Package ‘musclesyneRgies’

July 19, 2022

Title Extract Muscle Synergies from Electromyography

Version 1.2.5

Description Provides a framework to factorise electromyography (EMG) data. Tools are provided for raw data pre-processing, non negative matrix factorisation, classification of factorised data and plotting of obtained outcomes. In particular, reading from ASCII files is supported, along with wide-used filtering approaches to process EMG data. All steps include one or more sensible defaults that aim at simplifying the workflow. Yet, all functions are largely tunable at need. Example data sets are included.

License MIT + file LICENSE

URL https://github.com/alesantuz/musclesyneRgies

BugReports https://github.com/alesantuz/musclesyneRgies/issues

Depends R (>= 4.1.0)

Imports FNN, ggplot2, graphics, grDevices, gridExtra, plyr, proxy, reshape2, signal, stats, umap

Suggests knitr, markdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

LazyData true

RoxygenNote 7.2.1

NeedsCompilation no

Author Alessandro Santuz [aut, cre] (<https://orcid.org/0000-0002-6577-5101>)

Maintainer Alessandro Santuz <alessandro.santuz@gmail.com>

Repository CRAN

Date/Publication 2022-07-19 17:10:02 UTC
classify_kmeans

**R topics documented:**

- classify_kmeans .................................................. 2
- CoA ................................................................. 4
- filtEMG ............................................................ 4
- FILT_EMG ........................................................... 6
- FWHM ................................................................. 7
- HFD ................................................................. 8
- Hurst ............................................................... 9
- normEMG ............................................................ 10
- plot_classified_syns ............................................. 12
- plot_classified_syns_UMAP ...................................... 13
- plot_meanEMG .................................................... 14
- plot_rawEMG ...................................................... 15
- plot_syn_trials .................................................... 16
- primitive .......................................................... 18
- primitives ........................................................ 18
- rawdata ........................................................... 19
- RAW_DATA .......................................................... 21
- sMLE .............................................................. 22
- subsetEMG ........................................................ 23
- SYNS .............................................................. 24
- synsNMF .......................................................... 25

**Index**

<table>
<thead>
<tr>
<th>classify_kmeans</th>
<th>Muscle synergy classification with k-means</th>
</tr>
</thead>
</table>

**Description**

Muscle synergy classification with k-means

**Usage**

```r
classify_kmeans(x, MSE_lim = 0.001, inspect = FALSE, show_plot = FALSE)
```

**Arguments**

- `x` A list of musclesyneRgies objects
- `MSE_lim` Mean squared error threshold for determining the minimum number of clusters
- `inspect` Logical, ask for interactive re-ordering or go fully automated?
- `show_plot` Logical, to decide whether plots should be plotted in the active graphic device
Details

This function must be applied to a list with a sufficient amount of trials, otherwise the classification will not work. Typically, at least 10 trials for the same condition are needed for satisfactory classification. If `show_plot` is TRUE (default) plots are also shown in the active graphic device. Plots can then be saved with the preferred export method, such as `ggplot2::ggsave`. The algorithm used is the default for `stats::kmeans` (Hartigan and Wong, 1979), which is known for its robustness to local minima. Nonetheless, the stochastic nature of the algorithm should prompt the user to attempt a few classifications and analyse their stability, before drawing conclusions on e.g. the number of fundamental synergies and/or their function. While the default parameters are optimised for human locomotion, it is suggested to test the function with different mean squared error thresholds, which is a crucial quantity to determine the number of clusters. Inspection and plotting are as well highly recommended to gain more insight into the classification process.

Value

List of `musclesynergies` objects, each with elements:

- `syns` factorisation rank or minimum number of synergies
- `M` motor modules (time-invariant coefficients)
- `P` motor primitives (time-dependent coefficients)
- `V` original data
- `Vr` reconstructed data
- `iterations` number of iterations to convergence
- `R2` quality of reconstruction (coefficient of determination)
- `rank_type` was the rank fixed or variable?
- `classification` classification type (k-means)

Examples

```r
# Load some data
data(SYNS)
# Classify synergies
SYNS_classified <- classify_kmeans(SYNS)
```
CoA

*Centre of activity*

**Description**

Centre of activity

**Usage**

`CoA(x)`

**Arguments**

- `x`: A time series (numeric)

**Value**

The centre of activity of the time series, calculated with circular statistics

**References**


**Examples**

```r
# Number of users connected to the Internet through a server every minute
ts <- datasets::WWWusage[1:80]

# Calculate CoA
ts_CoA <- CoA(ts)

# Plot
plot(ts, ty = "l", xlab = "Time", ylab = "Number of users")
graphics::abline(v = ts_CoA, lwd = 2, lty = 2)
```

**filtEMG**

*To filter raw EMG*

**Description**

To filter raw EMG
Usage

filtEMG(
  x,
  demean = TRUE,
  rectif = "fullwave",
  HPf = 50,
  HPo = 4,
  LPf = 20,
  LPo = 4,
  min_sub = TRUE,
  ampl_norm = TRUE
)

Arguments

  x Object of class EMG with elements cycles and emg
demean Logical: should EMG be demeaned?
rectif Rectification type: "fullwave", "halfwave" or "none"
HPf High-pass filter cut-off frequency, use 0 to exclude high-pass filtering
HPo High-pass filter order
LPf Low-pass filter cut-off frequency, use 0 to exclude Low-pass filtering
LPo Low-pass filter order
min_sub Logical: should the minimum be subtracted?
ampl_norm Logical: should amplitude be normalised?

Details

Lists in the correct format can be created with the function rawdata(). The first column of each emg element must be time in the same units as those used for cycles (e.g., [s] or [ms]).

Value

Object of class EMG with elements:

- cycles data frame containing cycle timings, with as many columns as many cycle subdivisions are wanted
- emg data frame containing filtered EMG data in columns, first column is time

References

Examples

```r
# Load some data
data("RAW_DATA")
# Filter raw EMG
filtered_EMG <- lapply(
  RAW_DATA,
  function(x) {
    filtEMG(x,
      HPf = 50,
      HPo = 4,
      LPf = 20,
      LPo = 4
    )
  }
)
```

FILT_EMG

Filtered EMG example

Description

A list containing filtered and time-normalised electromyographic (EMG) human data from the right-side lower limb recorded during one walking trial.

Usage

FILT_EMG

Format

A list containing one object of class EMG with elements `cycles` and `emg`, both data frames.

ID0012_TW_01 Object of class EMG containing the two following data frames:

cycles  Gait cycle-timings, in seconds.
emg     Filtered and time-normalised EMG, first column is time in points, muscles named as:

- ME=gluteus medius
- MA=gluteus maximus
- FL=tensor fasciae latae
- RF=rectus femoris
- VM=vastus medialis
- VL=vastus lateralis
- ST=semitendinosus
- BF=biceps femoris
- TA=tibialis anterior
- PL=peroneus longus
- GM=gastrocnemius medialis
- GL=gastrocnemius lateralis
- SO=soleus
FWHM

Full width at half maximum

Description

Full width at half maximum

Usage

FWHM(x, sub_minimum = TRUE)

Arguments

x  A time series (numeric)
sub_minimum  Logical; should the minimum be subtracted before amplitude normalisation?

Value

The full width at half maximum of the time series.

References


Examples

# Number of users connected to the Internet through a server every minute
ts <- datasets::WWWusage

# Calculate FWHM
ts_FWHM <- FWHM(ts)

# Half maximum (for the plots)
hm <- min(ts) + (max(ts) - min(ts)) / 2
hm_plot <- ts
hm_plot[which(hm_plot > hm)] <- hm
hm_plot[which(hm_plot < hm)] <- NA

# Plots
plot(ts, ty = "l", xlab = "Time", ylab = "Number of users")
lines(hm_plot, lwd = 3, col = 2)
HFD

Higuchi's fractal dimension

Description

Higuchi’s fractal dimension

Usage

HFD(P, k_max = 10)

Arguments

- P: A time series (numeric)
- k_max: Maximum window length in points

Details

The Higuchi’s fractal dimension is a measure of local complexity and it increases together with the “roughness” of the time series at a single cycle level (thus the term “local”). Higuchi’s fractal dimension values range from 1 to 2, with increasing values correlating to increasingly complex data and Higuchi’s fractal dimension = 1.5 indicating random Gaussian noise (Higuchi, 1988; Anmuth et al., 1994; Kesić & Spasić, 2016) For motor primitives, only the most linear part of the log-log plot should be used, as reported in Santuz, Akay (2020).

Value

A list with elements:

- \text{loglog} containing the log-log plot from which the HFD is calculated
- \text{Higuchi} containing the Higuchi’s fractal dimension of the time series.

References


Examples

# Measurements of the annual flow of the river Nile at Aswan
flow <- datasets::Nile

# Calculate HFD
fractal_dimension <- HFD(flow)$Higuchi
message("Higuchi's fractal dimension: ", round(fractal_dimension, 3))

# Thirty-cycle locomotor primitive from Santuz & Akay (2020)
data(primitive)
fractal_dimension <- HFD(primitive$signal)$Higuchi
message("Higuchi's fractal dimension: ", round(fractal_dimension, 3))

<table>
<thead>
<tr>
<th>Hurst</th>
<th>Hurst exponent</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Description

Hurst exponent

Usage

Hurst(P, min_win = 2)

Arguments

P  A time series (numeric)
min_win  Minimum window length in points

Details

Hurst calculates the Hurst exponent based on the R/S approach as in Hurst (1951). The Hurst exponent is a measure of global complexity and it increases if the “accuracy” of the time series decreases across several cycles (thus the term “global”). The Hurst exponent can vary between 0 and 1. For 0.5 < Hurst exponent < 1, in the long-term high values in the time series (the motor primitive in our case) will be probably followed by other high values and a positive or negative trend is visible (Mandelbrot, 1983; Gneiting & Schlather, 2004). For 0 < Hurst exponent < 0.5, in the long term high values in the series will be probably followed by low values, with a frequent switch between high and low values (Mandelbrot, 1983; Gneiting & Schlather, 2004). Hurst exponent = 0.5 corresponds to a completely random series (Mandelbrot, 1983; Qian & Rasheed, 2004). In other words, values of Hurst exponent approaching 0.5 from both ends indicate more complex (or random) behaviour of the time series (Hurst, 1951). For motor primitives, the minimum window length should be bigger than the period (i.e. the length of each cycle), as reported in Santuz, Akay (2020).
Value

A list with elements:

- `loglog` containing the log-log plot from which the HFD is calculated
- `Hurst` containing the Higuchi’s fractal dimension of the time series.

References


Examples

# Measurements of the annual flow of the river Nile at Aswan
flow <- datasets::Nile

# Calculate Hurst exponent
H <- Hurst(flow)$Hurst
message("Hurst exponent: ", round(H, 3))

# Thirty-cycle locomotor primitive from Santuz & Akay (2020)
data(primitive)
H <- Hurst(primitive$signal, min_win = max(primitive$time))$Hurst
message("Hurst exponent: ", round(H, 3))

---

**normEMG**

To time-normalise filtered EMG

Description

To time-normalise filtered EMG
Usage

```
normEMG(x, trim = TRUE, cy_max = NA, cycle_div = NA)
```

Arguments

- `x` Object of class `EMG` with elements `cycles` and `emg`
- `trim` Logical: should first and last cycle be trimmed to remove filtering effects?
- `cy_max` Maximum number of cycles to be considered
- `cycle_div` A vector or one dimensional array with the number of points each cycle should be normalised to

Details

Lists in the correct format can be created with the function `rawdata()`. The first column of each `emg` element must be time in the same units as those used for `cycles` (e.g., [s] or [ms]).

Value

Object of class `EMG` with elements:

- `cycles` data frame containing cycle timings, with as many columns as many cycle subdivisions are wanted
- `emg` data frame containing filtered and time-normalised EMG data in columns, first column is time

References


Examples

```
# Load some data
data("RAW_DATA")
# Filter raw EMG
filtered_EMG <- lapply(RAW_DATA, function(x) {
  filtEMG(x, HPf = 50, HPo = 4, LPf = 20, LPo = 4)
})
# Time-normalise filtered EMG, including three cycles and trimming first and last
filt_norm_EMG <- lapply(filtered_EMG, function(x) {
  normEMG(
    x, 
    cy_max = 3, 
    cycle_div = c(100, 100)
  )
})
```
Description

Plot muscle synergies

Usage

plot_classified_syns(
  x,
  dark_mode = FALSE,
  line_size = 0.9,
  dot_size = 0.1,
  line_col = "black",
  sd_col = "grey80",
  condition = NA,
  show_plot = TRUE
)

Arguments

x List of objects of class musclesyneRgies (must be classified)
dark_mode To enable dark mode
line_size Line thickness
dot_size Dot size on motor modules
line_col Line colour
sd_col Standard deviation ribbon colour
condition Character: the condition that is being analysed, for archiving purposes
show_plot Logical, to decide whether plots should be plotted in the active graphic device

Details

If show_plot is TRUE (default) plots are also shown in the active graphic device. Plots can then be saved with the preferred export method, such as ggplot2::ggsave.

Value

Global plot containing the average classified muscle synergies and individual trials (motor modules) or standard deviations (motor primitives)
Examples

```r
# Load some data
data(SYNS)

# Classify synergies with k-means
SYNS_classified <- classify_kmeans(SYNS)

# Save plot of classified synergies
pp <- plot_classified_syns(SYNS_classified,
dark_mode = TRUE,
line_col = "tomato1",
sd_col = "tomato4",
condition = "TW",
show_plot = FALSE
)
```

Description

Plot 2D UMAP of muscle synergies

Usage

```r
plot_classified_syns_UMAP(x, condition, show_plot = TRUE)
```

Arguments

- `x`: List of objects of class `musclesyneRgies` (must be classified)
- `condition`: Character: the condition that is being analysed, for archiving purposes
- `show_plot`: Logical, to decide whether plots should be plotted in the active graphic device

Details

If `show_plot` is `TRUE` (default) plots are also shown in the active graphic device. Plots can then be saved with the preferred export method, such as `ggplot2::ggsave`.

Value

2D UMAP plot of classified synergies.
Examples

```r
# Load some data
data(SYNS)

# Classify synergies with k-means
SYNS_classified <- classify_kmeans(SYNS)

# Save plot
pp <- plot_classified_syns_UMAP(SYNS_classified, condition = "TW", show_plot = FALSE)
```

---

**plot_meanEMG**

*Plot EMG averaged across all cycles*

**Description**

Plot EMG averaged across all cycles

**Usage**

```r
plot_meanEMG(
  x, 
  trial, 
  row_number = NA, 
  col_number = 1, 
  dark_mode = FALSE, 
  line_size = 0.6, 
  line_col = "black", 
  show_plot = TRUE
)
```

**Arguments**

- **x** A data frame containing filtered EMG organised in columns
- **trial** Character: the name of the considered trial, for archiving purposes
- **row_number** How many rows should the final plot be divided into?
- **col_number** How many columns should the final plot be divided into?
- **dark_mode** To enable dark mode
- **line_size** Line thickness
- **line_col** Line colour
- **show_plot** Logical, to decide whether plots should be plotted in the active graphic device
Details

If `show_plot` is TRUE (default) plots are also shown in the active graphic device. Plots can then be saved with the preferred export method, such as `ggplot2::ggsave`.

Value

Exports average filtered and normalised EMG.

Examples

```r
# Load some data
data(FILT_EMG)

# Save a plot of the only present trial with the average filtered and time-normalised EMG
pp <- plot_meanEMG(FILT_EMG[[1]],
                trial = names(FILT_EMG)[1],
                row_number = 4,
                col_number = 4,
                dark_mode = TRUE,
                line_col = "tomato3",
                show_plot = FALSE
)
```

plot_rawEMG

Plot raw EMG

Description

Plot raw EMG

Usage

```r
plot_rawEMG(
    x,
    trial,
    plot_time = 3,
    start = 1,
    row_number = NA,
    col_number = 1,
    dark_mode = FALSE,
    line_size = 0.3,
    line_col = "black",
    show_plot = TRUE
)
```
plot_syn_trials

Arguments

- **x** Object of class EMG with elements `cycles` and `emg`
- **trial** Character: the name of the considered trial, for archiving purposes
- **plot_time** How many seconds of data should be plotted?
- **start** At which data point should the plot start?
- **row_number** How many rows should the final plot be divided into?
- **col_number** How many columns should the final plot be divided into?
- **dark_mode** To enable dark mode
- **line_size** Line thickness
- **line_col** Line colour
- **show_plot** Logical, to decide whether plots should be plotted in the active graphic device

Details

If `show_plot` is TRUE (default) plots are also shown in the active graphic device. Plots can then be saved with the preferred export method, such as `ggplot2::ggsave`.

Value

Plots raw EMG trials of the specified length.

Examples

```r
# Load some data
data(RAW_DATA)

# Save a plot with the first (and only) trial in RAW_DATA, first three seconds, in dark mode
plot_rawEMG(RAW_DATA[[1]],
   trial = names(RAW_DATA)[1],
   row_number = 4,
   col_number = 4,
   dark_mode = TRUE,
   line_col = "tomato3",
   show_plot = FALSE
)
```

plot_syn_trials  
*Plot muscle synergies (individual trials)*

Description

Plot muscle synergies (individual trials)
Usage

plot_syn_trials(
  x,
  max_syns,
  trial,
  dark_mode = FALSE,
  line_size = 0.6,
  line_col = "black",
  sd_col = "grey80",
  show_plot = TRUE
)

Arguments

  x          Object of class musclesyneRgies
max_syns    Number of synergies to be plotted or how many rows should the final panel be divided into
trial       Character: the name of the considered trial, for archiving purposes
dark_mode   To enable dark mode
line_size   Line thickness
line_col    Line colour
sd_col      Standard deviation ribbon colour
show_plot   Logical, to decide whether plots should be plotted in the active graphic device

Details

If show_plot is TRUE (default) plots are also shown in the active graphic device. Plots can then be saved with the preferred export method, such as ggplot2::ggsave.

Value

Plots of the unclassified synergies, trial by trial.

Examples

# Load some data
data(SYNS)

# Find maximum number of synergies
max_syns <- max(unlist(lapply(SYNS, function(x) x$syns)))

# Save a plot with the first (and only, in this case) trial in the list
pp <- plot_syn_trials(SYNS[[1]],
  max_syns = max_syns,
  trial = names(SYNS)[1],
  dark_mode = TRUE,
  line_size = 0.8,
  line_col = "tomato1",
  ...)
primitives


ds_col = "tomato4",
show_plot = FALSE
)

---

<table>
<thead>
<tr>
<th>primitive</th>
<th>Single motor primitive example (30 cycles)</th>
</tr>
</thead>
</table>

**Description**

A data frame containing one motor primitive extracted from one wild type mouse walking on a treadmill.

**Usage**

```
primitive
```

**Format**

A data frame of two columns:

- **time** Normalised time in points.
- **signal** Motor primitive

**Source**

doi:10.1152/jn.00360.2020

---

<table>
<thead>
<tr>
<th>primitives</th>
<th>All motor primitives of one synergy example (30 cycles)</th>
</tr>
</thead>
</table>

**Description**

A demo and incomplete musclesyneRgies object containing time info and three motor primitives extracted from one wild type mouse walking on a treadmill.

**Usage**

```
primitives
```
**Format**

A data frame of four columns:

- **time**: Normalised time in points.
- **Syn1**: Motor primitive of synergy 1
- **Syn2**: Motor primitive of synergy 2
- **Syn3**: Motor primitive of synergy 3

**Source**

doi:10.1152/jn.00360.2020

---

**rawdata**

*Import RData or ASCII data into R*

---

**Description**

Import RData or ASCII data into R

**Usage**

rawdata(path_cycles = NA, path_emg = NA, header_cycles, header_emg = TRUE)

**Arguments**

- **path_cycles**: Optional, path where cycle timing files are located
- **path_emg**: Optional, path where raw EMG files are located
- **header_cycles**: Logical, are the cycle files containing a named header (the header is optional)?
- **header_emg**: Logical, are the raw EMG files containing a named header (they should)?

**Details**

Supported are R lists saved as RData files or tab- or comma-separated files readable through `read.table()` or `read.csv()`. The first column of each raw emg file must be time in the same units as those used for the cycle timings (e.g., [s] or [ms]). If reading from RData files, please call cycles `CYCLE_TIMES.RData` and raw EMG `RAW_EMG.RData`. Lists must be saved with `save()`. 
Value

List of objects of class EMG, each with elements:

- cycles data frame containing cycle timings, with as many columns as many cycle subdivisions are wanted
- emg data frame containing raw EMG data in columns, first column must be time in the same units as in the cycle timings

Examples

# Load built-in data set
data("RAW_DATA")

# Get current working directory
data_path <- getwd()
data_path <- paste0(data_path, .Platform$file.sep)

# Create two conveniently-named subfolders if they don't already exist
# (if they exist, please make sure they're empty!)
dir.create("cycles", showWarnings = FALSE)
dir.create("emg", showWarnings = FALSE)

# Export ASCII data from built-in data set to the new subfolders
write.table(RAW_DATA[[1]]$cycles,
            file = paste0(data_path, "cycles", .Platform$file.sep, names(RAW_DATA)[1], ".txt"),
            sep = "\t", row.names = FALSE, col.names = FALSE)
write.table(RAW_DATA[[1]]$emg,
            file = paste0(data_path, "emg", .Platform$file.sep, names(RAW_DATA)[1], ".txt"),
            sep = "\t", row.names = FALSE)

# Run the function to parse ASCII files into objects of class 'EMG'
raw_data_from_files <- rawdata(
    path_cycles = paste0(data_path, "/cycles"),
    path_emg = paste0(data_path, "/emg"),
    header_cycles = FALSE)

# Check data in the new folders if needed before running the following (will delete!)

# Delete folders
unlink("cycles", recursive = TRUE)
unlink("emg", recursive = TRUE)
Raw EMG example

Description

A list containing electromyographic (EMG) human data from the right-side lower limb recorded during one walking trial.

Usage

A list containing one object of class EMG with elements cycles and emg, both data frames.

ID0012_TW_01 Object of class EMG containing the two following data frames:

- cycles  Gait cycle-timings, in seconds.
- emg    Raw EMG, first column is time in seconds, muscles named as:

  - ME=gluteus medius
  - MA=gluteus maximus
  - FL=tensor fasciae latae
  - RF=rectus femoris
  - VM=vastus medialis
  - VL=vastus lateralis
  - ST=semitendinosus
  - BF=biceps femoris
  - TA=tibialis anterior
  - PL=peroneus longus
  - GM=gastrocnemius medialis
  - GL=gastrocnemius lateralis
  - SO=soleus

Source

**sMLE**

(*Short-term maximum Lyapunov exponents*)

**Description**

Short-term maximum Lyapunov exponents

**Usage**

`sMLE(synergies, mean_period, future_pts, norm, pts, R2_threshold = 0.9)`

**Arguments**

- **synergies**: A `musclesyneRgies` object
- **mean_period**: To locate the nearest neighbour of each point on the state space trajectory
- **future_pts**: To limit the number of points "in the future" that are being searched
- **norm**: Type of normalisation ("u" for minimum subtraction and normalisation to the maximum, "z" for subtracting the mean and then divide by the standard deviation)
- **pts**: Minimum number of points needed to linearly approximate the first part of the divergence curve
- **R2_threshold**: Threshold for calculating the slope of the divergence curve

**Details**

The mean period is intended to exclude temporally close points. In gait, values are usually plus/minus half gait cycle. Future points usually correspond in gait to one to two gait cycles. Please consider that a sufficient amount of cycles in order to compute meaningful sMLE. For locomotor primitives, 30 gait cycles have been shown to be sensitive to perturbations (Santuz et al. 2020). However, in the more classical and widespread use on kinematic data, more are usually needed (Kang and Dingwell, 2006).

**Value**

A list with elements:

- **divergences**: containing the average logarithmic divergence curve
- **sMLE**: the short-term Maximum Lyapunov exponent
- **R2**: the goodness of fit of the most linear part of the divergence curve
References


Examples

```r
# Load some primitives
data("primitives")
# Calculate sMLE of motor primitives in the muscle synergy space
short_term_MLE <- sMLE(primitives,
    mean_period = 80,
    future_pts = 200,
    norm = "z",
    pts = 30
)
```

subsetEMG

Subset raw EMG

Description

Subset raw EMG

Usage

```r
subsetEMG(x, cy_max, cy_start = 1)
```

Arguments

- `x` Objects of class EMG with elements cycles and emg
- `cy_max` Maximum number of cycles to be considered
- `cy_start` From which cycle should the subset begin?

Details

Lists in the correct format can be created with the function `rawdata()`. The first column of each emg element must be time in the same units as those used for cycles (e.g., [s] or [ms]). For locomotion, thirty cycles are enough for proper synergy extraction (Oliveira et al. 2014).
Value

Object of class EMG with elements:

- cycles data frame containing cycle timings, with as many columns as many cycle subdivisions are wanted
- emg data frame containing raw EMG data in columns, first column is time

References


Examples

# Load some data
data("RAW_DATA")
# Subset example raw data to the first 3 cycles
RAW_DATA_sub <- lapply(
  RAW_DATA,
  function(x) {
    subsetEMG(x,
      cy_max = 3,
      cy_start = 1
    )
  });

SYNS

Muscle synergies example

Description

A list created by synsNMF containing muscle synergies extracted from 15 humans walking on a treadmill.

Usage

SYNS

Format

A list containing 15 objects of class musclesyneRgies, each of which represents a walking trial from a different person.

ID0012_TW_01 ID0001_TW_01 ID0002_TW_01 ID0003_TW_01 ID0004_TW_01 ID0005_TW_01 ID0006_TW_01 ID0007_TW_01

Objects of class musclesyneRgies containing the following items:
synsNMF

**syns**  Factorisation rank or minimum number of synergies.

**M**  Motor modules (time-invariant coefficients)

**P**  Motor primitives (time-dependent coefficients)

**V**  Original data, muscles named as:

- ME=gluteus medius
- MA=gluteus maximus
- FL=tensor fasciae latae
- RF=rectus femoris
- VM=vastus medialis
- VL=vastus lateralis
- ST=semitendinosus
- BF=biceps femoris
- TA=tibialis anterior
- PL=peroneus longus
- GM=gastrocnemius medialis
- GL=gastrocnemius lateralis
- SO=soleus

**Vr**  Reconstructed data, muscles named as in V

**iterations**  Number of iterations to convergence

**R2**  Quality of reconstruction (coefficient of determination)

**classification**  Classification type (e.g., none, k-means, NMF, etc.)

**Source**


---

**synsNMF**  

Non-negative matrix factorisation

**Description**

Non-negative matrix factorisation

**Usage**

```r
synsNMF(
  V,
  R2_target = 0.01,
  runs = 5,
  max_iter = 1000,
  last_iter = 20,
  MSE_min = 1e-04,
  fixed_syns = NA
)
```
Arguments

- **V**  
  EMG data frame to be reconstructed, usually filtered and time-normalised

- **R2_target**  
  Threshold to stop iterations for a certain factorisation rank

- **runs**  
  Number of repetitions for each rank to avoid local minima

- **max_iter**  
  Maximum number of iterations allowed for each rank

- **last_iter**  
  How many of the last iterations should be checked before stopping?

- **MSE_min**  
  Threshold on the mean squared error to choose the factorisation rank or minimum number of synergies

- **fixed_syns**  
  To impose the factorisation rank or number of synergies

Details

The first column of V must always contain time information.

Value

Object of class musclesyneRgies with elements:

- **syns**  
  factorisation rank or minimum number of synergies

- **M**  
  motor modules (time-invariant coefficients)

- **P**  
  motor primitives (time-dependent coefficients)

- **V**  
  original data

- **Vr**  
  reconstructed data

- **iterations**  
  number of iterations to convergence

- **R2**  
  quality of reconstruction (coefficient of determination)

- **rank_type**  
  was the rank fixed or variable?

- **classification**  
  classification type (e.g., none, k-means, NMF, etc.)

References


Examples

# Note that for bigger data sets one might want to run computation in parallel
# Load some data
data(FILT_EMG)
# Extract synergies (careful, rank is imposed here!)
SYNS <- lapply(FILT_EMG, synsNMF, fixed_syns = 4)
Index

* datasets
  FILT_EMG, 6
  primitive, 18
  primitives, 18
  RAW_DATA, 21
  SYNS, 24

classify_kmeans, 2
CoA, 4

FILT_EMG, 6
filtEMG, 4
FWHM, 7

HFD, 8
Hurst, 9

normEMG, 10

plot_classified_syns, 12
plot_classified_syns_UMAP, 13
plot_meanEMG, 14
plot_rawEMG, 15
plot_syn_trials, 16
primitive, 18
primitives, 18

RAW_DATA, 21
rawdata, 19

sMLE, 22
subsetEMG, 23
SYNS, 24
synsNMF, 25