# Package: diegr (via r-universe)

October 19, 2026

**Title** Dynamic and Interactive EEG Graphics

**Version** 0.3.1

**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** 2026-08-19

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.3

**Depends** R (>= 4.1.0)

**Imports** dplyr, rlang, ggplot2, gganimate, plotly, rgl, sf, scales, stats, purrr, tidyr, MASS

**LazyData** true

**Suggests** av, gifski, magick, knitr, rmarkdown, RSQLite, DBI, dbplyr, testthat (>= 3.0.0)

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**URL** https://gerslovaz.github.io/diegr/

**Config/Needs/website** rmarkdown

**Config/pak/sysreqs** libabsl-dev cmake libfreetype6-dev libgdal-dev gdal-bin libgeos-dev libglu1-mesa-dev make texlive libicu-dev libpng-dev libuv1-dev libgl1-mesa-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev zlib1g-dev

**Repository** https://gerslovaz.r-universe.dev

**Date/Publication** 2026-08-20 07:38:19 UTC

**RemoteUrl** https://github.com/gerslovaz/diegr

**RemoteRef** HEAD

**RemoteSha** 8250a2a287f916f681c52071a793d52ba3f25254

# Contents

animate_scalp . . . 2
animate_topo . . . 5
animate_topo_mean . . . 7
baseline_correction . . . 9
biosemi128 . . . 11
biosemi256 . . . 12
boxplot_epoch . . . 13
boxplot_rt . . . 14
boxplot_subject . . . 15
check_structure . . . 17
compute_mean . . . 18
create_scale . . . 21
epochdata . . . 23
HCGSN256 . . . 24
interactive_surfaceplot . . . 25
interactive_surfaceplot_curves . . . 26
interactive_waveforms . . . 28
make_triangulation . . . 29
outliers_epoch . . . 31
pick_data . . . 33
pick_region . . . 35
plot_point_mesh . . . 36
plot_time_mean . . . 39
plot_topo_mean . . . 41
point_mesh . . . 43
rtdata . . . 45
scalp_plot . . . 46
summary_stats_rt . . . 48
system1005 . . . 50
topo_plot . . . 51

**Index** **54**

---

`animate_scalp` *3D scalp plot animation in time*

---

## Description

Display a topographic 3D scalp animation of the change in amplitude over time. The function enables direct rendering in Rstudio Viewer or saving the animation in MP4 format or individual frames in PNG format to the chosen location.

## 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 these 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 | A "mesh" object (or a named list with the same structure) containing at least a D3 element with x, y and z coordinates of a point mesh used for computing the IM model, and a template element specifying the sensor montage. 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 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, z and sensor columns. The sensor labels must match the labels in sensor column in data. If not defined, the template specified in mesh$template (or the default "HCGSN256") is used. |
| template | The kind of sensor template montage used. Available options are "HCGSN256", "biosemi128", "biosemi256", and "system1005". Default setting is "HCGSN256". |
| 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.

If a mesh object is provided, its internal template name (mesh$template) overrides the template argument to ensure spatial consistency. When custom coords are provided, they are used for plotting the sensor locations. The template parameter (or mesh$template) is then used only for generating the background mesh if it is not provided.

## Value

If frames_dir is NULL, the function creates an interactive 3D animation in the rgl window and invisibly returns NULL. If frames_dir is specified, individual animation frames (PNG) are saved. If output_path is also specified, an MP4 video is encoded and the function invisibly returns NULL.

Additionally, the returned object carries a "diegr_metadata" attribute with metadata such as details about the mesh used for plotting.

## See Also

Static version: scalp_plot, animated 2D topo map: animate_topo

## 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 <- pick_data(epochdata, subject_rg = 1, epoch_rg = 5, time_rg = 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 these 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 a `D2` element with x and y coordinates of a point mesh used for computing the IM model, and a `template` element specifying the sensor montage. 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 template specified in mesh$template (or the default "HCGSN256") is used. |
| template | The kind of sensor template montage used. Available options are "HCGSN256", "biosemi128", "biosemi256", and "system1005". Default setting is "HCGSN256". |
| 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.

Notes: If a mesh object is provided, its internal template name (mesh$template) overrides the template argument to ensure spatial consistency.

When custom coords are provided, they are always used for plotting the sensor locations. The template parameter (or mesh$template) is then used only for generating the background mesh if it is not provided.

## Value

If output_path is NULL, the function returns a gganim object and prints the animation to the RStudio Viewer. If output_path is specified, the animation is saved to the given file path and the function invisibly returns NULL.

Additionally, the returned object carries a "diegr_metadata" attribute with metadata such as details about the mesh used for plotting.

## See Also

Static version: topo_plot, animated 3D scalp map: animate_scalp

## 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 <- pick_data(epochdata, subject_rg = 1, epoch_rg = 5, time_rg = 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. Required columns: `sensor` - sensor labels, `time` - numbers of time points, `average` - average signal values, `ci_low` and `ci_up` - 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 a D2 element with x and y coordinates of a point mesh used for computing the IM model, and a template element specifying the sensor montage. 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 template specified in mesh$template (or the default "HCGSN256") is used. |
| template | The kind of sensor template montage used. Available options are "HCGSN256", "biosemi128", "biosemi256", and "system1005". Default setting is "HCGSN256". |
| 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

Notes: If a mesh object is provided, its internal template name (mesh$template) overrides the template argument to ensure spatial consistency. When custom coords are provided, they are always used for plotting the sensor locations. The template parameter (or mesh$template) is then used only for generating the background mesh if it is not provided.

## Value

If output_path is NULL, the function returns a magick-image object containing the animated frames and prints it to the RStudio Viewer. If output_path is specified, the animation is saved as a GIF to the given file path and the function invisibly returns NULL. The gifski and magick packages are required for animation export.

Additionally, the returned object carries a "diegr_metadata" attribute with metadata such as details about the mesh used for plotting.

## See Also

animate_topo, compute_mean, baseline_correction, static version: plot_topo_mean

## 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 <- pick_data(epochdata, subject_rg = 2, epoch_rg = 1:13, time_rg = 1:10)
data_base <- baseline_correction(edata, baseline_range = 1:9) # baseline correction
data_mean <- compute_mean(data_base, amplitude = "signal_base",
 type = "jack", domain = "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

Compute amplitude values corrected to the selected baseline.

The function computes a baseline value within each epoch and subtracts it from the signal.

## Usage

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

## Arguments

| | |
|---|---|
| `data` | A data frame, tibble or a database table with input data, required columns: `time`, `epoch` and `signal`. Optional columns: `group`, `subject`, `sensor`, and `condition`, if present, are included in the grouping structure, see Details. |
| `baseline_range` | A numeric vector of time points used as the baseline (e.g., `baseline_range = 125:250`). |
| `type` | A character specifying the type of baseline correction. Currently, only `"absolute"` is supported, any other value results in an error. |

## Details

Grouping logic for baseline correction: The function relies on a strict grouping hierarchy: `"group" > "subject" > "sensor"`. The function dynamically identifies which of the optional variables are present in the data, preserves their hierarchical order, and uses them to group the data.

Dealing with `baseline_range`: If the values from `baseline_range` vector extend beyond the range of the `time` column, the baseline computation 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 NAs. In both cases, the function returns the output data along with a warning.

Notes:

- Rows with NA values in the signal column are ignored when computing the baseline, and a warning is issued.
- If any grouping variable present in the data contains only NA values, a warning is issued, as this may lead to invalid or uninformative grouping.

## 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. |

Additionally, the returned object carries a "diegr_metadata" attribute with metadata.

## Examples

```
# Computing baseline correction for subject 1 on first 9 points (points before stimulus),
# sensor "E1"
# a) Prepare data and compute
data01 <- epochdata |> dplyr::filter(.data$subject == 1 & .data$sensor == "E1")
basedata <- baseline_correction(data01, baseline_range = 1:9, type = "absolute")

## Note: You can also use baseline_correction() on the whole epochdata
## and then filter selected subject and sensor, the results are the same,
## the procedure above was chosen only for the speed of the example.

# 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)
```

`biosemi128` *Coordinates of the spherical BioSemi system with 128 electrodes*

## Description

A dataset containing the Cartesian coordinates of high-density EEG sensor positions of the BioSemi system in 3D space on an idealized spherical surface, along with their corresponding positions in 2D space. This template contains 128 electrode positions in the 3D space and for the 2D layout.

## Usage

```
data("biosemi128")
```

## Format

A list with the following elements:

**D2** A tibble with 3 columns containing x and y coordinates and sensor labels (according to the BioSemi naming convention) in 2D.

**D3** A tibble with 4 columns containing x, y and z coordinates and sensor labels (according to the BioSemi naming convention) in 3D. See 'Details' for more information.

**ROI** A factor containing the name of the region to which the corresponding sensor belongs. The levels are: "central", "frontal", "occipital", "parietal", and "temporal". Sensors were assigned to anatomical regions based on their closest spatial proximity (3D Euclidean distance) to the standardized 10-05 system electrodes.

## 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 (+). Because this is an idealized spherical model, the origin (0, 0, 0) represents the geometric center of the sphere.

The coordinates originate from the MNE-Python (3D montage) and FieldTrip (2D layout) GitHub repositories.

## Source

The MNE-Python GitHub repository, https://github.com/mne-tools/mne-python/tree/main/mne/channels/data/montages The FieldTrip GitHub repository, https://github.com/fieldtrip/fieldtrip/tree/master/template/layout

## References

BioSemi Headcaps: https://www.biosemi.com/headcap2.htm

## Examples

```
# A simple plot of sensor coordinates from BioSemi 128 system spherical template as points in 2D
plot(biosemi128$D2[,1:2], pch = 16, asp = 1)
```

---

`biosemi256` *Coordinates of the spherical BioSemi system with 256 electrodes*

---

## Description

A dataset containing the Cartesian coordinates of high-density EEG sensor positions of the BioSemi system in 3D space on an idealized spherical surface, along with their corresponding positions in 2D space. This template contains 256 electrode positions in the 3D space and for the 2D layout.

## Usage

```
data("biosemi256")
```

## Format

A list with the following elements:

**D2** A tibble with 3 columns containing x and y coordinates and sensor labels (according to the BioSemi naming convention) in 2D.

**D3** A tibble with 4 columns containing x, y and z coordinates and sensor labels (according to the BioSemi naming convention) in 3D. See 'Details' for more information.

**ROI** A factor containing the name of the region to which the corresponding sensor belongs. The levels are: "central", "frontal", "occipital", "parietal", and "temporal". Sensors were assigned to anatomical regions based on their closest spatial proximity (3D Euclidean distance) to the standardized 10-05 system electrodes.

## 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 (+). Because this is an idealized spherical model, the origin (0, 0, 0) represents the geometric center of the sphere.

The coordinates originate from the MNE-Python (3D montage) and FieldTrip (2D layout) GitHub repositories.

## Source

The MNE-Python GitHub repository, https://github.com/mne-tools/mne-python/tree/main/mne/channels/data/montages The FieldTrip GitHub repository, https://github.com/fieldtrip/fieldtrip/tree/master/template/layout

## References

BioSemi Headcaps: https://www.biosemi.com/headcap2.htm

## Examples

```
# A simple plot of sensor coordinates from BioSemi 256 system spherical template as points in 2D
plot(biosemi256$D2[,1:2], pch = 16, asp = 1)
```

---

# boxplot_epoch *Plot boxplots of EEG amplitude on epoch level*

---

## Description

Function for plotting interactive or static boxplots of EEG amplitude in individual epochs within the chosen time interval. The function assumes data from a single subject and a single sensor. Both the static `ggplot` output and the interactive `plotly` output can be easily customized and edited, while the interactive `plotly` output additionally allows easy identification of the epochs corresponding to outliers.

## Usage

```
boxplot_epoch(
  data,
  amplitude = "signal",
  epoch = NULL,
  time_lim,
  title_label = NULL,
  use_latex = TRUE,
  interactivity = TRUE
)
```

## Arguments

| | |
|---|---|
| `data` | A data frame or a database table with EEG dataset. Required columns: `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"`. |
| `epoch` | A vector with numbers of epochs to plot. If missing, boxplots are drawn for all avaliable epochs in `data`. |
| `time_lim` | A numeric vector with time range to plot. |
| `title_label` | A character string specifying the title of the plot. Defaults to `NULL` for plot without title. |
| `use_latex` | A logical value indicating whether to use LaTeX formatting for the y-axis title. The default is `TRUE`. |
| `interactivity` | Logical. Determines whether to render an interactive `plotly` graph (default) or a static `ggplot` graph. |

## Details

The input data frame or database table must contain at least following columns: 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 or ggplot2 object with boxplots of EEG amplitude for individual epochs.

Additionally, the returned object carries a "diegr_metadata" attribute with metadata.

## See Also

boxplot_subject

## Examples

```
# Interactive boxplots of signal from channel E34 for subject 1 (health control)
# in time points 10:20
exampledata <- epochdata |>
pick_data(subject_rg = 1, sensor_rg = "E34")
boxplot_epoch(exampledata, amplitude = "signal", time_lim = c(10:20),
title_label = "Subject 1, channel E34")

# static version of the previous graph
boxplot_epoch(exampledata, amplitude = "signal", time_lim = c(10:20),
title_label = "Subject 1, channel E34", interactivity = FALSE)
```

---

# boxplot_rt *Plot boxplots of response times*

---

## Description

Function for plotting interactive or static boxplots of response time in individual epochs for selected subjects. Both the static ggplot output and the interactive plotly output can be easily customized and edited, while the interactive plotly output additionally allows easy identification of the epoch numbers corresponding to outliers.

If the condition column is present in the input data, the boxplots are color-coded by condition.

## Usage

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

### Arguments

| | |
|---|---|
| `data` | A data frame or a database table containing response time data. Required columns: `subject`, `epoch`, `RT` (value of response time in ms). An optional `condition` column may be used to colour-code the boxplots. |
| `subject` | A vector with IDs of subjects to plot. If missing, boxplots are drawn for all available subjects in `data`. |
| `interactivity` | Logical. Determines whether to render an interactive `plotly` graph (default) or a static `ggplot` graph. |

### Value

A `plotly` or `ggplot2` object with boxplots of response times.

Additionally, the returned object carries a `"diegr_metadata"` attribute with metadata.

### Examples

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

# Display static boxplots for both example subjects
boxplot_rt(rtdata, interactivity = FALSE)

# Interactive boxplots per subject divided by condition
# a) add condition column to data (just for example)
data_cond <- rtdata
data_cond$condition <- c(rep("a", 7), rep("b", 7), rep("a", 8), rep("b",7))
# b) plot boxplots (colour-coded by condition)
boxplot_rt(data_cond)
```

---

## `boxplot_subject` *Plot boxplots of EEG amplitude across subjects*

---

### Description

Function for plotting interactive or static boxplots of EEG amplitude across subjects for a single epoch and channel, within a specified time interval. The function assumes data from a single epoch and a single sensor. Both the static `ggplot` output and the interactive `plotly` output can be easily customized and edited, while the interactive `plotly` output additionally allows easy identification of the subjects corresponding to outliers.

### Usage

```
boxplot_subject(
  data,
  amplitude = "signal",
  subject = NULL,
  time_lim,
```

```
  title_label = NULL,
  use_latex = TRUE,
  interactivity = 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. |
| time_lim | A numeric vector with time range to plot. |
| title_label | A character string specifying the title of the plot. Defaults to NULL for plot without title. |
| use_latex | A logical value indicating whether to use LaTeX formatting for the y-axis title. The default is TRUE. |
| interactivity | Logical. Determines whether to render an interactive plotly graph (default) or a static ggplot graph. |

## Details

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

Note: The function assumes that subject IDs are unique across the entire dataset. Using the same subject IDs in multiple groups may result in incorrect or misleading visualizations.

## Value

A plotly or ggplot2 object with boxplots of EEG amplitude across subjects.

Additionally, the returned object carries a "diegr_metadata" attribute with metadata.

## See Also

boxplot_epoch

## 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
exampledata <- epochdata |>
pick_data(sensor_rg = "E34", epoch_rg = 1)
boxplot_subject(exampledata, amplitude = "signal", time_lim = c(10:20),
```

```
title_label = "Sensor E34, epoch 1")

# Static version of the previous graph
boxplot_subject(exampledata, amplitude = "signal", time_lim = c(10:20),
title_label = "Sensor E34, epoch 1", interactivity = FALSE)
```

---

`check_structure` *Check data structure and print inferred hierarchy*

---

## Description

A diagnostic helper function to run before starting the `diegr` analysis. It infers the experimental hierarchy based on the standardized column names strictly required by the package and prints a readable summary. Supports data frames, tibbles, and database tables.

## Usage

```
check_structure(data, value_col = "signal")
```

## Arguments

| | |
|---|---|
| `data` | A data frame, tibble, or database table containing the EEG data in long format. |
| `value_col` | Character string specifying the signal amplitude column name (default `"signal"`). |

## Details

The `diegr` package does not strictly require all structural columns for every function (e.g., `group`, `condition`, and `epoch` may be optional). However, if they are present, they must follow the exact naming convention (`group`, `subject`, `sensor`, `epoch`, `condition`, `time`). Only the signal amplitude column can be custom-named via the `value_col` argument. This function helps verify which columns were correctly recognized.

## Value

The original data object invisibly, allowing it to be used in pipes.

## Examples

```
# Checking the structure of epochdata
check_structure(epochdata)
```

compute_mean *Calculate mean in temporal or spatial domain*

## Description

Calculate a pointwise or a jackknife (leave-one-out) average signal across temporal or spatial domain together with standard error and pointwise confidence interval (CI) bounds. Pointwise averages can be computed in two ways: standard (un-weighted) by default, or weighted using the values in the column specified by `weights_col`.

The function computes an average at group, subject, sensor/time point, condition or epoch level (according to the `level` parameter). For the option `level = "epoch"` the epochs are averaged etc. Function assumes pre-prepared data according to the chosen level.

## Usage

```
compute_mean(
  data,
  amplitude = "signal_base",
  domain = c("time", "space"),
  level = c("epoch", "condition", "sensor", "subject", "group"),
  type = c("point", "jack"),
  weights_col = NULL,
  R = NULL,
  alpha = 0.95
)
```

## Arguments

| | |
|---|---|
| `data` | A data frame, tibble or a database table with input data, required columns: `time` or `sensor` (according to the selected domain), the column with the EEG amplitude specified in the argument `amplitude` and columns corresponding to the selected `level`. |
| `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. |
| `domain` | A character specifying the domain over which the average is computed. One of `"time"` or `"space"`. Option `"time"` computes a time-resolved average at each time point, whereas `"space"` computes a space-resolved average at each sensor. |
| `level` | A character specifying the level of average calculation. The possible values are `"epoch"`,`"condition"`, `"sensor"`, `"subject"` and `"group"`. See Details for more information. |
| `type` | A character specifying the method of calculating the average, `"point"` for pointwise average and `"jack"` for jackknife leave-one-out average. |
| `weights_col` | A character specifying the name of the column containing observation weights. If `NULL`, un-weighted standard pointwise average is computed. |

| | |
|---|---|
| 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 function supports averaging at different hierarchical levels of the data (using `level` argument):

- `"epoch"`: averaging across epochs. Returns the average curve (time domain) or sensor array (space domain) for each combination of other grouping variables.
- `"condition"`: averages across experimental conditions.
- `"sensor"`: averages across sensors (space domain) or time points (time domain).
- `"subject"`: averages across subjects.
- `"group"`: averages across groups of subjects (highest aggregation level). The function assumes input adapted to the desired level of averaging (i.e. for `epoch` level the `epoch` column must be present in `data` etc.). For all levels higher than epochs, the averages of the lower level are assumed as the input data.

Weighted vs un-weighted average (`type = "point"`):

- If `weights_col` is `NULL`, each observation is treated equally (with weight = 1), producing a standard un-weighted mean, standard errors (SE), and CI.
- If `weights_col` is provided, a weighted average is computed using the values in the specified column as weights. SE and CI are computed based on the weighted variance.

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 standard mean or its weighted alternative (if `weights_col` is specified).
- 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`, `weights_col` or other columns, 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`, `domain` and `alpha` arguments. The statistics are saved in columns

- `average` for computed average amplitude value,
- `n` for number of observations used in average computing,
- `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.

Additionally, the returned object carries a `"diegr_metadata"` attribute with metadata such as level and domain used in computing.

## 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 <- epochdata |>
pick_data(subject_rg = 1, epoch_rg = 1:13, sensor_rg = "E1") |>
compute_mean(amplitude = "signal", level = "epoch", domain = "time")
str(avg_data)

# plot the result using interactive plot with pointwise CI
avg_data |>
pick_data(subject = 1) |>
interactive_waveforms(amplitude = "average", 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 1:9 (again without outlier epoch 14)
# a) prepare corrected data
basedata <- pick_data(epochdata, subject_rg = 1) |>
baseline_correction(baseline_range = 1:9, type = "absolute")
# b) filter time point 11 (without epoch 14) and compute the average
avg_data <- pick_data(basedata, time_rg = 11, epoch_rg = 1:13) |>
compute_mean(amplitude = "signal_base", level = "epoch", domain = "space", type = "jack")
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) compute mean from all epochs for each subject
mean_epoch <- epochdata |>
pick_data(time_rg = 11, epoch_rg = 1:13) |>
compute_mean(amplitude = "signal", level = "epoch", domain = "space", type = "point")
# b) compute mean on subject level
```

```
mean_subjects <- compute_mean(mean_epoch, amplitude = "average", level = "subject",
domain = "space", type = "point")
head(mean_subjects)
# c) compute weighted mean with number of observations as weights
weighted_mean_subjects <- compute_mean(mean_epoch, amplitude = "average", level = "subject",
domain = "space", type = "point", weights_col = "n")
head(weighted_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,
  type = c("redblue", "topo"),
  symmetric = TRUE
)
```

## 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`. |
| `type` | A character indicating the type of color palette to create. Available options: `"redblue"` (default value) for red-blue palette and `"topo"` for topographical palette, see Details for more information. |
| `symmetric` | A logical value indicating whether the color range should be strictly symmetrized around zero. Defaults to `TRUE`. |

## Details

The topographical palette (`type = "topo"`) 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.

The red-blue palette (`type = "redblue"`) has negative values corresponding to shades of blue and positive values corresponding to shades of red.

By default (`symmetric = TRUE`), to guarantee true perceptual uniformity and equal visual emphasis, the provided `col_range` is additionally automatically symmetrized around zero based on its maximum absolute value. For example, an input range of `c(-2, 5)` is internally expanded to `c(-5, 5)`.

For the default `redblue` palette, this symmetrization ensures that equivalent positive and negative amplitudes are mapped to identical color intensities. Furthermore, the `redblue` palette strictly anchors the zero-point to a neutral background color, preventing false perceptual boundaries. (Note that while the legacy `topo` palette is also symmetrized, it retains its characteristic blue-green transition at zero).

Setting `symmetric = FALSE` disables this zero-centering expansion, which may be useful for strictly unipolar data or heavily skewed distributions, though it sacrifices equal visual emphasis for opposite polarities.

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 $\leq 30$,
- `k = 0.03` for range $\geq 70$,
- `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.

Additionally, the returned object carries a `"diegr_metadata"` attribute with metadata such as the actual `k` used for creating the scale.

## Examples

```
# Create red-blue scale on interval (-10,10) with default step number
create_scale(col_range = c(-10,10), type = "redblue")

# Create an asymmetric topographic scale on interval c(-5,10) with small k (finer division)
CStopo <- create_scale(col_range = c(-5, 10), k = 0.02, type = "topo", symmetric = FALSE)
# plot colours of the scale as points
k_col <- length(CStopo$colors)
plot(1:k_col, rep(1, k_col), col = CStopo$colors, pch = 16,
 axes = FALSE, ylab = "", xlab = "")
```

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 HydroCel 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 (according to EGI GSN Technical Manual) in 3D. See 'Details' for more information.

**ROI** A 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_surfaceplot

*Interactive 3D surface topographic plot of EEG signal*

---

## Description

Creates an interactive 3D mesh surface plot of EEG amplitude in one time point using `plotly`. The dense point mesh forms the 2D base, while the interpolated signal amplitude determines the 3D elevation and surface color.

## Usage

```
interactive_surfaceplot(
  data,
  amplitude,
  mesh,
  coords = NULL,
  template = NULL,
  col_range = NULL,
  col_scale = NULL,
  show_sensors = TRUE
)
```

## Arguments

| | |
|---|---|
| `data` | A data frame, tibble, or a database table with input data to plot. It must contain at least two columns: `sensor` with sensor labels, and the column with the EEG amplitude specified in the argument `amplitude`. |
| `amplitude` | A character string specifying the name of the column from the input data containing EEG amplitude values. |
| `mesh` | A `"mesh"` object (or a named list with the same structure) containing at least a `D2` element with x and y coordinates of a point mesh used for computing the IM model, and a `template` element specifying the sensor montage. 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 template specified in mesh$template (or the default "HCGSN256") is used. |
| template | The kind of sensor template montage used. Available options are "HCGSN256", "biosemi128", "biosemi256", and "system1005". Default setting is "HCGSN256". |
| 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_sensors | Logical. Indicates whether original sensor locations should be visualized as black scatter points on the 3D surface. Default is TRUE. |

## Value

A plotly widget object containing the interactive 3D surface plot.

Additionally, the returned object carries a "diegr_metadata" attribute with metadata such as details about the data and plot.

## See Also

interactive_surfaceplot_curves, scalp_plot

## Examples

```
# Prepare a data structure:
edata <- pick_data(epochdata, subject_rg = 2, epoch_rg = 1:13, time_rg = 1:10)
data_base <- baseline_correction(edata, baseline_range = 1:9)
data_mean <- compute_mean(data_base, amplitude = "signal_base", type = "point", domain = "space")

# Create an interactive 3D surface plot of average signal in time point 10:
data_mean |>
dplyr::filter(time == 10) |>
interactive_surfaceplot(amplitude = "average", col_range = c(-10, 10))
```

---

# interactive_surfaceplot_curves

*Interactive 3D surface plot of EEG signal over time*

---

## Description

Creates an interactive 3D surface plot displaying the EEG signal amplitude across different sensors and time points. The x-axis represents time, the y-axis represents the sensors, and the z-axis (along with surface color) represents the signal amplitude.

## Usage

```
interactive_surfaceplot_curves(
  data,
  amplitude,
  sensor_ticks = NULL,
  col_range = NULL,
  col_scale = NULL
)
```

## Arguments

| | |
|---|---|
| data | A data frame, tibble, or a database table containing the EEG data. Required columns are: sensor, time, and the column with the EEG amplitude specified in the argument amplitude. |
| amplitude | A character string specifying the name of the column from the input data containing EEG amplitude values. |
| sensor_ticks | A character vector specifying which sensor labels should be displayed on the y-axis. If NULL, no specific labels are forced, but mismatched names will throw an error if provided. |
| 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. |

## Value

A plotly widget object containing the interactive 3D surface plot.

Additionally, the returned object carries a "diegr_metadata" attribute with metadata.

## See Also

interactive_surfaceplot, plot_topo_mean

## Examples

```
# Prepare data: Mean across epochs 1:13 for Subject 2
edata <- pick_data(epochdata, subject_rg = 2, epoch_rg = 1:13)
data_base <- baseline_correction(edata, baseline_range = 1:9)
data_mean <- compute_mean(data_base, amplitude = "signal_base", type = "point", domain = "time")

# Selected sensors to display on the axis
selected_sensors <- c("E1", "E21", "E41", "E61", "E81", "E101",
 "E121", "E141", "E161", "E181", "E201", "E221")

# Render the interactive plot
interactive_surfaceplot_curves(data_mean, amplitude = "average", sensor_ticks = selected_sensors)
```

`interactive_waveforms` *Plot interactive waveform graph*

## Description

Function for plotting time series of EEG signal colour-coded by epoch, condition, channel, subject or group (depending on selected `level` parameter) an interactive `plotly` graph. The function assumes that the input data have already been filtered to the desired subset according to the `level`. 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",
  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 `amplitude` and `level` parameter (see Details). |
| `amplitude` | A character specifying the name of the column from input data with an EEG amplitude values. Default is `"signal"`. |
| `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), `"condition"`, `"sensor"`, `"subject"` and `"group"`. See details for more information. |
| `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 column time (a column with time point numbers) and a column with the EEG amplitude (or average amplitude) specified in the argument amplitude. It must also contain at least one of the optional columns (according to the level parameter - for "sensor" level the column sensor is required etc.): group - a column with group identifiers, subject - a column with subject IDs, sensor - a column with sensor labels, epoch - a column with epoch numbers.

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.

Additionally, the returned object carries a "diegr_metadata" attribute with metadata such as plot and data details.

## 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
epochdata |>
pick_data(subject_rg = 1, sensor_rg = "E65") |>
interactive_waveforms(amplitude = "signal", t0 = 10, level = "epoch")

# 2) Plot sensor level waveforms with confidence bands for subject 1 and electrodes "E65" and "E182"
# a) preparing data
sendata <- epochdata |>
pick_data(subject_rg = 1, sensor_rg = c("E65", "E182")) |>
compute_mean(amplitude = "signal", domain = "time", level = "epoch")
# b) plot the waveforms without the average
interactive_waveforms(sendata, amplitude = "average", 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

Please note that the input coordinates must be unique. The function execution will automatically stop if duplicate sensor locations are found.

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 identifies epochs with outlier values in a numeric EEG amplitude variable in chosen time points. Outliers are detected separately at each time point within the groups present in the data. The function then summarizes how many times each epoch was marked as an outlier across all 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",
  time = NULL,
  method = c("iqr", "percentile", "hampel"),
  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: time, epoch and the column with EEG amplitude specified by amplitude parameter. Optional columns: group, subject, sensor, condition. |
| amplitude | A character specifying the name of the column from input data with an EEG amplitude values. Default is "signal". |
| 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 (default value), "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: epoch - a column with epoch numbers/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.

Note: For large datasets, the calculation can be time-consuming. It is recommended to pre-filter or subset the data before using this function to reduce computation 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.

Additionally, the returned object carries a `"diegr_metadata"` attribute with metadata such as method used for outlier detection.

## Examples

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

# 2. Outlier epoch detection for both subjects, electrode E45 for the whole time range
# using percentile method with 1 and 99 percentiles
outdata <- epochdata |>
pick_data(sensor_rg = "E45") |>
outliers_epoch(amplitude = "signal", method = "percentile", p = 0.99)
# see head of outliers data
head(outdata$outliers_data)

# 3. Outlier epoch detection for subject 2, electrode E45 for the whole time range
# using Hampel filter method and scaling factor = 4
epochdata |>
pick_data(subject_rg = 2, sensor_rg = "E45") |>
outliers_epoch(amplitude = "signal", method = "hampel", k_mad = 4)
```

---

`pick_data` *Subsets EEG data by group, subject, sensor, time, experimental condition or epoch*

---

## Description

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

## Usage

```
pick_data(
  data,
  group_rg = NULL,
  subject_rg = NULL,
  sensor_rg = NULL,
  condition_rg = NULL,
  epoch_rg = NULL,
  time_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. |
| group_rg | Optional vector of group identifiers to keep (character or numeric, matching data$group). If NULL (default), no filtering is applied based on group. |
| subject_rg | Optional vector of subject identifiers to keep (character or numeric, matching data$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$sensor). If NULL (default), no filtering is applied based on sensor. |
| condition_rg | Optional vector of experimental condition identifiers to keep (character or numeric, matching data$condition). If NULL (default), no filtering is applied based on condition. |
| epoch_rg | Optional vector of epoch identifiers to keep (character or numeric, matching data$epoch). If NULL (default), no filtering is applied based on epoch. |
| time_rg | Optional vector of time points to keep (numeric, matching data$time). If NULL (default), no filtering is applied based on time. |

## 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)


# Setting parameters outside the input data range (there is no subject 6 in epochdata)
# results in an error message
try(
pick_data(epochdata, subject_rg = 6,
 time_rg = 1:10, epoch_rg = 1:5)
 )
```

---

# 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,
  template = 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, the default template coordinates are used. |
| template | The kind of sensor template montage used. Available options are "HCGSN256", "biosemi128", "biosemi256", and "system1005". Default setting is "HCGSN256". |
| 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

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 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`) as either a 2D `ggplot` or 3D `rgl` plot depending on the requested 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,
  plot_dim = NULL
)
```

## Arguments

| | |
|---|---|
| mesh | An object of class "mesh" (typically returned by the point_mesh function), or another list with the same structure. It must contain dense mesh coordinates in D2 and/or D3 and a template string indicating the sensor montage used. |
| 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 the mesh and containing appropriate coordinate columns). If the value is NULL and sensors is set to TRUE, the template specified in mesh$template is used automatically. |
| plot_dim | A numeric value (2 or 3) specifying which dimension of the mesh to plot. If NULL (default), the function automatically plots 3D if available in the mesh object, otherwise it falls back to 2D. |

## Details

Please follow the instructions below when entering own_coordinates:

The output 2D 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 `D3` part with 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 when `plot_dim = 2`. Additionally, the returned ggplot object carries a `"diegr_metadata"` attribute with metadata such as the actual number of generated mesh points or the template used.

For `plot_dim = 3`, the mesh and sensors are drawn in the active `rgl` device.

## See Also

`point_mesh`

## Examples

```
# 2D polygon point mesh with all sensors from the BioSemi 128 template
# Note: for nice plot we recommend set par(mar = c(0,0,0,0))
M <- point_mesh(n = 4000, template = "biosemi128")
plot_point_mesh(M, plot_dim = 2)

## Note: the example opens a rgl 3D viewer
# Plotting 3D polygon point mesh with default settings
rgl::open3d()
plot_point_mesh(M, plot_dim = 3)

# 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, 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, 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 course of the average EEG signal amplitude with pointwise confidence intervals (CIs), colour-coded by a user-defined grouping variable such as experimental condition, subject or group. If the `condition_column` is `NULL`, all observations are treated as a single condition.

## Usage

```
plot_time_mean(
  data,
  condition_column = NULL,
  FS = 250,
  t0 = 1,
  transp = 0.4,
  y_limits = NULL,
  label_0ms = "stimulus",
  label_offset = c(0, 0),
  legend_title = "Condition"
)
```

## 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. |
| `condition_column` | Character string specifying the name of the column used to define conditions for plotting. If `NULL`, all observations are treated as a single condition. |
| `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). |
| `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 `NULL`for 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. |
| `legend_title` | Character string specifying the legend title shown in the plot. Default is `"Condition"`. For all observations treated as a single condition (`condition_column = NULL`) is plotted no legend. |

## Details

The output in the form of a ggplot object allows to easily edit the result image properties.

## Value

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

Additionally, the returned object carries a "diegr_metadata" attribute with metadata such as plot and data details.

## See Also

compute_mean, interactive version of time plot: interactive_waveforms

## Examples

```
# Plot average signal with CI bounds from the sensor E65 excluding outlier epochs (14 and 15)
# for subject 2 - part b) and for the both subjects treated as conditions - part c)

# a) preparing data
# a1) extract required data
edata <- pick_data(epochdata, sensor_rg = c("E65"), epoch_rg = 1:13)
# a2) baseline correction
data_base <- baseline_correction(edata, baseline_range = 1:9)
# a3) average computing
data_mean <- compute_mean(data_base, amplitude = "signal_base", type = "point")

# b) filter subject 2 and plot the average line with default settings
# (the whole dataset treated as one condition, no legend plotted)
data_mean2 <- data_mean |>
dplyr::filter(subject == 2) # or use pick_data(data_mean, subject_rg = 2)
plot_time_mean(data = data_mean2, t0 = 10)

# c) plot the time course by subject (treated as a condition)
plot_time_mean(data = data_mean, condition_column = "subject", t0 = 10, legend_title = "Subject")

# Plot average signal with CI bounds for subject 1 from three chosen sensors
# preparing data
edata <- pick_data(epochdata, subject_rg = 1, sensor_rg = c("E5", "E35" ,"E65"),
                   epoch_rg = 1:13)
data_base <- baseline_correction(edata, baseline_range = 1:9)
data_mean <- compute_mean(data_base, amplitude = "signal_base", type = "point")
# plot the time course by sensor (channel)
plot_time_mean(data = data_mean, condition_column = "sensor", t0 = 10, legend_title = "Channel")
```

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 a `D2` element with x and y coordinates of a point mesh used for computing the IM model, and a `template` element specifying the sensor montage. 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 template specified in `mesh$template` (or the default `"HCGSN256"`) is used. |
| `template` | The kind of sensor template montage used. Available options are `"HCGSN256"`, `"biosemi128"`, `"biosemi256"`, and `"system1005"`. Default setting is `"HCGSN256"`. |
| `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.

Notes: If a mesh object is provided, its internal template name (mesh$template) overrides the template argument to ensure spatial consistency. When custom coords are provided, they are always used for plotting the sensor locations. The template parameter (or mesh$template) is then used only for generating the background mesh if it is not provided.

## Value

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

Additionally, the returned object carries a "diegr_metadata" attribute with metadata such as details about the mesh used for plotting.

## See Also

topo_plot, compute_mean, animated version: animate_topo_mean

## 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 <- pick_data(epochdata, subject_rg = 2, epoch_rg = 1:13, time_rg = 1:10)
# a2) baseline correction (needed for suitable topographic map)
data_base <- baseline_correction(edata, baseline_range = 1:9)
# a3) average computing
data_mean <- data_base |>
dplyr::filter(time == 10) |>
compute_mean(amplitude = "signal_base", type = "jack", domain = "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 the sensor template montage used. Available options are "HCGSN256" (256-channel HydroCel Geodesic Sensor Net v.1.0), "biosemi128" (128-channel BioSemi), "biosemi256" (256-channel BioSemi), and "system1005" (the 10-05 system). |
| 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

Please note that the input coordinates must be unique. The function execution will automatically stop if duplicate sensor locations are found.

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

For the provided built-in templates, the (0,0) point of the resulting 2D mesh generally corresponds to the vertex of the head model (e.g., the Cz electrode).

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 2D layout; the function itself does not perform a projection onto a plane. 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.

Note on Matrix Conditioning: The interpolation matrix used in thin-plate spline interpolation can be ill-conditioned depending on the spatial configuration of input sensor locations. If this occurs, the function automatically uses a Moore-Penrose pseudoinverse (`MASS::ginv()`) instead of the standard matrix inverse to improve numerical stability and issues a warning.

## 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. |

Additionally, the returned object carries a `"diegr_metadata"` attribute with metadata such as the actual number of generated mesh points or the template used.

## References

EGI Geodesic Sensor Net Technical Manual (2024) Oostenveld, R., & Praamstra, P. (2001). The five percent electrode system for high-resolution EEG and ERP measurements. *Clinical Neurophysiology*, 112(4), 713-719. doi:10.1016/s13882457(00)005277

## 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")
# Inspect the attached metadata
attr(M, "diegr_metadata")

# Computing polygon 3D mesh with starting number 2000 points for BioSemi 128 template
M_bio <- point_mesh(dimension = 3, n = 2000, template = "biosemi128")

# 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.

The function assumes that the input data have already been filtered to the desired subset (e.g., group, subject, time point).

## 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 | A "mesh" object (or a named list with the same structure) containing at least a D3 element with x, y and z coordinates of a point mesh used for computing the IM model, and a template element specifying the sensor montage. 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 template specified in mesh$template (or the default "HCGSN256") is used. |
| template | The kind of sensor template montage used. Available options are "HCGSN256", "biosemi128", "biosemi256", and "system1005". Default setting is "HCGSN256". |
| 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, D3 data frame with x, y and z columns and template element specifying the used template. 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: This function focuses on visualization and does not perform any data subsetting. Users are expected to filter the data beforehand using standard dplyr verbs or pick_data function.

If a mesh object is provided, its internal template name (mesh$template) overrides the template argument to ensure spatial consistency. When custom coords are provided, they are used for plotting the sensor locations. The template parameter (or mesh$template) is then used only for generating the background mesh if it is not provided.

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().

## Value

Called for its side effect of drawing a 3D scalp map in an interactive rgl window. It returns an object of class "rglLowLevel" (an integer vector) containing the IDs of the drawn objects.

Additionally, the returned object carries a "diegr_metadata" attribute with metadata such as details about the mesh used for plotting.

## See Also

point_mesh, make_triangulation, create_scale, animated version: animate_scalp

## 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 <- pick_data(epochdata, subject_rg = 2, epoch_rg = 1:13, time_rg = 1:10)
# a2) baseline correction (needed for suitable topographic map)
data_base <- baseline_correction(edata, baseline_range = 1:9)
# a3) average computing
data_mean <- data_base |>
dplyr::filter(time == 10) |>
compute_mean(amplitude = "signal_base", type = "point", domain = "space")

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

---

# summary_stats_rt *Compute summary statistics of reaction times*

---

## Description

Calculates basic descriptive statistics of reaction time (RT). Statistics are computed separately for each combination of grouping variables present in the data (e.g., group, subject, condition).

Computed statistics include: the number of epochs, minimum, maximum, median, mean, and standard deviation of RT.

## Usage

```
summary_stats_rt(data)
```

## Arguments

| | |
|---|---|
| `data` | A data frame or a database table with reaction times dataset. Required columns are `epoch` and `RT` (value of reaction time in ms). Optional columns: `group`, `subject`, `condition` for computing summary statistics per group/subject/condition. |

## Value

A tibble with summary statistics of reaction times consisting of the following columns:

**group** Group identifier (only if present in the input data).

**subject** Subject identifier (only if present in the input data).

**condition** Experimental condition (only if present in the input data).

**n_epoch** Number of epochs.

**min_rt** Minimum reaction time.

**max_rt** Maximum reaction time.

**median_rt** Median reaction time.

**avg_rt** Mean reaction time.

**sd_rt** Standard deviation of reaction time.

## Examples

```
# 1. Summary statistics for rtdata
# two different subjects, no group or conditions - results are computed per subject
summary_stats_rt(rtdata)

# 2. Summary statistics for data with conditions
# a) create example data
data_cond <- rtdata
data_cond$condition <- c(rep("a", 7), rep("b", 7), rep("a", 8), rep("b",7))
# b) compute statistics per subject and condition
summary_stats_rt(data_cond)
# c) compute statistics per conditions regardless of subjects
# exclude "subject" column from computing
summary_stats_rt(data_cond[,-1])
```

`system1005` *Coordinates of the spherical 10-05 system sensors*

## Description

A dataset containing the Cartesian coordinates of high-density EEG sensor positions of the 10-05 system in 3D space on an idealized spherical surface, along with their corresponding positions in 2D space. This template contains 335 matched electrode positions for both the 3D space and the 2D layout. Fiducial markers and extremely low electrodes (such as those on the lower neck and cheeks) originally present in the full 10-05 system are excluded from both dimensions to maintain visual clarity and structural consistency in topographical modeling.

## Usage

```
data("system1005")
```

## Format

A list with the following elements:

**D2** A tibble with 3 columns containing x and y coordinates and sensor labels (according to the standard 10-05 naming convention) in 2D.

**D3** A tibble with 4 columns containing x, y and z coordinates and sensor labels (according to the standard 10-05 naming convention) in 3D. See 'Details' for more information.

**ROI** A factor containing the name of the region to which the corresponding sensor belongs. The levels are: "central", "frontal", "occipital", "parietal", and "temporal". See Details for more information about assigning sensors to locations.

## 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 (+). Because this is an idealized spherical model, the origin (0, 0, 0) represents the geometric center of the sphere. The electrodes are symmetrically distributed across this perfect spherical surface and the electrode Cz is located exactly at the apex of the sphere at coordinates (0, 0, 1).

The electrodes are mapped to anatomical regions (`ROI`) based on their name prefixes. Boundary electrodes are assigned to their closest primary region (e.g., 'FC' and 'FCC' are grouped under Central, while 'FFC' is Frontal; 'PO' is assigned to Occipital, and 'CP' to Parietal).

The coordinates originate from the MNE-Python GitHub repository.

## Source

The MNE-Python GitHub repository, https://github.com/mne-tools/mne-python/tree/main/mne/channels/data/montages

## References

Oostenveld, R., Praamstra, P. (2001). The five percent electrode system for high-resolution EEG and ERP measurements. *Clinical Neurophysiology*, 112(4), 713-719. doi:10.1016/s13882457(00)00527-7

## Examples

```
# A simple plot of sensor coordinates from 10-05 system spherical template as points in 2D
plot(system1005$D2[,1:2], pch = 16, asp = 1)
```

---

`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.

The function assumes that the input data have already been filtered to the desired subset (e.g., group, subject, time point).

## 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 a D2 element with x and y coordinates of a point mesh used for computing the IM model, and a template element specifying the sensor montage. 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 template specified in mesh$template (or the default "HCGSN256") is used. |
| template | The kind of sensor template montage used. Available options are "HCGSN256", "biosemi128", "biosemi256", and "system1005". Default setting is "HCGSN256". |
| 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. Default value is TRUE. |
| label_sensors | A logical value indicating whether the sensor labels should also be plotted. Default 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 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: If a mesh object is provided, its internal template name (mesh$template) overrides the template argument to ensure spatial consistency.

When custom coords are provided, they are always used for plotting the sensor locations. The template parameter (or mesh$template) is then used only for generating the background mesh if it is not provided.

This function focuses on visualization and does not perform any data subsetting. Users are expected to filter the data beforehand using standard dplyr verbs or pick_data function.

## Value

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

Additionally, the returned object carries a "diegr_metadata" attribute with metadata such as details about the mesh used for plotting.

## See Also

point_mesh, animated version: animate_topo, average topo map: 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 <- pick_data(epochdata, subject_rg = 2, epoch_rg = 1:13, time_rg = 1:10)
# a2) baseline correction (needed for suitable topographic map)
data_base <- baseline_correction(edata, baseline_range = 1:9)
# a3) average computing
data_mean <- data_base |>
dplyr::filter(time == 10) |>
compute_mean(amplitude = "signal_base", type = "jack", domain = "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


∗ **dataset**
biosemi128, 11
biosemi256, 12
epochdata, 23
HCGSN256, 24
rtdata, 45
system1005, 50

animate_scalp, 2, *6*, *48*
animate_topo, *4*, 5, *8*, *53*
animate_topo_mean, 7, *42*

baseline_correction, *8*, 9, *34*
biosemi128, 11
biosemi256, 12
boxplot_epoch, 13, *16*
boxplot_rt, 14
boxplot_subject, *14*, 15

check_structure, 17
compute_mean, *7, 8*, 18, *29*, *34*, *39–42*
create_scale, 21, *42*, *48*, *52*

epochdata, 23, *46*

geom_ribbon, *39*
gganimate::animate, *6*, *8*

HCGSN256, 24

interactive_surfaceplot, 25, *27*
interactive_surfaceplot_curves, 26, *26*
interactive_waveforms, 28, *40*

make_triangulation, *3*, 29, *47, 48*

outliers_epoch, 31

pick_data, 33, *48*, *52*
pick_region, *34*, 35
plot_point_mesh, 36
plot_time_mean, 39
plot_topo_mean, *8*, *27*, 41, *53*
plotly, *28*
point_mesh, *3–6*, *8*, *25*, *29*, *36–38*, *41*, 43, *47, 48*, *52, 53*

rtdata, 45

scale_fill_gradientn, *22*
scalp_plot, *4*, *26*, 46
shape3d, *46*
summary_stats_rt, 48
system1005, 50

topo_plot, *6*, *42*, 51