Package 'diegr'

September 23, 2025

Title Dynamic and Interactive EEG Graphics

Version 0.1.0

Description Allows to visualize high-density electroencephalography (HD-EEG) data through interactive plots and animations, enabling exploratory and communicative analysis of temporal-spatial brain signals. Funder: Masaryk University (Grant No. MUNI/A/1457/2023).

License MIT + file LICENSE

Date 2025-08-27

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 4.1.0)

Imports dplyr, rlang, ggplot2, gganimate, plotly, rgl, sp, scales, stats, purrr, tidyr

LazyData true

Suggests av, gifski, magick, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Zdeňka Geršlová [aut, cre] (ORCID:
 <https://orcid.org/0000-0003-1973-0479>),
 Stanislav Katina [rev] (ORCID: <https://orcid.org/0000-0002-3256-5482>,
 Reviewer and Supervisor),
 Martin Lamoš [ctb] (Provided anonymized data from the project AZV
 NU21-04-0044)

Maintainer Zdeňka Geršlová <gerslovaz@math.muni.cz>

Repository CRAN

Date/Publication 2025-09-23 10:40:02 UTC

Contents

---

animate_scalp   *3D scalp plot animation in time*

Description

3D scalp plot animation in time

Usage

```
animate_scalp(
  data,
  amplitude,
  mesh,
  tri,
  coords = NULL,
  template = NULL,
  col_range = NULL,
  col_scale = NULL,
  sec = 0.3,
  frames_dir = NULL,
```

```
    output_path = NULL,
    framerate = 3,
    cleanup = TRUE
)
```

Arguments

data	An input data frame or tibble with at least this required columns: time - the number of time point, sensor - the sensor label and the column with the EEG amplitude to plot specified in the argument amplitude.
amplitude	A character string naming the column with EEG amplitude values.
mesh	An object of class "mesh" (or a named list with the same structure) used for computing IM model. If not defined, the polygon point mesh with default settings from point_mesh function is used. See scalp_plot for details about the structure.
tri	A matrix with indices of the triangles. If missing, the triangulation is computed using make_triangulation function from D2 element of the mesh.
coords	Sensor coordinates as a tibble or data frame with named x, y and z columns of sensor coordinates and sensor column with sensor names. If not defined, the HCGSN256 template is used.
template	The kind of sensor template montage used. Currently the only available option is "HCGSN256" denoting the 256-channel HydroCel Geodesic Sensor Net v.1.0, which is also a default setting.
col_range	A vector with minimum and maximum value of the amplitude used in the colour palette for plotting. If not defined, the range of interpolated signal is used.
col_scale	Optionally, a colour scale to be utilised for plotting. If not defined, it is computed from col_range.
sec	The time interval used between individual animation frames, in seconds (default: 0.3).
frames_dir	Directory where the individual frames will be saved. If NULL, the video is only displayed in viewer and the frames are not saved.
output_path	Optional path to the output mp4 video file (".mp4" extension is required for correct rendering). If NULL, no video is created.
framerate	Number of frames per second for the output mp4 video (default: 3).
cleanup	Logical. Indicates, if all the PNG files should be deleted after encoding video. Default value is TRUE.

Details

Setting the parameter tri requires defining a mesh parameter. The parameter mesh should optimally be a "mesh" object (output from point_mesh function) or a list with the same structure (see point_mesh for more information). In that case, setting the argument tri is optional, and if it is absent, a triangulation based on the D2 element of the mesh is calculated and used in the plot. If the input mesh contains only 3D coordinates of a point mesh in D3 element, the use of previously created triangulation (through tri argument) is required.

Notes: For exporting the video, setting `frames_dir` together with `output_path` is required.

When specifying the `coords` and `template` at the same time, the `template` parameter takes precedence and the `coords` parameter is ignored.

Value

The output depends on the provided arguments:

- If `frames_dir` is specified, individual animation frames (PNG) are saved to that directory.
- If also `output_path` is specified, a video (MP4) is created and saved using the `av` package.
- Otherwise, the animation is displayed in an interactive rgl window.

See Also

scalp_plot

Examples

```
# This example may take a few seconds to render.
# Run only if you want to generate the full animation.
# Note: The example opens a rgl 3D viewer.
# Prepare a data structure:
s1e05 <- epochdata |> dplyr::filter(subject == 1 & epoch == 5 & time %in% c(10:20))
# Plot animation with default mesh and triangulation:
animate_scalp(s1e05, amplitude = "signal")
```

---

animate_topo   *Topographic map animation in time*

Description

Display a topographic animation of the change in amplitude over time. The function enables direct rendering in Rstudio Viewer or saving the animation in gif format to the chosen location.

Usage

```
animate_topo(
  data,
  amplitude,
  t_lim,
  FS = 250,
  t0 = 1,
  mesh,
  coords = NULL,
  template = NULL,
  col_range = NULL,
  col_scale = NULL,
```

```
    show_legend = TRUE,
    contour = FALSE,
    output_path = NULL,
    ...
)
```

Arguments

data	An input data frame or tibble with at least this required columns: `time` - the number of time point, `sensor` - the sensor label and the column with the EEG amplitude to plot specified in the argument `amplitude`.
amplitude	A character specifying the name of the column from input data with EEG amplitude values.
t_lim	A numeric vector of length 2 with limits of time points (i.e., the length of the timeline displayed below the animation).
FS	The sampling frequency. Default value is 250 Hz.
t0	Index of the zero time point, i.e. point, where 0 ms should be marked (most often time of the stimulus or time of the response).
mesh	A "mesh" object (or a named list with the same structure) containing at least D2 element with x and y coordinates of a point mesh used for computing IM model. If not defined, the point mesh with default settings from [point_mesh](#) function is used.
coords	Sensor coordinates as a tibble or data frame with named x, y columns of sensor coordinates and `sensor` column with sensor names. If not defined, the HCGSN256 template is used.
template	The kind of sensor template montage used. Currently the only available option is "HCGSN256" denoting the 256-channel HydroCel Geodesic Sensor Net v.1.0, which is also a default setting.
col_range	A vector with minimum and maximum value of the amplitude used in the colour palette for plotting. If not defined, the range of interpolated signal is used.
col_scale	Optionally, a colour scale to be utilised for plotting. If not defined, it is computed from `col_range`.
show_legend	Logical. Indicates, whether legend should be displayed beside the graph. Default value is TRUE.
contour	Logical. Indicates, whether contours should be plotted in the graph. Default value is FALSE.
output_path	File path where the animation will be saved using `gifski` renderer (optional). If not defined, the animation is plotted in the RStudio Viewer.
...	Additional parameters for animation according to [gganimate::animate](#).

Details

For more details about required mesh structure see [point_mesh](#) function. If the input mesh structure does not match this format, an error or incorrect function behavior may occur.

The time part of input data is assumed to be in numbers of time points, conversion to ms takes place inside the function for drawing the timeline labels. Due to the flexibility of the function (e.g. to mark and animate only a short section from the entire time course or to compare different data in the same time interval), it allows to enter and plot user-defined time ranges. If some values of the time are outside the `t_lim` range, the function writes a warning message - in that case the animation is still rendered, but the timeline will not match reality.

Note: When specifying the `coords` and `template` at the same time, the `template` parameter takes precedence and the `coords` parameter is ignored.

Value

If `output_path` is `NULL`, the function prints the animation to the RStudio Viewer. If `output_path` is specified, the animation is saved to the given file path and not displayed.

See Also

topo_plot

Examples

```
# This example may take a few seconds to render.
# Run only if you want to generate the full animation.
# Prepare a data structure:
s1e05 <- epochdata |> dplyr::filter(subject == 1 & epoch == 5 & time %in% c(10:20))
# Plot animation
# t0 = 10 indicates the time point of stimulus in epochdata,
# t_lim is the whole range of epochdata, we animate only a short period
animate_topo(s1e05, amplitude = "signal", t_lim = c(1,50), t0 = 10)
```

---

animate_topo_mean   *Animate EEG average topographic map with confidence bounds*

Description

An animation of the average signal time course as a topographic map along with the lower and upper bounds of the confidence interval. In the output, three facets are plotted per frame: CI lower, average, CI upper.

Usage

```
animate_topo_mean(
  data,
  t_lim,
  FS = 250,
  t0 = 1,
  mesh,
```

```
    coords = NULL,
    template = NULL,
    col_range = NULL,
    col_scale = NULL,
    show_legend = TRUE,
    contour = FALSE,
    output_path = NULL,
    ...
)
```

Arguments

data	A data frame, tibble or a database table with input data to plot. It should be an output from [compute_mean](#) function or an object with the same structure, containing columns: `sensor` with sensor labels and `average`, `ci_low`, `ci_up` with values of average signal and lower and upper CI bounds.
t_lim	Limits of time points (i.e., the length of the timeline displayed below the animation).
FS	The sampling frequency. Default value is 250 Hz.
t0	Index of the zero time point, i.e. point, where 0 ms should be marked (most often time of the stimulus or time of the response).
mesh	A ″mesh″ object (or a named list with the same structure) containing at least D2 element with x and y coordinates of a point mesh used for computing IM model. If not defined, the point mesh with default settings from [point_mesh](#) function is used.
coords	Sensor coordinates as a tibble or data frame with named x, y columns of sensor coordinates and `sensor` column with sensor names. If not defined, the HCGSN256 template is used.
template	The kind of sensor template montage used. Currently the only available option is ″HCGSN256″ denoting the 256-channel HydroCel Geodesic Sensor Net v.1.0, which is also a default setting.
col_range	A vector with minimum and maximum value of the amplitude used in the colour palette for plotting. If not defined, the range of the input signal is used.
col_scale	Optionally, a colour scale to be utilised for plotting. If not defined, it is computed from `col_range`.
show_legend	Logical. Indicates, whether legend should be displayed below the graph. Default value is TRUE.
contour	Logical. Indicates, whether contours should be plotted in the graph. Default value is FALSE.
output_path	File path where the animation will be saved using gifski renderer (optional). If not defined, the animation is plotted in the RStudio Viewer.
...	Additional parameters for animation according to [gganimate::animate](#).

Details

Note: When specifying the `coords` and `template` at the same time, the `template` parameter takes precedence and the `coords` parameter is ignored.

Value

If `output_path` is NULL, the function prints the animation to the RStudio Viewer. If `output_path` is specified, the animation is saved to the given file path and not displayed. The `gifski` and `magick` packages are required for animation export.

See Also

[animate_topo](#), [compute_mean](#), [baseline_correction](#)

Examples

```
# This example may take a few seconds to render.
# Run only if you want to generate the full animation.

# a) prepare data: compute the mean from baseline corrected signal for subject 2,
# first 10 points and only 13 epochs (epochs 14 and 15 are outliers)
edata <- epochdata |>
dplyr::filter(subject == 2 & time %in% 1:10 & epoch %in% 1:13)
data_base <- baseline_correction(edata, baseline_range = 1:10) # baseline correction
data_mean <- compute_mean(data_base, amplitude = "signal_base", subject = 2,
 type = "jack", group = "space") # compute mean
# b) render the animation
# (t0 = 10 because the time of the stimulus in epochdata is in time point 10)
animate_topo_mean(data_mean, t_lim = c(1,50), t0 = 10)
```

---

baseline_correction   *Baseline correction*

Description

Function for computing amplitude corrected to the selected baseline.

Usage

```
baseline_correction(data, baseline_range, type = "absolute")
```

Arguments

data	A data frame, tibble or a database table with input data, required columns: subject, sensor, time, signal, epoch.
baseline_range	A numeric vector of time points used as the baseline (e.g., `baseline_range = 125:250`).
type	A character indicating the type of baseline correction. Only the value `"absolute"` is allowed at this moment (other values will trigger an error).

Details

If the values from `baseline_range` vector are out of the range of the `time` column, the baseline calculation proceeds as follows:

1. If a part of the `baseline_range` vector is in the `time` column and part is outside its range, the baseline correction is computed only from the part inside a `time` range.

2. If the whole `baseline_range` vector is out of the `time` range, the `baseline` and also the `signal_base` values of the output are NA's. In both cases the function returns a warning message along with the output data frame or tibble.

Note: If there are NA values in the `signal` column, matching rows are ignored in the baseline calculation (which may bias the results) and the function prints a warning message.

Value

A data frame/tibble with added columns:

`signal_base`	Signal corrected by subtracting the baseline for each epoch.
`baseline`	A baseline value used for correction.

Examples

```
# Computing baseline correction on first 10 points, sensor "E1"
# a) Prepare data and compute
data01 <- epochdata |> dplyr::filter(.data$subject == 1 & .data$sensor == "E1")
basedata <- baseline_correction(data01, baseline_range = 1:10, type = "absolute")

# b) Plot raw (black line) and corrected (red line) signal for epoch 1
epoch1 <- basedata |> dplyr::filter(.data$epoch == 1)
plot(epoch1$signal, type = "l", ylim = c(-20, 30), main = "Raw (black) vs Corrected (red) Signal",
xlab = "time point", ylab = "amplitude")
lines(epoch1$signal_base, col = "red")


# Set baseline_range outside of time range
# results in NA's in baseline and signal_base columns
# also returns a warning message
basedata <- baseline_correction(data01, baseline_range = 70:80, type = "absolute")
head(basedata)
```

---

boxplot_epoch	*Plot interactive boxplots of epochs*

Description

Function for plotting interactive boxplots of EEG amplitude in individual epochs for selected subject and channel within the chosen time interval. The interactive `plotly` output enables to easily determine the epoch number from which outliers come and also allows to easily edit the image layout.

Usage

```
boxplot_epoch(
  data,
  amplitude = "signal",
  subject,
  channel,
  time_lim,
  use_latex = TRUE
)
```

Arguments

`data`	A data frame or a database table with EEG dataset. Required columns: subject, sensor, epoch, time and the column with EEG amplitude named as in `amplitude` parameter.
`amplitude`	A character specifying the name of the column from input data with an EEG amplitude values. Default is `"signal"`.
`subject`	An integer or character ID of selected subject to plot.
`channel`	An integer or character ID of channel to plot.
`time_lim`	A numeric vector with time range to plot.
`use_latex`	A logical value indicating whether to use LaTeX formatting for the y-axis title. The default is `TRUE`.

Details

The input data frame or database table must contain at least following columns: subject - a column with subject IDs, sensor - a column with sensor labels, epoch - a column with epoch numbers, time - a column with time point numbers, and a column with measured EEG signal values (or their averages) called as in `amplitude`.

Value

A `plotly` object with boxplots of EEG amplitude for individual epochs.

Examples

```
# Interactive boxplots of signal from channel E34 for subject 1 (health control)
# in chosen time points
boxplot_epoch(epochdata, amplitude = "signal", subject = 1, channel = "E34",
 time_lim = c(10:20))
```

---

boxplot_rt	*Plot interactive boxplots of response times*

Description

Function for plotting interactive boxplots of response time in individual epochs for selected subjects. The interactive plotly output enables to easily determine the epoch number from which outliers come and also allows to easily edit the image layout.

Usage

```
boxplot_rt(data, subject = NULL)
```

Arguments

data	A data frame or a database table with response times dataset. Required columns: subject, epoch, RT (value of response time in ms).
subject	A vector with IDs of subjects to plot. If missing, boxplots are drawn for all available subjects in data.

Value

A plotly object with boxplots of response times.

Examples

```
# Display interactive boxplots for both example subjects
boxplot_rt(rtdata)
```

---

boxplot_subject	*Plot interactive boxplots of subjects*

Description

Function for plotting interactive boxplots of EEG amplitude for individual subjects for selected channel within the chosen time interval. The interactive plotly output enables to easily determine the subjects with outlier amplitude and also allows to easily edit the image layout.

Usage

```
boxplot_subject(
  data,
  amplitude = "signal",
  subject = NULL,
  channel,
  time_lim,
  use_latex = TRUE
)
```

Arguments

`data`	A data frame or a database table with EEG dataset. Required columns: subject, sensor, time and the column with EEG amplitude named as in `amplitude` parameter.
`amplitude`	A character specifying the name of the column from input data with an EEG amplitude values. Default is `"signal"`.
`subject`	A vector with IDs of subjects to plot. If missing, boxplots are drawn for all avaliable subjects in `data`.
`channel`	An integer or character ID of channel to plot.
`time_lim`	A numeric vector with time range to plot.
`use_latex`	A logical value indicating whether to use LaTeX formatting for the y-axis title. The default is `TRUE`.

Details

The input data frame or database table must contain at least following columns: subject - a column with subject IDs, sensor - a column with sensor labels, time - a column with time point numbers, and a column with measured EEG signal values (or their averages) called as in `amplitude`.

Value

A `plotly` object with boxplots of EEG amplitude for subjects.

Examples

```
# Interactive boxplots of signal from channel E34 in epoch 1
# for both subjects in chosen time points
## Note: it has no statistical sense to make boxplot from only 2 observations, but
## larger example dataset is not possible due to size limit of the package
epoch1 <- epochdata|> dplyr::filter(epoch == 1)
boxplot_subject(epoch1, amplitude = "signal", channel = "E34", time_lim = c(10:20))
```

---

compute_mean   *Calculate mean in temporal or spatial domain*

Description

Function calculates a pointwise or a jackknife (leave-one-out) average signal for chosen subject (or more subjects) in temporal or spatial domain together with bootstrap or jackknife pointwise confidence interval (CI) bounds. The function computes an average at subject, sensor/time point or epoch level (according to the `level` parameter).

Usage

```
compute_mean(
  data,
  amplitude = "signal_base",
  subject = NULL,
  channel = NULL,
  time = NULL,
  ex_epoch = NULL,
  group = "time",
  level = "epoch",
  type = "point",
  R = NULL,
  alpha = 0.95
)
```

Arguments

data	A data frame, tibble or a database table with input data, required columns: subject, sensor, epoch (only for epoch level), time and the column with the EEG amplitude specified in the argument `amplitude`.
amplitude	A character specifying the name of the column from input data with an EEG amplitude values. Default is `"signal_base"` for computing average from baseline corrected signal.
subject	A vector of subject indices to be included in the average calculation. If NULL, all subjects present in input data are included.
channel	A character vector specifying the channels to be averaged. If NULL, all channels present in input data are included.
time	A numeric vector specifying the time points used for computing the average signal. If NULL, the whole time interval present in the input data is included.
ex_epoch	Optional. A vector of epoch indices to be excluded from the average calculation.
group	A character specifying the grouping factor. Default is `"time"` for calculation of the average at individual time points, other possibility is `"space"` for calculation of the average at individual space points (sensors).
level	A character specifying the level of average calculation. The possible values are `"epoch"`, `"sensor"` and `"subject"`. See details for more information.
type	A character specifying the method of calculating the average, `"point"` for pointwise arithmetic average (default) and `"jack"` for jackknife leave-one-out average.
R	The number of replications used in bootstrap interval calculation. Required only for computing pointwise mean. Default value is 1000.
alpha	A number indicating confidence interval level. The default value is 0.95 for 95% confidence intervals.

Details

The `level` parameter enables to choose the level at which the average is calculated:

- `"epoch"` means averaging on epoch level, the result is the average curve (from all included epochs) for individual sensors and subjects in the `group = "time"` case or the 2D sensor array average (from all included epochs) for individual time points and subjects in the `group = "space"` case.

- `"sensor"` means averaging on sensor or time point level, the result is the average curve from all included sensors for individual subjects in the `group = "time"` case or the average sensor array from all time points for individual subjects in the `group = "space"` case.

- `"subject"` means averaging on subject level, the result is the average curve or sensor array from all included subjects. The function assumes input adapted to the desired level of averaging (i.e. for epoch level the epoch column must be present etc.).

Computing standard error of the mean:

- for `type = "point"`, the standard error is computed as sample standard deviation divided by square root of the sample size

- for `type = "jack"`, the standard error is jackknife standard error of the mean (for the exact formula see Efron and Tibshirani 1994)

Computing point confidence intervals: For each average value, the upper and lower bounds of the point confidence interval are also available.

- Setting `type = "point"` and `R`: the bounds are computed using percentile method from bootstrapping with `R` replicates (can be slow for large amounts of data).

- Setting `type = "point"` without specifying `R`: the bounds are computed using standard error of the mean and approximation by the Student distribution.

- Setting `type = "jack"`: the bounds are computed using jackknife standard error of the mean and approximation by the Student t-distribution. Note: used method assumes equal variance and symmetric distribution, which may be problematic for very small samples.

Note: If there are NA's in `amplitude` column, corresponding rows are ignored in the average calculation and function prints a warning message.

Value

A tibble with resulting average and CI bounds according to the chosen `level`, `group` and `alpha` arguments. The statistics are saved in columns

- `average` for computed average amplitude value,

- `se` for standard error of the mean,

- `ci_low` for lower bound of the confidence interval and

- `ci_up` for upper bound of the confidence interval.

References

Efron B., Tibshirani RJ. *An Introduction to the Bootstrap.* Chapman & Hall/CRC; 1994.

Examples

```
# Average (pointwise) raw signal for subject 1 and electrode E1
# without outlier epoch 14
avg_data <- compute_mean(epochdata, amplitude = "signal", subject = 1, channel = "E1",
level = "epoch", ex_epoch = 14)
str(avg_data)

# plot the result using interactive plot with pointwise CI
interactive_waveforms(data = avg_data, amplitude = "average", subject = 1, t0 = 10,
level = "sensor", avg = FALSE, CI = TRUE)



# Jackknife average signal for subject 1 in all electrodes in time point 11 with baseline correction
# on interval 1:10 (again without outlier epoch 14)
# a) prepare corrected data
data01 <- epochdata |> dplyr::filter(.data$subject == 1)
basedata <- baseline_correction(data01, baseline_range = 1:10, type = "absolute")
# b) compute the average in time point 11
avg_data <- compute_mean(basedata, amplitude = "signal_base", time = 11, level = "epoch",
 group = "space", type = "jack", ex_epoch = 14)
str(avg_data)
# c) plot the result with topo_plot()
topo_plot(data = avg_data, amplitude = "average")

# Space average on subject level (average for all included subjects in time point 11)
# a) filter time point 11
data02 <- epochdata |> dplyr::filter(.data$time == 11)
# b) compute mean from all epochs for each subject
mean_epoch <- compute_mean(data02, amplitude = "signal", time = 11, level = "epoch",
group = "space", type = "point", ex_epoch = c(14,15))
# c) compute mean on subject level
mean_subjects <- compute_mean(mean_epoch, amplitude = "average", level = "subject",
group = "space", type = "point")
head(mean_subjects)
```

---

create_scale   *Create colour scale used in topographic figures*

Description

Create colour scale used in topographic figures

Usage

```
create_scale(col_range, k = NULL)
```

Arguments

col_range   A numeric vector with required range of the variable to be plotted in the colour
                   scale.

k   A number from interval (0,1) indicating a sequence step for the colour palette.
                   The smaller number, the finer division of the data range interval. See Details for
                   more information about auto-computing if NULL.

Details

The palette is created according to topographical colours: negative values correspond to shades
of blue and purple and positive values to shades of green, yellow and red. The zero value of the
variable is always at the border of blue and green shades. To compare results for different subjects
or conditions, set the same col_range for all cases. Otherwise, the colours are assigned separately
in each plot and are not consistent with each other.

The parameter k is set by default with respect to the range of col_range as follows:

- k = 0.1 for range ≤ 30,

- k = 0.03 for range ≥ 70,

- k = 0.04 otherwise.

Value

A list with two components:

colors   A vector with hexadecimal codes of palette colours.

breaks   A vector with breaks for cutting the data range.

The list is intended for use in [scale_fill_gradientn](#) or similar plotting calls.

Examples

```
# Create scale on interval (-10,10) with default step number
create_scale(col_range = c(-10,10))

# Create scale on interval c(-5,10) with small k (finer division)
create_scale(col_range = c(-5, 10), k = 0.02)
```

---

epochdata   *Example high-density (HD-EEG) epoched data*

Description

This dataset is a short slice of a HD-EEG dataset from a study investigating the impact of deep brain stimulation on patients with advanced Parkinson's disease (Madetko-Alster, 2025). During the experiment subjects performed a simple visual motor task (pressing the response button in case of target visual stimulus presentation). The data was measured by 256-channel HydroCel Geodesic Sensor Net and sampling frequency is 250 Hz. The study was carried out by Central European Institute of Technology in Brno and was supported by Czech Health Research Council AZV NU21-04-00445.

Example dataset contains amplitude values measured on chosen 204 channels in 50 time points (with the stimulus in the time point 10) for 2 representative subjects (one patient and one healthy control subject). From the total number of 50 epochs for each subject, 14 (or 15) epochs were selected for the sample dataset. This data is intended for testing EEG preprocessing and visualization methods.

Usage

```
data("epochdata")
```

Format

The data frame consist of 295 800 rows (50 time points x 204 sensors x 29 epochs) and five columns:

time Number of time point. Time point 10 corresponds to stimulus onset (0 ms) and the interval between two time points corresponds to the time period 4 ms.

signal HD-EEG signal amplitude, in microvolts.

epoch Factor variable with epoch number, 14 epochs for subject 1, 15 epochs for subject 2.

sensor Sensor label, according to labeling used in the EGI Geodesic Sensor Net Technical Manual.

subject Factor variable with subject ID, 1 - representative healthy control subject, 2 - representative patient subject.

Source

Central European Institute of Technology, Masaryk University, Brno, Czech Republic.

References

EGI Geodesic Sensor Net Technical Manual (2024) https://www.egi.com/knowledge-center

Madetko-Alster N., Alster P., Lamoš M., Šmahovská L., Boušek T., Rektor I. and Bočková M. The role of the somatosensory cortex in self-paced movement impairment in Parkinson's disease. Clinical Neurophysiology. 2025, vol. 171, 11-17.

Examples

```
# Data preview
head(epochdata)
```

HCGSN256   *Coordinates of 256-channel HCGSN sensors*

Description

A file containing the Cartesian coordinates of high-density EEG sensor positions in 3D space on the scalp surface and their positions in 2D space. The coordinates belong to 256-channel Hydro-Cel Geodesic Sensor Net (GSN) average template montage. This template contains 257 electrode positions (including reference).

Usage

```
data("HCGSN256")
```

Format

A list with following elements:

D2 A tibble with 3 columns containing x and y coordinates and sensor labels (according to EGI GSN Technical Manual) in 2D.

D3 A tibble with 4 columns containing x, y and z coordinates and sensor labels in 3D. See 'Details' for more information.

ROI Factor containing the name of the region to which the corresponding sensor belongs. The levels are: "central", "frontal", "occipital", "parietal", "temporal" and "face" for electrodes from the face area.

Details

The axis orientation in the 3D case is as follows:

- x-axis: left (-) to right (+),
- y-axis: posterior (-) to anterior (+),
- z-axis: inferior (-) to superior (+). The reference electrode (Cz) is fixed at point $(0, 0, Z)$, where Z is the positive height of Cz. The nasion is fixed at $(0, Y, Z)$. Since both the nasion and Cz are always fixed at $x = 0$, they are assumed to be in the same y plane. The origin is the center of the head, defined as the center of a sphere fit to fiducial points above the plane made up of the Left Preauricular Point (LPA), the Right Preauricular Point (RPA), and the nasion.

The 2D coordinates were created by EGI team by positioning the channels to maximize use of screen space and to preserve the head shape as much as possible.

The regions in ROI were determined by an expert from Central European Institute of Technology, Masaryk University, Brno, Czech Republic.

Source

Central European Institute of Technology, Masaryk University, Brno, Czech Republic.

References

EGI Geodesic Sensor Net Technical Manual (2024), <https://www.egi.com/knowledge-center>

Examples

```
# A simple plot of sensor coordinates from HCGSN256 template as points in 2D
plot(HCGSN256$D2[,1:2], pch = 16, asp = 1)
```

---

interactive_waveforms   *Plot interactive waveform graph*

Description

Function for plotting time series of EEG signal colour-coded by epoch, channel or subject (depending on selected level) in an interactive plotly graph. When using the function for plotting the average, there is an option to add a confidence band using the CI argument. The output in plotly format enables to easily edit the image layout.

Usage

```
interactive_waveforms(
  data,
  amplitude = "signal",
  subject = NULL,
  channel = NULL,
  FS = 250,
  t0 = NULL,
  col_palette,
  level = "epoch",
  avg = TRUE,
  CI = FALSE,
  use_latex = TRUE
)
```

Arguments

data	A data frame, tibble or a database table with input data containing a time column and columns corresponding to the selected level and amplitude parameter (see Details).
amplitude	A character specifying the name of the column from input data with an EEG amplitude values. Default is "signal".
subject	A subject(s) chosen to plot.
channel	A channel(s) to plot.
FS	The sampling frequency. Default value is 250 Hz.

t0   Index of the zero time point, i.e. point, where 0 ms should be marked (most
                often time of the stimulus or time of the response).

col_palette   Optionally, a colour palette for plotting lines. If missing, the rainbow palette is
                used. The expected length is the same (or higher) as the number of unique levels
                (e.g. number of epochs for level = "epoch").

level   A character specifying the level of the time curves. The possible values are
                "epoch" (default option), "sensor" and "subject". See details for more infor-
                mation.

avg   A logical value indicating, if the average black curve should be plotted. Default
                is TRUE.

CI   A logical value indicating, if the confidence ribbon should be plotted. Default is
                FALSE. See Details for more information.

use_latex   A logical value indicating whether to use LaTeX formatting for the y-axis title.
                The default is TRUE.

Details

The input data frame or database table must contain at least some of the following columns (accord-
ing to the level parameter - for "sensor" level no epoch column is needed etc.): subject - a column
with subject IDs, sensor - a column with sensor labels, time - a column with time point numbers,
epoch - a column with epoch numbers, signal (or other name specified in amplitude parameter)
- a column with measured EEG signal values. Note: The average signals must be pre-aggregated
before plotting at higher grouping levels, for example sensor level assumes a mean sensor signal in
the amplitude column (the input data for individual epochs together with sensor level setting will
result in a mess output).

Plotting confidence ribbon: To plot the confidence bands around the average lines (CI = TRUE), the
input data must include the ci_up and ci_low columns (as in the output tibble from compute_mean
function).

Value

A plotly object showing an interactive time series of the signal according to the chosen level.

Examples

```
# 1) Plot epoch waveforms with average curve for subject 1 and electrode "E65"
# with 250 sampling frequency rate (default) and 10 as zero time point
interactive_waveforms(epochdata, amplitude = "signal", subject = 1, channel = "E65",
t0 = 10, level = "epoch")
# 2) Plot sensor level waveforms with confidence bands for subject 1 and electrodes "E65" and "E182"
# a) preparing data
sendata <- compute_mean(epochdata, amplitude = "signal", subject = 1, channel = c("E65", "E182"),
 group = "time", level = "epoch")
# b) plot the waveforms without the average
interactive_waveforms(sendata, amplitude = "average", subject = 1, t0 = 10,
level = "sensor", avg = FALSE, CI = TRUE)
```

---

make_triangulation *Make triangulation of 2D point mesh*

Description

Function for creating Delaunay type-I triangulation (see Schumaker 2007) with consistent oriented edges adapted for a regular point mesh created by `point_mesh` function. See Details for more information.

Usage

```
make_triangulation(mesh)
```

Arguments

mesh   A data frame or tibble with named columns: `x`, `y` (required) and `index` (optionally, if missing, it will be generated internally). It should optimally be a `D2` element of a `"mesh"` object or a list with the same structure of uniformly spaced grid.

Details

The type-I Delaunay triangulation is a triangulation obtained by drawing in the north-east diagonals in all subrectangles of the triangulated area. Due to the regularity of the input mesh (in the sense of distances between mesh points), a simplified procedure is used: The triangulation is created within the individual strips and then bound together. The order of the vertices is chosen to maintain a consistent orientation of the triangles (for more details see Schneider 2003).

If the input mesh has not regular grid spacing, the result triangulation may not be meaningful and will not meet the Delaunay triangulation criteria.

Value

A three column matrix with indices of the vertices of the triangles. Each row represents one triangle, defined by three vertex indices pointing to rows in the input mesh.

References

Lai M-J, Schumaker LL. *Spline functions on triangulations.* Cambridge University Press; 2007.

Schneider PJ, Eberly DH. *Geometric Tools for Computer Graphics.* The Morgan Kaufmann Series in Computer Graphics. San Francisco: Morgan Kaufmann, 2003.

Examples

```
# a) Create small mesh for triangulation example
# using 204 electrodes from epochdata
M <- point_mesh(n = 500, template = "HCGSN256",
sensor_select = unique(epochdata$sensor))
```

```
# b) Make triangulation on this mesh
TRI <- make_triangulation(M$D2)
head(TRI)

# c) plot triangulation in 2D
# prepare empty plot
plot(M$D2, type = "n", main = "Triangulation plot",
xlab = "", ylab = "", asp = 1, axes = FALSE)
# create a structure for plotting
x0 <- c()
y0 <- c()
x1 <- c()
y1 <- c()
for (i in 1:nrow(TRI)) {
  v_indices <- TRI[i, ]
  v_coords <- M$D2[v_indices, ]
  x0 <- c(x0, v_coords[1, "x"], v_coords[2, "x"], v_coords[3, "x"])
  y0 <- c(y0, v_coords[1, "y"], v_coords[2, "y"], v_coords[3, "y"])
  x1 <- c(x1, v_coords[2, "x"], v_coords[3, "x"], v_coords[1, "x"])
  y1 <- c(y1, v_coords[2, "y"], v_coords[3, "y"], v_coords[1, "y"])
 }
# plot triangulation using segments
segments(x0, y0, x1, y1)

## Note: this code opens a rgl 3D viewer
# d) Plot the result triangulation as 3D wire model using rgl
 rgl::open3d()
 rgl::wire3d(rgl::mesh3d(M$D3$x, M$D3$y, M$D3$z, triangles = t(TRI)))
```

---

outliers_epoch   *Select outlier epochs*

Description

Function for selecting outlier epochs for one subject and one sensor in chosen time points. Epochs are marked as outliers based on one of the following criteria: interquartile range criterion, percentile approach or Hampel filter method.

Usage

```
outliers_epoch(
  data,
  amplitude = "signal",
  subject = NULL,
  sensor = NULL,
  time = NULL,
  method,
  k_iqr = 1.5,
```

```
    k_mad = 3,
    p = 0.975,
    print_tab = TRUE
)
```

Arguments

`data`	A data frame, tibble or a database table with input data, required columns: subject, sensor, time, epoch and the column with EEG amplitude named as in `amplitude` parameter.
`amplitude`	A character specifying the name of the column from input data with an EEG amplitude values. Default is `"signal"`.
`subject`	An integer or character ID of selected subject.
`sensor`	An integer or character ID of selected sensor.
`time`	A vector with time range for outliers detection. If not defined, the outliers are searched for all time points in the dataset.
`method`	A character denoting the method used for outlier detection. The options are: `"iqr"` for interquartile range (IQR) criterion, `"percentile"` for percentile method and `"hampel"` for Hampel filter method. See details for further information about methods.
`k_iqr`	A positive numeric value denoting the scaling factor used in the IQR criterion. Default value is `k_iqr = 1.5`.
`k_mad`	A positive numeric value denoting the scaling factor used in the Hampel filter method. Default value is `k_mad = 3`.
`p`	A probability value from `[0,1]` interval determining percentile to the percentile method (according to `probs` argument in `quantile()` function). The default value is set to 0.975 for the interval formed by the 2.5 and 97.5 percentiles.
`print_tab`	Logical. Indicates, whether result table should be printed in console. Default is `TRUE`.

Details

The input data frame or database table must contain at least following columns: subject - a column with subject IDs, sensor - a column with sensor labels, time - a column with time point numbers, signal (or other name specified in `amplitude` parameter) - a column with measured EEG signal values.

The outlier detection method is chosen through `method` argument. The possibilities are

- `iqr`: The interquartile range criterion, values outside the interval `[lower quartile – k_iqr * IQR, upper quartile + k_iqr * IQR]` are considered as outliers. IQR denotes interquartile range and `k_iqr` the scaling factor.

- `percentile`: The percentile method, values outside the interval defined by the chosen percentiles are considered as outliers. Note: chosing small p leads to marking too many (or all) values.

- hampel: The Hampel filter method, values outside the interval [median - k_mad * MAD, median + k_mad * MAD] are considered as outliers. MAD denotes median absolute deviation and k_mad the scaling factor. Each of the above methods operates independently per time point, not globally across time.

Value

A list with following components:

epoch_table   A data frame with epoch ID and the number of time points in which the epoch was evaluated as outlier. (Only epochs with occurrence of outliers in at least one time point are included.)

outliers_data A data frame with subset of data corresponding to the outliers found. (The full record for each flagged point from epoch_table.)

With the setting print_tab = TRUE, the epoch_table is also printed to the console.

Examples

```
# Outlier epoch detection for subject 2, electrode E45 for the whole time range with IQR method
outliers_epoch(epochdata, amplitude = "signal", subject = 2, sensor = "E45", method = "iqr")
# From the result table we see that epochs 14 and 15 were marked as outliers in 50 cases out of 50

# Outlier epoch detection for subject 2, electrode E45 for the whole time range
# using percentile method with 1 and 99 percentiles
outliers_epoch(epochdata, amplitude = "signal", subject = 2, sensor = "E45",
 method = "percentile", p = 0.99)
```

---

pick_data   *Subsets EEG data by subject, sensor, time, or epoch*

Description

Filters an input dataset by optional constraints on subject, sensor, time, and epoch. Filters are combined with logical AND, and exact value matching (%in%) is used.

Usage

```
pick_data(
  data,
  subject_rg = NULL,
  sensor_rg = NULL,
  time_rg = NULL,
  epoch_rg = NULL
)
```

Arguments

`data`	A data frame, tibble or database table with input data. Required columns depend on the further parameters: setting `subject_rg` requires `subject` column etc.
`subject_rg`	Optional vector of subject identifiers to keep (character or numeric, matching `data$subject`). If NULL (default), no filtering is applied based on subject.
`sensor_rg`	Optional vector of sensor identifiers to keep (character or numeric, matching `data$sensor`). If NULL (default), no filtering is applied based on sensor.
`time_rg`	Optional vector of time points to keep (numeric, matching `data$time`). If NULL (default), no filtering is applied based on time.
`epoch_rg`	Optional vector of epoch identifiers to keep (character or numeric, matching `data$epoch`). If NULL (default), no filtering is applied based on epoch.

Details

All filters are combined conjunctively (AND). Matching uses membership (`%in%`) with case-sensitive comparison for character columns. On database backends, very long *_rg vectors may not translate efficiently; consider pre-filtering or semi-joins.

Value

An object of the same class as `data` with rows filtered by the provided criteria; columns are unchanged. If all filters are NULL, the input is returned unmodified. If no rows match, the function ends with error message.

See Also

[compute_mean](#), [baseline_correction](#), [pick_region](#)

Examples

```
# Filtering epochs 1:5 and time points 1:10 for all subjects and sensor "E45"
data_subset <- pick_data(epochdata, sensor_rg = "E45",
 time_rg = 1:10, epoch_rg = 1:5)
head(data_subset)
```

---

pick_region	*Choose region of interest*

Description

The function extracts the selected regions or hemisphere (or a combination of both) from the specified sensor coordinates.

Usage

```
pick_region(
  coords = NULL,
  hemisphere = c("left", "right", "midline"),
  region = c("frontal", "central", "parietal", "occipital", "temporal", "face"),
  ROI = NULL,
  tol = 1e-06
)
```

Arguments

coords	A data frame, matrix or named tibble with numeric columns of "x" and "y" sensor coordinates. If not defined, HCGSN256 template is used. See details for more information about coordinate requirements.
hemisphere	A character vector denoting hemisphere to choose. Possible values: "left", "right", "midline" or any combination of them. If not defined, both hemispheres with midline are chosen.
region	A character vector denoting region to choose. Possible values: "frontal", "central", "parietal", "occipital", "temporal", "face" or any combination of them. If not defined, all regions are chosen.
ROI	A character or factor vector with labels of regions, aligned row-wise with coords. If not defined, the predefined vector (according to HCGSN256 template determined by an expert from Central European Institute of Technology, Masaryk University, Brno, Czech Republic) is used.
tol	A numeric value indicating tolerance for midline selection. (Values of x fulfilling abs(x) < tol are denoted as midline.) Default value is 1e-6.

Details

If the coords input is data frame or matrix with no named columns, the first column is considered as "x" coordinate and second as "y" coordinate. For the correct selection of the hemisphere with own coordinates, it is necessary that the 2D layout is oriented with the nose up and that the midline electrodes should have a zero x-coordinate (or approximately zero within tolerance). Otherwise, the results will not match reality.

Notes: The option hemisphere = "left" (respectively hemisphere = "right") means only the left hemisphere without the midline. If you want to include midline as well, use hemisphere = c("left", "midline") (respectively hemisphere = c("right", "midline")).

The matching of region/hemisphere is exact and the function will stop with an the function stops with an error if no coordinates match the requested region and hemisphere combination.

Value

A tibble or data frame subset of coords filtered by the selected region and hemisphere criteria.

See Also

point_mesh

Examples

```
# Choosing regions from HCGSN256 template
# a) temporal region in left hemisphere
pick_region(hemisphere = "left", region = "temporal")
# b) frontal and central region
region_fc <- pick_region(region = c("frontal", "central"))
head(region_fc)
# c) left hemisphere including midline
hemi_lm <- pick_region(hemisphere = c("left", "midline"))
head(hemi_lm)
# plot the result in c)
plot(hemi_lm$x, hemi_lm$y, pch = 16, asp = 1)
```

---

plot_point_mesh   *Plot point mesh*

Description

Plots a mesh of points (typically from [point_mesh](#), but not necessary) as either a 2D ggplot or 3D rgl plot depending on mesh dimension.

Usage

```
plot_point_mesh(
  mesh,
  sensors = TRUE,
  label_sensors = FALSE,
  sensor_select = NULL,
  names_vec = NULL,
  col = "gray",
  cex = 0.4,
  col_sensors = "green",
  own_coordinates = NULL
)
```

Arguments

mesh	A data frame or tibble with cartesian coordinates of point mesh to plot. It could be D2 or D3 element of output from [point_mesh](#) function or any data frame (or tibble) with named x and y (x, y and z, respectively) columns. See Details for more information.
sensors	A logical value indicating whether the sensor locations should also be plotted (default value is TRUE).
label_sensors	A logical value indicating whether the sensor labels should also be plotted (default value is FALSE).

sensor_select	Optionally, a vector with sensor labels selected from the template during a mesh building. It must be the same as the vector used to create the mesh that the function is supposed to draw, otherwise the final plot will be incorrect.
names_vec	A character vector of labels matching rows in own_coordinates. The argument is required when using own_coordinates together with setting label_sensors = TRUE, otherwise is optional.
col	The colour of mesh points (default colour is gray).
cex	The cex (size) argument for points of the mesh.
col_sensors	The colour of sensor locations points (default colour is green).
own_coordinates	
	A data frame or tibble with coordinates of the sensor locations (matching the dimensionality of mesh and containing appropriate coordinate columns). If the value is NULL and sensors is set to TRUE, the HCGSN256 template is used.

Details

Please follow the instructions below when entering own_coordinates:

The output plot is designed with frontal part of the brain above and occipital part of the brain bottom. The orientation of own_coordinates should be consistent with this. In other case the results could be distorted.

For displaying 3D rgl plot, the own_coordinates must contain the x, y and z coordinates of the sensors, otherwise the function does not work correctly.

The order of elements in names_vec must be consistent with elements of own_coordinates.

When both names_vec and own_coordinates are provided, it is essential that the length of names_vec matches the number of rows in own_coordinates, otherwise the names are not plotted (despite the setting label_sensors = TRUE).

Value

A ggplot object (for 2D mesh) or plots directly to rgl 3D viewer (for 3D mesh).

See Also

point_mesh()

Examples

```
# 2D polygon point mesh with all sensors from the HCGSN256 template
# and default settings
# Note: for nice plot we recommend set par(mar = c(0,0,0,0))
M <- point_mesh(n = 4000, template = "HCGSN256")
plot_point_mesh(M$D2)

## Note: the example opens a rgl 3D viewer
# Plotting 3D polygon point mesh with default settings
rgl::open3d()
plot_point_mesh(M$D3)
```

```
# Plotting 2D circle point mesh with sensors from epochdata as orange points
sensors <- unique(epochdata$sensor)
M <- point_mesh(dim = 2, n = 4000, template = "HCGSN256",
sensor_select = sensors, type = "circle")
plot_point_mesh(M$D2, sensor_select = sensors, col_sensors = "orange")

# Plotting the same mesh with marking only midline electrodes
midline <- HCGSN256$D2[c(8, 15, 21, 26, 78, 86, 95, 111, 117, 127, 136, 204),]
names_vec <- HCGSN256$D2$sensor[c(8, 15, 21, 26, 78, 86, 95, 111, 117, 127, 136, 204)]
plot_point_mesh(M$D2, label_sensors = TRUE, names_vec = names_vec, own_coordinates = midline)
```

plot_time_mean	*Plot time curve of average EEG signal with confidence interval*

Description

Plot a time curve of the average EEG signal amplitude together with pointwise confidence intervals (CIs).

Usage

```
plot_time_mean(
  data,
  FS = 250,
  t0 = 1,
  color = "red",
  fill = "lightsalmon",
  transp = 0.4,
  y_limits = NULL,
  label_0ms = "stimulus",
  label_offset = c(0, 0)
)
```

Arguments

data	A data frame, tibble or a database table with input data to plot. It should be an output from compute_mean function or an object with the same structure, containing columns: `time` with labels of time points and `average`, `ci_low`, `ci_up` with values of average signal and lower and upper CI bounds.
FS	The sampling frequency. Default value is 250 Hz.
t0	Index of the zero time point, i.e. point, where 0 ms should be marked (most often time of the stimulus or time of the response).
color	The color of line. Default value is red. The argument accepts standard R color names or hex codes.
fill	The color of CI bound, default value is lightsalmon. The argument accepts standard R color names or hex codes.

transp	A numeric value between 0 and 1 controlling the transparency of the confidence ribbon (corresponding to alpha parameter in [geom_ribbon](#) function).
y_limits	A numeric vector of length two, specifying the minimum and maximum y-axis limits. Defaults to NULLfor plot limits determined according to input data.
label_0ms	Character string for the annotation label at the 0ms mark. Default is "stimulus".
label_offset	A numeric vector of length two to offset the stimulus label. The first value indicates a horizontal shift, the second a vertical one. Default is c(0,0) for no shift.

Details

The output in the form of a ggplot object allows to easily edit the result image properties. For interactive version of plot see [interactive_waveforms](#) function.

Value

A ggplot object showing the time course of the average EEG signal with pointwise confidence intervals for chosen sensor.

Examples

```
# Plot average signal with CI bounds for subject 2 from the sensor E65
# excluding outlier epochs 14 and 15

# a) preparing data
# a1) extract required data
edata <- epochdata |>
dplyr::filter(subject == 2 & sensor == "E65" & epoch %in% 1:13)
# a2) baseline correction
data_base <- baseline_correction(edata, baseline_range = 1:10)
# a3) average computing
data_mean <- compute_mean(data_base, amplitude = "signal_base", subject = 2, channel = "E65",
 type = "point")
# b) plotting the average line with default settings
plot_time_mean(data = data_mean, t0 = 10)
```

---

plot_topo_mean   *Plot topographic map of average EEG signal*

Description

Plot a topographic circle or polygon map of the average EEG signal amplitude and its lower and upper confidence interval bounds using topographic colour scale. The thin-plate spline interpolation model IM: $\mathbb{R}^2 \to \mathbb{R}$ is used for signal interpolation between the sensor locations. The output in the form of a ggplot object allows to easily edit the result image properties.

Usage

```
plot_topo_mean(
  data,
  mesh,
  coords = NULL,
  template = NULL,
  col_range = NULL,
  col_scale = NULL,
  contour = FALSE,
  show_legend = TRUE,
  label_sensors = FALSE
)
```

Arguments

data	A data frame, tibble or a database table with input data to plot. It should be an output from [compute_mean](#) function or an object with the same structure, containing columns: sensor with sensor labels and average, ci_low, ci_up with values of average signal and its lower and upper CI bounds in one time point (or precomputed average of multiple time points).
mesh	A "mesh" object (or a named list with the same structure) containing at least D2 element with x and y coordinates of a point mesh used for computing IM model. If not defined, the point mesh with default settings from [point_mesh](#) function is used.
coords	Sensor coordinates as a tibble or data frame with named x, y and sensor columns. The sensor labels must match the labels in sensor column in data. If not defined, the HCGSN256 template is used.
template	The kind of sensor template montage used. Currently the only available option is "HCGSN256" denoting the 256-channel HydroCel Geodesic Sensor Net v.1.0, which is also a default setting.
col_range	A vector with minimum and maximum value of the amplitude used in the colour palette for plotting. If not defined, the range of input data (average and CI bounds) is used.
col_scale	Optionally, a colour scale to be utilised for plotting. It should be a list with colors and breaks components (usually created via [create_scale](#)). If not defined, it is computed from col_range.
contour	Logical. Indicates, whether contours should be plotted in the graph. Default value is FALSE.
show_legend	Logical. Indicates, whether legend should be displayed below the graph. Default value is TRUE.
label_sensors	A logical value indicating whether the sensor labels should also be plotted (default value is FALSE).

Details

The spline interpolation is done independently for each CI bound and average.

Note: When specifying the `coords` and `template` at the same time, the `template` parameter takes precedence and the `coords` parameter is ignored.

Value

A `ggplot` object showing the static topographic map of the signal divided into three panels: CI lower, mean, CI upper.

See Also

[animate_topo_mean](): animated version of the plot; [topo_plot]()

Examples

```
# Plot average topographic map with CI bounds of signal for subject 2 from the time point 10
# (the time of the stimulus) excluding outlier epochs 14 and 15

# a) preparing data
# a1) extract required data
edata <- epochdata |>
dplyr::filter(subject == 2 & time %in% 1:10 & epoch %in% 1:13)
# a2) baseline correction (needed for suitable topographic map)
data_base <- baseline_correction(edata, baseline_range = 1:10)
# a3) average computing
data_mean <- compute_mean(data_base, amplitude = "signal_base", subject = 2, time = 10,
 type = "jack", group = "space")
# a4) prepare a mesh for plotting
M <- point_mesh(dimension = 2, n = 3000, template = "HCGSN256",
sensor_select = unique(epochdata$sensor))

# b) plot the topographic map with legend
plot_topo_mean(data = data_mean, mesh = M, template = "HCGSN256", show_legend = TRUE)
```

---

point_mesh	*Create regular mesh of points*

Description

Function creates an object of class `"mesh"`, which is a list of data frames with coordinates of a regular (in the sense of the equidistant distance between mesh nodes) mesh of points on the space defined by sensor coordinates. Circular or polygonal shape of the result mesh is available. For the equivalence between 2D and 3D mesh and the possibility to compare models in different dimensions, the thin-plate spline interpolation model $\mathbb{R}^2 \to \mathbb{R}^3$ is used for creating 3D mesh.

Usage

```
point_mesh(
  dimension = c(2, 3),
  n = 10000,
```

```
    r,
    template = NULL,
    sensor_select = NULL,
    own_coordinates = NULL,
    type = "polygon"
)
```

Arguments

dimension	A number (or a vector) indicating a dimension of the mesh: 2 for two dimensional, 3 for three dimensional mesh and c(2,3) for both of them in one output (default setting).
n	Optionally, the required number of mesh points. Default setting is n = 10 000.
r	Optionally, desired radius of a circular mesh. If not defined, it is computed from the convex hull of sensor locations, based on maximum Euclidean distance from centroid.
template	A character denoting sensor template montage used. Currently the only available option is "HCGSN256" denoting the 256-channel HydroCel Geodesic Sensor Net v.1.0.
sensor_select	Optionally, a vector with sensor labels to select from the template. If not defined, all sensors from the template montage are used to create a mesh.
own_coordinates	
	Optionally, a list with own sensor coordinates for mesh building. See Details for more information.
type	A character indicating the shape of the mesh with 2 possible values: "circle" for circular mesh, "polygon" for irregular polygon shape with boundaries defined by sensor locations (default).

Details

If neither template nor own_coordinates is specified, "HCGSN256" template is used to create the mesh.

In the case of using Geodesic Sensor Net (template = 'HCGSN256'), the (0,0) point of the resulting 2D mesh corresponds to a reference electrode located at the vertex.

The number n for controlling the mesh density is only an approximate value. The final number of mesh nodes depends on the exact shape of the polygon (created as a convex hull of the sensor locations), and is only close to, not exactly equal to, the number n.

The own_coordinates enables computing a mesh from user's own sensor locations. The input must be a list containing following elements:

- D2 a tibble or data frame with sensor coordinates in named x and y columns,
- D3 a tibble or data frame with sensor coordinates in named x, y and z columns.

To build the appropriate meshes in both dimensions, it is necessary to have the input of 3D sensor locations and their corresponding projection onto a plane; the function itself does not perform this projection. It is also necessary to keep the same sensor locations order in D2 and D3 part of the coordinates.

Note: When specifying the `own_coordinates` and `template` at the same time, the `template` parameter takes precedence and the `own_coordinates` parameter is ignored.

Value

Returns a list of class `"mesh"` containing some (or all) of the following components:

`D2`	A data frame with x and y coordinates of the created two dimensional point mesh.
`D3`	A data frame with x, y and z coordinates of the created three dimensional point mesh.
`template`	A character indicating the template of the sensor coordinates used for mesh computing.
`r`	A radius of the circle used for mesh creating.

References

EGI Geodesic Sensor Net Technical Manual (2024)

Examples

```
# Computing circle 2D mesh with starting number 4000 points for HCGSN256 template
# using all electrodes
M <- point_mesh(dimension = 2, n = 4000, template = "HCGSN256", type = "circle")

# Computing polygon 3D mesh with starting number 2000 points and own coordinates
## Note: the coordinates are the same as for HCGSN256 template, it is
## just a mod example of using the own_coordinates parameter
M <- point_mesh(dimension = 3, n = 2000, own_coordinates = HCGSN256)

# Computing coordinates of a polygon mesh in 2D and 3D in one step (starting number 3000 points),
# using 204 electrodes selected for epochdata
# a) create vector with selected sensor labels
sensors <- unique(epochdata$sensor)
# b) create a mesh for selected sensors using sensor_select parameter
M <- point_mesh(n = 3000, template = "HCGSN256", sensor_select = sensors)
```

---

rtdata *Example response time data*

Description

This dataset is a short slice of a high-density (HD-EEG) dataset from a study investigating the impact of deep brain stimulation on patients with advanced Parkinson's disease (Madetko-Alster, 2025). The data contains response times (time between stimulus presentation and pressing the button) from the experiment involving a simple visual-motor task. The study was carried out by Central European Institute of Technology in Brno and was supported by Czech Health Research Council AZV NU21-04-00445.

Example dataset contains response time values in individual experiment epochs for 2 representative subjects (one patient and one healthy control subject).

Usage

```
data("rtdata")
```

Format

The data frame consists of 29 rows (14 for subject 1, 15 for subject 2) and three columns:

subject Factor variable with subject ID, 1 - representative healthy control subject, 2 - representative patient subject.

epoch Factor variable with epoch number (14 epochs for subject 1, 15 epochs for subject 2).

RT Response time in milliseconds.

Details

The epochs and subjects correspond to the sample dataset epochdata.

Source

Central European Institute of Technology, Masaryk University, Brno, Czech Republic.

References

Madetko-Alster N., Alster P., Lamoš M., Šmahovská L., Boušek T., Rektor I. and Bočková M. The role of the somatosensory cortex in self-paced movement impairment in Parkinson's disease. Clinical Neurophysiology. 2025, vol. 171, 11-17.

Examples

```
# Data preview
head(rtdata)
```

---

scalp_plot   *Plot scalp map of EEG signal*

Description

Plot a scalp polygon map of the EEG signal amplitude using topographic colour scale. The thin-plate spline interpolation model IM: $\mathbb{R}^3 \rightarrow \mathbb{R}$ is used for signal interpolation between the sensor locations. The shape3d function is used for plotting.

Usage

```
scalp_plot(
  data,
  amplitude,
  mesh,
  tri = NULL,
  coords = NULL,
  template = NULL,
  col_range = NULL,
  col_scale = NULL,
  view
)
```

Arguments

data	A data frame, tibble or a database table with input data to plot with at least two columns: `sensor` with sensor labels and the column with the EEG amplitude specified in the argument `amplitude`.
amplitude	A character specifying the name of the column from input data with EEG amplitude values.
mesh	An object of class `"mesh"` (or a named list with the same structure) used for computing IM model. If not defined, the polygon point mesh with default settings from `point_mesh` function is used. See details for more information about the structure.
tri	A three column matrix with indices of the vertices of the triangles. Each row represents one triangle, defined by three vertex indices. If missing, the triangulation is computed using `make_triangulation` function from D2 element of the mesh.
coords	Sensor coordinates as a tibble or data frame with named `x`, `y`, `z` and `sensor` columns. The `sensor` labels must match the labels in sensor column in `data`. If not defined, the HCGSN256 template is used.
template	The kind of sensor template montage used. Currently the only available option is `"HCGSN256"` denoting the 256-channel HydroCel Geodesic Sensor Net v.1.0, which is also a default setting.
col_range	A vector with minimum and maximum value of the amplitude used in the colour palette for plotting. If not defined, the range of interpolated signal is used.
col_scale	Optionally, a colour scale to use for plotting. If not defined, it is computed from `col_range`.
view	A character for creating a temporary rotated scene (according to neurological terminology). Possible values are: `"superior"`, `"anterior"`, `"posterior"`, `"left"`, `"right"`. If missing, the default view according to user settings is displayed. Note: Input coordinates corresponding to the positions in the HCGSN template are required to obtain an appropriate view.

Details

The parameter mesh should optimally be a "mesh" object (output from [point_mesh] function) or a list with the same structure: D2 data frame with x and y columns and D3 data frame with x, y and z columns. See [point_mesh] for more information. In that case, setting the argument tri is optional, and if it is absent, a triangulation based on the D2 element of the mesh is calculated and used in the plot. If the input mesh contains only 3D coordinates of a point mesh in D3 element, the use of previously created triangulation (through tri argument) is necessary. To compare results between 2D topographical plot and 3D scalp plot use the same mesh in both cases.

Be careful when choosing the argument col_range. If the amplitude in input data contains values outside the chosen range, this will cause "holes" in the resulting plot. To compare results for different subjects or conditions, set the same values of col_range and col_scale arguments in all cases. The default used scale is based on topographical colours with zero value always at the border of blue and green shades.

Notes: For correct rendering of a plot, the function requires an openGL-capable device. Displaying the rotated scalp map using the view argument requires previous call open3d(). When specifying the coords and template at the same time, the template parameter takes precedence and the coords parameter is ignored.

Value

A 3D scalp map rendered via rgl::shade3d() in an interactive window.

See Also

[animate_scalp], [point_mesh], [make_triangulation], [create_scale]

Examples

```
## Note: The example opens a rgl 3D viewer.
# Plot average scalp map of signal for subject 2 from the time point 10 (the time of the stimulus)
# the outliers (epoch 14 and 15) are extracted before computing

# a) preparing data
edata <- epochdata |>
dplyr::filter(subject == 2 & time %in% 1:10 & epoch %in% 1:13)
# a2) baseline correction (needed for suitable topographic map)
data_base <- baseline_correction(edata, baseline_range = 1:10)
# a3) average computing
data_mean <- compute_mean(data_base, amplitude = "signal_base", subject = 2, time = 10,
 type = "point", ex_epoch = c(14,15))

# b) plotting the scalp polygon map
scalp_plot(data_mean, amplitude = "average", col_range = c(-30, 15))
```

---

topo_plot   *Plot topographic map of EEG signal*

Description

Plot a topographic circle or polygon map of the EEG signal amplitude using topographic colour scale. The thin-plate spline interpolation model IM: $\mathbb{R}^2 \rightarrow \mathbb{R}$ is used for signal interpolation between the sensor locations. The output in the form of a ggplot object allows to easily edit the result image properties.

Usage

```
topo_plot(
  data,
  amplitude,
  mesh,
  coords = NULL,
  template = NULL,
  col_range = NULL,
  col_scale = NULL,
  contour = FALSE,
  show_legend = TRUE,
  label_sensors = FALSE
)
```

Arguments

data	A data frame, tibble or a database table with input data to plot with at least two columns: sensor with sensor labels and the column with the EEG amplitude specified in the argument amplitude.
amplitude	A character string naming the column with EEG amplitude values.
mesh	A "mesh" object (or a named list with the same structure) containing at least D2 element with x and y coordinates of a point mesh used for computing IM model. If not defined, the point mesh with default settings from [point_mesh](#) function is used.
coords	Sensor coordinates as a tibble or data frame with named x, y and sensor columns. The sensor labels must match the labels in sensor column in data. If not defined, the HCGSN256 template is used.
template	The kind of sensor template montage used. Currently the only available option is "HCGSN256" denoting the 256-channel HydroCel Geodesic Sensor Net v.1.0, which is also a default setting.
col_range	A vector with minimum and maximum value of the amplitude used in the colour palette for plotting. If not defined, the range of interpolated signal is used.
col_scale	Optionally, a colour scale to be utilised for plotting. It should be a list with colors and breaks components (usually created via [create_scale](#)). If not defined, it is computed from col_range.

contour   Logical. Indicates, whether contours should be plotted in the graph. Default
                 value is FALSE.

show_legend   Logical. Indicates, whether legend should be displayed beside the graph. De-
                 fault value is TRUE.

label_sensors   A logical value indicating whether the sensor labels should also be plotted (de-
                 fault value is FALSE).

Details

For more details about required mesh structure see point_mesh function. If the input mesh structure
does not match this format, an error or incorrect function behavior may occur.

Be careful when choosing the argument col_range. If the amplitude in input data contains values
outside the chosen range, this will cause "holes" in the resulting plot. To compare results for differ-
ent subjects or conditions, set the same values of col_range and col_scale arguments in all cases.
The default used scale is based on topographical colours with zero value always at the border of
blue and green shades.

Note: When specifying the coords and template at the same time, the template parameter takes
precedence and the coords parameter is ignored.

Value

A ggplot object showing an interpolated topographic map of EEG amplitude.

See Also

animate_topo, point_mesh, plot_topo_mean

Examples

```
# Plot average topographic map of signal for subject 2 from the time point 10
# (the time of the stimulus) without the outliers (epoch 14 and 15)

# a) preparing data
# a1) extract required data
edata <- epochdata |>
dplyr::filter(subject == 2 & time %in% 1:10 & epoch %in% 1:13)
# a2) baseline correction (needed for suitable topographic map)
data_base <- baseline_correction(edata, baseline_range = 1:10)
# a3) average computing
data_mean <- compute_mean(data_base, amplitude = "signal_base", subject = 2, time = 10,
 type = "jack", group = "space")


# b) plotting the topographic map with contours and legend
# interval (-30,15) is selected in consideration of the signal progress
topo_plot(data = data_mean, amplitude = "average", template = "HCGSN256",
col_range = c(-30, 15), contour = TRUE)

# c) plotting the same map without contours but with sensor labels
topo_plot(data = data_mean, amplitude = "average", template = "HCGSN256",
```

```
col_range = c(-30, 15), label_sensors = TRUE)
```

Index