Package: PhysioAnalysis (via r-universe)

May 16, 2026

Title Analysis and Visualization for PhysioExperiment Objects

Version 0.2.0

Author Yusuke Matsui

Maintainer Yusuke Matsui <you@example.com>

Description Provides analysis and visualization functions for physiological signal data including FFT, time-frequency analysis, epoching, connectivity analysis, statistical testing, and various plotting functions including topoplots and multichannel displays.

Depends R (>= 4.2), PhysioCore

Imports methods, SummarizedExperiment, S4Vectors, signal, stats, graphics, grDevices

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

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

Collate 'ops-fft.R' 'ops-timefreq.R' 'ops-epoch.R' 'ops-connectivity.R' 'stats-tests.R' 'vis-plot.R' 'vis-multichannel.R' 'vis-topomap.R' 'zzz.R'

VignetteBuilder knitr

RoxygenNote 7.3.3

Config/pak/sysreqs zlib1g-dev

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

Date/Publication 2026-03-16 11:31:08 UTC

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

RemoteRef HEAD

RemoteSha 671b600d26c20ba570261995c1b2beff372c2ac7

Contents

`.onLoad` *Package on-load hook*

Description

Package on-load hook

Usage

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

Arguments

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

`anovaEpochs` *ANOVA across conditions*

Description

Performs one-way ANOVA at each time point and channel across multiple conditions.

Usage

```
anovaEpochs(x, groups)
```

Arguments

`x`	An epoched PhysioExperiment object (4D data).
`groups`	Factor or character vector indicating group membership for each epoch. Can also be a column name from epoch_info metadata.

Value

A list containing:

`f_values`	Matrix of F-statistics (time x channel)
`p_values`	Matrix of p-values (time x channel)
`df_between`	Between-group degrees of freedom
`df_within`	Within-group degrees of freedom
`group_means`	Array of group means (time x channel x group)

References

Maris, E. & Oostenveld, R. (2007). "Nonparametric statistical testing of EEG- and MEG-data." *Journal of Neuroscience Methods*, 164(1), 177-190. doi:10.1016/j.jneumeth.2007.03.024

See Also

`tTestEpochs()` for pairwise comparisons, `clusterPermutationTest()` for multiple comparison correction, `effectSize()` for Cohen's d effect size.

Examples

```
# Create example epoched data with conditions
set.seed(123)
epochs <- array(rnorm(100 * 4 * 30 * 1), dim = c(100, 4, 30, 1))
pe <- PhysioExperiment(
  assays = list(epoched = epochs),
  samplingRate = 100,
  metadata = list(
    epoch_info = S4Vectors::DataFrame(
      condition = rep(c("A", "B", "C"), each = 10)
    )
  )
)
# ANOVA across conditions
result <- anovaEpochs(pe, groups = "condition")
```

---

averageEpochs *Average epochs*

Description

Computes the average across epochs, optionally by condition.

Usage

```
averageEpochs(x, by = NULL)
```

Arguments

`x`	An epoched PhysioExperiment object.
`by`	Optional column name in epoch_info to group by.

Value

A `PhysioExperiment` object with an `"averaged"` assay containing the mean signal across epochs. If by is specified, the third dimension corresponds to conditions.

References

Luck, S.J. (2014). "An Introduction to the Event-Related Potential Technique." 2nd ed. MIT Press.

See Also

`epochData()` to create epoched data, `grandAverage()` for averaging across subjects, `plotERP()` for ERP visualization.

bandPower	*Compute band power*

Description

Extracts power in specified frequency bands.

Usage

```
bandPower(x, bands = NULL, method = c("welch", "wavelet"), relative = FALSE)
```

Arguments

`x`	A PhysioExperiment object.
`bands`	Named list of frequency bands. Each element should be c(low, high). Default includes standard EEG bands.
`method`	Method: "welch" (PSD) or "wavelet".
`relative`	If TRUE, returns relative power (proportion of total).

Value

A `data.frame` with one row per channel and columns for the channel name and power in each frequency band. If `relative = TRUE`, values represent the proportion of total power.

References

Oppenheim, A.V. & Willsky, A.S. (1997). "Signals and Systems." 2nd ed. Prentice Hall.

See Also

`spectrogram()` for full time-frequency decomposition, `waveletTransform()` for wavelet-based power, `fftSignals()` for raw FFT, `plotPSD()` for power spectral density visualization.

Examples

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(2560), nrow = 256, ncol = 10)),
  colData = S4Vectors::DataFrame(label = paste0("Ch", 1:10)),
  samplingRate = 256
)

# Compute band power for standard EEG bands
bp <- bandPower(pe)
head(bp)

# Compute relative band power
bp_rel <- bandPower(pe, relative = TRUE)
```

`bootstrapCI` *Bootstrap confidence interval for ERP*

Description

Computes bootstrap confidence intervals for averaged epochs.

Usage

```
bootstrapCI(
  x,
  n_bootstrap = 1000L,
  ci_level = 0.95,
  condition = NULL,
  seed = NULL
)
```

Arguments

`x`	An epoched PhysioExperiment object (4D data).
`n_bootstrap`	Number of bootstrap iterations (default: 1000).
`ci_level`	Confidence interval level (default: 0.95).
`condition`	Epoch indices to include. If NULL, uses all epochs.
`seed`	Random seed for reproducibility.

Value

A list containing:

`mean`	Mean across epochs (time x channel)
`ci_lower`	Lower CI bound (time x channel)
`ci_upper`	Upper CI bound (time x channel)
`se`	Standard error (time x channel)

References

Maris, E. & Oostenveld, R. (2007). "Nonparametric statistical testing of EEG- and MEG-data." *Journal of Neuroscience Methods*, 164(1), 177-190. doi:10.1016/j.jneumeth.2007.03.024

See Also

`effectSize()` for Cohen's d effect size, `tTestEpochs()` for parametric significance testing, `plotERP()` for ERP visualization.

Examples

```
# Create example epoched data
set.seed(123)
epochs <- array(rnorm(100 * 4 * 20 * 1), dim = c(100, 4, 20, 1))
pe <- PhysioExperiment(
  assays = list(epoched = epochs),
  samplingRate = 100
)
# Bootstrap CI
result <- bootstrapCI(pe, n_bootstrap = 500)
```

`clusterPermutationTest` *Cluster-based permutation test*

Description

Performs cluster-based permutation testing for multiple comparison correction. Identifies clusters of significant effects and computes cluster-level p-values.

Usage

```
clusterPermutationTest(
  x,
  condition1 = NULL,
  condition2 = NULL,
  n_permutations = 1000L,
  cluster_threshold = 0.05,
  tail = 0L,
  seed = NULL
)
```

Arguments

`x`	An epoched PhysioExperiment object (4D data).
`condition1`	Indices for first condition.
`condition2`	Indices for second condition. If NULL, tests against zero.
`n_permutations`	Number of permutations (default: 1000).
`cluster_threshold`	Initial threshold for cluster formation (p-value).
`tail`	Test type: 0 = two-tailed, 1 = right tail, -1 = left tail.
`seed`	Random seed for reproducibility.

Value

A list containing:

clusters	List of identified clusters with their statistics
cluster_p	P-values for each cluster
t_obs	Observed t-values matrix
cluster_mask	Logical matrix indicating cluster membership

References

Maris, E. & Oostenveld, R. (2007). "Nonparametric statistical testing of EEG- and MEG-data." *Journal of Neuroscience Methods*, 164(1), 177-190. doi:10.1016/j.jneumeth.2007.03.024

See Also

`tTestEpochs()` for pointwise t-tests, `correctPValues()` for traditional multiple comparison correction, `findSignificantWindows()` for identifying significant time windows.

Examples

```
# Create example epoched data
set.seed(123)
epochs <- array(rnorm(50 * 4 * 20 * 1), dim = c(50, 4, 20, 1))
# Add effect to condition 2
epochs[20:30, 1:2, 11:20, 1] <- epochs[20:30, 1:2, 11:20, 1] + 1
pe <- PhysioExperiment(
  assays = list(epoched = epochs),
  samplingRate = 100
)
# Cluster permutation test
result <- clusterPermutationTest(pe, 1:10, 11:20, n_permutations = 100)
```

---

coherence *Connectivity Analysis for PhysioExperiment*

Description

Functions for computing functional connectivity between channels including coherence, phase synchrony measures, and correlation-based metrics. Compute coherence between channels

Usage

```
coherence(
  x,
  channels = NULL,
  freq_range = NULL,
  nperseg = 256L,
```

```
  noverlap = NULL,
  assay_name = NULL
)
```

Arguments

`x`	A PhysioExperiment object.
`channels`	Integer vector of channel indices to analyze. If NULL, uses all.
`freq_range`	Numeric vector of length 2 specifying frequency range (Hz).
`nperseg`	Number of samples per segment for Welch's method. Default is 256.
`noverlap`	Number of overlapping samples. Default is nperseg/2.
`assay_name`	Input assay name. If NULL, uses default assay.

Details

Calculates the magnitude-squared coherence between pairs of channels, which measures the linear correlation between signals as a function of frequency.

Coherence is computed using Welch's averaged periodogram method. Values range from 0 (no linear relationship) to 1 (perfect linear relationship).

Value

A list with components:

`coherence`	3D array (freq x channel x channel) of coherence values
`frequencies`	Frequency vector
`channel_names`	Channel names

References

Nolte, G., et al. (2004). "Identifying true brain interaction from EEG data using the imaginary part of coherency." Clinical Neurophysiology, 115(10), 2292-2307. doi:10.1016/j.clinph.2004.04.029

See Also

`crossSpectrum()` for the underlying cross-spectral density, `plv()` for phase-based connectivity, `connectivityMatrix()` for a unified connectivity interface.

Examples

```
# Create example with 4 channels
set.seed(123)
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(4000), nrow = 1000, ncol = 4)),
  colData = S4Vectors::DataFrame(label = c("Fz", "Cz", "Pz", "Oz")),
  samplingRate = 256
)
```

```
# Compute coherence
coh <- coherence(pe, freq_range = c(1, 50))
dim(coh$coherence)  # frequencies x channels x channels
```

`connectivityMatrix` *Compute connectivity matrix for a frequency band*

Description

High-level function to compute connectivity using various metrics.

Usage

```
connectivityMatrix(
  x,
  method = c("coherence", "plv", "pli", "wpli", "correlation"),
  freq_band = NULL,
  channels = NULL,
  assay_name = NULL
)
```

Arguments

`x`	A PhysioExperiment object.
`method`	Connectivity method: "coherence", "plv", "pli", "wpli", "correlation".
`freq_band`	Frequency band for phase-based methods.
`channels`	Integer vector of channel indices.
`assay_name`	Input assay name.

Value

A numeric matrix (channel x channel) of connectivity values. For `"coherence"`, returns the mean coherence across frequencies. For phase-based methods (`"plv"`, `"pli"`, `"wpli"`), returns values between 0 and 1. For `"correlation"`, returns values between -1 and 1.

References

Lachaux, J.-P., et al. (1999). "Measuring phase synchrony in brain signals." Human Brain Mapping, 8(4), 194-208. doi:10.1002/(SICI)1097-0193(1999)8:4<194::AID-HBM4>3.0.CO;2-C

See Also

`coherence()`, `plv()`, `pli()`, `wPLI()`, `correlationMatrix()` for individual connectivity methods.

Examples

```
set.seed(123)
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(4000), nrow = 1000, ncol = 4)),
  samplingRate = 256
)

# Compute PLV connectivity in alpha band
conn <- connectivityMatrix(pe, method = "plv", freq_band = c(8, 12))
```

`correctPValues` *Multiple comparison correction*

Description

Applies multiple comparison correction to p-values.

Usage

```
correctPValues(p_values, method = c("fdr", "bonferroni", "holm", "bh", "none"))
```

Arguments

`p_values`	Matrix or vector of p-values.
`method`	Correction method: "bonferroni", "holm", "fdr" (Benjamini-Hochberg), "bh" (alias for fdr), or "none".

Value

Corrected p-values in the same format as input.

References

Maris, E. & Oostenveld, R. (2007). "Nonparametric statistical testing of EEG- and MEG-data." *Journal of Neuroscience Methods*, 164(1), 177-190. doi:10.1016/j.jneumeth.2007.03.024

See Also

`clusterPermutationTest()` for cluster-based correction, `tTestEpochs()` for pointwise t-tests, `findSignificantWindows()` for identifying significant time windows.

Examples

```
# Example p-values
p <- matrix(runif(100), nrow = 10)
# Apply FDR correction
p_corrected <- correctPValues(p, method = "fdr")
```

correlationMatrix — *Compute correlation matrix between channels*

Description

Calculates the Pearson correlation coefficient between all channel pairs.

Usage

```
correlationMatrix(
  x,
  channels = NULL,
  method = c("pearson", "spearman", "kendall"),
  assay_name = NULL
)
```

Arguments

`x`	A PhysioExperiment object.
`channels`	Integer vector of channel indices.
`method`	Correlation method: "pearson", "spearman", or "kendall".
`assay_name`	Input assay name.

Value

A numeric correlation matrix (channel x channel) with values ranging from -1 to 1. Row and column names are set to channel names when available.

References

Oppenheim, A.V. & Willsky, A.S. (1997). "Signals and Systems." 2nd ed. Prentice Hall.

See Also

`coherence()` for frequency-domain coherence, `plv()` for phase-based connectivity, `connectivityMatrix()` for a unified connectivity interface.

Examples

```
set.seed(123)
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(4000), nrow = 1000, ncol = 4)),
  colData = S4Vectors::DataFrame(label = c("Fz", "Cz", "Pz", "Oz")),
  samplingRate = 256
)

# Compute Pearson correlation
cor_matrix <- correlationMatrix(pe)
```

crossSpectrum *Compute cross-spectral density*

Description

Calculates the cross-spectral density between channel pairs.

Usage

```
crossSpectrum(
  x,
  channels = NULL,
  nperseg = 256L,
  noverlap = NULL,
  assay_name = NULL
)
```

Arguments

`x`	A PhysioExperiment object.
`channels`	Integer vector of channel indices. If NULL, uses all.
`nperseg`	Number of samples per segment. Default is 256.
`noverlap`	Number of overlapping samples.
`assay_name`	Input assay name.

Value

A list with the following components:

`csd`	3D complex array of cross-spectral density values (frequency x channel x channel)
`frequencies`	Numeric vector of frequencies in Hz
`channel_names`	Character vector of channel names

References

Oppenheim, A.V. & Willsky, A.S. (1997). "Signals and Systems." 2nd ed. Prentice Hall.

See Also

`coherence()` for magnitude-squared coherence, `connectivityMatrix()` for a unified connectivity interface, `spectrogram()` for single-channel spectral analysis.

Examples

```
set.seed(123)
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(2000), nrow = 500, ncol = 4)),
  samplingRate = 256
)

# Compute cross-spectral density
csd <- crossSpectrum(pe)
```

---

effectSize — *Compute effect size (Cohen's d)*

Description

Calculates Cohen's d effect size at each time point and channel.

Usage

```
effectSize(x, condition1 = NULL, condition2 = NULL, pooled = TRUE)
```

Arguments

x	An epoched PhysioExperiment object (4D data).
condition1	Indices for first condition.
condition2	Indices for second condition. If NULL, computes d against zero.
pooled	If TRUE (default for two-sample), uses pooled standard deviation.

Value

A list containing:

d	Matrix of Cohen's d values (time x channel)
ci_lower	Lower 95% CI for d
ci_upper	Upper 95% CI for d

References

Maris, E. & Oostenveld, R. (2007). "Nonparametric statistical testing of EEG- and MEG-data." *Journal of Neuroscience Methods*, 164(1), 177-190. doi:10.1016/j.jneumeth.2007.03.024

See Also

tTestEpochs() for significance testing, bootstrapCI() for bootstrap confidence intervals, clusterPermutationTest() for cluster-based permutation testing.

Examples

```
# Create example epoched data
set.seed(123)
epochs <- array(rnorm(100 * 4 * 20 * 1), dim = c(100, 4, 20, 1))
pe <- PhysioExperiment(
  assays = list(epoched = epochs),
  samplingRate = 100
)
# Effect size for one-sample
result <- effectSize(pe, condition1 = 1:10)
# Effect size between conditions
result2 <- effectSize(pe, condition1 = 1:10, condition2 = 11:20)
```

---

epochData *Epoching operations for PhysioExperiment*

Description

Functions for segmenting continuous data into epochs/trials based on events. Epoch data around events

Usage

```
epochData(
  x,
  tmin = -0.2,
  tmax = 0.8,
  event_type = NULL,
  baseline = NULL,
  reject = NULL
)
```

Arguments

`x`	A PhysioExperiment object.
`tmin`	Time before event onset in seconds (negative for pre-stimulus).
`tmax`	Time after event onset in seconds.
`event_type`	Character vector of event types to epoch around. If NULL, uses all events.
`baseline`	Numeric vector of length 2 specifying baseline period (tmin, tmax) for baseline correction. NULL for no correction.
`reject`	Amplitude threshold for epoch rejection. NULL to keep all.

Details

Extracts epochs (segments) of data around specified events.

Value

A new PhysioExperiment object with a 4D assay (time x channel x epoch x sample) named "epoched". Metadata includes epoch_tmin, epoch_tmax, epoch_info (a DataFrame with epoch_id, event_type, event_value, event_onset), and n_epochs.

References

Luck, S.J. (2014). "An Introduction to the Event-Related Potential Technique." 2nd ed. MIT Press.

See Also

averageEpochs() to average across epochs, epochTimes() to get the time vector, tTestEpochs() for statistical testing on epoched data, plotERP() for ERP visualization.

Examples

```
# Create continuous data with events
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(1000 * 4), nrow = 1000)),
  samplingRate = 100
)
pe <- addEvents(pe, onset = c(1, 2, 3, 4, 5), type = "stimulus")

# Extract epochs: 200ms before to 800ms after stimulus
epochs <- epochData(pe, tmin = -0.2, tmax = 0.8)

# With baseline correction
epochs_bl <- epochData(pe, tmin = -0.2, tmax = 0.8,
                       baseline = c(-0.2, 0))

# With artifact rejection
epochs_clean <- epochData(pe, tmin = -0.2, tmax = 0.8,
                          reject = 100)
```

---

epochTimes *Get epoch time vector*

Description

Returns the time vector for epoched data relative to event onset.

Usage

```
epochTimes(x)
```

Arguments

x	An epoched PhysioExperiment object.

Value

Numeric vector of times in seconds relative to event onset (negative values represent pre-stimulus time).

References

Luck, S.J. (2014). "An Introduction to the Event-Related Potential Technique." 2nd ed. MIT Press.

See Also

`epochData()` which creates the epoched data, `averageEpochs()` for epoch averaging, `plotERP()` for ERP visualization.

`fftSignals` *Fast Fourier transform helper*

Description

Computes the discrete Fourier transform along the time axis of the default assay and stores the magnitude spectrum in a new assay named `"fft"`.

Usage

```
fftSignals(x)
```

Arguments

x	A `PhysioExperiment` object.

Value

A `PhysioExperiment` object with an additional `"fft"` assay containing the magnitude spectrum (same dimensions as the input assay).

References

Oppenheim, A.V. & Willsky, A.S. (1997). "Signals and Systems." 2nd ed. Prentice Hall.

See Also

`spectrogram()` for time-frequency analysis, `bandPower()` for frequency band power extraction, `plotPSD()` for power spectral density visualization.

findSignificantWindows *Find significant time windows*

Description

Identifies contiguous time periods with significant effects.

Usage

```
findSignificantWindows(p_values, times = NULL, alpha = 0.05, min_duration = 0)
```

Arguments

p_values	Vector of p-values across time.
times	Vector of time points.
alpha	Significance threshold (default: 0.05).
min_duration	Minimum duration of significant window in time units.

Value

A data.frame with columns: start, end, duration, min_p.

References

Maris, E. & Oostenveld, R. (2007). "Nonparametric statistical testing of EEG- and MEG-data." *Journal of Neuroscience Methods*, 164(1), 177-190. doi:10.1016/j.jneumeth.2007.03.024

See Also

correctPValues() for multiple comparison correction, clusterPermutationTest() for cluster-based permutation testing, tTestEpochs() for pointwise t-tests.

Examples

```
# Example p-values
times <- seq(-0.2, 0.8, length.out = 100)
p <- c(rep(0.5, 30), rep(0.01, 20), rep(0.5, 50))
windows <- findSignificantWindows(p, times)
```

grandAverage — *Grand average across subjects/samples*

Description

Computes grand average across multiple PhysioExperiment objects.

Usage

```
grandAverage(...)
```

Arguments

`...`	PhysioExperiment objects or a list of them.

Value

A `PhysioExperiment` object with a `"grand_average"` assay containing the mean across all input objects. Metadata includes `n_subjects` indicating the number of objects averaged.

References

Luck, S.J. (2014). "An Introduction to the Event-Related Potential Technique." 2nd ed. MIT Press.

See Also

`averageEpochs()` for within-subject averaging, `epochData()` to create epoched data, `plotERP()` for ERP visualization.

hilbertTransform — *Hilbert transform for instantaneous amplitude/phase*

Description

Computes the analytic signal using the Hilbert transform. The analytic signal can be used to extract instantaneous amplitude and phase.

Usage

```
hilbertTransform(x, output_assay = "analytic")
```

Arguments

`x`	A PhysioExperiment object.
`output_assay`	Name for the output assay.

Value

A `PhysioExperiment` object with an additional assay (default `"analytic"`) containing the complex-valued analytic signal.

References

Oppenheim, A.V. & Willsky, A.S. (1997). "Signals and Systems." 2nd ed. Prentice Hall.

See Also

`instantaneousAmplitude()` to extract the signal envelope, `instantaneousPhase()` to extract the instantaneous phase, `plv()` for phase-based connectivity.

Examples

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(500 * 4), nrow = 500)),
  samplingRate = 100
)

# Compute Hilbert transform
pe <- hilbertTransform(pe)

# Extract amplitude and phase
pe <- instantaneousAmplitude(pe)
pe <- instantaneousPhase(pe)
```

---

instantaneousAmplitude *Extract instantaneous amplitude (envelope)*

Description

Extracts the instantaneous amplitude (envelope) from the analytic signal.

Usage

```
instantaneousAmplitude(x, assay_name = "analytic", output_assay = "amplitude")
```

Arguments

`x`	A PhysioExperiment object with analytic signal.
`assay_name`	Name of the analytic signal assay.
`output_assay`	Name for the output assay.

Value

A `PhysioExperiment` object with an additional assay (default `"amplitude"`) containing the real-valued instantaneous amplitude (envelope) of the signal.

References

Oppenheim, A.V. & Willsky, A.S. (1997). "Signals and Systems." 2nd ed. Prentice Hall.

See Also

`hilbertTransform()` which must be called first, `instantaneousPhase()` for the companion phase extraction, `waveletTransform()` for alternative time-frequency decomposition.

Examples

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(500 * 4), nrow = 500)),
  samplingRate = 100
)

# First compute Hilbert transform, then extract amplitude
pe <- hilbertTransform(pe)
pe <- instantaneousAmplitude(pe)
```

`instantaneousPhase` *Extract instantaneous phase*

Description

Extracts the instantaneous phase from the analytic signal.

Usage

```
instantaneousPhase(x, assay_name = "analytic", output_assay = "phase")
```

Arguments

`x`	A PhysioExperiment object with analytic signal.
`assay_name`	Name of the analytic signal assay.
`output_assay`	Name for the output assay.

Value

A `PhysioExperiment` object with an additional assay (default `"phase"`) containing instantaneous phase values in radians (range $[-\pi, \pi]$).

References

Oppenheim, A.V. & Willsky, A.S. (1997). "Signals and Systems." 2nd ed. Prentice Hall.

See Also

`hilbertTransform()` which must be called first, `instantaneousAmplitude()` for the companion amplitude extraction, `plv()` for phase-based connectivity analysis.

Examples

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(500 * 4), nrow = 500)),
  samplingRate = 100
)

# First compute Hilbert transform, then extract phase
pe <- hilbertTransform(pe)
pe <- instantaneousPhase(pe)
```

---

pli — *Compute Phase Lag Index (PLI)*

Description

Calculates the Phase Lag Index between channel pairs, a measure of asymmetry in the phase difference distribution.

Usage

```
pli(x, freq_band, channels = NULL, assay_name = NULL)
```

Arguments

`x`	A PhysioExperiment object.
`freq_band`	Numeric vector of length 2 specifying frequency band (Hz).
`channels`	Integer vector of channel indices.
`assay_name`	Input assay name.

Details

PLI measures the asymmetry of the distribution of phase differences. Unlike PLV, PLI is insensitive to volume conduction effects that lead to zero-lag synchronization.

PLI = |mean(sign(Im(S_xy)))|, where S_xy is the cross-spectrum.

Value

A numeric matrix of PLI values (channel x channel) with values ranging from 0 (no asymmetry in phase distribution) to 1 (maximal asymmetry). Row and column names are set to channel names when available.

References

Lachaux, J.-P., et al. (1999). "Measuring phase synchrony in brain signals." Human Brain Mapping, 8(4), 194-208. doi:10.1002/(SICI)1097-0193(1999)8:4<194::AID-HBM4>3.0.CO;2-C

See Also

plv() for phase locking value, wPLI() for weighted phase lag index, coherence() for frequency-domain coherence, connectivityMatrix() for a unified connectivity interface.

Examples

```
set.seed(123)
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(4000), nrow = 1000, ncol = 4)),
  colData = S4Vectors::DataFrame(label = c("Fz", "Cz", "Pz", "Oz")),
  samplingRate = 256
)

# Compute PLI in alpha band (8-12 Hz)
pli_matrix <- pli(pe, freq_band = c(8, 12))
```

---

plotERP *Plot event-related potential (ERP) waveform*

Description

Generates an ERP plot from epoched data.

Usage

```
plotERP(x, channel = 1L, ci = 0.95, show_epochs = FALSE, epoch_alpha = 0.2)
```

Arguments

x	An epoched PhysioExperiment object.
channel	Integer index or character name of the channel.
ci	Confidence interval level (0-1). NULL for no CI.
show_epochs	Logical. If TRUE, shows individual epoch traces.
epoch_alpha	Alpha value for individual epoch traces.

Value

A ggplot object.

References

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

Delorme, A. & Makeig, S. (2004). "EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis." *Journal of Neuroscience Methods*, 134(1), 9-21. doi:10.1016/j.jneumeth.2003.10.009

See Also

`plotMultiChannel()` for multi-channel visualization, `plotTopomap()` for topographic maps, `epochData()` for creating epoched data, `bootstrapCI()` for bootstrap confidence intervals.

---

plotMultiChannel *Multi-channel visualization functions*

Description

Functions for visualizing multiple channels simultaneously. Plot multiple channels (butterfly or stacked)

Usage

```
plotMultiChannel(
  x,
  channels = NULL,
  sample = 1L,
  style = c("butterfly", "stacked"),
  offset = NULL,
  assay_name = NULL,
  colors = NULL
)
```

Arguments

`x`	A PhysioExperiment object.
`channels`	Integer vector of channel indices to plot. If NULL, plots all.
`sample`	Integer index for the sample (for 3D data).
`style`	Plot style: "butterfly" (overlaid) or "stacked" (offset).
`offset`	Numeric offset between channels for stacked plot.
`assay_name`	Optional assay name. If NULL, uses the default assay.
`colors`	Optional color vector for channels.

Details

Generates a plot showing multiple channels from the signal data.

Value

A ggplot object.

References

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

Delorme, A. & Makeig, S. (2004). "EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis." *Journal of Neuroscience Methods*, 134(1), 9-21. doi:10.1016/j.jneumeth.2003.10.009

See Also

`plotSignal()` for single-channel plots, `plotERP()` for event-related potential plots, `plotTopomap()` for topographic visualization.

Examples

```
# Create example data
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(500 * 4), nrow = 500)),
  colData = S4Vectors::DataFrame(label = c("Fz", "Cz", "Pz", "Oz")),
  samplingRate = 100
)

# Butterfly plot (all channels overlaid)
plotMultiChannel(pe, style = "butterfly")

# Stacked plot (channels offset vertically)
plotMultiChannel(pe, style = "stacked")

# Plot specific channels
plotMultiChannel(pe, channels = c(1, 3), style = "butterfly")
```

`plotPSD` *Plot power spectral density*

Description

Generates a PSD plot for specified channels.

Usage

```
plotPSD(x, channels = NULL, sample = 1L, log_scale = TRUE, freq_range = NULL)
```

Arguments

`x`	A PhysioExperiment object.
`channels`	Integer vector of channel indices. If NULL, plots all.
`sample`	Integer index for the sample (for 3D data).
`log_scale`	Logical. If TRUE, uses log scale for power.
`freq_range`	Numeric vector of length 2 specifying frequency range.

Value

A ggplot object.

References

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

Delorme, A. & Makeig, S. (2004). "EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis." *Journal of Neuroscience Methods*, 134(1), 9-21. doi:10.1016/j.jneumeth.2003.10.009

See Also

`plotSignal()` for time-domain signal plots, `fftSignals()` for FFT computation, `bandPower()` for frequency band power extraction, `plotSpectrogram()` for time-frequency visualization.

`plotSignal` *Plot a signal channel*

Description

Generates a simple line plot for the selected channel and sample from the default assay. Supports both 2D (time x channel) and 3D (time x channel x sample) assay arrays.

Usage

```
plotSignal(x, channel = 1L, sample = 1L, assay_name = NULL)
```

Arguments

`x`	A `PhysioExperiment` object.
`channel`	Integer index for the channel.
`sample`	Integer index for the sample (only used for 3D arrays).
`assay_name`	Optional assay name. If NULL, uses the default assay.

Value

A `ggplot` object.

References

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

See Also

`plotMultiChannel()` for multi-channel visualization, `plotPSD()` for power spectral density plots, `plotERP()` for event-related potential plots.

plotSpectrogram *Plot spectrogram*

Description

Creates a visualization of a spectrogram result.

Usage

```
plotSpectrogram(spec, freq_range = NULL, log_power = TRUE)
```

Arguments

spec	Spectrogram result from spectrogram().
freq_range	Optional frequency range to display.
log_power	If TRUE, displays log power (dB scale).

Value

A `ggplot` object showing the spectrogram as a filled raster plot with time on the x-axis and frequency on the y-axis.

References

Oppenheim, A.V. & Willsky, A.S. (1997). "Signals and Systems." 2nd ed. Prentice Hall.

See Also

`spectrogram()` to compute the spectrogram data, `plotPSD()` for power spectral density plots, `plotSignal()` for time-domain visualization.

Examples

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(1000 * 4), nrow = 1000)),
  samplingRate = 250
)

# Compute spectrogram
spec <- spectrogram(pe, channel = 1)

# Plot with frequency range filter
plotSpectrogram(spec, freq_range = c(1, 50))
```

plotTopomap *Topographic Map Visualization*

Description

Functions for plotting scalp topography maps showing the spatial distribution of signal values across electrode positions. Plot topographic map (scalp topography)

Usage

```
plotTopomap(
  x,
  values = NULL,
  time = NULL,
  channel_values = NULL,
  assay_name = NULL,
  resolution = 100L,
  contours = TRUE,
  head_shape = TRUE,
  electrodes = TRUE,
  palette = "RdBu",
  limits = NULL,
  title = NULL
)
```

Arguments

`x`	A PhysioExperiment object with electrode positions.
`values`	Optional numeric vector of values to plot. If NULL, uses values from the specified time point.
`time`	Time point in seconds to extract values (if values is NULL).
`channel_values`	Named vector of channel values (alternative to values).
`assay_name`	Optional assay name. If NULL, uses the default assay.
`resolution`	Grid resolution for interpolation. Default is 100.
`contours`	Logical. If TRUE, adds contour lines. Default is TRUE
`head_shape`	Logical. If TRUE, draws head outline. Default is TRUE.
`electrodes`	Logical. If TRUE, shows electrode positions. Default is TRUE.
`palette`	Color palette name or vector of colors.
`limits`	Numeric vector of length 2 for color scale limits.
`title`	Plot title. If NULL, auto-generated.

Details

Creates a 2D topographic map showing the spatial distribution of values across electrode positions on the scalp.

Value

A ggplot object.

References

Perrin, F., Pernier, J., Bertrand, O. & Echallier, J.F. (1989). "Spherical splines for scalp potential and current density mapping." *Electroencephalography and Clinical Neurophysiology*, 72(2), 184-187. doi:10.1016/00134694(89)901806

See Also

`plotTopomapSeries()` for topographic maps across time, `plotMultiChannel()` for multi-channel signal visualization, `plotERP()` for event-related potential plots.

Examples

```
# Create example with 10-20 electrode positions
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(400), nrow = 100, ncol = 4)),
  colData = S4Vectors::DataFrame(label = c("Fz", "Cz", "Pz", "Oz")),
  samplingRate = 100
)

# Apply 10-20 montage to get electrode positions
pe <- applyMontage(pe, "10-20")

# Plot topographic map at time = 0.5s
plotTopomap(pe, time = 0.5)

# Plot with custom values
plotTopomap(pe, values = c(1, 0.5, -0.5, -1))
```

---

plotTopomapSeries — *Plot topographic map animation*

Description

Creates a series of topographic maps across time.

Usage

```
plotTopomapSeries(x, times, ...)
```

Arguments

`x`	A PhysioExperiment object with electrode positions.
`times`	Numeric vector of time points to plot.
`...`	Additional arguments passed to plotTopomap.

Value

A list of ggplot objects.

References

Perrin, F., Pernier, J., Bertrand, O. & Echallier, J.F. (1989). "Spherical splines for scalp potential and current density mapping." *Electroencephalography and Clinical Neurophysiology*, 72(2), 184-187. doi:10.1016/00134694(89)901806

See Also

`plotTopomap()` for a single topographic map, `plotERP()` for event-related potential plots, `plotMultiChannel()` for multi-channel signal visualization.

Examples

```
## Not run:
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(400), nrow = 100, ncol = 4)),
  colData = S4Vectors::DataFrame(label = c("Fz", "Cz", "Pz", "Oz")),
  samplingRate = 100
)
pe <- applyMontage(pe, "10-20")

# Create topomaps at multiple time points
plots <- plotTopomapSeries(pe, times = c(0.1, 0.2, 0.3, 0.4))

## End(Not run)
```

---

plv *Compute Phase Locking Value (PLV)*

Description

Calculates the Phase Locking Value between channel pairs, measuring the consistency of phase difference across time.

Usage

```
plv(x, freq_band, channels = NULL, assay_name = NULL)
```

Arguments

`x`	A PhysioExperiment object.
`freq_band`	Numeric vector of length 2 specifying frequency band (Hz).
`channels`	Integer vector of channel indices.
`assay_name`	Input assay name.

Details

PLV measures the consistency of the phase difference between two signals. A value of 1 indicates perfect phase locking, while 0 indicates random phase relationship.

The signals are first bandpass filtered to the specified frequency band, then the analytic signal is computed using the Hilbert transform.

Value

A numeric matrix of PLV values (channel x channel) with values ranging from 0 (random phase relationship) to 1 (perfect phase locking). Row and column names are set to channel names when available.

References

Lachaux, J.-P., et al. (1999). "Measuring phase synchrony in brain signals." Human Brain Mapping, 8(4), 194-208. doi:10.1002/(SICI)1097-0193(1999)8:4<194::AID-HBM4>3.0.CO;2-C

See Also

`pli()` for phase lag index, `wPLI()` for weighted phase lag index, `coherence()` for frequency-domain coherence, `connectivityMatrix()` for a unified connectivity interface.

Examples

```
set.seed(123)
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(4000), nrow = 1000, ncol = 4)),
  colData = S4Vectors::DataFrame(label = c("Fz", "Cz", "Pz", "Oz")),
  samplingRate = 256
)

# Compute PLV in alpha band (8-12 Hz)
plv_matrix <- plv(pe, freq_band = c(8, 12))
```

`spectrogram` *Time-Frequency Analysis for PhysioExperiment*

Description

Functions for time-frequency analysis including wavelet transforms, spectrograms, and band power extraction. Compute spectrogram (Short-Time Fourier Transform)

Usage

```
spectrogram(
  x,
  window_size = 256L,
  overlap = 0.5,
  window_type = c("hanning", "hamming", "blackman", "rectangular"),
  channel = 1L,
  sample = 1L
)
```

Arguments

`x`	A PhysioExperiment object.
`window_size`	Window size in samples.
`overlap`	Overlap between windows (0-1).
`window_type`	Window function: "hanning", "hamming", "blackman", or "rectangular".
`channel`	Channel index to analyze.
`sample`	Sample index (for 3D data).

Details

Computes the spectrogram using STFT.

Value

A list with the following components:

`power`	Power spectrogram matrix (frequency x time)
`frequencies`	Numeric vector of frequencies in Hz
`times`	Numeric vector of time points in seconds
`sampling_rate`	The sampling rate used
`window_size`	Window size in samples
`overlap`	Overlap fraction

References

Oppenheim, A.V. & Willsky, A.S. (1997). "Signals and Systems." 2nd ed. Prentice Hall.

See Also

`waveletTransform()` for wavelet-based time-frequency analysis, `plotSpectrogram()` for visualization, `bandPower()` for band power extraction, `fftSignals()` for simple FFT.

Examples

```
# Create example data
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(1000 * 4), nrow = 1000)),
  samplingRate = 250
)

# Compute spectrogram for channel 1
spec <- spectrogram(pe, channel = 1)

# Plot spectrogram
plotSpectrogram(spec, freq_range = c(1, 40))
```

`tTestEpochs` *Statistical Testing for PhysioExperiment*

Description

Functions for statistical analysis of physiological signal data, including t-tests, ANOVA, cluster-based permutation tests, and effect sizes. Pointwise t-test across epochs

Usage

```
tTestEpochs(
  x,
  condition1 = NULL,
  condition2 = NULL,
  mu = 0,
  paired = FALSE,
  alternative = c("two.sided", "less", "greater"),
  var.equal = FALSE
)
```

Arguments

`x`	An epoched PhysioExperiment object (4D data).
`condition1`	Indices or logical vector for first condition epochs.
`condition2`	Indices or logical vector for second condition epochs. If NULL, performs one-sample t-test against mu.
`mu`	Value to test against for one-sample t-test (default: 0).
`paired`	Logical; if TRUE, performs paired t-test.
`alternative`	Alternative hypothesis: "two.sided", "less", or "greater".
`var.equal`	Logical; if TRUE, assumes equal variances.

Details

Performs t-tests at each time point and channel, comparing epochs against a baseline or between two conditions.

Value

A list containing:

t_values	Matrix of t-statistics (time x channel)
p_values	Matrix of p-values (time x channel)
df	Degrees of freedom
n1, n2	Sample sizes for each condition

References

Maris, E. & Oostenveld, R. (2007). "Nonparametric statistical testing of EEG- and MEG-data." *Journal of Neuroscience Methods*, 164(1), 177-190. doi:10.1016/j.jneumeth.2007.03.024

See Also

`anovaEpochs()` for multi-group comparisons, `clusterPermutationTest()` for multiple comparison correction, `effectSize()` for Cohen's d effect size, `correctPValues()` for p-value correction methods.

Examples

```
# Create example epoched data
set.seed(123)
epochs <- array(rnorm(100 * 4 * 20 * 1), dim = c(100, 4, 20, 1))
pe <- PhysioExperiment(
  assays = list(epoched = epochs),
  samplingRate = 100
)
# One-sample t-test against zero
result <- tTestEpochs(pe)
# Two-sample t-test comparing conditions
result2 <- tTestEpochs(pe, condition1 = 1:10, condition2 = 11:20)
```

---

waveletTransform *Wavelet transform*

Description

Computes the continuous wavelet transform using Morlet wavelets.

Usage

```
waveletTransform(
  x,
  frequencies = seq(1, 40, by = 1),
  n_cycles = 7,
  channel = 1L,
  sample = 1L
)
```

Arguments

`x`	A PhysioExperiment object.
`frequencies`	Numeric vector of frequencies to analyze.
`n_cycles`	Number of wavelet cycles (can be scalar or vector).
`channel`	Channel index to analyze.
`sample`	Sample index (for 3D data).

Value

A list with the following components:

`power`	Power matrix (frequency x time)
`phase`	Phase matrix in radians (frequency x time)
`frequencies`	Numeric vector of analyzed frequencies in Hz
`times`	Numeric vector of time points in seconds
`sampling_rate`	The sampling rate used
`n_cycles`	Number of wavelet cycles per frequency

References

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

See Also

`spectrogram()` for STFT-based time-frequency analysis, `bandPower()` for band power extraction, `hilbertTransform()` for analytic signal computation.

Examples

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(500 * 4), nrow = 500)),
  samplingRate = 100
)

# Compute wavelet transform (1-30 Hz)
wt <- waveletTransform(pe, frequencies = seq(1, 30), channel = 1)
```

```
# Access power and phase
dim(wt$power)  # frequency x time
```

---

wPLI *Compute weighted Phase Lag Index (wPLI)*

Description

Calculates the weighted Phase Lag Index, which is less sensitive to noise than standard PLI.

Usage

```
wPLI(x, freq_band, channels = NULL, assay_name = NULL)
```

Arguments

`x`	A PhysioExperiment object.
`freq_band`	Numeric vector of length 2 specifying frequency band (Hz).
`channels`	Integer vector of channel indices.
`assay_name`	Input assay name.

Details

wPLI weights the contribution of each phase difference by the magnitude of the imaginary component, reducing the influence of noise sources.

Value

A numeric matrix of wPLI values (channel x channel) with values ranging from 0 to 1. Row and column names are set to channel names when available.

References

Lachaux, J.-P., et al. (1999). "Measuring phase synchrony in brain signals." Human Brain Mapping, 8(4), 194-208. doi:10.1002/(SICI)1097-0193(1999)8:4<194::AID-HBM4>3.0.CO;2-C

See Also

`plv()` for phase locking value, `pli()` for phase lag index, `coherence()` for frequency-domain coherence, `connectivityMatrix()` for a unified connectivity interface.

Examples

```
set.seed(123)
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(4000), nrow = 1000, ncol = 4)),
  samplingRate = 256
)

# Compute wPLI in theta band (4-8 Hz)
wpli_matrix <- wPLI(pe, freq_band = c(4, 8))
```

Index