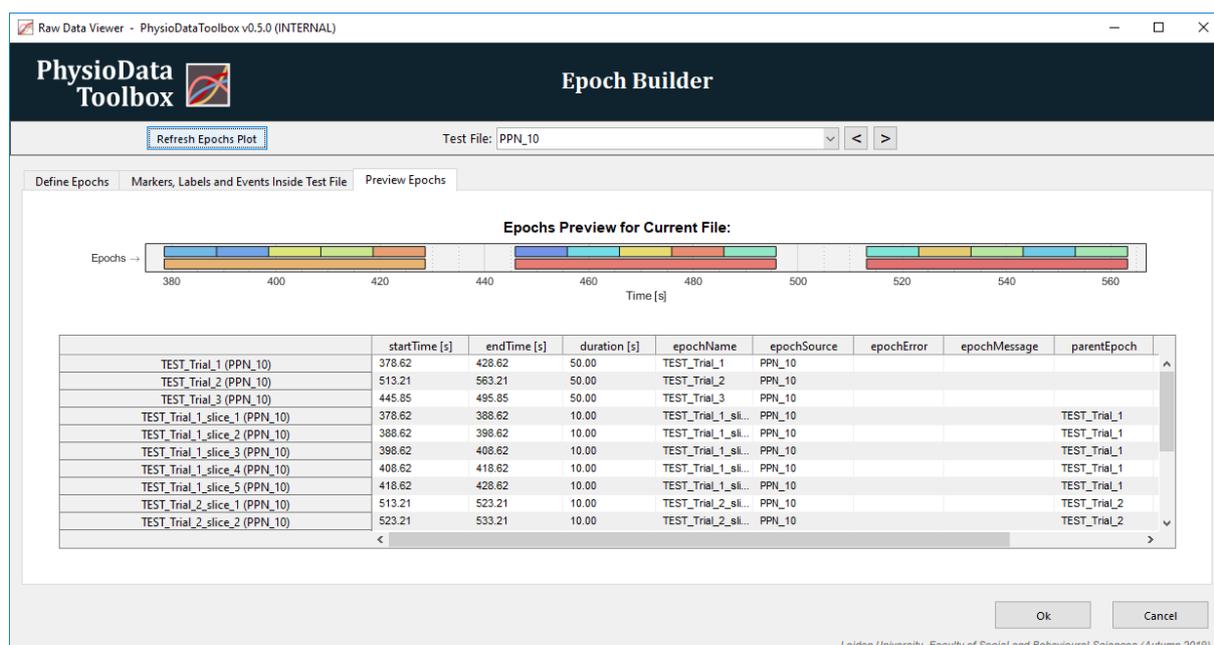


Figure 61: Testing the epochs definitions on a file.

Extra Options

Each successfully detected epoch can be further modified with a special command entered in the EpochModify column. Leave this field blank to disable modification. This version of the toolbox supports the following EpochModify command:

- **timeSlice(n):**
 - This command slices the epoch into sections with the duration given by n. The last slice contains the remainder; e.g., timeSlice(10) creates six 10-second slices and one 2-second slice out of a 62-second epoch. Since epoch duration may vary across files, not all files will have the same amount of slices.
- **allOccur():**
 - This command finds epochs that are defined in the same manner as the given epoch, except for their startOccur and endOccur parameters. Provided that the epoch's startChannel and endChannel; startValue and endValue; and startOccur and endOccur are equal, the command will find all epochs with increasing startOccur and endOccur values.

16.3 Defining Events

Events are created in the same manner as Epochs, with the exception that they only require a single time point and do not have a 'EventModify' column.

17 Tips and Advice

- **Referencing this toolbox:**
 - This toolbox can be referenced in your preferred style using the following information:
 - **Author:** E. E. (Elio) Sjak-Shie
 - **Date:** 2018
 - **Software name:** PhysioData Toolbox (Version 0.5)
 - **Website:** physiodatatoolbox.leidenuniv.nl
 - **Author affiliation:**
 - Research Support Department, Faculty of Social Sciences, Leiden University, The Netherlands.
 - Leiden Institute for Brain and Cognition, The Netherlands.
 - The toolbox is written in MATLAB 2017a, which can be referenced in-line using:
 - MATLAB Release 2017a (The MathWorks, Inc., Natick, MA, USA).
- **Save often and archive:**
 - If you are making changes you intend to keep, make prudent use of the ‘Save’ button in the Session Manager. No changes are stored unless explicitly saved by the user.
 - When the data are fully reviewed and corrected, create an archive using the ‘Export Data of Selected File(s)’ button in the Session Manager.
- **Undoing:**
 - The toolbox does not feature an Undo button. However, you can always revert to the last saved state by clicking the ‘Refresh Directory’ button, and then discarding the changes.
 - If you mistakenly overwrite user-corrections when applying new PhysioAnalyzer settings to PhysioData files, you can revert all files to their old states using the technique described in the bullet above—given that the old state was saved.
- **Use Excel to make the epoch definition rules.**
 - Making tables in Excel is much faster and easier compared to using the Epoch or Event Builders, which are limited in the cell editing and copy-pasting abilities.
- **Copy PhysioAnalyzers between files.**
 - If you have PhysioData files that already contain PhysioAnalyzers, and you want to use them in other similarly structured files:
 - Copy the settings from the desired file using the ‘Import from PhysioData File’ button in the Session Manager.
 - Click ‘Apply to PhysioData Files button’, and select ‘Do not Overwrite/Update Anything’ so that the files already containing the analyzers are not affected.
- **Refactoring the output:**
 - The results exported as an Excel file can be restructured in SPSS using the Restructure option in the Data menu.
 - In Excel, you can use ‘Format as Table’ and optionally ‘Summarize with PivotTable’ to quickly restructure, filter, and summarize your data.
- **Experimental design:**
 - Use deterministic and descriptive markers; i.e., make sure that a single epoch definition scheme can identify all necessary epochs while referencing only the available

markers, events or labels. The toolbox cannot apply logic or conditions when detecting and naming epochs, as such, avoid encoding epoch data in more than a single marker.

- **Use the toolbox to check the data during an experiment:**
 - It is a prudent exercise to make the Epoch Definition table when designing the experiment, before actually starting the data collection. Subsequently, during data or pilot studies, the toolbox and Epoch Definition can be used to check the physiological signals, verify the presence and suitability of the markers, and run preliminary descriptive analysis.

18 PhysioData Format

18.1 PhysioData File Format

The toolbox can only read data from PhysioData files, which are specifically formatted MATLAB MAT-files with the physioData extension. The PhysioData files are designed to only carry built-in MATLAB types; as such, no ‘PhysioData objects’ or ‘PhysioAnalyzer objects’ are saved, allowing the files to be read without needing the class definitions.

PhysioData files can be generated using the AcqKnowledge Converter Tool or a custom MATLAB script, and must adhere to the standard structure described in this section. The PhysioData Toolbox package contains several MATLAB scripts describing how to generate PhysioData files from different sources. **Also provided with the toolbox are MATLAB data-model classes for generating a PhysioData file and its contents, which significantly simplifies the process of converting data to the format described in this section.** See the ‘supplementaryCode’ subfolder inside the PhysioData Toolbox folder for the supplementary MATLAB code. Example conversion scripts are inside folders prefixed with ‘Convert_’.

PhysioData files can contains two types of raw data: signals and eye-tracking datasets. In addition, the files can contain timing data, such as events and epochs. Lastly, the files hold all states and settings for the PhysioAnalyzers—it is from these parameters, in combination with the raw data, that the PhysioAnalyzer data are generated; i.e., derived data, such as Heart Rate, is never saved in the PhysioData files, but rather dynamically generated during the analysis session.

The following tables describe the format of raw data inside PhysioData file. The convention used when describing a MATLAB variable is shown in Table 14.

Table 14: Data-structure visualization table. The name of the variable or file is given in the top right cell, followed with a description in the top left cell. After the top row, the variable-names, or fieldnames in the case of structs, are listed in the first column, with their size and data type described in the second. The last columns gives a description of each variable or field. Where appropriate, a nested table may be used; i.e., variables and fields inside the outer variable may be shown as a summary, as in row one, or as a nested table, as in the second row.

Name	<i>Description</i>	
[size type]		
Variable/field name	[size type]	<i>Variable/field description</i>
Variable/field name	<i>Description</i>	
[size type]		
Variable/field name	[size type]	<i>Variable/field description</i>

Table 15: Variables inside a PhysioData MATLAB file.

PhysioData file	<i>This table represents a MATLAB file with the physioData extension. It holds all raw and timing data, and the PhysioAnalyzer settings and states.</i>	
[1 x 1 MATLAB-file]		
data	[1 x 1 struct]	<i>The 'data' variable inside the PhysioData file holds all the raw data, see Table 16.</i>
epochs	[1 x 1 struct]	<i>The optional 'epochs' variable holds information about pre-generated epochs.</i>
physioDataInfo	[1 x 1 struct]	<i>The physioDataInfo variable inside the PhysioData file holds metadata about the file. This information is displayed by the raw data viewer in the 'File Info' tab.</i>
physioAnalyzerTemplates	[1 x 1 struct]	<i>The 'physioAnalyzerTemplates' variable contains the settings and states for all the PhysioAnalyzers inside the file, it is not currently documented and should not to be manually created or edited.</i>

Table 16: The data variable inside the PhysioData file.

data	<i>The 'data' variable inside the PhysioData file holds all the raw data. The fields within it are optional, but either the signals field or the eyeTracking field must at least be present.</i>	
[1 x 1 struct]		
labels	[1 x 1 struct]	<i>'labels' holds the labels associated with the signals(i.e. events). See 'labels' table. This variable is optional, and can be omitted or left empty.</i>
signals	[1 x 1 struct]	<i>Holds the raw signals. See 'signals' table. This variable is optional, and can be omitted or left empty.</i>
eyeTracking	[1 x n struct]	<i>Holds the eye-tracking datasets. See eyeTracking table. This variable is optional, and can be omitted or left empty.</i>

Table 17: The labels field inside the data variable.

labels	<i>The labels field inside the data variable holds the labels (i.e. events) that are associated with the signals inside the PhysioData file. This field can be omitted if no labels are present.</i>	
[1x1 struct]		
t	[k x 1 double]	<i>A single column vector with the times, in seconds, of the labels.</i>
channels	[1 x 1 cell]	<i>A single cell that holds a [k x 1 cell] cell array of strings from character arrays, with each string being the name/value of the label.</i>
channelUnits	[1 x 1 cell]	<i>A single cell containing a [1 x 1 cell] cell with the following string: '' (labels don't have units).</i>
channelNames	[1 x 1 cell]	<i>A single cell containing a [1 x 1 cell] cell with the following string: 'Labels'.</i>

Table 18: The signals field inside the data variable.

signals	<i>The signals field inside the data variable holds the raw signals. All fields described below must be present, and must conform to the format described below. This field can be omitted of the PhysioData file only contains eye-tracking data.</i>	
[1x1 struct]		
fs	[1 x 1 double]	<i>A scalar double indicating the sampling frequency of the signals data, in Hz. The time vector is dynamically calculated with the first sample at $t = 0$, with an sample interval of $1/fs$.</i>
channels	[1 x k cell]	<i>A cell array that holds one signal per cell; i.e. one time-series per channel. All channels must contain the same amount of samples in a single column. All signals share the same time-vector.</i>
channelUnits	[1 x k cell]	<i>A cell array of strings from character arrays, with each string being the unit of the corresponding channel.</i>
channelNames	[1 x k cell]	<i>A cell array of strings from character arrays, with each string being the name of the corresponding channel.</i>
channelDescription	[1 x k cell]	<i>A cell array of strings from character arrays, with each cell containing a description of the corresponding channel.</i>

Table 19: Data structure of the eyeTracking variable inside the PhysioData file.

eyeTracking	<i>A struct array for holding raw eye-tracking data. This field can be omitted if the PhysioData file only contains signals.</i>	
[1 x n struct]		
filename	[1 x n char]	<i>Name of the source file.</i>
name	[1 x n char]	<i>Name of the eye-tracking dataset.</i>
diameterUnit	[1 x n char]	<i>Unit of the pupil diameter. The current version of the toolbox assumes that the unit is 'mm'.</i>
raw_t_ms_max	[1 x 1 double]	<i>Maximum timestamp of raw data</i>
gazeData	[1 x k struct]	<i>Struct array holding data for k snapshots. See gazeData table. This field can be omitted if not relevant.</i>
diameter	<i>A scalar struct containing the time vector, in ms, and the raw measured pupil diameters of the left and right eyes, measured in millimeters. All three vectors must have the same amount of rows. This field can be omitted if not relevant.</i>	
[1 x 1 struct]		
t_ms	[n x 1 double]	<i>Time vector, in milliseconds.</i>
L	[n x 1 double]	<i>Raw pupil diameter of the left eye. Contains NaN where not available.</i>
R	[n x 1 double]	<i>Raw pupil diameter of the right eye. Contains NaN where not available.</i>
eventSections	<i>Struct array for holding k user defined epochs/sections. This field can be omitted if not relevant.</i>	
[1 x k struct]		
name	[1 x m char]	<i>Name of section.</i>
hit	[1 x 2 double]	<i>Start and end times of sections, in milliseconds.</i>
desc	[1 x m char]	<i>Description of sections.</i>

labels	<i>Scalar struct for holding label information. This field can be omitted if not relevant.</i>	
[1 x 1 struct]		
name	[m x 1 cell]	<i>A cell array of strings that holds the labels names.</i>
t_ms	[m x 1 double]	<i>Vector containing the label timestamps, in milliseconds.</i>

Table 20: gazeData structure

gazeData	<i>Struct array that holds gaze information for k snapshots.</i>	
[1 x k struct]		
snapshotName	[1 x m char]	<i>Name of the snapshot.</i>
snapshotImage	[H x W x 3 uint8]	<i>Snapshot image (optional).</i>
snapshotWidth	[1 x 1 double]	<i>Width of the snapshot.</i>
snapshotHeight	[1 x 1 double]	<i>Height of the snapshot.</i>
gazePoints	<i>Struct with the gaze coordinates.</i>	
[1 x 1 struct]		
t_ms	[n x 1 double]	<i>Time vector, in milliseconds.</i>
XY	[n x 2 double]	<i>X and Y coordinates of n gaze points.</i>
fixationPoints	<i>Struct with the fixation coordinates</i>	
[1 x 1 struct]		
t_ms	[n x 1 double]	<i>Time vector, in milliseconds.</i>
XY	[n x 2 double]	<i>X and Y coordinates of n fixation points.</i>

RoI	<i>Struct array with information about k Regions of Interests (RoIs).</i>	
[k x 1 struct]		
name	[1 x n char]	<i>Name of the RoI.</i>
hit	[n x 2 double]	<i>Start and end times of the RoI hits, in milliseconds. One row per hit. Hits may not overlap.</i>

Table 21: The epochs variable inside the PhysioData file.

epochs	<i>The epochs variable holds a single field, epochData, which is a table containing pre-generated epochs. This variable can be omitted if not necessary.</i>	
[1 x 1 struct]		
epochData	<i>The epochData table is a m by n table containing m pre-generated epochs. The table may contain extra metadata columns, which will be copied to output file. All metadata columns must be either a single column double vector, or a single column cell array of strings. The 'duration', 'epochSource', 'epochError', and 'epochMessage' variables are automatically added by the toolbox, and should therefore not be present as variable names in the pre-generated epochData table.</i>	
[m x n table]		
startTime	[m x 1 double]	startTime
endTime	[m x 1 double]	endTime
epochName	[m x 1 cell]	epochName

Table 22: The data variable inside the PhysioData file.

physioDataInfo	<i>The physioDataInfo variable inside the PhysioData file holds metadata about the file. This information is displayed by the raw data viewer in the 'File Info' tab.</i>	
[1 x 1 struct]		
sourceFilename	[1 x n char]	<i>A character array indicating the source of the PhysioData file. The AcqKnowledge converter writes the original path of the .acq file to this field.</i>
extracionUser	[1 x n char]	<i>A character array indicating the user that generated the PhysioData file. The AcqKnowledge converter writes the windows username to this field.</i>
extracionDate	[1 x n char]	<i>A character array indicating the time and date of the PhysioData file creation.</i>

19 Toolbox User Analytics

The Toolbox, which refers to both the File Converter and the PhysioData Toolbox, sends anonymous usage data to an online analytics server for tracking how often the Toolbox and its various features are used. These data are necessary for rationalizing the continued development and maintenance of the Toolbox, its various components and the documentations. No data about the PhysioData files themselves are ever sent.

The data collected consist of the following:

- **User and Device IDs:** In order to track the number of unique user and devices, unique and anonymous user and device identification strings are generated by cryptographically hashing local user-account and device hardware parameters. The data sent are anonymous and do not contain personally identifiable information, nor can such information be retrieved from them.
- **File Converters usage:** In order to track which file types are most often converted, the File Converter collects: the file converter type used during a convert action (e.g. BIOPAC, Biosemi, etc.); how many files the action was performed on; and which overwrite option was selected.
- **PhysioData Toolbox usage:** To focus development on the most often used features, the PhysioData Toolbox collects: the executed analysis type (epoch summaries, resampling, or archiving); the PhysioAnalyzer types included in the analysis (e.g. Blood Pressure module, ECG module, etc.); the number of files selected for analysis; the saving format(s) selected (MATLAB, Excel or TSV); and, whether the saving succeeded. In addition, when using the resampling feature, the selected resampling rate is collected.

Agreeing to this anonymous data collection feature of the Toolbox is currently a prerequisite for its use. If you have any questions, please email the development team.

