

Package: PhysioAnalysis (via r-universe)

October 3, 2026

Title Analysis and Visualization for PhysioExperiment Objects

Version 0.5.30

Description Provides analysis and visualization functions for physiological signal data stored in 'PhysioExperiment' objects, which extend the Bioconductor 'SummarizedExperiment' container. Methods include the fast Fourier transform ('FFT'), time-frequency decomposition, epoching, functional connectivity (coherence and phase-based measures), graph and network metrics, statistical parametric mapping, and cluster-based permutation testing. Visualization utilities cover scalp topographic maps ('topoplots'), multichannel signal displays, and network graphs. Compiled routines are provided through 'Rcpp' and 'RcppArmadillo'.

Depends R (>= 4.2), PhysioExperiment (>= 2.0.0)

Imports methods, SummarizedExperiment, S4Vectors, signal, stats, graphics, grDevices, ggplot2, parallel, Rcpp (>= 1.0.0)

LinkingTo Rcpp, RcppArmadillo

Suggests PhysioCore, igraph, knitr, rmarkdown, testthat (>= 3.2.0)

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

Collate 'PhysioAnalysis-package.R' 'RcppExports.R'
'utils-provenance.R' 'ops-fft.R' 'ops-timefreq.R'
'ops-specparam.R' 'ops-epoch.R' 'ops-epoch-sliding.R'
'ops-connectivity.R' 'ops-network.R' 'stats-spm.R'
'stats-spm-regress.R' 'stats-spm-manova.R'
'stats-spm-nonparam.R' 'stats-tests.R'
'stats-effectsizes-rank.R' 'stats-network.R' 'vis-plot.R'
'vis-multichannel.R' 'vis-topomap.R' 'vis-network.R'
'stats-functional-regression.R' 'stats-functional-mixed.R'
'reexports-stats.R' 'zzz.R'

VignetteBuilder knitr

RoxygenNote 7.3.3

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

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

Config/pak/sysreqs zlib1g-dev

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

Date/Publication 2026-10-03 13:11:42 UTC

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

RemoteRef HEAD

RemoteSha f29df30bcb0879694ff32e1bef9fab3a8ee21842

Contents

adfTest	4
adjacencyMatrix	5
anovaEpochs	6
arSpectrum	7
arYuleWalker	8
autocorrelation	9
averageEpochs	10
bandPower	10
betweennessCentrality	11
binarizeNetwork	12
bootstrapCI	13
cliffsDelta	14
clusteringCoefficient	15
clusterPermutationTest	15
coherence	17
connectivityMatrix	19
correctPValues	20
correlationMatrix	21
correlationTest	22
cronbachAlpha	22
crossSpectrum	23
dunnTest	24
effectSize	25
eigenvectorCentrality	26
epochData	27
epochSliding	29
epochTimes	29
fftSignals	30
findSignificantWindows	31
friedmanTest	32
functionalMixedModel	33
functionalRegression	34
globalEfficiency	35

grandAverage	36
graphLaplacian	36
hilbertTransform	37
instantaneousAmplitude	38
instantaneousPhase	39
interquartileRange	40
intraclassCorrelation	41
jarqueBeraTest	41
kendallTau	42
kpssTest	43
kruskalTest	44
ksNormalityTest	45
leveneTest	46
linearRegression	46
ljungBoxTest	47
localEfficiency	48
medianAbsDev	49
modularity	50
multipleRegression	50
nemenyiTest	51
networkBasedStatistic	52
nodeDegree	53
oneWayAnova	54
pairedTTest	55
partialAutocorrelation	55
partialCorrelation	56
pathLength	57
pli	58
plotAdjacencyMatrix	59
plotDynamicConnectivity	60
plotERP	61
plotMultiChannel	62
plotNBSnetwork	63
plotNetwork	64
plotNetworkMetrics	65
plotNetworkStability	66
plotPSD	66
plotSignal	67
plotSpecparam	68
plotSpectrogram	69
plotSPM	70
plotTopomap	71
plotTopomapSeries	73
plv	74
rankBiserial	75
scalarOnFunctionRegression	76
signalMoments	77
slidingWindowConnectivity	78

smallWorldness	79
specparam	80
specparamBiomarker	81
spectralClustering	82
spectralDecomposition	83
spectrogram	83
spmAnova	85
spmMANOVA	85
spmPairedTTest	86
spmRegression	87
spmSnPM	88
spmTTest	89
temporalStability	91
tfce	91
thresholdNetwork	92
trimmedMean	93
tTestEpochs	94
tukeyHSD	95
twoSampleTTest	96
waveletTransform	97
wPLI	98
Index	100

adfTest	<i>Augmented Dickey-Fuller (ADF) unit-root test statistic</i>
---------	---

Description

Computes the Augmented Dickey-Fuller test statistic for a unit root (non-stationarity) in a one-dimensional signal. The test regresses the first difference on a constant, the lagged level, and lag lagged first differences, $\Delta y_t = \alpha + \beta y_{t-1} + \sum_{i=1}^L \gamma_i \Delta y_{t-i} + \varepsilon_t$, and returns the t-statistic on β . A large negative statistic (below the Dickey-Fuller critical value) rejects the unit-root null in favour of stationarity. The statistic is the same deterministic OLS quantity as `statsmodels.tsa.stattools.adfuller(..., autolag = None)`.

Usage

```
adfTest(x, lag = 1L)
```

Arguments

- x A numeric vector (the time series).
- lag Number of augmenting lagged differences L (default 1).

Details

Note: only the *statistic* is computed here (it is an exact regression quantity). The critical values and p-value require the Dickey-Fuller / MacKinnon reference tables (an approximation), so they are not returned; compare the statistic against the tabulated critical value for the decision.

Value

A list with *statistic* (the ADF t-statistic on the lagged level), *n* (the regression sample size), and *lag*.

References

Dickey, D.A., Fuller, W.A. (1979). Distribution of the estimators for autoregressive time series with a unit root. *JASA*, 74, 427-431.

See Also

[autocorrelation](#), [ljungBoxTest](#)

Examples

```
set.seed(1)
adfTest(cumsum(rnorm(500)), lag = 1)$statistic # random walk -> not far below 0
```

adjacencyMatrix	Create adjacency matrix from connectivity
-----------------	---

Description

Converts a connectivity matrix into an adjacency matrix for network analysis.

Usage

```
adjacencyMatrix(connectivity, threshold = NULL, absolute = FALSE)
```

Arguments

connectivity	A connectivity matrix or result from connectivityMatrix().
threshold	Threshold value for binarization. If NULL, keeps weighted edges.
absolute	If TRUE, uses absolute values before thresholding.

Value

A square adjacency matrix.

Examples

```
set.seed(123)
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(500 * 4), nrow = 500, ncol = 4)),
  samplingRate = 100
)
conn <- correlationMatrix(pe)
adj <- adjacencyMatrix(conn, threshold = 0.3)
```

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](https://doi.org/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(epochs = 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")
```

arSpectrum

*Autoregressive (parametric) spectral density***Description**

Computes the parametric AR spectral density of a one-dimensional signal from an autoregressive model of order p fitted by Yule-Walker (via [arYuleWalker](#)): $S(f) = \sigma^2 / |1 - \sum_k \phi_k e^{-i2\pi f k}|^2$, evaluated on `n_freq` normalized frequencies in $[0, 0.5]$. Unlike the periodogram / Welch estimate, the AR spectrum is smooth and low-variance, with sharp peaks at resonant frequencies, using only order parameters. It matches `stats::spec.ar` (including its small-sample innovation-variance convention $N/(N - p - 1)$).

Usage

```
arSpectrum(x, order = 8L, n_freq = 500L)
```

Arguments

<code>x</code>	A numeric vector (the time series).
<code>order</code>	Autoregressive order p (default 8).
<code>n_freq</code>	Number of frequencies in $[0, 0.5]$ (default 500).

Value

A list with `freq` (normalized frequencies, cycles/sample), `spec` (the AR spectral density), and `order`.

References

Percival, D.B., Walden, A.T. (1993). Spectral Analysis for Physical Applications. Cambridge University Press.

See Also

[arYuleWalker](#), [spec.ar](#)

Examples

```
set.seed(1)
s <- arSpectrum(as.numeric(arima.sim(list(ar = c(0.6, -0.3)), 500)), order = 2)
s$freq[which.max(s$spec)] # peak (normalized) frequency
```

arYuleWalker

Autoregressive model fit by Yule-Walker (AR(p))

Description

Fits an autoregressive model of order `order` to a one-dimensional signal by the Yule-Walker method: the AR coefficients solve the Yule-Walker equations formed from the signal's biased demeaned autocovariance (obtained via the Durbin-Levinson recursion on [autocorrelation](#)), and the innovation variance is the residual (one-step prediction) variance. This is the same definition as `statsmodels.regression.linear_model.yule_walker(method = "mle")`. AR modelling underlies forecasting, parametric (AR) spectra, and – with the order chosen from the [partialAutocorrelation](#) cut-off – compact descriptions of oscillatory dynamics.

Usage

```
arYuleWalker(x, order = 4L)
```

Arguments

`x` A numeric vector (the time series).
`order` Autoregressive order p (default 4).

Value

A list with `ar` (the length-order AR coefficient vector $\phi_1, \dots, \phi_p$), `var_pred` (the innovation / one-step prediction variance), and `order`.

References

Box, G.E.P., Jenkins, G.M. (1976). Time Series Analysis: Forecasting and Control. Holden-Day.

See Also

[autocorrelation](#), [partialAutocorrelation](#), [ar.yw](#)

Examples

```
set.seed(1)
arYuleWalker(as.numeric(arima.sim(list(ar = c(0.5, -0.3)), 500)), order = 2)
```

autocorrelation	<i>Autocorrelation function (ACF)</i>
-----------------	---------------------------------------

Description

Computes the sample autocorrelation function of a one-dimensional signal at lags 0, 1, ..., lag_max using the biased estimator (dividing each autocovariance by N), with the mean removed by default – the same definition as `stats::acf()` and `statsmodels.tsa.acf(adjusted = FALSE)`. The autocorrelation measures how similar a signal is to a time-shifted copy of itself, and is the basic tool for detecting periodicity, rhythmicity, and the decorrelation time of a physiological signal.

Usage

```
autocorrelation(x, lag_max = 30L, demean = TRUE)
```

Arguments

x	A numeric vector (the time series).
lag_max	Maximum lag (default 30); the returned vector has lag_max + 1 values (lags 0 through lag_max).
demean	If TRUE (default), subtract the mean before computing the autocovariance (the standard definition).

Value

A numeric vector of autocorrelation coefficients at lags 0:lag_max; the lag-0 value is always 1.

References

Box, G.E.P., Jenkins, G.M. (1976). Time Series Analysis: Forecasting and Control. Holden-Day.

See Also

[acf](#)

Examples

```
set.seed(1)
autocorrelation(sin(seq(0, 20 * pi, length.out = 500)), lag_max = 20)
```

averageEpochs	<i>Average epochs</i>
---------------	-----------------------

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	<i>Compute band power</i>
-----------	---------------------------

Description

Extracts power in specified frequency bands.

Usage

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

Arguments

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

betweennessCentrality *Compute betweenness centrality*

Description

Calculates betweenness centrality for each node.

Usage

```
betweennessCentrality(
  adjacency,
  normalized = TRUE,
  use_cpp = TRUE,
  n_cores = 1L
)
```

Arguments

adjacency	An adjacency matrix.
normalized	If TRUE, normalizes by $(n-1)(n-2)/2$.
use_cpp	If TRUE (default), use the compiled C++ backend, falling back to R if unavailable.
n_cores	Number of OpenMP threads for the C++ backend (default 1).

Value

A numeric vector of betweenness centrality values.

Examples

```
# Star network - center node should have highest betweenness
adj <- matrix(0, 4, 4)
adj[1, 2:4] <- 1
adj[2:4, 1] <- 1
betweennessCentrality(adj)
```

binarizeNetwork	<i>Binarize network</i>
-----------------	-------------------------

Description

Converts a weighted adjacency matrix to a binary network.

Usage

```
binarizeNetwork(adjacency, threshold = 0)
```

Arguments

adjacency	A weighted adjacency matrix.
threshold	Threshold for binarization. Default 0 (any non-zero edge).

Value

A binary adjacency matrix (0s and 1s).

Examples

```
set.seed(123)
adj <- matrix(c(0, 0.5, 0.3, 0.5, 0, 0.8, 0.3, 0.8, 0), 3, 3)
bin <- binarizeNetwork(adj)
```

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](https://doi.org/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)
```

cliffsDelta

Cliff's delta for two independent samples

Description

Cliff's delta, $\text{delta} = P(X > Y) - P(X < Y)$, a non-parametric effect size in $[-1, 1]$ whose sign follows the direction of the difference. The confidence interval uses Cliff's (1993) consistent variance with his asymmetric in-bounds transform.

Usage

```
cliffsDelta(x, y, conf_level = 0.95)
```

Arguments

`x, y` Numeric vectors for the two samples.
`conf_level` Confidence level (default 0.95).

Value

A list with `delta`, `ci_lower`, `ci_upper`, the variance estimate and the sample sizes.

References

Cliff, N. (1993). Psychol Bull 114(3):494-509.

See Also

[rankBiserial\(\)](#), [effectSize\(\)](#)

Examples

```
cliffsDelta(c(5, 6, 7, 8), c(1, 2, 3, 9))
```

clusteringCoefficient *Compute clustering coefficient*

Description

Calculates the local clustering coefficient for each node.

Usage

```
clusteringCoefficient(  
  adjacency,  
  weighted = FALSE,  
  use_cpp = TRUE,  
  n_cores = 1L  
)
```

Arguments

adjacency	An adjacency matrix.
weighted	If TRUE, uses weighted clustering coefficient.
use_cpp	If TRUE (default), use the compiled C++ backend, falling back to R if unavailable.
n_cores	Number of OpenMP threads for the C++ backend (default 1).

Value

A numeric vector of clustering coefficients.

Examples

```
# Fully connected triangle  
adj <- matrix(c(0, 1, 1, 1, 0, 1, 1, 1, 0), 3, 3)  
clusteringCoefficient(adj) # Should be 1 for all nodes
```

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,
  n_cores = NULL,
  method = c("cluster", "tfce"),
  tfce_E = 0.5,
  tfce_H = 2,
  tfce_dh = NULL,
  adjacency = NULL
)
```

Arguments

<code>x</code>	An epoched <code>PhysioExperiment</code> object (4D data).
<code>condition1</code>	Indices for first condition.
<code>condition2</code>	Indices for second condition. If <code>NULL</code> , tests against zero.
<code>n_permutations</code>	Number of permutations (default: 1000).
<code>cluster_threshold</code>	Initial threshold for cluster formation (p-value).
<code>tail</code>	Test type: 0 = two-tailed, 1 = right tail, -1 = left tail.
<code>seed</code>	Random seed for reproducibility.
<code>n_cores</code>	Number of cores for parallel processing. Default <code>NULL</code> uses sequential processing. Set to <code>parallel::detectCores() - 1</code> for maximum speed.
<code>method</code>	"cluster" (default) for cluster-mass inference, or "tfce" for Threshold-Free Cluster Enhancement (tfce()) with a per-point max-statistic FWER correction.
<code>tfce_E</code> , <code>tfce_H</code>	Extent and height exponents for method = "tfce" (defaults: 0.5 and 2).
<code>tfce_dh</code>	Threshold step for TFCE (<code>NULL</code> integrates exactly over the distinct statistic levels).
<code>adjacency</code>	Optional channel-adjacency matrix for TFCE; <code>NULL</code> uses neighbouring-channel adjacency.

Value

A list containing:

<code>clusters</code>	List of identified clusters with their statistics
<code>cluster_p</code>	P-values for each cluster
<code>t_obs</code>	Observed t-values matrix
<code>cluster_mask</code>	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](https://doi.org/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(epochs = epochs),
  samplingRate = 100
)
# Cluster permutation test
result <- clusterPermutationTest(pe, 1:10, 11:20, n_permutations = 100)

# With parallel processing (faster for large datasets)
# result <- clusterPermutationTest(pe, 1:10, 11:20, n_cores = 4)
```

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

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

<code>coherence</code>	3D array (freq x channel x channel) of coherence values
<code>frequencies</code>	Frequency vector
<code>channel_names</code>	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	<i>Compute connectivity matrix for a frequency band</i>
--------------------	---

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 <code>PhysioExperiment</code> 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	<i>Multiple comparison correction</i>
----------------	---------------------------------------

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](https://doi.org/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	<i>Compute correlation matrix between channels</i>
-------------------	--

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 <code>PhysioExperiment</code> 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)
```

correlationTest	<i>Correlation Test</i>
-----------------	-------------------------

Description

Tests the association between two matched numeric vectors. Returns the correlation coefficient, its two-sided p-value (from the exact t-test on the Pearson/Spearman correlation) and the degrees of freedom. For method = "pearson" it reproduces `stats::cor.test` and `scipy.stats.pearsonr` bit-for-bit.

Usage

```
correlationTest(x, y = NULL, method = c("pearson", "spearman"))
```

Arguments

x	A numeric vector, OR (when y is NULL) a matrix whose first two columns are the paired variables.
y	A numeric vector matched to x, or NULL if x is a matrix.
method	Correlation method: "pearson" (default) or "spearman" (rank correlation; the t-based p is the large-sample approximation).

Value

A list with `statistic` (the correlation coefficient), `p_value` (two-sided) and `df` ($n - 2$).

See Also

[pairedTTest](#), [signalMoments](#).

Examples

```
set.seed(1); x <- rnorm(50); correlationTest(x, x + rnorm(50))$statistic # ~0.7
```

cronbachAlpha	<i>Cronbach's alpha (internal-consistency reliability)</i>
---------------	--

Description

Cronbach's alpha, the standard index of a scale's internal-consistency reliability: how consistently a set of k items measures a single construct. Computed from the item variances and the total-score variance. The raw alpha and the mean inter-item correlation reproduce base R, the psych package (`psych::alpha raw_alpha`) and `pingouin.cronbach_alpha` bit-for-bit.

Usage

```
cronbachAlpha(x)
```

Arguments

x A respondents-by-items numeric matrix or data frame (rows = respondents, columns = scale items). Rows with any missing value are dropped.

Value

A list with alpha (raw Cronbach's alpha), average_r (mean inter-item correlation), k (items) and n (respondents).

Examples

```
cronbachAlpha(matrix(round(runif(150, 1, 5)), ncol = 5))
```

crossSpectrum	<i>Compute cross-spectral density</i>
---------------	---------------------------------------

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)
```

dunnTest	<i>Dunn's test (post-hoc pairwise comparisons after Kruskal-Wallis)</i>
----------	---

Description

Dunn's test: all pairwise rank-based comparisons following a Kruskal-Wallis test ([kruskalTest](#)), using the shared tie-corrected rank variance and a chosen multiple-comparison adjustment. It is the non-parametric counterpart of Tukey's HSD ([tukeyHSD](#)). The z-statistics and adjusted p-values reproduce base R's Dunn formula, the PMCMRplus package and `scikit_posthocs.posthoc_dunn` to machine precision.

Usage

```
dunnTest(x, ..., p_adjust = c("bonferroni", "none", "holm", "BH"))
```

Arguments

x	Either a named list of $k \geq 2$ numeric group vectors, or the first group's numeric vector (with further groups passed through <code>...</code>).
...	Additional numeric group vectors when x is a single vector.
p_adjust	Multiple-comparison adjustment: "bonferroni" (default), "none", "holm" or "BH".

Value

A data frame with one row per pair: comparison, z, p_value (two-sided, unadjusted) and p_adj, carrying k, n as attributes.

See Also

[kruskalTest](#), [tukeyHSD](#)

Examples

```
dunnTest(list(a = rnorm(20), b = rnorm(20, 1), c = rnorm(20, 2)))
```

effectSize	<i>Compute effect size (Cohen's d / Hedges g)</i>
------------	---

Description

Calculates the standardized mean difference at each time point and channel, with an exact non-central-t confidence interval (Cumming 2014) and an optional Hedges small-sample bias correction.

Usage

```
effectSize(  
  x,  
  condition1 = NULL,  
  condition2 = NULL,  
  pooled = TRUE,  
  correction = c("none", "hedges"),  
  conf_level = 0.95  
)
```

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.
correction	"none" for Cohen's d (default), or "hedges" to apply the Hedges bias correction $J = 1 - 3 / (4 * df - 1)$ to the estimate and interval.
conf_level	Confidence level for the interval (default 0.95).

Value

A list containing:

<code>d</code>	Matrix of standardized mean differences (time x channel)
<code>ci_lower</code>	Lower confidence limit for <code>d</code> (non-central <code>t</code>)
<code>ci_upper</code>	Upper confidence limit for <code>d</code> (non-central <code>t</code>)
<code>correction</code>	The bias correction applied

References

Cumming, G. (2014). The New Statistics. *Psychol Sci* 25(1):7-29. Hedges, L.V. (1981). *J Educ Stat* 6(2):107-128.

See Also

[tTestEpochs\(\)](#) for significance testing, [bootstrapCI\(\)](#) for bootstrap confidence intervals, [rankBiserial\(\)](#) and [cliffsDelta\(\)](#) for rank-based effect sizes.

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(epochs = epochs),
  samplingRate = 100
)
# Effect size for one-sample
result <- effectSize(pe, condition1 = 1:10)
# Hedges g between conditions
result2 <- effectSize(pe, condition1 = 1:10, condition2 = 11:20,
  correction = "hedges")
```

eigenvectorCentrality *Compute eigenvector centrality*

Description

Calculates eigenvector centrality for each node.

Usage

```
eigenvectorCentrality(adjacency, max_iter = 100, tol = 1e-06, use_cpp = TRUE)
```

Arguments

adjacency	An adjacency matrix.
max_iter	Maximum iterations for power method.
tol	Convergence tolerance.
use_cpp	If TRUE (default), use the compiled C++ backend, falling back to R if unavailable.

Value

A numeric vector of eigenvector centrality values.

Examples

```
adj <- matrix(c(0, 1, 1, 1, 0, 1, 1, 1, 0), 3, 3)
eigenvectorCentrality(adj)
```

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,
  events = NULL,
  min_length = 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. Ignored if events is provided.

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.
events	An EventQuery object for advanced event filtering. If provided, overrides event_type.
min_length	Minimum epoch length in seconds when using variable-length epochs (tmax as event name). Epochs shorter than this are excluded.

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)
```

epochSliding	<i>Create epochs using sliding window</i>
--------------	---

Description

Segments continuous data into overlapping epochs using a sliding window approach. This is useful for time-frequency analysis or when events are not available.

Usage

```
epochSliding(x, window, step = NULL, baseline = NULL)
```

Arguments

x	A <code>PhysioExperiment</code> object
window	Window size in seconds
step	Step size in seconds (default: window/2 for 50% overlap)
baseline	Optional baseline correction window as c(start, end) in seconds relative to epoch start. NULL for no correction.

Value

`PhysioExperiment` with epoched data

Examples

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(1000 * 4), nrow = 1000, ncol = 4)),
  samplingRate = 100)
# Create 0.5 second windows with 0.1 second step
epoched <- epochSliding(pe, window = 0.5, step = 0.1)
```

epochTimes	<i>Get epoch time vector</i>
------------	------------------------------

Description

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

Usage

```
epochTimes(x)
```

Arguments

x	An epoched <code>PhysioExperiment</code> 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

<code>p_values</code>	Vector of p-values across time.
<code>times</code>	Vector of time points.
<code>alpha</code>	Significance threshold (default: 0.05).
<code>min_duration</code>	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](https://doi.org/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)
```

friedmanTest	<i>Friedman rank-sum test (non-parametric repeated-measures omnibus)</i>
--------------	--

Description

The non-parametric repeated-measures omnibus test: compares $k \geq 2$ matched treatments (columns) measured on the same N blocks/subjects (rows), by ranking within each block and testing whether the rank totals differ. It is the within-subject, distribution-free counterpart of the one-way ANOVA ([oneWayAnova](#)). The tie-corrected chi-square statistic and its p-value reproduce base R `stats::friedman.test` bit-for-bit and `scipy.stats.friedmanchisquare` to machine precision.

Usage

```
friedmanTest(x, ...)
```

Arguments

<code>x</code>	A blocks-by-treatments numeric matrix (rows = subjects/blocks, columns = conditions/treatments), or a list of k equal-length numeric vectors (one per treatment, aligned by block). Rows with any missing value are dropped (complete blocks only).
<code>...</code>	Additional treatment vectors when <code>x</code> is a single vector.

Value

A list with `statistic` (the tie-corrected Friedman chi-square), `p_value`, `df` ($k - 1$), `k` (treatments) and `n` (complete blocks).

See Also

[oneWayAnova](#), [pairedTTest](#)

Examples

```
m <- matrix(rnorm(40) + rep(c(0, 0.5, 1, 1.5), each = 10), nrow = 10)
friedmanTest(m)
```

functionalMixedModel *Functional random-intercept (multilevel) model for waveforms*

Description

Fits the subject random-intercept model to a set of curves nested in subjects. It splits the pointwise variance into between- and within-subject components (a functional ICC curve) and, when group is given, tests the between-subject fixed effect on the subject-mean curves so strides are not treated as independent.

Usage

```
functionalMixedModel(
  curves,
  subject,
  group = NULL,
  n_perm = 1000L,
  seed = NULL
)
```

Arguments

curves	An $N \times P$ matrix of curves: N observations (e.g. strides pooled over subjects), P domain points.
subject	Length- N grouping identifying the random-intercept level (the subject each curve belongs to).
group	Optional between-subject fixed effect (length N), constant within each subject (e.g. a patient/control label).
n_perm	Permutations for the curve-wide fixed-effect p-value (default 1000; 0 to skip).
seed	Optional RNG seed for the permutation test.

Value

a functional_mixed object: mean_curve, var_between, var_within, icc (functional ICC curve), subject_means (one row per subject), n_subjects, and – when group is given – f_curve, p_pointwise, p_global (subject-level fixed-effect test) plus group_diff (the signed difference curve for two groups).

References

Morris JS (2015) Functional regression, Annu Rev Stat Appl; Pini & Vantini (2017) interval-wise testing.

See Also

[functionalRegression\(\)](#), [spmTTest\(\)](#)

Examples

```
set.seed(1)
P <- 101; t <- seq(0, 1, length.out = P)
make <- function(off) t(sapply(1:12, function(k) off + sin(2 * pi * t) + rnorm(P, 0, 0.1)))
ctrl <- do.call(rbind, lapply(rnorm(6, 0, 0.5), make))
pat <- do.call(rbind, lapply(rnorm(6, 1.0, 0.5), make)) # shifted group
Y <- rbind(ctrl, pat)
subj <- rep(1:12, each = 12); grp <- rep(c("ctrl", "pat"), each = 72)
fm <- functionalMixedModel(Y, subj, grp, n_perm = 200)
fm$p_global # group difference, subject-level correct
```

functionalRegression *Function-on-scalar regression (coefficient curves)*

Description

Models a set of response CURVES as a linear function of one or more scalar predictors, giving a coefficient curve for each predictor (how the waveform changes per unit predictor) with pointwise inference and a domain-wide permutation test.

Usage

```
functionalRegression(curves, predictor, n_perm = 1000L, seed = NULL)
```

Arguments

curves	An $N \times P$ matrix: N observations (curves), P domain points (e.g. % gait cycle).
predictor	A length- N vector, $N \times q$ matrix, or data frame of scalar predictors. An intercept is added automatically.
n_perm	Permutations for the curve-wide (family-wise) p-value per coefficient (default 1000); set 0 to skip.
seed	Optional integer seed for the permutation test (reproducibility).

Value

a `fosr_result`: coefficients ($k \times P$, incl. intercept), `se`, `tval`, `p_pointwise` ($k \times P$), `p_global` (length k , family-wise), the fitted curves and terms.

References

Ramsay JO, Silverman BW (2005) Functional Data Analysis; Reiss PT, et al. (2010) function-on-scalar regression.

See Also

[scalarOnFunctionRegression\(\)](#), [spmRegression\(\)](#)

Examples

```

set.seed(1)
P <- 101; t <- seq(0, 1, length.out = P)
speed <- runif(40, 0.8, 1.6)
curves <- outer(speed, rep(1, P)) * matrix(sin(2 * pi * t), 40, P, byrow = TRUE) +
  matrix(rnorm(40 * P, 0, 0.05), 40, P)
fit <- functionalRegression(curves, speed, n_perm = 200)
fit$p_global           # speed coefficient curve is significant

```

globalEfficiency	<i>Compute global efficiency</i>
------------------	----------------------------------

Description

Calculates the global efficiency of a network.

Usage

```
globalEfficiency(adjacency, weighted = FALSE, use_cpp = TRUE)
```

Arguments

adjacency	An adjacency matrix.
weighted	If TRUE, uses weighted paths.
use_cpp	If TRUE (default), use the compiled C++ backend, falling back to R if unavailable.

Value

Global efficiency value (0-1).

Examples

```

# Fully connected - maximum efficiency
adj <- matrix(1, 4, 4)
diag(adj) <- 0
globalEfficiency(adj)

```

grandAverage	<i>Grand average across subjects/samples</i>
--------------	--

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.

graphLaplacian	<i>Compute graph Laplacian</i>
----------------	--------------------------------

Description

Calculates the graph Laplacian matrix.

Usage

```
graphLaplacian(adjacency, normalized = FALSE)
```

Arguments

adjacency	An adjacency matrix.
normalized	If TRUE, returns normalized Laplacian.

Value

The Laplacian matrix.

Examples

```
adj <- matrix(c(0, 1, 1, 1, 0, 1, 1, 1, 0), 3, 3)
L <- graphLaplacian(adj)
```

hilbertTransform	<i>Hilbert transform for instantaneous amplitude/phase</i>
------------------	--

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 <code>PhysioExperiment</code> 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	<i>Extract instantaneous phase</i>
--------------------	------------------------------------

Description

Extracts the instantaneous phase from the analytic signal.

Usage

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

Arguments

x	A <code>PhysioExperiment</code> 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)
```

interquartileRange	<i>Interquartile Range (robust spread)</i>
--------------------	--

Description

Computes the interquartile range (IQR) of a one-dimensional signal – the spread of the middle 50%, $Q_3 - Q_1$ – a robust, quantile-based measure of dispersion that ignores the tails entirely. The quartiles use R's default type-7 quantile (the same linear interpolation as `scipy.stats.iqr` and `numpy.percentile`), so the result matches both bit-for-bit.

Usage

```
interquartileRange(x, type = 7L)
```

Arguments

x	A numeric vector (the time series).
type	Quantile algorithm passed to quantile (default 7, matching base R, scipy and numpy).

Details

A quantile-based robust spread that complements [medianAbsDev](#) (a deviation-based robust scale) and [signalMoments](#) (the classical SD). For approximately Gaussian data, $\text{IQR}/(2\Phi^{-1}(0.75))$ ($\approx \text{IQR}/1.349$) is a robust estimate of the SD; a value well below the SD flags a skewed or heavy-tailed distribution.

Value

A single numeric value, the interquartile range.

See Also

[medianAbsDev](#) for the deviation-based robust scale, [signalMoments](#) for the classical moments.

Examples

```
interquartileRange(c(1, 2, 3, 4, 100)) # robust to the outlier
interquartileRange(rnorm(1000)) / 1.349 # ~1 (robust SD estimate)
```

intraclassCorrelation *Intraclass correlation coefficients (ICC)*

Description

The six McGraw & Wong (1996) intraclass correlation coefficients from a subjects-by-raters (or subjects-by-repeated-measurements) matrix: ICC1, ICC2, ICC3 (single measurement) and ICC1k, ICC2k, ICC3k (average of k measurements). The standard index of test-retest / inter-rater reliability, computed from a two-way ANOVA. Reproduces the psych package (psych: : ICC(lmer = FALSE)) to machine precision and pingouin.intraclass_corr bit-for-bit.

Usage

```
intraclassCorrelation(x)
```

Arguments

x A subjects-by-raters numeric matrix (rows = targets/subjects, columns = raters or repeated measurements). Rows with any missing value are dropped.

Value

A data frame with one row per ICC form (form, ICC), carrying n (subjects) and k (raters/measurements) as attributes.

Examples

```
m <- matrix(rnorm(30), ncol = 3) + rnorm(10)
intraclassCorrelation(m)
```

jarqueBeraTest *Jarque-Bera test for normality*

Description

Applies the Jarque-Bera goodness-of-fit test for normality (Jarque & Bera, 1980) to a one-dimensional signal: the statistic $JB = \frac{N}{6} \left(S^2 + \frac{K^2}{4} \right)$, where S is the skewness and K the excess kurtosis (from [signalMoments](#)), is compared to a chi-squared distribution with 2 degrees of freedom. It is the formal, moment-based counterpart of the descriptive near-Gaussian check: a large statistic rejects normality. Same definition as `scipy.stats.jarque_bera`.

Usage

```
jarqueBeraTest(x)
```

Arguments

x A numeric vector (the time series).

Value

A list with statistic (the JB statistic), p_value (the chi-squared upper-tail probability), and df (2).

References

Jarque, C.M., Bera, A.K. (1980). Efficient tests for normality, homoscedasticity and serial independence of regression residuals. *Economics Letters*, 6(3), 255-259.

See Also

[signalMoments](#)

Examples

```
set.seed(1)
jarqueBeraTest(rnorm(1000))$p_value    # Gaussian -> large p (fail to reject)
```

kendallTau

Kendall's tau-b rank correlation

Description

Kendall's tau-b, the concordance-based rank correlation: the normalised difference between concordant and discordant pairs, tie-corrected. It is the third rank-correlation method alongside Pearson's r ([correlationTest](#), method "pearson") and Spearman's rho (method "spearman"), and is the robust choice for ordinal or heavily-tied data and small samples. The tau-b statistic and the normal-approximation p-value reproduce base R `stats::cor.test(method = "kendall")` and `scipy.stats.kendalltau` bit-for-bit.

Usage

```
kendallTau(x, y = NULL)
```

Arguments

x Either the numeric predictor vector (with y the response), or a two-column matrix/data frame (column 1 = x, column 2 = y), or a list of two numeric vectors. Incomplete (x, y) pairs are dropped.

y The second numeric vector when x is a single vector.

Value

A list with statistic (tau-b), p_value (two-sided, normal-approximation), z (the approximation's z-statistic) and n.

See Also[correlationTest](#)**Examples**

```
kendallTau(c(1, 2, 3, 4, 5), c(2, 1, 4, 3, 5))
```

kpssTest	<i>KPSS test for stationarity</i>
----------	-----------------------------------

Description

Computes the KPSS statistic (Kwiatkowski, Phillips, Schmidt & Shin, 1992) for level stationarity of a one-dimensional signal. The residuals from regressing the signal on a constant are cumulatively summed, and the statistic is $\eta = N^{-2} \sum_t S_t^2 / \hat{\lambda}^2$, where S_t are the partial sums and $\hat{\lambda}^2$ is a Bartlett (Newey-West) long-run variance estimate with truncation lag. Unlike the Augmented Dickey-Fuller test, the KPSS *null* is stationarity: a *small* statistic (below the critical value) means the series is consistent with stationarity. The statistic is the same deterministic quantity as `statsmodels.tsa.stattools.kpss(regression = "c", nlags = lag)`.

Usage

```
kpssTest(x, lag = 10L)
```

Arguments

`x` A numeric vector (the time series).
`lag` Bartlett truncation lag L for the long-run variance (default 10).

Details

Note: only the *statistic* is computed here (an exact quantity). The KPSS critical values and p-value come from the reference tables (an approximation) and are not returned.

Value

A list with *statistic* (the KPSS η), *n* (the signal length), and *lag*.

References

Kwiatkowski, D., Phillips, P.C.B., Schmidt, P., Shin, Y. (1992). Testing the null hypothesis of stationarity against the alternative of a unit root. *Journal of Econometrics*, 54(1-3), 159-178.

See Also[adfTest](#), [autocorrelation](#)

Examples

```
set.seed(1)
kpssTest(rnorm(500), lag = 10)$statistic # stationary -> small statistic
```

kruskalTest	<i>Kruskal-Wallis rank-sum test (non-parametric between-groups omnibus)</i>
-------------	---

Description

The non-parametric between-groups omnibus test: pools all observations, ranks them, and tests whether the rank totals differ across $k \geq 2$ independent groups. It is the distribution-free counterpart of the one-way ANOVA ([oneWayAnova](#)) and, unlike it, makes no normality or equal-variance assumption. The tie-corrected statistic and its p-value reproduce base R `stats::kruskal.test` bit-for-bit (the H is accumulated in the same order) and `scipy.stats.kruskal` to machine precision.

Usage

```
kruskalTest(x, ...)
```

Arguments

<code>x</code>	Either a list of $k \geq 2$ numeric group vectors, or the first group's numeric vector (with further groups passed through <code>...</code>).
<code>...</code>	Additional numeric group vectors when <code>x</code> is a single vector.

Value

A list with `statistic` (the tie-corrected H), `p_value`, `df` ($k - 1$), `k` (number of groups) and `n` (total sample size).

See Also

[oneWayAnova](#), [friedmanTest](#)

Examples

```
kruskalTest(list(rnorm(20), rnorm(20, 1), rnorm(20, 2)))
```

ksNormalityTest

*Kolmogorov-Smirnov Test for Normality***Description**

Tests whether a one-dimensional signal is normally distributed by comparing its empirical cumulative distribution function (ECDF) to that of a normal distribution fitted to the sample (mean and SD). The statistic is the Kolmogorov-Smirnov distance $D = \sup_x |F_n(x) - \Phi((x - \mu)/\sigma)|$, the largest vertical gap between the two CDFs; the two-sided asymptotic (Kolmogorov) p-value follows. The D statistic reproduces `stats::ks.test` and `scipy.stats.kstest` bit-for-bit.

Usage

```
ksNormalityTest(x)
```

Arguments

`x` A numeric vector (the time series).

Details

A distribution-shape (empirical-CDF) normality test, complementary to the moment-based [jarqueBeraTest](#): KS responds to the overall shape of the distribution, Jarque-Bera to its skewness and kurtosis (the tails). The two can disagree – on a long, mildly non-Gaussian signal Jarque-Bera’s power often rejects while KS does not.

Value

A list with `statistic` (the KS distance D), `p_value` (two-sided asymptotic), and `n` (sample size).

Caveat

Because the normal parameters are estimated from the same sample, the naive p-value is anti-conservative (a Lilliefors correction gives an exact test); the D statistic itself is exact and is what this function certifies.

See Also

[jarqueBeraTest](#) for the moment-based normality test, [signalMoments](#) for the skewness and kurtosis.

Examples

```
set.seed(1); ksNormalityTest(rnorm(500))$p_value    # Gaussian -> large p
set.seed(1); ksNormalityTest(rexp(500))$statistic   # skewed -> large D
```

leveneTest	<i>Levene's test for homogeneity of variance</i>
------------	--

Description

Levene's test of whether $k \geq 2$ groups have equal variance, as a one-way ANOVA on the absolute deviations from each group's centre. With `center = "median"` (default) it is the robust Brown-Forsythe variant, appropriate when the data may be non-normal; `center = "mean"` is the classic Levene test. The F-statistic and p reproduce `scipy.stats.levene` and the `car` package (`car::leveneTest`) bit-for-bit.

Usage

```
leveneTest(x, ..., center = c("median", "mean"))
```

Arguments

<code>x</code>	Either a list of $k \geq 2$ numeric group vectors, or the first group's numeric vector (with further groups passed through <code>...</code>).
<code>...</code>	Additional numeric group vectors when <code>x</code> is a single vector.
<code>center</code>	"median" (default, Brown-Forsythe) or "mean" (classic Levene).

Value

A list with statistic (F), `p_value`, `df1` ($k - 1$), `df2` ($N - k$), `k`, `n` and `center`.

See Also

[oneWayAnova](#)

Examples

```
leveneTest(list(rnorm(20), rnorm(20, 0, 2), rnorm(20, 0, 0.5)))
```

linearRegression	<i>Simple linear regression (ordinary least squares)</i>
------------------	--

Description

Univariate OLS regression of `y` on `x`: the slope, intercept, R-squared, and the two-sided t-test of the slope. The slope, R-squared and p-value reproduce base R `stats::lm / summary.lm` and `scipy.stats.linregress` bit-for-bit (to machine precision against `lm`'s QR path).

Usage

```
linearRegression(x, y = NULL)
```

Arguments

x	Either the numeric predictor vector (with y the response), or a two-column matrix/data frame (column 1 = x, column 2 = y), or a list of two numeric vectors. Incomplete (x, y) pairs are dropped.
y	The numeric response vector when x is a single predictor vector.

Value

A list with slope, intercept, r_squared, statistic (the slope t-statistic), p_value (two-sided), df (n - 2) and n.

See Also

[correlationTest](#)

Examples

```
linearRegression(1:20, 2 * (1:20) + rnorm(20))
```

ljungBoxTest

Ljung-Box test for autocorrelation (white-noise / portmanteau test)

Description

Applies the Ljung-Box portmanteau test (Ljung & Box, 1978) to a one-dimensional signal: the statistic $Q = N(N + 2) \sum_{k=1}^h \hat{\rho}_k^2 / (N - k)$ aggregates the first lag squared autocorrelations (from [autocorrelation](#)), and under the null hypothesis that the series is white noise Q follows a chi-squared distribution with lag degrees of freedom. It is the standard formal test for the presence of autocorrelation (and, applied to model residuals, for model adequacy). Same definition as `statsmodels.stats.diagnostic.acorr_ljungbox` and `stats::Box.test(type = "Ljung-Box")`.

Usage

```
ljungBoxTest(x, lag = 10L)
```

Arguments

x	A numeric vector (the time series).
lag	Number of lags h to include (default 10), and the degrees of freedom of the reference chi-squared.

Value

A list with statistic (the Ljung-Box Q), p_value (the chi-squared upper-tail probability), and df (= lag).

References

Ljung, G.M., Box, G.E.P. (1978). On a measure of lack of fit in time series models. *Biometrika*, 65(2), 297-303.

See Also

[autocorrelation](#), [Box.test](#)

Examples

```
set.seed(1)
ljungBoxTest(rnorm(500), lag = 10)$p_value # white noise -> large p
```

localEfficiency	<i>Compute local efficiency</i>
-----------------	---------------------------------

Description

Calculates the local efficiency for each node.

Usage

```
localEfficiency(adjacency, weighted = FALSE)
```

Arguments

adjacency	An adjacency matrix.
weighted	If TRUE, uses weighted paths.

Value

A numeric vector of local efficiency values.

Examples

```
adj <- matrix(c(0, 1, 1, 1, 0, 1, 1, 1, 0), 3, 3)
localEfficiency(adj)
```

medianAbsDev

Median Absolute Deviation (robust dispersion)

Description

Computes the median absolute deviation (MAD) of a one-dimensional signal – $MAD = c \cdot \text{median}(|x - \text{median}(x)|)$ – a robust measure of dispersion that, unlike the standard deviation, is not inflated by a small number of extreme values (spikes, movement artifacts). With the default `constant = 1` it returns the raw MAD; set `constant = 1 / stats::qnorm(0.75)` (approximately 1.4826) to obtain the normal-consistent robust estimate of the standard deviation (the scale MAD and SD share for Gaussian data).

Usage

```
medianAbsDev(x, constant = 1)
```

Arguments

<code>x</code>	A numeric vector (the time series).
<code>constant</code>	Scale factor (default 1, the raw MAD). Use <code>1 / stats::qnorm(0.75)</code> for the normal-consistent robust SD estimate.

Details

The robust dispersion companion of [signalMoments](#) (whose SD is the classical, non-robust dispersion). Comparing the normal-consistent MAD with the SD is a quick artifact check: they agree for clean, near-Gaussian data and diverge (MAD-scale below SD) when heavy-tailed artifacts inflate the SD.

Value

A single numeric value, the (scaled) median absolute deviation.

See Also

[signalMoments](#) for the classical moments (mean, SD, skewness, kurtosis).

Examples

```
medianAbsDev(c(1, 2, 3, 4, 100)) # robust to the outlier
set.seed(1); medianAbsDev(rnorm(1000), 1 / stats::qnorm(0.75)) # ~1 (robust SD)
```

modularity	<i>Compute network modularity</i>
------------	-----------------------------------

Description

Calculates the modularity of a network given a community partition.

Usage

```
modularity(adjacency, communities)
```

Arguments

adjacency	An adjacency matrix.
communities	A vector of community assignments for each node.

Value

Modularity value (-0.5 to 1).

Examples

```
# Two clear communities
adj <- matrix(0, 6, 6)
adj[1:3, 1:3] <- 1
adj[4:6, 4:6] <- 1
diag(adj) <- 0
communities <- c(1, 1, 1, 2, 2, 2)
modularity(adj, communities)
```

multipleRegression	<i>Multiple linear regression (ordinary least squares)</i>
--------------------	--

Description

Multivariable OLS regression of a response on two or more predictors: the coefficient table (estimate, standard error, t-statistic, p-value) plus the overall model fit (R-squared, adjusted R-squared and the F-test). The many-predictor generalisation of [linearRegression](#). Computed via the same QR decomposition as base R, so the coefficients, R-squared and F reproduce stats::lm / summary.lm bit-for-bit and statsmodels OLS to machine precision.

Usage

```
multipleRegression(x)
```

Arguments

x A matrix or data frame whose first column is the response and whose remaining columns are the predictors. Incomplete rows are dropped.

Value

A data frame with one row per term (term, estimate, std_error, statistic, p_value), carrying r_squared, adj_r_squared, f_statistic, df1, df2, f_p_value and n as attributes.

See Also

[linearRegression](#), [partialCorrelation](#)

Examples

```
x <- matrix(rnorm(120), ncol = 3); colnames(x) <- c("y", "a", "b")
multipleRegression(x)
```

nemenyiTest	<i>Nemenyi post-hoc test (all pairwise comparisons after a Friedman test)</i>
-------------	---

Description

The Nemenyi test: all pairwise comparisons of the treatments following a Friedman repeated-measures test ([friedmanTest](#)), from the within-block mean ranks and the studentized-range distribution with single-step control of the family-wise error rate. It is the repeated-measures counterpart of Tukey's HSD ([tukeyHSD](#)) and Dunn's test ([dunnTest](#)). The statistics and p-values reproduce base R's Nemenyi formula, the PMCMRplus package and `scikit_posthocs.posthoc_nemenyi_friedman` to machine precision.

Usage

```
nemenyiTest(x)
```

Arguments

x A blocks-by-treatments numeric matrix (rows = subjects/blocks, columns = conditions/treatments); rows with any missing value are dropped.

Value

A data frame with one row per pair: comparison, statistic and p_adj, carrying k (treatments) and n (complete blocks) as attributes.

See Also

[friedmanTest](#), [tukeyHSD](#), [dunnTest](#)

Examples

```
m <- matrix(rnorm(40) + rep(c(0, 0.5, 1, 1.5), each = 10), nrow = 10)
colnames(m) <- c("a", "b", "c", "d"); nemenyiTest(m)
```

networkBasedStatistic *Network-Based Statistic (NBS)*

Description

Identifies connected subnetworks of edges that differ between two groups (or two within-subject conditions) of connectivity matrices, with family-wise error control from a permutation null on the largest suprathreshold component (Zalesky, Fornito & Bullmore 2010). Each edge is first tested with a mass-univariate t-test; edges whose statistic exceeds thresh form a suprathreshold graph whose connected components are the candidate subnetworks. Their size (edge count) or mass (summed statistic exceedance, Smith 2009) is compared against the permutation distribution of the largest component to obtain FWER-corrected component p-values.

Usage

```
networkBasedStatistic(
  mats_group1,
  mats_group2,
  thresh = 3,
  n_perm = 1000L,
  tail = c("both", "right", "left"),
  paired = FALSE,
  component = c("size", "mass"),
  directed = FALSE,
  alpha = 0.05,
  seed = NULL
)
```

Arguments

mats_group1	A list of n x n connectivity matrices or an n x n x N array (group 1, or condition 1 when paired).
mats_group2	The corresponding matrices for group 2 (or condition 2 for a paired design).
thresh	Primary edge-statistic (t) threshold (default: 3).
n_perm	Number of permutations (default: 1000).
tail	"both", "right" (group1 > group2), or "left".
paired	Logical; TRUE for a within-subject (paired) design, in which case the two inputs are paired condition matrices (default: FALSE).
component	Component measure: "size" (edge count) or "mass" (summed statistic exceedance).

directed	Logical; use all off-diagonal (directed) edges rather than the upper triangle (default: FALSE).
alpha	Significance level for the returned adjacency mask (default: 0.05).
seed	Optional RNG seed for reproducible permutations.

Value

A list with components (a data.frame of component size, mass, and p_value), component_edges (a list of node-pair matrices, one per component), the adjacency mask of edges in significant components, the edge_stats matrix, the suprathreshold mask, the permutation null_distribution, and the settings used.

References

Zalesky, A., Fornito, A., & Bullmore, E. T. (2010). Network-based statistic: identifying differences in brain networks. *NeuroImage*, 53(4), 1197-1207.

See Also

[plotNBSnetwork\(\)](#)

Examples

```
## Not run:
g1 <- replicate(20, matrix(rnorm(400), 20), simplify = FALSE)
g2 <- replicate(20, matrix(rnorm(400), 20), simplify = FALSE)
nbs <- networkBasedStatistic(g1, g2, thresh = 3, n_perm = 500)
nbs$components

## End(Not run)
```

nodeDegree	<i>Compute node degree</i>
------------	----------------------------

Description

Calculates the degree (number of connections) for each node.

Usage

```
nodeDegree(adjacency, weighted = FALSE, use_cpp = TRUE)
```

Arguments

adjacency	An adjacency matrix (binary or weighted).
weighted	If TRUE, returns weighted degree (strength).
use_cpp	If TRUE (default), use the compiled C++ backend, falling back to R if unavailable.

Value

A numeric vector of node degrees.

Examples

```
adj <- matrix(c(0, 1, 1, 1, 0, 0, 1, 0, 0), 3, 3)
nodeDegree(adj)
```

oneWayAnova

One-way ANOVA (omnibus F-test) across groups

Description

Classic one-way analysis of variance (equal-variance omnibus F-test) comparing the means of $k \geq 2$ independent groups. The F-statistic and its p-value reproduce base R `stats::oneway.test(var.equal = TRUE)` and `scipy.stats.f_oneway` bit-for-bit. This is the standard many-group generalisation of the independent two-sample t-test ([twoSampleTTest](#)).

Usage

```
oneWayAnova(x, ...)
```

Arguments

x Either a list of $k \geq 2$ numeric group vectors, or the first group's numeric vector (with further groups passed through `...`).

... Additional numeric group vectors when `x` is a single vector.

Value

A list with `statistic` (the F-ratio), `p_value`, `df1` (between-groups degrees of freedom, $k - 1$), `df2` (within-groups, $N - k$), `k` (number of groups) and `n` (total sample size).

See Also

[twoSampleTTest](#), [pairedTTest](#)

Examples

```
oneWayAnova(list(rnorm(20), rnorm(20, 1), rnorm(20, 2)))
```

pairedTTest	<i>Paired-Samples t-Test</i>
-------------	------------------------------

Description

Computes the two-sided paired-samples t-test comparing two matched vectors – the one-sample t-test of the within-pair differences $d = x - y$: $t = \bar{d}/(s_d/\sqrt{n})$ on $n - 1$ degrees of freedom. The standard test for before/after or condition-A/condition-B measurements on the same subjects. Reproduces `stats::t.test(x, y, paired = TRUE)` and `scipy.stats.ttest_rel` bit-for-bit.

Usage

```
pairedTTest(x, y = NULL)
```

Arguments

x	A numeric vector, OR (when y is NULL) a two-column matrix whose columns are the paired samples.
y	A numeric vector matched to x, or NULL if x is a two-column matrix.

Value

A list with `statistic` (the t-statistic of $x - y$), `p_value` (two-sided) and `df`.

See Also

[signalMoments](#), [jarqueBeraTest](#).

Examples

```
set.seed(1); pairedTTest(rnorm(20, 1), rnorm(20))$statistic # x shifted up -> positive t
```

partialAutocorrelation	<i>Partial autocorrelation function (PACF)</i>
------------------------	--

Description

Computes the sample partial autocorrelation function of a one-dimensional signal at lags 0, 1, ..., `lag_max`. The partial autocorrelation at lag k is the correlation between x_t and x_{t+k} with the linear effect of the intervening lags removed, obtained here by the Durbin-Levinson recursion on the biased, demeaned autocorrelation sequence (via [autocorrelation](#)) – the same definition as `stats::pacf` and `statsmodels.tsa.pacf(method = "ldb")`. Where the ACF *decays*, the PACF *cuts off* after the autoregressive order, so it is the standard tool for choosing AR model order.

Usage

```
partialAutocorrelation(x, lag_max = 20L)
```

Arguments

x	A numeric vector (the time series).
lag_max	Maximum lag (default 20); the returned vector has lag_max + 1 values (lags 0 through lag_max).

Value

A numeric vector of partial autocorrelation coefficients at lags 0:lag_max; the lag-0 value is 1 by convention and the lag-1 value equals the lag-1 autocorrelation.

References

Box, G.E.P., Jenkins, G.M. (1976). Time Series Analysis: Forecasting and Control. Holden-Day.

See Also

[autocorrelation](#), [pacf](#)

Examples

```
set.seed(1)
partialAutocorrelation(as.numeric(arima.sim(list(ar = 0.6), 500)), lag_max = 10)
```

partialCorrelation	<i>Partial correlation controlling for covariates</i>
--------------------	---

Description

The correlation between x and y after linearly removing one or more covariates z from both – the standard tool for asking whether an association survives adjustment for a confound. With method = "spearman" the variables are rank-transformed first (a rank / robust partial correlation). The coefficient reproduces base R's residual method and the ppcor package to machine precision.

Usage

```
partialCorrelation(x, y = NULL, z = NULL, method = c("pearson", "spearman"))
```

Arguments

x	Either the first numeric variable (with y and z given), or a matrix/data frame whose column 1 is x, column 2 is y and the remaining columns are the covariates. Incomplete rows are dropped.
y	The second numeric variable when x is a single vector.
z	The covariate(s) to control for: a numeric vector or matrix.
method	"pearson" (default) or "spearman" (rank-based).

Value

A list with `statistic` (the partial correlation), `p_value` (two-sided t-test), `df` ($n - 2 - k$), `n` and `k` (number of covariates).

See Also

[correlationTest](#), [kendallTau](#)

Examples

```
x <- rnorm(50); z <- rnorm(50); y <- 0.5 * z + rnorm(50)
partialCorrelation(x, y, z)
```

pathLength	<i>Compute shortest path length</i>
------------	-------------------------------------

Description

Calculates the shortest path length between all node pairs using Floyd-Warshall.

Usage

```
pathLength(adjacency, weighted = FALSE, use_cpp = TRUE)
```

Arguments

<code>adjacency</code>	An adjacency matrix.
<code>weighted</code>	If TRUE, uses edge weights as distances.
<code>use_cpp</code>	If TRUE (default), use the compiled C++ backend, falling back to R if unavailable.

Value

A matrix of shortest path lengths.

Examples

```
adj <- matrix(c(0, 1, 0, 1, 0, 1, 0, 1, 0), 3, 3)
pathLength(adj)
```

pli	<i>Compute Phase Lag Index (PLI)</i>
-----	--------------------------------------

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 <code>PhysioExperiment</code> 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.

$PLI = \text{lmean}(\text{sign}(\text{Im}(S_{xy})))$, 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))
```

plotAdjacencyMatrix	<i>Plot adjacency matrix heatmap</i>
---------------------	--------------------------------------

Description

Creates a heatmap visualization of an adjacency or connectivity matrix.

Usage

```
plotAdjacencyMatrix(
  adjacency,
  node_names = NULL,
  symmetric = TRUE,
  color_palette = "default",
  show_values = FALSE,
  title = "Connectivity Matrix"
)
```

Arguments

adjacency	An adjacency matrix or connectivity result.
node_names	Optional vector of node names.
symmetric	If TRUE, only shows lower triangle.
color_palette	Color palette: "default", "viridis", "heat", or custom.
show_values	If TRUE, displays values in cells.
title	Plot title.

Value

A ggplot object.

Examples

```
set.seed(123)
adj <- matrix(runif(16), 4, 4)
adj <- (adj + t(adj)) / 2
diag(adj) <- 1
plotAdjacencyMatrix(adj, node_names = c("Fz", "Cz", "Pz", "Oz"))
```

```
plotDynamicConnectivity
      Plot dynamic connectivity
```

Description

Creates a visualization of time-varying connectivity.

Usage

```
plotDynamicConnectivity(
  dyn_conn,
  node_pair = "mean",
  title = "Dynamic Connectivity"
)
```

Arguments

<code>dyn_conn</code>	Result from <code>slidingWindowConnectivity()</code> .
<code>node_pair</code>	Vector of two node indices to plot, or "mean" for average.
<code>title</code>	Plot title.

Value

A ggplot object.

Examples

```
set.seed(123)
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(1000 * 4), nrow = 1000, ncol = 4)),
  samplingRate = 100
)
dyn_conn <- slidingWindowConnectivity(pe, window_size = 200, step = 50)
plotDynamicConnectivity(dyn_conn, node_pair = c(1, 2))
```

plotERP	<i>Plot event-related potential (ERP) waveform</i>
---------	--

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	<i>Multi-channel visualization functions</i>
------------------	--

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](https://doi.org/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](https://doi.org/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")
```

plotNBSnetwork

Plot a significant NBS subnetwork

Description

Draws the edges belonging to significant NBS components on a circular node layout.

Usage

```
plotNBSnetwork(
  result,
  alpha = NULL,
  node_labels = NULL,
  title = "NBS subnetwork"
)
```

Arguments

result	A result from networkBasedStatistic() .
alpha	Significance level for the components to draw (default: uses the value stored in result).
node_labels	Optional character node labels.
title	Plot title.

Value

A ggplot object.

See Also

`networkBasedStatistic()`

Examples

```
## Not run:
plotNBSnetwork(networkBasedStatistic(g1, g2))

## End(Not run)
```

plotNetwork

Network Visualization for PhysioExperiment

Description

Functions for visualizing functional connectivity networks including network graphs, adjacency matrices, and network metrics. Plot network graph

Usage

```
plotNetwork(
  adjacency,
  node_names = NULL,
  node_size = "degree",
  edge_threshold = 0,
  layout = c("circle", "spring", "grid"),
  node_color = "#3498db",
  edge_color = "#7f8c8d",
  title = "Network Graph"
)
```

Arguments

<code>adjacency</code>	An adjacency matrix or connectivity result.
<code>node_names</code>	Optional vector of node names.
<code>node_size</code>	Node size. Can be "degree", "betweenness", "eigenvector", or numeric.
<code>edge_threshold</code>	Minimum edge weight to display.
<code>layout</code>	Layout algorithm: "circle", "spring", or "grid".
<code>node_color</code>	Node color or vector of colors.
<code>edge_color</code>	Edge color.
<code>title</code>	Plot title.

Details

Creates a network graph visualization with nodes and edges.

Value

A ggplot object.

Examples

```
set.seed(123)
adj <- matrix(runif(16), 4, 4)
adj <- (adj + t(adj)) / 2
diag(adj) <- 0
plotNetwork(adj, node_names = c("Fz", "Cz", "Pz", "Oz"))
```

plotNetworkMetrics	<i>Plot network metrics</i>
--------------------	-----------------------------

Description

Creates a bar plot of network metrics for each node.

Usage

```
plotNetworkMetrics(
  adjacency,
  metrics = c("degree", "clustering", "betweenness"),
  node_names = NULL,
  title = "Network Metrics"
)
```

Arguments

adjacency	An adjacency matrix.
metrics	Vector of metrics to compute: "degree", "clustering", "betweenness", "eigenvector", "local_efficiency".
node_names	Optional vector of node names.
title	Plot title.

Value

A ggplot object.

Examples

```
adj <- matrix(c(0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0), 4, 4)
plotNetworkMetrics(adj, node_names = c("Fz", "Cz", "Pz", "Oz"))
```

`plotNetworkStability` *Plot network stability over time*

Description

Visualizes the temporal stability of network topology.

Usage

```
plotNetworkStability(stability, title = "Network Stability")
```

Arguments

<code>stability</code>	Result from <code>temporalStability()</code> .
<code>title</code>	Plot title.

Value

A ggplot object.

Examples

```
set.seed(123)
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(1000 * 4), nrow = 1000, ncol = 4)),
  samplingRate = 100
)
dyn_conn <- slidingWindowConnectivity(pe, window_size = 200, step = 50)
stab <- temporalStability(dyn_conn)
plotNetworkStability(stab)
```

`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	<i>Plot a signal channel</i>
------------	------------------------------

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](https://doi.org/10.1007/9783319242774)

See Also

[plotMultiChannel\(\)](#) for multi-channel visualization, [plotPSD\(\)](#) for power spectral density plots, [plotERP\(\)](#) for event-related potential plots.

plotSpecparam	<i>Plot a specparam model fit</i>
---------------	-----------------------------------

Description

Plots the observed log power spectrum for one channel together with the full specparam model and the aperiodic-only fit.

Usage

```
plotSpecparam(result, channel = 1)
```

Arguments

result	A specparam_result from specparam() .
channel	Channel index or label to plot (default: 1).

Value

A ggplot object.

See Also

[specparam\(\)](#)

Examples

```
## Not run:
sp <- specparam(make_pe_2d(n_time = 4000, sr = 250))
plotSpecparam(sp, channel = 1)

## End(Not run)
```

plotSpectrogram	<i>Plot spectrogram</i>
-----------------	-------------------------

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))
```

plotSPM

*Plot SPM result***Description**

Visualizes SPM analysis results with significance thresholds.

Usage

```
plotSPM(
  x,
  time_axis = NULL,
  show_threshold = TRUE,
  show_clusters = TRUE,
  title = NULL
)
```

Arguments

<code>x</code>	An <code>spm_result</code> object.
<code>time_axis</code>	Optional time axis values.
<code>show_threshold</code>	Logical; if TRUE, shows significance threshold.
<code>show_clusters</code>	Logical; if TRUE, highlights significant clusters.
<code>title</code>	Plot title.

Value

A ggplot object.

Examples

```
# Create and plot SPM result
set.seed(123)
g1 <- matrix(rnorm(100 * 10), nrow = 100)
g2 <- matrix(rnorm(100 * 10), nrow = 100)
g2[40:60, ] <- g2[40:60, ] + 1.5

pe <- PhysioExperiment(
  assays = list(values = cbind(g1, g2)),
  samplingRate = 100
)
result <- spmTTest(pe, group1 = 1:10, group2 = 11:20)
plotSPM(result)
```

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,
  interpolation = c("idw", "spline"),
  spline_stiffness = 4L,
  spline_terms = 50L,
  spline_regularization = 0
)
```

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.

interpolation Interpolation method. "idw" preserves the Shepard inverse-distance-weighted default; "spline" uses Perrin spherical splines on an upper-hemisphere lift.

spline_stiffness Positive integer of at least 2 controlling the spherical-spline kernel. Default is 4.

spline_terms Positive integer number of Legendre terms. Default is 50.

spline_regularization Non-negative diagonal regularization applied to the electrode kernel. The default 0 gives an interpolating spline; positive values improve conditioning but need not reproduce electrode values.

Details

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

With `interpolation = "idw"`, scalp values are interpolated by inverse distance weighting (Shepard's method) with power 2. With `interpolation = "spline"`, planar montage coordinates are lifted to a shared upper unit hemisphere and evaluated with the Perrin spherical-spline kernel. This is spatial interpolation, not a surface Laplacian, current-source-density estimate, reference transformation, or source localization.

Value

A ggplot object.

References

Shepard, D. (1968). "A two-dimensional interpolation function for irregularly-spaced data." *Proceedings of the 1968 23rd ACM National Conference*, 517-524. doi:10.1145/800186.810616

Perrin F, Pernier J, Bertrand O, Echallier J. (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	<i>Plot topographic map animation</i>
-------------------	---------------------------------------

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() , including interpolation and the spherical-spline controls.

Details

The interpolation method and controls are forwarded unchanged to [plotTopomap\(\)](#).

Value

A list of ggplot objects.

References

Shepard, D. (1968). "A two-dimensional interpolation function for irregularly-spaced data." *Proceedings of the 1968 23rd ACM National Conference*, 517-524. doi:[10.1145/800186.810616](#)

Perrin F, Pernier J, Bertrand O, Echallier J. (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

```

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))

```

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))
```

rankBiserial

*Rank-biserial correlation for two independent samples***Description**

The rank-biserial correlation derived from the Mann-Whitney U statistic, equal to $2*U1/(n1*n2) - 1$ and to $1 - 2*U2/(n1*n2)$ (Kerby 2014), where $U1$ counts the pairs in which the first sample exceeds the second. For two independent samples this equals Cliff's delta; the confidence interval is Cliff's (1993) consistent-variance interval.

Usage

```
rankBiserial(x, y, conf_level = 0.95)
```

Arguments

`x, y` Numeric vectors for the two samples.
`conf_level` Confidence level (default 0.95).

Value

A list with `r` (rank-biserial correlation), `ci_lower`, `ci_upper`, the Mann-Whitney `u` for `x`, and the sample sizes.

References

Kerby, D.S. (2014). Compr Psychol 3:11.IT.3.1. Cliff, N. (1993). Psychol Bull 114(3):494-509.

See Also

[cliffsDelta\(\)](#), [effectSize\(\)](#)

Examples

```
rankBiserial(c(5, 6, 7, 8), c(1, 2, 3, 9))
```

```
scalarOnFunctionRegression
```

Scalar-on-function regression (functional coefficient)

Description

Predicts a scalar outcome from a predictor CURVE (e.g. a clinical score from a joint-moment waveform). The curve is reduced to functional principal components, the outcome is regressed on the scores, and the coefficients are mapped back to a coefficient function $\beta(t)$: where along the curve the predictor matters.

Usage

```
scalarOnFunctionRegression(y, curves, npc = NULL, ve = 0.95)
```

Arguments

y	Length-N numeric outcome.
curves	N x P matrix of predictor curves.
npc	Number of functional principal components (default: enough to explain ve variance).
ve	Variance-explained target for choosing npc when npc is NULL (default 0.95).

Value

a sofr_result: beta (length-P coefficient function), intercept, fitted, r_squared, npc, scores, pve (proportion variance explained by the retained PCs).

References

Reiss PT, Ogden RT (2007) scalar-on-function regression; Ramsay & Silverman (2005).

See Also

[functionalRegression\(\)](#)

Examples

```
set.seed(2)
P <- 80; t <- seq(0, 1, length.out = P)
X <- matrix(rnorm(50 * P), 50, P) + outer(rnorm(50), sin(2 * pi * t))
beta_true <- dnorm(t, 0.3, 0.05)
y <- as.numeric(scale(X %*% beta_true)) + rnorm(50, 0, 0.3)
m <- scalarOnFunctionRegression(y, X, npc = 5)
m$r_squared
```

signalMoments	<i>Signal distribution moments (mean, SD, skewness, kurtosis)</i>
---------------	---

Description

Computes the first four moments of a one-dimensional signal's amplitude distribution: the mean, the (population) standard deviation, the skewness, and the excess kurtosis. Skewness measures asymmetry (negative = a longer left tail); excess kurtosis measures tailedness relative to a Gaussian (0 = Gaussian, positive = heavier tails / more peaked). These are the standard shape descriptors used, for example, to flag EEG artifacts (high kurtosis) or characterize amplitude asymmetry. Definitions match `scipy.stats.skew` (biased) and `scipy.stats.kurtosis` (Fisher / excess, biased), with the population standard deviation (`ddof = 0`).

Usage

```
signalMoments(x)
```

Arguments

`x` A numeric vector (the time series).

Value

A list with mean, sd (population), skewness, and kurtosis (excess).

References

Joanes, D.N., Gill, C.A. (1998). Comparing measures of sample skewness and kurtosis. *The Statistician*, 47(1), 183-189.

See Also

[sd](#)

Examples

```
set.seed(1)
signalMoments(rnorm(1000))    # ~ 0 skewness, ~ 0 excess kurtosis for Gaussian
```

```
slidingWindowConnectivity
```

Sliding window connectivity

Description

Computes connectivity matrices over sliding time windows.

Usage

```
slidingWindowConnectivity(
  x,
  window_size = 256L,
  step = 64L,
  method = c("correlation", "coherence", "plv"),
  ...
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object.
<code>window_size</code>	Window size in samples.
<code>step</code>	Step size in samples.
<code>method</code>	Connectivity method: "correlation", "coherence", or "plv".
<code>...</code>	Additional arguments passed to connectivity function.

Value

A list with time-varying connectivity matrices.

Examples

```
set.seed(123)
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(1000 * 4), nrow = 1000, ncol = 4)),
  samplingRate = 100
)
dyn_conn <- slidingWindowConnectivity(pe, window_size = 200, step = 50)
```

smallWorldness	<i>Compute small-worldness</i>
----------------	--------------------------------

Description

Calculates the small-world coefficient (sigma) of a network.

Usage

```
smallWorldness(adjacency, n_rand = 100, n_cores = NULL)
```

Arguments

adjacency	An adjacency matrix.
n_rand	Number of random networks for comparison.
n_cores	Number of cores for parallel processing. Default NULL uses sequential processing.

Value

A list with small-worldness metrics.

Examples

```
# Create a small-world-like network
set.seed(123)
n <- 20
adj <- matrix(0, n, n)
for (i in 1:n) {
  adj[i, ((i) %% n) + 1] <- 1
  adj[i, ((i + 1) %% n) + 1] <- 1
}
adj <- adj + t(adj)
adj[adj > 0] <- 1
# Add some random long-range connections
adj[sample(which(adj == 0 & row(adj) < col(adj)), 5)] <- 1
adj <- adj + t(adj)
adj[adj > 0] <- 1
diag(adj) <- 0
sw <- smallWorldness(adj, n_rand = 10)
```

specparam

*Spectral parameterization (specparam / FOOOF)***Description**

Parameterizes each channel's power spectrum into an aperiodic 1/f component (offset, exponent, and an optional knee) plus a set of Gaussian periodic peaks, following the specparam / FOOOF algorithm (Donoghue et al. 2020). The aperiodic exponent is a marker of excitation/inhibition balance and cortical state; the peaks capture genuine oscillations separated from the 1/f background. The fit runs on the Welch PSD computed by the same machinery as `bandPower()`.

Usage

```
specparam(
  x,
  freq_range = c(1, 45),
  peak_width_limits = c(1, 12),
  max_n_peaks = 6L,
  aperiodic_mode = c("fixed", "knee"),
  min_peak_height = 0.05,
  peak_threshold = 2,
  nperseg = NULL,
  assay_name = NULL
)

## S3 method for class 'specparam_result'
print(x, ...)
```

Arguments

<code>x</code>	A <code>specparam_result</code> .
<code>freq_range</code>	Numeric length-2 frequency range in Hz to fit (default: <code>c(1, 45)</code>).
<code>peak_width_limits</code>	Numeric length-2 minimum and maximum peak bandwidth in Hz (default: <code>c(1, 12)</code>).
<code>max_n_peaks</code>	Maximum number of periodic peaks to fit (default: 6).
<code>aperiodic_mode</code>	"fixed" for a pure 1/f (offset, exponent) or "knee" to also fit a knee frequency.
<code>min_peak_height</code>	Minimum peak height above the aperiodic fit, in log10 power (default: 0.05).
<code>peak_threshold</code>	Minimum peak height in units of the standard deviation of the flattened spectrum (default: 2).
<code>nperseg</code>	Welch segment length in samples (default: about a 2-second segment for adequate low-frequency resolution).
<code>assay_name</code>	Input assay name (default: the default assay).
<code>...</code>	Ignored.

Value

An object of class "specparam_result": a list with aperiodic (a data.frame of per-channel offset, exponent, and knee), peaks (a data.frame of per-channel center frequency CF, power PW, and bandwidth BW), fit (per-channel R-squared and error), spectra (per-channel log spectra and fitted model for plotting), and the settings used.

References

Donoghue, T., et al. (2020). Parameterizing neural power spectra into periodic and aperiodic components. *Nature Neuroscience*, 23(12), 1655-1665.

See Also

[plotSpecparam\(\)](#), [specparamBiomarker\(\)](#), [bandPower\(\)](#)

Examples

```
## Not run:
pe <- make_pe_2d(n_time = 4000, n_channels = 4, sr = 250)
sp <- specparam(pe, freq_range = c(1, 40))
sp$aperiodic

## End(Not run)
```

specparamBiomarker	<i>Aperiodic parameter as a PhysioBiomarker</i>
--------------------	---

Description

Wraps a specparam aperiodic parameter (by default the exponent) for one channel as a reliability-characterised [PhysioBiomarker](#).

Usage

```
specparamBiomarker(
  result,
  channel = 1,
  param = c("exponent", "offset", "knee"),
  reliability = list(icc = NA_real_, sem = NA_real_)
)
```

Arguments

result	A specparam_result from specparam() .
channel	Channel index or label (default: 1).
param	Which aperiodic parameter to export: "exponent", "offset", or "knee".
reliability	Named list of reliability indices to attach (default: icc and sem placeholders).

Value

A [PhysioBiomarker](#).

See Also

[specparam\(\)](#), [PhysioExperiment::physioBiomarker\(\)](#)

Examples

```
## Not run:
sp <- specparam(make_pe_2d(n_time = 4000, sr = 250))
specparamBiomarker(sp, channel = 1)

## End(Not run)
```

spectralClustering	<i>Spectral clustering of network nodes</i>
--------------------	---

Description

Performs spectral clustering on the network.

Usage

```
spectralClustering(adjacency, n_clusters = 2, normalized = TRUE)
```

Arguments

adjacency	An adjacency matrix.
n_clusters	Number of clusters.
normalized	If TRUE, uses normalized Laplacian.

Value

A list with cluster assignments and spectral embedding.

Examples

```
# Create block-structured network
adj <- matrix(0, 6, 6)
adj[1:3, 1:3] <- 1
adj[4:6, 4:6] <- 1
adj[3, 4] <- adj[4, 3] <- 0.5
diag(adj) <- 0
clusters <- spectralClustering(adj, n_clusters = 2)
```

spectralDecomposition *Spectral decomposition of graph Laplacian*

Description

Computes eigenvalues and eigenvectors of the graph Laplacian.

Usage

```
spectralDecomposition(adjacency, normalized = TRUE, n_components = NULL)
```

Arguments

adjacency An adjacency matrix.
normalized If TRUE, uses normalized Laplacian.
n_components Number of components to return. If NULL, returns all.

Value

A list with eigenvalues and eigenvectors.

Examples

```
adj <- matrix(c(0, 1, 1, 1, 0, 1, 1, 1, 0), 3, 3)
spec <- spectralDecomposition(adj)
```

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

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

Details

Computes the spectrogram using STFT with proper power spectral density normalization. The returned power values are one-sided PSD estimates in V^2/Hz (non-DC/Nyquist bins are doubled).

Value

A list with the following components:

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

spmAnova	<i>SPM ANOVA (F-test)</i>
----------	---------------------------

Description

Performs SPM F-test for comparing multiple groups.

Usage

```
spmAnova(x, groups, alpha = 0.05)
```

Arguments

x	A <code>PhysioExperiment</code> object or matrix (time x observations).
groups	Factor or list indicating group membership.
alpha	Significance level.

Value

A list of class "spm_result" containing F-statistics and clusters.

Examples

```
# Three-group comparison
set.seed(123)
g1 <- matrix(rnorm(100 * 8), nrow = 100)
g2 <- matrix(rnorm(100 * 8), nrow = 100) + 0.5
g3 <- matrix(rnorm(100 * 8), nrow = 100) + 1.0

data <- cbind(g1, g2, g3)
groups <- factor(rep(c("A", "B", "C"), each = 8))

pe <- PhysioExperiment(assays = list(values = data), samplingRate = 100)
result <- spmAnova(pe, groups = groups)
```

spmMANOVA	<i>SPM one-way MANOVA (Hotelling's T^2 / chi-square vector field)</i>
-----------	--

Description

Computes a multivariate SPM field for a one-way MANOVA at every node, for vector-valued waveforms (e.g. 3D joint angles). With two groups the field is Hotelling's T^2 (reducing to the squared two-sample t-field for a single component); with more than two groups it is the chi-square field of the Bartlett-transformed Wilks' Λ . Field-level significance uses random-field theory (an F-field threshold for T^2 , a chi-square-field threshold for the X^2 field).

Usage

```
spmMANOVA(x, groups, vector_components = NULL, alpha = 0.05)
```

Arguments

- x A 3D array time x obs x component, or a list of component matrices (each time x obs), or a single time x obs matrix (one component).
- groups A grouping factor, one value per observation.
- vector_components Optional integer; only used to disambiguate a 2D matrix input (defaults to a single component).
- alpha Significance level (default 0.05).

Value

A list of class "spm_result" (test_type = "manova") with the statistic field (T2 or X2), RFT threshold, significant clusters, p_values, degrees of freedom, fwhm, and resel_count.

References

Pataky 2016 (vector-field 1D SPM); Worsley 1994 (chi-square RFT). spm1d.stats.manova1 / hotellings2.

See Also

```
spmTTest(), spmRegression(), spmSnPM()
```

Examples

```
set.seed(1)
# two groups, 3 components (e.g. hip flexion/abduction/rotation)
arr <- array(rnorm(50 * 16 * 3), c(50, 16, 3))
arr[20:30, 1:8, ] <- arr[20:30, 1:8, ] + 1.2
spmMANOVA(arr, groups = rep(c("A", "B"), each = 8))
```

spmPairedTTest	<i>SPM paired t-test</i>
----------------	--------------------------

Description

Performs SPM analysis for paired/repeated measures data.

Usage

```
spmPairedTTest(x, condition1, condition2, alpha = 0.05, two_tailed = TRUE)
```

Arguments

x	A <code>PhysioExperiment</code> object or matrix (time x observations).
condition1	Indices for first condition.
condition2	Indices for second condition.
alpha	Significance level.
two_tailed	Logical; if TRUE, performs two-tailed test.

Value

A list of class "spm_result".

Examples

```
# Pre-post intervention comparison
set.seed(123)
pre <- matrix(rnorm(100 * 15), nrow = 100)
post <- pre + 0.8 # Effect across all time points
post[30:50, ] <- post[30:50, ] + 0.5 # Additional effect

data <- cbind(pre, post)
pe <- PhysioExperiment(assays = list(values = data), samplingRate = 100)

result <- spmPairedTTest(pe, condition1 = 1:15, condition2 = 16:30)
```

spmRegression	<i>SPM linear-regression t-field</i>
---------------	--------------------------------------

Description

Computes an $SPM\{t\}$ field testing, at every node, the slope of a linear regression of the waveform on a continuous predictor. The node-wise statistic is the ordinary least-squares slope t-value (identical to `summary(lm(y ~ predictor))` at that node), and field-level significance uses the same random-field-theory threshold as `spmTTest()` with $df = n - 2$.

Usage

```
spmRegression(x, predictor, alpha = 0.05, two_tailed = TRUE)
```

Arguments

x	A <code>PhysioExperiment</code> or a numeric matrix (time x observations).
predictor	Numeric covariate, one value per observation (column).
alpha	Significance level (default 0.05).
two_tailed	Logical; two-tailed slope test (default TRUE).

Value

A list of class "spm_result" (test_type = "regression") with the t field, RFT threshold, significant clusters, pointwise p_values, df, fwhm, resel_count, and the fitted slope/intercept fields.

References

Pataky 2016; Friston et al. 2007 (RFT). spm1d.stats.regress.

See Also

[spmTTest\(\)](#), [spmMANOVA\(\)](#), [spmSnPM\(\)](#)

Examples

```
set.seed(1)
pred <- rnorm(20)
data <- matrix(rnorm(100 * 20), nrow = 100)
data[40:60, ] <- data[40:60, ] + outer(rep(1.5, 21), pred) # slope 30% window
spmRegression(data, pred)
```

 spmSnPM

Non-parametric (permutation) SPM

Description

Builds the permutation distribution of the field-maximum statistic to obtain a distribution-free critical threshold with strong FWER control (Nichols & Holmes 2002), plus permutation cluster- and set-level p-values. This is the validity cross-check for the parametric RFT thresholds: as the field smoothness grows the permutation threshold converges to the RFT threshold.

Usage

```
spmSnPM(
  x,
  groups = NULL,
  group1 = NULL,
  group2 = NULL,
  statistic = c("t", "F", "T2"),
  n_permutations = 1000,
  alpha = 0.05,
  two_tailed = TRUE,
  vector_components = NULL,
  seed = NULL
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> /matrix (time x obs) for t/F, or a 3D array / list of component matrices for T2.
<code>groups</code>	Grouping factor (one per observation) for F or T2, or a two-sample t.
<code>group1, group2</code>	Observation indices for a two-sample t (<code>group2 = NULL</code> gives a one-sample sign-flip test).
<code>statistic</code>	"t", "F", or "T2".
<code>n_permutations</code>	Number of random permutations (default 1000).
<code>alpha</code>	Significance level (default 0.05).
<code>two_tailed</code>	For t: use the two-tailed field maximum $\max t $.
<code>vector_components</code>	Passed to the T2 array coercion.
<code>seed</code>	Optional RNG seed for reproducibility.

Value

A list of class "spm_result" (`test_type = "snpm"`) with the observed statistic field, the permutation threshold, clusters with permutation p-values, pointwise permutation p-values, and the `perm_max` distribution.

References

Nichols & Holmes 2002; Pataky 2016. `spm1d.stats.nonparam`.

See Also

[spmTTest\(\)](#), [spmAnova\(\)](#), [spmMANOVA\(\)](#)

Examples

```
set.seed(1)
g1 <- matrix(rnorm(50 * 10), 50); g2 <- matrix(rnorm(50 * 10), 50)
g2[20:30, ] <- g2[20:30, ] + 1.5
spmSnPM(cbind(g1, g2), group1 = 1:10, group2 = 11:20,
         n_permutations = 200, seed = 1)
```

Description

Functions for statistical analysis of continuous waveform data using Statistical Parametric Mapping (SPM) methodology adapted from neuroimaging. These methods test hypotheses over entire waveforms rather than discrete points. SPM t-test for waveform comparison

Usage

```
spmTTest(x, group1 = NULL, group2 = NULL, alpha = 0.05, two_tailed = TRUE)
```

Arguments

x	A <code>PhysioExperiment</code> object or matrix (time x observations).
group1	Indices for first group (for two-sample test).
group2	Indices for second group. If <code>NULL</code> , performs one-sample test.
alpha	Significance level (default: 0.05).
two_tailed	Logical; if <code>TRUE</code> , performs two-tailed test.

Details

Performs a t-test at each time point and computes an SPM t-statistic map with Random Field Theory (RFT) correction for multiple comparisons.

SPM analyzes continuous biomechanical waveforms (e.g., joint angles, moments) by computing t-statistics at each time point and using Random Field Theory to control family-wise error rate across the entire waveform.

Value

A list of class "spm_result" containing:

t	T-statistic at each time point
threshold	Critical threshold from RFT
clusters	Significant clusters (start, end, extent, p-value)
p_values	Pointwise p-values
alpha	Significance level used

References

Pataky TC (2012). One-dimensional statistical parametric mapping in Python. *Computer Methods in Biomechanics and Biomedical Engineering*.

Examples

```
# Create example gait data (100 time points x 20 subjects)
set.seed(123)
# Group 1: normal gait
g1 <- matrix(rnorm(100 * 10), nrow = 100, ncol = 10)
# Group 2: altered gait (effect at 40-60% of cycle)
g2 <- matrix(rnorm(100 * 10), nrow = 100, ncol = 10)
g2[40:60, ] <- g2[40:60, ] + 1.5

data <- cbind(g1, g2)
pe <- PhysioExperiment(
  assays = list(values = data),
  samplingRate = 100
```

```
)

# Two-sample SPM t-test
result <- spmTTest(pe, group1 = 1:10, group2 = 11:20)
print(result)
```

temporalStability	<i>Compute temporal stability of network</i>
-------------------	--

Description

Calculates how stable the network topology is over time.

Usage

```
temporalStability(dyn_conn, metric = c("correlation", "distance"))
```

Arguments

dyn_conn	Result from slidingWindowConnectivity().
metric	Stability metric: "correlation" or "distance".

Value

A list with stability metrics.

Examples

```
set.seed(123)
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(1000 * 4), nrow = 1000, ncol = 4)),
  samplingRate = 100
)
dyn_conn <- slidingWindowConnectivity(pe, window_size = 200, step = 50)
stability <- temporalStability(dyn_conn)
```

tfce	<i>Threshold-Free Cluster Enhancement (TFCE)</i>
------	--

Description

Enhances a statistic map by integrating, at every point, the extent of the suprathreshold cluster containing that point raised to the power E times the threshold height raised to the power H , over all thresholds (Smith & Nichols 2009). This rewards signal that is either spatially broad or locally high without committing to a single cluster-forming threshold. Works on a time-by-channel (or time-by-frequency) matrix, or a 1-D vector, with time (and channel) adjacency.

Usage

```
tfce(stat_map, E = 0.5, H = 2, dh = NULL, adjacency = NULL, tail = 0)
```

Arguments

stat_map	A numeric matrix (time x channel) or a numeric vector.
E	Extent exponent (default: 0.5).
H	Height exponent (default: 2).
dh	Threshold step. NULL (default) integrates exactly over the distinct statistic levels; a numeric value uses a regular grid.
adjacency	Optional channel-adjacency matrix (columns of stat_map). NULL uses neighbouring-column (4-connectivity) adjacency.
tail	1 for positive effects, -1 for negative, or 0 (default) for a signed two-sided map (positive TFCE minus negative TFCE).

Value

A TFCE map with the same shape as stat_map.

References

Smith, S. M., & Nichols, T. E. (2009). Threshold-free cluster enhancement: addressing problems of smoothing, threshold dependence and localisation in cluster inference. *NeuroImage*, 44(1), 83-98.

See Also

[clusterPermutationTest\(\)](#)

Examples

```
m <- matrix(0, 100, 1); m[20:80, 1] <- 2; m[90:93, 1] <- 6
tf <- tfce(m)
c(broad = tf[50, 1], narrow = tf[91, 1])
```

thresholdNetwork

Threshold network by density

Description

Thresholds a connectivity matrix to achieve a target network density.

Usage

```
thresholdNetwork(connectivity, density = 0.2, absolute = TRUE)
```

Arguments

connectivity	A connectivity matrix.
density	Target density (proportion of edges to keep, 0-1).
absolute	If TRUE, uses absolute values for ranking.

Value

A thresholded adjacency matrix.

Examples

```
set.seed(123)
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(500 * 4), nrow = 500, ncol = 4)),
  samplingRate = 100
)
conn <- correlationMatrix(pe)
# Keep top 30% of connections
adj <- thresholdNetwork(conn, density = 0.3)
```

trimmedMean	<i>Trimmed Mean (robust location)</i>
-------------	---------------------------------------

Description

Computes the trimmed mean of a one-dimensional signal – the arithmetic mean after discarding the most extreme `trim` fraction of the values from each end. A robust measure of central tendency that, unlike the ordinary mean, is not pulled by a heavy tail or a few outliers, while (unlike the median) it still uses all of the retained data. It matches base R `mean(x, trim = ...)` and `scipy.stats.trim_mean` bit-for-bit, both of which discard $\lfloor n \text{ trim} \rfloor$ values from each end.

Usage

```
trimmedMean(x, trim = 0.1)
```

Arguments

x	A numeric vector (the time series).
trim	Fraction (0 to 0.5) of values trimmed from EACH end (default 0.1).

Details

A robust location estimator that completes the robust toolkit alongside [medianAbsDev](#) and [interquartileRange](#) (robust dispersion). On a skewed distribution the trimmed mean lies between the ordinary mean and the median, and moves toward the median as `trim` increases – a measure of how much the tail inflates the mean.

Value

A single numeric value, the trimmed mean.

See Also

[signalMoments](#) for the ordinary mean and SD, [medianAbsDev](#) and [interquartileRange](#) for robust dispersion.

Examples

```
trimmedMean(c(1, 2, 3, 4, 100), trim = 0.2) # 3 (robust); the mean is 22
set.seed(1); trimmedMean(rexp(1000), trim = 0.2) # below the tail-inflated mean
```

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:

<code>t_values</code>	Matrix of t-statistics (time x channel)
<code>p_values</code>	Matrix of p-values (time x channel)
<code>df</code>	Degrees of freedom
<code>n1, n2</code>	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(epochs = 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)
```

tukeyHSD	<i>Tukey HSD post-hoc test (all pairwise comparisons after one-way ANOVA)</i>
----------	---

Description

Tukey's Honest Significant Difference test: all pairwise mean comparisons following a one-way ANOVA ([oneWayAnova](#)), with p-values and confidence intervals from the studentized-range distribution and single-step control of the family-wise error rate. Reproduces base R `stats::TukeyHSD` and `scipy.stats.tukey_hsd` (machine precision).

Usage

```
tukeyHSD(x, ..., conf_level = 0.95)
```

Arguments

- x Either a named list of $k \geq 2$ numeric group vectors, or the first group’s numeric vector (with further groups passed through ... – use a named list to label the comparisons).
- ... Additional numeric group vectors when x is a single vector.
- conf_level Confidence level for the intervals (default 0.95).

Value

A data frame with one row per pair: comparison, diff (mean difference), lwr/upr (CI) and p_adj (family-wise-adjusted p-value), carrying k, df and n as attributes.

See Also

[oneWayAnova](#)

Examples

```
tukeyHSD(list(a = rnorm(20), b = rnorm(20, 1), c = rnorm(20, 2)))
```

twoSampleTTest	<i>Two-Sample t-Test</i>
----------------	--------------------------

Description

Computes the two-sided independent two-sample t-test comparing two groups. By default it is Welch’s t-test (unequal variances, Welch-Satterthwaite df) – matching `stats::t.test` and `scipy.stats.ttest_ind(equal = FALSE)`; set `var_equal = TRUE` for Student’s pooled-variance test. The standard test for a difference between two independent groups (e.g. older vs younger, patients vs controls). Reproduces both references bit-for-bit.

Usage

```
twoSampleTTest(x, y = NULL, var_equal = FALSE)
```

Arguments

- x A numeric vector (group 1), OR (when y is NULL) a list of two numeric vectors.
- y A numeric vector (group 2), or NULL if x is a two-element list.
- var_equal Logical; FALSE (default) for Welch’s t-test, TRUE for Student’s pooled-variance t-test.

Value

A list with statistic (the t-statistic of $x - y$), p_value (two-sided) and df.

See Also

[pairedTTest](#) for matched samples.

Examples

```
set.seed(1); twoSampleTTest(rnorm(20, 1), rnorm(30))$statistic # group 1 higher -> positive t
```

waveletTransform	<i>Wavelet transform</i>
------------------	--------------------------

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,
  normalization = c("L2", "L1")
)
```

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).
normalization	Wavelet normalization method: "L2" (default, divides by square root of sum of squared absolute values) or "L1" (divides by sum of absolute values). L2 normalization preserves energy across frequencies and is preferred for power comparisons.

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	<i>Compute weighted Phase Lag Index (wPLI)</i>
------	--

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 <code>PhysioExperiment</code> 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

acf, 9
adfTest, 4, 43
adjacencyMatrix, 5
anovaEpochs, 6
anovaEpochs(), 95
ar.yw, 8
arSpectrum, 7
arYuleWalker, 7, 8, 8
autocorrelation, 5, 8, 9, 43, 47, 48, 55, 56
averageEpochs, 10
averageEpochs(), 28, 30, 36

bandPower, 10
bandPower(), 30, 67, 80, 81, 84, 98
betweennessCentrality, 11
binarizeNetwork, 12
bootstrapCI, 13
bootstrapCI(), 26, 61
Box.test, 48

cliffsDelta, 14
cliffsDelta(), 26, 75
clusteringCoefficient, 15
clusterPermutationTest, 15
clusterPermutationTest(), 6, 20, 31, 92, 95
coherence, 17
coherence(), 19, 21, 24, 58, 75, 99
connectivityMatrix, 19
connectivityMatrix(), 18, 21, 24, 58, 75, 99
correctPValues, 20
correctPValues(), 17, 31, 95
correlationMatrix, 21
correlationMatrix(), 19
correlationTest, 22, 42, 43, 47, 57
cronbachAlpha, 22
crossSpectrum, 23
crossSpectrum(), 18

dunnTest, 24, 51

effectSize, 25
effectSize(), 6, 14, 75, 95
eigenvectorCentrality, 26
epochData, 27
epochData(), 10, 30, 36, 61
epochSliding, 29
epochTimes, 29
epochTimes(), 28

fftSignals, 30
fftSignals(), 11, 67, 84
findSignificantWindows, 31
findSignificantWindows(), 17, 20
friedmanTest, 32, 44, 51
functionalMixedModel, 33
functionalRegression, 34
functionalRegression(), 33, 76

globalEfficiency, 35
grandAverage, 36
grandAverage(), 10
graphLaplacian, 36

hilbertTransform, 37
hilbertTransform(), 38, 39, 98

instantaneousAmplitude, 38
instantaneousAmplitude(), 37, 39
instantaneousPhase, 39
instantaneousPhase(), 37, 38
interquartileRange, 40, 93, 94
intraclassCorrelation, 41

jarqueBeraTest, 41, 45, 55

kendallTau, 42, 57
kpssTest, 43
kruskalTest, 24, 25, 44
ksNormalityTest, 45

leveneTest, 46

- linearRegression, [46, 50, 51](#)
- ljungBoxTest, [5, 47](#)
- localEfficiency, [48](#)
- medianAbsDev, [40, 49, 93, 94](#)
- modularity, [50](#)
- multipleRegression, [50](#)
- nemenyiTest, [51](#)
- networkBasedStatistic, [52](#)
- networkBasedStatistic(), [63, 64](#)
- nodeDegree, [53](#)
- oneWayAnova, [32, 44, 46, 54, 95, 96](#)
- pacf, [56](#)
- pairedTTest, [22, 32, 54, 55, 97](#)
- partialAutocorrelation, [8, 55](#)
- partialCorrelation, [51, 56](#)
- pathLength, [57](#)
- PhysioBiomarker, [81, 82](#)
- PhysioExperiment::physioBiomarker(), [82](#)
- pli, [58](#)
- pli(), [19, 75, 99](#)
- plotAdjacencyMatrix, [59](#)
- plotDynamicConnectivity, [60](#)
- plotERP, [61](#)
- plotERP(), [10, 14, 28, 30, 36, 63, 68, 72, 73](#)
- plotMultiChannel, [62](#)
- plotMultiChannel(), [61, 68, 72, 73](#)
- plotNBSNetwork, [63](#)
- plotNBSNetwork(), [53](#)
- plotNetwork, [64](#)
- plotNetworkMetrics, [65](#)
- plotNetworkStability, [66](#)
- plotPSD, [66](#)
- plotPSD(), [11, 30, 68, 69](#)
- plotSignal, [67](#)
- plotSignal(), [63, 67, 69](#)
- plotSpecparam, [68](#)
- plotSpecparam(), [81](#)
- plotSpectrogram, [69](#)
- plotSpectrogram(), [67, 84](#)
- plotSPM, [70](#)
- plotTopomap, [71](#)
- plotTopomap(), [61, 63, 73](#)
- plotTopomapSeries, [73](#)
- plotTopomapSeries(), [72](#)
- plv, [74](#)
- plv(), [18, 19, 21, 37, 39, 58, 99](#)
- print.specparam_result(specparam), [80](#)
- quantile, [40](#)
- rankBiserial, [75](#)
- rankBiserial(), [14, 26](#)
- scalarOnFunctionRegression, [76](#)
- scalarOnFunctionRegression(), [34](#)
- sd, [77](#)
- signalMoments, [22, 40–42, 45, 49, 55, 77, 94](#)
- slidingWindowConnectivity, [78](#)
- smallWorldness, [79](#)
- spec.ar, [8](#)
- specparam, [80](#)
- specparam(), [68, 81, 82](#)
- specparamBiomarker, [81](#)
- specparamBiomarker(), [81](#)
- spectralClustering, [82](#)
- spectralDecomposition, [83](#)
- spectrogram, [83](#)
- spectrogram(), [11, 24, 30, 69, 98](#)
- spmAnova, [85](#)
- spmAnova(), [89](#)
- spmMANOVA, [85](#)
- spmMANOVA(), [88, 89](#)
- spmPairedTTest, [86](#)
- spmRegression, [87](#)
- spmRegression(), [34, 86](#)
- spmSnPM, [88](#)
- spmSnPM(), [86, 88](#)
- spmTTest, [89](#)
- spmTTest(), [33, 86–89](#)
- temporalStability, [91](#)
- tfce, [91](#)
- tfce(), [16](#)
- thresholdNetwork, [92](#)
- trimmedMean, [93](#)
- tTestEpochs, [94](#)
- tTestEpochs(), [6, 14, 17, 20, 26, 28, 31](#)
- tukeyHSD, [24, 25, 51, 95](#)
- twoSampleTTest, [54, 96](#)
- waveletTransform, [97](#)
- waveletTransform(), [11, 38, 84](#)
- wPLI, [98](#)
- wPLI(), [19, 58, 75](#)