Package: PhysioEEG (via r-universe)

May 16, 2026

Title EEG Analysis Functions for PhysioExperiment Objects

Version 0.2.0

Author Yusuke Matsui

Maintainer Yusuke Matsui <you@example.com>

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), PhysioCore

Imports SummarizedExperiment, S4Vectors, utils, stats

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

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Collate 'eeg-data.R' 'eeg-preprocess.R' 'eeg-ica.R' 'eeg-erp.R' 'eeg-source.R' 'eeg-microstate.R' 'eeg-sleep.R'

'eeg-timefreq.R' 'eeg-connectivity.R' 'eeg-bci.R'
'eeg-clinical.R' 'eeg-vis.R' 'zzz.R'

RoxygenNote 7.3.3

Config/pak/sysreqs zlib1g-dev

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

Date/Publication 2026-03-16 11:31:13 UTC

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

RemoteRef HEAD

RemoteSha 578ea8183885d97f2fe2114b2cd41d5259698748

Contents

eegArtifactReject . . . 3
eegAsymmetry . . . 4
eegBadChannels . . . 6
eegBCIclassify . . . 7
eegBCIfeatures . . . 8
eegBeamformer . . . 9
eegCoherence . . . 10
eegConnectivityMatrix . . . 12
eegCSP . . . 13
eegEpoch . . . 14
eegERPbaseline . . . 15
eegERPdetect . . . 16
eegERPdifference . . . 17
eegERPgrandAverage . . . 18
eegERPlatency . . . 19
eegERPmeasure . . . 20
eegERPtest . . . 21
eegERSP . . . 22
eegFilter . . . 24
eegForwardModel . . . 25
eegGrangerCausality . . . 26
eegICA . . . 27
eegICAdetect . . . 29
eegICAmix . . . 30
eegICAremove . . . 31
eegInterpolate . . . 32
eegITC . . . 33
eegKcomplexDetect . . . 34
eegMicrostateBackfit . . . 35
eegMicrostates . . . 36
eegMicrostateSequence . . . 38
eegMicrostateStats . . . 39
eegMontage . . . 40

eegMorletWavelet . . . 40
eegMotorImagery . . . 42
eegMultitaper . . . 43
eegPlotConnectivity . . . 44
eegPlotERP . . . 45
eegPlotHypnogram . . . 46
eegPlotICA . . . 47
eegPlotSignal . . . 48
eegPlotSource . . . 49
eegPlotSpectrogram . . . 50
eegPlotTopomap . . . 51
eegPlotTopomapSeries . . . 52
eegPLV . . . 53
eegPreprocess . . . 54
eegQEEG . . . 56
eegQuickStart . . . 57
eegRereference . . . 58
eegSleepMetrics . . . 59
eegSleepStage . . . 60
eegSlowing . . . 61
eegSlowWaveDetect . . . 62
eegSourceEstimate . . . 64
eegSourcePower . . . 65
eegSpikeDetect . . . 66
eegSpindleDetect . . . 68
eegSSVEP . . . 69
eegSTFT . . . 70
eegSuppression . . . 71
eegWPLI . . . 73
make_eeg . . . 74
make_eeg_bci . . . 75
make_eeg_erp . . . 75
make_eeg_sleep . . . 76
make_eeg_spikes . . . 77

Index **78**

`eegArtifactReject` *Reject artifacts in epoched EEG data*

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

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

```
## Not run:
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")

## End(Not run)
```

---

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

```
## Not run:
pe <- make_eeg(n_time = 5000, n_channels = 19, sr = 500)
asym <- eegAsymmetry(pe)
print(asym)

## End(Not run)
```

`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

`x`	A PhysioExperiment object.
`method`	Detection method(s) to apply: `"all"` runs all checks, or specify one or more of `"flat"`, `"noise"`, `"correlation"`.
`flat_threshold`	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

```
## Not run:
pe <- make_eeg(n_time = 5000, n_channels = 19, sr = 500)
bad_df <- eegBadChannels(pe)
bad_labels <- bad_df$channel[bad_df$is_bad]

## End(Not run)
```

`eegBCIclassify` *BCI Classification*

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 `attr(result, "cv_accuracy")`.
`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

```
## Not run:
pe <- make_eeg_bci(n_trials = 20, n_channels = 8, sr = 256)
labels <- 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")

## End(Not run)
```

---

eegBCIfeatures *BCI Feature Extraction*

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: list(mu = c(8, 13), beta = c(13, 30)).
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

```
## Not run:
pe <- make_eeg_bci(n_trials = 20, n_channels = 8, sr = 256)
features <- eegBCIfeatures(pe, method = "bandpower")

## End(Not run)
```

`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

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

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.

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()

Examples

```
## Not run:
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")

## End(Not run)
```

---

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

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

```
## Not run:
pe <- make_eeg(n_time = 5000, n_channels = 4, sr = 500)
pe <- eegCoherence(pe, method = "coherence", band = c(8, 13))
coh_matrix <- metadata(pe)$connectivity$matrix

## End(Not run)
```

---

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"),
  band = c(8, 13),
  ...
)
```

Arguments

x	A PhysioExperiment object with EEG data (2D: time x channels).
method	Connectivity method: "coherence" for magnitude-squared coherence, "plv" for Phase Locking Value, or "wpli" for weighted Phase Lag Index (default: "coherence").
band	Numeric vector of length 2 specifying the frequency band in Hz (default: c(8, 13) for alpha band).
...	Additional arguments passed to the underlying connectivity function (e.g., window_sec, overlap).

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

```
## Not run:
pe <- make_eeg(n_time = 5000, n_channels = 4, sr = 500)
conn <- eegConnectivityMatrix(pe, method = "plv", band = c(8, 13))
print(conn)

## End(Not run)
```

---

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	Output assay name for CSP features (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

```
## Not run:
pe <- make_eeg_bci(n_trials = 30, n_channels = 8, sr = 256)
labels <- metadata(pe)$labels
result <- eegCSP(pe, labels = labels, n_filters = 3)
csp_features <- SummarizedExperiment::assay(result, "csp")

## End(Not run)
```

---

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

x	A PhysioExperiment object with continuous (2D) data.
events	A data.frame with an onset_sec column specifying event times in seconds, OR an integer vector of sample indices.
limits	Numeric vector of length 2 specifying the epoch window relative to each event in seconds (default: c(-0.2, 0.8)).
baseline	Numeric vector of length 2 specifying the baseline window relative to each event in seconds (default: c(-0.2, 0)). Set to NULL to skip baseline correction.

assay_name	Name of the assay to epoch. If NULL, uses defaultAssay(x).
output_assay	Name of the output assay (default: "epoched").

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

```
## Not run:
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))
dim(SummarizedExperiment::assay(pe_ep, "epoched"))
# time x channels x epochs

## End(Not run)
```

---

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 PhysioExperiment object with epoched (3D) EEG data.
baseline	Baseline period in milliseconds as c(start, end). For example, c(-200, 0) for 200ms pre-stimulus baseline. Default is c(-200, 0).
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: "baseline_corrected").

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

```
## Not run:
pe <- make_eeg_erp(n_epochs = 40, sr = 250)
result <- eegERPbaseline(pe, baseline = c(-200, 0), epoch_start = -200)

## End(Not run)
```

---

eegERPdetect *Detect ERP Components*

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

```
## Not run:
pe <- make_eeg_erp(n_epochs = 40, sr = 250)
result <- eegERPdetect(pe, component = "P300")

## End(Not run)
```

---

eegERPdifference *ERP Difference Waveform*

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

```
## Not run:
pe1 <- make_eeg_erp(n_epochs = 20, sr = 250)
pe2 <- make_eeg_erp(n_epochs = 20, sr = 250)
result <- eegERPdifference(pe1, pe2)

## End(Not run)
```

---

eegERPgrandAverage *Compute Grand Average ERP*

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

...	PhysioExperiment objects to average, or a list of them.
assay_name	Input assay name (default: first assay).
output_assay	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

```
## Not run:
pe1 <- make_eeg_erp(n_epochs = 20, sr = 250)
pe2 <- make_eeg_erp(n_epochs = 20, sr = 250)
result <- eegERPgrandAverage(pe1, pe2)

## End(Not run)
```

---

eegERPlatency *Fractional Area Latency*

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

```
## Not run:
pe <- make_eeg_erp(n_epochs = 40, sr = 250)
result <- eegERPlatency(pe, window = c(250, 500), fraction = 0.5,
                        polarity = "positive")

## End(Not run)
```

---

eegERPmeasure *Measure ERP Amplitude*

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

```
## Not run:
pe <- make_eeg_erp(n_epochs = 40, sr = 250)
result <- eegERPmeasure(pe, window = c(250, 500), method = "peak",
                        polarity = "positive")

## End(Not run)
```

---

eegERPtest *Statistical Testing for ERP Differences*

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

```
## Not run:
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)

## End(Not run)
```

---

eegERSP *Event-Related Spectral Perturbation (ERSP)*

Description

Computes Event-Related Spectral Perturbation for epoched (3D) EEG data. ERSP 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

x	A PhysioExperiment object with epoched (3D) EEG data (time x channels x epochs).
baseline	Numeric vector of length 2 specifying the baseline time window in sample indices (e.g., c(1, 50) for the first 50 samples). Default is c(1, 50).
frequencies	Numeric vector of frequencies in Hz to analyze. If NULL, defaults to seq(1, 50, by = 1).
n_cycles	Number of cycles for the Morlet wavelet (default: 7).
assay_name	Name of the input assay. If NULL, the default assay is used.
output_assay	Name of the assay to store ERSP results (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

```
## Not run:
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))
ersp_data <- SummarizedExperiment::assay(pe_ersp, "ersp")
dim(ersp_data)  # time x frequencies x channels

## End(Not run)
```

`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 PhysioExperiment object.
`lowcut`	Low cutoff frequency in Hz. If NULL, no highpass filtering is applied (set both lowcut and highcut for bandpass).
`highcut`	High cutoff frequency in Hz. If NULL, 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 NULL, 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 NULL). For IIR, this is the Butterworth order (default: 4).
`assay_name`	Name of the assay to filter. If NULL, uses `defaultAssay(x)`.
`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

```
## Not run:
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)

## End(Not run)
```

---

eegForwardModel *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"),
  n_sources = 500,
  assay_name = NULL
)
```

Arguments

`x`	A PhysioExperiment object with EEG data.
`method`	Forward model method: `"spherical"` (current dipole in infinite homogeneous medium with conductivity 0.33 S/m) or `"bem_simplified"` (3-shell Berg and Scherg 1994 approximation using virtual dipoles at scaled eccentricities).
`n_sources`	Number of dipole sources to distribute inside the head model (default: 500).
`assay_name`	Name of the assay to reference for channel count. If `NULL`, 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.

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

```
## Not run:
pe <- make_eeg(n_time = 1000, n_channels = 19, sr = 250)
fm <- eegForwardModel(pe, method = "spherical", n_sources = 100)
dim(fm$leadfield)

## End(Not run)
```

`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

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

Details

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

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

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

```
## Not run:
pe <- make_eeg(n_time = 5000, n_channels = 4, sr = 500)
gc_df <- eegGrangerCausality(pe, order = 5, band = c(8, 13))
head(gc_df)

## End(Not run)
```

---

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

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

```
## Not run:
pe <- make_eeg(n_time = 5000, sr = 500)
result <- eegICA(pe, n_components = 4, method = "fastica")

## End(Not run)
```

`eegICAdetect` *Detect Artifact ICA Components*

Description

Automatically identifies artifact components using one of three methods: correlation with frontal channels, kurtosis, or spatial weight pattern.

Usage

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

Arguments

`x`	A PhysioExperiment object with ICA results (from `eegICA`).
`method`	Detection method: `"correlation"` (frontal channel correlation), `"kurtosis"` (excess kurtosis), or `"spatial"` (spatial weight pattern).
`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

```
## Not run:
pe <- make_eeg(n_time = 5000, sr = 500)
pe <- eegICA(pe, n_components = 4, method = "fastica")
artifacts <- eegICAdetect(pe, method = "kurtosis")

## End(Not run)
```

`eegICAmix` *Access ICA Metadata*

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

```
## Not run:
pe <- make_eeg(n_time = 5000, sr = 500)
pe <- eegICA(pe, n_components = 4, method = "fastica")
ica_info <- eegICAmix(pe)

## End(Not run)
```

`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 PhysioExperiment object with ICA results (from `eegICA`).
`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

```
## Not run:
pe <- make_eeg(n_time = 5000, sr = 500)
pe <- eegICA(pe, n_components = 4, method = "fastica")
pe <- eegICAremove(pe, components = c(1, 2))

## End(Not run)
```

`eegInterpolate` *Interpolate bad EEG channels*

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

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

```
## Not run:
pe <- make_eeg(n_time = 5000, n_channels = 19, sr = 500)
pe <- eegMontage(pe, system = "10-20")
bad_df <- eegBadChannels(pe)
bad_labels <- bad_df$channel[bad_df$is_bad]
if (length(bad_labels) > 0) {
  pe_clean <- eegInterpolate(pe, bad_labels)
}

## End(Not run)
```

`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

`x`	A PhysioExperiment object with epoched (3D) EEG data (time x channels x epochs).
`frequencies`	Numeric vector of frequencies in Hz to analyze. If `NULL`, defaults to `seq(1, 50, by = 1)`.
`n_cycles`	Number of cycles for the Morlet wavelet (default: 7).
`assay_name`	Name of the input assay. If `NULL`, the default assay is used.
`output_assay`	Name of the assay to store ITC results (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

```
## Not run:
pe <- make_eeg_erp(n_epochs = 40, n_channels = 2, sr = 250)
pe_itc <- eegITC(pe, frequencies = seq(5, 30, by = 5))
itc_data <- SummarizedExperiment::assay(pe_itc, "itc")
dim(itc_data)  # time x frequencies x channels

## End(Not run)
```

---

eegKcomplexDetect *Detect K-Complexes*

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 PhysioExperiment object with EEG data.
min_neg_amplitude	Minimum absolute negative peak amplitude in microvolts (default: 75). Peaks must be more negative than -min_neg_amplitude.
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 NULL, 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

```
## Not run:
pe <- make_eeg_sleep(n_time = 150000, n_channels = 2, sr = 500)
kcomplexes <- eegKcomplexDetect(pe)
head(kcomplexes)

## End(Not run)
```

---

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

```
## Not run:
pe1 <- make_eeg(n_time = 5000, n_channels = 19, sr = 500)
pe1 <- eegMicrostates(pe1, n_states = 4)
maps <- metadata(pe1)$microstates$maps

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

## End(Not run)
```

---

eegMicrostates *EEG Microstate Segmentation*

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

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

```
## Not run:
pe <- make_eeg(n_time = 5000, n_channels = 19, sr = 500)
pe <- eegMicrostates(pe, n_states = 4, method = "kmeans")
ms <- metadata(pe)$microstates

## End(Not run)
```

`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

```
## Not run:
pe <- make_eeg(n_time = 2000, n_channels = 19, sr = 500)
pe <- eegMicrostates(pe, n_states = 4)
seq_labels <- eegMicrostateSequence(pe)
table(seq_labels)

## End(Not run)
```

`eegMicrostateStats` *Compute Microstate Statistics*

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

```
## Not run:
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")

## End(Not run)
```

eegMontage *Assign electrode montage positions*

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

```
## Not run:
pe <- make_eeg(n_time = 5000, n_channels = 19, sr = 500)
pe <- eegMontage(pe, system = "10-20")
head(SummarizedExperiment::colData(pe))

## End(Not run)
```

eegMorletWavelet *Morlet Wavelet Transform for EEG*

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

x	A PhysioExperiment object with EEG data (2D: time x channels).
frequencies	Numeric vector of frequencies in Hz to analyze. If NULL, defaults to seq(1, 50, by = 1).
n_cycles	Number of cycles in the Morlet wavelet, controlling the trade-off between time and frequency resolution (default: 7).
assay_name	Name of the input assay. If NULL, the default assay is used.
output_assay	Name of the assay to store wavelet power results (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

```
## Not run:
pe <- make_eeg(n_time = 5000, n_channels = 4, sr = 500)
pe_wt <- eegMorletWavelet(pe, frequencies = seq(5, 40, by = 1))
wp <- SummarizedExperiment::assay(pe_wt, "wavelet_power")
dim(wp)  # time x frequencies x channels

## End(Not run)
```

`eegMotorImagery` *Motor Imagery ERD/ERS Computation*

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

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

```
## Not run:
pe <- make_eeg_bci(n_trials = 20, n_channels = 8, sr = 256)
result <- eegMotorImagery(pe)
erd_data <- SummarizedExperiment::assay(result, "erd_ers")

## End(Not run)
```

`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 PhysioExperiment 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 `NULL`, defaults to `floor(2 * bandwidth) - 1`.
`assay_name`	Name of the input assay. If `NULL`, the default assay is used.
`output_assay`	Name of the assay to store PSD results (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

```
## Not run:
pe <- make_eeg(n_time = 5000, n_channels = 19, sr = 500)
pe_mt <- eegMultitaper(pe, bandwidth = 4)
psd <- SummarizedExperiment::assay(pe_mt, "multitaper_psd")
dim(psd)  # frequencies x channels

## End(Not run)
```

---

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

`x`	A PhysioExperiment object with EEG data.
`method`	Display method: `"heatmap"` for a correlation/connectivity matrix heatmap or `"circle"` for a circular connectivity diagram.
`matrix`	Numeric matrix of connectivity values. If `NULL`, reads from `metadata(x)$connectivity$matrix`.
`threshold`	Numeric; only show connections above this value (default: 0).

labels	Character vector of channel labels. If NULL, uses row/column names of the matrix or channel labels from colData.
palette	Character name of the diverging color palette (default: "RdBu").

Value

A ggplot2 object.

Examples

```
## Not run:
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)

## End(Not run)
```

---

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

x	A PhysioExperiment object with epoched (3D) EEG data (time x channels x epochs).
channels	Character vector of channel labels to plot. If NULL, the grand average across all channels is used.
conditions	Character vector of condition labels to include. If NULL and `metadata(x)$conditions` exists, all conditions are plotted.
ci	Confidence level for the interval (default: 0.95).

show_ci	Logical; if TRUE, display confidence interval ribbon.
epoch_start	Numeric start time of each epoch in seconds for the x-axis (default: 0).
assay_name	Input assay name. If NULL, uses the default assay.

Value

A ggplot2 object.

Examples

```
## Not run:
pe <- make_eeg_erp(n_epochs = 40, sr = 250)
eegPlotERP(pe, channels = "Cz")

## End(Not run)
```

---

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

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

Value

A ggplot2 object.

Examples

```
## Not run:
pe <- make_eeg_sleep(n_time = 150000, n_channels = 2, sr = 500)
stages <- eegSleepStage(pe)
metadata(pe)$sleep_stages <- stages
eegPlotHypnogram(pe)

## End(Not run)
```

`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

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

Value

A ggplot2 object.

Examples

```
## Not run:
pe <- make_eeg(n_time = 5000, n_channels = 19, sr = 500)
pe <- eegICA(pe, n_components = 10)
eegPlotICA(pe, components = 1:5)

## End(Not run)
```

`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

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

Value

A ggplot2 object.

Examples

```
## Not run:
pe <- make_eeg(n_time = 2500, n_channels = 4, sr = 500)
eegPlotSignal(pe, mode = "stacked")

## End(Not run)
```

`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

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

Value

A ggplot2 object.

Examples

```
## Not run:
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")

## End(Not run)
```

`eegPlotSpectrogram` *Plot Spectrogram (Time-Frequency Heatmap)*

Description

Displays a time-frequency representation of EEG data as a heatmap. Expects the assay to contain 3D time-frequency data or a pre-computed spectrogram. For 2D data, a simple Welch-based spectrogram is computed using sliding window FFT.

Usage

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

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*log10(power)` (default: TRUE).
`palette`	Character name of the color palette (default: `"viridis"`).
`assay_name`	Input assay name. If `NULL`, uses the default assay.

Value

A ggplot2 object.

Examples

```
## Not run:
pe <- make_eeg(n_time = 5000, n_channels = 4, sr = 500)
eegPlotSpectrogram(pe, channel = 1)

## End(Not run)
```

`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

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

Value

A ggplot2 object.

Examples

```
## Not run:
pe <- make_eeg(n_time = 500, n_channels = 19, sr = 250)
eegPlotTopomap(pe, time = 0.5)

## End(Not run)
```

`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

```
## Not run:
pe <- make_eeg(n_time = 2500, n_channels = 19, sr = 500)
eegPlotTopomapSeries(pe, times = c(0.1, 0.2, 0.3))

## End(Not run)
```

`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

```
## Not run:
pe <- make_eeg(n_time = 5000, n_channels = 4, sr = 500)
plv_df <- eegPLV(pe, band = c(8, 13))
head(plv_df)

## End(Not run)
```

`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

`x`	A PhysioExperiment object with continuous (2D) EEG data.
`filter`	Logical; apply frequency filtering (default: TRUE).
`lowcut`	Low cutoff frequency in Hz for filtering (default: 0.1).
`highcut`	High cutoff frequency in Hz for filtering (default: 40).

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

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

```
## Not run:
pe <- make_eeg(n_time = 10000, n_channels = 19, sr = 500)
pe <- eegMontage(pe, system = "10-20")
events <- data.frame(onset_sec = c(1, 3, 5, 7, 9))
pe_proc <- eegPreprocess(pe, lowcut = 1, highcut = 40, notch = 50,
                         epoch = TRUE, events = events,
                         artifact_reject = TRUE)

## End(Not run)
```

eegQEEG *Quantitative EEG (QEEG) Analysis*

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

```
## Not run:
pe <- make_eeg(n_time = 5000, n_channels = 19, sr = 500)
pe_qeeg <- eegQEEG(pe)
qeeg_info <- metadata(pe_qeeg)$qeeg
print(qeeg_info$relative_power)

## End(Not run)
```

---

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", "mastoids", "cz", "rest", "channel"),
  ref_channels = NULL,
  exclude = NULL,
  assay_name = NULL,
  output_assay = "rereferenced"
)
```

Arguments

`x`	A PhysioExperiment object.
`ref_type`	Re-referencing scheme: `"average"` (common average), `"mastoids"` (linked mastoids), `"cz"` (Cz reference), `"rest"` (Reference Electrode Standardization Technique), or `"channel"` (user-specified channels).
`ref_channels`	Character vector of channel labels to use as reference (required for `ref_type = "channel"`).
`exclude`	Character vector of channel labels to exclude from the average reference calculation (only used for `ref_type = "average"`).
`assay_name`	Name of the assay to re-reference. If NULL, uses `defaultAssay(x)`.
`output_assay`	Name of the output assay (default: `"rereferenced"`).

Value

A PhysioExperiment object with re-referenced data in the specified output assay. Stores reference info in `metadata(x)$reference`.

Examples

```
## Not run:
pe <- make_eeg(n_time = 5000, n_channels = 19, sr = 500)
pe_avg <- eegRereference(pe, ref_type = "average")
pe_cz  <- eegRereference(pe, ref_type = "cz")

## End(Not run)
```

`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

```
## Not run:
pe <- make_eeg_sleep(n_time = 150000, n_channels = 2, sr = 500)
stages <- eegSleepStage(pe, epoch_sec = 30)
metadata(pe)$sleep_stages <- stages
metrics <- eegSleepMetrics(pe)
print(metrics)

## End(Not run)
```

`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

`x`	A PhysioExperiment object with EEG data.
`method`	Staging method: `"spectral"` (FFT power-based) or `"rule_based"` (synonym, same algorithm).
`epoch_sec`	Epoch duration in seconds (default: 30, AASM standard).
`assay_name`	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

```
## Not run:
pe <- make_eeg_sleep(n_time = 90000, n_channels = 2, sr = 500)
stages <- eegSleepStage(pe, epoch_sec = 30)
head(stages)

## End(Not run)
```

---

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

x	A PhysioExperiment object with EEG data.
method	Analysis method: "tdr" (theta/delta ratio), "dtar" (delta+theta over alpha+beta ratio), or "peak_frequency" (dominant frequency per channel).
bands	Named list of frequency bands for TDR and DTAR methods. Defaults to standard EEG bands.
assay_name	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

```
## Not run:
pe <- make_eeg(n_time = 5000, n_channels = 19, sr = 500)
result <- eegSlowing(pe, method = "dtar")
print(result)

## End(Not run)
```

---

eegSlowWaveDetect *Detect Slow Waves*

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

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

```
## Not run:
pe <- make_eeg_sleep(n_time = 150000, n_channels = 2, sr = 500)
slow_waves <- eegSlowWaveDetect(pe)
head(slow_waves)

## End(Not run)
```

`eegSourceEstimate` *EEG Source Estimation*

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"),
  lambda = 0.05,
  assay_name = NULL,
  output_assay = "source"
)
```

Arguments

`x`	A PhysioExperiment 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), or `"mne"` (minimum norm estimate).
`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"`).

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`, and `n_source_cols`.

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()

Examples

```
## Not run:
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")

## End(Not run)
```

---

eegSourcePower *Compute Source-Space Band Power*

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

```
## Not run:
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)

## End(Not run)
```

---

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 PhysioExperiment 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

```
## Not run:
pe <- make_eeg_spikes(n_time = 30000, n_channels = 19, sr = 500, n_spikes = 15)
spikes <- eegSpikeDetect(pe, method = "morphology")
head(spikes)

## End(Not run)
```

`eegSpindleDetect` *Detect Sleep Spindles*

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

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

```
## Not run:
pe <- make_eeg_sleep(n_time = 150000, n_channels = 2, sr = 500)
spindles <- eegSpindleDetect(pe)
head(spindles)

## End(Not run)
```

---

eegSSVEP *SSVEP Frequency Detection*

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 PhysioExperiment 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

```
## Not run:
pe <- make_eeg_bci(n_trials = 10, n_channels = 8, sr = 256)
result <- eegSSVEP(pe, frequencies = c(10, 12, 15), method = "cca")

## End(Not run)
```

---

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

x	A PhysioExperiment object with EEG data (2D: time x channels).
window_sec	Window length in seconds (default: 0.5).
overlap	Overlap fraction between adjacent windows, from 0 to 1 exclusive (default: 0.75).
window_type	Window function to apply: "hanning", "hamming", or "rectangular" (default: "hanning").
assay_name	Name of the input assay. If NULL, the default assay is used.
output_assay	Name of the assay to store STFT power results (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

```
## Not run:
pe <- make_eeg(n_time = 5000, n_channels = 4, sr = 500)
pe_stft <- eegSTFT(pe, window_sec = 0.5, overlap = 0.75)
sp <- SummarizedExperiment::assay(pe_stft, "stft_power")
dim(sp)  # time_bins x frequencies x channels

## End(Not run)
```

---

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

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

```
## Not run:
pe <- make_eeg(n_time = 10000, n_channels = 4, sr = 500)
result <- eegSuppression(pe, threshold = 10)
print(attr(result, "bsr"))

## End(Not run)
```

`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)
```

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.

Details

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

The debiased wPLI corrects for sample-size bias:

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

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^2 value. Can be slightly negative for very weak connectivity; clamped to 0 in such cases.

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()

Examples

```
## Not run:
pe <- make_eeg(n_time = 5000, n_channels = 4, sr = 500)
wpli_df <- eegWPLI(pe, band = c(8, 13))
head(wpli_df)

## End(Not run)
```

---

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.

See Also

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

Examples

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

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

`n_trials`	Number of trials per class (default: 30). Total trials will be `2 * n_trials`.
`n_channels`	Number of channels (default: 8).
`sr`	Sampling rate in Hz (default: 256).
`trial_sec`	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
```

Index

eegArtifactReject, 3
eegAsymmetry, 4
eegAsymmetry(), *57*, *62*, *67*
eegBadChannels, 6
eegBCIclassify, 7
eegBCIclassify(), *9*, *14*, *42*, *70*, *75*
eegBCIfeatures, *7*, 8
eegBCIfeatures(), *8*, *14*, *42*, *70*, *75*
eegBeamformer, 9
eegBeamformer(), *26*, *65, 66*
eegCoherence, 10
eegCoherence(), *13*, *27*, *53*, *74*
eegConnectivityMatrix, 12
eegConnectivityMatrix(), *12*, *27*, *53*, *74*
eegCSP, 13
eegCSP(), *8, 9*, *42*, *70*, *75*
eegEpoch, 14
eegEpoch(), *16, 17*
eegERPbaseline, 15
eegERPbaseline(), *17*, *20, 21*, *76*
eegERPdetect, 16
eegERPdetect(), *16*, *18*, *20–22*, *76*
eegERPdifference, 17
eegERPdifference(), *18*, *22*
eegERPgrandAverage, 18
eegERPgrandAverage(), *18*, *22*
eegERPlatency, 19
eegERPlatency(), *17*, *21*
eegERPmeasure, 20
eegERPmeasure(), *16–18*, *20*, *22*, *76*
eegERPtest, 21
eegERPtest(), *18*, *76*
eegERSP, 22
eegERSP(), *34*, *41*
eegFilter, 24
eegFilter(), *16*, *28*, *74*
eegForwardModel, *10*, 25, *64*
eegForwardModel(), *10*, *65, 66*
eegGrangerCausality, 26
eegGrangerCausality(), *13*
eegICA, 27
eegICA(), *29–31*
eegICAdetect, 29
eegICAdetect(), *28*, *30, 31*
eegICAmix, 30
eegICAmix(), *28, 29*, *31*
eegICAremove, 31
eegICAremove(), *28–30*
eegInterpolate, *6*, 32
eegITC, 33
eegITC(), *23*
eegKcomplexDetect, 34
eegKcomplexDetect(), *59*, *61*, *63*, *69*, *77*
eegMicrostateBackfit, 35
eegMicrostateBackfit(), *37–39*
eegMicrostates, 36, *39*
eegMicrostates(), *36*, *38, 39*
eegMicrostateSequence, 38
eegMicrostateSequence(), *36, 37*, *39*
eegMicrostateStats, 39
eegMicrostateStats(), *36–38*
eegMontage, *32*, 40
eegMorletWavelet, 40
eegMorletWavelet(), *23*, *34*, *44*, *71*
eegMotorImagery, 42
eegMotorImagery(), *8, 9*, *14*, *75*
eegMultitaper, 43
eegMultitaper(), *41*, *71*
eegPlotConnectivity, 44
eegPlotConnectivity(), *12, 13*, *53*, *74*
eegPlotERP, 45
eegPlotERP(), *17*, *20, 21*
eegPlotHypnogram, 46
eegPlotHypnogram(), *59*, *61*
eegPlotICA, 47
eegPlotSignal, 48
eegPlotSource, 49
eegPlotSource(), *65*

eegPlotSpectrogram, 50
eegPlotSpectrogram(), *23*, *34*, *41*, *44*, *57*, *71*
eegPlotTopomap, 51
eegPlotTopomapSeries, 52
eegPLV, 53
eegPLV(), *12, 13*, *27*, *74*
eegPreprocess, 54
eegPreprocess(), *28*
eegQEEG, 56
eegQEEG(), *5*, *62*, *67*, *72*, *77*
eegQuickStart, 57
eegRereference, 58
eegSleepMetrics, 59
eegSleepMetrics(), *35*, *61*, *63*, *69*, *77*
eegSleepStage, 60
eegSleepStage(), *35*, *59*, *63*, *69*, *77*
eegSlowing, 61
eegSlowing(), *5*, *57*, *67*, *72*, *77*
eegSlowWaveDetect, 62
eegSlowWaveDetect(), *35*, *59*, *61*, *69*
eegSourceEstimate, 64, *65*
eegSourceEstimate(), *10*, *26*, *66*
eegSourcePower, 65
eegSourcePower(), *10*, *26*, *65*
eegSpikeDetect, 66
eegSpikeDetect(), *5*, *57*, *62*, *72*, *77*
eegSpindleDetect, 68
eegSpindleDetect(), *35*, *59*, *61*, *63*, *77*
eegSSVEP, 69
eegSTFT, 70
eegSTFT(), *41*, *44*
eegSuppression, 71
eegSuppression(), *62*
eegWPLI, 73
eegWPLI(), *12, 13*, *27*, *53*

make_eeg, 74
make_eeg(), *57*, *75–77*
make_eeg_bci, 75
make_eeg_bci(), *57*, *74*
make_eeg_erp, 75
make_eeg_erp(), *57*, *74*
make_eeg_sleep, 76
make_eeg_sleep(), *57*, *74*
make_eeg_spikes, 77
make_eeg_spikes(), *74*