pobm Documentation

Author

Oct 18, 2020

CONTENTS:

CHAPTER

ONE

POBM PACKAGE

1.1 Subpackages

1.1.1 pobm.obm package

1.1.1.1 Submodules

1.1.1.2 pobm.obm.burden

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

Bases: object

Class that calculates Hypoxic Burden Features from spo2 time series. Suppose that the data has been preprocessed.

Parameters

- **begin** – List of indices of beginning of each desaturation event.
- **end** – List of indices of end of each desaturation event.
- **CT_Threshold** – Percentage of the time spent below the "CT_Threshold" % oxygen saturation level.
- **CA_Baseline** – Baseline to compute the CA feature. Default value is mean of the signal.

compute(*signal*)

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)
```

1.1.1.3 pobm.obm.complex

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

Bases: `object`

Class that calculates Complexity Features from spo2 time series. Suppose that the data has been preprocessed.

:param signal: 1-d array, of shape (N,) where N is the length of the signal CTM_Threshold: Radius of Central Tendency Measure. DFA_Window: Length of window to calculate DFA biomarker. M_Sampen: Embedding dimension to compute SampEn. R_Sampen: Tolerance to compute SampEn. M_ApEn: Embedding dimension to compute ApEn. R_ApEn: Tolerance to compute ApEn.

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

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

Returns

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:

```
from pobm.obm.complex import ComplexityMeasures

# Initialize the class with the desired parameters
complexity_class = ComplexityMeasures(CTM_Threshold=0.25, DFA_Window=20, M_
↪Sampen=3, R_Sampen=0.2, M_ApEn=2, R_ApEn=0.25)

# Compute the biomarkers
results_complexity = complexity_class.compute(spo2_signal)
```

1.1.1.4 pobm.obm.desat

class pobm.obm.desat.**DesaturationsMeasures**(*ODI_Threshold=3*)

Bases: `object`

Class that calculates the Desaturation Features from spo2 time series. Suppose that the data has been preprocessed.

Parameters **`ODI_Threshold`** – Threshold to compute Oxygen Desaturation Index.

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

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

Returns

DesaturationsMeasuresResults class containing the following features:

- 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.

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*)

run desaturation detector, implemented by Dr. Joachim Behar

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

Returns

ODIMeasureResult class containing the following features:

- ODI: the average number of desaturation events per hour.
- begin: List of indices of beginning of each desaturation event.

- end: List of indices of end of each desaturation event.

pobm.obm.desat.**desat_embedding**(*begin*, *end*)

Help function for the class

Returns helper arrays containing the information about desaturation lengths and areas.

1.1.1.5 pobm.obm.general

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

Bases: `object`

Class that calculates Overall General Features from spo2 time series. Suppose that the data has been preprocessed.

Parameters

- **ZC_Baseline** – Baseline for calculating number of zero-crossing points.
- **percentile** – Percentile to perform. For example, for percentile 1, the argument should be 1
- **M_Threshold** – Percentage of the signal M_Threshold % below median oxygen saturation. Typically use 1,2 or 5

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

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:

```python
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)
```

1.1.1.6 pobm.obm.periodicity

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

Bases: object

Function that calculates PRSA Features from spo2 time series. Suppose that the data has been preprocessed.

:param PRSA_Window: Fragment duration of PRSA. K_AC: Number of values to shift when computing autocorrelation

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

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**

Bases: object

Function that calculates PSD Features from spo2 time series. Suppose that the data has been preprocessed.

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

:param 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 between 0.014 and 0.033 Hz.
- PSD_ratio: The ratio between PSD_total and PSD_band.
- PDS_peak: The max value of the FFT into the band 0.014-0.033 Hz.

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)
```

1.1.1.7 Module contents

1.1.2 pobm.spo2 package

1.1.2.1 Submodules

1.1.2.2 pobm.spo2.single_biomarkers

`pobm.spo2.single_biomarkers.`**`apen`**(*signal*, *M_ApEn=2*, *R_ApEn=0.25*)

Compute the approximate entropy, according to the paper Utility of Approximate Entropy From Overnight Pulse Oximetry Data in the Diagnosis of the Obstructive Sleep Apnea Syndrome

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

Returns ApEn

`pobm.spo2.single_biomarkers.`**`dfa`**(*signal*, *DFA_Window=20*)

Compute DFA

Parameters

- **`signal`** – 1-d array, of shape (N,) where N is the length of the signal
- **`DFA_Window`** – Length of window to calculate DFA biomarker.

Returns DFA

`pobm.spo2.single_biomarkers.`**`lempel_ziv`**(*signal*)

Compute lempel-ziv, according to the paper Non-linear characteristics of blood oxygen saturation from nocturnal oximetry for obstructive sleep apnoea detection

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

Returns LZ

`pobm.spo2.single_biomarkers.`**`odi`**(*signal*, *ODI_Threshold=3*)

Calculates the ODI from spo2 time series. Suppose that the data has been preprocessed.

Parameters

- **`signal`** – The SpO2 signal, of shape (N,)
- **`ODI_Threshold`** – Threshold to compute Oxygen Desaturation Index.

Returns ODI

`pobm.spo2.single_biomarkers.`**`sampen`**(*signal*, *M_Sampen=3*, *R_Sampen=0.2*)

Compute the Sample Entropy

Parameters

- **`signal`** – 1-d array, of shape (N,) where N is the length of the signal
- **`M_Sampen`** – Embedding dimension to compute SampEn.

- **`R_Sampen`** – Tolerance to compute SampEn.

Returns SampEn

1.1.2.3 Module contents

1.2 Submodules

1.3 pobm.main

1.4 pobm.prep

`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
- **`(Optional)`** (*`treshold`*) – treshold parameter for block data filter.

Returns preprocessed signal, 1-d numpy array.

`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`** – 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
- **`(Optional)`** (*`FilterLength`*) – The length of the filter.

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

Range function. Remove values lower than 50 or greater than 100, considered as non-physiological

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

Returns preprocessed signal, 1-d numpy array.

1.5 Module contents

CHAPTER TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX