

Package: PhysioEEG (via r-universe)

October 3, 2026

Title EEG Analysis Functions for PhysioExperiment Objects

Version 0.7.16

Description Provides comprehensive electroencephalography (EEG) analysis functions for PhysioExperiment objects. Includes preprocessing (filtering, re-referencing, bad channel detection, interpolation, epoching, artifact rejection), independent component analysis (FastICA, Infomax, JADE) with automatic artifact detection, event-related potential (ERP) component detection and measurement (N100, P300, N400, P600, MMN), source localization (eLORETA, sLORETA, LCMV beamformer), EEG microstate analysis (K-means, AAHC), sleep staging (AASM criteria, spindle/K-complex/slow-wave detection), brain-computer interface features (CSP, SSVEP, motor imagery), clinical EEG analysis (spike detection, QEEG, asymmetry indices), time-frequency analysis (Morlet wavelet, STFT, multitaper, ERSP, ITC), connectivity analysis (coherence, PLV, wPLI, Granger causality), visualization (signal traces, ERP waveforms, topographic maps, spectrograms, connectivity plots, hypnograms, source maps), and simulated data generators for testing and demonstration.

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

Imports SummarizedExperiment, S4Vectors, utils, stats

Suggests PhysioCore, ggplot2, patchwork, signal, digest, knitr, rmarkdown, testthat (>= 3.2.0), reticulate, PhysioAnalysis, PhysioCrossModal, PhysioHeadModels

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Collate 'utils-provenance.R' 'eeg-data.R' 'eeg-preprocess.R'
'eeg-csd.R' 'eeg-ica.R' 'eeg-iclabeled.R' 'eeg-iclabeled-trained.R'
'eeg-erp.R' 'eeg-source.R' 'eeg-microstate.R' 'eeg-sleep.R'

'eeg-timefreq.R' 'eeg-timefreq-mask.R' 'eeg-glassbrain.R'
 'eeg-connectivity.R' 'eeg-connectogram.R' 'eeg-pac.R'
 'eeg-bci.R' 'eeg-decode.R' 'eeg-clinical.R' 'eeg-complexity.R'
 'eeg-aperiodic.R' 'eeg-vis.R' 'zzz.R'

RoxygenNote 7.3.3

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

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

Config/pak/sysreqs zlib1g-dev

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

Date/Publication 2026-10-03 13:11:43 UTC

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

RemoteRef HEAD

RemoteSha 7a1c2ee33672926cb2c5e0c6caf91608cd236ebc

Contents

dispersionEntropy	4
eegAperiodic	5
eegArtifactReject	7
eegAsymmetry	8
eegAutoReject	9
eegBadChannels	10
eegBCIclassify	11
eegBCIfeatures	12
eegBeamformer	13
eegBSI	15
eegCoherence	16
eegComodulogram	17
eegComplexity	19
eegConditionalGC	20
eegConnectivityMatrix	21
eegCSP	22
eegDAR	23
eegDecode	25
eegDipoleFit	27
eegDTF	28
eegEpoch	29
eegERPbaseline	30
eegERPdetect	31
eegERPdifference	32
eegERPgrandAverage	33
eegERPlatency	34

eegERPmeasure	35
eegERPtest	36
eegERSP	37
eegFilter	38
eegForwardModel	39
eegGrangerCausality	41
eegICA	42
eegICAdetect	43
eegICAmix	44
eegICAremove	45
eegICLabel	46
eegICLabelFlag	48
eegInterpolate	49
eegITC	50
eegKcomplexDetect	51
eegLateralization	52
eegMicrostateBackfit	54
eegMicrostates	55
eegMicrostateSequence	56
eegMicrostateStats	57
eegMontage	58
eegMorletWavelet	58
eegMotorImagery	60
eegMultitaper	61
eegPAC	62
eegPDC	63
eegPlotButterflyGFP	64
eegPlotConnectivity	65
eegPlotConnectogram	66
eegPlotERP	68
eegPlotGlassBrain	69
eegPlotHypnogram	70
eegPlotICA	71
eegPlotSignal	72
eegPlotSource	73
eegPlotSpectrogram	74
eegPlotTopomap	75
eegPlotTopomapSeries	76
eegPLV	77
eegPreprocess	78
eegQEEG	80
eegQuickStart	81
eegRereference	82
eegSleepMetrics	83
eegSleepStage	84
eegSlowing	86
eegSlowWaveDetect	87
eegSourceEstimate	88

eegSourcePower	89
eegSpikeDetect	90
eegSpindleDetect	91
eegSSVEP	93
eegSTFT	94
eegSuppression	95
eegSurfaceLaplacian	97
eegWPLI	98
fuzzyEntropy	99
incrementEntropy	100
katzFD	101
make_eeg	102
make_eeg_bci	103
make_eeg_erp	104
make_eeg_sleep	105
make_eeg_spikes	105
multiscaleEntropy	106
petrosianFD	107
phaseEntropy	108
renyiEntropy	109
slopeEntropy	110
svdEntropy	111
tsallisEntropy	112
Index	113

dispersionEntropy	<i>Dispersion Entropy</i>
-------------------	---------------------------

Description

Computes the dispersion entropy (DispEn) of a one-dimensional signal (Rostaghi & Azami, 2016) – a fast, robust symbolic-dynamics complexity measure. Unlike the amplitude-histogram entropies ([renyiEntropy](#), [tsallisEntropy](#)) and the ordinal permutation entropy, DispEn maps the signal to c amplitude classes through the normal cumulative distribution function (NCDF), forms embedding vectors of length dimension (the *dispersion patterns*), and takes the Shannon entropy of the pattern distribution, normalized by $\ln(c^m)$. The NCDF mapping makes it insensitive to outliers and much cheaper than sample entropy, with no threshold-crossing boundary effects. Set `reverse = TRUE` to return the reverse dispersion entropy (RDEn), the mean-square deviation of the pattern distribution from uniformity (larger = more regular / less complex).

Usage

```
dispersionEntropy(x, c = 6L, dimension = 3L, delay = 1L, reverse = FALSE)
```

Arguments

x	A numeric vector (the time series).
c	Number of amplitude classes / symbols (default 6; Rostaghi & Azami recommend 4–8).
dimension	Embedding dimension m of the dispersion patterns (default 3).
delay	Time delay (lag) in samples (default 1).
reverse	If TRUE, return the reverse dispersion entropy (RDEn) instead of DispEn (default FALSE).

Details

This implementation reproduces NeuroKit2's `entropy_dispersion` bit-for-bit, including its normalization convention (a base-2 Shannon term divided by the natural-log $\ln(c^m)$, so DispEn ranges in $[0, 1/\ln 2]$).

Value

A single numeric value: the dispersion entropy (or, if `reverse = TRUE`, the reverse dispersion entropy).

References

Rostaghi, M., & Azami, H. (2016). "Dispersion entropy: A measure for time-series analysis." *IEEE Signal Processing Letters*, 23(5), 610–614. doi:10.1109/LSP.2016.2542881

See Also

[renyiEntropy](#), [tsallisEntropy](#), [svdEntropy](#), [petrosianFD](#).

Examples

```
set.seed(1)
dispersionEntropy(rnorm(1000))           # DispEn (c = 6, m = 3)
dispersionEntropy(rnorm(1000), reverse = TRUE) # reverse dispersion entropy
```

eegAperiodic

Aperiodic (1/f) spectral parameterization of EEG

Description

Separates each channel's power spectrum into an **aperiodic** (1/f) component and **periodic** (oscillatory) peaks, following the `specparam` / `FOOOF` model (Donoghue et al. 2020). The aperiodic exponent is a widely used index of the excitation/inhibition balance and cortical state. Delegates the fit to `PhysioAnalysis::specparam()`.

Usage

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

Arguments

<code>pe</code>	A <code>PhysioExperiment</code> .
<code>freq_range</code>	Frequency range to fit, in Hz (default <code>c(1, 45)</code>).
<code>aperiodic_mode</code>	"fixed" (offset + exponent) or "knee" (offset + knee + exponent), the latter for spectra with a bend in log-log space.
<code>max_n_peaks</code>	Maximum number of oscillatory peaks per channel (default 6).
<code>peak_width_limits</code>	Min/max peak width in Hz (default <code>c(1, 12)</code>).
<code>min_peak_height</code>	Minimum peak height above the aperiodic fit (default 0.05).
<code>peak_threshold</code>	Peak detection threshold in SD of the flattened spectrum (default 2).
<code>assay_name</code>	Assay to use (default: the object's default assay).

Value

An `eeg_aperiodic` object: a list with `aperiodic` (per-channel data frame: `channel`, `exponent`, `offset`, optionally `knee`, `r_squared`, `error`), `peaks` (per-channel CF/PW/BW), `exponent` (a named per-channel vector, e.g. for `eegPlotTopomap()`), and the underlying `specparam_result`.

References

Donoghue et al. 2020, Nat Neurosci (specparam / FOOF).

See Also

[eegQEEG\(\)](#), [eegComplexity\(\)](#)

Examples

```
if (requireNamespace("PhysioAnalysis", quietly = TRUE)) {
  pe <- make_eeg(n_time = 2500, n_channels = 8, sr = 250)
  ap <- eegAperiodic(pe, freq_range = c(2, 40))
  ap$aperiodic      # per-channel exponent / offset
}
```

eegArtifactReject	<i>Reject artifacts in epoched EEG data</i>
-------------------	---

Description

Identifies and removes epochs contaminated by artifacts from 3D epoched EEG data. Supports multiple detection criteria: amplitude threshold, gradient (point-to-point voltage change), and joint probability.

Usage

```
eegArtifactReject(
  x,
  method = c("threshold", "gradient", "joint_probability"),
  threshold_uv = 100,
  gradient_uv_ms = 50,
  jp_threshold = 3,
  assay_name = NULL,
  output_assay = "clean"
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with 3D epoched data.
<code>method</code>	Artifact detection method: "threshold" (amplitude), "gradient" (point-to-point change), or "joint_probability" (log-power z-score).
<code>threshold_uv</code>	Maximum absolute amplitude in microvolts for threshold rejection (default: 100).
<code>gradient_uv_ms</code>	Maximum point-to-point change in uV/ms for gradient rejection (default: 50).
<code>jp_threshold</code>	Number of standard deviations for joint probability rejection (default: 3).
<code>assay_name</code>	Name of the assay to check. If <code>NULL</code> , uses <code>defaultAssay(x)</code> .
<code>output_assay</code>	Name of the output assay (default: "clean").

Value

A `PhysioExperiment` object with clean epochs in the specified output assay. Artifact log stored in `metadata(x)$artifact_log`.

Examples

```
pe <- make_eeg(n_time = 10000, n_channels = 19, sr = 500)
events <- data.frame(onset_sec = c(1, 3, 5, 7, 9))
pe_ep <- eegEpoch(pe, events, limits = c(-0.2, 0.8))
pe_clean <- eegArtifactReject(pe_ep, method = "threshold",
                             threshold_uv = 100, assay_name = "epoched")
```

eegAsymmetry

*Frontal Alpha Asymmetry***Description**

Computes frontal alpha asymmetry indices from paired electrode sites. For each pair (right, left), band power is computed via FFT and the asymmetry index is calculated as $\log(\text{power_right}) - \log(\text{power_left})$. Positive values indicate greater right-hemisphere alpha (typically associated with greater left-hemisphere cortical activity).

Usage

```
eegAsymmetry(x, pairs = NULL, band = c(8, 13), assay_name = NULL)
```

Arguments

x	A <code>PhysioExperiment</code> object with EEG data.
pairs	A list of character vectors of length 2, each specifying a (right, left) electrode pair. Defaults to <code>list(c("F4", "F3"), c("F8", "F7"))</code> .
band	Numeric vector of length 2 specifying the frequency band in Hz (default: <code>c(8, 13)</code> for alpha).
assay_name	Name of the input assay. If <code>NULL</code> , the default assay is used.

Value

A `data.frame` with columns:

pair Character label for the electrode pair.
left_channel Character name of the left electrode.
right_channel Character name of the right electrode.
left_power Numeric band power for the left electrode.
right_power Numeric band power for the right electrode.
asymmetry_index Numeric asymmetry: $\log(\text{right}) - \log(\text{left})$.

References

Nuwer, M. R., et al. (1999). IFCN standards for digital recording of clinical EEG. *Electroencephalography and Clinical Neurophysiology*, 106(3), 259-261.

See Also

[eegSpikeDetect\(\)](#), [eegQEEG\(\)](#), [eegSlowing\(\)](#)

Examples

```
pe <- make_eeg(n_time = 5000, n_channels = 19, sr = 500)
asym <- eegAsymmetry(pe)
print(asym)
```

eegAutoReject

Automated epoch rejection and interpolation (autoreject)

Description

Implements the autoreject algorithm (Jas et al. 2017) for epoched EEG. Per-channel peak-to-peak rejection thresholds are chosen by cross-validation, then each epoch is kept, repaired by interpolating its worst bad channels, or dropped, according to two hyperparameters: `consensus` (the fraction of bad channels above which an epoch is dropped) and `n_interpolate` (the number of worst bad channels to interpolate in a repaired epoch). Either hyperparameter is chosen by cross-validation when not supplied. Bad channels are repaired with spherical-spline interpolation (the same weights used by [eegInterpolate\(\)](#)), so electrode positions must be present (run [eegMontage\(\)](#) first).

Usage

```
eegAutoReject(
  x,
  consensus = NULL,
  n_interpolate = NULL,
  cv_folds = 5L,
  thresh_range = NULL,
  n_thresh = 20L,
  assay_name = "epoched",
  output_assay = "autoreject_clean"
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> with 3D epoched data (time x channels x epochs) in <code>metadata(x)[[assay_name]]</code> (from eegEpoch()) or in a 3D assay of that name (from make_eeg_erp()).
<code>consensus</code>	Fraction of bad channels (from 0 to 1) above which an epoch is dropped. <code>NULL</code> selects it by cross-validation.
<code>n_interpolate</code>	Number of worst bad channels to interpolate per repaired epoch. <code>NULL</code> selects it by cross-validation.
<code>cv_folds</code>	Number of cross-validation folds (default: 5).
<code>thresh_range</code>	Optional candidate peak-to-peak threshold grid: either a length-2 <code>c(min, max)</code> span (expanded to <code>n_thresh</code> values) or an explicit vector of candidate thresholds. <code>NULL</code> derives the span from the data.
<code>n_thresh</code>	Number of grid thresholds when <code>thresh_range</code> is <code>NULL</code> or a span (default: 20).
<code>assay_name</code>	Name of the 3D epoched data in metadata or assays (default: "epoched").
<code>output_assay</code>	Name for the cleaned 3D data stored in metadata (default: "autoreject_clean").

Details

The cross-validation objective, at every stage, is the root-mean-square error between the mean of the good/cleaned training epochs and the median of the held-out test epochs - the criterion introduced by autoreject.

Value

The `PhysioExperiment` with cleaned 3D epoched data in `metadata(x)[[output_assay]]` (dropped epochs removed, repaired epochs interpolated) and a rejection log in `metadata(x)$autoreject`: the per-channel thresholds, chosen consensus and `n_interpolate`, the epoch-by-channel `bad_matrix`, the interpolated channel-epochs, and the `dropped_epochs`. The step is recorded in provenance.

References

Jas, M., Engemann, D. A., Bekhti, Y., Raimondo, F., & Gramfort, A. (2017). Autoreject: Automated artifact rejection for MEG and EEG data. *NeuroImage*, 159, 417-429.

See Also

[eegArtifactReject\(\)](#), [eegInterpolate\(\)](#), [eegEpoch\(\)](#), [eegMontage\(\)](#)

Examples

```
pe <- make_eeg_erp(n_epochs = 40, n_channels = 19, sr = 250)
pe <- eegMontage(pe, system = "10-20")
pe <- eegAutoReject(pe, assay_name = "raw")
dim(S4Vectors::metadata(pe)$autoreject$bad_matrix)
```

eegBadChannels

Detect bad EEG channels

Description

Identifies bad (noisy, flat, or poorly correlated) EEG channels using multiple automated criteria. Channels flagged as bad can subsequently be interpolated using [eegInterpolate](#).

Usage

```
eegBadChannels(
  x,
  method = c("all", "flat", "noise", "correlation"),
  flat_threshold = 1e-06,
  noise_threshold = 4,
  corr_threshold = 0.4,
  assay_name = NULL
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object.
<code>method</code>	Detection method(s) to apply: "all" runs all checks, or specify one or more of "flat", "noise", "correlation".
<code>flat_threshold</code>	Variance threshold below which a channel is considered flat (default: 1e-6).

noise_threshold	Number of standard deviations above median variance to flag a channel as noisy (default: 4).
corr_threshold	Minimum mean correlation with other channels. Channels below this are flagged (default: 0.4).
assay_name	Name of the assay to analyze. If NULL, uses defaultAssay(x).

Value

A data.frame with columns: channel (label), is_bad (logical), reason (character description), score (numeric metric value).

Examples

```
pe <- make_eeg(n_time = 5000, n_channels = 19, sr = 500)
bad_df <- eegBadChannels(pe)
bad_labels <- bad_df$channel[bad_df$is_bad]
```

eegBCIclassify	<i>BCI Classification</i>
----------------	---------------------------

Description

Classifies BCI features using Linear Discriminant Analysis (LDA) or shrinkage LDA. Implements Fisher's LDA with optional Ledoit-Wolf shrinkage regularization for robust classification with high-dimensional or small-sample data. Optionally performs k-fold cross-validation.

Usage

```
eegBCIclassify(
  x,
  features = NULL,
  labels,
  method = c("lda", "shrinkage_lda"),
  cv_folds = NULL,
  assay_name = NULL
)
```

Arguments

x	A PhysioExperiment object with epoched (3D) EEG data.
features	Optional pre-computed feature matrix (n_trials x n_features). If NULL, features are extracted using eegBCIfeatures with the "bandpower" method.
labels	Character or factor vector of class labels, one per trial. Must contain exactly two unique classes.
method	Classification method: "lda" (Fisher's LDA) or "shrinkage_lda" (LDA with Ledoit-Wolf shrinkage).

cv_folds	Number of cross-validation folds (default: NULL, meaning no cross-validation). If set (e.g., 5), performs k-fold CV and reports out-of-fold predictions. The CV accuracy is stored as <code>attr(result, "cv_accuracy")</code> .
assay_name	Input assay name used when extracting features (default: first assay).

Value

A `data.frame` with columns: `trial`, `predicted_class`, `confidence`, and `true_class`. When `cv_folds` is not NULL, predictions are out-of-fold and `attr(result, "cv_accuracy")` contains the cross-validated accuracy. The trained LDA model (on all data) is stored in `metadata(x)$bci_model`, containing weights, threshold, classes, method, and class_means.

References

Blankertz, B., et al. (2008). Optimizing spatial filters for robust EEG single-trial analysis. *IEEE Signal Processing Magazine*, 25(1), 41-56.

See Also

[eegCSP\(\)](#), [eegBCIfeatures\(\)](#), [eegMotorImagery\(\)](#)

Examples

```
pe <- make_eeg_bci(n_trials = 20, n_channels = 8, sr = 256)
labels <- S4Vectors::metadata(pe)$labels
features <- eegBCIfeatures(pe, method = "bandpower")
result <- eegBCIclassify(pe, features = features, labels = labels, method = "lda")

# With 5-fold cross-validation
result_cv <- eegBCIclassify(pe, features = features, labels = labels,
                           method = "lda", cv_folds = 5)
attr(result_cv, "cv_accuracy")
```

eegBCIfeatures	<i>BCI Feature Extraction</i>
----------------	-------------------------------

Description

Extracts features from epoched EEG data for Brain-Computer Interface classification. Supports band power, CSP, and Riemannian geometry methods.

Usage

```
eegBCIfeatures(
  x,
  method = c("bandpower", "csp", "riemannian"),
  labels = NULL,
  bands = NULL,
  assay_name = NULL
)
```

Arguments

x	A PhysioExperiment object with epoched (3D) EEG data (time x channels x trials).
method	Feature extraction method: "bandpower" (log band power), "csp" (Common Spatial Pattern log-variance), or "riemannian" (tangent space projection of covariance matrices).
labels	Character or factor vector of class labels (required for "csp" method). One label per trial.
bands	Named list of frequency bands for "bandpower" method. Default: <code>list(mu = c(8, 13), beta = c(13, 30))</code> .
assay_name	Input assay name (default: first assay).

Value

A numeric matrix with `n_trials` rows and feature columns. Number of columns depends on method:

- "bandpower": `n_channels * n_bands`
- "csp": `2 * n_filters` (default: 6)
- "riemannian": `n_channels * (n_channels + 1) / 2`

References

Blankertz, B., et al. (2008). Optimizing spatial filters for robust EEG single-trial analysis. *IEEE Signal Processing Magazine*, 25(1), 41-56.

See Also

[eegCSP\(\)](#), [eegMotorImagery\(\)](#), [eegBCIclassify\(\)](#)

Examples

```
pe <- make_eeg_bci(n_trials = 20, n_channels = 8, sr = 256)
features <- eegBCIfeatures(pe, method = "bandpower")
```

eegBeamformer

EEG Beamformer Source Localization

Description

Applies spatial filtering (beamforming) to localize neural source power. Linearly Constrained Minimum Variance (LCMV) beamformer operates in the time domain. Dynamic Imaging of Coherent Sources (DICS) operates in the frequency domain.

Usage

```
eegBeamformer(
  x,
  forward_model,
  method = c("lcmv", "dics"),
  freq_range = NULL,
  assay_name = NULL,
  output_assay = "beamformer"
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with EEG data.
<code>forward_model</code>	A forward model list as returned by eegForwardModel .
<code>method</code>	Beamformer method: "lcmv" (Linearly Constrained Minimum Variance) or "dics" (Dynamic Imaging of Coherent Sources).
<code>freq_range</code>	Numeric vector of length 2 specifying frequency range in Hz for DICS method (e.g., <code>c(8, 13)</code> for alpha band). Ignored for LCMV.
<code>assay_name</code>	Name of the input assay. If <code>NULL</code> , the default assay is used.
<code>output_assay</code>	Name for the output assay containing beamformer results (default: "beamformer"). The metadata names "source_estimate", "beamformer_info", and "source_plot_default" are reserved.

Value

Modified `PhysioExperiment` with source power stored in `output_assay` as a matrix with one column named "power" (`n_sources` rows). Each value represents the estimated source power at the corresponding dipole location. Plotting metadata, including source positions, the output name, method, and coordinate provenance, is stored in `metadata(x)$beamformer_info`.

References

Pascual-Marqui, R. D. (2002). Standardized low-resolution brain electromagnetic tomography (sLORETA). *Methods and Findings in Experimental and Clinical Pharmacology*, 24(Suppl D), 5-12.

Van Veen, B. D., et al. (1997). Localization of brain electrical activity via linearly constrained minimum variance spatial filtering. *IEEE Transactions on Biomedical Engineering*, 44(9), 867-880.

See Also

[eegForwardModel\(\)](#), [eegSourceEstimate\(\)](#), [eegSourcePower\(\)](#), [eegPlotGlassBrain\(\)](#)

Examples

```
pe <- make_eeg(n_time = 1000, n_channels = 19, sr = 250)
fm <- eegForwardModel(pe, method = "spherical", n_sources = 50)
pe <- eegBeamformer(pe, fm, method = "lcmv")
```

eegBSI

*Brain Symmetry Index (BSI) and pairwise-derived pdBSI***Description**

Quantifies left/right spectral asymmetry across hemispheric electrode pairs, a quantitative-EEG stroke marker (van Putten 2007; Sheorajpanday et al. 2011). The global bsi is the spectrum-averaged absolute relative difference between summed right- and left-hemisphere power, bounded from 0 (symmetric) to 1 (one hemisphere silent). The pairwise-derived pdbsi averages the same relative difference per hemispheric pair; with directed = TRUE it keeps the sign (positive = right-dominant). Both are returned as reliability-characterised [physioBiomarker](#) objects with provenance and, for the global BSI, a normative reference range.

Usage

```
eegBSI(
  x,
  pairs = NULL,
  band = c(1, 25),
  directed = FALSE,
  method = "welch",
  age = NULL,
  montage = NULL,
  reliability = list(icc = NA_real_, sem = NA_real_),
  assay_name = NULL
)
```

Arguments

x	A PhysioExperiment object with 2D EEG data.
pairs	List of length-2 character vectors, each a c(right, left) hemispheric electrode pair. Defaults to the standard 10-20 hemispheric pairs.
band	Numeric length-2 frequency band in Hz (default: c(1, 25)).
directed	Logical; when TRUE the pairwise index keeps its sign (default: FALSE).
method	Spectral estimator (default: "welch").
age	Optional numeric age for the normative lookup.
montage	Optional character montage for the normative lookup.
reliability	Named list of reliability indices to attach (default: icc and sem placeholders).
assay_name	Input assay name (default: the default assay).

Value

A named list with two [PhysioBiomarker](#) objects: bsi (global, from 0 to 1) and pdbsi (pairwise-derived, signed when directed = TRUE).

References

- van Putten, M. J. A. M. (2007). The revised brain symmetry index. *Clinical Neurophysiology*, 118(11), 2362-2367.
- Sheorajpanday, R. V. A., et al. (2011). Quantitative EEG in ischemic stroke: correlation with functional status after 6 months. *Clinical Neurophysiology*, 122(5), 874-883.

See Also

`eegDAR()`, `eegAsymmetry()`, `PhysioExperiment::physioBiomarker()`

Examples

```
pe <- make_eeg(n_time = 5000, n_channels = 19, sr = 250)
res <- eegBSI(pe, age = 60)
res$bsi
```

eegCoherence

EEG Coherence Analysis

Description

Computes magnitude-squared coherence (MSC) or imaginary part of coherency between all EEG channel pairs using Welch's method. The signal is segmented into overlapping windows, each windowed with a Hanning function, and the cross-spectral density is averaged across windows.

Usage

```
eegCoherence(
  x,
  method = c("coherence", "imaginary"),
  window_sec = 2,
  overlap = 0.5,
  band = c(8, 13),
  assay_name = NULL
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with EEG data (2D: time x channels).
<code>method</code>	Coherence method: "coherence" for magnitude-squared coherence or "imaginary" for imaginary part of coherency (default: "coherence").
<code>window_sec</code>	Window length in seconds for Welch's method (default: 2).
<code>overlap</code>	Overlap fraction between adjacent windows, from 0 to 1 exclusive (default: 0.5).
<code>band</code>	Numeric vector of length 2 specifying the frequency band in Hz over which to average coherence (default: c(8, 13) for alpha band).
<code>assay_name</code>	Name of the input assay. If NULL, the default assay is used.

Details

Magnitude-squared coherence is defined as:

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

Imaginary coherence uses the imaginary part of coherency to reduce volume conduction artifacts:

$$ICoh(f) = |Im(S_{xy}(f)) / \sqrt{S_{xx}(f) \cdot S_{yy}(f)}|$$

Value

The input `PhysioExperiment` with connectivity results stored in `metadata(x)$connectivity`, a list containing:

matrix Numeric `n_channels` x `n_channels` matrix of band-averaged coherence values.

method Character string indicating the method used.

band Numeric vector of the frequency band used.

freqs Numeric vector of frequency bins.

spectra 3D array (`n_freqs` x `n_channels` x `n_channels`) of frequency-resolved coherence values.

References

Lachaux, J. P., et al. (1999). Measuring phase synchrony in brain signals. *Human Brain Mapping*, 8(4), 194-208.

See Also

[eegPLV\(\)](#), [eegWPLI\(\)](#), [eegConnectivityMatrix\(\)](#), [eegPlotConnectivity\(\)](#)

Examples

```
pe <- make_eeg(n_time = 5000, n_channels = 4, sr = 500)
pe <- eegCoherence(pe, method = "coherence", band = c(8, 13))
coh_matrix <- S4Vectors::metadata(pe)$connectivity$matrix
```

eegComodulogram	<i>Comodulogram (phase-frequency x amplitude-frequency PAC) for EEG</i>
-----------------	---

Description

Computes phase-amplitude coupling over a grid of phase and amplitude frequencies for one EEG channel, yielding a comodulogram whose peak locates the dominant coupling. Delegates to `PhysioCrossModal::comodulogram`.

Usage

```
eegComodulogram(
  pe,
  channel = 1L,
  phase_freqs = seq(2, 14, by = 2),
  amp_freqs = seq(20, 100, by = 10),
  method = c("tort", "canolty", "ozkurt", "plv"),
  phase_bw = 2,
  amp_bw = 10,
  n_bins = 18L,
  assay_name = NULL
)
```

Arguments

<code>pe</code>	A <code>PhysioExperiment</code> .
<code>channel</code>	Channel (index or label) to analyse.
<code>phase_freqs</code>	Phase centre frequencies in Hz (default <code>seq(2, 14, 2)</code>).
<code>amp_freqs</code>	Amplitude centre frequencies in Hz (default <code>seq(20, 100, 10)</code>).
<code>method</code>	PAC estimator (see eegPAC()).
<code>phase_bw, amp_bw</code>	Half-bandwidths in Hz (defaults 2 and 10).
<code>n_bins</code>	Phase bins for the Tort modulation index (default 18).
<code>assay_name</code>	Assay to use (default: the object's default assay).

Value

The comodulogram list from `PhysioCrossModal::comodulogram()` (a matrix of phase x amplitude frequencies, plus its peak).

See Also

[eegPAC\(\)](#)

Examples

```
if (requireNamespace("PhysioCrossModal", quietly = TRUE)) {
  pe <- make_eeg(n_time = 7500, n_channels = 2, sr = 250)
  cm <- eegComodulogram(pe, channel = 1)
  cm$peak
}
```

Description

Computes per-channel entropy, fractal-dimension and detrended-fluctuation complexity measures — the nonlinear family that complements the spectral, connectivity and source tools. For epoched (3-D) data each measure is computed per epoch and averaged across epochs.

Usage

```
eegComplexity(
  pe,
  measures = c("sample_entropy", "permutation_entropy", "lempel_ziv", "higuchi_fd",
    "dfa", "hjorth_mobility", "spectral_entropy"),
  assay_name = NULL,
  m = 2L,
  r = 0.2,
  tau = 1L,
  perm_order = 3L,
  mse_scales = 1:8,
  max_samples = 4000L
)
```

Arguments

<code>pe</code>	A <code>PhysioExperiment</code> .
<code>measures</code>	Character vector of measures to compute (see Details).
<code>assay_name</code>	Assay to use (default: the object's default assay).
<code>m</code>	Embedding dimension for the entropy measures (default 2; permutation entropy uses <code>perm_order</code>).
<code>r</code>	Tolerance as a fraction of each channel's SD for sample/approximate/ multiscale entropy (default 0.2).
<code>tau</code>	Time delay for permutation entropy (default 1).
<code>perm_order</code>	Embedding order for permutation entropy (default 3).
<code>mse_scales</code>	Scales for multiscale entropy (default 1:8).
<code>max_samples</code>	Cap on samples per channel/epoch for the $O(N^2)$ entropy measures; longer series are truncated with a warning (default 4000; NULL disables the cap).

Details

Available measures: "sample_entropy" (Richman–Moorman), "approximate_entropy" (Pincus), "permutation_entropy" (Bandt–Pompe), "multiscale_entropy" (Costa; returns the scale-averaged SampEn as a summary plus the per-scale curve in the attribute), "lempel_ziv" (Kaspar–Schuster), "higuchi_fd", "katz_fd", "dfa" (scaling exponent α), "hurst" (R/S), "hjorth_mobility", "hjorth_complexity", "spectral_entropy".

Value

A data frame with one row per channel and one column per requested measure (plus a channel column). The multiscale-entropy per-scale curves are attached as `attr(, "mse_scales") / attr(, "mse_curve")`.

References

Richman & Moorman 2000; Bandt & Pompe 2002; Costa 2002; Higuchi 1988; Katz 1988; Peng 1994; Hjorth 1970.

See Also

[eegQEEG\(\)](#), [eegAperiodic\(\)](#)

Examples

```
pe <- make_eeg(n_time = 512, n_channels = 4, sr = 128)
cx <- eegComplexity(pe, measures = c("permutation_entropy", "hjorth_mobility",
                                   "spectral_entropy"))
cx
```

eegConditionalGC

Conditional (multivariate) Granger causality

Description

Computes conditional Granger causality from source to target given a set of conditioning channels, using the log ratio of the target's residual variance in a VAR fitted without the source (target + conditioning) versus with it (target + source + conditioning); both VARs share the same order (Geweke; Barnett & Seth 2014). Unlike the pairwise [eegGrangerCausality\(\)](#), conditioning removes indirect influences, so for a chain the conditional GC along a purely indirect path is near zero.

Usage

```
eegConditionalGC(
  x,
  target,
  source,
  conditioning = NULL,
  order = NULL,
  method = "ols",
  assay_name = NULL
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with 2D EEG data (time x channels).
<code>target</code>	Target channel (label or index) - the effect.
<code>source</code>	Source channel (label or index) - the putative cause.
<code>conditioning</code>	Channels to condition on (labels or indices). Defaults to all remaining channels.
<code>order</code>	MVAR model order, or NULL to select automatically on the full channel set.
<code>method</code>	MVAR estimator passed to <code>PhysioExperiment::mvarFit()</code> (default: "ols").
<code>assay_name</code>	Input assay name (default: the default assay).

Value

A list with the conditional GC value (in nats, non-negative in theory), the target, source, and conditioning labels, and the VAR order used.

References

Barnett, L., & Seth, A. K. (2014). The MVGC multivariate Granger causality toolbox. *Journal of Neuroscience Methods*, 223, 50-68.

See Also

[eegGrangerCausality\(\)](#), [eegDTF\(\)](#), [eegPDC\(\)](#)

Examples

```
pe <- make_eeg(n_time = 4000, n_channels = 3, sr = 250)
eegConditionalGC(pe, target = 3, source = 1, conditioning = 2)
```

`eegConnectivityMatrix` *EEG Connectivity Matrix*

Description

Convenience wrapper that computes a symmetric `n_channels` x `n_channels` connectivity matrix using the specified method. Dispatches to the appropriate connectivity function and returns a named matrix suitable for visualization or graph-theoretic analysis.

Usage

```
eegConnectivityMatrix(
  x,
  method = c("coherence", "plv", "wpli", "dtf", "pdc"),
  band = c(8, 13),
  ...
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with EEG data (2D: time x channels).
<code>method</code>	Connectivity method: "coherence" for magnitude-squared coherence, "plv" for Phase Locking Value, "wpli" for weighted Phase Lag Index, or the directed measures "dtf" (<code>eegDTF()</code>) and "pdc" (<code>eegPDC()</code>) (default: "coherence"). For "dtf" and "pdc" the returned matrix is directed (rows = targets, columns = sources).
<code>band</code>	Numeric vector of length 2 specifying the frequency band in Hz (default: <code>c(8, 13)</code> for alpha band).
<code>...</code>	Additional arguments passed to the underlying connectivity function (e.g., <code>window_sec</code> , <code>overlap</code>).

Value

A named numeric matrix of dimension `n_channels` x `n_channels` with channel labels as row and column names. Diagonal elements are 1. Off-diagonal elements represent the band-averaged connectivity between channel pairs.

References

Lachaux, J. P., et al. (1999). Measuring phase synchrony in brain signals. *Human Brain Mapping*, 8(4), 194-208.

See Also

`eegCoherence()`, `eegPLV()`, `eegWPLI()`, `eegGrangerCausality()`, `eegPlotConnectivity()`

Examples

```
pe <- make_eeg(n_time = 5000, n_channels = 4, sr = 500)
conn <- eegConnectivityMatrix(pe, method = "plv", band = c(8, 13))
print(conn)
```

eegCSP

Common Spatial Pattern (CSP) Analysis

Description

Computes Common Spatial Pattern filters for two-class EEG discrimination. CSP maximizes the variance ratio between two conditions, making it a standard spatial filtering technique for motor imagery BCI.

Usage

```
eegCSP(x, labels, n_filters = 3, assay_name = NULL, output_assay = "csp")
```

Arguments

x	A PhysioExperiment object with epoched (3D) EEG data (time x channels x trials).
labels	Character or factor vector of class labels, one per trial. Must contain exactly two unique classes.
n_filters	Number of CSP filter pairs to retain (default: 3). The total number of spatial filters will be 2 * n_filters.
assay_name	Input assay name (default: first assay).
output_assay	Name the result is stored under. Despite the argument's name this is a key in metadata , NOT an assay: after the call assay(x, "csp") does not exist while metadata(x)\$csp holds the result (default: "csp").

Value

Modified PhysioExperiment with CSP log-variance features in output_assay (a matrix of trials x 2 * n_filters) and CSP filter information stored in metadata(x)\$csp as a list containing filters (spatial filter matrix), eigenvalues (selected eigenvalues), and classes (unique class labels).

References

Blankertz, B., et al. (2008). Optimizing spatial filters for robust EEG single-trial analysis. IEEE Signal Processing Magazine, 25(1), 41-56.

See Also

[eegBCIfeatures\(\)](#), [eegBCIclassify\(\)](#), [eegMotorImagery\(\)](#)

Examples

```
pe <- make_eeg_bci(n_trials = 30, n_channels = 8, sr = 256)
labels <- S4Vectors::metadata(pe)$labels
result <- eegCSP(pe, labels = labels, n_filters = 3)
# CSP filters and log-variance features are stored in metadata
csp_features <- S4Vectors::metadata(result)$csp_features
dim(csp_features) # trials x (2 * n_filters)
```

eegDAR	<i>Delta/Alpha Ratio (DAR)</i>
--------	--------------------------------

Description

Computes the delta-to-alpha band-power ratio per channel (or per region), a quantitative-EEG stroke marker whose elevation tracks ischaemic slowing (Finnigan et al. 2016). Band powers use a Welch estimator for a stable spectrum. Each ratio is returned as a reliability-characterised [physioBiomarker](#) carrying provenance (band, method, software version), reliability placeholders, and - when age (and montage) are supplied - a published reference range from [normativeLookup](#) so the value prints its normative percentile.

Usage

```
eegDAR(
  x,
  delta = c(1, 4),
  alpha = c(8, 13),
  regions = NULL,
  method = "welch",
  age = NULL,
  montage = NULL,
  reliability = list(icc = NA_real_, sem = NA_real_),
  assay_name = NULL
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with 2D EEG data.
<code>delta</code>	Numeric length-2 delta band in Hz (default: <code>c(1, 4)</code>).
<code>alpha</code>	Numeric length-2 alpha band in Hz (default: <code>c(8, 13)</code>).
<code>regions</code>	Optional named list mapping a region name to a character vector of channel labels; band powers are averaged within each region. When <code>NULL</code> (default) one biomarker is returned per channel.
<code>method</code>	Spectral estimator passed to the band-power routine (default: <code>"welch"</code>).
<code>age</code>	Optional numeric age used to select the normative reference row.
<code>montage</code>	Optional character montage for the normative lookup.
<code>reliability</code>	Named list of reliability indices to attach (default: <code>icc</code> and <code>sem</code> placeholders).
<code>assay_name</code>	Input assay name (default: the default assay).

Value

A named list of [PhysioBiomarker](#) objects, one per channel or region.

References

Finnigan, S., Wong, A., & Read, S. (2016). Defining abnormal slow EEG activity in acute ischaemic stroke: Delta/alpha ratio as an optimal QEEG index. *Clinical Neurophysiology*, 127(2), 1452-1459.

See Also

[eegBSI\(\)](#), [eegSlowing\(\)](#), [eegQEEG\(\)](#), [PhysioExperiment::physioBiomarker\(\)](#), [PhysioExperiment::normativeLookup\(\)](#)

Examples

```
pe <- make_eeg(n_time = 5000, n_channels = 19, sr = 250)
dar <- eegDAR(pe, age = 60)
dar[["C3"]]
```

eegDecode

*Leakage-free MVPA Decoding of Epochs EEG***Description**

Decodes trial class labels from epoched EEG with a strictly fold-safe pipeline: the spatial filter (CSP), feature extraction, feature scaling and classifier are all estimated on the training fold only and applied, frozen, to the test fold (Varoquaux et al., 2017). This avoids the label/data leakage of fitting CSP or scaling on the full dataset before cross-validation. Folds may respect a run/block grouping to avoid temporally-contiguous-epoch leakage.

Usage

```
eegDecode(
  x,
  labels = NULL,
  pipeline = c("csp+lda", "bandpower+lda", "riemannian"),
  cv = c("stratified-kfold", "leave-one-run-out"),
  n_folds = 5,
  groups = NULL,
  n_permutations = 200,
  inner_cv = NULL,
  assay_name = NULL,
  n_csp = 3,
  bands = NULL,
  seed = NULL
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> with an epoched (3D: time x channels x trials) assay.
<code>labels</code>	Class labels, one per trial. If <code>NULL</code> , taken from <code>metadata(x)\$labels</code> .
<code>pipeline</code>	Decoding pipeline: "csp+lda" (common spatial patterns then LDA), "bandpower+lda" (per-channel band-power then LDA), or "riemannian" (tangent-space covariance features then LDA).
<code>cv</code>	Cross-validation scheme: "stratified-kfold" or "leave-one-run-out" (the latter requires groups).
<code>n_folds</code>	Number of folds for k-fold CV (default: 5).
<code>groups</code>	Optional run/block grouping (one per trial); folds keep each group intact so contiguous epochs are never split across train/test.
<code>n_permutations</code>	Number of label permutations for the null distribution and p-value (default: 200; 0 to skip).
<code>inner_cv</code>	Optional number of inner CV folds for nested hyperparameter selection (shrinkage LDA vs plain LDA). <code>NULL</code> disables nesting.
<code>assay_name</code>	Input assay (default: <code>defaultAssay(x)</code>).

<code>n_csp</code>	Number of CSP filter pairs for "csp+lda" (default: 3).
<code>bands</code>	Named list of frequency bands for "bandpower+lda" (default: <code>list(mu = c(8, 13), beta = c(13, 30))</code>).
<code>seed</code>	Optional RNG seed for reproducible fold assignment/permutations.

Value

A list of class "eeg_decode" with:

accuracy pooled out-of-fold accuracy.

accuracy_ci 95 percent Wilson confidence interval for the accuracy.

fold_accuracy per-fold accuracy.

auc binary area under the ROC curve.

confusion confusion matrix (true x predicted).

permutation list with `p_value`, `null_accuracy`, `n_permutations`.

predictions a `data.frame` (trial, fold, true, predicted, score).

csp_filters per-fold CSP filters ("csp+lda" only).

References

Varoquaux, G., et al. (2017). "Assessing and tuning brain decoders: cross-validation, caveats, and guidelines." *NeuroImage*, 145, 166-179. doi:[10.1016/j.neuroimage.2016.10.038](https://doi.org/10.1016/j.neuroimage.2016.10.038)

Lemm, S., et al. (2011). "Introduction to machine learning for brain imaging." *NeuroImage*, 56(2), 387-399. doi:[10.1016/j.neuroimage.2010.11.004](https://doi.org/10.1016/j.neuroimage.2010.11.004)

See Also

[eegCSP](#), [eegBCIfeatures](#). The older [eegBCIclassify](#) cross-validation path is leakage-prone and deprecated in favour of `eegDecode`.

Examples

```
pe <- make_eeg_bci(n_trials = 20, n_channels = 8, sr = 128, trial_sec = 3)
res <- eegDecode(pe, pipeline = "csp+lda", n_permutations = 0, seed = 1)
res$accuracy
```

eegDipoleFit

*Fit an equivalent current dipole (ECD) to an EEG topography***Description**

Localizes the single equivalent current dipole that best explains a scalp topography, by nonlinearly optimizing the dipole position (a grid search inside the head sphere refined with Nelder-Mead) while the dipole moment is the closed-form least-squares fit at each candidate position. Uses the same forward physics as [eegForwardModel\(\)](#) and the electrode positions from `colData` / the montage. This is the classic focal-source complement to the distributed inverses in [eegSourceEstimate\(\)](#).

Usage

```
eegDipoleFit(
  x,
  time = NULL,
  method = c("spherical", "bem_simplified"),
  n_grid = 8L,
  assay_name = NULL
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> .
<code>time</code>	Sample index to fit (a single index), a length-2 range to average over, or <code>NULL</code> (default) to fit the peak global-field-power sample.
<code>method</code>	Forward physics: "spherical" (single-sphere) or "bem_simplified" (Berg-Scherg 3-shell).
<code>n_grid</code>	Grid resolution per axis for the initial search (default 8).
<code>assay_name</code>	Assay to use (default: the object's default assay).

Value

An `eeg_dipole_fit` object: position (x, y, z in normalized head radius), moment, orientation (unit), amplitude, `gof` (goodness of fit, variance explained), residual, and the fitted/observed topographies.

References

Scherg (1990); Mosher, Lewis & Leahy (1992).

See Also

[eegSourceEstimate\(\)](#), [eegForwardModel\(\)](#)

Examples

```
pe <- make_eeg(n_time = 100, n_channels = 19, sr = 100)
fit <- eegDipoleFit(pe)
fit$gof
```

eegDTF

Directed Transfer Function (DTF)

Description

Estimates frequency-resolved directed connectivity between EEG channels with the Directed Transfer Function (Kaminski & Blinowska 1991), computed from the transfer function $H(f)$ of a multivariate autoregressive (MVAR) model fitted with `PhysioExperiment::mvarFit()`. Because $H(f)$ inverts the whole VAR system, DTF reflects both direct and indirect pathways. The (default) normalized DTF satisfies $\sum_j \text{DTF}_{ij}(f)^2 = 1$ for each target i (inflow normalization); `ffDTF` normalizes across all frequencies instead (Korzeniewska 2003).

Usage

```
eegDTF(
  x,
  order = NULL,
  freqs = NULL,
  normalized = TRUE,
  ffDTF = FALSE,
  band = NULL,
  method = "ols",
  assay_name = NULL
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with 2D EEG data (time x channels).
<code>order</code>	MVAR model order, or <code>NULL</code> to select automatically.
<code>freqs</code>	Numeric vector of frequencies in Hz (default: 128 points from 0 to the Nyquist frequency).
<code>normalized</code>	Row-normalize the DTF (default: <code>TRUE</code>).
<code>ffDTF</code>	Use the full-frequency DTF normalization (default: <code>FALSE</code>).
<code>band</code>	Optional numeric length-2 band in Hz over which to average the stored connectivity matrix (default: all frequencies).
<code>method</code>	MVAR estimator passed to <code>PhysioExperiment::mvarFit()</code> (default: <code>"ols"</code>).
<code>assay_name</code>	Input assay name (default: the default assay).

Value

The `PhysioExperiment` with `metadata(x)$connectivity` set to a list containing the band-averaged directed matrix (rows = targets, columns = sources), the frequency-resolved array (`n_channels` x `n_channels` x `n_freqs`), the frequencies, and the settings used.

References

Kaminski, M. J., & Blinowska, K. J. (1991). A new method of the description of the information flow in the brain structures. *Biological Cybernetics*, 65(3), 203-210.

See Also

`eegPDC()`, `eegConditionalGC()`, `eegConnectivityMatrix()`, `PhysioExperiment::mvarFit()`

Examples

```
pe <- make_eeg(n_time = 4000, n_channels = 5, sr = 250)
pe <- eegDTF(pe, order = 5)
dim(S4Vectors::metadata(pe)$connectivity$array)
```

eegEpoch

Epoch continuous EEG data

Description

Segments continuous 2D EEG data into fixed-length epochs around events, producing a 3D array (time x channels x epochs). Optionally performs baseline correction by subtracting the mean of a pre-stimulus window.

Usage

```
eegEpoch(
  x,
  events,
  limits = c(-0.2, 0.8),
  baseline = c(-0.2, 0),
  assay_name = NULL,
  output_assay = "epoched"
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with continuous (2D) data.
<code>events</code>	A <code>data.frame</code> with an <code>onset_sec</code> column specifying event times in seconds, OR an integer vector of sample indices.
<code>limits</code>	Numeric vector of length 2 specifying the epoch window relative to each event in seconds (default: <code>c(-0.2, 0.8)</code>).

baseline	Numeric vector of length 2 specifying the baseline window relative to each event in seconds (default: <code>c(-0.2, 0)</code>). Set to <code>NULL</code> to skip baseline correction.
assay_name	Name of the assay to epoch. If <code>NULL</code> , uses <code>defaultAssay(x)</code> .
output_assay	Name of the output assay (default: <code>"epoched"</code>).

Value

A `PhysioExperiment` object with a 3D array (time x channels x epochs) in the specified output assay. Event information is stored in `metadata(x)$epoch_events`.

Examples

```
pe <- make_eeg(n_time = 10000, n_channels = 19, sr = 500)
events <- data.frame(onset_sec = c(1.0, 3.0, 5.0, 7.0))
pe_ep <- eegEpoch(pe, events, limits = c(-0.2, 0.8))
# the epoched array is stored in metadata(x)$epoched
dim(S4Vectors::metadata(pe_ep)$epoched) # time x channels x epochs
```

eegERPbaseline

Baseline Correct ERP Data

Description

Subtracts the mean amplitude of a pre-stimulus baseline period from each epoch. This is an essential preprocessing step before ERP measurement.

Usage

```
eegERPbaseline(
  x,
  baseline = c(-200, 0),
  epoch_start = 0,
  assay_name = NULL,
  output_assay = "baseline_corrected"
)
```

Arguments

x	A <code>PhysioExperiment</code> object with epoched (3D) EEG data.
baseline	Baseline period in milliseconds as <code>c(start, end)</code> . For example, <code>c(-200, 0)</code> for 200ms pre-stimulus baseline. Default is <code>c(-200, 0)</code> .
epoch_start	Start time of the epoch in milliseconds relative to stimulus onset. Default is 0 (epoch starts at stimulus).
assay_name	Input assay name (default: first assay).
output_assay	Output assay name (default: <code>"baseline_corrected"</code>).

Value

Modified PhysioExperiment with baseline-corrected data in output_assay. Creates a new 3D assay (time x channels x epochs) where each epoch has had the mean of its baseline period subtracted.

References

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

See Also

[eegERPdetect\(\)](#), [eegERPmeasure\(\)](#), [eegEpoch\(\)](#), [eegFilter\(\)](#)

Examples

```
pe <- make_eeg_erp(n_epochs = 40, sr = 250)
result <- eegERPbaseline(pe, baseline = c(-200, 0), epoch_start = -200)
```

eegERPdetect	<i>Detect ERP Components</i>
--------------	------------------------------

Description

Detects known event-related potential (ERP) components in epoched (3D) EEG data. Averages across epochs and finds peaks within predefined time windows.

Usage

```
eegERPdetect(
  x,
  component = c("N100", "P300", "N400", "P600", "MMN", "LPP"),
  channels = NULL,
  epoch_start = 0,
  assay_name = NULL
)
```

Arguments

x	A PhysioExperiment object with epoched (3D) EEG data.
component	ERP component to detect: "N100", "P300", "N400", "P600", "MMN", or "LPP".
channels	Character vector of channel labels to analyze. If NULL, all channels are used.
epoch_start	Start time of the epoch in milliseconds relative to stimulus onset. Default is 0 (epoch starts at stimulus).
assay_name	Input assay name (default: first assay).

Value

A data.frame with columns: channel (character label), component (character name), latency_ms (numeric peak latency in milliseconds), and amplitude (numeric peak amplitude).

References

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

See Also

[eegERPmeasure\(\)](#), [eegERPlatency\(\)](#), [eegERPbaseline\(\)](#), [eegEpoch\(\)](#), [eegPlotERP\(\)](#)

Examples

```
pe <- make_eeg_erp(n_epochs = 40, sr = 250)
result <- eegERPdetect(pe, component = "P300")
```

eegERPdifference	<i>ERP Difference Waveform</i>
------------------	--------------------------------

Description

Computes the difference waveform between two PhysioExperiment objects by subtracting the assay data of y from x.

Usage

```
eegERPdifference(x, y, assay_name = NULL, output_assay = "difference")
```

Arguments

x	A PhysioExperiment object (minuend).
y	A PhysioExperiment object (subtrahend). Must have the same dimensions as x.
assay_name	Input assay name (default: first assay).
output_assay	Output assay name (default: "difference").

Value

Modified x with difference waveform stored in output_assay. The difference assay has the same dimensions as the input assay.

References

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

See Also

[eegERPdetect\(\)](#), [eegERPmeasure\(\)](#), [eegERPtest\(\)](#), [eegERPgrandAverage\(\)](#)

Examples

```
pe1 <- make_eeg_erp(n_epochs = 20, sr = 250)
pe2 <- make_eeg_erp(n_epochs = 20, sr = 250)
result <- eegERPdifference(pe1, pe2)
```

eegERPgrandAverage	<i>Compute Grand Average ERP</i>
--------------------	----------------------------------

Description

Averages ERP waveforms across multiple `PhysioExperiment` objects (participants). Each input should already be averaged across trials.

Usage

```
eegERPgrandAverage(..., assay_name = NULL, output_assay = "grand_average")
```

Arguments

<code>...</code>	PhysioExperiment objects to average, or a list of them.
<code>assay_name</code>	Input assay name (default: first assay).
<code>output_assay</code>	Output assay name (default: "grand_average").

Value

A `PhysioExperiment` (the first input object) with the grand average waveform stored in `output_assay`. The grand average assay has the same dimensions as the input assays.

References

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

See Also

[eegERPdetect\(\)](#), [eegERPmeasure\(\)](#), [eegERPtest\(\)](#), [eegERPdifference\(\)](#)

Examples

```
pe1 <- make_eeg_erp(n_epochs = 20, sr = 250)
pe2 <- make_eeg_erp(n_epochs = 20, sr = 250)
result <- eegERPgrandAverage(pe1, pe2)
```

eegERPlatency	<i>Fractional Area Latency</i>
---------------	--------------------------------

Description

Computes the fractional area latency of an ERP component. This is the time point at which a specified fraction of the total area under the curve (in the given window) has accumulated.

Usage

```
eegERPlatency(
  x,
  window,
  fraction = 0.5,
  polarity = c("positive", "negative"),
  epoch_start = 0,
  assay_name = NULL
)
```

Arguments

x	A PhysioExperiment object with EEG data.
window	Numeric vector of length 2: c(start_ms, end_ms).
fraction	Fraction of total area (default: 0.5 for median latency).
polarity	Expected polarity: "positive" or "negative".
epoch_start	Start time of the epoch in milliseconds relative to stimulus onset. Default is 0 (epoch starts at stimulus).
assay_name	Input assay name (default: first assay).

Value

A data.frame with columns: channel (character label), latency_ms (numeric fractional area latency in milliseconds), and fraction (numeric fraction used).

References

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

See Also

[eegERPdetect\(\)](#), [eegERPmeasure\(\)](#), [eegERPbaseline\(\)](#), [eegPlotERP\(\)](#)

Examples

```
pe <- make_eeg_erp(n_epochs = 40, sr = 250)
result <- eegERPlatency(pe, window = c(250, 500), fraction = 0.5,
  polarity = "positive")
```

eegERPmeasure	<i>Measure ERP Amplitude</i>
---------------	------------------------------

Description

Measures ERP amplitude in a specified time window using peak, mean, or adaptive mean methods. For epoched (3D) data, averages across epochs first.

Usage

```
eegERPmeasure(
  x,
  window,
  method = c("peak", "mean", "adaptive_mean"),
  polarity = c("positive", "negative"),
  epoch_start = 0,
  assay_name = NULL
)
```

Arguments

x	A PhysioExperiment object with EEG data.
window	Numeric vector of length 2: c(start_ms, end_ms).
method	Measurement method: "peak" (peak amplitude), "mean" (mean amplitude), or "adaptive_mean" (mean in +/-25ms around peak).
polarity	Expected polarity: "positive" or "negative".
epoch_start	Start time of the epoch in milliseconds relative to stimulus onset. Default is 0 (epoch starts at stimulus).
assay_name	Input assay name (default: first assay).

Value

A data.frame with columns: channel (character label), amplitude (numeric measured amplitude), latency_ms (numeric latency in milliseconds), and method (character measurement method used).

References

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

See Also

[eegERPdetect\(\)](#), [eegERPlatency\(\)](#), [eegERPbaseline\(\)](#), [eegPlotERP\(\)](#)

Examples

```
pe <- make_eeg_erp(n_epochs = 40, sr = 250)
result <- eegERPmeasure(pe, window = c(250, 500), method = "peak",
  polarity = "positive")
```

Description

Performs permutation testing or cluster-based permutation testing to compare ERP waveforms between conditions.

Usage

```
eegERPtest(  
  x,  
  y,  
  method = c("permutation", "cluster"),  
  n_perm = 1000,  
  alpha = 0.05,  
  cluster_alpha = 0.05,  
  assay_name = NULL  
)
```

Arguments

x	A PhysioExperiment with epoched data for condition 1.
y	A PhysioExperiment with epoched data for condition 2.
method	Test method: "permutation" (pointwise permutation test) or "cluster" (cluster-based permutation test).
n_perm	Number of permutations (default: 1000).
alpha	Significance level (default: 0.05).
cluster_alpha	Cluster-forming threshold for individual t-tests (default: 0.05). Only used for "cluster" method.
assay_name	Input assay name (default: first assay).

Value

A data.frame with columns: time_sample (integer), t_statistic (numeric observed t-value), p_value (numeric permutation-based p-value), and significant (logical). For the "cluster" method, also includes cluster_id (integer cluster assignment) and cluster_p (numeric cluster-level corrected p-value).

References

Luck, S. J. (2014). An Introduction to the Event-Related Potential Technique (2nd ed.). MIT Press.
Maris, E., & Oostenveld, R. (2007). Nonparametric statistical testing of EEG- and MEG-data. Journal of Neuroscience Methods, 164(1), 177-190.

See Also

[eegERPdetect\(\)](#), [eegERPmeasure\(\)](#), [eegERPdifference\(\)](#), [eegERPgrandAverage\(\)](#)

Examples

```
pe1 <- make_eeg_erp(n_epochs = 20, sr = 250)
pe2 <- make_eeg_erp(n_epochs = 20, sr = 250)
result <- eegERPtest(pe1, pe2, method = "permutation", n_perm = 500)
```

eegERSP

*Event-Related Spectral Perturbation (ERSP)***Description**

Computes Event-Related Spectral Perturbation for epoched (3D) EEG data. ERSR quantifies event-related changes in spectral power relative to a baseline period, expressed in decibels (dB). Uses the Morlet wavelet transform to compute time-frequency decomposition for each epoch, then averages power across epochs and normalizes to baseline.

Usage

```
eegERSP(
  x,
  baseline = c(1, 50),
  frequencies = NULL,
  n_cycles = 7,
  assay_name = NULL,
  output_assay = "ersp_data"
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with epoched (3D) EEG data (time x channels x epochs).
<code>baseline</code>	Numeric vector of length 2 specifying the baseline time window in sample indices (e.g., <code>c(1, 50)</code> for the first 50 samples). Default is <code>c(1, 50)</code> .
<code>frequencies</code>	Numeric vector of frequencies in Hz to analyze. If <code>NULL</code> , defaults to <code>seq(1, 50, by = 1)</code> .
<code>n_cycles</code>	Number of cycles for the Morlet wavelet (default: 7).
<code>assay_name</code>	Name of the input assay. If <code>NULL</code> , the default assay is used.
<code>output_assay</code>	Name the result is stored under. Despite the argument's name this is a key in metadata , NOT an assay: after the call <code>assay(x, "the value given")</code> does not exist while <code>metadata(x)\$the value given</code> holds the result (default: "the value given"). (default: "ersp").

Value

Modified PhysioExperiment with:

- 3D ERSP array (time x frequencies x channels) in dB in output_assay
- Baseline info and frequency vector in metadata(x)\$ersp, a list containing frequencies (numeric vector), baseline (numeric vector), n_cycles (integer), and n_epochs (integer)

References

Tallon-Baudry, C., et al. (1997). Oscillatory gamma-band activity during conscious perception. *Trends in Cognitive Sciences*, 3(4), 151-162.

Makeig, S. (1993). Auditory event-related dynamics of the EEG spectrum and effects of exposure to tones. *Electroencephalography and Clinical Neurophysiology*, 86(4), 283-293.

See Also

[eegITC\(\)](#), [eegMorletWavelet\(\)](#), [eegPlotSpectrogram\(\)](#)

Examples

```
pe <- make_eeg_erp(n_epochs = 20, n_channels = 2, sr = 250)
pe_ersp <- eegERSP(pe, baseline = c(1, 50), frequencies = seq(5, 30, by = 5))
# results are stored in metadata(x)$ersp
str(S4Vectors::metadata(pe_ersp)$ersp, max.level = 1)
```

eegFilter

Filter EEG signals

Description

Applies frequency-domain filtering to EEG data stored in a PhysioExperiment object. Supports bandpass, highpass, lowpass, and notch filtering using either FIR (windowed-sinc) or IIR (Butterworth) methods.

Usage

```
eegFilter(
  x,
  lowcut = NULL,
  highcut = NULL,
  notch = NULL,
  method = c("fir", "iir"),
  order = NULL,
  assay_name = NULL,
  output_assay = "filtered"
)
```

Arguments

x	A <code>PhysioExperiment</code> object.
lowcut	Low cutoff frequency in Hz. If <code>NULL</code> , no highpass filtering is applied (set both lowcut and highcut for bandpass).
highcut	High cutoff frequency in Hz. If <code>NULL</code> , no lowpass filtering is applied (set both lowcut and highcut for bandpass).
notch	Notch filter center frequency in Hz (e.g., 50 or 60 for powerline noise). If <code>NULL</code> , no notch filter is applied. Bandwidth is +/- 2 Hz around center.
method	Filtering method: "fir" (default) for windowed-sinc FIR filter, or "iir" for Butterworth IIR filter.
order	Filter order. For FIR, this is the number of taps (auto-selected if <code>NULL</code>). For IIR, this is the Butterworth order (default: 4).
assay_name	Name of the assay to filter. If <code>NULL</code> , uses <code>defaultAssay(x)</code> .
output_assay	Name of the output assay (default: "filtered").

Details

For FIR mode, the function uses windowed-sinc filters with a Hamming window. For IIR mode, zero-phase Butterworth filtering is applied via `signal::filtfilt()`, with automatic fallback to FIR if the signal package is not available.

Value

A `PhysioExperiment` object with filtered data in the specified output assay.

Examples

```
pe <- make_eeg(n_time = 5000, n_channels = 19, sr = 500)
# Bandpass filter
pe_filt <- eegFilter(pe, lowcut = 1, highcut = 40)
# Highpass only
pe_hp <- eegFilter(pe, lowcut = 0.1)
# With notch at 50 Hz
pe_notch <- eegFilter(pe, lowcut = 1, highcut = 40, notch = 50)
# IIR Butterworth
pe_iir <- eegFilter(pe, lowcut = 1, highcut = 40, method = "iir", order = 4)
```

eegForwardModel1

Construct EEG Forward Model (Leadfield Matrix)

Description

Constructs a leadfield matrix that maps brain source activity to scalp electrode potentials. Uses electrode positions from `colData(x)` if available, otherwise falls back to standard 10-20 system positions on a unit sphere.

Usage

```
eegForwardModel(
  x,
  method = c("spherical", "bem_simplified", "bem", "sphere_analytic", "nyhead",
    "fsaverage"),
  n_sources = 500,
  assay_name = NULL
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with EEG data.
<code>method</code>	Forward model method: "spherical" (current dipole in infinite homogeneous medium with conductivity 0.33 S/m), "bem_simplified" (3-shell Berg and Scherg 1994 approximation), "sphere_analytic" (the analytic single-sphere leadfield over a structured spherical cortical source space), "bem" (a 3-shell boundary element method over the same source space), or "nyhead" / "fsaverage" (published head models loaded on demand). The last four require the PhysioHeadModels package; the "nyhead" / "fsaverage" methods additionally require the corresponding dataset to have been downloaded (see <code>PhysioHeadModels::fetchNYHead</code>).
<code>n_sources</code>	Number of dipole sources to distribute inside the head model (default: 500).
<code>assay_name</code>	Name of the assay to reference for channel count. If <code>NULL</code> , the default assay is used.

Value

A list with components:

leadfield Numeric matrix of dimensions `n_electrodes` x (`n_sources` * 3). Each source has 3 orientation columns (x, y, z).

source_positions Data frame with columns x, y, z for each source location.

electrode_positions Data frame with columns label, x, y, z for each electrode.

n_sources Integer number of source dipoles.

source_normals For the realistic methods, an `n_sources` x 3 matrix of cortical surface normals (absent for the legacy methods).

method The resolved forward-model method.

coordinate_provenance Source-space origin, units, and anatomical fidelity used by coordinate-aware plotting functions.

References

Pascual-Marqui, R. D. (2002). Standardized low-resolution brain electromagnetic tomography (sLORETA). *Methods and Findings in Experimental and Clinical Pharmacology*, 24(Suppl D), 5-12.

See Also

`eegSourceEstimate()`, `eegBeamformer()`, `eegSourcePower()`

Examples

```
pe <- make_eeg(n_time = 1000, n_channels = 19, sr = 250)
fm <- eegForwardModel(pe, method = "spherical", n_sources = 100)
dim(fm$leadfield)
```

eegGrangerCausality *EEG Spectral Granger Causality*

Description

Computes spectral Granger causality between all EEG channel pairs by fitting bivariate autoregressive (AR) models using Yule-Walker equations and computing the transfer function in the frequency domain. Unlike coherence-based measures, Granger causality is directional: GC from channel A to channel B is generally different from GC from B to A.

Usage

```
eegGrangerCausality(x, order = 5, band = NULL, assay_name = NULL)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with EEG data (2D: time x channels).
<code>order</code>	Integer AR model order for the bivariate model (default: 5).
<code>band</code>	Numeric vector of length 2 specifying the frequency band in Hz over which to average GC. If <code>NULL</code> , returns the average over all positive frequencies.
<code>assay_name</code>	Name of the input assay. If <code>NULL</code> , the default assay is used.

Details

The spectral Granger causality from channel x to channel y at frequency f is defined as:

$$GC_{x \rightarrow y}(f) = \log(S_{yy}(f)/(S_{yy}(f) - |H_{xy}(f)|^2 \cdot \Sigma_{xx}))$$

Value

A `data.frame` with columns:

from_channel Character or integer identifier of the source channel.

to_channel Character or integer identifier of the target channel.

gc_value Numeric Granger causality value (≥ 0). Higher values indicate stronger directed influence.

References

- Lachaux, J. P., et al. (1999). Measuring phase synchrony in brain signals. *Human Brain Mapping*, 8(4), 194-208.
- Granger, C. W. J. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, 37(3), 424-438.

See Also

[eegCoherence\(\)](#), [eegPLV\(\)](#), [eegWPLI\(\)](#), [eegConnectivityMatrix\(\)](#)

Examples

```
pe <- make_eeg(n_time = 5000, n_channels = 4, sr = 500)
gc_df <- eegGrangerCausality(pe, order = 5, band = c(8, 13))
head(gc_df)
```

eegICA

EEG Independent Component Analysis (ICA)

Description

Decomposes multi-channel EEG into independent components using ICA. Supports FastICA, Infomax, and JADE algorithms. Results are stored in the output assay (component activations) and in `metadata(x)$ica` (mixing and unmixing matrices).

Usage

```
eegICA(
  x,
  n_components = NULL,
  method = c("fastica", "infomax", "jade"),
  max_iter = 200L,
  tol = 1e-06,
  assay_name = NULL,
  output_assay = "ica_components"
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with EEG data.
<code>n_components</code>	Number of independent components to extract. Defaults to the number of channels.
<code>method</code>	ICA algorithm: "fastica", "infomax", or "jade".
<code>max_iter</code>	Maximum number of iterations (default: 200).
<code>tol</code>	Convergence tolerance (default: 1e-6).
<code>assay_name</code>	Input assay name (default: first assay).
<code>output_assay</code>	Output assay name (default: "ica").

Value

Modified `PhysioExperiment` with component activations in `output_assay` and ICA metadata in `metadata(x)$ica`. The ICA metadata list contains: `mixing` (mixing matrix `A`), `unmixing` (unmixing matrix), `mean` (channel means), `whiten` (whitening matrix), and `method` (algorithm used). The output assay has dimensions `n_time` x `n_components`.

References

Hyvarinen, A., & Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural Networks*, 13(4-5), 411-430.

Bell, A. J., & Sejnowski, T. J. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6), 1129-1159.

See Also

[eegICAremove\(\)](#), [eegICAdetect\(\)](#), [eegICAmix\(\)](#), [eegFilter\(\)](#), [eegPreprocess\(\)](#)

Examples

```
pe <- make_eeg(n_time = 5000, sr = 500)
result <- eegICA(pe, n_components = 4, method = "fastica")
```

eegICAdetect

Detect Artifact ICA Components

Description

Automatically identifies artifact components using one of four methods: correlation with frontal channels, kurtosis, spatial weight pattern, or the `ICLabel`-style seven-class classifier ([eegICLabel\(\)](#)). The `"iclable"` method delegates to [eegICLabel\(\)](#) and collapses its result to the two-class contract (any non-brain argmax is an "artifact"), with the score being one minus the brain probability.

Usage

```
eegICAdetect(
  x,
  method = c("correlation", "kurtosis", "spatial", "iclable"),
  threshold = 0.3,
  ica_assay = "ica_components"
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with ICA results (from <code>eegICA</code>).
<code>method</code>	Detection method: <code>"correlation"</code> (frontal channel correlation), <code>"kurtosis"</code> (excess kurtosis), <code>"spatial"</code> (spatial weight pattern), or <code>"iclable"</code> (seven-class classifier).

threshold	Threshold for artifact detection. For "correlation", absolute correlation > threshold marks artifact (default: 0.3).
ica_assay	Assay name containing ICA activations (default: "ica").

Value

A data.frame with columns: component (integer index), type ("artifact" or "neural"), method (detection method used), and score (numeric detection score).

References

Hyvarinen, A., & Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural Networks*, 13(4-5), 411-430.

Bell, A. J., & Sejnowski, T. J. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6), 1129-1159.

See Also

[eegICA\(\)](#), [eegICAremove\(\)](#), [eegICAmix\(\)](#)

Examples

```
pe <- make_eeg(n_time = 5000, sr = 500)
pe <- eegICA(pe, n_components = 4, method = "fastica")
artifacts <- eegICAdetect(pe, method = "kurtosis")
```

eegICAmix	<i>Access ICA Metadata</i>
-----------	----------------------------

Description

Returns the ICA results stored in `metadata(x)$ica`, including the mixing matrix, unmixing matrix, channel means, and whitening matrix.

Usage

```
eegICAmix(x)
```

Arguments

`x` A `PhysioExperiment` object with ICA results.

Value

A list containing: `mixing` (mixing matrix `A`, `n_channels` x `n_components`), `unmixing` (unmixing matrix, `n_components` x `n_channels`), `mean` (channel mean vector), `whiten` (whitening matrix), and `method` (character, algorithm used).

References

- Hyvarinen, A., & Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural Networks*, 13(4-5), 411-430.
- Bell, A. J., & Sejnowski, T. J. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6), 1129-1159.

See Also

[eegICA\(\)](#), [eegICAremove\(\)](#), [eegICAdetect\(\)](#)

Examples

```
pe <- make_eeg(n_time = 5000, sr = 500)
pe <- eegICA(pe, n_components = 4, method = "fastica")
ica_info <- eegICAmix(pe)
```

eegICAremove

Remove ICA Components from EEG

Description

Reconstructs EEG data with specified independent components removed. The removed components are zeroed out in the mixing matrix before back-projecting to channel space.

Usage

```
eegICAremove(
  x,
  components,
  ica_assay = "ica_components",
  output_assay = "ica_cleaned"
)
```

Arguments

x	A <code>PhysioExperiment</code> object with ICA results (from <code>eegICA</code>).
components	Integer vector of component indices to remove.
ica_assay	Assay name containing ICA component activations (default: "ica").
output_assay	Output assay name (default: "ica_cleaned").

Value

Modified `PhysioExperiment` with cleaned data in `output_assay`. The cleaned assay contains reconstructed channel data with specified components removed. Dimensions match the original data.

References

- Hyvarinen, A., & Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural Networks*, 13(4-5), 411-430.
- Bell, A. J., & Sejnowski, T. J. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6), 1129-1159.

See Also

[eegICA\(\)](#), [eegICAdetect\(\)](#), [eegICAmix\(\)](#)

Examples

```
pe <- make_eeg(n_time = 5000, sr = 500)
pe <- eegICA(pe, n_components = 4, method = "fastica")
pe <- eegICAremove(pe, components = c(1, 2))
```

eegICLabel

ICLabel-style ICA Component Classification

Description

Classifies independent components (from [eegICA\(\)](#)) into seven classes - brain, muscle, eye, heart, line_noise, channel_noise, and other - and returns a calibrated probability for each class per component together with the argmax label. The approach follows the ICLabel framework: interpretable spatial, spectral, and temporal features are extracted from each component and mapped to class probabilities by a lightweight multinomial-logistic (softmax) head.

Usage

```
eegICLabel(
  x,
  ica_assay = "ica_components",
  assay_name = NULL,
  line_freq = 50,
  backend = c("heuristic", "iclabel")
)
```

Arguments

- | | |
|------------|--|
| x | A PhysioExperiment object with ICA results (from eegICA()). |
| ica_assay | Metadata name holding the component activations (default: "ica_components"). |
| assay_name | Input assay used only to resolve channel labels (default: first assay). |
| line_freq | Mains line frequency in Hz used for the line-noise features (default: 50). Only used by the heuristic backend. |

backend Which classifier to use: "heuristic" (default) is the self-contained pure-R classifier (real features, hand-set multinomial weights); "iclabel" runs the genuine trained ICLabel CNN (Pion-Tonachini et al. 2019) by delegating to **mne-icalabel** via **reticulate** (needs a Python env with mne and mne-icalabel; channel labels must match a standard 10-20/10-10 montage for the scalp-map).

Details

Features per component:

- Spatial: frontal energy (eye/blink topography), focality (single channel dominance), topography kurtosis.
- Spectral: 1/f slope, high- versus low-frequency band ratio, low-frequency fraction, a genuine alpha-peak measure (8-12 Hz power above its theta/low-beta neighbours), and a line-noise power ratio and fraction.
- Temporal: lag-1 autocorrelation, activation kurtosis (spiky blink or ECG signatures), and a roughly 1 Hz periodicity measure for cardiac components.

The softmax weights are read from `inst/extdata/iclabel_weights.csv`; if that file is unavailable an identical built-in weight table is used, so the classifier always works.

Value

A data.frame with one row per component: `component` (integer index), one numeric column per class (brain, muscle, eye, heart, line_noise, channel_noise, other) holding probabilities that sum to 1 across the classes, and `label` (character argmax class).

References

Pion-Tonachini, L., Kreutz-Delgado, K., & Makeig, S. (2019). ICLabel: An automated electroencephalographic independent component classifier, dataset, and website. *NeuroImage*, 198, 181-197.

Winkler, I., Haufe, S., & Tangermann, M. (2011). Automatic classification of artifactual ICA-components for artifact removal in EEG signals. *Behavioral and Brain Functions*, 7, 30.

See Also

[eegICA\(\)](#), [eegICLabelFlag\(\)](#), [eegICAdetect\(\)](#), [eegICAremove\(\)](#)

Examples

```
pe <- make_eeg(n_time = 5000, sr = 500)
pe <- eegICA(pe, n_components = 10, method = "fastica")
probs <- eegICLabel(pe)
```

eegICLabelFlag

*Flag Artifact ICA Components from ICLabel Probabilities***Description**

Convenience wrapper over `eegICLabel()` that returns the indices of the components most likely to be artifacts, ready to pass to `eegICAremove()`. A component is flagged when its probability in any of the requested artifact classes exceeds `prob_threshold`.

Usage

```
eegICLabelFlag(
  x,
  prob_threshold = 0.5,
  classes = c("muscle", "eye", "heart", "line_noise", "channel_noise"),
  ica_assay = "ica_components",
  line_freq = 50
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with ICA results (from <code>eegICA()</code>).
<code>prob_threshold</code>	Probability above which a component is flagged (default: 0.5).
<code>classes</code>	Character vector of artifact classes to flag (default: all non-brain, non-other classes).
<code>ica_assay</code>	Metadata name holding the component activations (default: "ica_components").
<code>line_freq</code>	Mains line frequency in Hz (default: 50).

Value

An integer vector of flagged component indices (possibly empty), with an attribute "labels" giving the corresponding class of each.

See Also

`eegICLabel()`, `eegICAremove()`, `eegICAdetect()`

Examples

```
pe <- make_eeg(n_time = 5000, sr = 500)
pe <- eegICA(pe, n_components = 10, method = "fastica")
bad <- eegICLabelFlag(pe, prob_threshold = 0.5)
pe <- eegICAremove(pe, components = bad)
```

eegInterpolate	<i>Interpolate bad EEG channels</i>
----------------	-------------------------------------

Description

Replaces data in bad channels by interpolating from remaining good channels using either spherical spline interpolation (Perrin et al., 1989) or nearest-neighbor weighted averaging.

Usage

```
eegInterpolate(
  x,
  bad_channels,
  method = c("spline", "nearest"),
  assay_name = NULL,
  output_assay = "interpolated"
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object.
<code>bad_channels</code>	Character vector of channel labels to interpolate.
<code>method</code>	Interpolation method: "spline" for spherical spline (default) or "nearest" for inverse-distance weighted nearest neighbors.
<code>assay_name</code>	Name of the assay to interpolate. If <code>NULL</code> , uses <code>defaultAssay(x)</code> .
<code>output_assay</code>	Name of the output assay (default: "interpolated").

Details

Requires electrode positions (`pos_x`, `pos_y`, `pos_z`) in `colData`. Apply [eegMontage](#) first if positions are not set.

Value

A `PhysioExperiment` object with interpolated channels in the specified output assay.

Examples

```
pe <- make_eeg(n_time = 5000, n_channels = 19, sr = 500)
pe <- eegMontage(pe, system = "10-20")
# interpolate two named channels from their neighbours
pe_clean <- eegInterpolate(pe, bad_channels = c("C3", "C4"))
pe_clean
```

eegITC

*Inter-Trial Coherence (ITC)***Description**

Computes Inter-Trial Coherence (also known as phase-locking factor or phase-locking value) for epoched (3D) EEG data. ITC measures the consistency of oscillatory phase across trials at each time-frequency point. Values range from 0 (completely random phase) to 1 (perfectly phase-locked across all trials).

Usage

```
eegITC(
  x,
  frequencies = NULL,
  n_cycles = 7,
  assay_name = NULL,
  output_assay = "itc_data"
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with epoched (3D) EEG data (time x channels x epochs).
<code>frequencies</code>	Numeric vector of frequencies in Hz to analyze. If <code>NULL</code> , defaults to <code>seq(1, 50, by = 1)</code> .
<code>n_cycles</code>	Number of cycles for the Morlet wavelet (default: 7).
<code>assay_name</code>	Name of the input assay. If <code>NULL</code> , the default assay is used.
<code>output_assay</code>	Name the result is stored under. Despite the argument's name this is a key in metadata , NOT an assay: after the call <code>assay(x, "the value given")</code> does not exist while <code>metadata(x)\$the value given</code> holds the result (default: "the value given"). (default: "itc").

Value

Modified `PhysioExperiment` with:

- 3D ITC array (time x frequencies x channels) in `output_assay`, values in `[0, 1]`
- Frequency vector and parameters in `metadata(x)$itc`, a list containing frequencies (numeric vector), `n_cycles` (integer), and `n_epochs` (integer)

References

Tallon-Baudry, C., et al. (1997). Oscillatory gamma-band activity during conscious perception. *Trends in Cognitive Sciences*, 3(4), 151-162.

Makeig, S. (1993). Auditory event-related dynamics of the EEG spectrum and effects of exposure to tones. *Electroencephalography and Clinical Neurophysiology*, 86(4), 283-293.

See Also

[eegERSP\(\)](#), [eegMorletWavelet\(\)](#), [eegPlotSpectrogram\(\)](#)

Examples

```
pe <- make_eeg_erp(n_epochs = 20, n_channels = 2, sr = 250)
pe_itc <- eegITC(pe, frequencies = seq(5, 30, by = 5))
# results are stored in metadata(x)$itc
str(S4Vectors::metadata(pe_itc)$itc, max.level = 1)
```

eegKcomplexDetect	<i>Detect K-Complexes</i>
-------------------	---------------------------

Description

Identifies K-complexes in EEG data by lowpass filtering at 4 Hz, finding negative peaks exceeding a threshold amplitude, and verifying the characteristic negative-positive waveform morphology.

Usage

```
eegKcomplexDetect(
  x,
  min_neg_amplitude = 75,
  min_duration_ms = 500,
  max_duration_ms = 1500,
  assay_name = NULL
)
```

Arguments

x	A <code>PhysioExperiment</code> object with EEG data.
min_neg_amplitude	Minimum absolute negative peak amplitude in microvolts (default: 75). Peaks must be more negative than <code>-min_neg_amplitude</code> .
min_duration_ms	Minimum K-complex duration in milliseconds (default: 500).
max_duration_ms	Maximum K-complex duration in milliseconds (default: 1500).
assay_name	Input assay name. If <code>NULL</code> , uses the default assay.

Value

A `data.frame` with columns:

channel Integer channel index.

negative_peak_sample Integer sample of the negative peak.

positive_peak_sample Integer sample of the positive peak.
negative_amplitude Numeric amplitude at the negative peak.
positive_amplitude Numeric amplitude at the positive peak.
duration_ms Numeric total duration in milliseconds.

References

Berry, R. B., et al. (2017). AASM Scoring Manual Updates for 2017. Journal of Clinical Sleep Medicine, 13(5), 665-666.

See Also

[eegSleepStage\(\)](#), [eegSpindleDetect\(\)](#), [eegSlowWaveDetect\(\)](#), [eegSleepMetrics\(\)](#)

Examples

```
pe <- make_eeg_sleep(n_time = 150000, n_channels = 2, sr = 500)
kcomplexes <- eegKcomplexDetect(pe)
head(kcomplexes)
```

eegLateralization	<i>Motor-Imagery Lateralization Index (hemispheric ERD/ERS)</i>
-------------------	---

Description

Quantifies the hemispheric asymmetry of sensorimotor-rhythm activity during motor imagery, a rehabilitation-relevant index of affected- versus unaffected-hemisphere engagement. For each trial a laterality index $LI = (R - L) / (|R| + |L|)$ is computed from a right- and a left-hemisphere channel (or ROI), bounded from -1 to 1. With method = "erd" the activation is the event-related desynchronization magnitude from [eegMotorImagery\(\)](#) (a positive LI means the right hemisphere desynchronizes more, i.e. is more active); with method = "power" it is the raw band power (a positive LI means the right hemisphere has more power). A symmetric input gives $LI \approx 0$.

Usage

```
eegLateralization(
  x,
  left_ch = "C3",
  right_ch = "C4",
  band = c(8, 13),
  method = c("erd", "power"),
  baseline_fraction = 0.25,
  conf_level = 0.95,
  reliability = list(icc = NA_real_, sem = NA_real_),
  assay_name = NULL
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with epoched (3D) EEG data (time x channels x trials).
<code>left_ch</code>	Character label(s) of the left-hemisphere channel or ROI (default: "C3").
<code>right_ch</code>	Character label(s) of the right-hemisphere channel or ROI (default: "C4").
<code>band</code>	Numeric length-2 frequency band in Hz (default: c(8, 13), the mu rhythm).
<code>method</code>	Activation measure: "erd" (desynchronization magnitude, via eegMotorImagery()) or "power" (raw band power).
<code>baseline_fraction</code>	Fraction of each trial used as the ERD baseline (default: 0.25); only used by <code>method = "erd"</code> .
<code>conf_level</code>	Confidence level for the summary interval (default: 0.95).
<code>reliability</code>	Named list of reliability indices attached to the summary biomarker (default: <code>icc</code> and <code>sem</code> placeholders).
<code>assay_name</code>	Input assay name (default: the default assay).

Value

A list with `per_trial` (an `n_trials` x 1 matrix of laterality indices) and `summary` (a [PhysioBiomarker](#) holding the mean LI with its confidence interval, reliability, and provenance).

References

- Pfurtscheller, G., & Lopes da Silva, F. H. (1999). Event-related EEG/MEG synchronization and desynchronization: basic principles. *Clinical Neurophysiology*, 110(11), 1842-1857.
- Bai, O., et al. (2005). Asymmetric spatiotemporal patterns of event-related desynchronization preceding voluntary sequential finger movements. *Clinical Neurophysiology*, 116(5), 1213-1221.

See Also

[eegMotorImagery\(\)](#), [eegCSP\(\)](#), [PhysioExperiment::physioBiomarker\(\)](#)

Examples

```
pe <- make_eeg_bci(n_trials = 20, n_channels = 8, sr = 256)
li <- eegLateralization(pe, method = "power")
li$summary
```

eegMicrostateBackfit *Backfit Microstate Template Maps to EEG Data*

Description

Assigns each time point to the microstate template map with the highest absolute spatial correlation. This allows applying microstate maps derived from one dataset to another dataset.

Usage

```
eegMicrostateBackfit(x, maps, assay_name = NULL)
```

Arguments

x	A PhysioExperiment object with EEG data.
maps	Numeric matrix of microstate template maps (n_channels x n_states), as returned in metadata(x)\$microstates\$maps.
assay_name	Name of the input assay. If NULL, the default assay is used.

Value

Modified PhysioExperiment with updated microstate labels in metadata(x)\$microstates\$labels. Also stores the template maps in metadata(x)\$microstates\$maps and the number of states in metadata(x)\$microstates\$n_states.

References

Michel, C. M., & Koenig, T. (2018). EEG microstates as a tool for studying the temporal dynamics of whole-brain neuronal networks. *NeuroImage*, 180, 577-593.

See Also

[eegMicrostates\(\)](#), [eegMicrostateStats\(\)](#), [eegMicrostateSequence\(\)](#)

Examples

```
pe1 <- make_eeg(n_time = 5000, n_channels = 19, sr = 500)
pe1 <- eegMicrostates(pe1, n_states = 4)
maps <- S4Vectors::metadata(pe1)$microstates$maps

pe2 <- make_eeg(n_time = 3000, n_channels = 19, sr = 500)
pe2 <- eegMicrostateBackfit(pe2, maps)
```

eegMicrostates	<i>EEG Microstate Segmentation</i>
----------------	------------------------------------

Description

Performs microstate analysis on EEG data by identifying dominant scalp topographies at Global Field Power (GFP) peaks and assigning each time point to the best-matching microstate map. Supports polarity-invariant K-means, atomize-and-agglomerate hierarchical clustering (AAHC), and PCA-based extraction.

Usage

```
eegMicrostates(
  x,
  n_states = 4,
  method = c("kmeans", "aahc", "pca"),
  min_gfp = 1,
  assay_name = NULL
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with EEG data.
<code>n_states</code>	Number of microstate classes to extract (default: 4).
<code>method</code>	Clustering method: "kmeans" (polarity-invariant K-means), "aahc" (atomize and agglomerate hierarchical clustering), or "pca" (principal component analysis).
<code>min_gfp</code>	Percentile threshold (0-100) for GFP peak selection (default: 1.0). Only GFP peaks above this percentile are used for clustering.
<code>assay_name</code>	Name of the input assay. If NULL, the default assay is used.

Value

Modified `PhysioExperiment` with microstate results stored in `metadata(x)$microstates`, a list containing:

maps Numeric matrix of dimensions `n_channels` x `n_states`, each column a microstate topography.

labels Integer vector of length `n_time`, microstate assignment (1 to `n_states`) for each time point.

gfp Numeric vector of GFP values per time point.

n_states Integer number of microstate classes.

References

Michel, C. M., & Koenig, T. (2018). EEG microstates as a tool for studying the temporal dynamics of whole-brain neuronal networks. *NeuroImage*, 180, 577-593.

See Also

[eegMicrostateStats\(\)](#), [eegMicrostateBackfit\(\)](#), [eegMicrostateSequence\(\)](#)

Examples

```
pe <- make_eeg(n_time = 5000, n_channels = 19, sr = 500)
pe <- eegMicrostates(pe, n_states = 4, method = "kmeans")
ms <- S4Vectors::metadata(pe)$microstates
```

eegMicrostateSequence *Extract Microstate Sequence as Character Labels*

Description

Converts integer microstate labels to character labels ("A", "B", "C", ...).

Usage

```
eegMicrostateSequence(x)
```

Arguments

x A `PhysioExperiment` object with microstate labels in `metadata(x)$microstates$labels`.

Value

Character vector of length `n_time` with labels "A", "B", "C", etc. Each element corresponds to the microstate class assigned to that time point.

References

Michel, C. M., & Koenig, T. (2018). EEG microstates as a tool for studying the temporal dynamics of whole-brain neuronal networks. *NeuroImage*, 180, 577-593.

See Also

[eegMicrostates\(\)](#), [eegMicrostateStats\(\)](#), [eegMicrostateBackfit\(\)](#)

Examples

```
pe <- make_eeg(n_time = 2000, n_channels = 19, sr = 500)
pe <- eegMicrostates(pe, n_states = 4)
seq_labels <- eegMicrostateSequence(pe)
table(seq_labels)
```

eegMicrostateStats	<i>Compute Microstate Statistics</i>
--------------------	--------------------------------------

Description

Calculates temporal statistics for each microstate class from a segmented EEG recording: mean duration, occurrence rate, and time coverage. Also computes the transition probability matrix between states.

Usage

```
eegMicrostateStats(x)
```

Arguments

x A `PhysioExperiment` object with microstate labels in `metadata(x)$microstates` (from [eegMicrostates](#)).

Value

A `data.frame` with columns:

state Integer microstate class (1 to `n_states`).

duration_ms Mean duration of consecutive runs in milliseconds.

occurrence_per_sec Number of state occurrences (runs) per second.

coverage_pct Percentage of total time spent in this state.

The transition probability matrix (`n_states` x `n_states`) is stored as an attribute `"transition_matrix"`.

References

Michel, C. M., & Koenig, T. (2018). EEG microstates as a tool for studying the temporal dynamics of whole-brain neuronal networks. *NeuroImage*, 180, 577-593.

See Also

[eegMicrostates\(\)](#), [eegMicrostateBackfit\(\)](#), [eegMicrostateSequence\(\)](#)

Examples

```
pe <- make_eeg(n_time = 5000, n_channels = 19, sr = 500)
pe <- eegMicrostates(pe, n_states = 4, method = "kmeans")
stats <- eegMicrostateStats(pe)
print(stats)
attr(stats, "transition_matrix")
```

eegMontage	<i>Assign electrode montage positions</i>
------------	---

Description

Maps channel labels to standard electrode positions from a known montage system (10-20, 10-10, BioSemi64) or from a custom positions data.frame. Sets pos_x, pos_y, pos_z columns in colData for use by interpolation, topographic mapping, and source localization functions.

Usage

```
eegMontage(  
  x,  
  system = c("10-20", "10-10", "biosemi64", "custom"),  
  positions = NULL  
)
```

Arguments

- x A `PhysioExperiment` object.
- system Montage system: "10-20" (19 channels), "10-10" (~64 channels), "biosemi64" (64 BioSemi channels), or "custom" (user-provided positions).
- positions For system = "custom", a data.frame with columns label, pos_x, pos_y, pos_z.

Value

A `PhysioExperiment` object with updated colData containing position columns.

Examples

```
pe <- make_eeg(n_time = 5000, n_channels = 19, sr = 500)  
pe <- eegMontage(pe, system = "10-20")  
head(SummarizedExperiment::colData(pe))
```

eegMorletWavelet	<i>Morlet Wavelet Transform for EEG</i>
------------------	---

Description

Computes the continuous Morlet wavelet transform for multi-channel EEG data. For each specified frequency, a complex Morlet wavelet is constructed and convolved with each channel using FFT-based convolution for efficiency. Returns time-resolved power (and optionally phase) across frequencies.

Usage

```
eegMorletWavelet(
  x,
  frequencies = NULL,
  n_cycles = 7,
  assay_name = NULL,
  output_assay = "wavelet_power"
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with EEG data (2D: time x channels).
<code>frequencies</code>	Numeric vector of frequencies in Hz to analyze. If <code>NULL</code> , defaults to <code>seq(1, 50, by = 1)</code> .
<code>n_cycles</code>	Number of cycles in the Morlet wavelet, controlling the trade-off between time and frequency resolution (default: 7).
<code>assay_name</code>	Name of the input assay. If <code>NULL</code> , the default assay is used.
<code>output_assay</code>	Name the result is stored under. Despite the argument's name this is a key in metadata , NOT an assay: after the call <code>assay(x, "the value given")</code> does not exist while <code>metadata(x)\$the value given</code> holds the result (default: "the value given"). (default: "wavelet_power").

Value

Modified `PhysioExperiment` with:

- 3D power array (time x frequencies x channels) in `output_assay`
- Frequency vector and phase array in `metadata(x)$wavelet`, a list containing frequencies (numeric vector), `n_cycles` (integer), and phase (3D array of instantaneous phase values)

References

Tallon-Baudry, C., et al. (1997). Oscillatory gamma-band activity during conscious perception. *Trends in Cognitive Sciences*, 3(4), 151-162.

See Also

[eegSTFT\(\)](#), [eegMultitaper\(\)](#), [eegPlotSpectrogram\(\)](#), [eegERSP\(\)](#)

Examples

```
pe <- make_eeg(n_time = 1000, n_channels = 4, sr = 250)
pe_wt <- eegMorletWavelet(pe, frequencies = seq(5, 40, by = 5))
# results are stored in metadata(x)$wavelet
str(S4Vectors::metadata(pe_wt)$wavelet, max.level = 1)
```

eegMotorImagery	<i>Motor Imagery ERD/ERS Computation</i>
-----------------	--

Description

Computes event-related desynchronization (ERD) and event-related synchronization (ERS) for motor imagery BCI analysis. ERD/ERS is expressed as a percentage change from baseline power in specified frequency bands.

Usage

```
eegMotorImagery(
    x,
    bands = NULL,
    baseline_fraction = 0.25,
    assay_name = NULL,
    output_assay = "erd_ers"
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with epoched (3D) EEG data (time x channels x trials).
<code>bands</code>	Named list of frequency bands, each a numeric vector <code>c(low, high)</code> . Default: <code>list(mu = c(8, 13), beta = c(13, 30))</code> .
<code>baseline_fraction</code>	Fraction of each trial to use as baseline (default: 0.25, i.e., the first 25 percent of the trial).
<code>assay_name</code>	Input assay name (default: first assay).
<code>output_assay</code>	Output assay name (default: "erd_ers").

Value

Modified `PhysioExperiment` with ERD/ERS percentage values in `output_assay`. The output is a matrix of `n_trials` rows x `n_channels * n_bands` columns. Sets `metadata(x)$erd_ers_bands` with the band definitions used.

References

Blankertz, B., et al. (2008). Optimizing spatial filters for robust EEG single-trial analysis. *IEEE Signal Processing Magazine*, 25(1), 41-56.

See Also

[eegCSP\(\)](#), [eegBCIfeatures\(\)](#), [eegBCIclassify\(\)](#)

Examples

```
pe <- make_eeg_bci(n_trials = 20, n_channels = 8, sr = 256)
result <- eegMotorImagery(pe)
# ERD/ERS features are stored in metadata(x)$erd_ers
erd_data <- S4Vectors::metadata(result)$erd_ers
dim(erd_data) # trials x (channels * bands)
```

eegMultitaper

Multitaper Power Spectral Density for EEG

Description

Computes the multitaper power spectral density (PSD) estimate for multi-channel EEG data using Discrete Prolate Spheroidal Sequences (DPSS, Slepian tapers). The multitaper method provides an optimal bias-variance trade-off for spectral estimation compared to single-window methods.

Usage

```
eegMultitaper(
  x,
  bandwidth = 4,
  n_tapers = NULL,
  assay_name = NULL,
  output_assay = "multitaper_psd"
)
```

Arguments

x	A <code>PhysioExperiment</code> object with EEG data (2D: time x channels).
bandwidth	Time-half-bandwidth product (NW) controlling the spectral concentration of the tapers (default: 4). Higher values give smoother but lower-resolution estimates.
n_tapers	Number of DPSS tapers to use. If <code>NULL</code> , defaults to <code>floor(2 * bandwidth) - 1</code> .
assay_name	Name of the input assay. If <code>NULL</code> , the default assay is used.
output_assay	Name the result is stored under. Despite the argument's name this is a key in <code>metadata</code> , NOT an assay: after the call <code>assay(x, "the value given")</code> does not exist while <code>metadata(x)\$the value given</code> holds the result (default: "the value given"). (default: "multitaper_psd").

Value

Modified `PhysioExperiment` with:

- PSD matrix (frequencies x channels) in `output_assay`
- Frequency vector and taper parameters in `metadata(x)$multitaper`, a list containing frequencies (numeric vector), bandwidth (numeric NW parameter), and `n_tapers` (integer)

References

- Tallon-Baudry, C., et al. (1997). Oscillatory gamma-band activity during conscious perception. *Trends in Cognitive Sciences*, 3(4), 151-162.
- Thomson, D. J. (1982). Spectrum estimation and harmonic analysis. *Proceedings of the IEEE*, 70(9), 1055-1096.

See Also

`eegMorletWavelet()`, `eegSTFT()`, `eegPlotSpectrogram()`

Examples

```
pe <- make_eeg(n_time = 2000, n_channels = 4, sr = 250)
pe_mt <- eegMultitaper(pe, bandwidth = 4)
# results are stored in metadata(x)$multitaper
str(S4Vectors::metadata(pe_mt)$multitaper, max.level = 1)
```

eegPAC

Phase-amplitude coupling (cross-frequency coupling) for EEG

Description

Measures how the phase of a slow oscillation modulates the amplitude of a fast oscillation — within a single channel (`amp_channel = NULL`) or across two channels. Delegates the estimator to `PhysioCrossModal::phaseAmplitudeCoupling()`.

Usage

```
eegPAC(
  pe,
  phase_channel = 1L,
  amp_channel = NULL,
  phase_band = c(4, 8),
  amp_band = c(30, 80),
  method = c("tort", "canolty", "ozkurt", "plv"),
  n_bins = 18L,
  assay_name = NULL
)
```

Arguments

- | | |
|----------------------------|--|
| <code>pe</code> | A <code>PhysioExperiment</code> (continuous data; epoched data is concatenated). |
| <code>phase_channel</code> | Channel (index or label) supplying the modulating phase. |
| <code>amp_channel</code> | Channel supplying the modulated amplitude; <code>NULL</code> (default) uses <code>phase_channel</code> (within-channel PAC). |
| <code>phase_band</code> | Phase frequency band in Hz (default theta <code>c(4, 8)</code>). |

amp_band	Amplitude frequency band in Hz (default gamma c(30, 80)).
method	"tort" (modulation index), "canolty" (mean vector length), "ozkurt", or "plv".
n_bins	Number of phase bins for the Tort modulation index (default 18).
assay_name	Assay to use (default: the object's default assay).

Value

The PAC result from `PhysioCrossModal::phaseAmplitudeCoupling()` (a list with `pac` and, for "tort", the phase-amplitude distribution).

References

Tort et al. 2010; Canolty et al. 2006.

See Also

[eegComodulogram\(\)](#), [eegConnectivityMatrix\(\)](#)

Examples

```
if (requireNamespace("PhysioCrossModal", quietly = TRUE)) {
  pe <- make_eeg(n_time = 5000, n_channels = 4, sr = 250)
  eegPAC(pe, phase_channel = 1, phase_band = c(4, 8), amp_band = c(30, 80))
}
```

eegPDC

*Partial Directed Coherence (PDC)***Description**

Estimates frequency-resolved directed connectivity with Partial Directed Coherence (Baccala & Sameshima 2001), computed from the frequency-domain coefficient matrix $\bar{A}(f)$ of an MVAR model ([PhysioExperiment::mvarFit\(\)](#)). Unlike DTF, PDC reflects only *direct* channel-to-channel influences, so a purely indirect pathway gives PDC near zero. The (default) generalized PDC weights each row by the inverse residual standard deviation to make the measure scale-invariant (Baccala 2007). PDC satisfies $\sum_i \text{PDC}_{ij}(f)^2 = 1$ for each source j (outflow normalization).

Usage

```
eegPDC(
  x,
  order = NULL,
  freqs = NULL,
  generalized = TRUE,
  band = NULL,
  method = "ols",
  assay_name = NULL
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with 2D EEG data (time x channels).
<code>order</code>	MVAR model order, or <code>NULL</code> to select automatically.
<code>freqs</code>	Numeric vector of frequencies in Hz (default: 128 points from 0 to the Nyquist frequency).
<code>generalized</code>	Use generalized PDC (default: <code>TRUE</code>).
<code>band</code>	Optional numeric length-2 band in Hz over which to average the stored connectivity matrix (default: all frequencies).
<code>method</code>	MVAR estimator passed to <code>PhysioExperiment::mvarFit()</code> (default: <code>"ols"</code>).
<code>assay_name</code>	Input assay name (default: the default assay).

Value

The `PhysioExperiment` with `metadata(x)$connectivity` set as in `eegDTF()` (a band-averaged directed matrix, the frequency-resolved array, frequencies, and settings).

References

Baccala, L. A., & Sameshima, K. (2001). Partial directed coherence: a new concept in neural structure determination. *Biological Cybernetics*, 84(6), 463-474.

See Also

`eegDTF()`, `eegConditionalGC()`, `eegConnectivityMatrix()`

Examples

```
pe <- make_eeg(n_time = 4000, n_channels = 5, sr = 250)
pe <- eegPDC(pe, order = 5)
S4Vectors::metadata(pe)$connectivity$matrix
```

eegPlotButterflyGFP *Plot EEG Butterfly Traces with Global Field Power*

Description

Displays the selected continuous EEG channels as a butterfly plot above their Global Field Power (GFP). GFP is the spatial dispersion across the displayed channels at each time point, so its magnitude depends on the selected channels and EEG reference. It is not source-localized power.

Usage

```
eegPlotButterflyGFP(
  x,
  channels = NULL,
  time_range = NULL,
  show_events = FALSE,
  assay_name = NULL,
  gfp_definition = c("population", "sample"),
  heights = c(3, 1)
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object containing a numeric 2D continuous EEG assay (time x channels).
<code>channels</code>	Unique character vector of exact channel labels. If <code>NULL</code> , all channels are shown.
<code>time_range</code>	Optional finite increasing length-two vector specifying the inclusive display range in seconds. GFP is calculated before display windowing.
<code>show_events</code>	Logical; draw finite event times from <code>metadata(x)\$events</code> on both panels.
<code>assay_name</code>	Input assay name. If <code>NULL</code> , uses the default assay.
<code>gfp_definition</code>	Denominator used for spatial dispersion: "population" divides by <code>C</code> and matches <code>eegMicrostates()</code> ; "sample" divides by <code>C - 1</code> .
<code>heights</code>	Two finite positive values giving the butterfly and GFP panel height ratio.

Value

A patchwork object containing two ggplot panels.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE) && requireNamespace("patchwork", quietly = TRUE)) {
  pe <- make_eeg(n_time = 1500, n_channels = 19, sr = 250)
  eegPlotButterflyGFP(pe, channels = c("Fz", "Cz", "Pz"))
}
```

`eegPlotConnectivity` *Plot Connectivity Matrix or Circle*

Description

Visualizes EEG connectivity as either a heatmap (matrix) or a circular connectivity plot (circle). The connectivity matrix can be provided directly or read from `metadata(x)$connectivity$matrix`.

Usage

```
eegPlotConnectivity(
  x,
  method = c("heatmap", "circle"),
  matrix = NULL,
  threshold = 0,
  labels = NULL,
  palette = "RdBu"
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with EEG data.
<code>method</code>	Display method: "heatmap" for a correlation/connectivity matrix heatmap or "circle" for a circular connectivity diagram.
<code>matrix</code>	Numeric matrix of connectivity values. If <code>NULL</code> , reads from <code>metadata(x)\$connectivity\$matrix</code> .
<code>threshold</code>	Numeric; only show connections above this value (default: 0).
<code>labels</code>	Character vector of channel labels. If <code>NULL</code> , uses row/column names of the matrix or channel labels from <code>colData</code> .
<code>palette</code>	Character name of the diverging color palette (default: "RdBu").

Value

A `ggplot2` object.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  mat <- matrix(runif(16), 4, 4)
  diag(mat) <- 1
  pe <- make_eeg(n_time = 1000, n_channels = 4, sr = 250)
  eegPlotConnectivity(pe, method = "heatmap", matrix = mat)
}
```

<code>eegPlotConnectogram</code>	<i>Circular EEG connectogram</i>
----------------------------------	----------------------------------

Description

Draws a deterministic circular view of a connectivity matrix stored in `metadata(x)$connectivity$matrix`. Symmetric matrices are represented by one edge per unordered channel pair. Asymmetric matrices retain every ordered entry, following the `PhysioEEG` directed-matrix convention of rows as targets and columns as sources.

Usage

```
eegPlotConnectogram(
  x,
  order = c("hemisphere", "lobe", "cluster"),
  modules = NULL,
  threshold = NULL,
  bundle = TRUE
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> with a finite square numeric matrix at <code>metadata(x)\$connectivity\$matrix</code> .
<code>order</code>	Exactly one node ordering: "hemisphere", "lobe", or "cluster". Cluster ordering uses average-linkage clustering of symmetrized absolute connectivity and deterministic merge orientation.
<code>modules</code>	Optional module declaration. Supply a named character or factor vector, or a data frame with unique label and module columns covering every channel. If <code>NULL</code> , an exact <code>colData(x)\$module</code> column is used when present, otherwise lobe.
<code>threshold</code>	A finite non-negative scalar, or <code>NULL</code> for zero.
<code>bundle</code>	One non-missing logical scalar. If <code>TRUE</code> , edges follow deterministic cubic Bezier paths through module hubs; otherwise they are straight.

Details

The threshold is a strict absolute display threshold (`abs(value) > threshold`); it is not a p-value or a corrected significance mask. Hemisphere and lobe are inferred only for recognized 10-20/10-10 labels. Other labels remain "unknown" unless explicit `colData` columns are supplied. Label tie-breaking uses a stable UTF-8 byte key. Edge identifiers length-prefix both endpoint labels so labels containing the visible " -- " or " -> " separators cannot collide.

Value

A `ggplot2` object. Resolved node, edge, path, module-arc, and setting tables are available in `attr(plot, "connectogram_data")`.

See Also

[eegPlotConnectivity\(\)](#), [eegConnectivityMatrix\(\)](#)

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  pe <- make_eeg(n_time = 1000, n_channels = 4, sr = 250)
  mat <- matrix(c(
    1, 0.5, 0, -0.4,
    0.5, 1, 0.3, 0,
    0, 0.3, 1, 0.6,
```

```

    -0.4, 0, 0.6, 1
  ), 4, 4)
  labels <- SummarizedExperiment::colData(pe)$label
  dimnames(mat) <- list(labels, labels)
  S4Vectors::metadata(pe)$connectivity <- list(
    matrix = mat, method = "coherence", band = c(8, 13)
  )
  eegPlotConnectogram(pe, threshold = 0.2)
}

```

eegPlotERP

Plot ERP Waveform with Confidence Interval

Description

Plots event-related potential waveforms averaged across epochs, with optional confidence interval ribbons. Supports condition-based comparisons when `metadata(x)$conditions` is available.

Usage

```

eegPlotERP(
  x,
  channels = NULL,
  conditions = NULL,
  ci = 0.95,
  show_ci = TRUE,
  epoch_start = 0,
  assay_name = NULL
)

```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with epoched (3D) EEG data (time x channels x epochs).
<code>channels</code>	Character vector of channel labels to plot. If <code>NULL</code> , the grand average across all channels is used.
<code>conditions</code>	Character vector of condition labels to include. If <code>NULL</code> and <code>metadata(x)\$conditions</code> exists, all conditions are plotted.
<code>ci</code>	Confidence level for the interval (default: 0.95).
<code>show_ci</code>	Logical; if <code>TRUE</code> , display confidence interval ribbon.
<code>epoch_start</code>	Numeric start time of each epoch in seconds for the x-axis (default: 0).
<code>assay_name</code>	Input assay name. If <code>NULL</code> , uses the default assay.

Value

A `ggplot2` object.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  pe <- make_eeg_erp(n_epochs = 40, sr = 250)
  eegPlotERP(pe, channels = "Cz")
}
```

eegPlotGlassBrain

*Plot Source Localization as Multi-View Glass-Brain Projections***Description**

Creates deterministic sagittal, axial, and coronal maximum-intensity projections from source amplitudes with real three-dimensional coordinates. The bundled outline is schematic and is not an MRI, patient anatomy, or a validation of the inverse solution.

Usage

```
eegPlotGlassBrain(
  x,
  source_data = NULL,
  views = c("sagittal", "axial", "coronal"),
  threshold_pct = 90
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object.
<code>source_data</code>	Optional explicit source data. Supply a data frame with finite <code>x</code> , <code>y</code> , <code>z</code> , and amplitude columns, or a structured list with positions and declared amplitude/matrix reduction fields. If <code>NULL</code> , uses coordinate-bearing metadata from a new <code>eegSourceEstimate()</code> or <code>eegBeamformer()</code> result.
<code>views</code>	Non-empty unique character vector drawn exactly from <code>"sagittal"</code> , <code>"axial"</code> , and <code>"coronal"</code> . Request order is preserved.
<code>threshold_pct</code>	Finite percentile in <code>[0, 100]</code> . The threshold is the type-8 quantile of absolute amplitude over the complete source set; ties are retained.

Details

Projection axes are sagittal `y-z` with depth `x`, axial `x-y` with depth `z`, and coronal `x-z` with depth `y`. Free-orientation source estimates are reduced by root-sum-square over orientations at each time and RMS over time. This non-negative summary is not a signed instantaneous current. A fixed 80 by 80 display grid retains the greatest absolute amplitude in each projected bin, with exact ties resolved by the smallest source ID. The affine display transform comes from the complete unthresholded coordinate cloud and does not modify amplitudes. The bundled outline is schematic: the plot is not MRI registration, patient anatomy, or validation of localization accuracy. Bare amplitude vectors and two-dimensional coordinates are rejected.

Value

A patchwork object for normal projections. When fewer than three rows survive thresholding, returns the first requested view as an eegPlotSource() ggplot fallback without applying a second threshold. Immutable "glassbrain_data" attributes retain the resolved sources, panel data, display transform, and threshold settings.

See Also

`eegForwardModel()`, `eegSourceEstimate()`, `eegBeamformer()`, `eegPlotSource()`

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE) && requireNamespace("patchwork", quietly = TRUE)) {
  pe <- make_eeg(n_time = 500, n_channels = 19, sr = 250)
  fm <- eegForwardModel(pe, method = "spherical", n_sources = 50)
  localized <- eegSourceEstimate(pe, fm, method = "mne")
  eegPlotGlassBrain(localized)
}
```

eegPlotHypnogram

Plot Sleep Hypnogram

Description

Displays a sleep hypnogram showing sleep stage transitions over time. Stages are ordered with Wake at the top and N3 at the bottom, with a characteristic staircase pattern.

Usage

```
eegPlotHypnogram(x, stages = NULL, epoch_sec = 30, colors = NULL)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object (used for metadata access).
<code>stages</code>	A <code>data.frame</code> with columns <code>epoch</code> and <code>stage</code> , or <code>NULL</code> to read from <code>metadata(x)\$sleep_stages</code> .
<code>epoch_sec</code>	Epoch duration in seconds (default: 30).
<code>colors</code>	Named character vector of colors for each stage. If <code>NULL</code> , default colors are used.

Value

A `ggplot2` object.

Examples

```

if (requireNamespace("ggplot2", quietly = TRUE)) {
  pe <- make_eeg_sleep(n_time = 150000, n_channels = 2, sr = 500)
  stages <- eegSleepStage(pe)
  S4Vectors::metadata(pe)$sleep_stages <- stages
  eegPlotHypnogram(pe)
}

```

eegPlotICA

*Plot ICA Components***Description**

Displays ICA component time courses in a stacked layout similar to `eegPlotSignal`. Optionally shows topographic maps of the mixing matrix weights as a side panel annotation.

Usage

```

eegPlotICA(
  x,
  components = NULL,
  time_range = NULL,
  show_topography = TRUE,
  assay_name = NULL
)

```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with ICA results.
<code>components</code>	Integer vector of component indices to display. If <code>NULL</code> , the first 10 (or fewer) are shown.
<code>time_range</code>	Numeric vector of length 2 specifying time range in seconds. If <code>NULL</code> , all data is shown.
<code>show_topography</code>	Logical; if <code>TRUE</code> and ICA mixing matrix is available in <code>metadata(x)\$ica\$mixing</code> , include topographic inset labels.
<code>assay_name</code>	Input assay name. If <code>NULL</code> , uses "ica" if available, otherwise the default assay.

Value

A `ggplot2` object.

Examples

```

if (requireNamespace("ggplot2", quietly = TRUE)) {
  pe <- make_eeg(n_time = 5000, n_channels = 19, sr = 500)
  pe <- eegICA(pe, n_components = 10)
  eegPlotICA(pe, components = 1:5)
}

```

eegPlotSignal

*Plot Multi-Channel EEG Time Series***Description**

Displays multi-channel EEG time series in stacked, butterfly, or grid layout. Supports channel selection, time range restriction, and optional event markers.

Usage

```

eegPlotSignal(
  x,
  channels = NULL,
  time_range = NULL,
  mode = c("stacked", "butterfly", "grid"),
  scale = 1,
  show_events = FALSE,
  assay_name = NULL
)

```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with 2D EEG data (time x channels).
<code>channels</code>	Character vector of channel labels to display. If <code>NULL</code> , all channels are shown.
<code>time_range</code>	Numeric vector of length 2 specifying time range in seconds as <code>c(start, end)</code> . If <code>NULL</code> , all data is shown.
<code>mode</code>	Display mode: "stacked" (vertical offset per channel), "butterfly" (overlay all channels), or "grid" (faceted panels per channel).
<code>scale</code>	Numeric scaling factor for vertical offset in stacked mode (default: 1).
<code>show_events</code>	Logical; if <code>TRUE</code> and <code>metadata(x)\$events</code> exists, vertical lines are drawn at event times.
<code>assay_name</code>	Input assay name. If <code>NULL</code> , uses the default assay.

Value

A `ggplot2` object.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  pe <- make_eeg(n_time = 2500, n_channels = 4, sr = 500)
  eegPlotSignal(pe, mode = "stacked")
}
```

eegPlotSource

*Plot Source Localization Results***Description**

Visualizes source localization results as a 2D scatter plot or flat map projection. Sources are sized and colored by amplitude, with optional thresholding to show only the strongest activations.

Usage

```
eegPlotSource(
  x,
  source_data = NULL,
  method = c("scatter", "flatmap"),
  threshold_pct = 80
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object.
<code>source_data</code>	Named numeric vector of source amplitudes or a <code>data.frame</code> with columns <code>x</code> , <code>y</code> , and <code>amplitude</code> . If <code>NULL</code> , reads from <code>metadata(x)\$source_estimate</code> .
<code>method</code>	Display method: "scatter" for points on a 2D brain outline or "flatmap" for filled regions using interpolation.
<code>threshold_pct</code>	Numeric percentile threshold (0-100). Only sources above this percentile are displayed (default: 80).

Value

A `ggplot2` object.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE) && requireNamespace("patchwork", quietly = TRUE)) {
  pe <- make_eeg(n_time = 1000, n_channels = 19, sr = 250)
  src <- data.frame(x = runif(50, -1, 1), y = runif(50, -1, 1),
    amplitude = rnorm(50)^2)
  eegPlotSource(pe, source_data = src, method = "scatter")
}
```

eegPlotSpectrogram *Plot Spectrogram (Time-Frequency Heatmap)*

Description

Displays a time-frequency representation of EEG data as a heatmap. It uses an unambiguous stored STFT, Morlet, or ERSP product when available. For a continuous 2D assay, it computes the established sliding-window FFT.

Usage

```
eegPlotSpectrogram(
  x,
  channel = 1,
  freq_range = NULL,
  time_range = NULL,
  log_power = TRUE,
  palette = "viridis",
  assay_name = NULL,
  mask = NULL,
  mask_alpha = 0.4,
  contour = TRUE
)
```

Arguments

x	A PhysioExperiment object with EEG data.
channel	Integer or character specifying which channel to display (default: 1).
freq_range	Numeric vector of length 2 for frequency axis limits. If NULL, the full range is shown.
time_range	Numeric vector of length 2 for time axis limits. If NULL, the full range is shown.
log_power	Logical; if TRUE, plot $10 \cdot \log_{10}(\text{power})$ (default: TRUE).
palette	Character name of the color palette (default: "viridis").
assay_name	Input assay or time-frequency metadata product name. If NULL, uses one unambiguous STFT, Morlet, or ERSP product when present, otherwise computes the legacy sliding FFT from the default assay.
mask	Optional display mask. Supply a logical time-by-frequency matrix, a p-value matrix with an explicit alpha attribute, or a private time-frequency inference result. Matrix dimensions and dimnames must exactly match the complete, unfiltered time-frequency axes.
mask_alpha	Opacity for bins outside mask, as one finite value in $[0, 1]$ (default: 0.4).
contour	Logical; draw non-interpolated boundaries between included and excluded mask bins when both are present (default: TRUE).

Details

A mask controls opacity and optional boundaries only. It never replaces, zeros, or otherwise changes the plotted power values. Matrix masks describe the complete time-by-frequency grid before `freq_range` or `time_range` is applied.

Statistical masks must be computed from repeated, independent and exchangeable observations, not inferred from the single spectrogram being plotted. The package-private cluster helper subtracts the declared baseline within each replicate and frequency, uses sign-separated four-neighbour components and a maximum cluster-mass sign-flip null, and applies the conservative plus-one correction. Its corrected p-values are cluster-level evidence; they do not make each enclosed bin individually significant or establish a neurophysiological mechanism.

Value

A ggplot2 object.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  pe <- make_eeg(n_time = 5000, n_channels = 4, sr = 500)
  eegPlotSpectrogram(pe, channel = 1)
}
```

eegPlotTopomap

Plot EEG Topographic Map

Description

Creates a 2D scalp topographic map using inverse distance weighted interpolation. Electrode positions are read from `colData(x)` (columns `pos_x`, `pos_y`) or default 10-20 positions are used as fallback. The plot includes a head outline, nose, and ears.

Usage

```
eegPlotTopomap(
  x,
  time = NULL,
  values = NULL,
  assay_name = NULL,
  resolution = 100,
  palette = "RdBu",
  contours = TRUE,
  electrodes = TRUE
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with EEG data.
<code>time</code>	Numeric time point in seconds at which to extract values from the assay. If <code>NULL</code> and <code>values</code> is also <code>NULL</code> , the mean across all time points is used.
<code>values</code>	Named numeric vector of channel values to plot directly. If provided, overrides data extraction from the assay.
<code>assay_name</code>	Input assay name. If <code>NULL</code> , uses the default assay.
<code>resolution</code>	Integer grid resolution for interpolation (default: 100).
<code>palette</code>	Character name of the diverging color palette (default: "RdBu").
<code>contours</code>	Logical; if <code>TRUE</code> , add contour lines.
<code>electrodes</code>	Logical; if <code>TRUE</code> , show electrode positions as points.

Value

A `ggplot2` object.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  pe <- make_eeg(n_time = 500, n_channels = 19, sr = 250)
  eegPlotTopomap(pe, time = 0.5)
}
```

eegPlotTopomapSeries *Plot Topographic Map Series*

Description

Creates multiple topographic maps at specified time points arranged in a grid layout with a shared or independent color scale.

Usage

```
eegPlotTopomapSeries(
  x,
  times,
  assay_name = NULL,
  ncol = NULL,
  palette = "RdBu",
  shared_limits = TRUE
)
```

Arguments

x	A PhysioExperiment object with EEG data.
times	Numeric vector of time points in seconds.
assay_name	Input assay name. If NULL, uses the default assay.
ncol	Integer number of columns in the grid layout. If NULL, auto-calculated.
palette	Character name of the diverging color palette (default: "RdBu").
shared_limits	Logical; if TRUE, use the same color scale across all panels.

Value

A ggplot2 object with faceted topomaps.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  pe <- make_eeg(n_time = 2500, n_channels = 19, sr = 500)
  eegPlotTopomapSeries(pe, times = c(0.1, 0.2, 0.3))
}
```

eegPLV

*EEG Phase Locking Value (PLV)***Description**

Computes the Phase Locking Value between all EEG channel pairs within a specified frequency band. Each channel is bandpass filtered, instantaneous phase is extracted via the Hilbert transform, and PLV is computed as the mean resultant length of the phase difference distribution.

Usage

```
eegPLV(x, band = c(8, 13), assay_name = NULL)
```

Arguments

x	A PhysioExperiment object with EEG data (2D: time x channels).
band	Numeric vector of length 2 specifying the frequency band in Hz for bandpass filtering before phase extraction (default: c(8, 13) for alpha band).
assay_name	Name of the input assay. If NULL, the default assay is used.

Details

$$PLV = |1/N \sum_{t=1}^N \exp(i(\phi_x(t) - \phi_y(t)))|$$

Value

A data.frame with columns:

channel1 Character or integer identifier of the first channel.

channel2 Character or integer identifier of the second channel.

plv Numeric PLV value in $[0, 1]$.

References

Lachaux, J. P., et al. (1999). Measuring phase synchrony in brain signals. *Human Brain Mapping*, 8(4), 194-208.

See Also

[eegCoherence\(\)](#), [eegWPLI\(\)](#), [eegConnectivityMatrix\(\)](#), [eegPlotConnectivity\(\)](#)

Examples

```
pe <- make_eeg(n_time = 5000, n_channels = 4, sr = 500)
plv_df <- eegPLV(pe, band = c(8, 13))
head(plv_df)
```

eegPreprocess

Full EEG preprocessing pipeline

Description

Convenience wrapper that runs a complete EEG preprocessing pipeline in sequence: filtering, re-referencing, bad channel detection and interpolation, optional ICA, epoching, and artifact rejection. Each step can be toggled on or off.

Usage

```
eegPreprocess(
  x,
  filter = TRUE,
  lowcut = 0.1,
  highcut = 40,
  notch = NULL,
  rereference = TRUE,
  ref_type = "average",
  bad_channels = TRUE,
  interpolate = TRUE,
  ica = FALSE,
  epoch = FALSE,
  events = NULL,
  epoch_limits = c(-0.2, 0.8),
```

```

baseline = c(-0.2, 0),
artifact_reject = FALSE,
threshold_uv = 100,
assay_name = NULL,
verbose = TRUE
)

```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with continuous (2D) EEG data.
<code>filter</code>	Logical; apply frequency filtering (default: <code>TRUE</code>).
<code>lowcut</code>	Low cutoff frequency in Hz for filtering (default: 0.1).
<code>highcut</code>	High cutoff frequency in Hz for filtering (default: 40).
<code>notch</code>	Notch filter center frequency in Hz (<code>NULL</code> for none).
<code>rereference</code>	Logical; apply re-referencing (default: <code>TRUE</code>).
<code>ref_type</code>	Re-referencing type (default: "average").
<code>bad_channels</code>	Logical; detect and report bad channels (default: <code>TRUE</code>).
<code>interpolate</code>	Logical; interpolate detected bad channels (default: <code>TRUE</code>). Requires electrode positions in <code>colData</code> (apply <code>eegMontage</code> first or this will be skipped with a warning).
<code>ica</code>	Logical; apply ICA-based artifact removal (default: <code>FALSE</code>). Requires the <code>eegICA</code> function to be available.
<code>epoch</code>	Logical; epoch the data around events (default: <code>FALSE</code>).
<code>events</code>	Event data for epoching (<code>data.frame</code> or integer vector). Required if <code>epoch = TRUE</code> .
<code>epoch_limits</code>	Epoch window in seconds (default: <code>c(-0.2, 0.8)</code>).
<code>baseline</code>	Baseline window in seconds (default: <code>c(-0.2, 0)</code>).
<code>artifact_reject</code>	Logical; reject artifact epochs (default: <code>FALSE</code>). Only applies if <code>epoch = TRUE</code> .
<code>threshold_uv</code>	Amplitude threshold for artifact rejection (default: 100).
<code>assay_name</code>	Starting assay name. If <code>NULL</code> , uses <code>defaultAssay(x)</code> .
<code>verbose</code>	Logical; print progress messages (default: <code>TRUE</code>).

Value

A `PhysioExperiment` object with processed data. The final assay name depends on which steps are enabled. Processing log is stored in `metadata(x)$preprocess_log`.

Examples

```

pe <- make_eeg(n_time = 10000, n_channels = 19, sr = 500)
pe <- eegMontage(pe, system = "10-20")
# filter + average-reference; bad-channel detection/interpolation disabled
# so the example does not depend on the montage's neighbour geometry

```

```
pe_proc <- eegPreprocess(pe, lowcut = 1, highcut = 40, notch = 50,
                        bad_channels = FALSE, interpolate = FALSE,
                        verbose = FALSE)

pe_proc
```

eegQEEG	<i>Quantitative EEG (QEEG) Analysis</i>
---------	---

Description

Computes absolute and relative spectral band powers for each channel using Welch’s method (windowed FFT averaging). Results are stored as a new assay (channels x bands matrix of absolute power) and as relative power in metadata(x)\$qeeg.

Usage

```
eegQEEG(
  x,
  bands = NULL,
  window_sec = 2,
  overlap = 0.5,
  assay_name = NULL,
  output_assay = "qeeg"
)
```

Arguments

x	A PhysioExperiment object with EEG data.
bands	Named list of frequency bands. Each element is a numeric vector of length 2 specifying the lower and upper frequency in Hz. Defaults to standard EEG bands: delta (1-4), theta (4-8), alpha (8-13), beta (13-30), gamma (30-50).
window_sec	Window length in seconds for Welch’s method (default: 2).
overlap	Overlap fraction between windows, 0 to 1 (default: 0.5).
assay_name	Name of the input assay. If NULL, the default assay is used.
output_assay	Name of the assay to store absolute power results (default: "qeeg").

Value

Modified PhysioExperiment with:

- Absolute power matrix (n_channels x n_bands) in output_assay
- Band definitions and relative power in metadata(x)\$qeeg, a list containing bands, absolute_power, relative_power, band_names, window_sec, and overlap.

References

Nuwer, M. R., et al. (1999). IFCN standards for digital recording of clinical EEG. *Electroencephalography and Clinical Neurophysiology*, 106(3), 259-261.

Thatcher, R. W. (2010). Validity and reliability of quantitative electroencephalography. *Journal of Neurotherapy*, 14(2), 122-152.

See Also

[eegSpikeDetect\(\)](#), [eegAsymmetry\(\)](#), [eegSlowing\(\)](#), [eegPlotSpectrogram\(\)](#)

Examples

```
pe <- make_eeg(n_time = 5000, n_channels = 19, sr = 500)
pe_qeeg <- eegQEEG(pe)
qeeg_info <- S4Vectors::metadata(pe_qeeg)$qeeg
print(qeeg_info$relative_power)
```

eegQuickStart

PhysioEEG Quick Start Guide

Description

Prints a guided walkthrough with runnable code examples for selected EEG analysis workflows. Covers ERP analysis, sleep staging, BCI classification, and connectivity analysis.

Usage

```
eegQuickStart(workflow = "all")
```

Arguments

workflow	Character string: one of "erp", "sleep", "bci", "connectivity", or "all" (default: "all").
----------	--

Value

Invisibly returns NULL. Prints guide to console.

See Also

[make_eeg\(\)](#), [make_eeg_erp\(\)](#), [make_eeg_sleep\(\)](#), [make_eeg_bci\(\)](#)

Examples

```
eegQuickStart("erp")
```

eegRereference

*Re-reference EEG data***Description**

Applies a re-referencing scheme to EEG data. Re-referencing transforms the data by subtracting a reference signal from all channels, which can improve spatial resolution and comparability across studies.

Usage

```
eegRereference(
  x,
  ref_type = c("average", "robust", "median", "mastoids", "cz", "rest", "channel"),
  ref_channels = NULL,
  exclude = NULL,
  assay_name = NULL,
  output_assay = "rereferenced",
  robust_noise_sd = 4,
  robust_max_iter = 5
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object.
<code>ref_type</code>	Re-referencing scheme: "average" (common average), "robust" (PREP robust average reference: iteratively detect bad channels and exclude them from the average, Bigdely-Shamlo et al., 2015), "median" (channel-wise median reference, robust to outlier channels), "mastoids" (linked mastoids), "cz" (Cz reference), "rest" (Reference Electrode Standardization Technique), or "channel" (user-specified channels).
<code>ref_channels</code>	Character vector of channel labels to use as reference (required for <code>ref_type = "channel"</code>).
<code>exclude</code>	Character vector of channel labels to exclude from the average reference calculation (only used for <code>ref_type = "average"</code>).
<code>assay_name</code>	Name of the assay to re-reference. If <code>NULL</code> , uses <code>defaultAssay(x)</code> .
<code>output_assay</code>	Name of the output assay (default: "rereferenced").
<code>robust_noise_sd</code>	Noise threshold (SDs above the median channel variance) for bad-channel detection in <code>ref_type = "robust"</code> (default: 4).
<code>robust_max_iter</code>	Maximum PREP iterations for <code>ref_type = "robust"</code> (default: 5).

Value

A `PhysioExperiment` object with re-referenced data in the specified output assay. Stores reference info (including the channels excluded by the robust reference) in `metadata(x)$reference` and logs the step in the object's provenance.

References

Bigdely-Shamlo, N., et al. (2015). "The PREP pipeline: standardized preprocessing for large-scale EEG analysis." *Frontiers in Neuroinformatics*, 9, 16. doi:[10.3389/fninf.2015.00016](https://doi.org/10.3389/fninf.2015.00016)

Examples

```
pe <- make_eeg(n_time = 5000, n_channels = 19, sr = 500)
pe_avg <- eegRereference(pe, ref_type = "average")
pe_robust <- eegRereference(pe, ref_type = "robust")
pe_median <- eegRereference(pe, ref_type = "median")
```

eegSleepMetrics

Compute Sleep Metrics

Description

Calculates summary sleep architecture metrics from staged sleep data. Requires that `eegSleepStage()` has been run and results are stored in `metadata(x)$sleep_stages`.

Usage

```
eegSleepMetrics(x)
```

Arguments

`x` A `PhysioExperiment` object with sleep staging results in `metadata(x)$sleep_stages`.

Value

A `data.frame` with columns:

metric Character name of the sleep metric.

value Numeric value of the metric.

unit Character unit of measurement.

Metrics include:

- `total_sleep_time`: Time in N1+N2+N3+REM (minutes)
- `sleep_efficiency`: TST / total recording time * 100 (percent)
- `waso`: Wake after sleep onset (minutes)
- `sleep_latency`: Time to first non-Wake epoch (minutes)

- rem_latency: Time from first sleep to first REM (minutes)
- pct_N1: Percentage of TST in N1
- pct_N2: Percentage of TST in N2
- pct_N3: Percentage of TST in N3
- pct_REM: Percentage of TST in REM
- pct_W: Percentage of total time in Wake

References

Berry, R. B., et al. (2017). AASM Scoring Manual Updates for 2017. *Journal of Clinical Sleep Medicine*, 13(5), 665-666.

See Also

`eegSleepStage()`, `eegSpindleDetect()`, `eegKcomplexDetect()`, `eegSlowWaveDetect()`, `eegPlotHypnogram()`

Examples

```
pe <- make_eeg_sleep(n_time = 150000, n_channels = 2, sr = 500)
stages <- eegSleepStage(pe, epoch_sec = 30)
S4Vectors::metadata(pe)$sleep_stages <- stages
metrics <- eegSleepMetrics(pe)
print(metrics)
```

eegSleepStage

Automatic Sleep Staging

Description

Classifies EEG epochs into sleep stages (Wake, N1, N2, N3, REM) using spectral power analysis based on simplified AASM criteria. Each epoch is scored by computing spectral band powers (delta, theta, alpha, sigma, beta) and applying rule-based classification.

Usage

```
eegSleepStage(
  x,
  method = c("spectral", "rule_based"),
  epoch_sec = 30,
  assay_name = NULL
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with EEG data.
<code>method</code>	Staging method: "spectral" (FFT power-based) or "rule_based" (synonym, same algorithm).
<code>epoch_sec</code>	Epoch duration in seconds (default: 30, AASM standard).
<code>assay_name</code>	Input assay name. If NULL, uses the default assay.

Value

A `data.frame` with columns:

epoch Integer epoch number.

stage Character sleep stage: "W", "N1", "N2", "N3", or "REM".

start_sample Integer start sample of the epoch.

end_sample Integer end sample of the epoch.

delta_power Numeric delta band power (0.5-4 Hz).

theta_power Numeric theta band power (4-8 Hz).

alpha_power Numeric alpha band power (8-13 Hz).

sigma_power Numeric sigma band power (12-16 Hz).

beta_power Numeric beta band power (16-30 Hz).

The result is also stored in `metadata(x)$sleep_stages`.

References

Berry, R. B., et al. (2017). AASM Scoring Manual Updates for 2017. *Journal of Clinical Sleep Medicine*, 13(5), 665-666.

See Also

[eegSpindleDetect\(\)](#), [eegKcomplexDetect\(\)](#), [eegSlowWaveDetect\(\)](#), [eegSleepMetrics\(\)](#), [eegPlotHypnogram\(\)](#)

Examples

```
pe <- make_eeg_sleep(n_time = 90000, n_channels = 2, sr = 500)
stages <- eegSleepStage(pe, epoch_sec = 30)
head(stages)
```

eegSlowing

*EEG Slowing Detection***Description**

Detects pathological EEG slowing using spectral analysis. Supports three methods: theta/delta ratio (TDR), delta-theta/alpha-beta ratio (DTAR), and peak frequency analysis. Each channel is classified into normal, mild, moderate, or severe slowing categories.

Usage

```
eegSlowing(
  x,
  method = c("tdr", "dtar", "peak_frequency"),
  bands = NULL,
  assay_name = NULL
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with EEG data.
<code>method</code>	Analysis method: "tdr" (theta/delta ratio), "dtar" (delta+theta over alpha+beta ratio), or "peak_frequency" (dominant frequency per channel).
<code>bands</code>	Named list of frequency bands for TDR and DTAR methods. Defaults to standard EEG bands.
<code>assay_name</code>	Name of the input assay. If NULL, the default assay is used.

Value

A `data.frame` with columns:

channel Integer channel index.

metric Character name of the metric used.

value Numeric value of the metric.

classification Character: "normal", "mild_slowing", "moderate_slowing", or "severe_slowing".

References

Nuwer, M. R., et al. (1999). IFCN standards for digital recording of clinical EEG. *Electroencephalography and Clinical Neurophysiology*, 106(3), 259-261.

See Also

[eegSpikeDetect\(\)](#), [eegQEEG\(\)](#), [eegAsymmetry\(\)](#), [eegSuppression\(\)](#)

Examples

```
pe <- make_eeg(n_time = 5000, n_channels = 19, sr = 500)
result <- eegSlowing(pe, method = "dtar")
print(result)
```

eegSlowWaveDetect	<i>Detect Slow Waves</i>
-------------------	--------------------------

Description

Identifies slow oscillations in EEG data by bandpass filtering in the slow wave frequency range, finding zero crossings, and measuring negative half-wave amplitudes and slopes.

Usage

```
eegSlowWaveDetect(
  x,
  min_amplitude = 75,
  freq_range = c(0.5, 2),
  min_duration_ms = 250,
  max_duration_ms = 1000,
  assay_name = NULL
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with EEG data.
<code>min_amplitude</code>	Minimum absolute negative peak amplitude in microvolts (default: 75).
<code>freq_range</code>	Numeric vector of length 2 specifying the slow wave frequency range in Hz (default: <code>c(0.5, 2)</code>).
<code>min_duration_ms</code>	Minimum half-wave duration in milliseconds (default: 250).
<code>max_duration_ms</code>	Maximum half-wave duration in milliseconds (default: 1000).
<code>assay_name</code>	Input assay name. If <code>NULL</code> , uses the default assay.

Value

A `data.frame` with columns:

channel Integer channel index.
start_sample Integer sample at first zero crossing.
end_sample Integer sample at second zero crossing.
negative_peak Numeric negative peak amplitude.
positive_peak Numeric positive peak amplitude.
duration_ms Numeric half-wave duration in milliseconds.
slope Numeric slope from negative to positive peak (microvolts per millisecond).

References

Berry, R. B., et al. (2017). AASM Scoring Manual Updates for 2017. *Journal of Clinical Sleep Medicine*, 13(5), 665-666.

See Also

[eegSleepStage\(\)](#), [eegSpindleDetect\(\)](#), [eegKcomplexDetect\(\)](#), [eegSleepMetrics\(\)](#)

Examples

```
pe <- make_eeg_sleep(n_time = 150000, n_channels = 2, sr = 500)
slow_waves <- eegSlowWaveDetect(pe)
head(slow_waves)
```

eegSourceEstimate	<i>EEG Source Estimation</i>
-------------------	------------------------------

Description

Estimates brain source activity from scalp EEG data using distributed source imaging methods. Requires a forward model from [eegForwardModel](#).

Usage

```
eegSourceEstimate(
  x,
  forward_model,
  method = c("sloreta", "eloreta", "mne", "dspm"),
  lambda = 0.05,
  assay_name = NULL,
  output_assay = "source"
)
```

Arguments

x	A <code>PhysioExperiment</code> object with EEG data.
forward_model	A forward model list as returned by eegForwardModel .
method	Source estimation method: "sloreta" (standardized low-resolution tomography), "eloreta" (exact low-resolution tomography), "mne" (minimum norm estimate), or "dspm" (dynamic statistical parametric mapping; MNE noise-normalized by each source's projected noise sensitivity, Dale et al. 2000).
lambda	Regularization parameter (default: 0.05). Higher values produce smoother solutions.
assay_name	Name of the input assay. If NULL, the default assay is used.
output_assay	Name for the output assay containing source estimates (default: "source"). The metadata names "source_estimate", "beamformer_info", and "source_plot_default" are reserved.

Value

Modified PhysioExperiment with source estimates stored in output_assay. The assay is a matrix of dimensions n_time x (n_sources * 3). Sets metadata(x)\$source_estimate with a list containing: method, lambda, n_sources, n_source_cols, source_positions, output_assay, orientation_count, and coordinate provenance.

References

Pascual-Marqui, R. D. (2002). Standardized low-resolution brain electromagnetic tomography (sLORETA). Methods and Findings in Experimental and Clinical Pharmacology, 24(Suppl D), 5-12.

See Also

[eegForwardModel\(\)](#), [eegBeamformer\(\)](#), [eegSourcePower\(\)](#), [eegPlotSource\(\)](#), [eegPlotGlassBrain\(\)](#)

Examples

```
pe <- make_eeg(n_time = 500, n_channels = 19, sr = 250)
fm <- eegForwardModel(pe, method = "spherical", n_sources = 50)
pe <- eegSourceEstimate(pe, fm, method = "mne")
```

eegSourcePower	<i>Compute Source-Space Band Power</i>
----------------	--

Description

Computes spectral band power for each source from source-estimated EEG data. Requires prior source estimation via [eegSourceEstimate](#).

Usage

```
eegSourcePower(x, bands = NULL, source_assay = "source")
```

Arguments

- x A PhysioExperiment object with source estimates.
- bands Named list of frequency bands, each a numeric vector of length 2 (lower, upper Hz). If NULL, uses standard EEG bands: delta (1-4), theta (4-8), alpha (8-13), beta (13-30), gamma (30-50).
- source_assay Name of the assay containing source data (default: "source").

Value

A data.frame with columns:
source_id Integer source index.
band Character name of the frequency band.
power Numeric spectral power in the band.

References

Pascual-Marqui, R. D. (2002). Standardized low-resolution brain electromagnetic tomography (sLORETA). *Methods and Findings in Experimental and Clinical Pharmacology*, 24(Suppl D), 5-12.

See Also

[eegSourceEstimate\(\)](#), [eegForwardModel\(\)](#), [eegBeamformer\(\)](#)

Examples

```
pe <- make_eeg(n_time = 1000, n_channels = 19, sr = 250)
fm <- eegForwardModel(pe, method = "spherical", n_sources = 20)
pe <- eegSourceEstimate(pe, fm, method = "mne")
bp <- eegSourcePower(pe)
head(bp)
```

eegSpikeDetect

Detect Epileptic Spikes in EEG Data

Description

Identifies epileptiform spike discharges in multi-channel EEG using either morphology-based derivative analysis or template matching via cross-correlation. The morphology method detects sharp transients by thresholding the first derivative, while the template method cross-correlates a canonical spike waveform with each channel.

Usage

```
eegSpikeDetect(
  x,
  method = c("morphology", "template"),
  threshold_sd = 4,
  min_duration_ms = 20,
  max_duration_ms = 200,
  min_amplitude = 50,
  assay_name = NULL
)
```

Arguments

x	A <code>PhysioExperiment</code> object with EEG data.
method	Detection method: "morphology" (derivative-based) or "template" (cross-correlation with canonical spike shape).
threshold_sd	Number of standard deviations above the mean derivative magnitude for detection (default: 4).

min_duration_ms Minimum spike duration in milliseconds (default: 20).
max_duration_ms Maximum spike duration in milliseconds (default: 200).
min_amplitude Minimum peak amplitude in microvolts for a valid spike detection (default: 50).
assay_name Name of the assay to use. If NULL, the default assay is used.

Value

A data.frame with columns:

channel Integer channel index.
sample Integer sample index of the spike peak.
time_sec Time of the spike in seconds.
amplitude Peak amplitude at the spike location.
duration_ms Estimated spike duration in milliseconds.
confidence Confidence score for the detection.

References

Nuwer, M. R., et al. (1999). IFCN standards for digital recording of clinical EEG. *Electroencephalography and Clinical Neurophysiology*, 106(3), 259-261.

See Also

[eegQEEG\(\)](#), [eegAsymmetry\(\)](#), [eegSlowing\(\)](#)

Examples

```
pe <- make_eeg_spikes(n_time = 30000, n_channels = 19, sr = 500, n_spikes = 15)
spikes <- eegSpikeDetect(pe, method = "morphology")
head(spikes)
```

eegSpindleDetect	<i>Detect Sleep Spindles</i>
------------------	------------------------------

Description

Identifies sleep spindles in EEG data by bandpass filtering in the sigma frequency range, computing an RMS envelope, and detecting continuous segments exceeding a threshold. Spindle frequency is estimated from zero crossings in the bandpassed signal.

Usage

```
eegSpindleDetect(
  x,
  method = c("sigma", "wavelet"),
  freq_range = c(11, 16),
  min_duration_ms = 500,
  max_duration_ms = 2000,
  threshold_sd = 1.5,
  assay_name = NULL
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with EEG data.
<code>method</code>	Detection method: "sigma" (sigma-band filtering) or "wavelet" (synonym, same algorithm).
<code>freq_range</code>	Numeric vector of length 2 specifying the spindle frequency range in Hz (default: <code>c(11, 16)</code>).
<code>min_duration_ms</code>	Minimum spindle duration in milliseconds (default: 500).
<code>max_duration_ms</code>	Maximum spindle duration in milliseconds (default: 2000).
<code>threshold_sd</code>	Number of standard deviations above the mean for detection threshold (default: 1.5).
<code>assay_name</code>	Input assay name. If <code>NULL</code> , uses the default assay.

Value

A `data.frame` with columns:

channel Integer channel index.
start_sample Integer start sample of the spindle.
end_sample Integer end sample of the spindle.
duration_ms Numeric spindle duration in milliseconds.
peak_sample Integer sample index of peak amplitude.
peak_amplitude Numeric peak RMS amplitude.
frequency_hz Numeric estimated spindle frequency in Hz.

References

Berry, R. B., et al. (2017). AASM Scoring Manual Updates for 2017. *Journal of Clinical Sleep Medicine*, 13(5), 665-666.

Molle, M., et al. (2011). Fast and slow spindles during the sleep slow oscillation. *Sleep*, 34(10), 1411-1421.

See Also

`eegSleepStage()`, `eegKcomplexDetect()`, `eegSlowWaveDetect()`, `eegSleepMetrics()`

Examples

```
pe <- make_eeg_sleep(n_time = 150000, n_channels = 2, sr = 500)
spindles <- eegSpindleDetect(pe)
head(spindles)
```

eegSSVEP	<i>SSVEP Frequency Detection</i>
----------	----------------------------------

Description

Detects Steady-State Visual Evoked Potentials using canonical correlation analysis (CCA) or filter-bank CCA (FBCCA). For each candidate stimulus frequency, a CCA is computed between the EEG data and sinusoidal reference signals at the frequency and its harmonics.

Usage

```
eegSSVEP(
  x,
  frequencies,
  n_harmonics = 3,
  method = c("cca", "fbcca"),
  assay_name = NULL
)
```

Arguments

x	A <code>PhysioExperiment</code> object with EEG data. If 3D (time x channels x trials), data is averaged across trials before analysis.
frequencies	Numeric vector of target stimulus frequencies in Hz.
n_harmonics	Number of harmonics to include in reference signals (default: 3).
method	Detection method: "cca" (canonical correlation analysis) or "fbcca" (filter-bank CCA).
assay_name	Input assay name (default: first assay).

Value

A `data.frame` with columns: `frequency` (numeric target frequency in Hz), `correlation` (numeric CCA correlation), `snr` (numeric signal-to-noise ratio), and `predicted_class` (numeric predicted stimulus frequency).

References

- Blankertz, B., et al. (2008). Optimizing spatial filters for robust EEG single-trial analysis. *IEEE Signal Processing Magazine*, 25(1), 41-56.
- Norcia, A. M., et al. (2015). The steady-state visual evoked potential in vision research: a review. *Journal of Vision*, 15(6), 4.

See Also

[eegCSP\(\)](#), [eegBCIfeatures\(\)](#), [eegBCIclassify\(\)](#)

Examples

```
pe <- make_eeg_bci(n_trials = 10, n_channels = 8, sr = 256)
result <- eegSSVEP(pe, frequencies = c(10, 12, 15), method = "cca")
```

eegSTFT

Short-Time Fourier Transform for EEG

Description

Computes the Short-Time Fourier Transform (STFT) spectrogram for multi-channel EEG data. Uses a sliding window with configurable overlap and window function to produce a time-frequency power representation.

Usage

```
eegSTFT(
  x,
  window_sec = 0.5,
  overlap = 0.75,
  window_type = c("hanning", "hamming", "rectangular"),
  assay_name = NULL,
  output_assay = "stft_power"
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with EEG data (2D: time x channels).
<code>window_sec</code>	Window length in seconds (default: 0.5).
<code>overlap</code>	Overlap fraction between adjacent windows, from 0 to 1 exclusive (default: 0.75).
<code>window_type</code>	Window function to apply: "hanning", "hamming", or "rectangular" (default: "hanning").
<code>assay_name</code>	Name of the input assay. If NULL, the default assay is used.
<code>output_assay</code>	Name the result is stored under. Despite the argument's name this is a key in metadata , NOT an assay: after the call <code>assay(x, "the value given")</code> does not exist while <code>metadata(x)\$the value given</code> holds the result (default: "the value given"). (default: "stft_power").

Value

Modified PhysioExperiment with:

- 3D power array (time_bins x frequencies x channels) in output_assay
- Time bin centers, frequency vector, and parameters in metadata(x)\$stft, a list containing time_axis, freq_axis, window_sec, overlap, window_type, window_length, and hop_size

References

Tallon-Baudry, C., et al. (1997). Oscillatory gamma-band activity during conscious perception. Trends in Cognitive Sciences, 3(4), 151-162.

See Also

[eegMorletWavelet\(\)](#), [eegMultitaper\(\)](#), [eegPlotSpectrogram\(\)](#)

Examples

```
pe <- make_eeg(n_time = 2000, n_channels = 4, sr = 250)
pe_stft <- eegSTFT(pe, window_sec = 0.5, overlap = 0.75)
# results are stored in metadata(x)$stft
str(S4Vectors::metadata(pe_stft)$stft, max.level = 1)
```

eegSuppression

Burst-Suppression Detection

Description

Identifies periods of burst and suppression in EEG data by computing the root mean square (RMS) amplitude in sliding windows. Suppression segments are defined as consecutive windows where RMS falls below the specified threshold for at least min_duration_ms.

Usage

```
eegSuppression(
  x,
  threshold = 10,
  min_duration_ms = 500,
  window_ms = 500,
  assay_name = NULL
)
```

Arguments

<code>x</code>	A <code>PhysioExperiment</code> object with EEG data.
<code>threshold</code>	RMS amplitude threshold below which signal is considered suppressed (default: 10).
<code>min_duration_ms</code>	Minimum duration in milliseconds for a segment to be classified as suppression (default: 500).
<code>window_ms</code>	Window length in milliseconds for RMS computation (default: 500).
<code>assay_name</code>	Name of the input assay. If <code>NULL</code> , the default assay is used.

Value

A `data.frame` with columns:

type Character: "burst" or "suppression".

start_sample Integer start sample of the segment.

end_sample Integer end sample of the segment.

duration_ms Numeric duration of the segment in milliseconds.

An attribute "bsr" (Burst-Suppression Ratio) is attached, representing the percentage of total time in suppression.

References

Nuwer, M. R., et al. (1999). IFCN standards for digital recording of clinical EEG. *Electroencephalography and Clinical Neurophysiology*, 106(3), 259-261.

See Also

[eegSpikeDetect\(\)](#), [eegQEEG\(\)](#), [eegSlowing\(\)](#)

Examples

```
pe <- make_eeg(n_time = 10000, n_channels = 4, sr = 500)
result <- eegSuppression(pe, threshold = 10)
print(attr(result, "bsr"))
```

eegSurfaceLaplacian *Surface Laplacian (current source density) of EEG*

Description

Applies the spherical-spline surface Laplacian (current source density, CSD) of Perrin et al. (1989). CSD is **reference-free** and acts as a spatial high-pass filter, deblurring volume conduction so that each channel reflects the radial current under it rather than the whole-head average — sharpening focal topographies and attenuating broadly distributed activity. It needs electrode positions (from colData columns pos_x/pos_y/pos_z, else matched to a standard 10-10/10-20 montage by channel label).

Usage

```
eegSurfaceLaplacian(
  x,
  order = 4L,
  lambda = 1e-05,
  n_terms = 50L,
  assay_name = NULL,
  output_assay = "csd"
)
```

Arguments

x	A <code>PhysioExperiment</code> .
order	Spline flexibility m (default 4; higher = stiffer).
lambda	Smoothing / regularization added to the spline system (default $1e-5$).
n_terms	Number of Legendre terms in the g/h series (default 50).
assay_name	Assay to transform (default: the object's default assay).
output_assay	Name of the assay to store the CSD in (default "csd").

Value

The `PhysioExperiment` with the CSD stored in `output_assay` (same shape as the input assay; units are $\mu V/m^2$ up to a scale/head-radius constant).

References

Perrin, Pernier, Bertrand & Echallier (1989), *Electroenceph Clin Neurophysiol*; Kayser & Tenke (2006), *Clin Neurophysiol*.

See Also

[eegRereference\(\)](#), [eegPlotTopomap\(\)](#), [eegConnectivityMatrix\(\)](#)

Examples

```
pe <- make_eeg(n_time = 500, n_channels = 19, sr = 250)
pe <- eegSurfaceLaplacian(pe)
"csd" %in% SummarizedExperiment::assayNames(pe)
```

eegWPLI

*EEG Weighted Phase Lag Index (wPLI)***Description**

Computes the Weighted Phase Lag Index and its debiased variant between all EEG channel pairs. The wPLI reduces the influence of volume conduction by weighting the phase differences by the magnitude of the imaginary part of the cross-spectrum, computed via Welch's method.

Usage

```
eegWPLI(
  x,
  band = c(8, 13),
  window_sec = 2,
  overlap = 0.5,
  assay_name = NULL,
  debiased = TRUE
)
```

Arguments

x	A PhysioExperiment object with EEG data (2D: time x channels).
band	Numeric vector of length 2 specifying the frequency band in Hz over which to compute wPLI (default: c(8, 13) for alpha band).
window_sec	Window length in seconds for spectral estimation (default: 2).
overlap	Overlap fraction between adjacent windows, from 0 to 1 exclusive (default: 0.5).
assay_name	Name of the input assay. If NULL, the default assay is used.
debiased	Logical; also compute the unbiased debiased wPLI (default TRUE).

Details

$$wPLI = |mean(Im(S_{xy}))|/mean(|Im(S_{xy})|)$$

The debiased wPLI corrects for sample-size bias:

$$wPLI_{debiased}^2 = (N \cdot wPLI^2 - 1)/(N - 1)$$

Value

A data.frame with columns:

channel1 Character or integer identifier of the first channel.

channel2 Character or integer identifier of the second channel.

wpli Numeric wPLI value in $[0, 1]$.

wpli_debiased Numeric debiased wPLI (Vinck et al., 2011, Eq. 6), computed by the shared [wpliEstimate](#) estimator so it matches `PhysioCrossModal::weightedPLI`. It is unbiased and therefore distributes around 0 for independent signals (it is *not* clamped to be non-negative).

References

Lachaux, J. P., et al. (1999). Measuring phase synchrony in brain signals. *Human Brain Mapping*, 8(4), 194-208.

Vinck, M., et al. (2011). An improved index of phase-synchronization for electrophysiological data in the presence of volume-conduction, noise and sample-size bias. *NeuroImage*, 55(4), 1548-1565.

See Also

[eegCoherence\(\)](#), [eegPLV\(\)](#), [eegConnectivityMatrix\(\)](#), [eegPlotConnectivity\(\)](#), [wpliEstimate](#)

Examples

```
pe <- make_eeg(n_time = 5000, n_channels = 4, sr = 500)
wpli_df <- eegWPLI(pe, band = c(8, 13))
head(wpli_df)
```

fuzzyEntropy

Fuzzy Entropy

Description

Computes the fuzzy entropy (FuzzyEn) of a one-dimensional signal (Chen et al., 2007) – the *fuzzy* generalization of the sample entropy. Sample entropy counts template matches with a crisp threshold (a Heaviside step: two vectors either match within tolerance r or do not); fuzzy entropy replaces that hard step with a smooth exponential membership $\mu = \exp(-(d^n)/r)$, so near-matches contribute partially. This makes FuzzyEn markedly more robust and continuous than sample entropy, especially on short or noisy segments where the crisp count is unstable. Each embedding vector has its own baseline (mean) removed before the Chebyshev distance is taken, following Chen et al.

Usage

```
fuzzyEntropy(x, dimension = 2L, r = 0.2, delay = 1L, n = 1)
```

Arguments

x	A numeric vector (the time series).
dimension	Embedding dimension m (default 2).
r	Tolerance as a fraction of the signal's standard deviation; the distance tolerance is $r * \text{sd}(x)$ (default 0.2, matching NeuroKit2's tolerance = "sd").
delay	Time delay (lag) in samples (default 1).
n	Fuzzy power (exponent) of the membership function (default 1).

Details

This implementation reproduces NeuroKit2's `entropy_fuzzy` bit-for-bit (the default tolerance = "sd" there corresponds to $r = 0.2$).

Value

A single numeric value, the fuzzy entropy.

References

Chen, W., Zhuang, J., Yu, W., & Wang, Z. (2009). "Measuring complexity using FuzzyEn, ApEn, and SampEn." *Medical Engineering & Physics*, 31(1), 61–68. doi:10.1016/j.medengphy.2008.04.005

See Also

[dispersionEntropy](#), [svdEntropy](#), [renyiEntropy](#), [tsallisEntropy](#).

Examples

```
set.seed(1)
fuzzyEntropy(rnorm(300), dimension = 2) # fuzzy generalization of sample entropy
```

incrementEntropy	<i>Increment Entropy</i>
------------------	--------------------------

Description

Computes the increment entropy (IncrEn) of a one-dimensional signal (Liu et al., 2016) – a symbolic complexity measure defined on the signal's *increments* (successive differences) rather than its amplitudes. Each increment in an embedding vector is mapped to a word letter combining its *sign* ($-$, 0 , $+$) with a *magnitude class* in $\{0, 1, \dots, q\}$ obtained by quantizing $|\text{increment}|$ relative to the vector's own standard deviation; IncrEn is the Shannon entropy of the resulting word distribution, normalized by $m - 1$. Because it encodes both the direction and the graded size of changes, it is sensitive to dynamics that amplitude-based entropies miss, and is a distinct mechanism from the amplitude-histogram ([renyiEntropy](#), [tsallisEntropy](#)), NCDF-pattern ([dispersionEntropy](#)) and template-matching ([fuzzyEntropy](#)) entropies.

Usage

```
incrementEntropy(x, dimension = 2L, q = 4L)
```

Arguments

x	A numeric vector (the time series).
dimension	Embedding dimension m of the increment words (default 2).
q	Number of magnitude quantization levels (default 4); each increment is binned into one of $q + 1$ magnitude classes by its size relative to the embedding vector's standard deviation.

Details

This implementation reproduces NeuroKit2's `entropy_increment` bit-for-bit.

Value

A single numeric value, the increment entropy.

References

Liu, X., Jiang, A., Xu, N., & Xue, J. (2016). "Increment Entropy as a Measure of Complexity for Time Series." *Entropy*, 18(1), 22. doi:10.3390/e18010022

See Also

[dispersionEntropy](#), [fuzzyEntropy](#), [renyiEntropy](#), [tsallisEntropy](#).

Examples

```
set.seed(1)
incrementEntropy(rnorm(1000), dimension = 2, q = 4)
```

katzFD

Katz Fractal Dimension

Description

Computes Katz's fractal dimension (KFD) of a one-dimensional signal (Katz, 1988) – a waveform-complexity measure derived from the total length of the signal curve relative to its extent. Let L be the sum of the absolute successive differences (the curve length), a their mean, and d the maximum absolute distance from the first sample; then $KFD = \log_{10}(L/a) / \log_{10}(d/a)$. A smoother, more regular signal has a lower KFD; a more convoluted (noise-like) waveform a higher one.

Usage

```
katzFD(x)
```

Arguments

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

Details

This uses the amplitude-only convention adopted by the de-facto reference implementations (Python `antropy`, `NeuroKit2`, `pyeeg`, `mne-features`), in which both the successive distances and the extent are measured in the signal's amplitude units alone; it reproduces `antropy.katz_fd` and `NeuroKit2's fractal_katz` bit-for-bit. (Katz's original definition uses the Euclidean distance in the sample-index/amplitude plane, which mixes the two axes' units, is scale-dependent, and is not comparable across tools; the amplitude-only form is what the field uses and what makes KFD reproducible.)

Value

A single numeric value, Katz's fractal dimension.

References

Katz, M. J. (1988). "Fractals and the analysis of waveforms." *Computers in Biology and Medicine*, 18(3), 145–156. doi:[10.1016/00104825\(88\)900418](https://doi.org/10.1016/00104825(88)900418)

See Also

[petrosianFD](#) for the derivative-sign-change fractal view, [svdEntropy](#), [eegComplexity](#) for the multi-measure wrapper.

Examples

```
katzFD(sin(seq(0, 20 * pi, length.out = 500))) # low (smooth oscillation)
set.seed(1); katzFD(rnorm(500))                 # higher (noise-like)
```

make_eeg

Create Simulated EEG Data

Description

Generates multi-channel EEG with alpha (10 Hz) and beta (20 Hz) oscillations plus pink noise. Channels are labeled using the International 10-20 system.

Usage

```
make_eeg(n_time = 5000, n_channels = 19, sr = 500)
```

Arguments

`n_time` Number of time points (default: 5000 = 10s at 500 Hz).
`n_channels` Number of EEG channels (default: 19, standard 10-20).
`sr` Sampling rate in Hz (default: 500).

Value

A `PhysioExperiment` with simulated EEG in the "raw" assay. Channel labels follow the International 10-20 system (Fp1, Fp2, F7, ..., O1, O2). Column data contains `label` and `type` fields.

What this generator does and does not model

Each channel is generated independently, so the channels are only weakly correlated: on a 19-channel, 10 s record the median absolute between-channel correlation is about 0.26. Real EEG is spatially smooth and neighbouring electrodes correlate strongly, which several methods rely on. In particular `eegBadChannels` judges a channel partly by its correlation with the others against `corr_threshold` (default 0.4), so on this generator's output it flags every channel as bad – and anything downstream that needs good channels, such as `eegInterpolate`, then has none to work with. That is the generator being independent, not the detector being wrong. Use it for per-channel operations – filtering, spectra, epoching, per-channel measures – and use real data, or a generator with a shared spatial component, when exercising anything that assumes channels covary.

```
pe <- make_eeg(n_time = 2500, n_channels = 4, sr = 250) dim(SummarizedExperiment::assay(pe,
"raw")) # 2500 x 4
```

See Also

`make_eeg_erp()`, `make_eeg_sleep()`, `make_eeg_bci()`, `make_eeg_spikes()`, `eegFilter()`, `eegCoherence()`

make_eeg_bci

Create EEG with Motor Imagery / SSVEP Patterns for BCI

Description

Generates epoched EEG with two classes: left motor imagery (mu ERD over C4) and right motor imagery (mu ERD over C3). Also includes SSVEP at specified frequencies at occipital channels. Data is stored as a 3D array (time x channels x trials).

Usage

```
make_eeg_bci(n_trials = 30, n_channels = 8, sr = 256, trial_sec = 4)
```

Arguments

<code>n_trials</code>	Number of trials per class (default: 30). Total trials will be $2 * n_trials$.
<code>n_channels</code>	Number of channels (default: 8).
<code>sr</code>	Sampling rate in Hz (default: 256).
<code>trial_sec</code>	Trial duration in seconds (default: 4.0).

Value

A `PhysioExperiment` with a 3D "raw" assay (time x channels x total_trials). Sets `metadata(x)$labels` with a character vector of class labels ("left" or "right") for each trial.

See Also

[make_eeg\(\)](#), [eegCSP\(\)](#), [eegBCIfeatures\(\)](#), [eegBCIclassify\(\)](#), [eegMotorImagery\(\)](#)

Examples

```
pe <- make_eeg_bci(n_trials = 5, n_channels = 4, sr = 128)
dim(SummarizedExperiment::assay(pe, "raw")) # 512 x 4 x 10
```

make_eeg_erp	Create EEG with Embedded ERP Components
--------------	---

Description

Generates epoched EEG (3D array: time x channels x epochs) with known ERP components: N100 (negative peak at ~100ms) and P300 (positive peak at ~300ms). Target epochs have larger P300 amplitude than standard epochs.

Usage

```
make_eeg_erp(n_epochs = 40, n_channels = 19, sr = 250, epoch_sec = 1)
```

Arguments

- n_epochs Number of epochs (default: 40).
- n_channels Number of channels (default: 19).
- sr Sampling rate in Hz (default: 250).
- epoch_sec Epoch duration in seconds (default: 1.0).

Value

A `PhysioExperiment` with a 3D "raw" assay (time x channels x epochs) containing simulated ERP data. Sets `metadata(x)$conditions` with a character vector of epoch conditions ("target" or "standard").

See Also

[make_eeg\(\)](#), [eegERPdetect\(\)](#), [eegERPmeasure\(\)](#), [eegERPbaseline\(\)](#), [eegERPtest\(\)](#)

Examples

```
pe <- make_eeg_erp(n_epochs = 10, n_channels = 4, sr = 250)
dim(SummarizedExperiment::assay(pe, "raw")) # 250 x 4 x 10
```

make_eeg_sleep	Create EEG with Sleep Stage Characteristics
----------------	---

Description

Generates continuous EEG with segments having distinct frequency content matching AASM sleep stages: Wake (alpha), N1 (theta), N2 (spindles + K), N3 (delta/SWA), REM (mixed low-voltage). Each 30-second epoch is assigned a stage in a cyclic pattern.

Usage

```
make_eeg_sleep(n_time = 150000, n_channels = 2, sr = 500)
```

Arguments

n_time	Number of time points (default: 150000 = 5min at 500Hz).
n_channels	Number of channels (default: 2, C3 and C4).
sr	Sampling rate in Hz (default: 500).

Value

A `PhysioExperiment` with simulated sleep EEG in the "raw" assay. Sets `metadata(x)$sleep_stages` with a `data.frame` containing columns: `epoch`, `stage` (ground truth), `start_sample`, and `end_sample`.

See Also

[make_eeg\(\)](#), [eegSleepStage\(\)](#), [eegSpindleDetect\(\)](#), [eegKcomplexDetect\(\)](#), [eegSleepMetrics\(\)](#)

Examples

```
pe <- make_eeg_sleep(n_time = 30000, n_channels = 2, sr = 500)
dim(SummarizedExperiment::assay(pe, "raw")) # 30000 x 2
```

make_eeg_spikes	Create EEG with Embedded Epileptic Spikes
-----------------	---

Description

Generates multi-channel EEG with sharp transient spikes at known locations. Spikes are ~50-100ms duration with high amplitude and sharp morphology, inserted at random time points across a subset of channels.

Usage

```
make_eeg_spikes(n_time = 30000, n_channels = 19, sr = 500, n_spikes = 15)
```

Arguments

n_time	Number of time points (default: 30000 = 60s at 500Hz).
n_channels	Number of channels (default: 19).
sr	Sampling rate in Hz (default: 500).
n_spikes	Number of epileptic spikes to insert (default: 15).

Value

A `PhysioExperiment` with simulated clinical EEG in the "raw" assay. Sets `metadata(x)$spike_locations` with a `data.frame` containing columns: `spike_id`, `sample` (sample index), `time_sec` (time in seconds), and `channels` (list column of affected channel indices).

See Also

`make_eeg()`, `eegSpikeDetect()`, `eegQEEG()`, `eegSlowing()`

Examples

```
pe <- make_eeg_spikes(n_time = 5000, n_channels = 4, sr = 250, n_spikes = 3)
dim(SummarizedExperiment::assay(pe, "raw")) # 5000 x 4
```

multiscaleEntropy	<i>Multiscale Entropy</i>
-------------------	---------------------------

Description

Computes the multiscale entropy (MSE; Costa, Goldberger & Peng, 2002) of a one-dimensional signal: the sample entropy of the signal after coarse-graining it at a series of time scales. At scale s the signal is partitioned into non-overlapping windows of length s , each window is replaced by its mean, and the sample entropy (embedding m , tolerance r times the SD of the ORIGINAL signal, held fixed across scales) of that coarse-grained series is the MSE at scale s . A signal whose MSE stays high or rises at coarse scales carries structure across time scales; white noise, by contrast, has an MSE that falls monotonically from scale 1.

Usage

```
multiscaleEntropy(x, m = 2L, r = 0.2, scales = 1:8)
```

Arguments

x	A numeric vector (the time series).
m	Embedding dimension for the sample entropy (default 2).
r	Tolerance as a fraction of the ORIGINAL signal's SD (default 0.2), held fixed across all scales (Costa's convention).
scales	Integer vector of time scales (default 1:8).

Value

A named numeric vector of sample entropies, one per scale (scale1, scale2, ...). NA for a scale whose coarse-grained series is too short or yields no template matches.

References

Costa, M., Goldberger, A. L., & Peng, C.-K. (2002). "Multiscale entropy analysis of complex physiologic time series." *Physical Review Letters*, 89(6), 068102. doi:10.1103/PhysRevLett.89.068102

See Also

[eegComplexity](#) for the multi-measure wrapper; [PhysioMoCap](#)'s `sampleEntropy` for the single-scale measure.

Examples

```
set.seed(1); multiscaleEntropy(rnorm(1000), scales = 1:5) # noise: falls with scale
```

petrosianFD

Petrosian Fractal Dimension

Description

Computes the Petrosian fractal dimension of a one-dimensional signal (Petrosian, 1995) – a fast waveform-complexity measure based on the number of sign changes in the signal's derivative (i.e. the density of local extrema). A smooth, oscillatory signal has few derivative sign changes and a Petrosian FD near 1; a complex or noise-like signal has many and a higher FD. $P = \log_{10} N / (\log_{10} N + \log_{10}(N / (N + 0.4 N_\delta)))$, where N is the signal length and N_δ the number of sign changes in the first difference.

Usage

```
petrosianFD(x)
```

Arguments

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

Value

A single numeric value, the Petrosian fractal dimension (typically in $[1, \sim 1.1]$ for physiological signals).

References

Petrosian, A. (1995). "Kolmogorov complexity of finite sequences and recognition of different preictal EEG patterns." In *Proceedings of the Eighth IEEE Symposium on Computer-Based Medical Systems*, 212–217. doi:10.1109/CBMS.1995.465426

See Also

[svdEntropy](#) for the singular-value complexity view, [eegComplexity](#) for the multi-measure wrapper.

Examples

```
petrosianFD(sin(seq(0, 20 * pi, length.out = 500))) # low (smooth oscillation)
set.seed(1); petrosianFD(rnorm(500))                 # higher (noise-like)
```

phaseEntropy

Phase Entropy

Description

Computes the phase entropy (PhasEn) of a one-dimensional signal (Rohila & Sharma, 2019) from its Second-Order Difference Plot (SODP). Each point plots the first difference $x(n + \tau) - x(n)$ against the second difference $x(n + 2\tau) - x(n + \tau)$; the polar angle of each point is binned into k equal sectors of $[0, 2\pi)$, and PhasEn is the (normalized) Shannon entropy of the angle-weighted sector distribution. It summarizes how the signal's local rate-of-change and its change-of-rate co-vary – a phase-plane geometry distinct from the amplitude, template-matching and symbolic (dispersion/increment/slope) entropies.

Usage

```
phaseEntropy(x, delay = 1L, k = 4L)
```

Arguments

x	A numeric vector (the time series).
delay	Time delay (lag) in samples for the differences (default 1).
k	Number of angular sectors of the SODP (default 4).

Details

This implementation reproduces NeuroKit2's `entropy_phase` bit-for-bit, including its normalization (a base-2 Shannon term divided by the natural-log $\ln k$, so PhasEn ranges in $[0, 1/\ln 2]$).

Value

A single numeric value, the phase entropy.

References

Rohila, A., & Sharma, A. (2019). "Phase entropy: a new complexity measure for heart rate variability." *Physiological Measurement*, 40(10), 105006. doi:10.1088/13616579/ab499e

See Also

[slopeEntropy](#), [incrementEntropy](#), [dispersionEntropy](#).

Examples

```
set.seed(1)
phaseEntropy(rnorm(1000), k = 4)
```

renyiEntropy

Renyi Entropy

Description

Computes the Renyi entropy of order alpha of a one-dimensional signal (Renyi, 1961) – a one-parameter generalization of the Shannon entropy of the signal's amplitude distribution. The signal is discretized into bins equal-width bins to form a probability distribution p , and $H_\alpha = \frac{1}{1-\alpha} \ln(\sum_i p_i^\alpha)$ is returned in nats. Special cases: $\alpha \rightarrow 1$ is the Shannon entropy, $\alpha = 2$ the collision entropy, and $\alpha \rightarrow \infty$ the min-entropy. H_α is non-increasing in α .

Usage

```
renyiEntropy(x, alpha = 2, bins = 16L)
```

Arguments

x	A numeric vector (the time series).
alpha	Order of the Renyi entropy (default 2, the collision entropy); alpha = 1 returns the Shannon entropy.
bins	Number of equal-width bins for the amplitude histogram (default 16).

Value

A single numeric value, the Renyi entropy in nats.

References

Renyi, A. (1961). "On measures of entropy and information." In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*, Volume 1, 547–561.

See Also

[svdEntropy](#), [petrosianFD](#), [eegComplexity](#) for other complexity measures.

Examples

```
set.seed(1)
renyiEntropy(rnorm(1000), alpha = 2) # collision entropy
renyiEntropy(rnorm(1000), alpha = 1) # = Shannon entropy
```

slopeEntropy

*Slope Entropy***Description**

Computes the slope entropy (SlopEn) of a one-dimensional signal (Cuesta-Frau, 2019) – a symbolic complexity measure defined on the *slopes* between successive samples. Each first difference is turned into the *angle* $\theta = \arctan(\Delta x)$ (in degrees) and mapped to one of five symbols by two angle thresholds: steep-up ($> \gamma$), gentle-up ($(\delta, \gamma]$), flat ($[-\delta, \delta]$), gentle-down and steep-down; SlopEn is the Shannon entropy of the resulting length-(m-1) slope patterns. By combining a coarse (steep/gentle) and fine (flat) angular resolution it captures the shape of local trends, a mechanism distinct from the amplitude, NCDF-pattern, template-matching and increment-magnitude entropies.

Usage

```
slopeEntropy(x, dimension = 3L, thresholds = c(0.1, 45), delay = 1L)
```

Arguments

x	A numeric vector (the time series).
dimension	Embedding dimension m (default 3); slope patterns have length dimension - 1.
thresholds	Increasing positive angle thresholds (degrees) that split the slope into symbol classes (default c(0.1, 45): a flat band within +/-0.1 deg, a gentle band up to +/-45 deg, and a steep class beyond).
delay	Time delay (lag) in samples for the difference (default 1).

Details

This implementation reproduces NeuroKit2's entropy_slope bit-for-bit.

Value

A single numeric value, the slope entropy (bits).

References

Cuesta-Frau, D. (2019). "Slope Entropy: A New Time Series Complexity Estimator Based on Both Symbolic Patterns and Amplitude Information." *Entropy*, 21(12), 1167. doi:10.3390/e21121167

See Also

[incrementEntropy](#), [dispersionEntropy](#), [fuzzyEntropy](#).

Examples

```
set.seed(1)
slopeEntropy(rnorm(1000), dimension = 3)
```

svdEntropy

*Singular Value Decomposition (SVD) Entropy***Description**

Computes the SVD entropy of a one-dimensional signal (Roberts et al., 1999) – a measure of the signal's dimensionality / the number of eigenvectors needed to explain it. The signal is time-delay embedded into an order-dimensional matrix; the normalized singular values of that matrix are treated as a probability distribution, and their Shannon entropy is the SVD entropy. Low SVD entropy indicates a low-dimensional (structured, e.g. oscillatory) signal; high SVD entropy indicates a high-dimensional (complex or noise-like) signal.

Usage

```
svdEntropy(x, order = 3L, delay = 1L, normalize = FALSE)
```

Arguments

x	A numeric vector (the time series).
order	Embedding dimension (default 3); the number of singular values.
delay	Embedding delay / lag in samples (default 1).
normalize	If TRUE, divide by $\log_2(\text{order})$ to bound the result in $[0, 1]$; otherwise return the entropy in bits (default FALSE).

Value

A single numeric value, the SVD entropy (in bits, or normalized).

References

Roberts, S. J., Penny, W., & Rezek, I. (1999). "Temporal and spatial complexity measures for electroencephalogram based brain-computer interfacing." *Medical & Biological Engineering & Computing*, 37(1), 93–98. doi:[10.1007/BF02513272](https://doi.org/10.1007/BF02513272)

See Also

[eegComplexity](#) for the multi-measure complexity wrapper (permutation entropy, Lempel-Ziv, Hjorth, DFA, ...).

Examples

```
set.seed(1)
svdEntropy(sin(seq(0, 20 * pi, length.out = 500))) # low (structured)
svdEntropy(rnorm(500))                             # high (noise-like)
```

tsallisEntropy	<i>Tsallis Entropy</i>
----------------	------------------------

Description

Computes the Tsallis entropy of order q of a one-dimensional signal (Tsallis, 1988) – the *non-additive* one-parameter generalization of the Shannon entropy of the signal's amplitude distribution, complementary to the (additive) [renyiEntropy](#). The signal is discretized into bins equal-width bins to form a probability distribution p , and $S_q = \frac{1 - \sum_i p_i^q}{q-1}$ is returned (equivalently $\sum_i p_i \ln_q(1/p_i)$ with the q -logarithm). $q \rightarrow 1$ recovers the Shannon entropy (in nats); unlike Renyi, Tsallis is non-additive ($S_q(A, B) = S_q(A) + S_q(B) + (1 - q)S_q(A)S_q(B)$ for independent A, B), so Renyi and Tsallis agree only at $q = 1$.

Usage

```
tsallisEntropy(x, q = 2, bins = 16L)
```

Arguments

<code>x</code>	A numeric vector (the time series).
<code>q</code>	Order of the Tsallis entropy (default 2); $q = 1$ returns the Shannon entropy.
<code>bins</code>	Number of equal-width bins for the amplitude histogram (default 16).

Value

A single numeric value, the Tsallis entropy.

References

Tsallis, C. (1988). "Possible generalization of Boltzmann-Gibbs statistics." *Journal of Statistical Physics*, 52(1-2), 479–487. doi:[10.1007/BF01016429](#)

See Also

[renyiEntropy](#) for the additive generalization, [svdEntropy](#), [petrosianFD](#).

Examples

```
set.seed(1)
tsallisEntropy(rnorm(1000), q = 2) # non-additive (collision-like)
tsallisEntropy(rnorm(1000), q = 1) # = Shannon entropy
```

Index

dispersionEntropy, [4](#), [100](#), [101](#), [109](#), [110](#)

eegAperiodic, [5](#)
eegAperiodic(), [20](#)
eegArtifactReject, [7](#)
eegArtifactReject(), [10](#)
eegAsymmetry, [8](#)
eegAsymmetry(), [16](#), [81](#), [86](#), [91](#)
eegAutoReject, [9](#)
eegBadChannels, [10](#), [103](#)
eegBCIclassify, [11](#), [26](#)
eegBCIclassify(), [13](#), [23](#), [60](#), [94](#), [104](#)
eegBCIfeatures, [11](#), [12](#), [26](#)
eegBCIfeatures(), [12](#), [23](#), [60](#), [94](#), [104](#)
eegBeamformer, [13](#)
eegBeamformer(), [41](#), [70](#), [89](#), [90](#)
eegBSI, [15](#)
eegBSI(), [24](#)
eegCoherence, [16](#)
eegCoherence(), [22](#), [42](#), [78](#), [99](#), [103](#)
eegComodulogram, [17](#)
eegComodulogram(), [63](#)
eegComplexity, [19](#), [102](#), [107–109](#), [111](#)
eegComplexity(), [6](#)
eegConditionalGC, [20](#)
eegConditionalGC(), [29](#), [64](#)
eegConnectivityMatrix, [21](#)
eegConnectivityMatrix(), [17](#), [29](#), [42](#), [63](#),
[64](#), [67](#), [78](#), [97](#), [99](#)
eegCSP, [22](#), [26](#)
eegCSP(), [12](#), [13](#), [53](#), [60](#), [94](#), [104](#)
eegDAR, [23](#)
eegDAR(), [16](#)
eegDecode, [25](#)
eegDipoleFit, [27](#)
eegDTF, [28](#)
eegDTF(), [21](#), [22](#), [64](#)
eegEpoch, [29](#)
eegEpoch(), [9](#), [10](#), [31](#), [32](#)
eegERPbaseline, [30](#)

eegERPbaseline(), [32](#), [34](#), [35](#), [104](#)
eegERPdetect, [31](#)
eegERPdetect(), [31–35](#), [37](#), [104](#)
eegERPdifference, [32](#)
eegERPdifference(), [33](#), [37](#)
eegERPgrandAverage, [33](#)
eegERPgrandAverage(), [32](#), [37](#)
eegERPlatency, [34](#)
eegERPlatency(), [32](#), [35](#)
eegERPmeasure, [35](#)
eegERPmeasure(), [31–34](#), [37](#), [104](#)
eegERPtest, [36](#)
eegERPtest(), [32](#), [33](#), [104](#)
eegERSP, [37](#)
eegERSP(), [51](#), [59](#)
eegFilter, [38](#)
eegFilter(), [31](#), [43](#), [103](#)
eegForwardModel, [14](#), [39](#), [88](#)
eegForwardModel(), [14](#), [27](#), [70](#), [89](#), [90](#)
eegGrangerCausality, [41](#)
eegGrangerCausality(), [20–22](#)
eegICA, [42](#)
eegICA(), [44–48](#)
eegICAdetect, [43](#)
eegICAdetect(), [43](#), [45–48](#)
eegICAmix, [44](#)
eegICAmix(), [43](#), [44](#), [46](#)
eegICAremove, [45](#)
eegICAremove(), [43–45](#), [47](#), [48](#)
eegICLabel, [46](#)
eegICLabel(), [43](#), [48](#)
eegICLabelFlag, [48](#)
eegICLabelFlag(), [47](#)
eegInterpolate, [10](#), [49](#), [103](#)
eegInterpolate(), [9](#), [10](#)
eegITC, [50](#)
eegITC(), [38](#)
eegKcomplexDetect, [51](#)
eegKcomplexDetect(), [84](#), [85](#), [88](#), [93](#), [105](#)

- eegLateralization, 52
- eegMicrostateBackfit, 54
- eegMicrostateBackfit(), 56, 57
- eegMicrostates, 55, 57
- eegMicrostates(), 54, 56, 57
- eegMicrostateSequence, 56
- eegMicrostateSequence(), 54, 56, 57
- eegMicrostateStats, 57
- eegMicrostateStats(), 54, 56
- eegMontage, 49, 58
- eegMontage(), 9, 10
- eegMorletWavelet, 58
- eegMorletWavelet(), 38, 51, 62, 95
- eegMotorImagery, 60
- eegMotorImagery(), 12, 13, 23, 52, 53, 104
- eegMultitaper, 61
- eegMultitaper(), 59, 95
- eegPAC, 62
- eegPAC(), 18
- eegPDC, 63
- eegPDC(), 21, 22, 29
- eegPlotButterflyGFP, 64
- eegPlotConnectivity, 65
- eegPlotConnectivity(), 17, 22, 67, 78, 99
- eegPlotConnectogram, 66
- eegPlotERP, 68
- eegPlotERP(), 32, 34, 35
- eegPlotGlassBrain, 69
- eegPlotGlassBrain(), 14, 89
- eegPlotHypnogram, 70
- eegPlotHypnogram(), 84, 85
- eegPlotICA, 71
- eegPlotSignal, 72
- eegPlotSource, 73
- eegPlotSource(), 70, 89
- eegPlotSpectrogram, 74
- eegPlotSpectrogram(), 38, 51, 59, 62, 81, 95
- eegPlotTopomap, 75
- eegPlotTopomap(), 6, 97
- eegPlotTopomapSeries, 76
- eegPLV, 77
- eegPLV(), 17, 22, 42, 99
- eegPreprocess, 78
- eegPreprocess(), 43
- eegQEEG, 80
- eegQEEG(), 6, 8, 20, 24, 86, 91, 96, 106
- eegQuickStart, 81
- eegRereference, 82
- eegRereference(), 97
- eegSleepMetrics, 83
- eegSleepMetrics(), 52, 85, 88, 93, 105
- eegSleepStage, 84
- eegSleepStage(), 52, 84, 88, 93, 105
- eegSlowing, 86
- eegSlowing(), 8, 24, 81, 91, 96, 106
- eegSlowWaveDetect, 87
- eegSlowWaveDetect(), 52, 84, 85, 93
- eegSourceEstimate, 88, 89
- eegSourceEstimate(), 14, 27, 41, 70, 90
- eegSourcePower, 89
- eegSourcePower(), 14, 41, 89
- eegSpikeDetect, 90
- eegSpikeDetect(), 8, 81, 86, 96, 106
- eegSpindleDetect, 91
- eegSpindleDetect(), 52, 84, 85, 88, 105
- eegSSVEP, 93
- eegSTFT, 94
- eegSTFT(), 59, 62
- eegSuppression, 95
- eegSuppression(), 86
- eegSurfaceLaplacian, 97
- eegWPLI, 98
- eegWPLI(), 17, 22, 42, 78
- fuzzyEntropy, 99, 100, 101, 110
- incrementEntropy, 100, 109, 110
- katzFD, 101
- make_eeg, 102
- make_eeg(), 81, 104–106
- make_eeg_bci, 103
- make_eeg_bci(), 81, 103
- make_eeg_erp, 104
- make_eeg_erp(), 9, 81, 103
- make_eeg_sleep, 105
- make_eeg_sleep(), 81, 103
- make_eeg_spikes, 105
- make_eeg_spikes(), 103
- metadata, 23, 37, 50, 59, 61, 94
- multiscaleEntropy, 106
- normativeLookup, 23
- petrosianFD, 5, 102, 107, 109, 112
- phaseEntropy, 108
- PhysioBiomarker, 15, 24, 53

physioBiomarker, [15](#), [23](#)
PhysioExperiment::mvarFit(), [21](#), [28](#), [29](#),
 [63](#), [64](#)
PhysioExperiment::normativeLookup(),
 [24](#)
PhysioExperiment::physioBiomarker(),
 [16](#), [24](#), [53](#)

renyiEntropy, [4](#), [5](#), [100](#), [101](#), [109](#), [112](#)

slopeEntropy, [109](#), [110](#)
svdEntropy, [5](#), [100](#), [102](#), [108](#), [109](#), [111](#), [112](#)

tsallisEntropy, [4](#), [5](#), [100](#), [101](#), [112](#)

wpliEstimate, [99](#)