Pysiology Documentation

Release 0.0.9 - Beta version

Giulio Gabrieli

Oct 29, 2019

Contents:

Version: 0.0.9.4

Github Pypi

CHAPTER 1

Introduction

Pysiology is a python package developed with the intent of make physiological analysis accessible to researchers worldwide.

PySiology is a tool designed with novice users in mind, and it provides a simple way to estimate features from ECG, EMG and EDA signals. Features estimation is done chaining functions, and for each method default value are provided. This allows for a fast estimation of more than 50 features. Furthermore, PySiology provides tutorials that allows not only to learn how to use the package, but also the methodologies behind its modules.

Expert users can customize each estimation parameter, add their own and share their results with other users.

CHAPTER 2

Installation

To install Pysiology on your machines, you can use pip or install it manually using the setup.py file.

Since Pysiology is based on Python, you will need Python installed on your machine as well.

This will install all the required dependencies as well.

2.1 Download and Installation

2.1.1 Using pip (suggested method)

Open a terminal and type:

```
pip install pysiology
```

2.1.2 Using setup.py

Download or clone the repository, extract it and open a terminal inside the root of the project. Then type:

```
python setup.py install
```

2.2 Testing

To test the installation of the package, run the file test.py or:

```
import pysiology
print(pysiology.__version__)
```

If everything went good, you should see the installed version.

CHAPTER 3

Modules

3.1 Electrocardiography analysis module

electrocardiography.**analyzeECG**(*rawECGSignal*, *samplerate*, *preprocessing=True*, *highpass=0.5*, *lowpass=2.5*, *min_dist=500*, *peakThresh=0.3*, *PeakTreshAbs=False*, *ibi=True*, *bpm=True*, *sdnn=True*, *sdsd=True*, *rmssd=True*, *pnn50=True*, *pnn20=True*, *pnn50pnn20=True*, *freqAnalysis=True*, *freqAnalysisFiltered=True*)

This is a simple entrypoint for ECG analysis.

You can use this function as model for your analysis or to extrapolate several features from an ECG signal.

You can specify which features to evaluate or to exclude, as well as cutoff for frequencies and filter.

Parameters

- **data** (*list*) – ECG signal
- **samplerate** (*int*) – samplerate of the signal in Hz
- **preprocessing** (*boolean*) – whether to perform a simple preprocessing of the signal automatically
- **highpass** (*boolean*) – cutoff frequency for the high pass filter
- **lowpass** (*boolean*) – cutoff frequency for the low pass filter
- **min_dist** (*int*) – minimum distance between peaks in ms. Used for peak detection
- **peakThresh** (*float*) – Normalized threshold. Only the peaks with amplitude higher than the threshold will be detected by the peak detection utils
- **PeakTreshAbs** (*boolean*) – If True, the peakThresh value will be interpreted as an absolute value, instead of a normalized threshold.
- **ibi** (*boolean*) – whether or not to perform the IBI analysis
- **bpm** (*boolean*) – whether or not to perform the BPM analysis

- **sdnn** (*boolean*) – whether or not to perform the sdnn analysis
- **sdsd** (*boolean*) – whether or not to perform the sdsd analysis
- **rmssd** (*boolean*) – whether or not to perform the rmssd analysis
- **pnn50** (*boolean*) – whether or not to perform the pNN50 analysis
- **pnn20** (*boolean*) – whether or not to perform the pNN20 analysis
- **pnn50pnn20** (*boolean*) – whether or not to perform the pNN50 on pNN20 ratio analysis
- **freqAnalysis** (*boolean*) – whether or not to perform a frequency analysis analysis
- **freqAnalysisFiltered** (*boolean*) – whether or not to perform a frequency analysis automatically filtering the signal

Returns a dictionary containing the results of the ECG analysis

Return type list

electrocardiography.**butter_highpass**(*cutoff*, *fs*, *order=5*)

This functions generates a higpass butter filter

Parameters

- **cutoff** (*float*) – cutoff frequency
- **cutoff** – cutoff frequency
- **fs** (*float*) – samplerate of the signal
- **order** (*int*) – order of the Butter Filter

Returns butter highpass filter

Return type list

electrocardiography.**butter_highpass_filter**(*data*, *cutoff*, *fs*, *order*)

This functions apply a butter highpass filter to a signal

Parameters

- **data** (*list*) – ECG signal
- **cutoff** (*float*) – cutoff frequency
- **cutoff** – cutoff frequency
- **fs** (*float*) – samplerate of the signal
- **order** (*int*) – order of the Butter Filter

Returns highpass filtered ECG signal

Return type list

electrocardiography.**butter_lowpass**(*cutoff*, *fs*, *order=5*)

This functions generates a lowpass butter filter

Parameters

- **cutoff** (*float*) – cutoff frequency
- **cutoff** – cutoff frequency
- **fs** (*float*) – samplerate of the signal
- **order** (*int*) – order of the Butter Filter

Returns butter lowpass filter

Return type list

electrocardiography.**butter_lowpass_filter**(*data*, *cutoff*, *fs*, *order*)

This functions apply a butter lowpass filter to a signal

Parameters

- **data** (*list*) – ECG signal
- **cutoff** (*float*) – cutoff frequency
- **cutoff** – cutoff frequency
- **fs** (*float*) – samplerate of the signal
- **order** (*int*) – order of the Butter Filter

Returns lowpass filtered ECG signal

Return type list

electrocardiography.**getBPM**(*npeaks*, *nsample*, *samplerate*)

This function returns the BPM of a discrete heart signal.

Input: number of peaks of the ECG signal,number of samples, samplerate of the signal

Output: BPM

Parameters

- **npeak** (*int*) – number of peaks of the ECG signal
- **nsample** (*int*) – number of samples of the ECG signal
- **samplerate** (*int*) – samplerate of the signal in Hz

Returns BPM of the ECG signal

Return type float

electrocardiography.**getFrequencies**(*rawECGSignal*, *samplerate*, *llc=0.04*, *ulc=0.15*, *lhc=0.15*, *uhc=0.4*, *lvlc=0.0033*, *hvlc=0.04*)

This functions returns the sum of the PSD of low Frequencies, high frequencies and very low frequencies.

Default Values have been adapted from: Blood, J. D., Wu, J., Chaplin, T. M., Hommer, R., Vazquez, L., Rutherford, H. J., ... & Crowley, M. J. (2015). The variable heart: high frequency and very low frequency correlates of depressive symptoms in children and adolescents. Journal of affective disorders, 186, 119-126.

It returns a dictionary with the values for high, low and very low frequencies

Parameters

- **rawECGSignal** (*list*) – raw ECG signal
- **samplerate** (*int*) – samplerate of the ECG signal
- **llc** (*float*) – lower cutoff of low frequencies
- **ulc** (*float*) – upper cutoff of low frequencies
- **lhc** (*float*) – lower cutoff of high frequencies
- **uhc** (*float*) – high cutoff of high frequencies
- **lvlc** (*float*) – lower cutoff of very low frequencies
- **uvlc** (*float*) – upper cutoff of very low frequencies

Returns a dictionary containing the results of the frequency analysis

Return type dictionary

electrocardiography.**getIBI**(*peaks*, *samplerate*)
This function returns the IBI of a discrete heart signal.

Input: peaks and samplerate of the ECG signal

Output: IBI in ms

Parameters

- **peaks** (*list*) – list of peaks of the ECG signal
- **samplerate** (*int*) – samplerate of the signal in Hz

Returns the mean IBI of the ECG signal

Return type IBI (in ms) as float value

electrocardiography.**getPNN20**(*peaks*, *samplerate*)
This functions evaluate pNN20, the proportion of differences greater than 20ms.

Input: peaks of the ECG signal,samplerate of the signal

Output: proportion of number of pairs of successive peaks that diffear by more than 20ms

Parameters

- **peaks** (*list*) – list of peaks in the ECG signal
- **samplerate** (*int*) – samplerate of the signal in Hz

Returns the pNN20 of the ECG signal

Return type float

electrocardiography.**getPNN50**(*peaks*, *samplerate*)
This functions evaluate pNN50, the proportion of differences greater than 50ms.

Input: peaks of the ECG signal,samplerate of the signal

Output: proportion of number of pairs of successive peaks that diffear by more than 50ms

Parameters

- **peaks** (*list*) – list of peaks in the ECG signal
- **samplerate** (*int*) – samplerate of the signal in Hz

Returns the pNN50 of the ECG signal

Return type float

electrocardiography.**getRMSSD**(*peaks*, *samplerate*)
This functions evaluate the root mean square of successive differences between adjacent R-R intervals.

RMSSD = sqrt((1 / (N - 1)) * sum(i=1 –> N)(RRdiff i - mean(RRdiff))**2)

Input: peaks of the ECG signal,samplerate of the signal.

Output: the root mean square of successive differences between adjacent R-R intervals.

Parameters

- **peaks** (*list*) – list of peaks in the ECG signal
- **samplerate** (*int*) – samplerate of the signal in Hz

Returns the RMSSD of the ECG signal

Return type float

electrocardiography.**getSDNN**(*peaks*, *samplerate*)

This functions evaluate the standard deviation of intervals between heartbeats. It is often calculated over 24h period, or over short peridos of 5 mins.

SDNN reflects all the cyclic components responsible for variability in the period of recording, therefore it represents total variability

SDNN = sqrt((1/N-1) * sum(i=1 –> N)(rri - rrmean)^2)

Input: peaks of the ECG signal,samplerate of the signal

Output: standard deviations of Intervals between heartbeats.

Parameters

- **peaks** (*list*) – list of peaks in the ECG signal
- **samplerate** (*int*) – samplerate of the signal in Hz

Returns the SDNN of the ECG signal

Return type float

electrocardiography.**getSDSD**(*peaks*, *samplerate*)

This functions evaluate the the standard deviation of successive differences between adjacent R-R intervals.

SDSD: sqrt((1 / (N - 1)) * sum(i=1 –> N)(RR i - mean(RR))**2)

Input: peaks of the ECG signal,samplerate of the signal

Output: the standard deviation of successive differences between adjacent R-R intervals

Parameters

- **peaks** (*list*) – list of peaks in the ECG signal
- **samplerate** (*int*) – samplerate of the signal in Hz

Returns the SDSD of the ECG signal

Return type float

3.2 Electrodermal Activity analysis module

electrodermalactivity.**GSRSCRFeaturesExtraction**(*filteredGSRSignal*, *samplerate*, *peak*)

This functions extract GSR SCR features: http://eda-explorer.media.mit.edu/static/SCR_withFeatures.png

- **Input:**
  - rawGSRSignal: filtered GSR Signal as list
  - samplerate: samplerate of the signak§
  - peak: list of peaks [peakStart, max, peakend]
- **Output:**
  - dict: {riseTime,Amplitude,EDAatApex,DecayTime (50%),SCRWidth (50%)}

Parameters

- **rawGSRSignal** (*list*) – raw GSR Signal

- **samplerate** (*int*) – samplerate of the GSR signal in Hz
- **peak** (*list*) – a list containing the peak onset, max and offset indexes (as returned by the function findPeakOnsetAndOffset)

Returns a dictionary with the results of the extracted features

Return type dict

electrodermalactivity.**analyzeGSR**(*rawGSRSignal*, *samplerate*, *preprocessing=True*, *lowpass=1*, *highpass=0.05*, *phasic_seconds=10*)

Entry point for gsr analysis. Signal is filtered and downsampled, then a phasic filter is applied

- **Input:**
  - rawGSRSignal = gsr signal as list
  - samplerate = samplerate of the signal
  - preprocessing = wether to perform or not a preprocessing of the signal
  - lowpass = cutoff for lowpass filter
  - highpass = cutoff for highpass filter
- **Output:**
  - dictionary with all the results

Parameters

- **rawGSRSignal** (*list*) – raw GSR Signal
- **samplerate** (*int*) – samplerate of the GSR signal in Hz
- **preprocessing** (*true*) – wether to perform or not an automatic preprocessing of the signal
- **lowpass** (*float*) – cutoff frequency for the lowpass filter
- **highpass** (*float*) – cutoff frequency for the highpass filter

Returns a dictionary with the results of the automatic GSR analysis

Return type dict

electrodermalactivity.**butter_highpass**(*cutoff*, *fs*, *order=5*)

This functions generates a higpass butter filter

Parameters

- **cutoff** (*float*) – cutoff frequency
- **cutoff** – cutoff frequency
- **fs** (*float*) – samplerate of the signal
- **order** (*int*) – order of the Butter Filter

Returns butter highpass filter

Return type list

electrodermalactivity.**butter_highpass_filter**(*data*, *cutoff*, *fs*, *order*)

This functions apply a butter highpass filter to a signal

Parameters

- **data** (*list*) – ECG signal
- **cutoff** (*float*) – cutoff frequency
- **cutoff** – cutoff frequency
- **fs** (*float*) – samplerate of the signal
- **order** (*int*) – order of the Butter Filter

Returns highpass filtered ECG signal

Return type list

electrodermalactivity.**butter_lowpass**(*cutoff*, *fs*, *order=5*)

This functions generates a lowpass butter filter

Parameters

- **cutoff** (*float*) – cutoff frequency
- **cutoff** – cutoff frequency
- **fs** (*float*) – samplerate of the signal
- **order** (*int*) – order of the Butter Filter

Returns butter lowpass filter

Return type list

electrodermalactivity.**butter_lowpass_filter**(*data*, *cutoff*, *fs*, *order*)

This functions apply a butter lowpass filter to a signal

Parameters

- **data** (*list*) – ECG signal
- **cutoff** (*float*) – cutoff frequency
- **cutoff** – cutoff frequency
- **fs** (*float*) – samplerate of the signal
- **order** (*int*) – order of the Butter Filter

Returns lowpass filtered ECG signal

Return type list

electrodermalactivity.**findPeakOnsetAndOffset**(*rawGSRSignal*, *onset=0.01*, *offset=0*)

This functions finds the peaks of a GSR signal

- **Input:**
  - rawGSRSignal = GSR signal as list
  - onset = onset for Peak Detection (uS)
  - offset = offset for Peak Detection (uS)
- **Output:**
  - multi dimensional list, [onsetIndex,maxIndex,offsetIndex] x nPeaks

Parameters

- **rawGSRSignal** (*list*) – GSR Signal to analyze
- **onset** (*float*) – onset value for peak detection (in uS)

- **offset** (*float*) – onset value for peak detection (in uS)

Returns list of the peaks in the signal

Return type float

electrodermalactivity.**getPhasicAndTonic**(*rawGSRSignal*, *samplerate*, *seconds=4*)

This function returns the phasic and tonic components of a singnal.

- **Input:**
  - rawGSRSignal = gsr signal as list
  - samplerate = samplerate of the signal
  - seconds = number of seconds before and after each timepoint to use in order to compute the filtered value
- **Output:**
  - List containing the phasic and tonic components

Parameters

- **rawGSRSignal** (*list*) – raw GSR Signal
- **samplerate** (*int*) – samplerate of the GSR signal in Hz
- **seconds** – seconds to use to apply the phasic filter
- **seconds** – int

Returns phasic and tonic signals

Return type list

electrodermalactivity.**phasicGSRFilter**(*rawGSRSignal*, *samplerate*, *seconds=4*)

Apply a phasic filter to the signal, with +- X seconds from each sample. Default is 4 seconds

- **Input:**
  - rawGSRSignal = gsr signal as list
  - samplerate = samplerate of the signal
  - seconds = number of seconds before and after each timepoint to use in order to compute the filtered value
- **Output:**
  - phasic signal

Parameters

- **rawGSRSignal** (*list*) – raw GSR Signal
- **samplerate** (*int*) – samplerate of the GSR signal in Hz
- **seconds** – seconds to use to apply the phasic filter
- **seconds** – int

Returns filtered signal

Return type list

electrodermalactivity.**tonicGSRFilter**(*rawGSRSignal*, *samplerate*, *seconds=4*)

Apply a modified filter to the signal, with +- X seconds from each sample, in order to extract the tonic component. Default is 4 seconds

- **Input:**
  - rawGSRSignal = gsr signal as list
  - samplerate = samplerate of the signal
  - seconds = number of seconds before and after each timepoint to use in order to compute the filtered value
- **Output:**
  - tonic signal

Parameters

- **rawGSRSignal** (*list*) – raw GSR Signal
- **samplerate** (*int*) – samplerate of the GSR signal in Hz
- **seconds** – seconds to use to apply the phasic filter
- **seconds** – int

Returns filtered signal

Return type list

3.3 Electromyography analysis module

electromyography.**analyzeEMG**(*rawEMGSignal*, *samplerate*, *preprocessing=True*, *lowpass=50*, *highpass=20*, *threshold=0.01*, *nseg=3*, *phasic_seconds=4*)

This functions acts as entrypoint for the EMG Analysis.

- **Input:**
  - rawEMGSignal = raw signal as list
  - samplerate = samplerate of the signal
  - lowpass = lowpass cutoff in Hz
  - highpass = highpass cutoff in Hz
  - threshold for the evaluation of ZC,MYOP,WAMP,SSC
  - nseg = number of segments for MAVSLPk, MHW,MTW
- **Output:**
  - results dictionary

electromyography.**butter_highpass**(*cutoff*, *fs*, *order=5*)

This functions generates a higpass butter filter

Parameters

- **cutoff** (*float*) – cutoff frequency
- **cutoff** – cutoff frequency
- **fs** (*float*) – samplerate of the signal

- **order** (*int*) – order of the Butter Filter

Returns butter highpass filter

Return type list

electromyography.**butter_highpass_filter**(*data*, *cutoff*, *fs*, *order*)
This functions apply a butter highpass filter to a signal

Parameters

- **data** (*list*) – ECG signal
- **cutoff** (*float*) – cutoff frequency
- **cutoff** – cutoff frequency
- **fs** (*float*) – samplerate of the signal
- **order** (*int*) – order of the Butter Filter

Returns highpass filtered ECG signal

Return type list

electromyography.**butter_lowpass**(*cutoff*, *fs*, *order=5*)
This functions generates a lowpass butter filter

Parameters

- **cutoff** (*float*) – cutoff frequency
- **cutoff** – cutoff frequency
- **fs** (*float*) – samplerate of the signal
- **order** (*int*) – order of the Butter Filter

Returns butter lowpass filter

Return type list

electromyography.**butter_lowpass_filter**(*data*, *cutoff*, *fs*, *order*)
This functions apply a butter lowpass filter to a signal

Parameters

- **data** (*list*) – ECG signal
- **cutoff** (*float*) – cutoff frequency
- **cutoff** – cutoff frequency
- **fs** (*float*) – samplerate of the signal
- **order** (*int*) – order of the Butter Filter

Returns lowpass filtered ECG signal

Return type list

electromyography.**getAAC**(*rawEMGSignal*)
Get the Average amplitude change.:

```
AAC = 1/N * sum(|x(i+1) - xi|) for i = 1 --> N-1
```

- **Input:**
  - raw EMG Signal as list

- **Output:**
  - Average amplitude change of the signal

Parameters **rawEMGSignal** (*list*) – the raw EMG signal

Returns Average Amplitude Change of the signal

Return type float

electromyography.**getAFB**(*rawEMGSignal*, *samplerate*, *windowSize=32*)

Get the amplitude at first Burst.

Reference: Du, S., & Vuskovic, M. (2004, November). Temporal vs. spectral approach to feature extraction from prehensile EMG signals. In Information Reuse and Integration, 2004. IRI 2004. Proceedings of the 2004 IEEE International Conference on (pp. 344-350). IEEE.

- **Input:**
  - rawEMGSignal as list
  - samplerate of the signal in Hz (sample / s)
  - windowSize = window size in ms
- **Output:**
  - amplitude at first burst

Parameters

- **rawEMGSignal** (*list*) – the raw EMG signal
- **samplerate** (*int*) – samplerate of the signal int Hz
- **windowSize** – window size in ms to use for the analysis

Returns Amplitute ad first Burst

Return type float

electromyography.**getDASDV**(*rawEMGSignal*)

Get the standard deviation value of the the wavelength.:

```
DASDV = sqrt( (1 / (N-1)) * sum((x[i+1] - x[i])**2 ) for i = 1 --> N - 1
```

- **Input:**
  - raw EMG Signal
- **Output:**
  - DASDV

Parameters **rawEMGSignal** (*list*) – the raw EMG signal

Returns standard deviation value of the the wavelength

Return type float

electromyography.**getFR**(*rawEMGPowerSpectrum*, *frequencies*, *llc=30*, *ulc=250*, *lhc=250*, *uhc=500*)

This functions evaluate the frequency ratio of the power spectrum.

Cut-off value can be decidec experimentally or from the MNF Feature See: Oskoei, M.A., Hu, H. (2006). GA-based feature subset selection for myoelectric classification.

- **Input:**
  - raw EMG power spectrum as list,
  - frequencies as list,
  - llc = lower low cutoff
  - ulc = upper low cutoff
  - lhc = lower high cutoff
  - uhc = upper high cutoff
- **Output:**
  - Frequency Ratio

Parameters

- **rawEMGPowerSpectrum** (*list*) – power spectrum of the EMG signal
- **frequencies** (*list*) – frequencies of the PSD
- **llc** (*float*) – lower cutoff frequency for the low frequency components
- **ulc** (*float*) – upper cutoff frequency for the low frequency components
- **lhc** (*float*) – lower cutoff frequency for the high frequency components
- **uhc** (*float*) – upper cutoff frequency for the high frequency components

Returns frequencies ratio of the EMG power spectrum

Return type float

electromyography.**getHIST**(*rawEMGSignal*, *nseg=9*, *threshold=50*)

Histograms is an extension version of ZC and WAMP features.

- **Input:**
  - raw EMG Signal as list
  - nseg = number of segment to analyze
  - threshold = threshold to use to avoid DC fluctuations
- **Output:**
  - get zc/wamp for each segment

Parameters

- **rawEMGSignal** (*list*) – the raw EMG signal
- **nseg** (*int*) – number of segments to analyze
- **threshold** (*int*) – value to sum / substract to the zero when evaluating the crossing.

Returns Willison amplitude

Return type float

electromyography.**getIEMG**(*rawEMGSignal*)
This function compute the sum of absolute values of EMG signal Amplitude.:

```
IEMG = sum(|xi|) for i = 1 --> N
```

- **Input:**
  - raw EMG Signal as list
- **Output:**
  - integrated EMG

Parameters **rawEMGSignal** (*list*) – the raw EMG signal

Returns the IEMG of the EMG Signal

Return type float

electromyography.**getLOG**(*rawEMGSignal*)
LOG is a feature that provides an estimate of the muscle contraction force.:

```
LOG = e^((1/N) * sum(|xi|)) for x i = 1 --> N
```

- **Input:**
  - raw EMG Signal
- Output = * LOG

Parameters **rawEMGSignal** (*list*) – the raw EMG signal

Returns LOG feature of the EMG Signal

Return type float

electromyography.**getMAV**(*rawEMGSignal*)
Thif functions compute the average of EMG signal Amplitude.:

```
MAV = 1/N * sum(|xi|) for i = 1 --> N
```

- **Input:**
  - raw EMG Signal as list
- **Output:**
  - Mean Absolute Value

Parameters **rawEMGSignal** (*list*) – the raw EMG signal

Returns the MAV of the EMG Signal

Return type float

electromyography.**getMAV1**(*rawEMGSignal*)
This functoin evaluate Average of EMG signal Amplitude, using the modified version n°.1.:

```
IEMG = 1/N * sum(wi|xi|) for i = 1 --> N
wi = {
      1 if 0.25N <= i <= 0.75N,
      0.5 otherwise
      }
```

- **Input:**
  - raw EMG Signal as list
- **Output:**
  - Mean Absolute Value

Parameters **rawEMGSignal** (*list*) – the raw EMG signal

Returns the MAV (modified version n. 1) of the EMG Signal

Return type float

electromyography.**getMAV2**(*rawEMGSignal*)

This functoin evaluate Average of EMG signal Amplitude, using the modified version n°.2.:

```
IEMG = 1/N * sum(wi|xi|) for i = 1 --> N
wi = {
      1 if 0.25N <= i <= 0.75N,
      4i/N if i < 0.25N
      4(i-N)/N otherwise
      }
```

- **Input:**
  - raw EMG Signal as list
- **Output:**
  - Mean Absolute Value

Parameters **rawEMGSignal** (*list*) – the raw EMG signal

Returns the MAV (modified version n. 2) of the EMG Signal

Return type float

electromyography.**getMAVSLPk**(*rawEMGSignal*, *nseg*)

Mean Absolute value slope is a modified versions of MAV feature.

The MAVs of adiacent segments are determinated.

```
MAVSLPk = MAV[k+1] - MAV[k]; k = 1,..,k+1
```

- **Input:**
  - raw EMG signal as list
  - nseg = number of segments to evaluate
- **Output:**
  - list of MAVs

Parameters

- **rawEMGSignal** (*list*) – the raw EMG signal
- **nseg** (*int*) – number of segments to evaluate

Returns Mean absolute slope value

Return type float

electromyography.**getMDF**(*rawEMGPowerSpectrum*, *frequencies*)

Obtain the Median Frequency of the PSD.

MDF is a frequency at which the spectrum is divided into two regions with equal amplitude, in other words, MDF is half of TTP feature

- **Input:**
  - raw EMG Power Spectrum
  - frequencies
- **Output:**
  - Median Frequency (Hz)

Parameters

- **rawEMGPowerSpectrum** (*list*) – power spectrum of the EMG signal
- **frequencies** (*list*) – frequencies of the PSD

Returns median frequency of the EMG power spectrum

Return type float

electromyography.**getMNF**(*rawEMGPowerSpectrum*, *frequencies*)

Obtain the mean frequency of the EMG signal, evaluated as the sum of product of the EMG power spectrum and the frequency divided by total sum of the spectrum intensity:

```
MNF = sum(fPj) / sum(Pj) for j = 1 -> M
M = length of the frequency bin
Pj = power at freqeuncy bin j
fJ = frequency of the spectrum at frequency bin j
```

- **Input:**
  - rawEMGPowerSpectrum: PSD as list
  - frequencies: frequencies of the PSD spectrum as list
- **Output:**
  - Mean Frequency of the PSD

Parameters

- **rawEMGPowerSpectrum** (*list*) – power spectrum of the EMG signal
- **frequencies** (*list*) – frequencies of the PSD

Returns mean frequency of the EMG power spectrum

Return type float

electromyography.**getMNP**(*rawEMGPowerSpectrum*)

This functions evaluate the mean power of the spectrum.:

```
Mean Power = sum(Pj) / M, j = 1 --> M, M = len of the spectrum
```

- **Input:**
  - EMG power spectrum
- **Output:**
  - mean power

Parameters

- **rawEMGPowerSpectrum** (*list*) – power spectrum of the EMG signal
- **frequencies** (*list*) – frequencies of the PSD

Returns mean power of the EMG power spectrum

Return type float

electromyography.**getMYOP**(*rawEMGSignal*, *threshold*)

The myopulse percentage rate (MYOP) is an average value of myopulse output. It is defined as one absolute value of the EMG signal exceed a pre-defined thershold value.

```
MYOP = (1/N) * sum(|f(xi)|) for i = 1 --> N
f(x) = {
        1 if x >= threshold
        0 otherwise
}
```

- **Input:**
  - rawEMGSignal = EMG signal as list
  - threshold = threshold to avoid fluctuations caused by noise and low voltage fluctuations
- **Output:**
  - Myopulse percentage rate

Parameters

- **rawEMGSignal** (*list*) – the raw EMG signal
- **threshold** (*int*) – value to sum / substract to the zero when evaluating the crossing.

Returns Myopulse percentage rate of the signal

Return type float

electromyography.**getPSR**(*rawEMGPowerSpectrum*, *frequencies*, *n=20*, *fmin=10*, *fmax=500*)

This function computes the Power Spectrum Ratio of the signal, defined as: Ratio between the energy P0 which is nearby the maximum value of the EMG power spectrum and the energy P which is the whole energy of the EMG power spectrum

- **Input:**
  - EMG power spectrum
  - frequencies as list

- n = range around f0 to evaluate P0
- fmin = min frequency
- fmax = max frequency

Parameters

- **rawEMGPowerSpectrum** (*list*) – power spectrum of the EMG signal
- **frequencies** (*list*) – frequencies of the PSD
- **n** (*int*) – range of frequencies around f0 to evaluate
- **fmin** (*int*) – min frequency to evaluate
- **fmax** (*int*) – lmaximum frequency to evaluate

Returns Power spectrum ratio of the EMG power spectrum

Return type float

electromyography.**getPeakFrequency**(*rawEMGPowerSpectrum*, *frequencies*)

Obtain the frequency at which the maximum peak occur

- **Input:**
  - raw EMG Power Spectrum as list
  - frequencies as list
- **Output:**
  - frequency in Hz

Parameters

- **rawEMGPowerSpectrum** (*list*) – power spectrum of the EMG signal
- **frequencies** (*list*) – frequencies of the PSD

Returns peakfrequency of the EMG Power spectrum

Return type float

electromyography.**getRMS**(*rawEMGSignal*)

Get the root mean square of a signal.:

```
RMS = (sqrt( (1 / N) * sum(xi**2))) for i = 1 --> N
```

- **Input:**
  - raw EMG Signal as list
- **Output:**
  - Root mean square of the signal

Parameters **rawEMGSignal** (*list*) – the raw EMG signal

Returns Root mean square of the EMG signal

Return type float

electromyography.**getSM**(*rawEMGPowerSpectrum*, *frequencies*, *order*)

Get the spectral moment of a spectrum:

```
SM = sum(fj*(Pj**order)), j = 1 --> M
```

- **Input:**
  - raw EMG Power Spectrum
  - frequencies as list
  - order (int)
- **Output:**
  - SM of order = order

Parameters

- **rawEMGPowerSpectrum** (*list*) – power spectrum of the EMG signal
- **frequencies** (*list*) – frequencies of the PSD
- **order** (*int*) – order to the moment

Returns Spectral moment of order X of the EMG power spectrum

Return type float

electromyography.**getSSC**(*rawEMGSignal*, *threshold*)

Number of times the slope of the EMG signal changes sign.:

```
SSC = sum(f( (x[i] - x[i-1]) X (x[i] - x[i+1]))) for i = 2 --> n-1

f(x){
    1 if x >= threshold
    0 otherwise
}
```

- **Input:**
  - raw EMG Signal
- **Output:**
  - number of Slope Changes

Parameters

- **rawEMGSignal** (*list*) – the raw EMG signal
- **threshold** (*int*) – value to sum / substract to the zero when evaluating the crossing.

Returns Number of slope's sign changes

Return type int

electromyography.**getSSI**(*rawEMGSignal*)

This function compute the summation of square values of the EMG signal.:

```
SSI = sum(xi**2) for i = 1 --> N
```

- **Input:**
  - raw EMG Signal as list
- **Output:**
  - Simple Square Integral

Parameters **rawEMGSignal** (*list*) – the raw EMG signal

Returns SSI of the signal

Return type float

electromyography.**getTM**(*rawEMGSignal*, *order*)

This function compute the Temporal Moment of order X of the EMG signal.:

```
TM = (1 / N * sum(xi**order) for i = 1 --> N
```

- **Input:**
  - raw EMG Signal as list
- **Output:**
  - TM of order = order

Parameters

- **rawEMGSignal** (*list*) – the raw EMG signal
- **order** (*int*) – order the the TM function

Returns Temporal Moment of order X of the EMG signal

Return type float

electromyography.**getTTP**(*rawEMGPowerSpectrum*)

This functions evaluate the aggregate of the EMG power spectrum (aka Zero Spectral Moment)

- **Input:**
  - raw EMG Power Spectrum
- **Output:**
  - Total Power

Parameters

- **rawEMGPowerSpectrum** (*list*) – power spectrum of the EMG signal
- **frequencies** (*list*) – frequencies of the PSD

Returns total power of the EMG power spectrum

Return type float

electromyography.**getVAR**(*rawEMGSignal*)

Summation of average square values of the deviation of a variable.:

```
VAR = (1 / (N - 1)) * sum(xi**2) for i = 1 --> N
```

- **Input:**
  - raw EMG Signal as list
- **Output:**
  - Summation of the average square values

Parameters **rawEMGSignal** (*list*) – the raw EMG signal

Returns the VAR of the EMG Signal

Return type float

electromyography.**getVCF**(*SM0*, *SM1*, *SM2*)

This function evaluate the variance of the central freuency of the PSD.:

```
VCF = (1 / SM0)*sum(Pj*(fj - fc)**2),j = 1 --> M, = SM2 / SM0 - (SM1 /SM0) **2
```

- **Input:**
  - SM0: spectral moment of order 0
  - SM1: spectral moment of order 1
  - SM2: spectral moment of order 0
- **Output:**
  - Variance of Central frequency of the Power spectrum

Parameters

- **SM0** (*float*) – Spectral moment of order 0
- **SM1** (*float*) – Spectral moment of order 1
- **SM2** (*float*) – Spectral moment of order 2

Returns Variance of central frequency

Return type float

electromyography.**getWAMP**(*rawEMGSignal*, *threshold*)

Wilson or Willison amplitude is a measure of frequency information. It is a number of time resulting from difference between the EMG signal of two adjoining segments, that exceed a threshold.:

```
WAMP = sum( f(|x[i] - x[i+1]|)) for n = 1 --> n-1
f(x){
    1 if x >= threshold
    0 otherwise
}
```

- **Input:**
  - rawEMGSignal = EMG signal as list
  - threshold = threshold to avoid fluctuations caused by noise and low voltage fluctuations
- **Output:**
  - Wilson Amplitude value

Parameters

- **rawEMGSignal** (*list*) – the raw EMG signal
- **threshold** (*int*) – value to sum / substract to the zero when evaluating the crossing.

Returns Willison amplitude

Return type float

electromyography.**getWL**(*rawEMGSignal*)

Get the waveform length of the signal, a measure of complexity of the EMG Signal.:

```
WL = sum(|x(i+1) - xi|) for i = 1 --> N-1
```

- **Input:**
  - raw EMG Signal as list
- **Output:**
  - wavelength of the signal

Parameters **rawEMGSignal** (*list*) – the raw EMG signal

Returns Waveform length of the signal

Return type float

electromyography.**getZC**(*rawEMGSignal*, *threshold*)

How many times does the signal crosses the 0 (+-threshold).:

```
ZC = sum([sgn(x[i] X x[i+1]) intersecated |x[i] - x[i+1]| >= threshold]) for i =␣
↪1 --> N - 1
sign(x) = {
             1, if x >= threshold
             0, otherwise
         }
```

- **Input:**
  - rawEMGSignal = EMG signal as list
  - threshold = threshold to use in order to avoid fluctuations caused by noise and low voltage fluctuations
- **Output:**
  - ZC index

Parameters

- **rawEMGSignal** (*list*) – the raw EMG signal
- **threshold** (*int*) – value to sum / substract to the zero when evaluating the crossing.

Returns Number of times the signal crosses the 0 (+- threshold)

Return type float

electromyography.**phasicFilter**(*rawEMGSignal*, *samplerate*, *seconds=4*)

Apply a phasic filter to the signal, with +-4 seconds from each sample

- **Input:**
  - rawEMGSignal = emg signal as list
  - samplerate = samplerate of the signal
- **Output:**
  - phasic filtered signal

Parameters

- **rawEMGSignal** (*list*) – the raw EMG signal
- **samplerate** (*int*) – samplerate of the signal in Hz

Returns the phasic filtered signal

Return type list

CHAPTER 4

Tutorials

Quickstart v.0.0.9

CHAPTER 5

Cite

If you use PySiology in your work, please cite:

```
Gabrieli G., Azhari A., Esposito G. (2020) PySiology: A Python Package for␣
↪Physiological Feature Extraction. In: Esposito A., Faundez-Zanuy M., Morabito F.,␣
↪Pasero E. (eds) Neural Approaches to Dynamics of Signal Exchanges. Smart Innovation,
↪ Systems and Technologies, vol 151. Springer, Singapore
```

CHAPTER 6

References

6.1 Articles

6.1.1 ECG

1. Blood, J. D., Wu, J., Chaplin, T. M., Hommer, R., Vazquez, L., Rutherford, H. J. & Crowley, M. J. (2015). The variable heart: high frequency and very low frequency correlates of depressive symptoms in children and adolescents. Journal of affective disorders, 186, 119-126.
2. Cai, J., Liu, G., & Hao, M. (2009, July). The research on emotion recognition from ECG signal. In Information Technology and Computer Science, 2009. ITCS 2009. International Conference on (Vol. 1, pp. 497-500). IEEE.
3. Valderas, M. T., Bolea, J., Laguna, P., Vallverdú, M., & Bailón, R. (2015, August). Human emotion recognition using heart rate variability analysis with spectral bands based on respiration. In Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE (pp. 6134-6137). IEEE.
4. Ferdinando, H., Seppänen, T., & Alasaarela, E. (2016, October). Comparing features from ECG pattern and HRV analysis for emotion recognition system. In Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), 2016 IEEE Conference on (pp. 1-6). IEEE.

6.1.2 EDA

1. Lim, C. L., Rennie, C., Barry, R. J., Bahramali, H., Lazzaro, I., Manor, B., & Gordon, E. (1997). Decomposing skin conductance into tonic and phasic components. International Journal of Psychophysiology, 25(2), 97-109.
2. Chaspari, T., Tsiartas, A., Duker, L. I. S., Cermak, S. A., & Narayanan, S. S. (2016, August). EDA-Gram: Designing electrodermal activity fingerprints for visualization and feature extraction. In Engineering in Medicine and Biology Society (EMBC), 2016 IEEE 38th Annual International Conference of the (pp. 403-406). IEEE.
3. ALADAĞ, S., GÜVEN, A., Özbek, H., & DOLU, N. (2015). Elektrodermal Aktivite Sinyallerinde Gürültü Giderme Yöntemlerinin Karşılaştırılması A Comparison of Denoising Methods for Electrodermal Activity Signals. Vogue, 15(18).

4. Benedek, M., & Kaernbach, C. (2010). A continuous measure of phasic electrodermal activity. Journal of neuroscience methods, 190(1), 80-91.

5. Fowles, D. C., Christie, M. J., Edelberg, R., Grings, W. W., Lykken, D. T., & Venables, P. H. (1981). Publication recommendations for electrodermal measurements. Psychophysiology, 18(3), 232-239.

6.1.3 EMG

1. Phinyomark, A., Phukpattaranont, P., & Limsakul, C. (2012). Feature reduction and selection for EMG signal classification. Expert Systems with Applications, 39(8), 7420-7431.

2. Reaz, M. B. I., Hussain, M. S., & Mohd-Yasin, F. (2006). Techniques of EMG signal analysis: detection, processing, classification and applications. Biological procedures online, 8(1), 11.

3. Du, S., & Vuskovic, M. (2004, November). Temporal vs. spectral approach to feature extraction from prehensile EMG signals. In Information Reuse and Integration, 2004. IRI 2004. Proceedings of the 2004 IEEE International Conference on (pp. 344-350). IEEE.

4. Künecke, J., Hildebrandt, A., Recio, G., Sommer, W., & Wilhelm, O. (2014). Facial EMG responses to emotional expressions are related to emotion perception ability. PloS one, 9(1), e84053.

6.2 Websites

- http://www.paulvangent.com

6.3 Other resources

1. GSR Everything you need to know about Galvanic Skin Response to push your insights into emotional behavior, iMotions

CHAPTER 7

Indices

- genindex
- modindex
- search

Python Module Index

e

Index

P

T