pobm Documentation

Author

Jan 20, 2021

POBM API REFERENCE:

Oximetry digital biomarkers for the analysis of continuous oximetry (SpO2) time series.

Based on the paper Levy Jeremy, Álvarez Daniel, Rosenberg Aviv A., del Campo Felix and Behar Joachim A. "Oximetry digital biomarkers for assessing respiratory function during sleep: standards of measurement, physiological interpretation, and clinical use". Accepted for publication in NPJ Digital Medicine

CHAPTER 1

Description

Five types of biomarkers may be evaluated:

1. General statistics: time-based statistics describing the oxygen saturation time series data distribution.
2. Complexity: quantify the presence of long-range correlations in non-stationary time series.
3. Periodicity: quantify consecutive events creating some periodicity in the oxygen saturation time series.
4. Desaturations: time-based measures that are descriptive statistics of the desaturation patterns happening throughout the time series.
5. Hypoxic burden: time-based measures quantifying the overall degree of hypoxemia imposed to the heart and other organs during the recording period.

CHAPTER 2

Installation

Available on pip, with the command: pip install pobm

pip project: https://pypi.org/project/pobm/

CHAPTER 3

Requirements

numpy==1.18.2

scikit-learn==0.22.2

scipy==1.4.1

lempel-ziv-complexity==0.2.2

All the requirements are installed when the toolbox is installed, no need for additional commands.

CHAPTER 4

Documentation

Available at https://oximetry-toolbox.readthedocs.io/en/latest/

4.1 pobm package

4.1.1 pobm.obm package

4.1.1.1 pobm.obm.burden

class pobm.obm.burden.**HypoxicBurdenMeasures**(*begin: numpy.array*, *end: numpy.array*, *CT_Threshold: float = 90*, *CA_Baseline: float = None*)

Bases: `object`

Class that calculates hypoxic burden features from SpO2 time series.

Parameters

- **begin** (*Numpy array*) – Numpy array of indices of beginning of each desaturation event.
- **end** (*Numpy array*) – Numpy array of indices of end of each desaturation event. begin and end should have the same length.
- **CT_Threshold** (*float, optional*) – Percentage of the time spent below the "CT_Threshold" % oxygen saturation level.
- **CA_Baseline** (*float, optional*) – Baseline to compute the CA feature. Default value is mean of the signal.

compute(*signal*)

Computes all the biomarkers of this category.

Parameters **signal** – 1-d array, of shape (N,) where N is the length of the signal

Returns

HypoxicBurdenMeasuresResults class containing the following features:

- CA: Integral SpO2 below the xx SpO2 level normalized by the total recording time
- CT: Percentage of the time spent below the xx% oxygen saturation level
- POD: Percentage of oxygen desaturation events
- AODmax: The area under the oxygen desaturation event curve, using the maximum SpO2 value as baseline and normalized by the total recording time
- AOD100: Cumulative area of desaturations under the 100% SpO2 level as baseline and normalized by the total recording time

Example:

```
from pobm.obm.burden import HypoxicBurdenMeasures

# Initialize the class with the desired parameters
hypoxic_class = HypoxicBurdenMeasures(results_desat.begin, results_desat.end,␣
↪CT_Threshold=90, CA_Baseline=90)

# Compute the biomarkers
results_hypoxic = hypoxic_class.compute(spo2_signal)
```

comp_ca (*signal*)
Compute the cumulative area biomarker

Parameters **signal** – 1-d array, of shape (N,) where N is the length of the signal

Returns CA, the cumulative area (float)

comp_ct (*signal*)
Compute the cumulative time biomarker

Parameters **signal** – 1-d array, of shape (N,) where N is the length of the signal

Returns CT, the cumulative time (float)

4.1.1.2 pobm.obm.complex

class pobm.obm.complex.**ComplexityMeasures**(*CTM_Threshold: float = 0.25, DFA_Window: int = 20, M_Sampen: int = 3, R_Sampen: float = 0.2, M_ApEn: int = 2, R_ApEn: float = 0.25*)

Bases: object

Class that calculates complexity features from SpO2 time series.

Parameters

- **CTM_Threshold** (*float, optional*) – Radius of Central Tendency Measure.
- **DFA_Window** (*int, optional*) – Length of window to calculate DFA biomarker.
- **M_Sampen** (*int, optional*) – Embedding dimension to compute SampEn.
- **R_Sampen** (*float, optional*) – Tolerance to compute SampEn.
- **M_ApEn** (*int, optional*) – Embedding dimension to compute ApEn.
- **R_ApEn** (*float, optional*) – Tolerance to compute ApEn.

compute(*signal*) → pobm._ResultsClasses.ComplexityMeasuresResults
Computes all the biomarkers of this category. :param signal: 1-d array, of shape (N,) where N is the length of the signal :return: ComplexityMeasuresResults class containing the following features:

- ApEn: Approximate Entropy.
- LZ: Lempel-Ziv complexity.
- CTM: Central Tendency Measure.
- SampEn: Sample Entropy.
- DFA: Detrended Fluctuation Analysis.

Example:

comp_apen(*signal*)
Compute the approximate entropy, according to [1]_ :param signal: 1-d array, of shape (N,) where N is the length of the signal :return: ApEn (float) .. [1] Pincus, S. M. Approximate entropy as a measure of system complexity. Proc. Natl. Acad. Sci. U. S. A. 88, 2297–2301 (1991).

comp_lz(*signal*)
Compute Lempel-Ziv, according to [2]_ :param signal: 1-d array, of shape (N,) where N is the length of the signal :return: LZ (float) .. [2] Lempel, A. & Ziv, J. On the Complexity of Finite Sequences. IEEE Trans. Inf. Theory 22, 75–81 (1976).

comp_ctm(*signal*)
Compute CTM, according to [3]_ :param signal: 1-d array, of shape (N,) where N is the length of the signal :return: CTM (float) .. [3] Cohen, M. E., Hudson, D. L. & Deedwania, P. C. Applying continuous chaotic modeling to cardiac signal analysis. IEEE Eng. Med. Biol. Mag. 15, 97–102 (1996).

comp_sampen(*signal*)
Compute the sample entropy, according to [4]_ :param signal: 1-d array, of shape (N,) where N is the length of the signal :return: SampEn (float) .. [4] Richman, J. S. & Moorman, J. R. Physiological time-series analysis using approximate entropy and sample entropy. Am J Physiol-Heart C 278, H2039–H2049 (2000).

comp_dfa(*signal*)
Compute DFA, Detrended Fluctuation Analysis according to [5]_ :param signal: 1-d array, of shape (N,) where N is the length of the signal :return: DFA (float) .. [5] Peng, C. -K., Havlin, S., Stanley, H. E. & Goldberger, A. L. Quantification of scaling exponents and crossover phenomena in nonstationary heartbeat time series. Chaos An Interdiscip. J. Nonlinear Sci. 5, 82–87 (1995).

4.1.1.3 pobm.obm.desat

class pobm.obm.desat.**DesaturationsMeasures**(*ODI_Threshold: int = 3*, *hard_threshold: int = 90*, *relative: bool = True*, *desat_max_length: int = 90*)
Bases: object

Class that calculates the desaturation features from SpO2 time series.

Parameters

- **ODI_Threshold** (*int, optional*) – Threshold to compute Oxygen Desaturation Index.
- **hard_threshold** (*int, optional*) – Hard threshold to detect desaturations.
- **relative** (*bool, optional*) – Whether to use a relative or hard threshold to detect desaturations.

- **desat_max_length** (*int, optional*) – The maximum length of desaturations.

compute (*signal*) → pobm._ResultsClasses.DesaturationsMeasuresResults

Computes all the biomarkers of this category.

Parameters **signal** – 1-d array, of shape (N,) where N is the length of the signal

Returns

DesaturationsMeasuresResults class containing the following features:

- ODI: Oxygen Desaturation Index
- DL_u: Mean of desaturation length
- DL_sd: Standard deviation of desaturation length
- DA100_u: Mean of desaturation area using 100% as baseline.
- DA100_sd: Standard deviation of desaturation area using 100% as baseline
- DAmax_u: Mean of desaturation area using max value as baseline.
- DAmax_sd: Standard deviation of desaturation area using max value as baseline
- DD100_u: Mean of depth desaturation from 100%.
- DD100_sd: Standard deviation of depth desaturation from 100%.
- DDmax_u: Mean of depth desaturation from max value.
- DDmax_sd: Standard deviation of depth desaturation from max value.
- DS_u: Mean of the desaturation slope.
- DS_sd: Standard deviation of the desaturation slope.
- TD_u: Mean of time between two consecutive desaturation events.
- TD_sd: Standard deviation of time between 2 consecutive desaturation events.
- begin: List of indices of beginning of each desaturation event
- end: List of indices of end of each desaturation event

Example:

```
from pobm.obm.desat import DesaturationsMeasures

# Initialize the class with the desired parameters
desat_class = DesaturationsMeasures(ODI_Threshold=3)

# Compute the biomarkers
results_desat = desat_class.compute(spo2_signal)
```

desaturation_detector (*signal*)

Runs desaturation detector, compute the ODI biomarker according to[6]

Parameters **signal** – The SpO2 signal, of shape (N,)

Returns ODI: the average number of desaturation events per hour (int).

[6] Jung, D. W. et al. Real-Time Automatic Apneic Event Detection Using Nocturnal Pulse Oximetry. IEEE Trans. Biomed. Eng. 65, 706–712 (2018).

4.1.1.4 pobm.obm.general

class pobm.obm.general.**OverallGeneralMeasures**(*ZC_Baseline: float = None, percentile: int = 1, M_Threshold: int = 2, DI_Window: int = 12*)

Bases: object

Class that calculates overall general features from SpO2 time series.

Parameters

- **ZC_Baseline** (*int, optional*) – Baseline for calculating number of zero-crossing points.
- **percentile** (*int, optional*) – Percentile to perform. For example, for percentile 1, the argument should be 1
- **M_Threshold** (*int, optional*) – Percentage of the signal M_Threshold % below median oxygen saturation. Typically use 1,2 or 5
- **DI_Window** (*int, optional*) – Length of window to calculate the Delta Index.

compute(*signal*) → pobm._ResultsClasses.OverallGeneralMeasuresResult

Computes all the biomarkers of this category.

Parameters **signal** – 1-d array, of shape (N,) where N is the length of the signal

Returns

OveralGeneralMeasuresResult class containing the following features:

- AV: Average of the signal.
- MED: Median of the signal.
- Min: Minimum value of the signal.
- SD: Std of the signal.
- RG: SpO2 range (difference between the max and min value).
- P: percentile.
- M: Percentage of the signal x% below median oxygen saturation.
- ZC: Number of zero-crossing points.
- DI: Delta Index.

Example:

```
from pobm.obm.general import OverallGeneralMeasures

# Initialize the class with the desired parameters
statistics_class = OverallGeneralMeasures(ZC_Baseline=90, percentile=1, M_
↪Threshold=2, DI_Window=12)

# Compute the biomarkers
results_statistics = statistics_class.compute(spo2_signal)
```

4.1.1.5 pobm.obm.periodicity

class pobm.obm.periodicity.**PRSAMeasures**(*PRSA_Window: int = 10, K_AC: int = 2*)

Bases: object

Class that calculates PRSA features from SpO2 time series.

Parameters

- **PRSA_Window** (*int, optional*) – Fragment duration of PRSA.
- **K_AC** (*int, optional*) – Number of values to shift when computing autocorrelation

compute (*signal*) → pobm._ResultsClasses.PRSAResults
Computes all the biomarkers of this category.

Parameters **signal** – 1-d array, of shape (N,) where N is the length of the signal

Returns

PRSAResults class containing the following features:

- PRSAc: PRSA capacity.
- PRSAad: PRSA amplitude difference.
- PRSAos: PRSA overall slope.
- PRSAsb: PRSA slope before the anchor point.
- PRSAsa: PRSA slope after the anchor point.
- AC: Autocorrelation.

Example:

```
from pobm.obm.periodicity import PRSAMeasures

# Initialize the class with the desired parameters
prsa_class = PRSAMeasures(PRSA_Window=10, K_AC=2)

# Compute the biomarkers
results_PRSA = prsa_class.compute(spo2_signal)
```

class pobm.obm.periodicity.**PSDMeasures** (*frequency_low_threshold: float = 0.014, frequency_high_threshold: float = 0.033*)

Bases: object

Class that calculates PSD features from SpO2 time series.

Parameters

- **frequency_low_threshold** (*float, optional*) – Low threshold for the PSD_band biomarker.
- **frequency_high_threshold** (*float, optional*) – High threshold for the PSD_band biomarker.

compute (*signal*) → pobm._ResultsClasses.PSDResults
Computes all the biomarkers of this category.

Parameters **signal** – The SpO2 signal, of shape (N,)

Returns

PSDResults class containing the following features:

- PSD_total: The amplitude of the spectral signal.
- PSD_band: The amplitude of the signal multiplied by a band-pass filter in the desired band.

- PSD_ratio: The ratio between PSD_total and PSD_band.
- PDS_peak: The max value of the FFT into the desired band.

Example:

```
from pobm.obm.periodicity import PSDMeasures

# Initialize the class with the desired parameters
psd_class = PSDMeasures()

# Compute the biomarkers
results_PSD = psd_class.compute(spo2_signal)
```

4.1.1.6 Module contents

4.1.2 pobm.prep

pobm.prep.**set_range**(*signal*, *Range_min=50*, *Range_max=100*)

Range function. Remove values lower than Range_min or greater than Range_max, considered as non-physiological

Parameters

- **signal** – 1-d array, of shape (N,) where N is the length of the signal
- **Range_min** (*int, optional*) – minimum value for removing the data
- **Range_max** (*int, optional*) – maximum value for removing the data

Returns preprocessed signal, 1-d numpy array.

pobm.prep.**resamp_spo2**(*signal*, *OriginalFreq*)

Resample the SpO2 signal to 1Hz. Assumption: any missing/abnormal values are represented as 'np.nan'

Parameters

- **signal** – 1-d array, of shape (N,) where N is the length of the signal
- **OriginalFreq** – the original frequency.

Returns resampled signal, 1-d numpy array, the resampled spo2 time series at 1Hz

pobm.prep.**dfilter**(*signal*, *Diff=4*)

Apply Delta Filter to the signal.

Parameters

- **signal** – 1-d array, of shape (N,) where N is the length of the signal
- **Diff** (*int, optional*) – parameter of the delta filter.

Returns preprocessed signal, 1-d numpy array.

pobm.prep.**median_spo2**(*signal_spo2*, *FilterLength=9*)

Apply a median filter to the SpO2 signal. Median filter used to smooth the spo2 time series and avoid sporadic increase/decrease of spo2 which could affect the detection of the desaturations. Assumption: any missing/abnormal values are represented as 'np.nan'

Parameters

- **signal** – 1-d array, of shape (N,) where N is the length of the signal
- **FilterLength** (*int, optional*) – The length of the filter.

Returns preprocessed signal, 1-d numpy array.

pobm.prep.**block_data**(*signal*, *treshold=50*)
Apply a block data filter to the SpO2 signal.

Parameters

- **signal** – 1-d array, of shape (N,) where N is the length of the signal
- **treshold** (*int, optional*) – treshold parameter for block data filter.

Returns preprocessed signal, 1-d numpy array.

4.1.3 Module contents

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index