Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

TDSpy: An open-source implementation of time delay stability analysis

Tabea F.A. Steinbrinker [a], Dagmar Krefting [a,b,c], Ronny P. Bartsch [d], Jan W. Kantelhardt [e], Nicolai Spicher [a,b,c,*]

[a] *Department of Medical Informatics, University Medical Center Göttingen, Göttingen, Germany*
[b] *DZHK (German Centre for Cardiovascular Research), partner site Göttingen, Göttingen, Germany*
[c] *Campus Institute Data Science, Göttingen, Germany*
[d] *Department of Physics, Bar-Ilan University, Ramat Gan, Israel*
[e] *Institute of Physics, Martin-Luther-University Halle-Wittenberg, Halle, Germany*

ARTICLE INFO

Keywords:
Time delay stability
Multivariate time series analysis
Physiological time series
Electrocardiography
Electroencephalography
Network physiology

ABSTRACT

Time Delay Stability (TDS) is an established tool for analyzing the interaction between physiological systems in the human organism. Time series are measured with sensors from different organ systems and are analyzed pairwise. Each pair is characterized by a TDS link strength and by combining these to a network, insights into underlying physiological mechanisms can be obtained. Computing TDS is based on heuristic computations with multiple open parameters. In the past, research groups working with TDS have implemented their own algorithms in different programming languages, which posed the risk of differences between implementations and parameters, leading to a lack of reproducibility. Therefore, we propose a reference implementation written in Python 3, entitled TDSpython (*TDSpy*) that we make publicly available via the Python Package Index (PyPI). In this paper, we give a comprehensive description of the implementation, demonstrate its usage on publicly-available sleep research data, and evaluate its suitability by reproducing published studies. In addition, we apply *TDSpy* to data from comatose patients, emphasizing its generalizability.

Code metadata

Current code version	1.0.0
Permanent link to code/repository used for this code version	Source: https://github.com/ElsevierSoftwareX/SOFTX-D-24-00292
Permanent link to Reproducible Capsule	None
Legal Code License	MIT license
Code versioning system used	Git
Software code languages, tools, and services used	Python
Compilation requirements, operating environments & dependencies	edfrd 0.7, matplotlib 3.7.0, neurokit2 0.2.3, networkx 3.9, numpy 1.24.2, pandas 1.5.3, pytest 7.2.1 scipy 1.10.1
If available Link to developer documentation/manual	https://github.com/nspi/tdspy/
Support email for questions	tabea.steinbrinker@med.uni-goettingen.de, nicolai.spicher@med.uni-goettingen.de

1. Motivation and significance

The analysis of different physiological time series acquired in parallel finds wide application in health monitoring. Signals can either (i) measure the same organ system and thereby improve accuracy in case one of the signals is distorted or (ii) measure different organ systems. An example for the first use case is heartbeat detection with different modalities to improve robustness to noise [1], while examples of the second use case are freezing of gait detection in Parkinson's disease based on brain and motion data [2–4] or characterization of the coordination between heart and brain activity during sleep [5,6]. While many methods from the field stem from traditional signal processing without any learning, machine learning methods become increasingly popular in the field, *e.g.* graph neural networks [7] or convolutional neural networks [8].

* Corresponding author at: Department of Medical Informatics, University Medical Center Göttingen, Göttingen, Germany.
E-mail address: nicolai.spicher@med.uni-goettingen.de (N. Spicher).

https://doi.org/10.1016/j.softx.2024.101890
Received 3 June 2024; Received in revised form 28 August 2024; Accepted 4 September 2024
Available online 12 September 2024

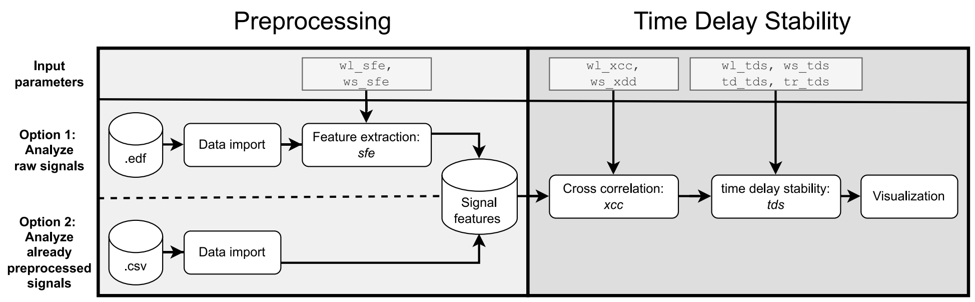

Fig. 1. Structure of *TDSpy* consisting of two parts with the first being for time series preprocessing, *i.e.* feature extraction. Extracted features are time series as well and can also be directly loaded if users want to use other implementations. The second part is the implementation of TDS as described in Section 1.1. The library requires six (without feature extraction) or eight input parameters (with feature extraction) shown in Tbl. 3.

Since its inception in 2012 [9], the framework of TDS has become a popular method for the quantification of dynamical interactions between multivariate time series, *e.g.* EEG, ECG, EMG, electrooculography (EOG) or respiration signals. These measurements contain information representing the state of different organ systems and TDS measures the degree of connectivity between the different organs, *i.e.* their network organization.

TDS has been applied successfully to data from different fields of research – *e.g.* overnight sleep studies – and revealed distinct patterns in the network organizations with respect to sleep stages [9,10]. Moreover, a dependency between network characteristics and age as well as gender was shown [11] and disease-specific changes were revealed, *e.g.* in patients suffering from obstructive sleep apnea [12], narcolepsy [13], or insomnia [14]. Furthermore, it was used for analyzing interactions of brain waves and muscle activity [15,16] and modified TDS methods were proposed [17,18].

In the past, TDS has been used by various groups from diverse contexts using their own implementations. There is an open-source implementation available (https://github.com/somnonetz/physiological-networks-tds) but it is written in MATLAB, a proprietary software associated with license fees, and also requires several toolboxes associated with additional costs. The authors are not aware of any other open-source implementation. Hence, the aim of this work is to establish a well-documented and easy-to-use implementation of TDS.

1.1. Background: Time delay stability

TDS is a quantitative measure of the coupling between organ systems. They are measured with different sensors resulting in time series and two systems are assumed to have be coupled if the changes in one time series lead to corresponding changes in the other, and vice versa. These changes might appear after a certain time delay, hence periods with a stable time delay between two time series indicate a physiological coupling between the underlying organ systems. To account for the diversity in sensor types (*e.g.* sampling rate, bit depth), TDS typically does not process raw time series (*e.g.* ECG) but derived time series containing features (*e.g.* HR).

Let x and y be two time series derived from two different organ systems where each consists of N values. We split both time series in windows of L samples with the windows overlapping by M samples, resulting in a number of $N_L = \lfloor \frac{N-L}{L-M} \rfloor + 1$ windows. Subsequently, samples in each window are normalized separately to zero mean and unit standard deviation.

The cross-correlation $C^{\nu}_{xy}(\tau)$ of two windows x^{ν} and y^{ν} with $\nu = 1, 2, \dots, N_L$ is computed via

$$C^{\nu}_{xy}(\tau) = \frac{1}{L} \sum_{i=1}^{L} x^{\nu}_i \times y^{\nu}_{i+\tau}, \quad (1)$$

where $\times$ denotes element-wise multiplication and τ denotes the lag of y^{ν} to x^{ν}. For the following analysis, the value of $C^{\nu}_{xy}(\tau)$ is not of interest but instead the position where the cross-correlation reaches its maximum:

$$\tau_{\nu} = \arg\max_{\tau} |C^{\nu}_{xy}(\tau)|. \quad (2)$$

This position is the delay where both time series are best matching and the basic idea of TDS is to measure the stability in τ_{ν} over time using a windowing approach. Hence, the series of τ_{ν} is split in windows of $\hat{L}$ samples with the windows overlapping by $\bar{M}$ samples. It is checked whether the delay values τ_{ν} in a window stay ”approximately” stable over multiple windows. The stability criterion can be manually defined by two variables, η and ζ. η indicates a threshold between two τ_{ν} values in the window so they are assumed to be stable and ζ indicates a threshold ratio how many of the τ_{ν} values need to be stable so the whole window is assumed to be stable.

The output is a binary time series $TDS_{x,y}$ of length $N_L - \hat{L}$, where 1 denotes a stable connection and 0 a unstable one. The so-called “link strength” $l_{TDS}(x, y, s)$ between two time series in a certain stage s is then computed as a fraction of stable connections

$$l_{TDS}(x, y, s) = \frac{\sum_{\nu(s)} TDS_{x,y}(\nu(s))}{N_L(s)}, \qquad \nu(s) = 1, 2, \dots, N_L(s), \quad (3)$$

with $N_L(s)$ denoting the number of samples assigned to the state s.

2. Software description

We propose *TDSpy*[1] written in Python3, a popular programming language with simple syntax which is the de-facto standard in data science and follows a free-software license. The main objective of *TDSpy* is to equip the users with a well-documented, easy-to-use and validated implementation of the previously introduced concept. In the following, we give detailed information on its implementation and testing and demonstrate its usage on a freely-available sleep study. In order to evaluate how far *TDSpy* reproduces previous findings, we apply it to data of the SIESTA sleep study used in previous publications [9,11] and compare results qualitatively and quantitatively.

2.1. Implementation

TDSpy builds upon a number of de-facto standard and free-to-use Python libraries given in Tbl. 1. Fig. 1 gives an overview of the structure of the library: It consists of two parts, use-case specific functions for loading and preprocessing data (Fig. 1: light gray area) and the main part of *TDSpy* which is a consolidated and validated implementation of TDS described in Section 1.1 (Fig. 1: dark gray area). Regarding the preprocessing part, users can decide whether to use *TDSpy* for importing raw time series and feature extraction (Option 1) or if they want to use other feature extraction implementations (Option 2).

[1] https://pypi.org/project/TDSpy/

Table 1
External dependencies of *TDSpy*.

Library	Version	Usage	License
`edfrd`	0.7	read European Data Format (EDF)-files	LGPL-3.0
`Matplotlib`	3.7.0	visualization of results as matrix	based on PSF License
`Neurokit2`	0.2.3	heart and breathing rate computation	MIT license
`networkx`	3.9	visualization of results as graph	3-clause BSD license
`NumPy`	1.24.2	data matrix representation, processing	BSD
`pandas`	1.5.3	read comma-separated values (CSV)-files	New BSD License
`pytest`	7.2.1	unit testing	MIT license
`SciPy`	1.10.1	cross correlation	New BSD License

Table 2
Overview of main functions of *TDSpy*.

Part	Function	Usage	Section
Preprocessing	`read_all_EDF_channels`	read all channels of a file in EDF	2.1.1
	`feature_extraction`	extract features from raw time series	2.1.2
TDS	`cross_correlation`	cross-correlation analysis of all signal pairs	2.1.3
	`getStability`	compute TDS of cross-correlation results	2.1.4
Visualization	`plot_graph`	plot results as a graph	2.1.5
	`plot_matrix`	plot results as a matrix	2.1.5

The main TDS computation is performed in two steps which are cross correlation (*xcc*) between two time series followed by TDS quantification, following the description in Section 1.1. Finally, the library offers different visualization methods. In the following, each part of the library is introduced with a focus on the implementation details.

The available functions are summarized in Tbl. 2 with the most relevant variables being shown in Tbl. 3. We provide default values extracted from [9] and users can adjust them to their specific needs.

2.1.1. Data import

TDSpy offers to load time series data in CSV format or EDF (https://www.edfplus.info/). The first module is based on the pandas library [19] and the second on the edfrd library [20] which both are freely-available via PyPI. They can be used to (i) load raw time series data which is then processed in the next step for feature extraction or (ii) load features already extracted with external algorithm. Next to the time series data for TDS processing, users can provide an optional CSV file containing labels for signal segments. These labels can be used to split the analysis with respect to time, *e.g.* for different stages of an experiment.

2.1.2. sfe : signal feature extraction and resampling

In this step, feature time series are derived from raw time series loaded in the previous step. This step is optional and users can decide to load their own already extracted and re-sampled features. In the selection of the feature extraction methods, we decided to use basic methods. Our rationale was to provide robust and generic methods with a bottom baseline performance. Users are invited to implement more sophisticated or specific methods.

To make time series comparable for later analysis, feature time series are resampled to a uniform rate, *i.e.* have the same number of data points per time interval. Following [9], we use a sliding window approach with a duration (`wl_sfe`) and a shift parameter (`ws_sfe`) which are applied to all extracted time series.

TDSpy offers signal extraction methods for EEG, ECG, EMG, EOG, and respiratory time series, following the suggestions in [9]: As different brain waves indicate different physiological states, we separate the EEG signal into different frequency bands. Per default the EEG data is split into 5 different frequency bands, δ waves $f_\delta = [0.5, 3.5)$ Hz, θ waves $f_\theta = [4.0, 7.5)$ Hz, α waves $f_\alpha = [8.0, 11.5)$ Hz, σ waves $f_\sigma = [12.0, 15.5)$ Hz and β waves $f_\beta = [16.0, 19.5)$ Hz. Regarding ECG time series, channels are processed individually with the continuous HR being extracted using the `nk.ecg_process` function of the Neurokit2 library. The muscular (EMG) and ocular activity (EOG) is extracted by computing the variance of the respective time series. The continuous breathing rate is derived from a respiratory signal using `nk.rsp_rate` of the Neurokit2 library.

2.1.3. xcc : cross-correlation analysis

This step realizes cross-correlation analysis of all pairs of time series (Section 1.1: $C^v_{xy}(\tau)$). At first, all possible pairs of time series are identified. Subsequently, *xcc* of each combination is computed using a sliding window. Both time series are split into windows of `wl_xcc` duration (Section 1.1: L) with a shift of `ws_xcc` (Section 1.1: $L - M$). Within each window, both time series are normalized using the standardized z-score. For computing the correlation between both normalized time series, functions `scipy.signal.correlate` and `scipy.signal.correlation_lags` provided by the scipy library are used. Finally, the index of the maximum amplitude of the resulting correlation signal (Section 1.1: $\arg\max_\tau |C^v_{xy}(\tau)|$), representing the delay between two time series is detected for all combinations of time series.

2.1.4. tds : time delay stability analysis

This part of the library accepts the delay values computed by *xcc* analysis and computes their stability in a sliding window approach depicted in Fig. 2. The delay values are processed using a window of length `wl_tds` (Section 1.1: $\bar{L}$) and shift `ws_tds` (Section 1.1: $\bar{L} - \bar{M}$), beginning on the first delay value. For each window, the stability delays are evaluated using two variables: `td_tds` (Section 2: η) defines the threshold between delays, *i.e.* the vertical spread of the window, and `tr_tds` (Section 2: ζ) defines the ratio threshold of how many delays are inside `td_tds` (dark blue points) vs. outside (green points). If the ratio is larger than a defined threshold (`tr_tds`), this window is assumed to be stable and marked with a value of 1. Otherwise, it is marked with a value of 0.

An optional vector (Section 1.1: s) can be submitted containing information on different stages during acquisition of the experiment. If no vector is submitted, the link strength is computed for the whole signal. If a vector is submitted, a link strength value is computed for each stage.

2.1.5. Visualization

TDSpy provides two different ways to visualize results: A matrix visualization and a graph visualization depicted in Figs. 3 and 4, respectively. Using `matplotlib`, the link strengths between time series are plotted as a matrix with an entry in the matrix representing the coupling between the time series marked in the respective row and column in percent. The matrix is symmetric and has values equal to 100% on the diagonal. The colormap and its range can be freely chosen. Based on the library `networkx`, *TDSpy* offers a visualization of the network, drawing an edge between the nodes if the link strength is greater than a user-defined threshold. In [9] a threshold of 7% is proposed based on surrogate data analysis which acts as standard value. As other works suggested other thresholds [10], this value can be freely adjusted.

Table 3
Key variables of *TDSpy* defining either window lengths (`wl_*`), window shifts (`ws_*`) or thresholds.

Part	Sym.	Variable	Usage
Pre-processing		`wl_sfe`	window length of signal feature extraction (*sfe*) in seconds (default: 2 s)
		`ws_sfe`	window shift of *sfe* in seconds (default: 1 s)
Time Delay Stability	L	`wl_xcc`	window length of *xcc* in seconds (default: 60 s)
	M	`ws_xcc`	window shift of *xcc* in seconds (default: 30 s)
	$\bar{L}$	`wl_tds`	number of delays analyzed in a TDS window (default: 5)
	$\bar{M}$	`ws_tds`	number of delays shifted between two TDS windows (default: 1)
	η	`td_tds`	threshold for differences between delays (default: 2)
	ζ	`tr_tds`	threshold for ratio between number of delays in-/outside of `td_tds` (default: 80%)

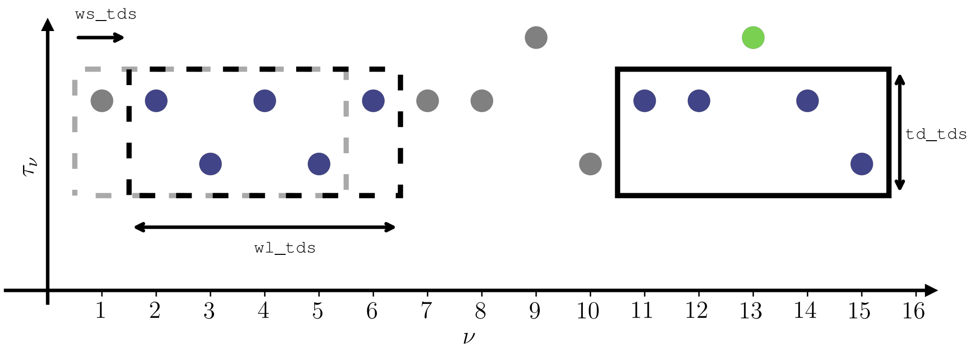

Fig. 2. Computation of TDS: A window with fixed length (`wl_tds`) and height (`td_tds`) is shifted over the delay values τ_ν. For the first and the second window (dashed gray and black lines, respectively) all delay values are within `td_tds` (dark blue points), and therefore these windows are assumed to be stable. For the last window (black line), one delay value (green point) is outside `td_tds`, resulting in a ratio of stability 4/5 = 80%. If this value is greater or equal than `tr_tds`, the window is also assumed to be stable.

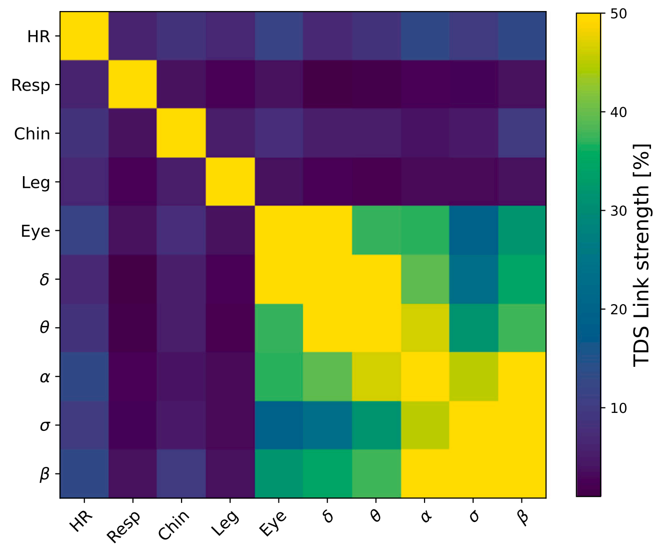

Fig. 3. Matrix visualization of a healthy subject (22 years, female) in light sleep from the SIESTA sleep database. Colors indicate the percentage of windows quantified as stable out of all windows during light sleep. Colormap: *viridis*, Range: [0,50].

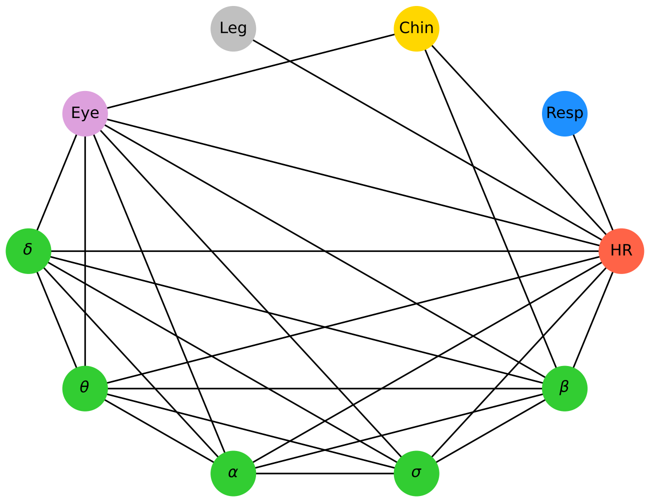

Fig. 4. Graph visualization of the same data: HR is extracted from ECG, Resp from the airflow signal, chin and leg motion from EMG, and the eye activity from EOG. Furthermore, five EEG frequency bands are shown. The threshold is set to 7% as defined in [9].

2.1.6. *Testing*

We provide several tests for verification of all essential functions of the source code using the `pytest` and `numpy.testing` libraries. Real data was used to check if file reading and processing works as expected. Moreover, synthetic data was used to test if the output of a function is of the correct type (`isinstance`) and value (`numpy.testing.assert_equal`). For example, to test the cross correlation analysis (Section 2.1.3) a synthetic signal and a displaced copy were generated and the result compared to the expected result. Whenever possible, signals with random noise were used to increase robustness and tests were repeated multiple times (`@pytest.mark.repeat()`).

3. Illustrative examples

3.0.1. *Validation with public data*

As a benchmark for the usefulness of the proposed library, we demonstrate its ”out-of-the-box” usage on publicly-available polysom - nography file provided by the St. Vincent's University Hospital/University College Dublin Sleep Apnea Database (ucddb) [21] available as an EDF file and a CSV file which contains the sleep stages following Rechtschaffen and Kales rules. The database contains polysomnography recordings from adult subjects with suspected sleep-disordered breathing and we select the first subject (id: `ucddb002`, male, BMI: 33.9, AHI: 23, duration: 6.2 h). For the analysis with *TDSpy*, we follow option 1 depicted in Fig. 1, *i.e.* we compute the features from the raw time series. The sample code in Listing 1 demonstrates the usage of *TDSpy* for computing results for the wake stage.

Listing 1: TDSpy applied to data from ucddb sleep apnea database

```
import neurokit2 as nk
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# TDSpy functions
from TDSpy.sn_TDS import sn_TDS_no_feature_extraction as TDS
from TDSpy.feature_extraction.sn_getEEGBandPower import
    ↪ sn_getEEGBandPower
from TDSpy.feature_extraction.sn_getVariance import sn_getVariance
from TDSpy.tools.sn_plotTDS import plot_TDS
from TDSpy.tools.edf_reader import read_all_EDF_channels

# Preparation: Please download these files:
# https://physionet.org/content/ucddb/1.0.0/ucddb002.rec
# https://physionet.org/content/ucddb/1.0.0/ucddb002_stage.txt

def main():
    sleep_stage = 0 # wake
    dur = 16000 # seconds

    # Read EDF
    data_dict, data_dict_sampling_rate = read_all_EDF_channels("ucddb002.
        ↪ rec", startrecord=0, endrecord=dur)

    # Read sleep stages
    stages = pd.read_csv("ucddb002_stage.txt", header=None)
    stages = stages.to_numpy()

    # Read only single stage
    stage_idx = np.where(stages == sleep_stage)[0]
    stage_idx = stage_idx[stage_idx < dur/30]

    # Process single ECG channel
    ecg_signal = nk.ecg_process(data_dict['ECG'], sampling_rate=
        ↪ data_dict_sampling_rate['ECG'], method='neurokit')
    hr_signal_resampled = nk.signal_resample(ecg_signal[0]['ECG_Rate'],
        ↪ sampling_rate=data_dict_sampling_rate['ECG'],
        ↪ desired_sampling_rate=1, method="interpolation")

    # Process single RESP channel
    rsp_rate = nk.rsp_rate(data_dict['Flow'], sampling_rate=8, method="
        ↪ trough")
    rsp_signal_resampled = nk.signal_resample(rsp_rate, sampling_rate=
        ↪ data_dict_sampling_rate['Flow'], desired_sampling_rate=1,
        ↪ method="interpolation")

    # Process single EMG channel
    emg_signal = data_dict['EMG']
    emg_var = sn_getVariance(emg_signal, sf=data_dict_sampling_rate['
        ↪ EMG'])

    # Process single EOG channel
    eog_signal = data_dict['Lefteye']
    eog_var = sn_getVariance(eog_signal, sf=data_dict_sampling_rate['
        ↪ Lefteye'])

    # Process single EEG channel
    eeg_signal = data_dict['C3A2']
    fpb, _ = sn_getEEGBandPower(eeg_signal, sf=data_dict_sampling_rate['
        ↪ C3A2'], bandlimits=np.array([[0.5, 4, 8, 12, 16], [3.5, 7.5,
        ↪ 11.5, 15.5, 19.5]]))

    # Compute TDS
    data_dict = {'HR': hr_signal_resampled, 'Resp': rsp_signal_resampled, '
        ↪ Chin': emg_var, 'Eye': eog_var, 'Delta': fpb[:,0], 'Theta': fpb
        ↪ [:,1], 'Alpha': fpb[:,2], 'Sigma': fpb[:,3], 'Beta': fpb[:,4]}
    tds, combination, stages = TDS(data_dict=data_dict)

    # Plot result
    nk.signal_plot(pd.DataFrame(data_dict), subplots=True)
    plt.show()

    # Limit to sleep stage
    tds = tds[:,stage_idx[:-1]]

    # Matrix plot
    plot_TDS(tds, combination)

if _name_ == '_main_':
    main()
```

Table 4
Quantitative analysis of differences to [9] and [11]. We analyzed the difference in TDS values (percentages) for all pairs of physiological systems.

	Mean of abs. error	Std. of abs. error
Subset 1: Bashan et al. [9]	1.59	1.63
Subset 2: Krefting et al. [11]	0.07	0.13

Fig. 5 shows the resulting network visualizations for the stages wake, light sleep, and deep sleep. Our results are very similar to the findings reported in [9]. One can clearly see a transition during sleep with the highest number of links during wakefulness, lower numbers of links in light sleep and the lowest number in deep sleep. The dominance of the brain–brain links during deep sleep with links to the eyes and chin and furthermore, the missing links to respiration in light sleep [9] can be observed.

3.0.2. Validation via reproduction

In addition, we use *TDSpy* to replicate earlier works that analyzed the SIESTA sleep database [22,23]. It contains 600 polysomnography recordings of 197 healthy subjects and 97 subjects showing high-prevalence sleep disorders, *e.g.* sleep apnea, insomnia, and mood disorders. The data was recorded from 1997 to 2000 in eight European sleep centers with all subjects giving informed consent.

We compare our results to the results reported in the landmark paper [9] introducing TDS and the other is a work focusing on the analysis of age and gender effects during sleep [11]. Hence, we analyze two subsets of the SIESTA sleep database. Subset 1: The first paper [9] used a subset of the SIESTA database by selecting 36 healthy young subjects (18 female, age range 20–40). They applied TDS to the EEG, ECG, EOG, EMG, as well as the respiration data. Subset 2: The second paper [11] considered the 197 healthy controls (103 females, age range 20–95), of the SIESTA database. After excluding twelve noisy PSG recordings, 385 recordings remained for analysis. For processing the SIESTA data, we focus the analysis on the computation of TDS and not the preprocessing of the raw time series. Hence, we use the preprocessed time series from [9,11] and follow option 2 depicted in Fig. 1. For the analysis with *TDSpy*, the default parameters of the algorithm are used as shown in Tbl. 3.

First, we analyze results qualitatively. Fig. 6 shows the comparison between the results reported in [9] and computed using *TDSpy*. The values for the left matrix were provided by the authors of [9]. As can be seen, both matrices follow a similar overall trend with low link strengths for Resp and leg EMG and high link strengths between the different EEG bands. Rather high differences can be observed for the θ–α link with a mismatch around 6% and the leg–chin link around 4%. Fig. 7 shows the comparison to the results described in [11] (Subset 2). The two panels are visually quasi-identical.

Tbl. 4 gives an overview of the differences for both datasets as a quantitative analysis. The agreement is nearly perfect for subset 2, since the mean absolute deviations of the TDS percentages are merely 0.07% points with a standard deviation of the absolute error only slightly larger. These mean absolute deviations are two orders of magnitude smaller than the standard threshold for a significant link set at 7% in [9] (see Section 2.1.5). For subset 1, the deviations are much smaller than this threshold, although the application of *TDSpy* sometimes yields smaller values than reported [9].

3.0.3. Validation via dataset from a different domain

In order to analyze if *TDSpy* is useful beyond sleep datasets, we apply it to data from another medical domain. We apply *TDSpy* to data from a patient of the iCARE database [24,25] which provides continuous EEG data from coma due to hypoxic-ischemic brain injury involving patients with cardiac arrest. We select the first 8 h of data from the patient being reported as having a Cerebral Performance Category (CPC) score of 1, *i.e.* having good neurological function and

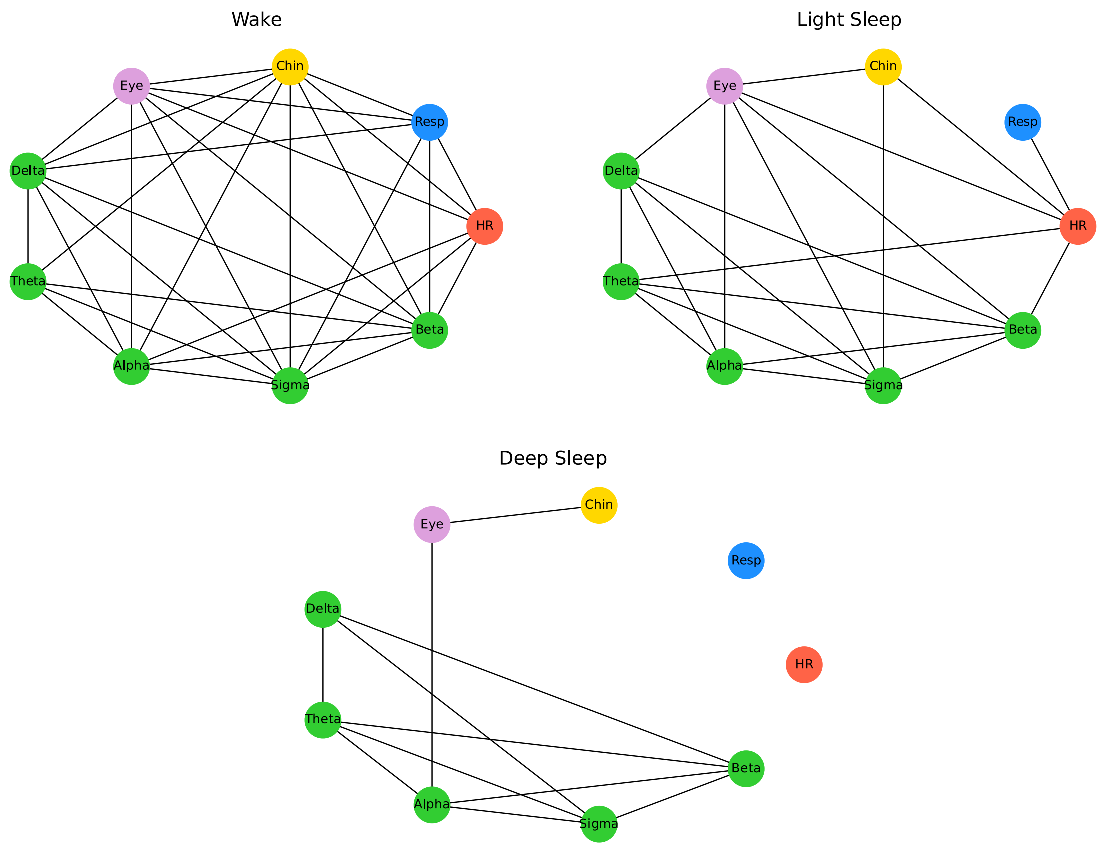

Fig. 5. Graph visualization for a single subject with the threshold for link strength being defined as 7% according to [9]. Graphs are computed for the stages wake (left), light sleep (right), and deep sleep (below). The results can be reproduced using the source code provided in Listing 1.

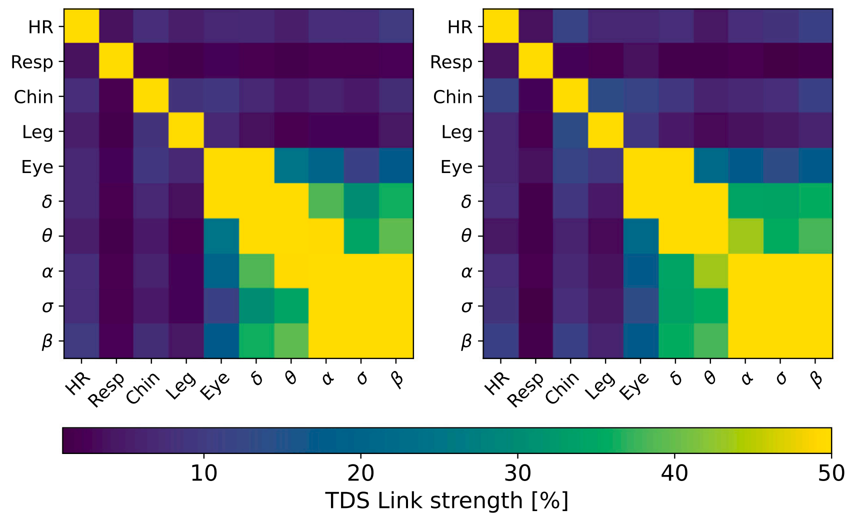

Fig. 6. Results for Subset 1 reported in [9] (left) and processed using *TDSpy* (right). Colormap: *viridis*, Range: [0,50].

independent for activities of daily living. Interestingly, results show very similar behavior to TDS results reported in [10] for healthy subjects. Strong coupling and symmetry can be observed in the left and right hemispheres.

4. Impact

In this work, we proposed *TDSpy*, an open-source implementation of TDS that we validated by comparing results to the original paper [9] and a work that focused on age and gender dependency [11]. The mismatch to the second work is negligible while the mismatch to the original publication is higher, reaching the highest mean absolute error of 4.1 % during wakefulness of the subjects. It thus seems that the differences are mainly caused by non-stationarities and/or different correction of measurement errors, both of which are more frequent during wakefulness than during non-REM sleep. In addition, they might result from undocumented mitigation strategies in the algorithms used in [9].

Unfortunately, the original source code is not available anymore, limiting our possibilities to analyze the source of this mismatch. The authors reported that the original source code was written in MATLAB

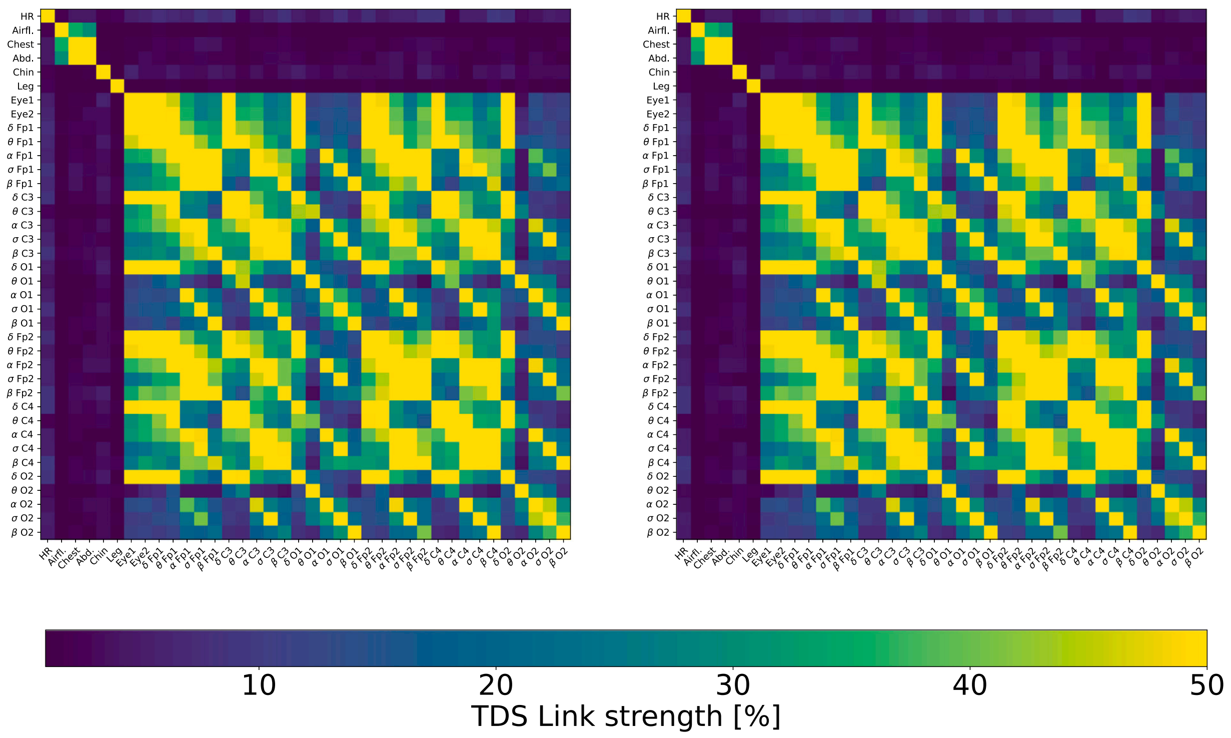

Fig. 7. Results for Subset 2 reported in [11] (left) and processed using *TDSpy* (right). Colormap: *viridis*, Range: [0,50].

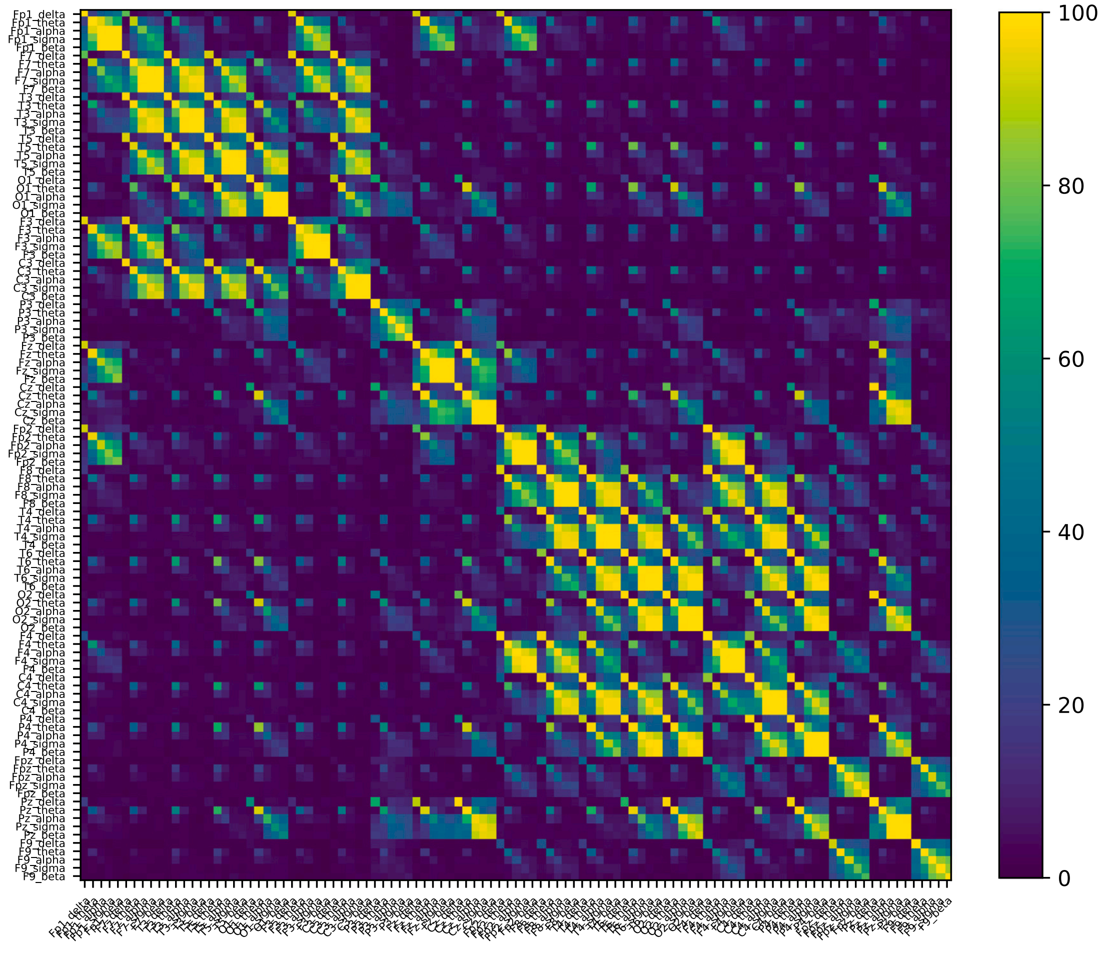

Fig. 8. Matrix visualization of the first eight hours of coma of subject 0306 from hospital E (73 years, male) as provided by the iCARE database.

using its cross-correlation implementation (`xcorr()`). *TDSpy* makes use of the functions `correlate()` and `correlation_lags()` provided by `SciPy` which might also explain smaller differences. However, the subset used in the second paper [11] is much larger and also includes the subjects of [9]; hence, we assume that our implementation is suited for practical usage.

We want to highlight that *TDSpy* was designed to be lightweight and seamlessly to integrate into existing analysis pipelines. By providing methods for reading data from CSV and especially EDF files, two major file types in biosignal analysis are supported. Further file types, *e.g.* MATLAB files could be read effortlessly by the already loaded library SciPy. Regarding output, we offer two popular visualization methods with the graph and matrix visualization. Due to the modular design of *TDSpy*, other visualization methods could be added without much effort.

The feature extraction step of *TDSpy* is evidently the one most prone to errors due to the risk of noisy signals leading to biased features extracted. As multimodal signals are often acquired in challenging environments, *e.g.* during surgery [26], there is the risk of a low signal-to-noise ratio. Low-/high-/bandpass-filters are outside of the scope of this library but we add references within the manual link to useful libraries such as SciPy, Neurokit2, or MNE.

TDSpy offers for the first time a validated and open-source implementation of the concept of Time Delay Stability analysis. It enables answering research questions in the emerging field of multimodal time series analysis in the healthcare domain. Examples include the analysis of data acquired from wearable devices such as smart watches [27] and monitoring of patients during surgery [28] or coma [29]. A recent work [26] has demonstrated that time series measured during surgery are feasible for multivariate analysis. The application of TDS to this dataset will be addressed in future work.

The presented results of applying TDS to the EEG data of comatose patients (Fig. 8) shows strong similarity to the work of Bartsch et al. [10]. However, while this work focused on sleep data and measured 6 scalp locations only, 22 EEG channels from multiple hundreds of subjects in coma are available in the iCARE dataset. While conventional EEG features such as coherence and power bands have already been analyzed for CPC prediction [25,29,30], analyzing the dataset w.r.t. TDS coupling could be a promising avenue for future work.

The authors hope that by offering this library, Time Delay Stability analysis might also be available to a wider scientific audience and other fields of research.

5. Conclusions

Reproducibility is a cornerstone of good scientific practice but especially in the field of computational algorithms, it is often not addressed adequately [31]. As the field of software development is so fast-paced, working environments are changing rapidly, rendering source code or binary tools outdated. Due to lack of standardization, in many cases relevant parameters are not documented or important detail on the algorithms are missing. This could lead to different implementations depending on the interpretation of the developer and potentially biased results. Hence, with *TDSpy* we aim to contribute to a scientific culture driven by open and reproducible algorithms.

CRediT authorship contribution statement

Tabea F.A. Steinbrinker: Writing – review & editing, Writing – original draft, Visualization, Software, Conceptualization. **Dagmar Krefting:** Writing – review & editing, Validation, Resources, Funding acquisition, Conceptualization. **Ronny P. Bartsch:** Writing – review & editing, Validation, Conceptualization. **Jan W. Kantelhardt:** Writing – review & editing, Validation, Conceptualization. **Nicolai Spicher:** Writing – review & editing, Writing – original draft, Supervision, Project administration, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors do not have permission to share data.

Acknowledgments

The authors thank Sebastian Schmale of University Medical Center Göttingen for his help with publishing the package on PyPi. We also thank Asaf Markuza for helpful discussions in the early stages of this project.

This work was supported by the Lower Saxony "Vorab" of the Volkswagen Foundation and the Ministry for Science and Culture of Lower Saxony, Germany under Grant 76211-12-1/21.

References

[1] De Cooman T, Goovaerts G, Varon C, Widjaja D, Willemen T, Van Huffel S. Heart beat detection in multimodal data using automatic relevant signal detection. Physiol Meas 2015;36(8):1691–704. http://dx.doi.org/10.1088/0967-3334/36/8/1691.

[2] Wang Y, Beuving F, Nonnekes J, Cohen MX, Long X, Aarts RM, et al. Freezing of gait detection in Parkinson's disease via multimodal analysis of EEG and accelerometer signals. In: Annual international conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual international conference, vol. 2020. 2020, p. 847–50. http://dx.doi.org/10.1109/EMBC44109.2020.9175288.

[3] Günther M, Bartsch RP, Miron-Shahar Y, Hassin-Baer S, Inzelberg R, Kurths J, et al. Coupling between leg muscle activation and EEG during normal walking, intentional stops, and freezing of gait in parkinson's disease. Front Physiol 2019;10:870.

[4] Asher EE, Plotnik M, Günther M, Moshel S, Levy O, Havlin S, et al. Connectivity of EEG synchronization networks increases for Parkinson's disease patients with freezing of gait. Commun Biol 2021;4(1):1017.

[5] Mikutta C, Wenke M, Spiegelhalder K, Hertenstein E, Maier JG, Schneider CL, et al. Co-ordination of brain and heart oscillations during non-rapid eye movement sleep. J Sleep Res 2022;31(2):e13466. http://dx.doi.org/10.1111/jsr.13466.

[6] Lin A, Liu KK, Bartsch RP, Ivanov PC. Delay-correlation landscape reveals characteristic time delays of brain rhythms and heart interactions. Phil Trans R Soc A 2016;374(2067):20150182.

[7] Tang S, Dunnmon JA, Liangqiong Q, Saab KK, Baykaner T, Lee-Messer C, et al. Modeling multivariate biosignals with graph neural networks and structured state space models. In: Mortazavi BJ, Sarker T, Beam A, Ho JC, editors. Proceedings of the conference on health, inference, and learning. Proceedings of machine learning research, vol. 209, PMLR; 2023, p. 50–71.

[8] Ma YJ, Zschocke J, Glos M, Kluge M, Penzel T, Kantelhardt JW, et al. Automatic sleep-stage classification of heart rate and actigraphy data using deep and transfer learning approaches. Comput Biol Med 2023;163:107193.

[9] Bashan A, Bartsch RP, Kantelhardt JW, Havlin S, Ivanov PC. Network physiology reveals relations between network topology and physiological function. Nature Commun 2012;3(1):702. http://dx.doi.org/10.1038/ncomms1705, URL https://www.nature.com/articles/ncomms1705.

[10] Bartsch RP, Liu KKL, Bashan A, Ivanov PC. Network physiology: How organ systems dynamically interact. PLoS One 2015;10(11):1–36. http://dx.doi.org/10.1371/journal.pone.0142143.

[11] Krefting D, Jansen C, Penzel T, Han F, Kantelhardt JW. Age and gender dependency of physiological networks in sleep. Physiol Meas 2017;38(5):959–75. http://dx.doi.org/10.1088/1361-6579/aa614e, URL https://iopscience.iop.org/article/10.1088/1361-6579/aa614e.

[12] Jansen C, Hodel S, Penzel T, Spott M, Krefting D. Feature relevance in physiological networks for classification of obstructive sleep apnea. Physiol Meas 2018;39(12):124003. http://dx.doi.org/10.1088/1361-6579/aaf0c9, URL https://iopscience.iop.org/article/10.1088/1361-6579/aaf0c9.

[13] Son DY, Kwon HB, Lee DS, Jin HW, Jeong JH, Kim J, et al. Changes in physiological network connectivity of body system in narcolepsy during REM sleep. Comput Biol Med 2021;136:104762. http://dx.doi.org/10.1016/j.compbiomed.2021.104762, URL https://linkinghub.elsevier.com/retrieve/pii/S0010482521005564.

[14] Jansen C, Penzel T, Hodel S, Breuer S, Spott M, Krefting D. Network physiology in insomnia patients: Assessment of relevant changes in network topology with interpretable machine learning models. Chaos 2019;29(12):123129. http://dx.doi.org/10.1063/1.5128003, URL https://pubs.aip.org/cha/article/29/12/123129/1029211/Network-physiology-in-insomnia-patients-Assessment.

[15] Rizzo R, Garcia-Retortillo S, Ivanov PC. Dynamic networks of physiologic interactions of brain waves and rhythms in muscle activity. Hum Mov Sci 2022;84:102971. http://dx.doi.org/10.1016/j.humov.2022.102971, URL https://linkinghub.elsevier.com/retrieve/pii/S0167945722000513.

[16] Rizzo R, Zhang X, Wang JWJL, Lombardi F, Ivanov PC. Network physiology of cortico–muscular interactions. Front Physiol 2020;11:558070. http://dx.doi.org/10.3389/fphys.2020.558070, URL https://www.frontiersin.org/articles/10.3389/fphys.2020.558070/full.

[17] Tolston MT, Funke GJ, Shockley K. Comparison of cross-correlation and joint-recurrence quantification analysis based methods for estimating coupling strength in non-linear systems. Front Appl Math Stat 2020;6:1. http://dx.doi.org/10.3389/fams.2020.00001, URL https://www.frontiersin.org/article/10.3389/fams.2020.00001/full.

[18] Liu KK, Bartsch RP, Ma QD, Ivanov PC. Major component analysis of dynamic networks of physiologic organ interactions. In: Journal of physics: Conference series, vol. 640, no. 1, IOP Publishing; 2015, 012013.

[19] The pandas development team. Pandas-dev/pandas: Pandas. 2020, http://dx.doi.org/10.5281/zenodo.3509134.

[20] Jansen C. edfrd · PyPI. 2020, URL https://pypi.org/project/edfrd/.

[21] Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PC, Mark RG, et al. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. Circulation 2000;101(23). http://dx.doi.org/10.1161/01.CIR.101.23.e215, URL https://www.ahajournals.org/doi/10.1161/01.CIR.101.23.e215.

[22] Klosh G, Kemp B, Penzel T, Schlogl A, Rappelsberger P, Trenker E, et al. The SIESTA project polygraphic and clinical database. IEEE Eng Med Biol Mag 2001;20(3):51–7. http://dx.doi.org/10.1109/51.932725, URL http://ieeexplore.ieee.org/document/932725/.

[23] Penzel T, Kemp B, Klosch G, Schlogl A, Hasan J, Varri A, et al. Acquisition of biomedical signals databases. IEEE Eng Med Biol Mag 2001;20(3):25–32. http://dx.doi.org/10.1109/51.932721, URL http://ieeexplore.ieee.org/document/932721/.

[24] Amorim E, Zheng W, Lee JW, Herman S, Ghassemi M, Sivaraju A, et al. I-CARE: International Cardiac Arrest REsearch consortium Database (version 2.1). 2023, http://dx.doi.org/10.13026/m33r-bj81.

[25] Amorim E, Zheng W-L, Ghassemi MM, Aghaeeaval M, Kandhare P, Karukonda V, et al. The international cardiac arrest research consortium electroencephalography database. Crit Care Med 2023;51(12):1802–11. http://dx.doi.org/10.1097/CCM.0000000000006074, URL https://journals.lww.com/10.1097/CCM.0000000000006074.

[26] Idrobo-Ávila E, Bognár G, Krefting D, Penzel T, Kovács P, Spicher N. Quantifying the suitability of biosignals acquired during surgery for multimodal analysis. IEEE Open J Eng Med Biol 2024;5:250–60. http://dx.doi.org/10.1109/OJEMB.2024.3379733.

[27] Li D, Vaidya J, Wang M, Bush B, Lu C, Kollef M, et al. Feasibility study of monitoring deterioration of outpatients using multimodal data collected by wearables. ACM Trans Comput Healthc 2020;1(1). http://dx.doi.org/10.1145/3344256.

[28] Lee H-C, Park Y, Yoon SB, Yang SM, Park D, Jung C-W. VitalDB, a high-fidelity multi-parameter vital signs database in surgical patients. Sci Data 2022;9(1):279. http://dx.doi.org/10.1038/s41597-022-01411-5, URL https://www.nature.com/articles/s41597-022-01411-5.

[29] Reyna MA, Amorim E, Sameni R, Weigle J, Elola A, Rad AB, et al. Predicting neurological recovery from coma after cardiac arrest: The George B. Moody PhysioNet challenge 2023. In: 2023 computing in cardiology (CinC), vol. 50. 2023, p. 1–4.

[30] Hempel P, Zaschke P, Goldammer M, Spicher N. Fusion of features with neural networks for prediction of secondary neurological outcome after cardiac arrest. In: 2023 computing in cardiology (CinC), vol. 50. 2023, p. 1–4. http://dx.doi.org/10.22489/CinC.2023.210.

[31] Shenouda J, Bajwa WU. A guide to computational reproducibility in signal processing and machine learning [tips & tricks]. IEEE Signal Process Mag 2023;40(2):141–51. http://dx.doi.org/10.1109/MSP.2022.3217659.