

# Package: PhysioEMG (via r-universe)

October 3, 2026

**Title** EMG Analysis Functions for PhysioExperiment Objects

**Version** 0.4.2

**Description** Provides electromyography (EMG) analysis functions for PhysioExperiment objects. Includes envelope extraction (RMS, Hilbert), muscle activation onset detection (Hodges-Bui, Teager-Kaiser), fatigue analysis (median frequency shift), and muscle synergy decomposition (NMF, PCA, ICA).

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

**Imports** SummarizedExperiment, S4Vectors, stats, ggplot2, patchwork

**Suggests** PhysioCore, signal, PhysioECG, PhysioHDEMG, testthat (>= 3.2.0), knitr, rmarkdown

**VignetteBuilder** knitr

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**Roxygen** list(markdown = TRUE)

**Collate** 'utils-provenance.R' 'emg-amplitude.R' 'emg-features.R'  
'emg-data.R' 'emg-onset.R' 'emg-fatigue.R' 'emg-mfcv.R'  
'emg-cocontraction.R' 'emg-decompose.R' 'emg-tensor.R'  
'emg-convolutive.R' 'emg-synergy-mixture.R'  
'emg-hdemg-bridge.R' 'emg-quality.R' 'emg-seniam.R'  
'emg-plot.R' 'emg-network.R' 'emg-adl.R' 'emg-spacetime.R'  
'zzz.R'

**RoxygenNote** 7.3.3

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

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

**Config/pak/sysreqs** zlib1g-dev

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

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

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

**RemoteRef** HEAD

**RemoteSha** 96d6050e46e80e95e072a3730e432d53525b2a7e

## Contents

|                                      |    |
|--------------------------------------|----|
| convolutiveSynergy . . . . .         | 3  |
| coordinationNetwork . . . . .        | 4  |
| emgADLActivation . . . . .           | 5  |
| emgAmplitudeFeatures . . . . .       | 7  |
| emgAmplitudeNormalize . . . . .      | 8  |
| emgCoContraction . . . . .           | 10 |
| emgCoherenceNetwork . . . . .        | 12 |
| emgCoordinationStructure . . . . .   | 13 |
| emgDimitrovIndex . . . . .           | 14 |
| emgDirectedGCNetwork . . . . .       | 15 |
| emgDynamicWaveletNetwork . . . . .   | 16 |
| emgEnvelope . . . . .                | 18 |
| emgFatigue . . . . .                 | 19 |
| emgFatigueIndex . . . . .            | 20 |
| emgFatigueSlope . . . . .            | 21 |
| emgInterpretNetworkKG . . . . .      | 22 |
| emgMFCV . . . . .                    | 23 |
| emgOnsetDetect . . . . .             | 25 |
| emgPartialCoherenceNetwork . . . . . | 27 |
| emgQCgate . . . . .                  | 28 |
| emgQualityCheck . . . . .            | 29 |
| emgRemoveECG . . . . .               | 31 |
| emgSpectralMoments . . . . .         | 32 |
| emgWPLINetwork . . . . .             | 33 |
| getEMGElectrode . . . . .            | 34 |
| hdEMGDecompose . . . . .             | 35 |
| make_emg . . . . .                   | 36 |
| make_emg_contraction . . . . .       | 37 |
| make_emg_fatigue . . . . .           | 38 |
| muscleSynergy . . . . .              | 39 |
| muscleSynergyMixture . . . . .       | 40 |
| muscleSynergyOrder . . . . .         | 42 |
| muscleSynergyTensor . . . . .        | 43 |
| muscleSynergyTensorOrder . . . . .   | 45 |
| physioSynergyMixture . . . . .       | 45 |
| plotCoactivation . . . . .           | 46 |
| plotEmgEnvelope . . . . .            | 48 |
| plotEmgFatigue . . . . .             | 49 |
| plotEmgOnset . . . . .               | 50 |
| plotSynergy . . . . .                | 51 |

*convolutiveSynergy* 3

|                              |    |
|------------------------------|----|
| reconstructTucker . . . . .  | 52 |
| seniamMuscles . . . . .      | 52 |
| setEMGElectrode . . . . .    | 53 |
| spaceByTimeSynergy . . . . . | 54 |
| splitHalfStability . . . . . | 56 |
| synergyCompare . . . . .     | 57 |
| synergyReconstruct . . . . . | 58 |
| tuckerSynergy . . . . .      | 59 |

**Index** 61

---

|                           |                                                    |
|---------------------------|----------------------------------------------------|
| <i>convolutiveSynergy</i> | <i>Time-varying (convolutive) muscle synergies</i> |
|---------------------------|----------------------------------------------------|

---

## Description

Extracts shift-invariant spatiotemporal muscle synergies by convolutive NMF. Each of the `n_synergies` components is a `muscle x L` template that the model can place at any time with any amplitude, so a synergy captures a fixed *spatiotemporal* pattern of muscle activation (d'Avella time-varying synergies), not just a spatial weighting.

## Usage

```
convolutiveSynergy(  
    emg,  
    n_synergies,  
    L,  
    max_iter = 300,  
    tol = 1e-06,  
    restarts = 3,  
    seed = NULL  
)
```

## Arguments

|                          |                                                                    |
|--------------------------|--------------------------------------------------------------------|
| <code>emg</code>         | A non-negative <code>muscle x time</code> matrix (rows = muscles). |
| <code>n_synergies</code> | Number of synergies.                                               |
| <code>L</code>           | Temporal length of each synergy (samples).                         |
| <code>max_iter</code>    | Maximum multiplicative-update iterations (default 300).            |
| <code>tol</code>         | Relative-error convergence tolerance (default 1e-6).               |
| <code>restarts</code>    | Random restarts; the best is kept (default 3).                     |
| <code>seed</code>        | Optional integer seed.                                             |

## Value

An object of class `"convolutive_synergy"`: a list with synergies (a `muscle x synergy x L` array), activations (`synergy x time`), `vaf`, `iterations`, `n_synergies`, and `L`.

## References

d'Avella A, Saltiel P, Bizzi E (2003). Combinations of muscle synergies in the construction of a natural motor behavior. *Nat Neurosci* 6:300-308. Smaragdis P (2004). Non-negative matrix factor deconvolution. *ICA*.

## Examples

```
set.seed(1)
M <- 5; Tt <- 200; N <- 2; L <- 15
W0 <- array(0, c(M, N, L))
for (nn in 1:N) W0[, nn, ] <- outer(runif(M), dnorm(1:L, L / 2, 3))
H0 <- matrix(0, N, Tt); H0[1, c(30, 120)] <- 1; H0[2, c(70, 160)] <- 1
V <- matrix(0, M, Tt)
for (tau in 0:(L - 1)) V <- V +
  W0[, , tau + 1] %*% cbind(matrix(0, N, tau), H0[, 1:(Tt - tau), drop = FALSE])
fit <- convolutiveSynergy(V, n_synergies = 2, L = 15, seed = 1)
fit$vaf
```

---

|                     |                                                                           |
|---------------------|---------------------------------------------------------------------------|
| coordinationNetwork | <i>Build an inter-muscle coordination network with a chosen estimator</i> |
|---------------------|---------------------------------------------------------------------------|

---

## Description

A single entry point over the four inter-muscle connectivity estimators ([emgCoherenceNetwork\(\)](#), [emgPartialCoherenceNetwork\(\)](#), [emgWPLINetwork\(\)](#), [emgDirectedGCNetwork\(\)](#)): builds the coordination network of a multi-muscle EMG recording with the chosen method and routes the relevant parameters to the underlying estimator. Use it (with [splitHalfStability\(\)](#)) to answer "which muscles co-activate, and is that structure real?".

## Usage

```
coordinationNetwork(
  x,
  method = c("coherence", "partial_coherence", "wpli", "directed_gc"),
  freq_band = NULL,
  channels = NULL,
  nperseg = 256L,
  noverlap = NULL,
  max_lag = 10L,
  debiased = FALSE,
  assay_name = NULL,
  ...
)
```

**Arguments**

|                                |                                                                                                                                                                                                        |
|--------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>                 | A <code>PhysioExperiment</code> holding a multi-channel EMG assay.                                                                                                                                     |
| <code>method</code>            | Estimator: "coherence" (magnitude-squared coherence), "partial_coherence" (coherence with shared influence removed), "wpli" (weighted phase-lag index), or "directed_gc" (directed Granger causality). |
| <code>freq_band</code>         | Optional <code>c(low, high)</code> Hz band for the spectral estimators.                                                                                                                                |
| <code>channels</code>          | Optional integer channel indices to include (default all).                                                                                                                                             |
| <code>nperseg, noverlap</code> | Welch segment length / overlap for the spectral estimators (coherence / partial coherence / wPLI).                                                                                                     |
| <code>max_lag</code>           | Model order for directed Granger causality.                                                                                                                                                            |
| <code>debiased</code>          | Use the debiased wPLI estimator.                                                                                                                                                                       |
| <code>assay_name</code>        | Assay to build the network from (default: the default assay).                                                                                                                                          |
| <code>...</code>               | Passed through to the underlying estimator.                                                                                                                                                            |

**Value**

The estimator's network object – a list whose `$network` is the inter-muscle adjacency matrix.

**See Also**

[splitHalfStability\(\)](#), [emgCoordinationStructure\(\)](#), [emgCoherenceNetwork\(\)](#)

**Examples**

```
pe <- make_emg(n_time = 2000, n_channels = 4, sr = 1000)
net <- coordinationNetwork(pe, method = "coherence", freq_band = c(20, 150))
net$network
```

---

emgADLActivation

---

*EMG muscle-activation summary of an ADL task*


---

**Description**

Summarises how an agonist-antagonist muscle pair drives one ADL task from an EMG envelope: the co-contraction index (via [emgCoContraction\(\)](#)), each muscle's peak and mean activation (as percent of a maximal reference when `mvc_data` is given, else raw envelope units), and the fraction of the task each muscle is active. Higher antagonist co-contraction means less efficient, more guarded movement.

**Usage**

```
emgADLActivation(
  x,
  agonist,
  antagonist,
  mvc_data = NULL,
  task = c("reaching", "drinking", "feeding", "dressing", "grooming"),
  cc_method = c("falconer_winter", "rudolph", "frost"),
  active_frac = 0.2,
  assay_name = NULL
)
```

**Arguments**

|                                  |                                                                                                                             |
|----------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>                   | A <code>PhysioExperiment</code> of EMG envelope data (rectified / smoothed).                                                |
| <code>agonist, antagonist</code> | Channel index or <code>colData\$label</code> of the pair.                                                                   |
| <code>mvc_data</code>            | Optional <code>PhysioExperiment</code> of maximal-effort (MVC) data with the same channels; enables percent-MVC amplitudes. |
| <code>task</code>                | ADL task realised: "reaching" (d445), "drinking" (d560), "feeding" (d550), "dressing" (d540) or "grooming" (d520).          |
| <code>cc_method</code>           | Co-contraction index method (see <a href="#">emgCoContraction()</a> ).                                                      |
| <code>active_frac</code>         | Activation threshold as a fraction of a muscle's own peak (default 0.2) for the active-time summary.                        |
| <code>assay_name</code>          | EMG assay to use (default: the first assay).                                                                                |

**Value**

an `emg_adl_activation` list: `task`, `icf_code`, `cocontraction_index`, `unit` ("%MVC" or "envelope"), per-muscle `agonist_peak/agonist_mean/antagonist_peak/antagonist_mean` and `agonist_active_frac/antagonist_active_frac`.

**See Also**

[emgCoContraction\(\)](#), [emgAmplitudeNormalize\(\)](#)

**Examples**

```
set.seed(1)
n <- 1000; t <- seq_len(n)
ag <- exp(-((t - 500) / 150)^2) + rnorm(n, 0, 0.02) # agonist burst
an <- 0.2 * exp(-((t - 500) / 150)^2) + rnorm(n, 0, 0.02) # low antagonist
pe <- PhysioExperiment::PhysioExperiment(
  assays = S4Vectors::SimpleList(envelope = cbind(BIC = ag, TRI = an)),
  colData = S4Vectors::DataFrame(label = c("BIC", "TRI"), type = "EMG"),
  samplingRate = 1000)
emgADLActivation(pe, "BIC", "TRI", task = "drinking")$cocontraction_index
```

---

emgAmplitudeFeatures    *EMG Amplitude Features (ARV, MAV, iEMG, RMS)*


---

### Description

Computes time-domain amplitude features of an EMG signal over a sliding window. The signal is divided into overlapping windows and, for each window and channel, the requested features are computed:

arv Average rectified value,  $\frac{1}{N} \sum |x_i|$ .

mav Mean absolute value,  $\frac{1}{N} \sum |x_i|$  (identical to ARV for an unweighted window).

iemg Integrated EMG,  $\sum |x_i| \Delta t$  with  $\Delta t = 1/f_s$ ; the time integral of the rectified signal, in signal-units times seconds.

rms Root mean square,  $\sqrt{\frac{1}{N} \sum x_i^2}$ .

The window geometry (window and step sizes in samples) matches `emgFatigue()` and `emgSpectralMoments()` for identical `window_sec` and `overlap`, so amplitude and spectral features align window-for-window.

### Usage

```
emgAmplitudeFeatures(
  x,
  features = c("arv", "mav", "iemg", "rms"),
  window_sec = 1,
  overlap = 0.5,
  assay_name = NULL
)
```

### Arguments

|                         |                                                                                                              |
|-------------------------|--------------------------------------------------------------------------------------------------------------|
| <code>x</code>          | A <code>PhysioExperiment</code> object with EMG data.                                                        |
| <code>features</code>   | Character vector selecting which features to return; any of "arv", "mav", "iemg", "rms" (default: all four). |
| <code>window_sec</code> | Analysis window in seconds (default: 1.0).                                                                   |
| <code>overlap</code>    | Overlap fraction between consecutive windows, in $[0, 1]$ (default: 0.5).                                    |
| <code>assay_name</code> | Input assay name (default: first assay).                                                                     |

### Value

A `data.frame` with one row per analysis window per channel, with columns:

**channel** Integer channel index.

**window** Integer window index within the channel (1-based).

**time\_sec** Window start time in seconds.

**arv, mav, iemg, rms** The requested amplitude features (only the columns named in `features` are present).

If no full window fits in the signal, a 0-row `data.frame` with these columns is returned.

## References

- Merletti, R. & Parker, P.A. (2004). "Electromyography: Physiology, Engineering, and Non-Invasive Applications." Wiley-IEEE Press. doi:10.1002/0471678384
- Hermens, H.J. et al. (2000). "Development of recommendations for SEMG sensors and sensor placement procedures (SENIAM)." Journal of Electromyography and Kinesiology, 10(5), 361-374. doi:10.1016/S1050-6411(00)00027-4

## See Also

[emgEnvelope\(\)](#) for a per-sample amplitude envelope, [emgFatigue\(\)](#) for windowed spectral fatigue features, [emgSpectralMoments\(\)](#) for spectral moments on the same window grid

## Examples

```
# 2 s of two-channel EMG at 1000 Hz
set.seed(1)
m <- matrix(rnorm(2000 * 2, sd = 0.3), nrow = 2000, ncol = 2)
pe <- PhysioExperiment(
  assays = list(raw = m),
  colData = S4Vectors::DataFrame(label = c("EMG1", "EMG2"),
                                   type = c("EMG", "EMG")),
  samplingRate = 1000)
feats <- emgAmplitudeFeatures(pe, window_sec = 0.5, overlap = 0.5)
head(feats)
```

---

emgAmplitudeNormalize *Normalize EMG Amplitude*

---

## Description

Normalizes EMG amplitude data by a per-channel reference value. "mvc" divides by the per-channel maximum of a maximum-voluntary-contraction trial (percentage-of-MVC). "peak" divides by the within-trial peak so each channel ranges from 0 to 1. "rvc" divides by the per-channel mean amplitude of a sub-maximal reference voluntary contraction trial (percentage-of-RVC). "dynamic\_peak" divides by a centered moving maximum, tracking a time-varying peak within the trial.

## Usage

```
emgAmplitudeNormalize(
  x,
  method = c("mvc", "peak", "rvc", "dynamic_peak"),
  mvc_data = NULL,
  rvc_data = NULL,
  rvc_window = NULL,
  assay_name = NULL,
  output_assay = "normalized"
)
```

**Arguments**

|              |                                                                                                                                                                                                                                                                                                               |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x            | A <code>PhysioExperiment</code> object (amplitude data).                                                                                                                                                                                                                                                      |
| method       | Normalization method: "mvc" (maximum voluntary contraction), "peak" (within-trial peak), "rvc" (sub-maximal reference voluntary contraction), or "dynamic_peak" (centered moving maximum).                                                                                                                    |
| mvc_data     | A <code>PhysioExperiment</code> containing MVC trial data (required for "mvc"). Must have the same number of channels as x.                                                                                                                                                                                   |
| rvc_data     | A <code>PhysioExperiment</code> containing the sub-maximal reference-task data (required for "rvc"). Must have the same number of channels as x.                                                                                                                                                              |
| rvc_window   | Optional window. For "rvc", a length-2 numeric <code>c(start, end)</code> in seconds selecting the portion of <code>rvc_data</code> over which the mean reference amplitude is computed (default: the whole reference trial). For "dynamic_peak", the moving-maximum window length in seconds (default: 0.5). |
| assay_name   | Assay to normalize (default: first assay).                                                                                                                                                                                                                                                                    |
| output_assay | Output assay name (default: "normalized").                                                                                                                                                                                                                                                                    |

**Details**

Input is assumed to be amplitude data (a rectified signal or an `emgEnvelope()` output). MVC and RVC normalization are scale-invariant to a gain applied to both the signal and its reference trial.

**Value**

A `PhysioExperiment` object with an additional assay named `output_assay` containing normalized amplitude values. For "peak" and "dynamic\_peak", values are scaled to a peak. For "mvc"/"rvc", values are proportions of the reference (multiply by 100 for percentage-of-MVC / percentage-of-RVC).

**References**

- De Luca, C.J. (1997). "The use of surface electromyography in biomechanics." *Journal of Applied Biomechanics*, 13(2), 135-163. doi:10.1123/jab.13.2.135
- Yang, J.F. & Winter, D.A. (1984). "Electromyographic amplitude normalization methods: improving their sensitivity as diagnostic tools in gait analysis." *Archives of Physical Medicine and Rehabilitation*, 65(9), 517-521.
- Burden, A. & Bartlett, R. (1999). "Normalisation of EMG amplitude: an evaluation and comparison of old and new methods." *Medical Engineering & Physics*, 21(4), 247-257. doi:10.1016/S1350-4533(99)00054-5

**See Also**

`emgEnvelope()` for computing amplitude envelopes prior to normalization, `emgAmplitudeFeatures()` for windowed amplitude features, `emgFatigue()` for fatigue analysis, `emgOnsetDetect()` for muscle activation onset detection

## Examples

```
set.seed(1)
amp <- matrix(abs(rnorm(1000 * 2, sd = 0.3)), nrow = 1000, ncol = 2)
pe <- PhysioExperiment(
  assays = list(raw = amp),
  colData = S4Vectors::DataFrame(label = c("EMG1", "EMG2"),
                                   type = c("EMG", "EMG")),
  samplingRate = 1000)
# percentage-of-RVC using a sub-maximal reference (here, the trial itself)
pe_rvc <- emgAmplitudeNormalize(pe, method = "rvc", rvc_data = pe)
colMeans(SummarizedExperiment::assay(pe_rvc, "normalized")) # ~ 1
```

---

|                  |                                 |
|------------------|---------------------------------|
| emgCoContraction | <i>EMG Co-Contraction Index</i> |
|------------------|---------------------------------|

---

## Description

Quantifies simultaneous activation of an agonist-antagonist muscle pair from their amplitude envelopes, using one of three established indices. Input channels are treated as (non-negative) amplitude envelopes; with `normalize = TRUE` each is first scaled to its own peak so the two muscles are amplitude-comparable.

## Usage

```
emgCoContraction(
  x,
  agonist,
  antagonist,
  method = c("falconer_winter", "rudolph", "frost"),
  normalize = TRUE,
  window_sec = NULL,
  assay_name = NULL
)
```

## Arguments

|                                  |                                                                                                                                                                         |
|----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>                   | A <code>PhysioExperiment</code> object of EMG amplitude envelopes.                                                                                                      |
| <code>agonist, antagonist</code> | Channel index (integer) or channel label (character) of the two muscles. The indices are interchangeable for all three methods (the indices are symmetric in the pair). |
| <code>method</code>              | Co-contraction index: "falconer_winter" (default), "rudolph", or "frost".                                                                                               |
| <code>normalize</code>           | If <code>TRUE</code> (default), peak-normalize each channel to [0, 1] before computing the index.                                                                       |
| <code>window_sec</code>          | Optional non-overlapping window length in seconds for a time-resolved index. If <code>NULL</code> (default) only the whole-signal summary is returned.                  |
| <code>assay_name</code>          | Input assay name (default: first assay).                                                                                                                                |

## Details

"falconer\_winter" Falconer & Winter (1985) percent co-contraction:  $2 \sum \min(a, b) / \sum(a + b) \times 100$ . 100 when the two envelopes are identical, 0 when they never overlap.

"rudolph" Rudolph et al. (2000) co-contraction index: the mean over time of  $(\ell/h)(\ell + h)$  with  $\ell = \min(a, b)$ ,  $h = \max(a, b)$ . Symmetric in the two muscles and non-negative.

"frost" Frost et al. (1997) variant: the time-average of the instantaneous co-activation ratio  $2 \min(a, b) / (a + b)$ , as a percentage.

## Value

A list with:

**method** The method used.

**summary** The whole-signal co-contraction index (a scalar).

**timeseries** If window\_sec is given, a data.frame with columns window, time\_sec and cci (one row per window); otherwise NULL.

## References

Falconer, K. & Winter, D.A. (1985). "Quantitative assessment of co-contraction at the ankle joint in walking." *Electromyography and Clinical Neurophysiology*, 25(2-3), 135-149.

Rudolph, K.S., Axe, M.J. & Snyder-Mackler, L. (2000). "Dynamic stability after ACL injury: who can hop?" *Knee Surgery, Sports Traumatology, Arthroscopy*, 8(5), 262-269. doi:10.1007/s001670000130

Frost, G., Dowling, J., Dyson, K. & Bar-Or, O. (1997). "Cocontraction in three age groups of children during treadmill locomotion." *Journal of Electromyography and Kinesiology*, 7(3), 179-186. doi:10.1016/S1050-6411(97)84626-3

## See Also

[emgEnvelope\(\)](#) for computing amplitude envelopes, [emgAmplitudeNormalize\(\)](#) for reference normalization, [emgOnsetDetect\(\)](#) for activation timing

## Examples

```
set.seed(1)
t <- seq_len(1000)
a <- exp(-((t - 400) / 120)^2)      # agonist envelope
b <- exp(-((t - 600) / 120)^2)      # antagonist envelope
pe <- PhysioExperiment(
  assays = list(env = cbind(a, b)),
  colData = S4Vectors::DataFrame(label = c("TA", "GAS"),
                                   type = c("EMG", "EMG")),
  samplingRate = 1000)
emgCoContraction(pe, agonist = "TA", antagonist = "GAS")$summary
```

---

emgCoherenceNetwork      *EMG Coherence Network Analysis*


---

## Description

Builds a static muscle coordination network from pairwise magnitude-squared coherence between EMG channels.

## Usage

```
emgCoherenceNetwork(
  x,
  freq_band = NULL,
  channels = NULL,
  nperseg = 256L,
  noverlap = NULL,
  assay_name = NULL,
  aggregate = c("mean", "max", "median"),
  threshold = NULL
)
```

## Arguments

|                         |                                                                                                       |
|-------------------------|-------------------------------------------------------------------------------------------------------|
| <code>x</code>          | A <code>PhysioExperiment</code> object.                                                               |
| <code>freq_band</code>  | Optional numeric vector <code>c(low, high)</code> in Hz. If <code>NULL</code> , uses all frequencies. |
| <code>channels</code>   | Integer vector of channel indices to include. If <code>NULL</code> , uses all.                        |
| <code>nperseg</code>    | Segment length for Welch estimation (default: 256).                                                   |
| <code>noverlap</code>   | Overlap length (default: <code>floor(nperseg / 2)</code> ).                                           |
| <code>assay_name</code> | Input assay name. If <code>NULL</code> , uses default assay.                                          |
| <code>aggregate</code>  | Aggregation across frequency bins: "mean", "max", or "median".                                        |
| <code>threshold</code>  | Optional threshold for binary adjacency matrix.                                                       |

## Value

A list with:

**network** Numeric matrix (channel x channel) of coherence strength.

**adjacency** Logical matrix after thresholding, or `NULL`.

**coherence** 3D array (freq x channel x channel).

**frequencies** Frequency vector (Hz) corresponding to coherence.

**channel\_names** Channel labels used in the network.

## See Also

[emgDynamicWaveletNetwork\(\)](#) for time-varying networks, [emgInterpretNetworkKG\(\)](#) for annotation-aware interpretation.

## Examples

```
pe <- make_emg(n_time = 2000, n_channels = 4, sr = 1000)
res <- emgCoherenceNetwork(pe, freq_band = c(20, 150))
res$network
```

---

emgCoordinationStructure

*EMG Coordination Structure Summary from Network Topology*

---

## Description

Quantifies higher-order muscle coordination structure from a weighted network matrix, including module structure, efficiency, and node roles.

## Usage

```
emgCoordinationStructure(
  network,
  threshold = NULL,
  n_modules = NULL,
  max_modules = 6L,
  directed = FALSE,
  symmetrize = c("mean", "max", "min"),
  normalize = TRUE
)
```

## Arguments

|             |                                                                                                                       |
|-------------|-----------------------------------------------------------------------------------------------------------------------|
| network     | Numeric square matrix (channels x channels) or a list containing \$network.                                           |
| threshold   | Optional edge-weight threshold. Values below threshold are set to zero before topology estimation.                    |
| n_modules   | Optional number of modules. If NULL, chooses a value automatically by maximizing weighted modularity over candidates. |
| max_modules | Maximum number of candidate modules for automatic search.                                                             |
| directed    | Logical; set TRUE if network is directed/asymmetric.                                                                  |
| symmetrize  | Method to convert directed matrices to undirected form: "mean", "max", or "min".                                      |
| normalize   | Logical; if TRUE, rescales weights to $[0, 1]$ .                                                                      |

## Value

A list with:

**network** Processed undirected weighted network matrix.  
**node\_metrics** Data.frame of node-level topology features.  
**modules** Named integer vector of module assignments.  
**summary** Global network topology summary.

**See Also**

[emgCoherenceNetwork\(\)](#), [emgDynamicWaveletNetwork\(\)](#)

**Examples**

```
pe <- make_emg(n_time = 2000, n_channels = 4, sr = 1000)
net <- emgCoherenceNetwork(pe, freq_band = c(20, 150))
cs <- emgCoordinationStructure(net)
cs$summary
```

---

emgDimitrovIndex

---

*EMG Dimitrov Spectral Fatigue Index (FInsm5)*


---

**Description**

Computes spectral moments  $M(k) = \int f^k PSD(f) df$  for  $k = -1, 0, \dots, 5$  over sliding windows and the Dimitrov fatigue index  $FInsm5 = M(-1)/M(5)$  per window. FInsm5 rises steeply with fatigue because spectral compression toward low frequencies simultaneously increases the low-frequency moment  $M(-1)$  and decreases the high-frequency moment  $M(5)$ . The per-channel FInsm5-vs-time regression slope is attached as the "finsm5\_slope" attribute.

**Usage**

```
emgDimitrovIndex(x, window_sec = 1, overlap = 0.5, assay_name = NULL)
```

**Arguments**

|            |                                                  |
|------------|--------------------------------------------------|
| x          | A PhysioExperiment object with EMG data.         |
| window_sec | Analysis window in seconds (default: 1.0).       |
| overlap    | Overlap fraction between windows (default: 0.5). |
| assay_name | Input assay name (default: first assay).         |

**Value**

A data.frame with one row per channel per window, containing columns channel, window, time\_sec, the moments m\_minus1, m0, m1, m2, m3, m4, m5, and finsm5. The per-channel FInsm5 regression slope (index per minute) is available via attr(result, "finsm5\_slope").

**References**

Dimitrov, G.V. et al. (2006). "Muscle fatigue during dynamic contractions assessed by new spectral indices." *Medicine & Science in Sports & Exercise*, 38(11), 1971-1979. doi:10.1249/01.mss.0000233794.31659.6d

**See Also**

[emgFatigueSlope\(\)](#) for MDF/MNF regression slopes, [emgSpectralMoments\(\)](#) for the M0/M1/M2 moments, [emgFatigue\(\)](#) for median/mean frequency tracking

**Examples**

```

sr <- 1000; n <- 6000
sig <- sin(2 * pi * 80 * seq_len(n) / sr) + rnorm(n, sd = 0.1)
pe <- PhysioExperiment(assays = list(raw = matrix(sig, ncol = 1)),
                        samplingRate = sr)
di <- emgDimitrovIndex(pe, window_sec = 1)
head(di)
attr(di, "finsm5_slope")

```

---

emgDirectedGCNetwork    *Directed EMG Network via Pairwise Granger Causality*

---

**Description**

Estimates a directed muscle coordination network using pairwise Granger causality in the time domain.

**Usage**

```

emgDirectedGCNetwork(
  x,
  channels = NULL,
  assay_name = NULL,
  max_lag = 10L,
  score = c("f_stat", "delta_r2"),
  threshold = NULL,
  p_value_cutoff = NULL,
  standardize = TRUE
)

```

**Arguments**

|                |                                                                  |
|----------------|------------------------------------------------------------------|
| x              | A PhysioExperiment object.                                       |
| channels       | Integer vector of channel indices to include. If NULL, uses all. |
| assay_name     | Input assay name. If NULL, uses default assay.                   |
| max_lag        | Lag order (in samples) for autoregressive modeling.              |
| score          | Directed edge metric: "f_stat" or "delta_r2".                    |
| threshold      | Optional threshold for adjacency based on selected score.        |
| p_value_cutoff | Optional p-value threshold for adjacency.                        |
| standardize    | Logical; if TRUE, z-score each channel before GC.                |

**Value**

A list with:

**network** Directed numeric matrix (source x target).

**p\_values** Directed matrix of GC p-values.

**adjacency** Logical directed matrix, or NULL.

**channel\_names** Channel labels used in the network.

**lag** Lag order used for modeling.

**Examples**

```
pe <- make_emg(n_time = 2000, n_channels = 4, sr = 1000)
res <- emgDirectedGCNetwork(pe, max_lag = 5)
res$network
```

---

emgDynamicWaveletNetwork

*Dynamic Wavelet Coherence Network for EMG*

---

**Description**

Builds a time-varying coordination network by computing wavelet coherence between channel pairs and aggregating coherence within sliding windows.

**Usage**

```
emgDynamicWaveletNetwork(
  x,
  frequencies = seq(5, 120, by = 5),
  freq_band = NULL,
  channels = NULL,
  window_sec = 0.5,
  step_sec = 0.1,
  n_cycles = 7,
  smoothing_cycles = 3,
  assay_name = NULL,
  aggregate = c("mean", "max", "median"),
  threshold = NULL,
  respect_coi = TRUE
)
```

**Arguments**

|                               |                                                                                 |
|-------------------------------|---------------------------------------------------------------------------------|
| <code>x</code>                | A <code>PhysioExperiment</code> object.                                         |
| <code>frequencies</code>      | Numeric vector of wavelet center frequencies (Hz).                              |
| <code>freq_band</code>        | Optional numeric vector <code>c(low, high)</code> for aggregation.              |
| <code>channels</code>         | Integer vector of channel indices to include. If <code>NULL</code> , uses all.  |
| <code>window_sec</code>       | Sliding window length in seconds.                                               |
| <code>step_sec</code>         | Sliding window step in seconds.                                                 |
| <code>n_cycles</code>         | Number of Morlet cycles (default: 7).                                           |
| <code>smoothing_cycles</code> | Smoothing width in cycles (default: 3).                                         |
| <code>assay_name</code>       | Input assay name. If <code>NULL</code> , uses default assay.                    |
| <code>aggregate</code>        | Aggregation across time-frequency bins: "mean", "max", or "median".             |
| <code>threshold</code>        | Optional threshold for binary adjacency network per window.                     |
| <code>respect_coi</code>      | Logical; if <code>TRUE</code> , masks frequencies below COI before aggregation. |

**Value**

A list with:

- network** 3D array (window x channel x channel).
- adjacency** Logical 3D array thresholded from network, or `NULL`.
- window\_times** Window center times in seconds.
- static\_summary** Mean network across windows.
- frequencies** Frequency vector used for wavelet transform.
- coi** Cone-of-influence frequency at each time sample.

**See Also**

[emgCoherenceNetwork\(\)](#) for static spectral network, [emgInterpretNetworkKG\(\)](#) for annotation-aware interpretation.

**Examples**

```
pe <- make_emg(n_time = 1000, n_channels = 3, sr = 1000)
res <- emgDynamicWaveletNetwork(pe, frequencies = seq(20, 100, by = 20),
                               window_sec = 0.25, step_sec = 0.1)
dim(res$network) # windows x channels x channels
```

emgEnvelope

*EMG Envelope Extraction***Description**

Extracts the amplitude envelope from EMG signals using various methods. The RMS method computes root mean square over a sliding window. The Hilbert method uses the analytic signal via the Hilbert transform. The lowpass method rectifies the signal and applies a moving-average lowpass filter.

**Usage**

```
emgEnvelope(
  x,
  method = c("rms", "hilbert", "lowpass"),
  window_ms = 50,
  cutoff = 6,
  assay_name = NULL,
  output_assay = "envelope"
)
```

**Arguments**

|                           |                                                                                                                          |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>            | A <code>PhysioExperiment</code> object with EMG data.                                                                    |
| <code>method</code>       | Envelope method: "rms" (root mean square), "hilbert" (Hilbert transform), or "lowpass" (rectification + lowpass filter). |
| <code>window_ms</code>    | Window size in milliseconds for RMS method (default: 50).                                                                |
| <code>cutoff</code>       | Cutoff frequency in Hz for lowpass method (default: 6).                                                                  |
| <code>assay_name</code>   | Input assay name (default: first assay).                                                                                 |
| <code>output_assay</code> | Output assay name (default: "envelope").                                                                                 |

**Value**

A `PhysioExperiment` object with an additional assay named `output_assay` containing the amplitude envelope. The envelope matrix has the same dimensions as the input (time x channels) with non-negative values representing instantaneous signal amplitude.

**References**

De Luca, C.J. (1997). "The use of surface electromyography in biomechanics." *Journal of Applied Biomechanics*, 13(2), 135-163. doi:10.1123/jab.13.2.135

Merletti, R. & Parker, P.A. (2004). "Electromyography: Physiology, Engineering, and Non-Invasive Applications." Wiley-IEEE Press. doi:10.1002/0471678384

**See Also**

[emgAmplitudeNormalize\(\)](#) for normalizing envelope values, [emgOnsetDetect\(\)](#) for onset detection from envelope data, [emgFatigue\(\)](#) for fatigue analysis using spectral features

**Examples**

```
pe <- make_emg(n_time = 2000, n_channels = 4, sr = 1000)
pe <- emgEnvelope(pe, method = "rms", window_ms = 50)
dim(SummarizedExperiment::assay(pe, "envelope"))
```

emgFatigue

*EMG Fatigue Analysis***Description**

Tracks median and mean frequency over time to assess muscle fatigue. Decreasing median frequency indicates fatigue due to reduced motor unit conduction velocity. The signal is divided into overlapping windows and the power spectral density is computed via FFT for each window.

**Usage**

```
emgFatigue(x, window_sec = 1, overlap = 0.5, assay_name = NULL)
```

**Arguments**

|                   |                                                       |
|-------------------|-------------------------------------------------------|
| <b>x</b>          | A <code>PhysioExperiment</code> object with EMG data. |
| <b>window_sec</b> | Analysis window in seconds (default: 1.0).            |
| <b>overlap</b>    | Overlap fraction between windows (default: 0.5).      |
| <b>assay_name</b> | Input assay name (default: first assay).              |

**Value**

A data.frame with one row per channel per window, containing columns:

**channel** Integer channel index.  
**window** Integer window number (1-indexed).  
**time\_sec** Start time of the window in seconds.  
**median\_freq** Median frequency (Hz) at which 50 percent of the spectral power is below.  
**mean\_freq** Power-weighted mean frequency (Hz).  
**rms\_amplitude** Root mean square amplitude of the window.

**References**

De Luca, C.J. (1984). "Myoelectrical manifestations of localized muscular fatigue in humans." *Critical Reviews in Biomedical Engineering*, 11(4), 251-279.

Merletti, R. & Parker, P.A. (2004). "Electromyography: Physiology, Engineering, and Non-Invasive Applications." Wiley-IEEE Press. doi:10.1002/0471678384

**See Also**

[emgFatigueIndex\(\)](#) for a summary fatigue metric, [emgSpectralMoments\(\)](#) for spectral moment analysis, [emgEnvelope\(\)](#) for amplitude envelope extraction

**Examples**

```
pe <- make_emg_fatigue(n_time = 10000, sr = 1000)
fat <- emgFatigue(pe, window_sec = 1.0)
head(fat)
```

emgFatigueIndex

*EMG Fatigue Index***Description**

Computes a fatigue index as the ratio of final to initial median frequency. Values less than 1 indicate fatigue (frequency decrease). The signal is internally analyzed with [emgFatigue\(\)](#) using 0.5-second windows at 50 percent overlap, then the initial and final portions are compared.

**Usage**

```
emgFatigueIndex(x, initial_pct = 0.2, final_pct = 0.2, assay_name = NULL)
```

**Arguments**

|                    |                                                                |
|--------------------|----------------------------------------------------------------|
| <b>x</b>           | A <code>PhysioExperiment</code> object with EMG data.          |
| <b>initial_pct</b> | Percentage of signal used for initial estimate (default: 0.2). |
| <b>final_pct</b>   | Percentage of signal used for final estimate (default: 0.2).   |
| <b>assay_name</b>  | Input assay name (default: first assay).                       |

**Value**

A `data.frame` with one row per channel, containing columns:

**channel** Integer channel index.

**fatigue\_index** Ratio of final to initial median frequency. Values less than 1 indicate fatigue.

**initial\_mdf** Mean median frequency (Hz) in the initial portion.

**final\_mdf** Mean median frequency (Hz) in the final portion.

**References**

De Luca, C.J. (1984). "Myoelectrical manifestations of localized muscular fatigue in humans." *Critical Reviews in Biomedical Engineering*, 11(4), 251-279.

Merletti, R. & Parker, P.A. (2004). "Electromyography: Physiology, Engineering, and Non-Invasive Applications." Wiley-IEEE Press. doi:10.1002/0471678384

See Also

[emgFatigue\(\)](#) for detailed windowed fatigue tracking, [emgSpectralMoments\(\)](#) for spectral moment analysis, [emgEnvelope\(\)](#) for amplitude envelope extraction

Examples

```
pe <- make_emg_fatigue(n_time = 10000, sr = 1000)
emgFatigueIndex(pe)
```

---

|                 |                                     |
|-----------------|-------------------------------------|
| emgFatigueSlope | <i>EMG Fatigue Regression Slope</i> |
|-----------------|-------------------------------------|

---

Description

Fits a linear regression of a spectral fatigue metric (median or mean frequency) against time across sliding windows and reports the fatigue slope per channel. A negative slope indicates myoelectric fatigue (spectral compression toward lower frequencies).

Usage

```
emgFatigueSlope(
  x,
  feature = c("mdf", "mnf"),
  normalize = TRUE,
  window_sec = 1,
  overlap = 0.5,
  assay_name = NULL
)
```

Arguments

|            |                                                                                             |
|------------|---------------------------------------------------------------------------------------------|
| x          | A <code>PhysioExperiment</code> object with EMG data.                                       |
| feature    | Metric to regress: "mdf" (median frequency) or "mnf" (mean frequency).                      |
| normalize  | If TRUE, also report the slope as a percentage of the initial (intercept) value per minute. |
| window_sec | Analysis window in seconds (default: 1.0).                                                  |
| overlap    | Overlap fraction between windows (default: 0.5).                                            |
| assay_name | Input assay name (default: first assay).                                                    |

Value

A `data.frame` with one row per channel, containing columns:

- channel** Integer channel index.
- slope\_hz\_per\_min** Regression slope in Hz per minute (negative under fatigue).

**norm\_slope\_pct\_per\_min** Slope as a percentage of the initial value per minute (NA if normalize = FALSE).

**intercept\_hz** Fitted value at time zero (initial frequency), Hz.

**r\_squared** Coefficient of determination of the fit.

**p\_value** Two-sided p-value for the slope.

## References

Merletti, R. & Parker, P.A. (2004). "Electromyography: Physiology, Engineering, and Non-Invasive Applications." Wiley-IEEE Press. doi:10.1002/0471678384

## See Also

[emgFatigue\(\)](#) for the underlying windowed frequency estimates, [emgDimitrovIndex\(\)](#) for the spectral-moment fatigue index, [emgFatigueIndex\(\)](#) for the initial/final ratio metric

## Examples

```
# decreasing-frequency (fatiguing) signal
sr <- 1000; n <- 8000; t <- seq_len(n) / sr
f <- 90 - 3 * t # 90 Hz falling to ~66 Hz
sig <- sin(2 * pi * cumsum(f) / sr) + rnorm(n, sd = 0.1)
pe <- PhysioExperiment(assays = list(raw = matrix(sig, ncol = 1)),
  samplingRate = sr)
emgFatigueSlope(pe, feature = "mdf")
```

---

emgInterpretNetworkKG *Interpret EMG Network with Knowledge-Graph Metadata*

---

## Description

Adds metadata/KG context to high-weight network edges. This function does not require a specific backend and works with user-provided tables exported from PhysioAnnotationHub/physioKG workflows.

## Usage

```
emgInterpretNetworkKG(
  network,
  node_metadata = NULL,
  kg_edges = NULL,
  threshold = NULL,
  top_n = 20L,
  window = NULL
)
```

**Arguments**

|                            |                                                                                                                                                                                          |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>network</code>       | A square matrix (channels x channels) or 3D array (window x channels x channels).                                                                                                        |
| <code>node_metadata</code> | Optional data.frame containing at least channel. Additional columns (e.g. <code>kg_node</code> , <code>muscle_name</code> , <code>muscle_group</code> ) are carried into the edge table. |
| <code>kg_edges</code>      | Optional data.frame of KG links. Expected columns are <code>node_a</code> , <code>node_b</code> , and optional <code>relation</code> .                                                   |
| <code>threshold</code>     | Optional edge threshold. Default is 75th percentile of upper-triangle weights.                                                                                                           |
| <code>top_n</code>         | Maximum number of edges returned after thresholding.                                                                                                                                     |
| <code>window</code>        | Optional window index when <code>network</code> is a 3D array. If <code>NULL</code> , uses the mean across windows.                                                                      |

**Value**

A list with:

**edge\_table** Ranked edge table with optional metadata/KG annotations.

**threshold** Applied threshold value.

**network\_matrix** Matrix used for interpretation.

**summary** List of summary statistics.

**kg\_relation\_summary** Relation counts from matched KG links, or `NULL`.

**Examples**

```
pe <- make_emg(n_time = 2000, n_channels = 4, sr = 1000)
net <- emgCoherenceNetwork(pe, freq_band = c(20, 150))
interp <- emgInterpretNetworkKG(net$network, top_n = 5)
interp$edge_table
```

---

emgMFCV

---

*Muscle-Fiber Conduction Velocity (MFCV)*


---

**Description**

Estimates muscle-fiber conduction velocity from the propagation delay of the EMG signal between pairs of electrodes aligned along the fiber direction. The delay between the two channels of each pair is estimated either by cross-correlation (with parabolic sub-sample interpolation) or from the slope of the cross-spectrum phase, and the velocity is the inter-electrode distance divided by the delay.

**Usage**

```

emgMFCV(
  x,
  electrode_pairs,
  ied_mm,
  method = c("xcorr", "phase"),
  band = c(20, 250),
  max_lag_ms = 25,
  assay_name = NULL
)

```

**Arguments**

|                              |                                                                                                                                                                                                                    |
|------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>               | A <code>PhysioExperiment</code> object with EMG data.                                                                                                                                                              |
| <code>electrode_pairs</code> | Channel pairs aligned along the muscle fibres, given as a list of length-2 integer vectors <code>list(c(prox, dist), ...)</code> or as a two-column matrix (one pair per row; column 1 proximal, column 2 distal). |
| <code>ied_mm</code>          | Inter-electrode distance in millimetres. A single value applied to all pairs, or one value per pair.                                                                                                               |
| <code>method</code>          | Delay-estimation method: "xcorr" (cross-correlation, default) or "phase" (cross-spectrum phase slope).                                                                                                             |
| <code>band</code>            | Frequency band <code>c(fmin, fmax)</code> in Hz used by the "phase" method (default <code>c(20, 250)</code> ).                                                                                                     |
| <code>max_lag_ms</code>      | Maximum delay searched by the "xcorr" method, in milliseconds (default: 25).                                                                                                                                       |
| <code>assay_name</code>      | Input assay name (default: first assay).                                                                                                                                                                           |

**Value**

A `data.frame` with one row per electrode pair, containing columns:

**pair** Integer pair index.

**ch1, ch2** Proximal and distal channel indices.

**delay\_ms** Estimated propagation delay in milliseconds.

**velocity\_m\_s** Conduction velocity in metres per second (`ied_mm / delay`).

**quality** Peak cross-correlation ("xcorr") or phase-fit  $R^2$  ("phase").

**References**

Farina, D. & Merletti, R. (2000). "Comparison of algorithms for estimation of EMG variables during voluntary isometric contractions." *Journal of Electromyography and Kinesiology*, 10(5), 337-349. doi:10.1016/S1050-6411(00)00025-0

Merletti, R. & Parker, P.A. (2004). "Electromyography: Physiology, Engineering, and Non-Invasive Applications." Wiley-IEEE Press. doi:10.1002/0471678384

**See Also**

[emgFatigueSlope\(\)](#) and [emgDimitrovIndex\(\)](#) for spectral fatigue metrics, [emgFatigue\(\)](#) for median/mean frequency tracking

**Examples**

```
# two electrodes 10 mm apart; distal is the proximal signal delayed 3 samples
sr <- 2000
set.seed(1)
prox <- as.numeric(stats::filter(rnorm(4000), rep(1, 5), sides = 2))
prox[is.na(prox)] <- 0
dist <- c(rep(0, 3), prox[seq_len(length(prox) - 3)])
pe <- PhysioExperiment(assays = list(raw = cbind(prox, dist)),
  samplingRate = sr)
emgMFCV(pe, electrode_pairs = list(c(1, 2)), ied_mm = 10)
```

emgOnsetDetect

*Detect EMG Muscle Activation Onset and Offset***Description**

Identifies when muscle activation begins and ends. The Hodges-Bui method rectifies the signal, smooths it with a centered moving average (`smooth_ms`, as in the original method's low-pass conditioning), and thresholds the smoothed trace at a multiple of baseline standard deviations. The Teager-Kaiser method applies the Teager-Kaiser energy operator, then the same rectify-and-smooth conditioning, before thresholding. Without smoothing (`smooth_ms = 0`) the rectified band-limited signal dips below threshold at every carrier zero-crossing, so contiguous suprathreshold runs rarely reach `min_duration_ms` and genuine bursts go undetected. The Bonato method uses a statistical double-threshold on the (optionally whitened) signal, declaring activation when at least `r` of `m` successive samples exceed a chi-square threshold set from a target false-alarm probability. The AGLR method applies Staude & Wolf's approximated generalized-likelihood-ratio change detector for a variance increase and refines the onset to the estimated change point.

**Usage**

```
emgOnsetDetect(
  x,
  method = c("hodges_bui", "teager_kaiser", "bonato", "aglr"),
  threshold_sd = 3,
  baseline_sec = 0.2,
  min_duration_ms = 50,
  smooth_ms = 25,
  r = 5L,
  m = 10L,
  whiten = TRUE,
  false_alarm = 0.05,
  assay_name = NULL
)
```

**Arguments**

|                              |                                                                                                                                                                                                                                                                                                                                                                                                                    |
|------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>               | A <code>PhysioExperiment</code> object with EMG data.                                                                                                                                                                                                                                                                                                                                                              |
| <code>method</code>          | Detection method: "hodges_bui" (baseline SD threshold), "teager_kaiser" (Teager-Kaiser energy operator), "bonato" (Bonato double-threshold) or "aglr" (approximated generalized likelihood ratio).                                                                                                                                                                                                                 |
| <code>threshold_sd</code>    | Number of baseline SDs above mean for the threshold in the "hodges_bui" and "teager_kaiser" methods (default: 3).                                                                                                                                                                                                                                                                                                  |
| <code>baseline_sec</code>    | Duration of the baseline period in seconds from the signal start, used to estimate baseline statistics (default: 0.2).                                                                                                                                                                                                                                                                                             |
| <code>min_duration_ms</code> | Minimum activation duration in ms to accept (default: 50).                                                                                                                                                                                                                                                                                                                                                         |
| <code>smooth_ms</code>       | Width in ms of the centered moving-average window applied to the rectified signal ("hodges_bui") or rectified Teager-Kaiser trace ("teager_kaiser") before baseline estimation and thresholding (default: 25, in the 10–50 ms range examined by Hodges & Bui 1996). Set to 0 to disable smoothing (pre-0.2.1 behavior). Ignored by the "bonato" and "aglr" methods, which operate on unsmoothed samples by design. |
| <code>r, m</code>            | Bonato double-threshold parameters: activation requires at least <code>r</code> of <code>m</code> successive samples above the detection threshold (defaults: 5 of 10). For "aglr", <code>m</code> is the sliding analysis-window length in samples.                                                                                                                                                               |
| <code>whiten</code>          | If TRUE (default), whiten the signal with an AR model estimated from the baseline before applying the "bonato"/"aglr" statistics, so the baseline approximates white Gaussian noise.                                                                                                                                                                                                                               |
| <code>false_alarm</code>     | Target false-alarm probability that sets the detection threshold for the "bonato" (per <code>m</code> -sample window) and "aglr" (per sample) methods (default: 0.05).                                                                                                                                                                                                                                             |
| <code>assay_name</code>      | Input assay name (default: first assay).                                                                                                                                                                                                                                                                                                                                                                           |

**Value**

A list with two `data.frames`:

**onsets** A `data.frame` with columns `channel` (integer channel index), `sample` (sample index of onset), and `time_sec` (onset time in seconds). For "bonato"/"aglr" a `stat` column holds the peak detector statistic of the activation.

**offsets** A `data.frame` with the matching `channel`, `sample`, `time_sec` (and `stat`) for each offset.

If no activations are detected, both `data.frames` have zero rows.

**References**

Hodges, P.W. & Bui, B.H. (1996). "A comparison of computer-based methods for the determination of onset of muscle contraction using electromyography." *Electroencephalography and Clinical Neurophysiology*, 101(6), 511-519. doi:10.1016/S0921-884X(96)95190-5

Bonato, P., D'Alessio, T. & Knaflitz, M. (1998). "A statistical method for the measurement of muscle activation intervals from surface myoelectric signal during gait." *IEEE Transactions on Biomedical Engineering*, 45(3), 287-299. doi:10.1109/10.661154

Staude, G. & Wolf, W. (1999). "Objective motor response onset detection in surface myoelectric signals." *Medical Engineering & Physics*, 21(6-7), 449-467. doi:10.1016/S1350-4533(99)00067-3

Solnik, S., Rider, P., Steinweg, K., DeVita, P. & Hortobagyi, T. (2010). "Teager-Kaiser energy operator signal conditioning improves EMG onset detection." *European Journal of Applied Physiology*, 110(3), 489-498. doi:10.1007/s00421-010-1521-8

### See Also

[emgEnvelope\(\)](#) for computing amplitude envelopes, [emgAmplitudeNormalize\(\)](#) for amplitude normalization, [emgFatigue\(\)](#) for fatigue analysis

### Examples

```
pe <- make_emg_contraction()
emgOnsetDetect(pe, method = "bonato")$onsets
```

---

emgPartialCoherenceNetwork

*EMG Partial Coherence Network Analysis*

---

### Description

Builds a static network from pairwise partial coherence, estimated from the inverse cross-spectral density matrix at each frequency.

### Usage

```
emgPartialCoherenceNetwork(
  x,
  freq_band = NULL,
  channels = NULL,
  nperseg = 256L,
  noverlap = NULL,
  assay_name = NULL,
  aggregate = c("mean", "max", "median"),
  threshold = NULL,
  ridge = 1e-06
)
```

### Arguments

|           |                                                                                |
|-----------|--------------------------------------------------------------------------------|
| x         | A <code>PhysioExperiment</code> object.                                        |
| freq_band | Optional numeric vector <code>c(low, high)</code> in Hz.                       |
| channels  | Integer vector of channel indices to include. If <code>NULL</code> , uses all. |
| nperseg   | Segment length for Welch estimation (default: 256).                            |
| noverlap  | Overlap length (default: <code>floor(nperseg / 2)</code> ).                    |

|            |                                                                |
|------------|----------------------------------------------------------------|
| assay_name | Input assay name. If NULL, uses default assay.                 |
| aggregate  | Aggregation across frequency bins: "mean", "max", or "median". |
| threshold  | Optional threshold for binary adjacency matrix.                |
| ridge      | Ridge regularization added to spectral matrix inversion.       |

### Value

A list with:

**network** Numeric matrix (channel x channel) of partial coherence.

**adjacency** Logical matrix after thresholding, or NULL.

**partial\_coherence** 3D array (freq x channel x channel).

**frequencies** Frequency vector (Hz).

**channel\_names** Channel labels used in the network.

### See Also

[emgCoherenceNetwork\(\)](#), [emgWPLINetwork\(\)](#)

### Examples

```
pe <- make_emg(n_time = 2000, n_channels = 4, sr = 1000)
res <- emgPartialCoherenceNetwork(pe, freq_band = c(20, 150))
res$network
```

---

emgQCgate

---

*EMG Quality-Control Gate*


---

### Description

Applies pass/fail thresholds to the [emgQualityCheck\(\)](#) metrics and reports, per channel, whether the signal is acceptable and, if not, why.

### Usage

```
emgQCgate(
  x,
  min_snr = 6,
  max_powerline = 0.1,
  max_clipping = 1,
  reject_ecg = TRUE,
  assay_name = NULL,
  ...
)
```

**Arguments**

|                            |                                                             |
|----------------------------|-------------------------------------------------------------|
| <code>x</code>             | A <code>PhysioExperiment</code> object with EMG data.       |
| <code>min_snr</code>       | Minimum acceptable SNR in dB (default: 6).                  |
| <code>max_powerline</code> | Maximum acceptable power-line ratio (default: 0.10).        |
| <code>max_clipping</code>  | Maximum acceptable clipping percentage (default: 1).        |
| <code>reject_ecg</code>    | If TRUE (default), ECG-contaminated channels fail the gate. |
| <code>assay_name</code>    | Input assay name (default: first assay).                    |
| <code>...</code>           | Passed to <code>emgQualityCheck()</code> .                  |

**Value**

A list with `pass` (TRUE if every channel passes) and `channels`, a data.frame with columns `channel`, `pass` and `reasons` (a comma-separated string of failed criteria, or "").

**References**

McManus, L., De Vito, G. & Lowery, M.M. (2020). "Analysis and biophysics of surface EMG for physiotherapists and kinesiologists." *Frontiers in Neurology*, 11, 576729. doi:10.3389/fneur.2020.576729

**See Also**

`emgQualityCheck()`, `emgRemoveECG()`

**Examples**

```
set.seed(1)
sig <- rnorm(4000, sd = 0.02); sig[1500:2500] <- rnorm(1001, sd = 0.4)
pe <- PhysioExperiment(assays = list(raw = matrix(sig, ncol = 1)),
                      samplingRate = 1000)
emgQCgate(pe)$pass
```

---

emgQualityCheck

---

*EMG Signal Quality Check*


---

**Description**

Computes per-channel surface-EMG quality metrics: signal-to-noise ratio, resting baseline noise, power-line interference ratio at 50 and 60 Hz, clipping/saturation percentage, and an ECG-contamination flag based on the periodicity of the QRS-band envelope. When the `PhysioECG` package is available the R-peak count it detects is reported as an additional diagnostic.

**Usage**

```
emgQualityCheck(
  x,
  ecg_score_threshold = 0.5,
  rms_window_ms = 100,
  assay_name = NULL
)
```

**Arguments**

**x** A `PhysioExperiment` object with EMG data.

**ecg\_score\_threshold** QRS-band periodicity score above which a channel is flagged as ECG-contaminated (default: 0.5).

**rms\_window\_ms** Window (ms) for the moving-RMS used to estimate signal and rest levels (default: 100).

**assay\_name** Input assay name (default: first assay).

**Value**

A `data.frame` with one row per channel:

**channel** Integer channel index.

**snr\_db** Signal-to-noise ratio in dB (active vs rest RMS).

**baseline\_noise** Resting baseline RMS (noise floor).

**powerline\_50, powerline\_60** Fraction of total power within +/-1 Hz of 50 and 60 Hz.

**powerline\_ratio** The larger of `powerline_50/powerline_60`.

**clipping\_pct** Percentage of samples at the signal extremes (ADC saturation).

**ecg\_score** QRS-band periodicity score from 0 (none) to 1 (strong).

**ecg\_contamination** TRUE if `ecg_score` exceeds the threshold.

**References**

McManus, L., De Vito, G. & Lowery, M.M. (2020). "Analysis and biophysics of surface EMG for physiotherapists and kinesiologists." *Frontiers in Neurology*, 11, 576729. doi:10.3389/fneur.2020.576729

**See Also**

[emgRemoveECG\(\)](#) to remove ECG contamination, [emgQCgate\(\)](#) for pass/fail gating, [emgEnvelope\(\)](#)

**Examples**

```
set.seed(1)
sig <- rnorm(4000, sd = 0.02)
sig[1500:2500] <- rnorm(1001, sd = 0.4) # a contraction
pe <- PhysioExperiment(assays = list(raw = matrix(sig, ncol = 1)),
  samplingRate = 1000)
emgQualityCheck(pe)
```

emgRemoveECG

*Remove ECG Contamination from EMG***Description**

Removes cardiac (ECG/QRS) contamination from surface EMG using a high-pass filter, QRS template subtraction, or gating (blanking) of the QRS complexes.

**Usage**

```
emgRemoveECG(
  x,
  method = c("highpass", "template", "gating"),
  ecg_channel = NULL,
  hp_cutoff = 30,
  qrs_ms = 60,
  assay_name = NULL,
  output_assay = "ecg_removed"
)
```

**Arguments**

|              |                                                                                                                                                                               |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x            | A PhysioExperiment object with EMG data.                                                                                                                                      |
| method       | Removal method: "highpass" (default; FFT high-pass at hp_cutoff), "template" (average-QRS template subtraction), or "gating" (linear-interpolation blanking of QRS windows).  |
| ecg_channel  | Optional channel index of a dedicated ECG reference used for R-peak detection in the "template"/"gating" methods; if NULL, R-peaks are detected from each EMG channel itself. |
| hp_cutoff    | High-pass cutoff in Hz for the "highpass" method (default: 30).                                                                                                               |
| qrs_ms       | Half-width in ms of the QRS window for "template"/"gating" (default: 60).                                                                                                     |
| assay_name   | Input assay name (default: first assay).                                                                                                                                      |
| output_assay | Output assay name (default: "ecg_removed").                                                                                                                                   |

**Value**

A PhysioExperiment object with an added assay output\_assay containing the cleaned signal.

**References**

Drake, J.D.M. & Callaghan, J.P. (2006). "Elimination of electrocardiogram contamination from electromyogram signals: An evaluation of currently used removal techniques." *Journal of Electromyography and Kinesiology*, 16(2), 175-187. doi:10.1016/j.jelekin.2005.07.003

Willigenburg, N.W., Daffertshofer, A., Kingma, I. & van Dieen, J.H. (2012). "Removing ECG contamination from EMG recordings." *Journal of Electromyography and Kinesiology*, 22(3), 485-493. doi:10.1016/j.jelekin.2012.01.001

**See Also**

[emgQualityCheck\(\)](#) to detect contamination, [emgQCgate\(\)](#)

**Examples**

```
set.seed(1)
sr <- 1000; n <- 5000
emg <- rnorm(n, sd = 0.1)
qrs <- numeric(n)
for (p in seq(500, n - 200, by = 1000)) qrs[p + -15:15] <- 3 * exp(-(-15:15)^2 / 20)
pe <- PhysioExperiment(assays = list(raw = matrix(emg + qrs, ncol = 1)),
                        samplingRate = sr)
clean <- emgRemoveECG(pe, method = "highpass")
```

---

|                    |                             |
|--------------------|-----------------------------|
| emgSpectralMoments | <i>EMG Spectral Moments</i> |
|--------------------|-----------------------------|

---

**Description**

Computes spectral moments (M0, M1, M2) over sliding windows. M0 is total power, M1 is the first spectral moment (related to mean frequency), and M2 is the second moment (related to bandwidth). These can be combined to derive the mean frequency (M1/M0) and spectral bandwidth.

**Usage**

```
emgSpectralMoments(x, window_sec = 1, overlap = 0.5, assay_name = NULL)
```

**Arguments**

|                   |                                            |
|-------------------|--------------------------------------------|
| <b>x</b>          | A PhysioExperiment object with EMG data.   |
| <b>window_sec</b> | Analysis window in seconds (default: 1.0). |
| <b>overlap</b>    | Overlap fraction (default: 0.5).           |
| <b>assay_name</b> | Input assay name (default: first assay).   |

**Value**

A data.frame with one row per channel per window, containing columns:

- channel** Integer channel index.
- window** Integer window number (1-indexed).
- m0** Zeroth spectral moment (total power).
- m1** First spectral moment (frequency-weighted power).
- m2** Second spectral moment (frequency-squared-weighted power).

## References

De Luca, C.J. (1984). "Myoelectrical manifestations of localized muscular fatigue in humans." *Critical Reviews in Biomedical Engineering*, 11(4), 251-279.

Merletti, R. & Parker, P.A. (2004). "Electromyography: Physiology, Engineering, and Non-Invasive Applications." Wiley-IEEE Press. doi:10.1002/0471678384

## See Also

[emgFatigue\(\)](#) for median and mean frequency tracking, [emgFatigueIndex\(\)](#) for summary fatigue metric, [emgEnvelope\(\)](#) for time-domain amplitude analysis

## Examples

```
pe <- make_emg_fatigue(n_time = 10000, sr = 1000)
sm <- emgSpectralMoments(pe, window_sec = 1.0)
head(sm)
```

---

emgWPLINetwork

---

*EMG Weighted Phase-Lag Index (wPLI) Network*


---

## Description

Builds a static network from pairwise weighted phase-lag index (wPLI), which is less sensitive to zero-lag coupling artifacts.

## Usage

```
emgWPLINetwork(
  x,
  freq_band = NULL,
  channels = NULL,
  nperseg = 256L,
  noverlap = NULL,
  assay_name = NULL,
  aggregate = c("mean", "max", "median"),
  threshold = NULL,
  debiased = FALSE
)
```

## Arguments

|                        |                                                                                |
|------------------------|--------------------------------------------------------------------------------|
| <code>x</code>         | A <code>PhysioExperiment</code> object.                                        |
| <code>freq_band</code> | Optional numeric vector <code>c(low, high)</code> in Hz.                       |
| <code>channels</code>  | Integer vector of channel indices to include. If <code>NULL</code> , uses all. |
| <code>nperseg</code>   | Segment length for Welch estimation (default: 256).                            |
| <code>noverlap</code>  | Overlap length (default: <code>floor(nperseg / 2)</code> ).                    |

|            |                                                                |
|------------|----------------------------------------------------------------|
| assay_name | Input assay name. If NULL, uses default assay.                 |
| aggregate  | Aggregation across frequency bins: "mean", "max", or "median". |
| threshold  | Optional threshold for binary adjacency matrix.                |
| debiased   | Logical; if TRUE, uses debiased wPLI estimator.                |

Value

- A list with:
- network** Numeric matrix (channel x channel) of wPLI values.
  - adjacency** Logical matrix after thresholding, or NULL.
  - wpli** 3D array (freq x channel x channel).
  - frequencies** Frequency vector (Hz).
  - channel\_names** Channel labels used in the network.

See Also

[emgCoherenceNetwork\(\)](#), [emgPartialCoherenceNetwork\(\)](#)

Examples

```
pe <- make_emg(n_time = 2000, n_channels = 4, sr = 1000)
res <- emgWPLINetwork(pe, freq_band = c(20, 150))
res$network
```

---

|                 |                                      |
|-----------------|--------------------------------------|
| getEMGElectrode | <i>Get SENIAM Electrode Metadata</i> |
|-----------------|--------------------------------------|

---

Description

Retrieves the SENIAM electrode/muscle metadata previously attached with [setEMGElectrode\(\)](#).

Usage

```
getEMGElectrode(x)
```

Arguments

x                    A `PhysioExperiment` object.

Value

A `data.frame` with one row per channel and columns `channel`, `muscle`, `placement`, `orientation`, `ied_mm`, `reference` and `side` (NA where unset).

**See Also**

[setEMGElectrode\(\)](#), [seniamMuscles\(\)](#)

**Examples**

```
pe <- PhysioExperiment(assays = list(raw = matrix(rnorm(100), ncol = 1)),
                        samplingRate = 1000)
pe <- setEMGElectrode(pe, muscle = "Tibialis anterior")
getEMGElectrode(pe)
```

---

hdEMGDecompose

---

*Decompose HD-sEMG into motor units (PhysioHDEMG bridge)*


---

**Description**

Convenience bridge that forwards to `PhysioHDEMG::hdEMGDecompose()` for high-density surface EMG motor-unit decomposition by convolutive blind source separation. The `PhysioHDEMG` package must be installed. See [PhysioHDEMG::hdEMGDecompose\(\)](#) for the full argument list and return value.

**Usage**

```
hdEMGDecompose(x, ...)
```

**Arguments**

|                  |                                                                                                                                               |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>   | HD-sEMG data: an <code>n_time</code> x <code>n_channels</code> matrix, a <code>PhysioExperiment</code> , or an <code>hdemg_sim</code> object. |
| <code>...</code> | Further arguments passed to <code>PhysioHDEMG::hdEMGDecompose()</code> .                                                                      |

**Value**

An `hdemg_decomposition` object (see `PhysioHDEMG`).

**See Also**

[muscleSynergy\(\)](#) for whole-muscle synergy decomposition.

**Examples**

```
# Requires the optional PhysioHDEMG backend (Suggests).
if (requireNamespace("PhysioHDEMG", quietly = TRUE)) {
  sim <- PhysioHDEMG::make_hdemg_sim(n_units = 2, duration_sec = 1)
  dec <- hdEMGDecompose(sim, n_units = 4)
}
```

---

`make_emg`*Create a Basic EMG PhysioExperiment*

---

## Description

Generates a synthetic multi-channel EMG `PhysioExperiment` object with simulated muscle activity. Each channel contains baseline noise with a burst of higher-amplitude activity in the middle 40 percent of the signal (from 30 to 70 percent), mimicking a typical voluntary contraction.

## Usage

```
make_emg(n_time = 2000, n_channels = 4, sr = 1000)
```

## Arguments

|                         |                                        |
|-------------------------|----------------------------------------|
| <code>n_time</code>     | Number of time points (default: 2000). |
| <code>n_channels</code> | Number of EMG channels (default: 4).   |
| <code>sr</code>         | Sampling rate in Hz (default: 1000).   |

## Value

A `PhysioExperiment` object with a single "raw" assay containing simulated EMG data (time x channels matrix), channel metadata in `colData`, and the specified sampling rate.

## References

Merletti, R. & Parker, P.A. (2004). "Electromyography: Physiology, Engineering, and Non-Invasive Applications." Wiley-IEEE Press. doi:10.1002/0471678384

## See Also

[make\\_emg\\_contraction\(\)](#) for EMG with a defined contraction window, [make\\_emg\\_fatigue\(\)](#) for EMG with fatigue progression, [emgEnvelope\(\)](#) for extracting amplitude envelopes

## Examples

```
pe <- make_emg()
pe
dim(SummarizedExperiment::assay(pe, "raw"))
```

---

|                      |                                                    |
|----------------------|----------------------------------------------------|
| make_emg_contraction | Create EMG PhysioExperiment with Known Contraction |
|----------------------|----------------------------------------------------|

---

## Description

Generates a synthetic single-channel EMG PhysioExperiment with a clearly defined contraction period. The contraction region has substantially higher amplitude than the baseline, making it suitable for testing onset detection algorithms.

## Usage

```
make_emg_contraction(  
  n_time = 5000,  
  sr = 1000,  
  contraction_start = 0.3,  
  contraction_end = 0.7,  
  baseline_sd = 0.01,  
  contraction_sd = 0.5  
)
```

## Arguments

|                   |                                                               |
|-------------------|---------------------------------------------------------------|
| n_time            | Number of time points (default: 5000).                        |
| sr                | Sampling rate in Hz (default: 1000).                          |
| contraction_start | Proportion of signal where contraction begins (default: 0.3). |
| contraction_end   | Proportion of signal where contraction ends (default: 0.7).   |
| baseline_sd       | Standard deviation of baseline noise (default: 0.01).         |
| contraction_sd    | Standard deviation of contraction activity (default: 0.5).    |

## Value

A PhysioExperiment object with a single "raw" assay containing a one-channel EMG signal with a known contraction window. The contraction region is defined by contraction\_start and contraction\_end as proportions of total signal length.

## References

Hodges, P.W. & Bui, B.H. (1996). "A comparison of computer-based methods for the determination of onset of muscle contraction using electromyography." *Electroencephalography and Clinical Neurophysiology*, 101(6), 511-519. doi:10.1016/S0921-884X(96)95190-5

## See Also

[emgOnsetDetect\(\)](#) for detecting the contraction onset, [emgEnvelope\(\)](#) for extracting the amplitude envelope, [make\\_emg\(\)](#) for basic multi-channel EMG, [make\\_emg\\_fatigue\(\)](#) for EMG with fatigue progression

**Examples**

```
pe <- make_emg_contraction()
pe
onset <- emgOnsetDetect(pe)
onset$onsets
```

---

make\_emg\_fatigue

---

*Create EMG PhysioExperiment with Fatigue Progression*


---

**Description**

Generates a synthetic single-channel EMG PhysioExperiment that simulates a fatiguing isometric contraction. The median frequency of the signal decreases progressively over time (from 80 Hz to approximately 44 Hz across 10 segments), mimicking the spectral compression characteristic of muscle fatigue.

**Usage**

```
make_emg_fatigue(n_time = 10000, sr = 1000)
```

**Arguments**

|        |                                         |
|--------|-----------------------------------------|
| n_time | Number of time points (default: 10000). |
| sr     | Sampling rate in Hz (default: 1000).    |

**Value**

A PhysioExperiment object with a single "raw" assay containing a one-channel EMG signal exhibiting progressive median frequency decrease across 10 equal-length segments.

**References**

De Luca, C.J. (1997). "The use of surface electromyography in biomechanics." *Journal of Applied Biomechanics*, 13(2), 135-163. doi:10.1123/jab.13.2.135

Merletti, R. & Parker, P.A. (2004). "Electromyography: Physiology, Engineering, and Non-Invasive Applications." Wiley-IEEE Press. doi:10.1002/0471678384

**See Also**

[emgFatigue\(\)](#) for tracking median frequency over time, [emgFatigueIndex\(\)](#) for computing a summary fatigue metric, [emgSpectralMoments\(\)](#) for spectral moment analysis, [make\\_emg\(\)](#) for basic multi-channel EMG

**Examples**

```
pe <- make_emg_fatigue()
pe
fatigue <- emgFatigue(pe)
head(fatigue)
```

---

muscleSynergy

*Muscle Synergy Decomposition*


---

## Description

Decomposes multi-channel EMG into muscle synergies using matrix factorization.

## Usage

```
muscleSynergy(
  x,
  n_synergies,
  method = c("nmf", "pca", "ica"),
  n_restarts = 1L,
  max_iter = 200L,
  tol = 1e-04,
  seed = NULL,
  assay_name = NULL
)
```

## Arguments

|                          |                                                                                                                                                       |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>           | A <code>PhysioExperiment</code> object with multi-channel EMG.                                                                                        |
| <code>n_synergies</code> | Number of synergies to extract.                                                                                                                       |
| <code>method</code>      | Decomposition method: "nmf" (non-negative matrix factorization), "pca" (principal component analysis), or "ica" (independent component analysis).     |
| <code>n_restarts</code>  | Number of random NMF restarts; the fit with the lowest reconstruction error (highest VAF) is returned (default: 1). Ignored for "pca".                |
| <code>max_iter</code>    | Maximum iterations for NMF (default: 200).                                                                                                            |
| <code>tol</code>         | Convergence tolerance for NMF (default: 1e-4).                                                                                                        |
| <code>seed</code>        | Optional integer seed for reproducible random initialization (NMF restarts and ICA). If set, the returned best-of-restarts solution is deterministic. |
| <code>assay_name</code>  | Input assay name (default: first assay).                                                                                                              |

## Value

A list with:

- `W`: Synergy weight matrix (`n_synergies` x channels)
- `H`: Activation pattern matrix (time x `n_synergies`)
- `vaf`: Variance accounted for (0-1)
- `method`: Method used
- `n_restarts`: Number of restarts used
- `all_vaf`: VAF of each NMF restart (NA for `pca/ica`)

- `convergence`: For NMF, a list with the best restart's iterations and converged flag (NULL otherwise)
- `original_data`: Original data matrix for reconstruction

## References

- Lee, D.D. & Seung, H.S. (1999). "Learning the parts of objects by non-negative matrix factorization." *Nature*, 401(6755), 788-791. doi:10.1038/44565
- Tresch, M.C., Cheung, V.C.K. & d'Avella, A. (2006). "Matrix factorization algorithms for the identification of muscle synergies." *Journal of Neurophysiology*, 95(4), 2199-2212. doi:10.1152/jn.00222.2005

## See Also

[muscleSynergyOrder\(\)](#) for model-order (synergy count) selection, [synergyReconstruct\(\)](#) for reconstructing data from synergies, [synergyCompare\(\)](#) for comparing synergy solutions

## Examples

```
pe <- make_emg(n_time = 2000, n_channels = 4, sr = 1000)
syn <- muscleSynergy(pe, n_synergies = 2, method = "nmf", seed = 1)
syn$vaf
dim(syn$W) # n_synergies x channels
```

---

muscleSynergyMixture    *Mixture-based muscle synergy analysis*

---

## Description

Fits a mixture of factor analysers to per-subject muscle activation data to identify subgroups of subjects with distinct muscle-synergy structure, rather than forcing a single global synergy model on a heterogeneous population. Each cluster's factor loading matrix is that subgroup's synergy set.

## Usage

```
muscleSynergyMixture(
  list_of_data,
  n_clusters,
  n_factors,
  method = c("mfa", "mpca"),
  max_iter = 100,
  tol = 1e-04,
  n_init = 5,
  seed = NULL
)
```

**Arguments**

|                           |                                                                                                                                                                         |
|---------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>list_of_data</code> | A list of non-empty numeric matrices, one per subject, each time x muscle (rows = samples/time, columns = muscles; all subjects share the same set of muscles/columns). |
| <code>n_clusters</code>   | Number of subgroups (mixture components).                                                                                                                               |
| <code>n_factors</code>    | Number of synergies (factors) per subgroup.                                                                                                                             |
| <code>method</code>       | "mfa" (mixture of factor analysers, per-muscle diagonal noise; default) or "mpca" (mixture of probabilistic PCA, isotropic noise).                                      |
| <code>max_iter</code>     | Maximum EM iterations (default 100).                                                                                                                                    |
| <code>tol</code>          | Log-likelihood convergence tolerance (default 1e-4).                                                                                                                    |
| <code>n_init</code>       | Random/k-means restarts; the best log-likelihood is kept (default 5).                                                                                                   |
| <code>seed</code>         | Optional integer seed.                                                                                                                                                  |

**Value**

An object of class "synergy\_mixture": a list with cluster (subgroup label per subject), synergies (length-n\_clusters list of muscle x factor loading matrices), uniqueness (per-cluster diagonal noise), mean (per-cluster mean), proportions (mixing weights), responsibilities (subject x cluster), loglik, and bic.

**References**

Matsui Y. Mixture-Based Muscle Synergy Analysis. SSRN preprint. <https://ssrn.com/abstract=6798399>

Ghahramani Z, Hinton GE (1996). The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, University of Toronto.

Matsui Y. synergyMixR: Mixture-Based Muscle Synergy Analysis. <https://github.com/matsui-lab/synergyMixR>

**Examples**

```
set.seed(1)
M <- 6; r <- 2
LA <- matrix(abs(rnorm(M * r)), M, r)      # subgroup A synergies
LB <- matrix(abs(rnorm(M * r)), M, r)      # subgroup B synergies
gen <- function(L, n = 60) t(L %*% matrix(rnorm(r * n), r, n) +
                                matrix(rnorm(M * n, 0, 0.3), M, n))
data <- c(replicate(6, gen(LA), simplify = FALSE),
          replicate(6, gen(LB), simplify = FALSE))
fit <- muscleSynergyMixture(data, n_clusters = 2, n_factors = 2, seed = 1)
fit$cluster
```

---

muscleSynergyOrder      *Select Muscle Synergy Model Order*

---

### Description

Sweeps the number of synergies from 1 to max\_synergies, fits a multi-restart NMF at each, and selects the model order from the variance-accounted-for (VAF) curve. The selected order is the smallest number of synergies whose global VAF reaches vaf\_threshold; if the threshold is never reached, the knee (elbow) of the VAF curve is used instead.

### Usage

```
muscleSynergyOrder(
  x,
  max_synergies,
  vaf_threshold = 0.9,
  per_synergy_vaf = 0.75,
  n_restarts = 10L,
  max_iter = 200L,
  tol = 1e-04,
  seed = NULL,
  assay_name = NULL
)
```

### Arguments

|                 |                                                                                                              |
|-----------------|--------------------------------------------------------------------------------------------------------------|
| x               | A PhysioExperiment object with multi-channel EMG.                                                            |
| max_synergies   | Maximum number of synergies to test (capped at the number of channels).                                      |
| vaf_threshold   | Global VAF threshold for selection (default: 0.90).                                                          |
| per_synergy_vaf | Per-channel VAF each muscle should reach at the selected order; reported as a quality check (default: 0.75). |
| n_restarts      | Number of NMF restarts per synergy count (default: 10).                                                      |
| max_iter        | Maximum NMF iterations (default: 200).                                                                       |
| tol             | NMF convergence tolerance (default: 1e-4).                                                                   |
| seed            | Optional integer seed making the whole sweep reproducible.                                                   |
| assay_name      | Input assay name (default: first assay).                                                                     |

### Value

A list with:

**vaf\_curve** A data.frame with columns n\_synergies and vaf.

**selected** The selected number of synergies.

**selection\_rule** "vaf\_threshold" or "knee".

**vaf\_threshold, per\_synergy\_vaf** The thresholds used.

**per\_channel\_vaf** Per-channel VAF at the selected order.

**per\_synergy\_vaf\_met** TRUE if every channel's VAF meets per\_synergy\_vaf.

**fit** The `muscleSynergy()` result at the selected order.

## References

Cheung, V.C.K. et al. (2005). "Central and sensory contributions to the activation and organization of muscle synergies during natural motor behaviors." *Journal of Neuroscience*, 25(27), 6419-6434. doi:10.1523/JNEUROSCI.4904-04.2005

Tresch, M.C., Cheung, V.C.K. & d'Avella, A. (2006). "Matrix factorization algorithms for the identification of muscle synergies." *Journal of Neurophysiology*, 95(4), 2199-2212. doi:10.1152/jn.00222.2005

## See Also

`muscleSynergy()` for the underlying decomposition

## Examples

```
# three synergies driving disjoint muscle pairs
set.seed(1)
tt <- seq_len(500)
H <- sapply(c(150, 250, 350), function(c0) exp(-((tt - c0) / 40)^2))
W <- rbind(c(1, 1, 0, 0, 0, 0), c(0, 0, 1, 1, 0, 0), c(0, 0, 0, 0, 1, 1))
data <- H %*% W + matrix(abs(rnorm(500 * 6, sd = 0.02))), 500, 6)
pe <- PhysioExperiment(assays = list(env = data), samplingRate = 1000)
ord <- muscleSynergyOrder(pe, max_synergies = 5, seed = 1)
ord$selected
ord$vaf_curve
```

---

|                     |                                                                   |
|---------------------|-------------------------------------------------------------------|
| muscleSynergyTensor | <i>Muscle-synergy tensor decomposition (non-negative PARAFAC)</i> |
|---------------------|-------------------------------------------------------------------|

---

## Description

Non-negative CANDECOMP/PARAFAC of a three-way EMG tensor `muscle x time x trial`. Each of the `n_synergies` components is a rank-one term with a muscle weight vector, a temporal activation profile, and a per-trial loading — the multiway analogue of a muscle synergy that captures how the same spatial/temporal modules are reused and rescaled across trials or conditions.

**Usage**

```
muscleSynergyTensor(
  tensor,
  n_synergies,
  max_iter = 500,
  tol = 1e-06,
  restarts = 5,
  seed = NULL
)
```

**Arguments**

|             |                                                                         |
|-------------|-------------------------------------------------------------------------|
| tensor      | A non-negative numeric array with dimensions muscle x time x trial.     |
| n_synergies | Number of synergies (rank) to extract.                                  |
| max_iter    | Maximum multiplicative-update iterations (default 500).                 |
| tol         | Relative reconstruction-error tolerance for convergence (default 1e-6). |
| restarts    | Random restarts; the best (lowest error) is kept (default 5).           |
| seed        | Optional integer seed for reproducibility.                              |

**Value**

An object of class "synergy\_tensor": a list with muscle\_weights (muscle x synergy), temporal (time x synergy), trial\_loadings (trial x synergy), vaf (variance accounted for), iterations, and n\_synergies.

**References**

Cichocki A, Zdunek R, Phan AH, Amari S (2009). Nonnegative Matrix and Tensor Factorizations. Wiley.

**Examples**

```
set.seed(1)
M <- 6; Tt <- 50; K <- 8; R <- 2
A0 <- matrix(runif(M * R), M, R); B0 <- matrix(runif(Tt * R), Tt, R)
C0 <- matrix(runif(K * R), K, R)
X <- array(0, c(M, Tt, K))
for (r in 1:R) for (k in 1:K) X[, , k] <- X[, , k] +
  (A0[, r] %o% B0[, r]) * C0[k, r]
fit <- muscleSynergyTensor(X, n_synergies = 2, restarts = 3, seed = 1)
fit$vaf
```

---

muscleSynergyTensorOrder

*Select the number of tensor synergies by a VAF threshold*


---

### Description

Fits non-negative PARAFAC for increasing rank and returns the smallest number of synergies whose reconstruction VAF reaches `vaf_threshold`.

### Usage

```
muscleSynergyTensorOrder(tensor, max_synergies = 6, vaf_threshold = 0.9, ...)
```

### Arguments

`tensor`            A non-negative muscle x time x trial array.  
`max_synergies`    Largest rank to try (default 6).  
`vaf_threshold`    VAF target (default 0.90).  
`...`              Passed to `muscleSynergyTensor()`.

### Value

A list with `n_synergies` (selected), `vaf` (per rank), and `fit` (the selected "synergy\_tensor").

### Examples

```
set.seed(1)
M <- 6; Tt <- 40; K <- 8; R <- 2
A0 <- matrix(runif(M * R), M, R); B0 <- matrix(runif(Tt * R), Tt, R)
C0 <- matrix(runif(K * R), K, R)
X <- array(0, c(M, Tt, K))
for (r in 1:R) for (k in 1:K) X[, , k] <- X[, , k] +
  (A0[, r] %o% B0[, r]) * C0[k, r]
ord <- muscleSynergyTensorOrder(X, max_synergies = 3, seed = 1)
ord$n_synergies
```

---

physioSynergyMixture    *Mixture-based muscle synergy analysis on PhysioExperiment objects*


---

### Description

Thin entry point that runs `muscleSynergyMixture()` directly on ecosystem data: it reads each subject's time x muscle activation from a `PhysioExperiment` (or accepts matrices) and fits the subgroup mixture.

**Usage**

```

physioSynergyMixture(
  pe_list,
  n_clusters,
  n_factors,
  method = c("mfa", "mpca"),
  assay_name = NULL,
  ...
)

```

**Arguments**

|                                    |                                                                                                                          |
|------------------------------------|--------------------------------------------------------------------------------------------------------------------------|
| <code>pe_list</code>               | A list with one element per subject, each a <code>PhysioExperiment</code> (its assay is read) or a time x muscle matrix. |
| <code>n_clusters, n_factors</code> | Number of subgroups and synergies (factors).                                                                             |
| <code>method</code>                | "mfa" or "mpca" (see <a href="#">muscleSynergyMixture()</a> ).                                                           |
| <code>assay_name</code>            | Assay to read from each <code>PhysioExperiment</code> (default: its default assay).                                      |
| <code>...</code>                   | Further arguments passed to <a href="#">muscleSynergyMixture()</a> .                                                     |

**Value**

A "synergy\_mixture" object.

**Examples**

```

set.seed(1)
mk <- function(L) {
  pe <- PhysioExperiment::PhysioExperiment(
    assays = list(raw = t(L %*% matrix(rnorm(2 * 50), 2, 50) +
      matrix(rnorm(6 * 50, 0, .3), 6, 50))),
    samplingRate = 100)
  pe
}
LA <- matrix(abs(rnorm(12)), 6, 2); LB <- matrix(abs(rnorm(12)), 6, 2)
pes <- c(replicate(5, mk(LA), simplify = FALSE),
  replicate(5, mk(LB), simplify = FALSE))
fit <- physioSynergyMixture(pes, n_clusters = 2, n_factors = 2, seed = 1)
fit$cluster

```

---

plotCoactivation

*Plot Agonist-Antagonist Co-Activation*


---

**Description**

Overlays the agonist and antagonist envelopes and shades their common (co-contraction) area, annotated with the co-contraction index.

**Usage**

```
plotCoactivation(  
  x,  
  agonist,  
  antagonist,  
  method = c("falconer_winter", "rudolph", "frost"),  
  normalize = TRUE,  
  assay_name = NULL  
)
```

**Arguments**

|                     |                                                                                    |
|---------------------|------------------------------------------------------------------------------------|
| x                   | A PhysioExperiment object of EMG envelopes.                                        |
| agonist, antagonist | Channel index or label of the two muscles.                                         |
| method              | Co-contraction index for the annotation (see <a href="#">emgCoContraction()</a> ). |
| normalize           | Peak-normalize each channel before plotting (default: TRUE).                       |
| assay_name          | Input assay name (default: first assay).                                           |

**Value**

A ggplot object.

**References**

Falconer, K. & Winter, D.A. (1985). "Quantitative assessment of co-contraction at the ankle joint in walking." *Electromyography and Clinical Neurophysiology*, 25(2-3), 135-149.

**See Also**

[emgCoContraction\(\)](#)

**Examples**

```
t <- seq_len(1000)  
a <- exp(-((t - 400) / 120)^2); b <- exp(-((t - 600) / 120)^2)  
pe <- PhysioExperiment(assays = list(env = cbind(a, b)),  
  colData = S4Vectors::DataFrame(label = c("TA", "GAS")), samplingRate = 1000)  
plotCoactivation(pe, "TA", "GAS")
```

---

|                 |                                     |
|-----------------|-------------------------------------|
| plotEmgEnvelope | <i>Plot EMG Signal and Envelope</i> |
|-----------------|-------------------------------------|

---

## Description

Plots the raw signal and its amplitude envelope for one channel, optionally shading the resting noise floor from [emgQualityCheck\(\)](#).

## Usage

```
plotEmgEnvelope(
  x,
  channel = 1,
  method = "rms",
  window_ms = 50,
  qc = TRUE,
  assay_name = NULL
)
```

## Arguments

|            |                                                                           |
|------------|---------------------------------------------------------------------------|
| x          | A <code>PhysioExperiment</code> object with EMG data.                     |
| channel    | Channel index or label to plot (default: 1).                              |
| method     | Envelope method passed to <a href="#">emgEnvelope()</a> (default: "rms"). |
| window_ms  | Envelope window in ms (default: 50).                                      |
| qc         | If TRUE (default), shade the resting noise floor and annotate SNR.        |
| assay_name | Input assay name (default: first assay).                                  |

## Value

A `ggplot` object.

## References

De Luca, C.J. (1997). "The use of surface electromyography in biomechanics." *Journal of Applied Biomechanics*, 13(2), 135-163. doi:10.1123/jab.13.2.135

## See Also

[emgEnvelope\(\)](#), [emgQualityCheck\(\)](#)

## Examples

```
pe <- make_emg(n_time = 2000, n_channels = 1)
plotEmgEnvelope(pe)
```

---

|                |                                             |
|----------------|---------------------------------------------|
| plotEmgFatigue | <i>Plot EMG Spectral-Fatigue Regression</i> |
|----------------|---------------------------------------------|

---

**Description**

Scatters the windowed median or mean frequency against time with the fitted fatigue-regression line and slope annotation.

**Usage**

```
plotEmgFatigue(  
  x,  
  feature = c("mdf", "mnf"),  
  window_sec = 1,  
  overlap = 0.5,  
  assay_name = NULL  
)
```

**Arguments**

|                     |                                                     |
|---------------------|-----------------------------------------------------|
| x                   | A PhysioExperiment object with EMG data.            |
| feature             | "mdf" (median frequency) or "mnf" (mean frequency). |
| window_sec, overlap | Windowing passed to <a href="#">emgFatigue()</a> .  |
| assay_name          | Input assay name (default: first assay).            |

**Value**

A ggplot object (faceted by channel).

**References**

Merletti, R. & Parker, P.A. (2004). "Electromyography: Physiology, Engineering, and Non-Invasive Applications." Wiley-IEEE Press. doi:10.1002/0471678384

**See Also**

[emgFatigue\(\)](#), [emgFatigueSlope\(\)](#)

**Examples**

```
pe <- make_emg_fatigue()  
plotEmgFatigue(pe)
```

---

|              |                                               |
|--------------|-----------------------------------------------|
| plotEmgOnset | <i>Plot EMG Activation Onsets and Offsets</i> |
|--------------|-----------------------------------------------|

---

## Description

Plots the rectified signal with detected activation intervals shaded and onset/offset times marked.

## Usage

```
plotEmgOnset(x, channel = 1, method = "hodges_bui", assay_name = NULL, ...)
```

## Arguments

|            |                                                                               |
|------------|-------------------------------------------------------------------------------|
| x          | A <code>PhysioExperiment</code> object with EMG data.                         |
| channel    | Channel index or label to plot (default: 1).                                  |
| method     | Onset method passed to <code>emgOnsetDetect()</code> (default: "hodges_bui"). |
| assay_name | Input assay name (default: first assay).                                      |
| ...        | Further arguments for <code>emgOnsetDetect()</code> .                         |

## Value

A `ggplot` object.

## References

Hodges, P.W. & Bui, B.H. (1996). "A comparison of computer-based methods for the determination of onset of muscle contraction using electromyography." *Electroencephalography and Clinical Neurophysiology*, 101(6), 511-519. doi:10.1016/S0921-884X(96)95190-5

## See Also

[emgOnsetDetect\(\)](#)

## Examples

```
pe <- make_emg_contraction()
plotEmgOnset(pe)
```

---

plotSynergy

*Plot Muscle Synergies*


---

## Description

Renders a three-panel figure for a muscle-synergy decomposition: the synergy weight matrix  $W$  (heatmap), the activation time courses  $H$ , and the VAF curve.

## Usage

```
plotSynergy(x)
```

## Arguments

$x$  A [muscleSynergy\(\)](#) result, or a [muscleSynergyOrder\(\)](#) result (whose VAF curve and selected order are used).

## Value

A patchwork/ggplot object combining the three panels.

## References

Ting, L.H. & Chvatal, S.A. (2010). "Decomposing muscle activity in motor tasks: methods and interpretation." In Motor Control (Oxford).

## See Also

[muscleSynergy\(\)](#), [muscleSynergyOrder\(\)](#)

## Examples

```
set.seed(1)
tt <- seq_len(400)
H <- sapply(c(120, 200, 280), function(c0) exp(-((tt - c0) / 30)^2))
W <- rbind(c(1, 1, 0, 0, 0, 0), c(0, 0, 1, 1, 0, 0), c(0, 0, 0, 0, 1, 1))
data <- H %*% W + matrix(abs(rnorm(400 * 6, sd = 0.02))), 400, 6)
pe <- PhysioExperiment(assays = list(env = data), samplingRate = 1000)
ord <- muscleSynergyOrder(pe, max_synergies = 5, seed = 1)
plotSynergy(ord)
```

---

|                   |                                            |
|-------------------|--------------------------------------------|
| reconstructTucker | <i>Reconstruct a Tucker model (helper)</i> |
|-------------------|--------------------------------------------|

---

**Description**

Rebuilds the muscle x time x trial tensor from a Tucker core and factors; exposed mainly for examples and testing.

**Usage**

```
reconstructTucker(core, spatial, temporal, trial)
```

**Arguments**

core                    A  $P \times Q \times R$  core array.  
 spatial, temporal, trial                    Factor matrices.

**Value**

The reconstructed [muscle, time, trial] array.

**Examples**

```
set.seed(1)
A <- matrix(stats::runif(6 * 2), 6, 2)
B <- matrix(stats::runif(40 * 2), 40, 2)
C <- matrix(stats::runif(8 * 2), 8, 2)
G <- array(stats::runif(2 * 2 * 2), c(2, 2, 2))
X <- reconstructTucker(G, A, B, C)
dim(X) # 6 x 40 x 8
```

---

|               |                              |
|---------------|------------------------------|
| seniamMuscles | <i>SENIAM Muscle Catalog</i> |
|---------------|------------------------------|

---

**Description**

Returns the bundled catalog of SENIAM-recommended surface-EMG electrode locations (Hermens et al. 2000): for each muscle the recommended sensor location, the electrode orientation (fibre direction) and the recommended inter-electrode distance (mm).

**Usage**

```
seniamMuscles()
```

**Value**

A data.frame with columns muscle, location, orientation and ied\_mm.

**References**

Hermens, H.J., Freriks, B., Disselhorst-Klug, C. & Rau, G. (2000). "Development of recommendations for SEMG sensors and sensor placement procedures." Journal of Electromyography and Kinesiology, 10(5), 361-374. doi:10.1016/S1050-6411(00)00027-4

**See Also**

[setEMGElectrode\(\)](#) to attach electrode metadata to a PhysioExperiment, [getEMGElectrode\(\)](#) to retrieve it

**Examples**

```
head(seniamMuscles())
```

---

|                 |                                      |
|-----------------|--------------------------------------|
| setEMGElectrode | <i>Set SENIAM Electrode Metadata</i> |
|-----------------|--------------------------------------|

---

**Description**

Attaches standardized SENIAM electrode/muscle metadata to the channels of a PhysioExperiment, written as seniam\_\* columns of colData. Muscle names are validated against [seniamMuscles\(\)](#); a recognized muscle also back-fills its recommended electrode location, orientation and inter-electrode distance when those are not supplied.

**Usage**

```
setEMGElectrode(  
  x,  
  muscle,  
  placement = NULL,  
  ied_mm = NULL,  
  reference = NULL,  
  side = NULL,  
  channel = NULL  
)
```

**Arguments**

|           |                                                                                                      |
|-----------|------------------------------------------------------------------------------------------------------|
| x         | A PhysioExperiment object with EMG channels.                                                         |
| muscle    | Muscle name(s) (character). Length 1 (applied to all targeted channels) or one per targeted channel. |
| placement | Optional electrode location text; back-filled from the catalog when NULL.                            |

|           |                                                                                                         |
|-----------|---------------------------------------------------------------------------------------------------------|
| ied_mm    | Optional inter-electrode distance in mm; back-filled from the catalog (SENIAM default 20 mm) when NULL. |
| reference | Optional reference-electrode description.                                                               |
| side      | Optional body side ("left"/"right"/"L"/"R"/...).                                                        |
| channel   | Channels to annotate: integer indices, channel labels, or NULL (default) for all channels.              |

### Value

The `PhysioExperiment` with `seniam_muscle`, `seniam_placement`, `seniam_orientation`, `seniam_ied_mm`, `seniam_reference` and `seniam_side` added to `colData`.

### References

Hermens, H.J. et al. (2000). "Development of recommendations for SEMG sensors and sensor placement procedures." *Journal of Electromyography and Kinesiology*, 10(5), 361-374. doi:10.1016/S1050-6411(00)00027-4

### See Also

[getEMGElectrode\(\)](#), [seniamMuscles\(\)](#), [emgMFCV\(\)](#) which can use the stored inter-electrode distance

### Examples

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(200), ncol = 2)),
  colData = S4Vectors::DataFrame(label = c("ch1", "ch2")),
  samplingRate = 1000)
pe <- setEMGElectrode(pe, muscle = c("Biceps brachii", "Triceps brachii"),
  side = "right")
getEMGElectrode(pe)
```

---

spaceByTimeSynergy      *Space-by-time muscle synergy decomposition (sNM3F)*

---

### Description

Decomposes a set of trials' muscle x time EMG into shared SPATIAL synergies (which muscles group together), shared TEMPORAL synergies (when they are active) and a per-trial activation matrix linking them (Delis et al. 2014).

**Usage**

```
spaceByTimeSynergy(
  trials,
  n_spatial,
  n_temporal,
  max_iter = 500L,
  tol = 1e-06,
  n_restart = 5L,
  seed = NULL
)
```

**Arguments**

|                                    |                                                                                                  |
|------------------------------------|--------------------------------------------------------------------------------------------------|
| <code>trials</code>                | A list of muscle x time non-negative matrices (one per trial), or a [muscle, time, trial] array. |
| <code>n_spatial, n_temporal</code> | Numbers of spatial and temporal synergies.                                                       |
| <code>max_iter, tol</code>         | Multiplicative-update iterations and convergence tolerance.                                      |
| <code>n_restart</code>             | Random restarts (best kept; default 5).                                                          |
| <code>seed</code>                  | Optional RNG seed.                                                                               |

**Value**

a spacetime\_synergy: spatial (muscle x n\_spatial), temporal (time x n\_temporal),  
activation (list of n\_spatial x n\_temporal per trial), vaf, iterations.

**References**

Delis I, et al. (2014) Front Comput Neurosci 8:118.

**See Also**

[muscleSynergy\(\)](#), [tuckerSynergy\(\)](#)

**Examples**

```
set.seed(1)
Ws <- matrix(stats::runif(6 * 2), 6, 2); Wt <- matrix(stats::runif(50 * 2), 50, 2)
trials <- lapply(1:10, function(n) Ws %*% (matrix(stats::runif(4), 2, 2)) %*% t(Wt))
fit <- spaceByTimeSynergy(trials, 2, 2, n_restart = 2)
fit$vaf
```

---

splitHalfStability      *Split-half stability of inter-muscle coordination networks*


---

## Description

Tests whether an inter-muscle coordination network is real signal or a split-specific artefact: the recording is split into two independent halves, the same network is built on each with [coordinationNetwork\(\)](#), and the edge weights (upper triangle) and node strengths (row sums) are correlated between the halves. A correlation near 1 means each half rebuilds the same network. Different estimators trade sensitivity for specificity differently, and a sparse graph's node strengths are more split-sensitive than its edges, so build several and trust what reproduces in BOTH edges and topology.

## Usage

```
splitHalfStability(
  x,
  methods = c("coherence", "partial_coherence", "wpli", "directed_gc"),
  freq_band = NULL,
  channels = NULL,
  nperseg = 256L,
  noverlap = NULL,
  max_lag = 10L,
  debiased = FALSE,
  assay_name = NULL
)
```

## Arguments

**x**                      A `PhysioExperiment` holding a multi-channel EMG assay.

**methods**              Estimators to test (default: all four; see [coordinationNetwork\(\)](#)).

**freq\_band, channels, nperseg, noverlap, max\_lag, debiased, assay\_name**  
                          Passed to [coordinationNetwork\(\)](#) for each half.

## Value

A data frame (class `split_half_stability`), one row per method, with `method`, `split_half_edge_correlation`, `split_half_node_strength_correlation`, `first_half_mean_edge`, and `second_half_mean_edge`.

## See Also

[coordinationNetwork\(\)](#), [emgCoordinationStructure\(\)](#)

## Examples

```
pe <- make_emg(n_time = 2000, n_channels = 4, sr = 1000)
splitHalfStability(pe, methods = "coherence", freq_band = c(20, 150))
```

---

|                |                                    |
|----------------|------------------------------------|
| synergyCompare | <i>Compare Two Synergy Results</i> |
|----------------|------------------------------------|

---

**Description**

Computes pairwise correlation between synergy weight vectors from two decompositions. Uses best-match pairing.

**Usage**

```
synergyCompare(result1, result2)
```

**Arguments**

|         |                                                    |
|---------|----------------------------------------------------|
| result1 | First result from <a href="#">muscleSynergy</a> .  |
| result2 | Second result from <a href="#">muscleSynergy</a> . |

**Value**

A data.frame with columns: synergy1, synergy2, correlation.

**References**

De Luca, C.J. (1997). "The use of surface electromyography in biomechanics." *Journal of Applied Biomechanics*, 13(2), 135-163. doi:10.1123/jab.13.2.135

Merletti, R. & Parker, P.A. (2004). "Electromyography: Physiology, Engineering, and Non-Invasive Applications." Wiley-IEEE Press. doi:10.1002/0471678384

**See Also**

[muscleSynergy\(\)](#) for computing synergy decompositions, [synergyReconstruct\(\)](#) for reconstructing data from synergies

**Examples**

```
pe <- make_emg(n_time = 2000, n_channels = 4, sr = 1000)
s1 <- muscleSynergy(pe, n_synergies = 2, method = "nmf", seed = 1)
s2 <- muscleSynergy(pe, n_synergies = 2, method = "nmf", seed = 2)
synergyCompare(s1, s2)
```

---

|                    |                                        |
|--------------------|----------------------------------------|
| synergyReconstruct | <i>Reconstruct Data from Synergies</i> |
|--------------------|----------------------------------------|

---

## Description

Reconstructs EMG data using a subset of synergies.

## Usage

```
synergyReconstruct(synergy_result, n_synergies)
```

## Arguments

`synergy_result` Result from [muscleSynergy](#).  
`n_synergies` Number of synergies to use for reconstruction.

## Value

A list with:

- `reconstructed`: Reconstructed data matrix
- `vaf`: VAF of the reconstruction

## References

De Luca, C.J. (1997). "The use of surface electromyography in biomechanics." *Journal of Applied Biomechanics*, 13(2), 135-163. doi:10.1123/jab.13.2.135

## See Also

[muscleSynergy\(\)](#) for computing the initial decomposition, [synergyCompare\(\)](#) for comparing synergy solutions

## Examples

```
pe <- make_emg(n_time = 2000, n_channels = 4, sr = 1000)
syn <- muscleSynergy(pe, n_synergies = 3, method = "nmf", seed = 1)
rec <- synergyReconstruct(syn, n_synergies = 2)
rec$vaf
```

tuckerSynergy

*Non-negative Tucker (hierarchical) muscle-synergy decomposition***Description**

A hierarchical tensor synergy model: factorises the muscle x time x trial EMG tensor into non-negative spatial, temporal and trial factor matrices plus a CORE tensor that weights every combination of them. Unlike PARAFAC ([muscleSynergyTensor\(\)](#)), the core is not diagonal, so a spatial synergy can pair with several temporal synergies – capturing interactions PARAFAC cannot.

**Usage**

```
tuckerSynergy(
  trials,
  ranks,
  max_iter = 300L,
  tol = 1e-06,
  n_restart = 3L,
  seed = NULL
)
```

**Arguments**

|               |                                                                    |
|---------------|--------------------------------------------------------------------|
| trials        | A [muscle, time, trial] array or a list of muscle x time matrices. |
| ranks         | Length-3 factor ranks c(spatial, temporal, trial).                 |
| max_iter, tol | Iterations and convergence tolerance.                              |
| n_restart     | Random restarts (default 3).                                       |
| seed          | Optional RNG seed.                                                 |

**Value**

a tucker\_synergy: spatial (muscle x P), temporal (time x Q), trial (trial x R), core (P x Q x R), vaf, iterations.

**References**

Kim YD, Choi S (2007) non-negative Tucker; Tucker LR (1966).

**See Also**

[muscleSynergyTensor\(\)](#), [spaceByTimeSynergy\(\)](#)

**Examples**

```
set.seed(1)
A <- matrix(stats::runif(6*2),6,2); B <- matrix(stats::runif(40*2),40,2); C <- matrix(stats::runif(8*2),8,2)
G <- array(stats::runif(2*2*2), c(2,2,2))
X <- reconstructTucker(G, A, B, C)
tuckerSynergy(X, ranks = c(2, 2, 2), n_restart = 1)$vaf
```

# Index

convolutiveSynergy, 3  
coordinationNetwork, 4  
coordinationNetwork(), 56  
  
emgADLActivation, 5  
emgAmplitudeFeatures, 7  
emgAmplitudeFeatures(), 9  
emgAmplitudeNormalize, 8  
emgAmplitudeNormalize(), 6, 11, 19, 27  
emgCoContraction, 10  
emgCoContraction(), 5, 6, 47  
emgCoherenceNetwork, 12  
emgCoherenceNetwork(), 4, 5, 14, 17, 28, 34  
emgCoordinationStructure, 13  
emgCoordinationStructure(), 5, 56  
emgDimitrovIndex, 14  
emgDimitrovIndex(), 22, 25  
emgDirectedGCNetwork, 15  
emgDirectedGCNetwork(), 4  
emgDynamicWaveletNetwork, 16  
emgDynamicWaveletNetwork(), 12, 14  
emgEnvelope, 18  
emgEnvelope(), 8, 9, 11, 20, 21, 27, 30, 33, 36, 37, 48  
emgFatigue, 19  
emgFatigue(), 7–9, 14, 19–22, 25, 27, 33, 38, 49  
emgFatigueIndex, 20  
emgFatigueIndex(), 20, 22, 33, 38  
emgFatigueSlope, 21  
emgFatigueSlope(), 14, 25, 49  
emgInterpretNetworkKG, 22  
emgInterpretNetworkKG(), 12, 17  
emgMFCV, 23  
emgMFCV(), 54  
emgOnsetDetect, 25  
emgOnsetDetect(), 9, 11, 19, 37, 50  
emgPartialCoherenceNetwork, 27  
emgPartialCoherenceNetwork(), 4, 34  
emgQCgate, 28  
emgQCgate(), 30, 32  
emgQualityCheck, 29  
emgQualityCheck(), 28, 29, 32, 48  
emgRemoveECG, 31  
emgRemoveECG(), 29, 30  
emgSpectralMoments, 32  
emgSpectralMoments(), 7, 8, 14, 20, 21, 38  
emgWPLINetwork, 33  
emgWPLINetwork(), 4, 28  
  
getEMGElectrode, 34  
getEMGElectrode(), 53, 54  
  
hdEMGDecompose, 35  
  
make\_emg, 36  
make\_emg(), 37, 38  
make\_emg\_contraction, 37  
make\_emg\_contraction(), 36  
make\_emg\_fatigue, 38  
make\_emg\_fatigue(), 36, 37  
muscleSynergy, 39, 57, 58  
muscleSynergy(), 35, 43, 51, 55, 57, 58  
muscleSynergyMixture, 40  
muscleSynergyMixture(), 45, 46  
muscleSynergyOrder, 42  
muscleSynergyOrder(), 40, 51  
muscleSynergyTensor, 43  
muscleSynergyTensor(), 45, 59  
muscleSynergyTensorOrder, 45  
  
PhysioHDEMG::hdEMGDecompose(), 35  
physioSynergyMixture, 45  
plotCoactivation, 46  
plotEmgEnvelope, 48  
plotEmgFatigue, 49  
plotEmgOnset, 50  
plotSynergy, 51  
  
reconstructTucker, 52

seniamMuscles, [52](#)  
seniamMuscles(), [35](#), [53](#), [54](#)  
setEMGElectrode, [53](#)  
setEMGElectrode(), [34](#), [35](#), [53](#)  
spaceByTimeSynergy, [54](#)  
spaceByTimeSynergy(), [59](#)  
splitHalfStability, [56](#)  
splitHalfStability(), [4](#), [5](#)  
synergyCompare, [57](#)  
synergyCompare(), [40](#), [58](#)  
synergyReconstruct, [58](#)  
synergyReconstruct(), [40](#), [57](#)  
  
tuckerSynergy, [59](#)  
tuckerSynergy(), [55](#)