

Package: PhysioNetPhysiology (via r-universe)

October 3, 2026

Type Package

Title Network Physiology and Time-Delay-Stability Organ-Interaction Networks

Version 0.3.1

Description Tools for Network Physiology: the study of dynamic interactions among diverse physiological systems (brain, heart, respiration, muscle). Implements Time Delay Stability (TDS) analysis to identify stable dynamic couplings between signals, builds organ-interaction networks whose links are significance-tested against phase-randomised surrogates, and quantifies how those networks reconfigure across physiological states (e.g. sleep stages). Designed to operate on the multimodal PhysioExperiment data model, but the core engine works on plain multivariate time series. Methods follow Bashan et al. (2012) <[doi:10.1038/ncomms1705](https://doi.org/10.1038/ncomms1705)> and Bartsch et al. (2015) <[doi:10.1371/journal.pone.0142143](https://doi.org/10.1371/journal.pone.0142143)>.

License MIT + file LICENSE

URL <https://github.com/x-biosignal/PhysioNetPhysiology>,
<https://x-biosignal.r-universe.dev/PhysioNetPhysiology>,
<https://x-biosignal.github.io/PhysioNetPhysiology>

BugReports <https://github.com/x-biosignal/PhysioNetPhysiology/issues>

Encoding UTF-8

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.2)

Imports stats, graphics, grDevices, utils, methods, Rcpp

LinkingTo Rcpp

Suggests PhysioCore, PhysioCrossModal, SummarizedExperiment, S4Vectors, igraph, ggplot2, knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Repository <https://x-biosignal.r-universe.dev>

Date/Publication 2026-10-03 13:11:49 UTC

RemoteUrl <https://github.com/x-biosignal/PhysioNetPhysiology>

RemoteRef HEAD

RemoteSha 2897d0b24bd638725d2670b24ea3eb038b58c760

Contents

aggregateNetwork	3
cardiacPhaseFromPeaks	3
cardiorespiratoryCoupling	4
communityDetection	5
heartbeatEvokedPotential	6
hepAmplitude	7
hepSignificance	7
instantaneousPhase	8
layerSimilarity	9
moduleRoles	9
multiplexDegree	10
multiplexNetwork	11
multiplexParticipation	11
networkMetrics	12
nmPhaseSynchronization	13
participationCoefficient	13
phaseSynchronizationIndex	14
physioNodeMatrix	15
plotDelayStability	16
plotHEP	17
plotSynchrogram	17
plotTDSnetwork	18
plotTDSreconfiguration	18
smallWorldness	19
supraAdjacency	20
synchrogram	20
tdsNetwork	21
tdsNetworkByState	22
tdsReconfiguration	23
tdsSurrogateThreshold	24
timeDelayStability	25
withinModuleDegreeZ	26

Index	28
--------------	-----------

aggregateNetwork	<i>Aggregate a multiplex into a single weighted network</i>
------------------	---

Description

Aggregate a multiplex into a single weighted network

Usage

```
aggregateNetwork(x, method = c("sum", "mean"))
```

Arguments

x	A "multiplex" object.
method	"sum" (default) or "mean" over layers.

Value

A single weighted adjacency matrix.

Examples

```
L1 <- matrix(c(0,1,0, 1,0,1, 0,1,0), 3)
L2 <- matrix(c(0,0,1, 0,0,0, 1,0,0), 3)
aggregateNetwork(multiplexNetwork(list(alpha = L1, beta = L2)))
```

cardiacPhaseFromPeaks	<i>Cardiac phase from R-peak marks</i>
-----------------------	--

Description

Builds a continuous cardiac phase that advances linearly from 0 to 2π between consecutive R-peaks (wrapped per beat). Samples outside the first/last beat are NA.

Usage

```
cardiacPhaseFromPeaks(rpeaks, n)
```

Arguments

rpeaks	Integer sample indices of R-peaks (1-based).
n	Length of the output phase vector (number of samples).

Value

Numeric vector of wrapped cardiac phase in $[0, 2\pi)$, length n.

Examples

```
rpeaks <- seq(10, 1000, by = 25)
ph <- cardiacPhaseFromPeaks(rpeaks, 1000)
```

cardiorespiratoryCoupling

Analyse cardiorespiratory coupling

Description

High-level convenience wrapper: extracts respiratory phase (band-passed Hilbert) and cardiac phase (from R-peaks), scans n:m phase locking, and returns the dominant heartbeats-per-breath ratio, its locking index, and the synchrogram.

Usage

```
cardiorespiratoryCoupling(
  resp,
  rpeaks,
  sampling_rate,
  resp_band = c(0.1, 0.5),
  max_ratio = 8L
)
```

Arguments

<code>resp</code>	Respiration signal.
<code>rpeaks</code>	Integer R-peak sample indices (same time base as <code>resp</code>).
<code>sampling_rate</code>	Sampling rate in Hz.
<code>resp_band</code>	Respiratory passband in Hz (default <code>c(0.1, 0.5)</code>).
<code>max_ratio</code>	Largest heartbeats-per-breath ratio to consider (default 8).

Value

An object of class "cardioresp": a list with `ratio` (a "heart:resp" string), `n`, `m`, `lambda` (locking index), `sync_index` (1:1 phase-locking value), `synchrogram`, and the table of scanned ratios.

Examples

```
t <- seq(0, 120, by = 0.01); fs <- 100
resp <- sin(2 * pi * 0.25 * t)
card <- sin(2 * pi * 1.0 * t) # 4 beats per breath
rpeaks <- which(diff(sign(card)) > 0)
cr <- cardiorespiratoryCoupling(resp, rpeaks, fs)
cr$ratio
```

communityDetection	<i>Detect communities in a physiological network</i>
--------------------	--

Description

Partitions a network into communities (modules) and reports the modularity Q . The membership can be passed to [moduleRoles\(\)](#) / [participationCoefficient\(\)](#), closing the loop those role metrics need. Delegates to igraph.

Usage

```
communityDetection(  
  x,  
  method = c("louvain", "fast_greedy", "walktrap", "leading_eigen", "infomap"),  
  seed = NULL  
)
```

Arguments

x	A tds_network (from tdsNetwork()) or a weighted adjacency matrix.
method	Community-detection algorithm: "louvain" (default), "fast_greedy", "walktrap", "leading_eigen", or "infomap".
seed	Optional integer seed (some algorithms are stochastic).

Value

A community_partition object: membership (named integer vector), modularity (Q), n_communities, sizes, and method.

See Also

[networkMetrics\(\)](#), [moduleRoles\(\)](#)

Examples

```
set.seed(1)  
A <- matrix(0, 8, 8)  
A[1:4, 1:4] <- 0.9; A[5:8, 5:8] <- 0.9; A[1, 5] <- A[5, 1] <- 0.2  
diag(A) <- 0  
communityDetection(A)
```

 heartbeatEvokedPotential

Heartbeat-evoked potential (HEP)

Description

Epochs a brain signal around each R-peak, baseline-corrects, and averages to the heartbeat-evoked potential.

Usage

```
heartbeatEvokedPotential(
  eeg,
  rpeaks,
  sampling_rate,
  window = c(-0.1, 0.6),
  baseline = c(-0.1, 0)
)
```

Arguments

eeg	Numeric brain signal (one channel).
rpeaks	Integer R-peak sample indices (same time base as eeg).
sampling_rate	Sampling rate in Hz.
window	Epoch window <code>c(pre, post)</code> in seconds relative to the R-peak (default <code>c(-0.1, 0.6)</code>).
baseline	Baseline window <code>c(from, to)</code> in seconds for per-epoch correction; NULL to skip (default <code>c(-0.1, 0)</code>).

Value

An object of class "hep": a list with `time` (s, relative to the R-peak), `hep` (averaged waveform), `epochs` (epochs x time matrix), `n` (number of epochs), and `sampling_rate`.

Examples

```
set.seed(1); fs <- 250; n <- 30 * fs
rpeaks <- seq(fs, n - fs, by = round(0.9 * fs))
eeg <- rnorm(n)
hep <- heartbeatEvokedPotential(eeg, rpeaks, fs)
```

hepAmplitude	<i>Amplitude of the HEP in a component window</i>
--------------	---

Description

Amplitude of the HEP in a component window

Usage

```
hepAmplitude(hep, component = c(0.2, 0.4), fun = c("mean", "peak"))
```

Arguments

hep	A "hep" object.
component	c(from, to) seconds defining the component window (default c(0.2, 0.4), a common HEP interval).
fun	Summary over the window: "mean" (default) or "peak" (largest absolute value).

Value

A single amplitude value.

Examples

```
set.seed(1); fs <- 250; n <- 30 * fs
rpeaks <- seq(fs, n - fs, by = round(0.9 * fs))
hep <- heartbeatEvokedPotential(rnorm(n), rpeaks, fs)
hepAmplitude(hep, component = c(0.2, 0.4))
```

hepSignificance	<i>Permutation significance of the HEP component</i>
-----------------	--

Description

Tests whether the HEP component amplitude is larger than expected if the brain signal were not time-locked to the heartbeat, by recomputing the component amplitude for `n_perm` datasets with circularly shifted R-peaks.

Usage

```

hepSignificance(
  eeg,
  rpeaks,
  sampling_rate,
  window = c(-0.1, 0.6),
  baseline = c(-0.1, 0),
  component = c(0.2, 0.4),
  n_perm = 200
)

```

Arguments

eeg, rpeaks, sampling_rate, window, baseline
 As in [heartbeatEvokedPotential\(\)](#).

component Component window in seconds (see [hepAmplitude\(\)](#)).

n_perm Number of permutations (default 200).

Value

A list with observed (lcomponent amplitudel), null (permutation amplitudes), and p_value (one-sided).

Examples

```

set.seed(1); fs <- 250; n <- 20 * fs
rpeaks <- seq(fs, n - fs, by = round(0.9 * fs))
hepSignificance(rnorm(n), rpeaks, fs, n_perm = 50)$p_value

```

instantaneousPhase	<i>Instantaneous phase of a signal</i>
--------------------	--

Description

Analytic-signal (Hilbert) instantaneous phase, optionally after isolating a frequency band. Physiological oscillations (respiration, a cardiac rhythm) should be reasonably narrow-band for the phase to be meaningful; pass band to bandpass first.

Usage

```
instantaneousPhase(x, sampling_rate = 1, band = NULL)
```

Arguments

x Numeric signal.

sampling_rate Sampling rate in Hz (default 1).

band Optional c(low, high) passband in Hz.

Value

Numeric vector of wrapped phases in $(-\pi, \pi]$.

Examples

```
t <- seq(0, 10, by = 0.01)
ph <- instantaneousPhase(sin(2 * pi * 1 * t), sampling_rate = 100)
```

layerSimilarity	<i>Pairwise layer similarity</i>
-----------------	----------------------------------

Description

Cosine similarity between the (upper-triangular) edge-weight vectors of each pair of layers: 1 = identical coupling structure, 0 = disjoint.

Usage

```
layerSimilarity(x)
```

Arguments

x A "multiplex" object.

Value

A `n_layers` x `n_layers` similarity matrix.

Examples

```
L1 <- matrix(c(0,1,0, 1,0,1, 0,1,0), 3)
L2 <- matrix(c(0,0,1, 0,0,0, 1,0,0), 3)
layerSimilarity(multiplexNetwork(list(alpha = L1, beta = L2)))
```

moduleRoles	<i>Node module roles (Guimera-Amaral)</i>
-------------	---

Description

Combines the within-module degree z-score and participation coefficient into the seven-role classification of Guimera & Amaral (2005).

Usage

```
moduleRoles(adj, communities)
```

Arguments

`adj` A square adjacency matrix.

`communities` A vector of module labels, one per node.

Value

A data frame with node, z (within-module degree z-score), P (participation coefficient), and role.

Examples

```
adj <- matrix(0, 6, 6)
adj[1, 2:3] <- adj[2:3, 1] <- 1
adj[4, 5] <- adj[5, 4] <- adj[5, 6] <- adj[6, 5] <- 1
adj[3, 4] <- adj[4, 3] <- 1
moduleRoles(adj, c(1, 1, 1, 2, 2, 2))
```

multiplexDegree	<i>Per-layer and overlapping (multiplex) degree</i>
-----------------	---

Description

Per-layer and overlapping (multiplex) degree

Usage

```
multiplexDegree(x)
```

Arguments

`x` A "multiplex" object.

Value

A list with `per_layer` (nodes x layers degree matrix) and `overlapping` (summed degree across layers per node).

Examples

```
L1 <- matrix(c(0,1,0, 1,0,1, 0,1,0), 3)
L2 <- matrix(c(0,0,1, 0,0,0, 1,0,0), 3)
mp <- multiplexNetwork(list(alpha = L1, beta = L2))
multiplexDegree(mp)
```

multiplexNetwork	<i>Construct a multiplex network</i>
------------------	--------------------------------------

Description

Construct a multiplex network

Usage

```
multiplexNetwork(layers, node_names = NULL)
```

Arguments

layers	A named list of adjacency matrices, all the same size and node order (one layer per coupling channel or frequency band).
node_names	Optional node names (default: the row names of the first layer, or $V_1 \dots V_n$).

Value

An object of class "multiplex": a list with layers, nodes, n_layers, layer_names.

Examples

```
L1 <- matrix(c(0,1,0, 1,0,1, 0,1,0), 3)
L2 <- matrix(c(0,0,1, 0,0,0, 1,0,0), 3)
mp <- multiplexNetwork(list(alpha = L1, beta = L2))
```

multiplexParticipation	<i>Multiplex participation coefficient</i>
------------------------	--

Description

How evenly a node's connections spread across layers. 0 = all links confined to one layer; 1 = links distributed uniformly across all layers (Battiston et al. 2014).

Usage

```
multiplexParticipation(x)
```

Arguments

x	A "multiplex" object.
---	-----------------------

Value

A named numeric vector per node, in $[0, 1]$.

Examples

```
L1 <- matrix(c(0,1,0, 1,0,1, 0,1,0), 3)
L2 <- matrix(c(0,0,1, 0,0,0, 1,0,0), 3)
multiplexParticipation(multiplexNetwork(list(alpha = L1, beta = L2)))
```

networkMetrics

Graph-topology metrics of a physiological network

Description

Computes the standard network-science metrics on a weighted physiological network: node-level (degree, strength, weighted clustering, betweenness, closeness, eigenvector centrality, local efficiency) and global (density, transitivity, characteristic path length, global/local efficiency, degree assortativity, and modularity from a community partition). Edge weights are treated as coupling strengths; path-based metrics use distances $1/\text{weight}$ (stronger coupling = shorter path). Delegates to `igraph`.

Usage

```
networkMetrics(x, directed = FALSE, community_method = "louvain")
```

Arguments

`x` A `tds_network` or a weighted adjacency matrix.

`directed` Treat the network as directed (default FALSE).

`community_method` Community algorithm for the modularity summary (see [communityDetection\(\)](#)).

Value

A `network_metrics` object: global (named list), node (data frame of per-node metrics), membership, and directed.

See Also

[communityDetection\(\)](#), [smallWorldness\(\)](#), [moduleRoles\(\)](#)

Examples

```
set.seed(1)
A <- matrix(runif(64, 0, 0.3), 8, 8); A <- (A + t(A)) / 2; diag(A) <- 0
nm <- networkMetrics(A)
nm$global$global_efficiency
```

nmPhaseSynchronization

Scan n:m phase-locking over a grid of ratios

Description

Computes $\lambda_{n,m}$ for every coprime (n, m) in the requested range and returns the dominant ratio. The frequency ratio f1 : f2 implied by the winning (n, m) is m : n.

Usage

```
nmPhaseSynchronization(phi1, phi2, orders_n = 1:8, orders_m = 1:8)
```

Arguments

phi1, phi2 Instantaneous phases.
orders_n, orders_m Integer vectors of orders to scan.

Value

A list with table (data frame of n, m, lambda), and the dominant n, m, lambda, and ratio (a "f1:f2" string = m:n).

Examples

```
t <- seq(0, 120, by = 0.01)
resp <- sin(2 * pi * 0.25 * t)
card <- sin(2 * pi * 1.0 * t)                            # 4x faster: locked 4:1
pr <- instantaneousPhase(resp, 100)
pc <- instantaneousPhase(card, 100)
nmPhaseSynchronization(pc, pr)$ratio
```

participationCoefficient

Participation coefficient

Description

$P_i = 1 - \sum_s (\kappa_{is}/k_i)^2$, where κ_{is} is node i's summed connection weight to module s and k_i its total strength. 0 = all links inside one module; approaches 1 as links spread evenly over many modules.

Usage

```
participationCoefficient(adj, communities)
```

Arguments

adj A square (weighted or binary) adjacency matrix.
 communities A vector of module labels, one per node.

Value

A named numeric vector of participation coefficients in $[0, 1]$.

Examples

```
adj <- matrix(0, 4, 4); adj[1,2] <- adj[2,1] <- 1; adj[3,4] <- adj[4,3] <- 1
participationCoefficient(adj, c(1, 1, 2, 2))
```

phaseSynchronizationIndex
n:m phase-locking index

Description

The $n:m$ phase-synchronization index $\lambda_{n,m} = |\langle e^{i(n\phi_1 - m\phi_2)} \rangle|$, in $[0, 1]$. For two rhythms whose frequency ratio is $f_1 : f_2 = m : n$, the term $n\phi_1 - m\phi_2$ is (near-)constant and λ approaches 1.

Usage

```
phaseSynchronizationIndex(phi1, phi2, n = 1L, m = 1L)
```

Arguments

phi1, phi2 Instantaneous phases (e.g. from [instantaneousPhase\(\)](#)).
 n, m Integer orders (default 1:1).

Value

A single locking index in $[0, 1]$.

Examples

```
t <- seq(0, 60, by = 0.01)
p1 <- instantaneousPhase(sin(2 * pi * 1 * t), 100)
p2 <- instantaneousPhase(sin(2 * pi * 1 * t), 100)
phaseSynchronizationIndex(p1, p2)
```

physioNodeMatrix	<i>Assemble a physiological node matrix</i>
------------------	---

Description

Builds the time-by-nodes matrix that `timeDelayStability()` consumes, where each node is one physiological variable (an organ system or a derived feature such as an EEG band-power, heart rate, respiration amplitude, or an EMG envelope). Inputs are a named list of equal-length numeric vectors, a numeric matrix, or a data frame.

Usage

```
physioNodeMatrix(
  x,
  nodes = NULL,
  standardize = FALSE,
  assay = NULL,
  channels = NULL,
  feature = c("raw", "envelope"),
  target_rate = NULL
)
```

Arguments

<code>x</code>	A named list of equal-length numeric vectors, a numeric matrix (time x nodes), a data frame of numeric columns, or a <code>PhysioExperiment</code> / <code>MultiPhysioExperiment</code> object (channels/modalities become nodes).
<code>nodes</code>	Optional character vector of node names (defaults to the names of <code>x</code>).
<code>standardize</code>	If TRUE, z-normalise each column (default FALSE; TDS z-normalises per window internally, so this is usually unnecessary).
<code>assay</code>	For a <code>PhysioExperiment</code> , which assay to read (default: its default assay).
<code>channels</code>	For a <code>PhysioExperiment</code> , an optional subset of channels.
<code>feature</code>	Per-channel feature: "raw" (default) or "envelope" (the analytic-signal amplitude — often the right node series for coupling).
<code>target_rate</code>	Optional rate (Hz) to down-sample each channel to, by block-averaging. Required for a <code>MultiPhysioExperiment</code> whose experiments have differing sampling rates.

Details

In Network Physiology, coupling is assessed between the *dynamics* of these variables, so nodes are typically instantaneous features sampled on a common, relatively slow grid (e.g. 1 Hz), not the raw high-rate waveforms. Assemble such features with the relevant `Physio*` package and pass them here.

Value

A numeric matrix, time (rows) by nodes (columns), with column names.

Examples

```
nodes <- list(
  hr = cumsum(rnorm(500)),
  resp = sin(seq(0, 20 * pi, length.out = 500)),
  emg = abs(rnorm(500))
)
M <- physioNodeMatrix(nodes)
dim(M)
```

plotDelayStability	<i>Plot the optimal-lag series and stable periods for a pair</i>
--------------------	--

Description

Draws the classic Time-Delay-Stability trace: the per-window optimal lag with windows inside stable periods highlighted. Long flat stretches are the stable couplings TDS rewards.

Usage

```
plotDelayStability(x, i, j, ...)
```

Arguments

<code>x</code>	A "tds" object from <code>timeDelayStability()</code> .
<code>i, j</code>	The two nodes (names or indices) whose coupling to plot.
<code>...</code>	Passed to <code>graphics::plot()</code> .

Value

`invisible(NULL)`; called for the plot.

plotHEP	<i>Plot a heartbeat-evoked potential</i>
---------	--

Description

Plot a heartbeat-evoked potential

Usage

```
plotHEP(x, main = "Heartbeat-evoked potential", ...)
```

Arguments

x	A "hep" object.
main	Title.
...	Passed to <code>graphics::plot()</code> .

Value

`invisible(NULL)`.

plotSynchrogram	<i>Plot a cardiorespiratory synchrogram</i>
-----------------	---

Description

Plot a cardiorespiratory synchrogram

Usage

```
plotSynchrogram(  
  x,  
  sampling_rate = NULL,  
  main = "Cardiorespiratory synchrogram",  
  ...  
)
```

Arguments

x	A "synchrogram" data frame or a "cardioresp" object.
sampling_rate	Sampling rate, to label the x-axis in seconds (optional).
main	Plot title.
...	Passed to <code>graphics::plot()</code> .

Value

`invisible(NULL)`.

plotTDSnetwork	<i>Plot a TDS organ-interaction network</i>
----------------	---

Description

Circular node layout; edge width and colour encode TDS strength.

Usage

```
plotTDSnetwork(x, main = "TDS organ-interaction network", ...)
```

Arguments

x	A "tds_network" object.
main	Plot title.
...	Ignored.

Value

invisible(NULL).

plotTDSreconfiguration	<i>Plot network reconfiguration across states</i>
------------------------	---

Description

Heatmap of per-state node degree (states x nodes): how each system's number of stable couplings changes across physiological states.

Usage

```
plotTDSreconfiguration(x, main = "TDS network reconfiguration", ...)
```

Arguments

x	A "tds_state" object.
main	Plot title.
...	Ignored.

Value

invisible(NULL).

smallWorldness	<i>Small-worldness of a physiological network</i>
----------------	---

Description

Quantifies small-world organization on the binarized network: the Humphries-Gurney $\sigma = (C/C_{\text{rand}})/(L/L_{\text{rand}})$ (small-world when > 1 , comparing clustering and path length to degree-preserving random graphs) and the Telford $\omega = L_{\text{rand}}/L - C/C_{\text{latt}}$ (near 0 = small-world; $\rightarrow 1$ random; $\rightarrow -1$ lattice). Delegates the randomization and lattice references to `igraph`.

Usage

```
smallWorldness(x, n_rand = 50L, seed = NULL)
```

Arguments

<code>x</code>	A <code>tds_network</code> or a weighted adjacency matrix (binarized by presence of an edge).
<code>n_rand</code>	Number of degree-preserving random graphs (default 50).
<code>seed</code>	Optional integer seed for the randomization.

Value

A `small_world` object: `sigma`, `omega`, and the components `C`, `L`, `C_rand`, `L_rand`, `C_latt`, `n_rand`.

References

Humphries & Gurney 2008; Telford et al. 2011.

See Also

[networkMetrics\(\)](#)

Examples

```
g <- igraph::sample_smallworld(1, 40, 4, 0.05)      # a small-world graph
sw <- smallWorldness(as.matrix(igraph::as_adjacency_matrix(g)), n_rand = 20, seed = 1)
sw$sigma
```

supraAdjacency	<i>Supra-adjacency matrix of a multiplex</i>
----------------	--

Description

Builds the $(N \times L) \times (N \times L)$ block matrix with the layer adjacencies on the diagonal blocks and interlayer coupling on the identity off-diagonal blocks (categorical coupling of a node to its own replicas across layers).

Usage

```
supraAdjacency(x, interlayer = 1)
```

Arguments

x	A "multiplex" object.
interlayer	Interlayer coupling weight (default 1).

Value

A $(N \times L) \times (N \times L)$ numeric matrix.

Examples

```
L1 <- matrix(c(0,1,0, 1,0,1, 0,1,0), 3)
L2 <- matrix(c(0,0,1, 0,0,0, 1,0,0), 3)
supraAdjacency(multiplexNetwork(list(alpha = L1, beta = L2)))
```

synchrogram	<i>Cardiorespiratory synchrogram</i>
-------------	--------------------------------------

Description

The classic Schafer-Rosenblum synchrogram: the respiratory phase observed at each heartbeat, wrapped into m respiratory cycles. Under $n:m$ locking (n heartbeats per m breaths) the points collapse onto n horizontal bands.

Usage

```
synchrogram(resp_phase, rpeaks, m = 1L)
```

Arguments

resp_phase	Instantaneous respiratory phase (length = n samples), e.g. from instantaneousPhase() .
rpeaks	Integer R-peak sample indices.
m	Number of respiratory cycles to wrap into (default 1).

Value

A data frame with beat (R-peak sample index) and psi (respiratory phase at that beat, in $[0, m)$), of class "synchrogram".

Examples

```
t <- seq(0, 120, by = 0.01); fs <- 100
resp_phase <- instantaneousPhase(sin(2 * pi * 0.25 * t), fs)
rpeaks <- which(diff(sign(sin(2 * pi * 1.0 * t))) > 0) # ~1 Hz beats
sg <- synchrogram(resp_phase, rpeaks, m = 1)
```

tdsNetwork

*Build an organ-interaction network from TDS***Description**

Thresholds a TDS matrix into a weighted, undirected network: nodes are physiological systems and edges are stable dynamic couplings whose weight is the TDS percentage. Edges below threshold are removed.

Usage

```
tdsNetwork(
  x,
  threshold = NULL,
  surrogate = FALSE,
  X = NULL,
  n_surrogates = 50,
  quantile = 0.95
)
```

Arguments

x	A "tds" object from timeDelayStability() , or a symmetric numeric TDS matrix.
threshold	Minimum TDS percentage for an edge to be kept. If NULL (default) and surrogate = TRUE, a surrogate threshold is computed with tdsSurrogateThreshold() ; otherwise a value must be supplied.
surrogate	Logical; if TRUE and threshold is NULL, derive the threshold from phase-randomised surrogates. Requires x to be a "tds" object (so the underlying series and windowing are available) — supply the original node matrix via X.
X	Original node matrix, required when surrogate = TRUE.
n_surrogates, quantile	Passed to tdsSurrogateThreshold() .

Value

An object of class "tds_network": a list with adjacency (weighted, thresholded), threshold, nodes, degree (number of links per node), strength (summed edge weight per node), and edges (a data frame of the surviving links).

Examples

```
set.seed(1)
n <- 3000
x <- as.numeric(stats::filter(rnorm(n), rep(1/5, 5), sides = 2)); x[is.na(x)] <- 0
y <- c(rep(0, 5), x[seq_len(n - 5)]) + rnorm(n, sd = 0.3)
z <- rnorm(n)
X <- cbind(brain = x, heart = y, muscle = z)
res <- timeDelayStability(X, window = 100, step = 50, max_lag = 20, min_stable = 3)
net <- tdsNetwork(res, threshold = 40)
net$edges
```

tdsNetworkByState

State-resolved organ-interaction networks

Description

Builds one TDS network per physiological state and summarises how the network reconfigures between states. TDS is computed continuously over the whole recording; each window is assigned to a state, and a pair's per-state TDS is the percentage of that state's windows that fall inside a stable coupling period.

Usage

```
tdsNetworkByState(
  X,
  states,
  sampling_rate = 1,
  window = NULL,
  step = NULL,
  max_lag = NULL,
  window_sec = NULL,
  step_sec = NULL,
  max_lag_sec = NULL,
  stability_tol = 1L,
  min_stable = 4L,
  threshold = NULL,
  n_surrogates = 50,
  min_windows = 3L
)
```

Arguments

<code>X</code>	Node matrix (or any input accepted by <code>timeDelayStability()</code>).
<code>states</code>	A per-sample state label vector of length <code>nrow(X)</code> (character or factor). Windows spanning a state boundary take the majority label.
<code>sampling_rate</code> , <code>window</code> , <code>step</code> , <code>max_lag</code> , <code>window_sec</code> , <code>step_sec</code> , <code>max_lag_sec</code> , <code>stability_tol</code> , <code>min_stable</code>	Windowing / stability arguments passed to <code>timeDelayStability()</code> .
<code>threshold</code>	Link threshold (TDS percentage). If NULL (default), a single global surrogate threshold is computed with <code>tdsSurrogateThreshold()</code> and applied to every state for comparability.
<code>n_surrogates</code>	Surrogates for the automatic threshold (default 50).
<code>min_windows</code>	Minimum number of windows a state must contain to be analysed (default 3); smaller states are dropped with a warning.

Value

An object of class "tds_state": a list with `states` (analysed levels), `tds` (named list of per-state TDS matrices), `networks` (named list of `tdsNetwork()` results), `threshold`, `window_state`, and `reconfiguration` (a data frame of per-state link count, density, and mean degree, plus a state x node degree matrix in degree).

Examples

```
set.seed(1)
n <- 4000
x <- as.numeric(stats::filter(rnorm(n), rep(1/5, 5), sides = 2)); x[is.na(x)] <- 0
half <- n / 2
# heart couples to brain only in state "B"
y <- rnorm(n, sd = 0.3)
y[(half + 6):n] <- x[half:(n - 6)] + rnorm(half - 5, sd = 0.3)
X <- cbind(brain = x, heart = y)
st <- rep(c("A", "B"), each = half)
res <- tdsNetworkByState(X, st, window = 100, step = 50, max_lag = 20,
                        min_stable = 3, threshold = 30)
res$reconfiguration
```

<code>tdsReconfiguration</code>	<i>Link-level reconfiguration between two states</i>
---------------------------------	--

Description

Returns, for every node pair, the TDS in each of two states and their difference — the edges that appear, vanish, or strengthen between states.

Usage

```
tdsReconfiguration(x, from, to)
```

Arguments

x	A "tds_state" object.
from, to	State labels to compare.

Value

A data frame of from, to (node pair), the TDS in each state, and del ta, ordered by absolute change.

tdsSurrogateThreshold	<i>Surrogate significance threshold for TDS links</i>
-----------------------	---

Description

Generates n_surrogates phase-randomised surrogate datasets (each column randomised independently, breaking cross-column timing while preserving each column's power spectrum), recomputes the TDS matrix for each, and returns a high quantile of the pooled off-diagonal surrogate TDS values. Links in the real network exceeding this threshold are unlikely to arise from matched spectra alone.

Usage

```
tdsSurrogateThreshold(X, n_surrogates = 50, quantile = 0.95, ...)
```

Arguments

X	Node matrix (or any input accepted by timeDelayStability()).
n_surrogates	Number of surrogate datasets (default 50).
quantile	Quantile of the surrogate TDS distribution to use as the threshold (default 0.95).
...	Windowing / stability arguments passed to timeDelayStability() .

Value

A single numeric TDS percentage threshold, with attribute "surrogate_tds" holding the pooled surrogate values.

Examples

```
set.seed(1)
n <- 2000
x <- as.numeric(stats::filter(rnorm(n), rep(1/5, 5), sides = 2)); x[is.na(x)] <- 0
y <- c(rep(0, 5), x[seq_len(n - 5)]) + rnorm(n, sd = 0.3)
X <- cbind(x = x, y = y)
thr <- tdsSurrogateThreshold(X, n_surrogates = 20,
                             window = 100, step = 50, max_lag = 20,
                             min_stable = 3)

thr
```

timeDelayStability	<i>Time Delay Stability across a multivariate physiological series</i>
--------------------	--

Description

Computes pairwise Time Delay Stability (TDS) for every pair of columns in *X*. Each column is treated as one physiological node (e.g. an EEG band-power, heart rate, respiration, an EMG envelope). The result is a symmetric matrix of TDS strengths (percentage of time each pair is stably coupled).

Usage

```
timeDelayStability(
  X,
  sampling_rate = 1,
  window = NULL,
  step = NULL,
  max_lag = NULL,
  window_sec = NULL,
  step_sec = NULL,
  max_lag_sec = NULL,
  stability_tol = 1L,
  min_stable = 4L,
  nodes = NULL
)
```

Arguments

<i>X</i>	A numeric matrix or data frame, time (rows) by nodes (columns), or an object accepted by <code>physioNodeMatrix()</code> (named list of equal-length vectors).
<i>sampling_rate</i>	Sampling rate in Hz of the rows of <i>X</i> (default 1).
<i>window</i> , <i>step</i> , <i>max_lag</i>	Window length, step, and maximum lag in samples (override the <i>*_sec</i> arguments when supplied).

`window_sec`, `step_sec`, `max_lag_sec`
 The same quantities in **seconds**. Defaults: `window_sec` = 60, `step` = `window`/2, `max_lag` = `window`/4.
`stability_tol` Maximum change in optimal lag (in samples) between consecutive windows for the pair to be considered stable (default 1).
`min_stable` Minimum number of consecutive windows forming a stable period (default 4).
`nodes` Optional character vector of node names (defaults to the column names of `X`).

Details

Windowing may be given in samples (`window`, `step`, `max_lag`) or in seconds (`window_sec`, `step_sec`, `max_lag_sec`, using `sampling_rate`). Sample arguments take precedence. Sensible defaults derive `step` and `max_lag` from the `window` when unspecified.

Value

An object of class "tds": a list with `tds` (nodes x nodes TDS percentage matrix, diagonal 0), `stable_lag` (nodes x nodes dominant stable lag in samples), `pairs` (per-pair detail incl. `tau0` and `in_stable`), `nodes`, `n_windows`, and `params`.

References

Bashan et al. (2012) [doi:10.1038/ncomms1705](https://doi.org/10.1038/ncomms1705).

Examples

```

set.seed(1)
n <- 3000
x <- as.numeric(stats::filter(rnorm(n), rep(1/5, 5), sides = 2))
x[is.na(x)] <- 0
y <- c(rep(0, 5), x[seq_len(n - 5)]) + rnorm(n, sd = 0.3) # y lags x by 5
z <- rnorm(n) # unrelated
X <- cbind(x = x, y = y, z = z)
res <- timeDelayStability(X, window = 100, step = 50, max_lag = 20,
  min_stable = 3)
res$tds

```

`withinModuleDegreeZ` *Within-module degree z-score*

Description

Standardises each node's within-module strength relative to the other nodes in its module. High values mark within-module hubs.

Usage

```
withinModuleDegreeZ(adj, communities)
```

Arguments

`adj` A square adjacency matrix.
`communities` A vector of module labels, one per node.

Value

A named numeric vector of z-scores.

Examples

```
adj <- matrix(0, 6, 6)
adj[1, 2:3] <- adj[2:3, 1] <- 1                    # module 1: node 1 hub
adj[4, 5] <- adj[5, 4] <- adj[5, 6] <- adj[6, 5] <- 1   # module 2: 4-5-6 path
adj[3, 4] <- adj[4, 3] <- 1                    # bridge
withinModuleDegreeZ(adj, c(1, 1, 1, 2, 2, 2))
```

Index

aggregateNetwork, [3](#)

cardiacPhaseFromPeaks, [3](#)

cardiorespiratoryCoupling, [4](#)

communityDetection, [5](#)

communityDetection(), [12](#)

graphics::plot(), [16, 17](#)

heartbeatEvokedPotential, [6](#)

heartbeatEvokedPotential(), [8](#)

hepAmplitude, [7](#)

hepAmplitude(), [8](#)

hepSignificance, [7](#)

instantaneousPhase, [8](#)

instantaneousPhase(), [14, 20](#)

layerSimilarity, [9](#)

moduleRoles, [9](#)

moduleRoles(), [5, 12](#)

multiplexDegree, [10](#)

multiplexNetwork, [11](#)

multiplexParticipation, [11](#)

networkMetrics, [12](#)

networkMetrics(), [5, 19](#)

nmPhaseSynchronization, [13](#)

participationCoefficient, [13](#)

participationCoefficient(), [5](#)

phaseSynchronizationIndex, [14](#)

physioNodeMatrix, [15](#)

physioNodeMatrix(), [25](#)

plotDelayStability, [16](#)

plotHEP, [17](#)

plotSynchrogram, [17](#)

plotTDSnetwork, [18](#)

plotTDSreconfiguration, [18](#)

smallWorldness, [19](#)

smallWorldness(), [12](#)

supraAdjacency, [20](#)

synchrogram, [20](#)

tdsNetwork, [21](#)

tdsNetwork(), [5, 23](#)

tdsNetworkByState, [22](#)

tdsReconfiguration, [23](#)

tdsSurrogateThreshold, [24](#)

tdsSurrogateThreshold(), [21, 23](#)

timeDelayStability, [25](#)

timeDelayStability(), [15, 16, 21, 23, 24](#)

withinModuleDegreeZ, [26](#)