Package: PhysioCrossModal (via r-universe)

May 16, 2026

Title Cross-Modal Coupling Analysis for PhysioExperiment Objects

Version 0.4.0

Author Yusuke Matsui

Maintainer Yusuke Matsui <mail.to.matsui@gmail.com>

Description Provides cross-modal coupling, connectivity, and synchrony analysis between physiological signals of different modalities (EEG, EMG, ECG, EDA, MoCap, fNIRS, etc.). Introduces a MultiPhysioExperiment container class for holding multiple PhysioExperiment objects at different sampling rates with temporal alignment. Includes spectral coherence, phase synchrony (PLV, PLI, wPLI), directed coupling (Granger causality), and time-domain coupling (cross-correlation) methods.

Depends R (>= 4.2), PhysioCore

Imports methods, parallel, SummarizedExperiment, S4Vectors, stats

Suggests signal, PhysioPreprocess, testthat (>= 3.2.0), ggplot2, knitr, rmarkdown

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Collate 'MultiPhysioExperiment-class.R' 'MultiPhysioExperiment-methods.R' 'utils-spectral.R' 'align-signals.R' 'coupling-spectral.R' 'coupling-phase.R' 'coupling-directed.R' 'coupling-timedom.R' 'coupling-wavelet.R' 'coupling-wrapper.R' 'coupling-matrix.R' 'stats-significance.R' 'benchmark-lodo.R' 'crossmodal-data.R' 'vis-coupling.R' 'zzz.R'

RoxygenNote 7.3.3

Config/pak/sysreqs zlib1g-dev

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

Date/Publication 2026-03-16 11:31:32 UTC

RemoteUrl https://github.com/x-biosignal/PhysioCrossModal

RemoteRef HEAD

RemoteSha 1eb22e10e887da5546e621f3d4e5afad99bdef8c

Contents

`.onLoad` *Package on-load hook*

Description

Package on-load hook

Usage

```
.onLoad(libname, pkg)
```

Arguments

`libname`	Library path.
`pkg`	Package name.

`[,MultiPhysioExperiment,ANY,ANY,ANY-method`
Subset a MultiPhysioExperiment by time range and/or modality

Description

Provides flexible subsetting of `MultiPhysioExperiment` objects:

- `mpe[, "eeg"]` – select modalities by name, returns a new MPE
- `mpe[c(1.0, 5.0), ]` – select a time window in seconds across all modalities, accounting for each modality's sampling rate
- `mpe[c(1.0, 5.0), "eeg"]` – both time and modality subsetting

Usage

```
## S4 method for signature 'MultiPhysioExperiment,ANY,ANY,ANY'
x[i, j, ..., drop = FALSE]
```

Arguments

`x`	A `MultiPhysioExperiment` object.
`i`	Numeric vector of length 2 specifying time range `[tmin, tmax]` in seconds, or missing.
`j`	Character vector of modality names, or missing.
`...`	Additional arguments (not used).
`drop`	Logical (not used).

Details

When `i` is a numeric vector of length 2, it is interpreted as a time range `[tmin, tmax]` in seconds. For each modality, the appropriate sample indices are computed from its sampling rate: `start_idx = floor(tmin * sr) + 1`, `end_idx = floor(tmax * sr) + 1`, clamped to valid bounds.

Value

A `MultiPhysioExperiment` object with the selected modalities and/or time windows.

Examples

```
eeg_data <- matrix(rnorm(2500 * 4), nrow = 2500, ncol = 4)
emg_data <- matrix(rnorm(5000 * 2), nrow = 5000, ncol = 2)
pe_eeg <- PhysioExperiment(
  assays = list(raw = eeg_data),
  samplingRate = 500
)
pe_emg <- PhysioExperiment(
  assays = list(raw = emg_data),
  samplingRate = 1000
)
mpe <- MultiPhysioExperiment(EEG = pe_eeg, EMG = pe_emg)

# Subset by modality
mpe_eeg <- mpe[, "EEG"]
modalities(mpe_eeg)  # "EEG"

# Subset by time range (seconds)
mpe_win <- mpe[c(1.0, 3.0), ]

# Combined
mpe_sub <- mpe[c(1.0, 3.0), "EEG"]
```

`[[,MultiPhysioExperiment,ANY,ANY-method`

Extract a single modality from a MultiPhysioExperiment

Description

Extract a single modality from a MultiPhysioExperiment

Usage

```
## S4 method for signature 'MultiPhysioExperiment,ANY,ANY'
x[[i, j, ...]]
```

Arguments

x	A MultiPhysioExperiment object.
i	Modality name (character) or index (integer).
j	Not used.
...	Additional arguments (not used).

Value

A PhysioExperiment object.

Examples

```
eeg_data <- matrix(rnorm(500 * 2), nrow = 500, ncol = 2)
pe_eeg <- PhysioExperiment(
  assays = list(raw = eeg_data),
  samplingRate = 250
)
mpe <- MultiPhysioExperiment(EEG = pe_eeg)
mpe[["EEG"]]
```

---

alignment *Get or set temporal alignment metadata*

Description

Get or set temporal alignment metadata

Usage

```
alignment(x)

## S4 method for signature 'MultiPhysioExperiment'
alignment(x)

alignment(x) <- value

## S4 replacement method for signature 'MultiPhysioExperiment'
alignment(x) <- value
```

Arguments

x	A MultiPhysioExperiment object.
value	A DataFrame with alignment metadata.

Value

A DataFrame.

See Also

alignSignals(), experiments(), samplingRates()

Examples

```
eeg_data <- matrix(rnorm(500 * 2), nrow = 500, ncol = 2)
pe_eeg <- PhysioExperiment(
  assays = list(raw = eeg_data),
  samplingRate = 250
)
mpe <- MultiPhysioExperiment(EEG = pe_eeg)
alignment(mpe)
```

---

alignSignals *Align multiple PhysioExperiment objects to a common sampling rate*

Description

Takes two or more named PhysioExperiment objects and resamples them so they all share a single sampling rate, then wraps the result in a MultiPhysioExperiment.

Usage

```
alignSignals(
  ...,
  method = c("lowest_rate", "common_rate", "resample"),
  target_rate = NULL
)
```

Arguments

...	Named PhysioExperiment objects.
method	Character: one of "lowest_rate" (default), "common_rate", or "resample".
target_rate	Numeric scalar: required when method = "resample".

Details

Three alignment strategies are available:

"lowest_rate" (default) Resample all signals to the lowest sampling rate present.

"common_rate" Resample all signals to the highest sampling rate present.

"resample" Resample all signals to the rate given by target_rate (which must be supplied).

If every input already shares the same sampling rate, no resampling is performed regardless of the chosen method.

Value

A MultiPhysioExperiment containing the (possibly resampled) input objects.

See Also

alignToRate(), mergePhysio(), MultiPhysioExperiment(), couplingAnalysis()

Examples

```
eeg <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(500 * 2), nrow = 500)),
  samplingRate = 500
)
emg <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(1000 * 2), nrow = 1000)),
  samplingRate = 1000
)
mpe <- alignSignals(EEG = eeg, EMG = emg, method = "lowest_rate")
```

---

alignToRate *Resample a PhysioExperiment to a target sampling rate*

Description

Resamples all assays in a PhysioExperiment object so that the resulting data correspond to the specified target sampling rate. If the **PhysioPreprocess** package is available and method = "linear", its resample() function is used; otherwise an internal interpolation is applied.

Usage

```
alignToRate(x, target_rate, method = c("linear", "spline", "fft"))
```

Arguments

x	A PhysioExperiment object.
target_rate	Numeric scalar: desired sampling rate in Hz.
method	Character: resampling method. One of "linear" (default), "spline", or "fft".

Value

A new PhysioExperiment with resampled data and updated samplingRate. The assay name is preserved from the original.

See Also

alignSignals(), mergePhysio(), MultiPhysioExperiment()

Examples

```
eeg_data <- matrix(rnorm(1000 * 4), nrow = 1000, ncol = 4)
pe <- PhysioExperiment(
  assays = list(raw = eeg_data),
  colData = S4Vectors::DataFrame(
    label = paste0("Ch", 1:4),
    type = rep("EEG", 4)
  ),
  samplingRate = 1000
)
pe500 <- alignToRate(pe, target_rate = 500)
samplingRate(pe500)  # 500
```

`bootstrapCI` *Bootstrap confidence interval for coupling*

Description

Computes a bootstrap confidence interval for the coupling statistic between two signals using the moving-block bootstrap (to preserve temporal autocorrelation).

Usage

```
bootstrapCI(
  x,
  y = NULL,
  sr = NULL,
  method,
  n_boot = 199L,
  ci = 0.95,
  block_len = NULL,
  modality_x = NULL,
  modality_y = NULL,
  channels_x = 1L,
  channels_y = 1L,
  cores = 1L,
  ...
)
```

Arguments

`x`	Numeric vector, PhysioExperiment, or MultiPhysioExperiment.
`y`	Numeric vector or PhysioExperiment, or NULL when `x` is an MPE.
`sr`	Numeric sampling rate in Hz (required when x/y are numeric).
`method`	Character coupling method (same options as `surrogateTest`).
`n_boot`	Integer number of bootstrap replicates (default 199).

`ci` Numeric confidence level (default 0.95).

`block_len` Integer block length for block bootstrap, or NULL for automatic (`ceiling(sqrt(n))`).

`modality_x`, `modality_y`
Character modality names for MPE input.

`channels_x`, `channels_y`
Integer channel indices (default 1).

`cores` Integer number of parallel cores to use (default `1L`). When `cores > 1`, bootstrap replicates are computed in parallel.

`...` Additional arguments passed to the coupling function.

Value

A list with components:

observed The full coupling result from the original signals.

statistic Numeric scalar: the extracted coupling statistic.

ci_lower Numeric scalar: lower CI bound.

ci_upper Numeric scalar: upper CI bound.

ci_level Numeric scalar: confidence level used.

boot_distribution Numeric vector of bootstrap statistics.

References

Efron, B., & Tibshirani, R. J. (1993). *An Introduction to the Bootstrap*. Chapman & Hall/CRC.

See Also

`surrogateTest()`, `couplingAnalysis()`

Examples

```
sr <- 500
t <- seq(0, 2, length.out = sr * 2)
x <- sin(2 * pi * 10 * t) + 0.3 * rnorm(length(t))
y <- 0.8 * sin(2 * pi * 10 * t) + 0.3 * rnorm(length(t))
result <- bootstrapCI(x, y, sr = sr, method = "coherence",
                      n_boot = 19, nperseg = 128L)
c(result$ci_lower, result$ci_upper)
```

coherence *Magnitude-squared coherence between two signals*

Description

Computes the magnitude-squared coherence between two signals, which quantifies the linear frequency-domain relationship between them. The coherence is defined as:

$$C_{xy}(f) = \frac{|S_{xy}(f)|^2}{S_{xx}(f) \cdot S_{yy}(f)}$$

where S_{xy} is the cross-spectral density and S_{xx}, S_{yy} are the auto-spectral densities.

Usage

```
coherence(
  x,
  y = NULL,
  sr = NULL,
  freq_range = NULL,
  nperseg = 256L,
  noverlap = NULL,
  method = c("welch", "multitaper"),
  modality_x = NULL,
  modality_y = NULL,
  channels_x = 1L,
  channels_y = 1L
)
```

Arguments

x	Numeric vector, PhysioExperiment, or MultiPhysioExperiment.
y	Numeric vector or PhysioExperiment, or NULL when x is an MPE.
sr	Numeric sampling rate in Hz (required when x/y are numeric).
freq_range	Numeric vector of length 2 giving the frequency range `c(low, high)` in Hz to restrict the output. NULL returns all frequencies.
nperseg	Integer segment length for Welch's method (default 256).
noverlap	Integer overlap between segments, or NULL for `floor(nperseg / 2)`.
method	Character spectral estimation method. Currently only `"welch"` is implemented; `"multitaper"` is reserved for future use.
modality_x	Character modality name in MPE for the x signal.
modality_y	Character modality name in MPE for the y signal.
channels_x	Integer which channel to extract from x (default 1).
channels_y	Integer which channel to extract from y (default 1).

Details

Unlike the `coherence()` function in PhysioAnalysis (which computes coherence across all channel pairs within a single PhysioExperiment), this function is designed for cross-modal analysis: it takes two separate signals (or PhysioExperiment / MultiPhysioExperiment objects) and computes the pairwise coherence between them.

Value

A named list with components:

coherence Numeric vector of magnitude-squared coherence values in $[0, 1]$ at each frequency.

frequencies Numeric vector of corresponding frequencies in Hz.

confidence_limit Numeric scalar giving the 95\ threshold: `1 - alpha^(1 / (L - 1))` where alpha = 0.05 and L is the number of segments (Halliday et al. 1995). Coherence values above this limit are statistically significant.

n_segments Integer number of segments used in Welch estimation.

References

Carter, G. C. (1987). Coherence and time delay estimation. *Proceedings of the IEEE*, 75(2), 236–255.

Halliday, D. M., Rosenberg, J. R., Amjad, A. M., Breeze, P., Conway, B. A., & Farmer, S. F. (1995). A framework for the analysis of mixed time series/point process data – theory and application to the study of physiological tremor, single motor unit discharges and electromyograms. *Progress in Biophysics and Molecular Biology*, 64(2–3), 237–278.

See Also

`crossSpectrum()`, `multitaperCoherence()`, `coherenceMatrix()`, `couplingAnalysis()`

Examples

```
# Two coupled 20 Hz sinusoids
sr <- 500
t <- seq(0, 10, length.out = sr * 10)
x <- sin(2 * pi * 20 * t) + 0.2 * rnorm(length(t))
y <- 0.8 * sin(2 * pi * 20 * t) + 0.2 * rnorm(length(t))
result <- coherence(x, y, sr = sr)
plot(result$frequencies, result$coherence, type = "l",
     xlab = "Frequency (Hz)", ylab = "Coherence")
abline(h = result$confidence_limit, lty = 2, col = "red")
```

`coherenceMatrix` *Coherence matrix across channel pairs*

Description

Computes magnitude-squared coherence for all pairs of channels between two sets of signals (or between all channels within a single set). Returns a matrix of peak coherence values plus the full spectra.

Usage

```
coherenceMatrix(
  x,
  y = NULL,
  sr = NULL,
  channels_x = NULL,
  channels_y = NULL,
  modality_x = NULL,
  modality_y = NULL,
  ...
)
```

Arguments

`x`	A PhysioExperiment or MultiPhysioExperiment object.
`y`	A PhysioExperiment, or NULL when `x` is an MPE.
`sr`	Numeric sampling rate in Hz (used only for numeric inputs).
`channels_x`	Integer vector of channel indices for x, or NULL for all channels.
`channels_y`	Integer vector of channel indices for y, or NULL for all channels.
`modality_x, modality_y`	Character modality names for MPE input.
`...`	Additional arguments passed to `coherence` (e.g. `nperseg`, `freq_range`).

Value

A list with components:

matrix Numeric matrix of peak coherence values, with dimensions `[n_channels_x x n_channels_y]`.

spectra List of lists containing the full coherence result for each pair.

frequencies Numeric vector of frequencies from the first pair.

channel_names_x Character vector of x channel names.

channel_names_y Character vector of y channel names.

References

Carter, G. C. (1987). Coherence and time delay estimation. *Proceedings of the IEEE*, 75(2), 236–255.

See Also

coherence(), couplingMatrix(), plotCouplingMatrix(), surrogateMatrixTest()

Examples

```
sr <- 200; n <- sr * 2
pe1 <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(n * 2), nrow = n)),
  samplingRate = sr
)
pe2 <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(n * 2), nrow = n)),
  samplingRate = sr
)
result <- coherenceMatrix(pe1, pe2, nperseg = 64L)
result$matrix
```

---

couplingAnalysis *Unified interface for cross-modal coupling analysis*

Description

Dispatches to the appropriate coupling function based on the `method` parameter. Accepts the same flexible inputs as the underlying functions: two numeric vectors (with `sr`), two PhysioExperiment objects, or a MultiPhysioExperiment with named modalities.

Usage

```
couplingAnalysis(
  x,
  y = NULL,
  mpe = NULL,
  modality_x = NULL,
  modality_y = NULL,
  channels_x = 1L,
  channels_y = 1L,
  method = c("coherence", "plv", "pli", "wpli", "granger", "crosscorrelation",
    "wavelet_coherence", "wavelet_plv", "multitaper_coherence"),
  sr = NULL,
  ...
)
```

Arguments

x	Numeric vector, PhysioExperiment, or MultiPhysioExperiment. When mpe is provided this argument is ignored.
y	Numeric vector or PhysioExperiment (NULL when x is an MPE or when mpe is provided).
mpe	A MultiPhysioExperiment object. When supplied, x and y are ignored and signals are extracted from mpe using modality_x / modality_y.
modality_x, modality_y	Character names of the modalities to extract from mpe.
channels_x, channels_y	Integer channel indices to extract (default 1).
method	Character string specifying the coupling method. One of: "coherence", "plv", "pli", "wpli", "granger", "crosscorrelation", "wavelet_coherence", or "wavelet_plv".
sr	Numeric sampling rate in Hz. Required when x and y are numeric vectors.
...	Additional arguments passed to the specific coupling function (e.g. freq_band, order, max_lag, nperseg, etc.).

Value

The result from the dispatched coupling function. See individual function documentation for details:

- "coherence": see coherence
- "plv": see phaseLockingValue
- "pli": see phaseLagIndex
- "wpli": see weightedPLI
- "granger": see grangerCausality
- "crosscorrelation": see crossCorrelation
- "wavelet_coherence": see waveletCoherence
- "wavelet_plv": see waveletPLV

References

Carter, G. C. (1987). Coherence and time delay estimation. *Proceedings of the IEEE*, 75(2), 236–255.

Lachaux, J.-P., Rodriguez, E., Martinerie, J., & Varela, F. J. (1999). Measuring phase synchrony in brain signals. *Human Brain Mapping*, 8(4), 194–208.

Granger, C. W. J. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, 37(3), 424–438.

See Also

coherence(), phaseLockingValue(), grangerCausality(), crossCorrelation(), surrogateTest()

Examples

```
# Numeric vectors
sr <- 500
t <- seq(0, 10, length.out = sr * 10)
x <- sin(2 * pi * 20 * t) + 0.2 * rnorm(length(t))
y <- 0.8 * sin(2 * pi * 20 * t) + 0.2 * rnorm(length(t))

# Coherence
res <- couplingAnalysis(x, y, method = "coherence", sr = sr)

# Cross-correlation
res <- couplingAnalysis(x, y, method = "crosscorrelation", sr = sr)
```

`couplingMatrix` *Generic coupling matrix across channel pairs*

Description

Computes any coupling method for all pairs of channels between two sets of signals, extracting a scalar statistic for each pair. This is the multi-channel generalisation of `couplingAnalysis`.

Usage

```
couplingMatrix(
  x,
  y = NULL,
  sr = NULL,
  method,
  channels_x = NULL,
  channels_y = NULL,
  modality_x = NULL,
  modality_y = NULL,
  ...
)
```

Arguments

`x`	A PhysioExperiment or MultiPhysioExperiment object.
`y`	A PhysioExperiment, or NULL when `x` is an MPE.
`sr`	Numeric sampling rate in Hz.
`method`	Character coupling method, one of `"coherence"`, `"plv"`, `"pli"`, `"wpli"`, `"granger"`, `"crosscorrelation"`.
`channels_x, channels_y`	Integer vectors of channel indices, or NULL for all channels.
`modality_x, modality_y`	Character modality names for MPE input.
`...`	Additional arguments passed to the coupling function.

Value

A list with components:

matrix Numeric matrix of coupling values.

method Character method used.

channel_names_x Character vector of x channel names.

channel_names_y Character vector of y channel names.

References

Carter, G. C. (1987). Coherence and time delay estimation. *Proceedings of the IEEE*, 75(2), 236–255.

See Also

`couplingAnalysis()`, `coherenceMatrix()`, `plotCouplingMatrix()`, `surrogateMatrixTest()`

Examples

```
sr <- 200; n <- sr * 2
pe1 <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(n * 2), nrow = n)),
  samplingRate = sr
)
pe2 <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(n * 2), nrow = n)),
  samplingRate = sr
)
result <- couplingMatrix(pe1, pe2, method = "coherence", nperseg = 64L)
result$matrix
```

---

crossCorrelation *Cross-correlation between two signals*

Description

Computes the cross-correlation between two signals at various lags, measuring their time-domain coupling. Cross-correlation quantifies how similar one signal is to a time-shifted version of another. A positive peak lag indicates that y leads x (i.e., y must be shifted forward to align with x), while a negative peak lag indicates x leads y.

Usage

```
crossCorrelation(
  x,
  y = NULL,
  sr = NULL,
  max_lag = NULL,
  normalize = TRUE,
  modality_x = NULL,
  modality_y = NULL,
  channels_x = 1L,
  channels_y = 1L
)
```

Arguments

x	Numeric vector, PhysioExperiment, or MultiPhysioExperiment.
y	Numeric vector or PhysioExperiment, or NULL when x is a MultiPhysioExperiment.
sr	Numeric sampling rate in Hz (required when x/y are numeric).
max_lag	Integer maximum lag in samples. If NULL, defaults to `floor(length(x) / 4)`.
normalize	Logical; if TRUE (default), compute normalized cross-correlation (Pearson-like, values in [-1, 1]).
modality_x	Character modality name in MPE for the x signal.
modality_y	Character modality name in MPE for the y signal.
channels_x	Integer which channel to extract from x (default 1).
channels_y	Integer which channel to extract from y (default 1).

Details

The function accepts numeric vectors, PhysioExperiment objects, or a MultiPhysioExperiment with named modalities, using `.extract_signal_pair()` internally for flexible input handling.

Value

A named list with components:

correlation Numeric vector of cross-correlation values at each lag.

lags Integer vector of lag values in samples.

lag_seconds Numeric vector of lag values in seconds.

peak_lag Integer lag (in samples) at which the absolute correlation is maximised.

peak_lag_seconds Numeric peak lag converted to seconds.

peak_correlation Numeric cross-correlation value at the peak lag.

References

Chatfield, C. (2004). *The Analysis of Time Series: An Introduction* (6th ed.). Chapman & Hall/CRC.

See Also

slidingCrossCorrelation(), coherence(), couplingAnalysis()

Examples

```
# Simple example: y is a delayed copy of x
sr <- 500
set.seed(1)
x <- rnorm(1000)
y <- c(rep(0, 10), x[1:990])
result <- crossCorrelation(x, y, sr = sr)
result$peak_lag       # should be 10
result$peak_lag_seconds
```

---

crossSpectrum *Cross-spectral density between two signals*

Description

Computes the cross-spectral density (CSD) between two signals using Welch's method. The CSD captures both the magnitude and phase relationship between two signals as a function of frequency.

Usage

```
crossSpectrum(
  x,
  y = NULL,
  sr = NULL,
  nperseg = 256L,
  noverlap = NULL,
  modality_x = NULL,
  modality_y = NULL,
  channels_x = 1L,
  channels_y = 1L
)
```

Arguments

x	Numeric vector, PhysioExperiment, or MultiPhysioExperiment.
y	Numeric vector or PhysioExperiment, or NULL when x is an MPE.
sr	Numeric sampling rate in Hz (required when x/y are numeric).
nperseg	Integer segment length for Welch's method (default 256).
noverlap	Integer overlap between segments, or NULL for floor(nperseg / 2).
modality_x	Character modality name in MPE for the x signal.
modality_y	Character modality name in MPE for the y signal.
channels_x	Integer which channel to extract from x (default 1).
channels_y	Integer which channel to extract from y (default 1).

Value

A named list with components:

csd Complex vector of cross-spectral density values.

frequencies Numeric vector of corresponding frequencies in Hz.

magnitude Numeric vector of CSD magnitude (`Mod(csd)`).

phase Numeric vector of CSD phase in radians (`Arg(csd)`).

References

Carter, G. C. (1987). Coherence and time delay estimation. *Proceedings of the IEEE*, 75(2), 236–255.

Halliday, D. M., Rosenberg, J. R., Amjad, A. M., Breeze, P., Conway, B. A., & Farmer, S. F. (1995). A framework for the analysis of mixed time series/point process data – theory and application to the study of physiological tremor, single motor unit discharges and electromyograms. *Progress in Biophysics and Molecular Biology*, 64(2–3), 237–278.

See Also

`coherence()`, `multitaperCoherence()`, `couplingAnalysis()`

Examples

```
sr <- 500
t <- seq(0, 5, length.out = sr * 5)
x <- sin(2 * pi * 10 * t) + 0.1 * rnorm(length(t))
y <- cos(2 * pi * 10 * t) + 0.1 * rnorm(length(t))
result <- crossSpectrum(x, y, sr = sr)
plot(result$frequencies, result$magnitude, type = "l",
     xlab = "Frequency (Hz)", ylab = "|CSD|")
```

---

experiments *Get or set the experiments list*

Description

Get or set the experiments list

Usage

```
experiments(x)

## S4 method for signature 'MultiPhysioExperiment'
experiments(x)

experiments(x) <- value

## S4 replacement method for signature 'MultiPhysioExperiment'
experiments(x) <- value
```

Arguments

x	A MultiPhysioExperiment object.
value	A named list of PhysioExperiment objects.

Value

For the getter, a named list of PhysioExperiment objects. For the setter, the modified MultiPhysioExperiment (returned invisibly).

See Also

MultiPhysioExperiment(), modalities(), samplingRates()

Examples

```
eeg_data <- matrix(rnorm(500 * 2), nrow = 500, ncol = 2)
pe_eeg <- PhysioExperiment(
  assays = list(raw = eeg_data),
  samplingRate = 250
)
mpe <- MultiPhysioExperiment(EEG = pe_eeg)
experiments(mpe)
```

---

grangerCausality *Granger Causality Between Two Signals*

Description

Computes directed coupling between two physiological signals using Granger causality. The method fits bivariate autoregressive (AR) models to quantify how much one signal helps predict the other beyond its own past. Both time-domain and spectral (frequency-resolved) Granger causality are supported.

Usage

```
grangerCausality(
  x,
  y = NULL,
  sr = NULL,
  order = 5L,
  freq_range = NULL,
  method = c("parametric", "nonparametric"),
  modality_x = NULL,
  modality_y = NULL,
  channels_x = 1L,
  channels_y = 1L
)
```

Arguments

x	Numeric vector, PhysioExperiment, or MultiPhysioExperiment.
y	Numeric vector or PhysioExperiment, or NULL when x is a MultiPhysioExperiment.
sr	Numeric sampling rate in Hz (required when x/y are numeric vectors).
order	Integer AR model order (default 5). Higher order captures longer-range temporal dependencies but requires more data.
freq_range	Numeric vector of length 2 specifying the frequency band in Hz over which to compute spectral Granger causality. If NULL (default), returns time-domain GC values.
method	Character; one of "parametric" (default) or "nonparametric". Currently only parametric is implemented.
modality_x	Character modality name in MultiPhysioExperiment for the x signal.
modality_y	Character modality name in MultiPhysioExperiment for the y signal.
channels_x	Integer channel index to extract from x (default 1).
channels_y	Integer channel index to extract from y (default 1).

Details

For the time-domain (parametric) method, Granger causality from x to y is defined as:

$$GC_{x \to y} = \log(\sigma^2_{restricted} / \sigma^2_{unrestricted})$$

where the restricted model predicts y from its own past only, and the unrestricted model predicts y from both its own past and x's past.

For spectral Granger causality (when freq_range is specified), the AR coefficients are transformed to the frequency domain via the transfer function $H(f) = A(f)^{-1}$, and the spectral GC is computed as:

$$GC_{x \to y}(f) = \log(S_{yy}^{restricted}(f) / S_{yy}^{unrestricted}(f))$$

The returned values are then averaged over the specified frequency band.

Note: The spectral GC implementation uses separate restricted and unrestricted univariate AR models, which is an approximation to the full Geweke (1982) spectral decomposition based on a 2x2 bivariate VAR model. Results may differ from toolboxes that implement the full Geweke decomposition (e.g., MVGC, FieldTrip). For exact spectral GC, use the time-domain method (freq_range = NULL) which correctly implements the Granger (1969) log-ratio formulation.

Value

A list with components:

gc_xy Numeric Granger causality from x to y (x drives y). A positive value indicates that x's past helps predict y beyond y's own past.

gc_yx Numeric Granger causality from y to x (y drives x).

net_gc Numeric net Granger causality (gc_xy - gc_yx). Positive values indicate that x drives y more than y drives x.

order Integer AR model order used.

References

Granger, C. W. J. (1969). Investigating causal relations by econometric models and cross-spectral methods. Econometrica, 37(3), 424-438.

Geweke, J. (1982). Measurement of linear dependence and feedback between multiple time series. Journal of the American Statistical Association, 77(378), 304-313.

See Also

`coherence()`, `phaseLockingValue()`

Examples

```
# Create directed signals: x drives y with a 5-sample lag
set.seed(42)
n <- 5000
x <- rnorm(n)
y <- numeric(n)
for (i in 6:n) y[i] <- 0.6 * x[i - 5] + 0.4 * rnorm(1)

result <- grangerCausality(x, y, sr = 500, order = 10)
result$gc_xy   # Should be positive (x drives y)
result$net_gc  # Should be positive
```

`length,MultiPhysioExperiment-method`

Additional methods for MultiPhysioExperiment

Description

Provides subsetting (`[`), `length()`, and `names()` methods for `MultiPhysioExperiment` objects. Length of a MultiPhysioExperiment

Usage

```
## S4 method for signature 'MultiPhysioExperiment'
length(x)
```

Arguments

`x`	A `MultiPhysioExperiment` object.

Details

Returns the number of modalities (same as `nModalities`).

Value

Integer: number of modalities.

Examples

```
eeg_data <- matrix(rnorm(500 * 2), nrow = 500, ncol = 2)
pe_eeg <- PhysioExperiment(
  assays = list(raw = eeg_data),
  samplingRate = 250
)
mpe <- MultiPhysioExperiment(EEG = pe_eeg)
length(mpe)  # 1
```

---

lodoGeneralization *Leave-one-site-out generalization benchmark*

Description

Evaluates model transportability by training on all-but-one sites and testing on the held-out site (LODO). Supports continuous outcomes (`family = "gaussian"`) and binary outcomes (`family = "binomial"`).

Usage

```
lodoGeneralization(
  data,
  outcome,
  site,
  features = NULL,
  family = c("gaussian", "binomial"),
  positive_class = NULL,
  threshold = 0.5,
  min_train_rows = 20L,
  scale_features = TRUE
)
```

Arguments

`data`	Data frame containing outcome, site label, and features.
`outcome`	Outcome column name.
`site`	Site/facility column name used for LODO splitting.
`features`	Feature column names. If `NULL`, numeric columns excluding `outcome` and `site` are used.
`family`	Modeling family: `"gaussian"` or `"binomial"`.
`positive_class`	Positive class label for binomial metrics. If `NULL`, the second factor level is used.
`threshold`	Classification threshold for binomial predictions.
`min_train_rows`	Minimum training rows required per fold.
`scale_features`	Logical; z-score features using training statistics.

Value

A list with:

fold_metrics Per-site metrics.
predictions Row-level held-out predictions.
aggregate Mean metrics across folds.
settings Benchmark settings and feature list.

See Also

couplingAnalysis(), surrogateTest()

Examples

```
set.seed(1)
n <- 120
df <- data.frame(
  site_id = rep(c("A", "B", "C"), each = 40),
  f1 = rnorm(n),
  f2 = rnorm(n)
)
df$y <- 0.8 * df$f1 - 0.4 * df$f2 + rnorm(n, sd = 0.2)

res <- lodoGeneralization(
  data = df,
  outcome = "y",
  site = "site_id",
  family = "gaussian"
)
head(res$fold_metrics)
```

---

make_coupled_signals *Generate coupled sinusoidal signals*

Description

Creates a pair of sinusoidal signals with a shared oscillatory component at a specified coupling frequency and controllable noise level. Useful for demonstrating and testing spectral coherence, phase synchrony, and other cross-modal coupling measures.

Usage

```
make_coupled_signals(
  sr1 = 500,
  sr2 = 500,
  coupling_freq = 20,
  coupling_strength = 0.8,
  noise = 0.2,
  duration = 10
)
```

Arguments

`sr1`	Numeric sampling rate in Hz for the first signal (default 500).
`sr2`	Numeric sampling rate in Hz for the second signal (default 500).
`coupling_freq`	Numeric frequency in Hz of the shared oscillatory component (default 20).
`coupling_strength`	Numeric amplitude of the shared component (default 0.8).
`noise`	Numeric standard deviation of the additive Gaussian noise (default 0.2).
`duration`	Numeric duration in seconds (default 10).

Details

Both signals share the same sinusoidal component at `coupling_freq`, scaled by `coupling_strength`, with additive Gaussian noise scaled by `noise`.

Value

A named list with components:

x Numeric vector – first coupled signal.

y Numeric vector – second coupled signal.

sr_x Numeric sampling rate of signal x.

sr_y Numeric sampling rate of signal y.

coupling_freq Numeric coupling frequency used.

References

Carter, G. C. (1987). Coherence and time delay estimation. *Proceedings of the IEEE*, 75(2), 236–255.

See Also

`make_eeg_emg()`, `make_directed_signals()`, `coherence()`, `couplingAnalysis()`

Examples

```
signals <- make_coupled_signals(sr1 = 500, sr2 = 500,
                                coupling_freq = 20,
                                coupling_strength = 0.8,
                                noise = 0.2, duration = 10)
result <- coherence(signals$x, signals$y, sr = signals$sr_x)
```

`make_directed_signals` *Generate directed coupling signals*

Description

Creates a pair of signals where `x` drives `y` with a specified lag and coupling strength. Signal `x` is white noise, and `y` is a mixture of a lagged copy of `x` and independent noise. This is useful for testing directed coupling measures such as Granger causality.

Usage

```
make_directed_signals(n = 5000, sr = 500, lag_samples = 10, coupling = 0.7)
```

Arguments

`n`	Integer number of samples (default 5000).
`sr`	Numeric sampling rate in Hz (default 500).
`lag_samples`	Integer number of samples by which `x` leads `y` (default 10).
`coupling`	Numeric coupling strength in [0, 1] (default 0.7).

Details

The generating model is:

$$y(t) = \text{coupling} \cdot x(t - \text{lag\_samples}) + (1 - \text{coupling}) \cdot \epsilon(t)$$

where $\epsilon(t) \sim N(0, 1)$.

Value

A named list with components:

x Numeric vector – driving signal (white noise).

y Numeric vector – driven signal (lagged mixture).

sr Numeric sampling rate.

lag_samples Integer lag used.

References

Granger, C. W. J. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, 37(3), 424–438.

See Also

`grangerCausality()`, `make_coupled_signals()`, `crossCorrelation()`, `couplingAnalysis()`

Examples

```
signals <- make_directed_signals(n = 5000, sr = 500,
                                 lag_samples = 10, coupling = 0.7)
result <- grangerCausality(signals$x, signals$y, sr = signals$sr,
                           order = 15)
result$gc_xy   # should be positive (x drives y)
result$net_gc  # should be positive
```

---

make_eeg_emg *Generate simulated EEG-EMG PhysioExperiment pair*

Description

Creates a pair of `PhysioExperiment` objects simulating simultaneous EEG and EMG recordings with corticomuscular coherence (CMC) at a specified frequency. The first channel of each modality contains a shared oscillatory component; remaining channels contain independent noise.

Usage

```
make_eeg_emg(
  n_sec = 10,
  n_eeg_ch = 3,
  n_emg_ch = 2,
  sr_eeg = 500,
  sr_emg = 1000,
  cmc_freq = 20
)
```

Arguments

`n_sec`	Numeric recording duration in seconds (default 10).
`n_eeg_ch`	Integer number of EEG channels (default 3).
`n_emg_ch`	Integer number of EMG channels (default 2).
`sr_eeg`	Numeric EEG sampling rate in Hz (default 500).
`sr_emg`	Numeric EMG sampling rate in Hz (default 1000).
`cmc_freq`	Numeric corticomuscular coherence frequency in Hz (default 20).

Details

This function is useful for demonstrating and testing multi-channel and multi-modal coupling analyses such as coherence matrices and `MultiPhysioExperiment` workflows.

Value

A named list with components:

eeg A `PhysioExperiment` object with EEG data (`n_eeg_ch` channels at `sr_eeg` Hz).

emg A `PhysioExperiment` object with EMG data (`n_emg_ch` channels at `sr_emg` Hz).

References

Halliday, D. M., Rosenberg, J. R., Amjad, A. M., Breeze, P., Conway, B. A., & Farmer, S. F. (1995). A framework for the analysis of mixed time series/point process data – theory and application to the study of physiological tremor, single motor unit discharges and electromyograms. *Progress in Biophysics and Molecular Biology*, 64(2–3), 237–278.

See Also

`make_coupled_signals()`, `make_directed_signals()`, `MultiPhysioExperiment()`, `coherenceMatrix()`

Examples

```
data <- make_eeg_emg(n_sec = 5, n_eeg_ch = 3, n_emg_ch = 2)
mpe <- MultiPhysioExperiment(EEG = data$eeg, EMG = data$emg)
modalities(mpe)
```

`mergePhysio` *Merge two PhysioExperiment objects by combining channels*

Description

Horizontally concatenates the channels (columns) of two `PhysioExperiment` objects that share the **same** sampling rate. Channel labels are prefixed to avoid name collisions.

Usage

```
mergePhysio(x, y, prefix = c("x_", "y_"))
```

Arguments

`x`	A `PhysioExperiment` object.
`y`	A `PhysioExperiment` object.
`prefix`	Character vector of length 2: prefixes added to channel labels from `x` and `y` respectively (default `c("x_", "y_")`).

Details

Both objects must have the same number of time points (rows) and the same sampling rate. Only the first assay of each object is merged.

Value

A single `PhysioExperiment` with the columns of both inputs.

See Also

`alignToRate()`, `alignSignals()`, `MultiPhysioExperiment()`

Examples

```
pe1 <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(100 * 4), nrow = 100)),
  colData = S4Vectors::DataFrame(label = paste0("A", 1:4)),
  samplingRate = 500
)
pe2 <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(100 * 4), nrow = 100)),
  colData = S4Vectors::DataFrame(label = paste0("B", 1:4)),
  samplingRate = 500
)
merged <- mergePhysio(pe1, pe2)
```

---

modalities *Get modality names*

Description

Returns a character vector of the names of the modalities stored in the object.

Usage

```
modalities(x)

## S4 method for signature 'MultiPhysioExperiment'
modalities(x)
```

Arguments

x	A MultiPhysioExperiment object.

Value

Character vector of modality names (e.g., c("EEG", "EMG")).

See Also

experiments(), nModalities(), samplingRates()

Examples

```
eeg_data <- matrix(rnorm(500 * 2), nrow = 500, ncol = 2)
emg_data <- matrix(rnorm(1000 * 2), nrow = 1000, ncol = 2)
pe_eeg <- PhysioExperiment(
  assays = list(raw = eeg_data),
  samplingRate = 250
)
pe_emg <- PhysioExperiment(
  assays = list(raw = emg_data),
```

```
  samplingRate = 1000
)
mpe <- MultiPhysioExperiment(EEG = pe_eeg, EMG = pe_emg)
modalities(mpe)
```

`MultiPhysioExperiment` *Construct a MultiPhysioExperiment object*

Description

Creates a container that holds multiple `PhysioExperiment` objects recorded simultaneously at potentially different sampling rates.

Usage

```
MultiPhysioExperiment(..., experiments = list(), alignment = NULL)
```

Arguments

`...`	Named `PhysioExperiment` objects, one per modality.
`experiments`	Alternatively, a named list of `PhysioExperiment` objects. If both `...` and `experiments` are provided, they are combined.
`alignment`	Optional `DataFrame` with temporal alignment metadata. When `NULL` (the default), a default alignment table is built from the supplied experiments.

Value

A `MultiPhysioExperiment-class` instance containing the supplied experiments, alignment metadata, and an empty coupling results cache.

References

Huber, W., et al. (2015). "Orchestrating high-throughput genomic analysis with Bioconductor." Nature Methods, 12(2), 115-121. doi:10.1038/nmeth.3252

See Also

`experiments()`, `modalities()`, `samplingRates()`, `alignSignals()`, `couplingAnalysis()`

Examples

```
# Create two PhysioExperiment objects with different sampling rates
eeg_data <- matrix(rnorm(500 * 4), nrow = 500, ncol = 4)
emg_data <- matrix(rnorm(1000 * 2), nrow = 1000, ncol = 2)

pe_eeg <- PhysioExperiment(
  assays = list(raw = eeg_data),
  colData = S4Vectors::DataFrame(
```

```
    label = c("Fz", "Cz", "Pz", "Oz"),
    type = rep("EEG", 4)
  ),
  samplingRate = 250
)

pe_emg <- PhysioExperiment(
  assays = list(raw = emg_data),
  colData = S4Vectors::DataFrame(
    label = c("EMG1", "EMG2"),
    type = rep("EMG", 2)
  ),
  samplingRate = 1000
)

# Construct MultiPhysioExperiment
mpe <- MultiPhysioExperiment(EEG = pe_eeg, EMG = pe_emg)
mpe
```

`MultiPhysioExperiment-class` *MultiPhysioExperiment class definition*

Description

The `MultiPhysioExperiment` class holds multiple `PhysioExperiment` objects representing different signal modalities (e.g., EEG, EMG, ECG) recorded simultaneously but potentially at different sampling rates. It provides temporal alignment metadata and a cache for coupling analysis results.

Slots

`experiments` Named list of `PhysioExperiment` objects.

`alignment` `DataFrame` with temporal alignment metadata (one row per modality).

`sampleMap` `DataFrame` mapping samples across modalities.

`couplingResults` List of cached coupling analysis results.

References

Huber, W., et al. (2015). "Orchestrating high-throughput genomic analysis with Bioconductor." Nature Methods, 12(2), 115-121. doi:10.1038/nmeth.3252

See Also

`MultiPhysioExperiment()`, `experiments()`, `modalities()`, `alignSignals()`

`multitaperCoherence` *Multitaper coherence*

Description

Computes magnitude-squared coherence using multitaper spectral estimation with Discrete Prolate Spheroidal Sequences (DPSS / Slepian tapers). Multitaper estimation provides lower variance than Welch's method for short signals or when frequency resolution is critical.

Usage

```
multitaperCoherence(
  x,
  y = NULL,
  sr = NULL,
  nw = 4,
  k = NULL,
  freq_range = NULL,
  modality_x = NULL,
  modality_y = NULL,
  channels_x = 1L,
  channels_y = 1L,
  ...
)
```

Arguments

`x`	Numeric vector, PhysioExperiment, or MultiPhysioExperiment.
`y`	Numeric vector or PhysioExperiment, or NULL when `x` is an MPE.
`sr`	Numeric sampling rate in Hz (required when x/y are numeric).
`nw`	Numeric time-bandwidth product (default 4). Controls the trade-off between frequency resolution and variance.
`k`	Integer number of tapers (default `2 * nw - 1`).
`freq_range`	Optional numeric vector `c(low, high)` to restrict output frequencies.
`modality_x, modality_y`	Character modality names for MPE input.
`channels_x, channels_y`	Integer channel indices (default 1).
`...`	Currently unused.

Value

A list with components:

coherence Numeric vector of magnitude-squared coherence values in $[0, 1]$.

frequencies Numeric vector of corresponding frequencies in Hz.

confidence_limit Numeric scalar giving the 95\ threshold based on the number of tapers.

References

Thomson, D. J. (1982). Spectrum estimation and harmonic analysis. *Proceedings of the IEEE*, 70(9), 1055–1096.

Carter, G. C. (1987). Coherence and time delay estimation. *Proceedings of the IEEE*, 75(2), 236–255.

See Also

`coherence()`, `crossSpectrum()`, `couplingAnalysis()`

Examples

```
sr <- 500
t <- seq(0, 10, length.out = sr * 10)
x <- sin(2 * pi * 20 * t) + 0.2 * rnorm(length(t))
y <- 0.8 * sin(2 * pi * 20 * t) + 0.2 * rnorm(length(t))
result <- multitaperCoherence(x, y, sr = sr)
```

---

names,MultiPhysioExperiment-method

Names of a MultiPhysioExperiment

Description

Returns modality names (same as `modalities`).

Usage

```
## S4 method for signature 'MultiPhysioExperiment'
names(x)
```

Arguments

x	A `MultiPhysioExperiment` object.

Value

Character vector of modality names.

Examples

```
eeg_data <- matrix(rnorm(500 * 2), nrow = 500, ncol = 2)
emg_data <- matrix(rnorm(1000 * 2), nrow = 1000, ncol = 2)
pe_eeg <- PhysioExperiment(
  assays = list(raw = eeg_data),
  samplingRate = 250
)
pe_emg <- PhysioExperiment(
```

```
  assays = list(raw = emg_data),
  samplingRate = 1000
)
mpe <- MultiPhysioExperiment(EEG = pe_eeg, EMG = pe_emg)
names(mpe)  # c("EEG", "EMG")
```

`nModalities` *Get number of modalities*

Description

Get number of modalities

Usage

```
nModalities(x)

## S4 method for signature 'MultiPhysioExperiment'
nModalities(x)
```

Arguments

`x`	A `MultiPhysioExperiment` object.

Value

Integer scalar: the number of modalities stored in the object.

See Also

`modalities()`, `experiments()`

Examples

```
eeg_data <- matrix(rnorm(500 * 2), nrow = 500, ncol = 2)
pe_eeg <- PhysioExperiment(
  assays = list(raw = eeg_data),
  samplingRate = 250
)
mpe <- MultiPhysioExperiment(EEG = pe_eeg)
nModalities(mpe)
```

`phaseLagIndex` *Phase Lag Index (PLI)*

Description

Computes the Phase Lag Index between two signals. PLI measures the asymmetry of the distribution of phase differences and is insensitive to zero-lag (volume conduction) effects.

Usage

```
phaseLagIndex(
  x,
  y = NULL,
  sr = NULL,
  freq_band,
  modality_x = NULL,
  modality_y = NULL,
  channels_x = 1L,
  channels_y = 1L
)
```

Arguments

`x`	Numeric vector, PhysioExperiment, or MultiPhysioExperiment.
`y`	Numeric vector or PhysioExperiment (NULL when x is an MPE).
`sr`	Numeric sampling rate in Hz (required when x/y are numeric).
`freq_band`	Numeric vector of length 2 specifying the frequency band `c(low, high)` in Hz. Required.
`modality_x`	Character modality name in MPE for x signal.
`modality_y`	Character modality name in MPE for y signal.
`channels_x`	Integer channel index to extract from x (default 1).
`channels_y`	Integer channel index to extract from y (default 1).

Details

$$\text{PLI} = \left| \frac{1}{N} \sum_{t=1}^{N} \text{sign}(\sin(\phi_x(t) - \phi_y(t))) \right|$$

A PLI of 0 indicates either no coupling or symmetric (zero-lag) coupling. A PLI of 1 indicates perfect non-zero-lag phase locking.

Value

A named list with components:

pli Numeric scalar in [0, 1] – the Phase Lag Index.

phase_diff Numeric vector of instantaneous phase differences (radians).

References

Stam, C. J., Nolte, G., & Daffertshofer, A. (2007). Phase lag index: Assessment of functional connectivity from multi channel EEG and MEG with diminished bias from common sources. *Human Brain Mapping*, 28(11), 1178–1193.

See Also

`phaseLockingValue()`, `weightedPLI()`, `couplingAnalysis()`

Examples

```
sr <- 500
t <- seq(0, 2, length.out = sr * 2)
x <- sin(2 * pi * 10 * t)
y <- sin(2 * pi * 10 * t + pi / 3)
result <- phaseLagIndex(x, y, sr = sr, freq_band = c(8, 12))
result$pli
```

---

phaseLockingValue *Phase Locking Value (PLV)*

Description

Computes the Phase Locking Value between two signals, measuring the consistency of phase difference across time. A PLV of 1 indicates perfect phase locking; a PLV of 0 indicates no consistent phase relationship.

Usage

```
phaseLockingValue(
  x,
  y = NULL,
  sr = NULL,
  freq_band,
  method = c("hilbert", "wavelet"),
  modality_x = NULL,
  modality_y = NULL,
  channels_x = 1L,
  channels_y = 1L
)
```

Arguments

x	Numeric vector, PhysioExperiment, or MultiPhysioExperiment.
y	Numeric vector or PhysioExperiment (NULL when x is an MPE).
sr	Numeric sampling rate in Hz (required when x/y are numeric).
freq_band	Numeric vector of length 2 specifying the frequency band c(low, high) in Hz. Required.
method	Character; phase extraction method. Currently only "hilbert" is supported. "wavelet" is reserved for future use.
modality_x	Character modality name in MPE for x signal.
modality_y	Character modality name in MPE for y signal.
channels_x	Integer channel index to extract from x (default 1).
channels_y	Integer channel index to extract from y (default 1).

Details

The signals are bandpass-filtered to freq_band and then Hilbert- transformed to extract instantaneous phase. PLV is computed as:

$$\text{PLV} = \left| \frac{1}{N} \sum_{t=1}^{N} e^{i(\phi_x(t) - \phi_y(t))} \right|$$

Value

A named list with components:

plv Numeric scalar in [0, 1] – the Phase Locking Value.

phase_diff Numeric vector of instantaneous phase differences (radians, [-pi, pi]).

References

Lachaux, J.-P., Rodriguez, E., Martinerie, J., & Varela, F. J. (1999). Measuring phase synchrony in brain signals. *Human Brain Mapping*, 8(4), 194–208.

See Also

phaseLagIndex(), weightedPLI(), waveletPLV(), couplingAnalysis()

Examples

```
# Two phase-locked sinusoids
sr <- 500
t <- seq(0, 2, length.out = sr * 2)
x <- sin(2 * pi * 10 * t)
y <- sin(2 * pi * 10 * t + pi / 4)
result <- phaseLockingValue(x, y, sr = sr, freq_band = c(8, 12))
result$plv
```

`plotCoherenceSpectrum` *Plot coherence spectrum*

Description

Displays the coherence as a function of frequency from the result of `coherence`. Optionally draws the 95\ a horizontal dashed line.

Usage

```
plotCoherenceSpectrum(
  result,
  show_threshold = TRUE,
  title = "Coherence Spectrum",
  ...
)
```

Arguments

`result`	A list returned by `coherence`, containing at least `$coherence` and `$frequencies`. If `$confidence_limit` is present it is drawn as a threshold line.
`show_threshold`	Logical; if TRUE (default) and a confidence limit is available, draw it on the plot.
`title`	Character string for the plot title (default `"Coherence Spectrum"`).
`...`	Additional arguments (currently unused).

Value

A `ggplot` object.

References

Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. doi:10.1007/9783319242774

See Also

`coherence()`, `multitaperCoherence()`, `plotCouplingMatrix()`

Examples

```
sr <- 500
t <- seq(0, 10, length.out = sr * 10)
x <- sin(2 * pi * 20 * t) + 0.2 * rnorm(length(t))
y <- 0.8 * sin(2 * pi * 20 * t) + 0.2 * rnorm(length(t))
res <- coherence(x, y, sr = sr)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  plotCoherenceSpectrum(res)
}
```

plotCouplingMatrix *Plot a coupling matrix as a heatmap*

Description

Displays a matrix of coupling values (e.g., coherence between all channel pairs) as a colour-coded heatmap using `ggplot2::geom_tile()`.

Usage

```
plotCouplingMatrix(
  mat,
  title = "Coupling Matrix",
  low_colour = "white",
  high_colour = "#2166AC",
  ...
)
```

Arguments

`mat`	Numeric matrix of coupling values. Row and column names, if present, are used as axis labels.
`title`	Character string for the plot title (default `"Coupling Matrix"`).
`low_colour`	Character colour for the low end of the scale (default `"white"`).
`high_colour`	Character colour for the high end of the scale (default `"#2166AC"`).
`...`	Additional arguments passed to `ggplot2::theme()`.

Value

A `ggplot` object.

References

Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. doi:10.1007/9783319242774

See Also

`coherenceMatrix()`, `couplingMatrix()`, `surrogateMatrixTest()`

Examples

```
mat <- matrix(runif(16), nrow = 4,
              dimnames = list(paste0("Ch", 1:4), paste0("Ch", 1:4)))
if (requireNamespace("ggplot2", quietly = TRUE)) {
  plotCouplingMatrix(mat)
}
```

`plotCouplingTimecourse` *Plot time-varying coupling from sliding-window analysis*

Description

Displays the peak cross-correlation over time from the result of `slidingCrossCorrelation`.

Usage

```
plotCouplingTimecourse(result, title = "Coupling Time Course", ...)
```

Arguments

`result`	A list returned by `slidingCrossCorrelation`, containing at least `$times` and `$peak_correlations`.
`title`	Character string for the plot title (default `"Coupling Time Course"`).
`...`	Additional arguments (currently unused).

Value

A `ggplot` object.

References

Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. doi:10.1007/9783319242774

See Also

`slidingCrossCorrelation()`, `crossCorrelation()`, `plotCoherenceSpectrum()`

Examples

```
sr <- 500
set.seed(1)
x <- rnorm(5000)
y <- c(rep(0, 10), x[1:4990])
res <- slidingCrossCorrelation(x, y, sr = sr,
                               window_sec = 1, step_sec = 0.5)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  plotCouplingTimecourse(res)
}
```

plotWaveletCoherence *Plot wavelet coherence time-frequency map*

Description

Displays wavelet coherence (or wavelet PLV) as a filled time-frequency heatmap with optional Cone of Influence (COI) overlay.

Usage

```
plotWaveletCoherence(
  result,
  show_coi = TRUE,
  title = "Wavelet Coherence",
  fill_label = "Coherence",
  ...
)
```

Arguments

result	List returned by waveletCoherence or waveletPLV.
show_coi	Logical; overlay COI boundary (default TRUE if coi is present in the result).
title	Character plot title (default "Wavelet Coherence").
fill_label	Character legend label (default "Coherence").
...	Additional arguments passed to ggplot2::theme().

Value

A ggplot object.

References

Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. doi:10.1007/9783319242774

See Also

waveletCoherence(), waveletPLV(), plotCoherenceSpectrum()

Examples

```
sr <- 200
t <- seq(0, 2, length.out = sr * 2)
x <- sin(2 * pi * 10 * t) + 0.3 * rnorm(length(t))
y <- 0.8 * sin(2 * pi * 10 * t) + 0.3 * rnorm(length(t))
result <- waveletCoherence(x, y, sr = sr, frequencies = seq(5, 20))
if (requireNamespace("ggplot2", quietly = TRUE)) {
```

```
  plotWaveletCoherence(result)
}
```

`samplingRates` *Get sampling rates for all modalities*

Description

Returns a named numeric vector of sampling rates, one per modality.

Usage

```
samplingRates(x)

## S4 method for signature 'MultiPhysioExperiment'
samplingRates(x)
```

Arguments

`x`	A `MultiPhysioExperiment` object.

Value

Named numeric vector of sampling rates in Hz (e.g., `c(EEG = 500, EMG = 1000)`).

See Also

`modalities()`, `experiments()`, `alignToRate()`

Examples

```
eeg_data <- matrix(rnorm(500 * 2), nrow = 500, ncol = 2)
emg_data <- matrix(rnorm(1000 * 2), nrow = 1000, ncol = 2)
pe_eeg <- PhysioExperiment(
  assays = list(raw = eeg_data),
  samplingRate = 250
)
pe_emg <- PhysioExperiment(
  assays = list(raw = emg_data),
  samplingRate = 1000
)
mpe <- MultiPhysioExperiment(EEG = pe_eeg, EMG = pe_emg)
samplingRates(mpe)
```

show,MultiPhysioExperiment-method

Show method for MultiPhysioExperiment

Description

Displays a human-readable summary of the object.

Usage

```
## S4 method for signature 'MultiPhysioExperiment'
show(object)
```

Arguments

object	A MultiPhysioExperiment object.

Examples

```
eeg_data <- matrix(rnorm(500 * 2), nrow = 500, ncol = 2)
pe_eeg <- PhysioExperiment(
  assays = list(raw = eeg_data),
  samplingRate = 250
)
mpe <- MultiPhysioExperiment(EEG = pe_eeg)
mpe
```

slidingCrossCorrelation

Sliding-window cross-correlation

Description

Computes cross-correlation in sliding (overlapping) windows to track how time-domain coupling varies over time. For each window position, crossCorrelation is called and the results are assembled into a matrix.

Usage

```
slidingCrossCorrelation(
  x,
  y = NULL,
  sr = NULL,
  window_sec = 1,
  step_sec = 0.5,
  max_lag = NULL,
```

```
  modality_x = NULL,
  modality_y = NULL,
  channels_x = 1L,
  channels_y = 1L
)
```

Arguments

x	Numeric vector, PhysioExperiment, or MultiPhysioExperiment.
y	Numeric vector or PhysioExperiment, or NULL when x is a MultiPhysioExperiment.
sr	Numeric sampling rate in Hz (required when x/y are numeric).
window_sec	Numeric window length in seconds (default 1).
step_sec	Numeric step size in seconds (default 0.5).
max_lag	Integer maximum lag in samples for each window. If NULL, defaults to `floor(window_samples / 4)`.
modality_x	Character modality name in MPE for the x signal.
modality_y	Character modality name in MPE for the y signal.
channels_x	Integer which channel to extract from x (default 1).
channels_y	Integer which channel to extract from y (default 1).

Value

A named list with components:

correlations Numeric matrix of dimensions (n_windows x n_lags) containing cross-correlation values.

times Numeric vector of window centre times in seconds.

lags Integer vector of lag values in samples.

peak_lags Numeric vector of peak lags (in samples) for each window.

peak_correlations Numeric vector of peak correlation values for each window.

References

Chatfield, C. (2004). *The Analysis of Time Series: An Introduction* (6th ed.). Chapman & Hall/CRC.

See Also

`crossCorrelation()`, `plotCouplingTimecourse()`, `couplingAnalysis()`

Examples

```
sr <- 500
set.seed(1)
x <- rnorm(5000)
y <- c(rep(0, 10), x[1:4990])
result <- slidingCrossCorrelation(x, y, sr = sr,
                                  window_sec = 1, step_sec = 0.5)
dim(result$correlations)
result$peak_lags
```

`surrogateMatrixTest` *Surrogate-based significance test for coupling matrices*

Description

Tests each element of a coupling matrix for significance using surrogate testing, with correction for multiple comparisons (FDR or Bonferroni).

Usage

```
surrogateMatrixTest(
  x,
  y = NULL,
  method,
  n_surrogates = 199L,
  surrogate_type = c("phase", "timeshift"),
  correction = c("fdr", "bonferroni", "none"),
  alpha = 0.05,
  channels_x = NULL,
  channels_y = NULL,
  modality_x = NULL,
  modality_y = NULL,
  cores = 1L,
  ...
)
```

Arguments

`x`	PhysioExperiment or MultiPhysioExperiment.
`y`	PhysioExperiment or NULL (when `x` is an MPE).
`method`	Character coupling method (same options as `couplingAnalysis`).
`n_surrogates`	Integer number of surrogates (default 199).
`surrogate_type`	Character surrogate generation method: `"phase"` (default) or `"timeshift"`.
`correction`	Character correction method: `"fdr"` (default), `"bonferroni"`, or `"none"`.
`alpha`	Numeric significance level (default 0.05).

`channels_x`, `channels_y`
: Integer vectors of channel indices, or NULL for all channels.

`modality_x`, `modality_y`
: Character modality names for MPE input.

`cores`
: Integer number of parallel cores (default 1).

`...`
: Additional arguments passed to the coupling function.

Value

A list with components:

matrix Coupling matrix (observed values).

p_values Matrix of raw p-values (same dimensions).

p_adjusted Matrix of corrected p-values.

significant Logical matrix indicating significance.

correction Character correction method used.

alpha Numeric significance level.

References

Theiler, J., Eubank, S., Longtin, A., Galdrikian, B., & Farmer, J. D. (1992). Testing for nonlinearity in time series: the method of surrogate data. *Physica D: Nonlinear Phenomena*, 58(1–4), 77–94.

Benjamini, Y., & Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(1), 289–300.

See Also

`surrogateTest()`, `couplingMatrix()`, `coherenceMatrix()`

`surrogateTest` *Surrogate-based significance test for coupling*

Description

Tests whether the observed coupling between two signals is statistically significant by comparing it against a null distribution generated from surrogate signals. Surrogates are created by randomising the phases (or circularly shifting) one signal, thereby destroying the cross-signal relationship while preserving the autocorrelation structure.

Usage

```
surrogateTest(
  x,
  y = NULL,
  sr = NULL,
  method,
  n_surrogates = 199L,
  surrogate_type = c("phase", "timeshift"),
  modality_x = NULL,
  modality_y = NULL,
  channels_x = 1L,
  channels_y = 1L,
  cores = 1L,
  ...
)
```

Arguments

x	Numeric vector, PhysioExperiment, or MultiPhysioExperiment.
y	Numeric vector or PhysioExperiment, or NULL when x is an MPE.
sr	Numeric sampling rate in Hz (required when x/y are numeric).
method	Character coupling method, one of "coherence", "plv", "pli", "wpli", "granger", "crosscorrelation", "wavelet_coherence", "wavelet_plv".
n_surrogates	Integer number of surrogates (default 199).
surrogate_type	Character surrogate generation method: "phase" (default) or "timeshift".
modality_x, modality_y	Character modality names for MPE input.
channels_x, channels_y	Integer channel indices (default 1).
cores	Integer number of parallel cores to use (default 1L). When cores > 1, surrogate computations are distributed across cores using parallel::mclapply() (Unix) or parallel::parLapply() (Windows). Falls back to sequential computation on error.
...	Additional arguments passed to the coupling function (e.g. freq_band, nperseg, order).

Details

The p-value is computed as (sum(surr >= obs) + 1) / (n_surr + 1), following the conservative correction of Phipson & Smyth (2010).

Value

A list with components:

observed The full coupling result from the original signals.

statistic Numeric scalar: the extracted coupling statistic.

p_value Numeric scalar: surrogate p-value.

surrogate_distribution Numeric vector of surrogate statistics.

threshold_95 Numeric scalar: 95th percentile of surrogates.

References

Theiler, J., Eubank, S., Longtin, A., Galdrikian, B., & Farmer, J. D. (1992). Testing for nonlinearity in time series: the method of surrogate data. *Physica D: Nonlinear Phenomena*, 58(1–4), 77–94.

Phipson, B., & Smyth, G. K. (2010). Permutation P-values should never be zero: calculating exact P-values when permutations are randomly drawn. *Statistical Applications in Genetics and Molecular Biology*, 9(1), Article 39.

See Also

`bootstrapCI()`, `surrogateMatrixTest()`, `couplingAnalysis()`

Examples

```
sr <- 500
t <- seq(0, 2, length.out = sr * 2)
x <- sin(2 * pi * 10 * t) + 0.3 * rnorm(length(t))
y <- 0.8 * sin(2 * pi * 10 * t) + 0.3 * rnorm(length(t))
result <- surrogateTest(x, y, sr = sr, method = "coherence",
                        n_surrogates = 19, nperseg = 128L)
result$p_value
```

---

waveletCoherence *Time-frequency wavelet coherence*

Description

Computes wavelet coherence between two signals using complex Morlet wavelets. The cross-wavelet spectrum and auto-spectra are smoothed with a Gaussian temporal window (width proportional to `smoothing_cycles / frequency`), and coherence is computed as:

Usage

```
waveletCoherence(
  x,
  y = NULL,
  sr = NULL,
  frequencies = seq(1, 40, by = 1),
  n_cycles = 7,
  smoothing_cycles = 3,
  modality_x = NULL,
  modality_y = NULL,
```

```
  channels_x = 1L,
  channels_y = 1L,
  ...
)
```

Arguments

`x`	Numeric vector, PhysioExperiment, or MultiPhysioExperiment.
`y`	Numeric vector or PhysioExperiment, or NULL when `x` is an MPE.
`sr`	Numeric sampling rate in Hz (required when x/y are numeric).
`frequencies`	Numeric vector of centre frequencies in Hz (default `seq(1, 40, by = 1)`).
`n_cycles`	Numeric number of wavelet cycles (default 7).
`smoothing_cycles`	Numeric number of cycles for the temporal smoothing Gaussian (default 3).
`modality_x`, `modality_y`	Character modality names for MPE input.
`channels_x`, `channels_y`	Integer channel indices (default 1).
`...`	Currently unused.

Details

$$C(t,f) = \frac{|\langle W_{xy}(t,f)\rangle|^2}{\langle |W_x(t,f)|^2\rangle \cdot \langle |W_y(t,f)|^2\rangle}$$

Value

A list with components:

coherence Numeric matrix `[time x frequency]` of coherence values in $[0, 1]$.

phase Numeric matrix `[time x frequency]` of phase differences (radians).

frequencies Numeric vector of centre frequencies.

times Numeric vector of time points (seconds from start).

coi Numeric vector of Cone of Influence frequencies. At each time point, frequencies below this value are affected by edge artifacts.

References

Torrence, C., & Compo, G. P. (1998). A practical guide to wavelet analysis. *Bulletin of the American Meteorological Society*, 79(1), 61–78.

Grinsted, A., Moore, J. C., & Jevrejeva, S. (2004). Application of the cross wavelet transform and wavelet coherence to geophysical time series. *Nonlinear Processes in Geophysics*, 11(5/6), 561–566.

See Also

waveletPLV(), coherence(), plotWaveletCoherence(), couplingAnalysis()

Examples

```
sr <- 200
t <- seq(0, 2, length.out = sr * 2)
x <- sin(2 * pi * 10 * t) + 0.3 * rnorm(length(t))
y <- 0.8 * sin(2 * pi * 10 * t) + 0.3 * rnorm(length(t))
result <- waveletCoherence(x, y, sr = sr, frequencies = seq(5, 20))
```

---

waveletPLV *Time-frequency wavelet PLV*

Description

Computes Phase Locking Value in the time-frequency domain using complex Morlet wavelets. The phase difference between the two signals is computed at each time-frequency point, and PLV is the magnitude of the smoothed unit-phase vector:

Usage

```
waveletPLV(
  x,
  y = NULL,
  sr = NULL,
  frequencies = seq(1, 40, by = 1),
  n_cycles = 7,
  smoothing_cycles = 3,
  modality_x = NULL,
  modality_y = NULL,
  channels_x = 1L,
  channels_y = 1L,
  ...
)
```

Arguments

x	Numeric vector, PhysioExperiment, or MultiPhysioExperiment.
y	Numeric vector or PhysioExperiment, or NULL when x is an MPE.
sr	Numeric sampling rate in Hz (required when x/y are numeric).
frequencies	Numeric vector of centre frequencies in Hz (default seq(1, 40, by = 1)).
n_cycles	Numeric number of wavelet cycles (default 7).
smoothing_cycles	Numeric number of cycles for the temporal smoothing Gaussian (default 3).

`modality_x`, `modality_y`
: Character modality names for MPE input.

`channels_x`, `channels_y`
: Integer channel indices (default 1).

`...`
: Currently unused.

Details

$$\mathrm{PLV}(t, f) = \left| \langle e^{i\Delta\phi(t,f)} \rangle \right|$$

Value

A list with components:

plv Numeric matrix `[time x frequency]` of PLV values in $[0, 1]$.

frequencies Numeric vector of centre frequencies.

times Numeric vector of time points (seconds from start).

coi Numeric vector of Cone of Influence frequencies. At each time point, frequencies below this value are affected by edge artifacts.

References

Torrence, C., & Compo, G. P. (1998). A practical guide to wavelet analysis. *Bulletin of the American Meteorological Society*, 79(1), 61–78.

Lachaux, J.-P., Rodriguez, E., Martinerie, J., & Varela, F. J. (1999). Measuring phase synchrony in brain signals. *Human Brain Mapping*, 8(4), 194–208.

See Also

`waveletCoherence()`, `phaseLockingValue()`, `plotWaveletCoherence()`, `couplingAnalysis()`

Examples

```
sr <- 200
t <- seq(0, 2, length.out = sr * 2)
x <- sin(2 * pi * 10 * t) + 0.3 * rnorm(length(t))
y <- sin(2 * pi * 10 * t + pi/4) + 0.3 * rnorm(length(t))
result <- waveletPLV(x, y, sr = sr, frequencies = seq(5, 20))
```

`weightedPLI` *Weighted Phase Lag Index (wPLI)*

Description

Computes the weighted Phase Lag Index between two signals. wPLI weights each phase-difference sample by the magnitude of the imaginary component of the cross-spectrum, reducing the influence of noise sources that produce phase differences near 0 or pi.

Usage

```
weightedPLI(
  x,
  y = NULL,
  sr = NULL,
  freq_band,
  debiased = TRUE,
  modality_x = NULL,
  modality_y = NULL,
  channels_x = 1L,
  channels_y = 1L
)
```

Arguments

`x`	Numeric vector, PhysioExperiment, or MultiPhysioExperiment.
`y`	Numeric vector or PhysioExperiment (NULL when `x` is an MPE).
`sr`	Numeric sampling rate in Hz (required when x/y are numeric).
`freq_band`	Numeric vector of length 2 specifying the frequency band `c(low, high)` in Hz. Required.
`debiased`	Logical; if TRUE (default), also compute the debiased wPLI estimator.
`modality_x`	Character modality name in MPE for x signal.
`modality_y`	Character modality name in MPE for y signal.
`channels_x`	Integer channel index to extract from x (default 1).
`channels_y`	Integer channel index to extract from y (default 1).

Details

$$\text{wPLI} = \frac{|\sum |\text{Im}(S_{xy})| \cdot \text{sign}(\text{Im}(S_{xy}))|}{\sum |\text{Im}(S_{xy})|}$$

When `debiased = TRUE`, the debiased estimator from Vinck et al. (2011) is also returned:

$$\text{wPLI}_{\text{debiased}} = \frac{(\sum \text{Im}(S_{xy}))^2 - \sum \text{Im}(S_{xy})^2}{(\sum |\text{Im}(S_{xy})|)^2 - \sum \text{Im}(S_{xy})^2}$$

Value

A named list with components:

wpli Numeric scalar in [0, 1] – the weighted PLI.

wpli_debiased Numeric scalar – the debiased wPLI, or NULL if `debiased = FALSE`.

n_samples Integer – number of time-domain samples used.

References

Vinck, M., Oostenveld, R., van Wingerden, M., Battaglia, F., & Pennartz, C. M. A. (2011). An improved index of phase-synchronization for electrophysiological data in the presence of volume-conduction, noise and sample-size bias. *NeuroImage*, 55(4), 1548–1565.

Lachaux, J.-P., Rodriguez, E., Martinerie, J., & Varela, F. J. (1999). Measuring phase synchrony in brain signals. *Human Brain Mapping*, 8(4), 194–208.

See Also

`phaseLockingValue()`, `phaseLagIndex()`, `couplingAnalysis()`

Examples

```
sr <- 500
t <- seq(0, 2, length.out = sr * 2)
x <- sin(2 * pi * 10 * t)
y <- sin(2 * pi * 10 * t + pi / 4)
result <- weightedPLI(x, y, sr = sr, freq_band = c(8, 12))
result$wpli
```

Index