signalanalysis

Release 0.1

Philip Gemmell

Oct 19, 2021

CONTENTS:

signalanalysis is a library including various tools for the reading, analysis and plotting of ECG and VCG data. It is designed to be as agnostic as possible for the types of data that it can read. Currently, it can read ECG data from:

1. CARP simulations of whole torso activity, using existing projects from *CARPutil <https://git.opencarp.org/openCARP/carputils>*;
2. .csv and .dat records;
3. wfdb file formats, using *wfdb-python <https://github.com/MIT-LCP/wfdb-python>*

Futher details of how to use these functions are in *Usage*, with the finer points within the files themselves.

CHAPTER

ONE

USAGE

1.1 Installation & Getting Started

To use signalanalysis, it is highly recommended to install using `pipenv`, the virtual environment, to ensure that dependencies are where possible maintained. Clone the repository, then install the requirements as follows:

```
user@home:~$ git clone git@github.com:philip-gemmell/signalanalysis.git
user@home:~$ cd signalanalysis
user@home:~/signalanalysis$ pipenv install
```

Once the repository is cloned, it is currently the case that all work must be done within the Python3 environment. However, it is recommended to use the virtual environment from pipenv rather than the system-wide Python3 (after entering a pipenv shell, it is quit using the `exit` command as shown)

```
user@home:~/signalanalysis$ pipenv shell
(signalanalysis) user@home:~/signalanalysis$ python3
>>> import signalanalysis
>>> quit()
(signalanalysis) user@home:~/signalanalysis$ exit
user@home:~/signalanalysis$
```

The project is arranged into various subdivisions. The required analysis/plotting packages for the ECG/VCG are separated out, and require separate importing. See the next section, and individual help files for further details.

- signalanalysis
  - signalanalysis.general
  - signalanalysis.ecg
  - signalanalysis.vcg
- signalplot
- tools

1.2 Reading ECG/VCG data

Currently, only ECG reading is supported; VCG data is calculated from ECG data using the Kors method (see method). ECG files are read upon the instantiation of the ECG class, though it is possible to re-read data if required for some reason.

```
>>> import signalanalysis.ecg
>>> import signalanalysis.vcg
>>> ecg_example = signalanalysis.ecg.Ecg("filename")
>>> vcg_example = signalanalysis.vcg.Vcg(ecg_example)
```

1.3 Shifting to classes from methods

Previously, all ECG/VCG data was extracted and stored in DataFrames, and most of the modules in this code currently support this format. However, it is planned to shift the main focus of the project to use classes, which allow encapsulation of linked data in one data structure. While the focus of this documentation will be future-facing, and look at using the classes, note that sometimes access to the raw, underlying DataFrames will still be required. To that end, the raw data can be accessed as the .data attribute:

```
>>> ecg_class = signalanalysis.ecg.Ecg("filename") # Returns an Ecg class
>>> vcg_class = signalanalysis.vcg.Vcg(ecg_class)  # Returns a Vcg class
>>> ecg_data = ecg_class.data                      # Returns a Pandas DataFrame of the␣
↪underlying data
>>> vcg_data = vcg_class.data                      # Returns a Pandas DataFrame of the␣
↪underlying data
```

CHAPTER

TWO

DOCUTILS DOCUMENTATION

This is the index of all modules and their associated methods and classes, documenting their use, parameters and return values. See *Usage* for a more step-by-step introduction to the intended use cases.

Broadly speaking, the modules are split thus:

- `signalanalysis` covers the analysis scripts for ECG/VCG analysis (e.g. calculating QRS duration)
- `signalplot` covers plotting methods (e.g. plotting the ECG leads on a single figure with annotation, plotting a 3D plot of VCG (including animation!))
- `tools` covers more general use tools that are not limited to ECG/VCG analysis.

2.1 signalanalysis

2.1.1 signalanalysis.ecg

Functions

get_ecg_from_electrodes	Converts electrode phi_e data to ECG lead data
get_electrode_phie	Extract phi_e data corresponding to ECG electrode locations
get_qrs_start	Calculates start of QRS complex using method of Hermans et al. (2017).

continues on next page

Table 3 – continued from previous page

read_ecg_from_csv	Extract ECG data from CSV file exported from St Jude Medical ECG recording
read_ecg_from_dat	Read ECG data from .dat file
read_ecg_from_igb	Translate the phie.igb file(s) to 10-lead, 12-trace ECG data

signalanalysis.ecg.get_ecg_from_electrodes

signalanalysis.ecg.**get_ecg_from_electrodes**(*electrode_data: pandas.core.frame.DataFrame*) → pandas.core.frame.DataFrame

Converts electrode phi_e data to ECG lead data

Takes dictionary of phi_e data for 10-lead ECG, and converts these data to standard ECG trace data

Parameters **electrode_data** (*pd.DataFrame*) – Dictionary with keys corresponding to lead locations

Returns **ecg** – Dictionary with keys corresponding to the ECG traces

Return type pd.DataFrame

signalanalysis.ecg.get_electrode_phie

signalanalysis.ecg.**get_electrode_phie**(*phie_data: numpy.ndarray*, *electrode_file: str*) → pandas.core.frame.DataFrame

Extract phi_e data corresponding to ECG electrode locations

Parameters

- **phie_data** (*np.ndarray*) – Numpy array that holds all phie data for all nodes in a given mesh
- **electrode_file** (*str*) – File containing entries corresponding to the nodes of the mesh which determine the location of the 10 leads for the ECG. Will default to very project specific location. The input text file has each node on a separate line (zero-indexed), with the node locations given in order: V1, V2, V3, V4, V5, V6, RA, LA, RL, LL.

Returns **electrode_data** – Dataframe of phie data for each node, with the dictionary key labelling which node it is.

Return type pd.DataFrame

Notes

For the .igb data used thus far, the *electrode_file* can be found at `tests/12LeadElectrodes.dat`

signalanalysis.ecg.get_qrs_start

signalanalysis.ecg.**get_qrs_start**(*ecgs: Union[pandas.core.frame.DataFrame, List[pandas.core.frame.DataFrame]]*, *unipolar_only: bool = True*, *plot_result: bool = False*) → List[float]

Calculates start of QRS complex using method of Hermans et al. (2017)

Calculates the start of the QRS complex by a simplified version of the work presented in[1], wherein the point of maximum second derivative of the ECG RMS signal is used as the start of the QRS complex

Parameters

- **ecgs** (*pd.DataFrame or list of pd.DataFrame*) – ECG data to analyse
- **unipolar_only** (*bool, optional*) – Whether to use only unipolar leads to calculate RMS, default=True
- **plot_result** (*bool, optional*) – Whether to plot the results for error-checking, default=False

Returns **qrs_starts** – QRS start times

Return type list of float

Notes

For further details of the action of unipolar_only, see general_analysis.get_signal_rms

It is faster to use scipy.ndimage.laplace() rather than np.gradient(np.gradient)), but preliminary checks indicated some edge problems that might throw off the results.

References

signalanalysis.ecg.read_ecg_from_csv

signalanalysis.ecg.**read_ecg_from_csv**(*filename: str*, *normalise: bool = False*) → pandas.core.frame.DataFrame

Extract ECG data from CSV file exported from St Jude Medical ECG recording

Parameters

- **filename** (*str*) – Name/location of the .dat file to read
- **normalise** (*bool, optional*) – Whether or not to normalise the ECG signals on a per-lead basis, default=True

Returns **ecg** – Extracted data for the 12-lead ECG

Return type list of pd.DataFrame

[1] Hermans BJM, Vink AS, Bennis FC, Filippini LH, Meijborg VMF, Wilde AAM, Pison L, Postema PG, Delhaas T, "The development and validation of an easy to use automatic QT-interval algorithm," PLoS ONE, 12(9), 1–14 (2017), https://doi.org/10.1371/journal.pone.0184352

Notes

Relies on the first line of the .csv labelling the 12 leads of the ECG, in the form ['I', 'II', 'III', 'aVR', 'aVL', 'aVF', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6'], and the time data under the label 't_ref'

signalanalysis.ecg.read_ecg_from_dat

signalanalysis.ecg.**read_ecg_from_dat**(*filename: str*, *normalise: bool = False*) → pandas.core.frame.DataFrame

Read ECG data from .dat file

Parameters

- **filename** (*str*) – Name/location of the .dat file to read
- **normalise** (*bool, optional*) – Whether or not to normalise the ECG signals on a per-lead basis, default=False

Returns **ecg** – Extracted data for the 12-lead ECG

Return type pd.DataFrame

signalanalysis.ecg.read_ecg_from_igb

signalanalysis.ecg.**read_ecg_from_igb**(*filename: str*, *electrode_file*, *dt: float*, *normalise: bool = False*) → pandas.core.frame.DataFrame

Translate the phie.igb file(s) to 10-lead, 12-trace ECG data

Extracts the complete mesh data from the phie.igb file using CARPutils, which contains the data for the body surface potential for an entire human torso, before then extracting only those nodes that are relevant to the 12-lead ECG, before converting to the ECG itself

Parameters

- **filename** (*str*) – Filename for the phie.igb data to extract
- **electrode_file** (*str*) – File which contains the node indices in the mesh that correspond to the placement of the leads for the 10-lead ECG. Default given in get_electrode_phie function.
- **dt** (*float*) – Time interval from which to construct the time data to associate with the ECG, default=0.002s (2ms)
- **normalise** (*bool, optional*) – Whether or not to normalise the ECG signals on a per-lead basis, default=False

Returns **ecgs** – DataFrame with Vm data for each of the labelled leads (the dictionary keys are the names of the leads)

Return type pd.DataFrame

Notes

For the .igb data used thus far, the *electrode_file* can be found at `tests/12LeadElectrodes.dat`, and *dt* is 0.002s

References

https://carpentry.medunigraz.at/carputils/generated/carputils.carpio.igb.IGBFile.html#carputils.carpio.igb.IGBFile

Classes

Ecg	Base class to encapsulate data regarding an ECG recording, inheriting from *signalanalysis.general.Signal*

signalanalysis.ecg.Ecg

class signalanalysis.ecg.**Ecg**(*filename: str*, ***kwargs*)

Bases: *signalanalysis.general.Signal*

Base class to encapsulate data regarding an ECG recording, inheriting from *signalanalysis.general.Signal*

See also:

signalanalysis.general.Signal

read(*filename*)
Reads in the data from the original file. Called upon initialisation

read_ecg_from_wfdb(*filename*, *normalise=False*)
Reads data from a WFDB type series of files, e.g. from the Lobachevsky ECG database (https://physionet.org/content/ludb/1.0.1/)

get_n_beats(*threshold=0.5*, *min_separation=0.2*)
Calculates the number of beats given in the recording

get_qrs_start()
Calculates the start of the QRS complex

Methods

apply_filter	Apply a given filter to the data, using their respective arguments as required
get_n_beats	Calculate the number of beats in a given signal, and save the individual beats to the object for later use
get_qrs_start	Calculates start of QRS complex using method of Hermans et al. (2017).
get_rms	Supplement the *signalanalysis.general.Signal.get_rms()* with *unipolar_only*

continues on next page

Table 5 – continued from previous page

get_twave_end	
plot	Method to plot the data of the ECG
read	Reads in the data from the original file.
read_ecg_from_wfdb	Read data from a waveform database file format
reset	Reset all properties of the class

apply_filter(***kwargs*)

Apply a given filter to the data, using their respective arguments as required

See also:

tools.maths.filter_butterworth() Potential filter

tools.maths.filter_savitzkygolay() Potential filter

get_n_beats(*threshold: float = 0.5*, *min_separation: float = 0.2*, *reset_index: bool = True*, *plot: bool = False*, ***kwargs*)

Calculate the number of beats in a given signal, and save the individual beats to the object for later use

When given the raw data of a given signal (ECG or VCG), will estimate the number of beats recorded in the trace based on the RMS of the signal exceeding a threshold value. The estimated individual beats will then be saved in a list in a lossless manner, i.e. saved as [ECG1, ECG2, . . . , ECG(n)], where ECG1=[0:peak2], ECG2=[peak1:peak3], . . . , ECGn=[peak(n-1):end]

Parameters

- **threshold** (*float {0<1}*) – Minimum value to search for for a peak in RMS signal to determine when a beat has occurred, default=0.5
- **min_separation** (*float*) – Minimum time (in s) that should be used to separate separate beats, default=0.2s
- **reset_index** (*bool*) – Whether to reset the time index for the separated beats so that they all start from zero (true), or whether to leave them with the original time index (false), default=True
- **plot** (*bool*) – Whether to plot results of beat detection, default=False
- **unipolar_only** (*bool, optional*) – Only appropriate for ECG data. Whether to use only unipolar ECG leads to calculate RMS, default=True

Returns **self.n_beats** – Number of beats detected in signal

Return type int

See also:

signalanalysis.general.Signal.get_rms() RMS signal calculation required for getting n_beats

get_qrs_start(*unipolar_only: bool = True*, *min_separation: float = 0.05*, *plot_result: bool = False*)

Calculates start of QRS complex using method of Hermans et al. (2017)

Calculates the start of the QRS complex by a simplified version of the work presented in[1], wherein the point of maximum second derivative of the ECG RMS signal is used as the start of the QRS complex

Parameters

[1] Hermans BJM, Vink AS, Bennis FC, Filippini LH, Meijborg VMF, Wilde AAM, Pison L, Postema PG, Delhaas T, "The development and validation of an easy to use automatic QT-interval algorithm," PLoS ONE, 12(9), 1–14 (2017), https://doi.org/10.1371/journal.pone.0184352

- **self** (Ecg) – ECG data to analyse
- **unipolar_only** (*bool, optional*) – Whether to use only unipolar leads to calculate RMS, default=True
- **min_separation** (*float, optional*) – Minimum separation from the peak used to detect various beats, default=0.05s
- **plot_result** (*bool, optional*) – Whether to plot the results for error-checking, default=False

Returns **self.qrs_start** – QRS start times

Return type list of float

Notes

For further details of the action of unipolar_only, see general_analysis.get_signal_rms

It is faster to use scipy.ndimage.laplace() rather than np.gradient(np.gradient)), but preliminary checks indicated some edge problems that might throw off the results.

References

get_rms(*preprocess_data: Optional[pandas.core.frame.DataFrame] = None*, *drop_columns: Optional[List[str]] = None*, *unipolar_only: bool = True*)

Supplement the *signalanalysis.general.Signal.get_rms()* with *unipolar_only*

Parameters

- **preprocess_data** (*pd.DataFrame, optional*) – See *signalanalysis.general.Signal.get_rms()*
- **drop_columns** (*list of str, optional*) – See *signalanalysis.general.Signal.get_rms()*
- **unipolar_only** (*optional*) – Whether to use only unipolar ECG leads to calculate RMS, default=True

See also:

signalanalysis.general.Signal.get_rms()

Notes

If *unipolar_only* is set to true, then ECG RMS is calculated using only 'unipolar' leads. This uses V1-6, and the non-augmented limb leads (VF, VL and VR)

$$VF = LL - V_{WCT} = \frac{2}{3}aVF$$

$$VL = LA - V_{WCT} = \frac{2}{3}aVL$$

$$VR = RA - V_{WCT} = \frac{2}{3}aVR$$

plot(*separate_beats: bool = False*, *separate_figs: bool = False*)
Method to plot the data of the ECG

Passes the function to the ECG plotting script, with the option to plot the individual beats instead of the entire signal; also, to plot these on individual figures or overlaid on the same figure

Parameters

- **separate_beats** (*bool, optional*) – Whether to plot the entire signal (false), or to plot the individual detected beats one after the other, default=False
- **separate_figs** (*bool, optional*) – When plotting the separate beats (if requested), whether to plot the individual beats one on top of the other on a single figure (true), or on separate figures (false), default=False

Returns

Return type fig, ax

read(*filename: str*, *normalise: bool = False*, ***kwargs*)
Reads in the data from the original file. Called upon initialisation

Parameters

- **filename** (*str*) – Location for data (either filename or directory)
- **normalise** (*bool, optional*) – Whether or not to normalise the individual leads (no real biophysical rationale for doing this, to be honest), default=False

See also:

signalanalysis.ecg.Ecg.read_ecg_from_wfdb underlying read method for wfdb files

read_ecg_from_wfdb(*filename: str*, *sample_rate: float*, *comments_file: Optional[str] = None*, *normalise: bool = False*)
Read data from a waveform database file format

Parameters

- **filename** (*str*) – Base filename for data (e.g. /path/to/data/1 will read /path/to/data/1.{avf, avl, avr, dat, hea}
- **sample_rate** (*float*) – Rate at which data is recorded, e.g. 500 Hz
- **comments_file** (*str, optional*) – .csv file containing additional data for the ECG, if available
- **normalise** (*bool, optional*) – Whether to normalise all individual leads, default=False

Notes

The waveform database format is used by several different ECG repositories on www.physionet.org, and the finer points of each import can be difficult to seamlessly integrate. Where values are subject to change between datasets, they are not available as defaults. The required values are thus, but it should be remembered that there remains a certain level of hard-coding (for example, in the use of *comments_file* to extract data).

- Lobachevsky
  - sample_rate = 500
- PTB-XL