

Package: ravebuiltins (via r-universe)

October 15, 2026

Type Package

Title Builtin Modules for 'RAVE'

Version 1.99.0

Description This package provides builtin modules for 'RAVE'. It aims to analyze and visualize 'iEEG' data from different perspectives.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

Depends R (>= 4.1.0)

Imports methods, stats, grDevices, graphics, utils, data.table, rave, ravedash, ravepipeline, ravecore, ravetools, magrittr (>= 1.5), circular (>= 0.4-93), lmerTest (>= 3.1-0), emmeans (>= 1.4.8), rlang (>= 0.3.0), shiny (>= 1.2.0), rutabaga

Suggests fastmap, rstudioapi (>= 0.9.0)

Remotes github::dipterix/rutabaga, github::beauchamplab/rave

Additional_repositories <https://rave-ieeg.r-universe.dev>

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake libfftw3-dev git libglpk-dev make libicu-dev libpng-dev libuv1-dev libxml2-dev libssl-dev python3 libzmq3-dev zlib1g-dev

Repository <https://rave-ieeg.r-universe.dev>

Date/Publication 2026-08-24 03:58:25 UTC

RemoteUrl <https://github.com/beauchamplab/ravebuiltins>

RemoteRef migrate3

RemoteSha 880119c1b4c79a5dffc36895d04081f8bf660f6e

Contents

by_condition_welch	2
by_trial_erp_map	2
draw_cut_point	3
draw_many_heat_maps	3
erp_over_time_plot	5
expand_heatmap	5
get_heatmap_palette	6
get_palette	7
make_label_with_count	7
over_time_plot	8
plot_grouped_data	8
rave_title	10
remove_count_from_label	10
remove_gyrus_sulcus_labels	11
remove_hemisphere_labels	12

Index	13
-------	----

by_condition_welch	Welch <i>periodogram per condition</i>
--------------------	--

Description

Welch periodogram per condition

Usage

by_condition_welch(results, ...)

Arguments

results	results returned by module
...	other parameters passed to module output

by_trial_erp_map	<i>By Trial Plot for ERP data</i>
------------------	-----------------------------------

Description

By Trial Plot for ERP data

Usage

by_trial_erp_map(results, ...)

Arguments

results	results returned by module
...	other parameters passed to module output

draw_cut_point	<i>Draws an orange, dashed horizontal line at cut. Checks for not null and length > 0</i>
----------------	--

Description

Draws an orange, dashed horizontal line at cut. Checks for not null and length > 0

Usage

```
draw_cut_point(cut = NULL)
```

Arguments

cut	the location(s) of the lines
-----	------------------------------

Value

the value of cut (invisibly)

draw_many_heat_maps	<i>Draw several heatmaps in a row and (optionally) a color bar</i>
---------------------	--

Description

Easy way to make a bunch of heatmaps with consistent look/feel and get a color bar. By default it is setup for time/freq, but by swapping labels and decorators you can do anything.

Usage

```
draw_many_heat_maps(
  hmaps,
  max_zlim = 0,
  percentile_range = FALSE,
  log_scale = FALSE,
  show_color_bar = TRUE,
  useRaster = TRUE,
  wide = FALSE,
  PANEL.FIRST = NULL,
  PANEL.LAST = NULL,
  PANEL.COLOR_BAR = NULL,
```

```

axes = c(TRUE, TRUE),
plot_time_range = NULL,
special_case_first_plot = FALSE,
max_columns = 2,
decorate_all_plots = FALSE,
center_multipanel_title = FALSE,
ignore_time_range = NULL,
marginal_text_fields = c("Subject ID", "Electrode", "Frequency"),
extra_plot_parameters = NULL,
do_layout = TRUE,
...
)

```

Arguments

<code>hmaps</code>	data to draw heatmaps
<code>max_zlim</code>	<code>zlim</code> that trims <code>z</code> value
<code>percentile_range</code>	whether to draw in percentile
<code>log_scale</code>	draw <code>y</code> in log scale?
<code>show_color_bar</code>	show color legend to the right? Future: Will will check to see if this parameter is a function. If so, we can call it to allow arbitrary legends in the right-most (half) panel
<code>useRaster, ...</code>	passed to <code>image()</code>
<code>wide</code>	boolean. should we use a wider margin on the left? defaults to false
<code>PANEL.FIRST</code>	a function that is called after each plot window has been created, but before any rendering is done. In truth, this is currently called AFTER the call to <code>image()</code> , so if you draw within the plotting region it will overwrite the heatmap. To fix this requires editing <code>make_image</code> to allow for a function to be called after creation but before rendering. Don't depend on this call order, use <code>PANEL.LAST</code> if you want to draw things on top of the heatmap
<code>PANEL.LAST</code>	a function that is called after the rendering of each heat map. It is not called after the rendering of the color bar.
<code>PANEL.COLOR_BAR</code>	a function to adjust color bar width
<code>axes</code>	vector of logicals, whether to draw axis
<code>plot_time_range</code>	<code>x</code> range, similar to <code>xlim</code>
<code>special_case_first_plot</code>	logical, whether the first panel gets its own treatment (for example a wider left margin) instead of the shared layout
<code>max_columns</code>	maximum number of heatmap columns before wrapping onto additional rows (default 2)
<code>decorate_all_plots</code>	logical, whether <code>PANEL.FIRST</code> / <code>PANEL.LAST</code> run on every panel rather than only the first

center_multipanel_title	logical, whether a multi-panel title is centered across the whole figure instead of over the first panel
ignore_time_range	optional time range to exclude from the drawn data
marginal_text_fields	character vector naming which fields are written into the plot margin (default Subject ID, Electrode, Frequency)
extra_plot_parameters	optional named list of additional graphical parameters forwarded to the individual heatmap panels
do_layout	logical, whether to call layout_heat_maps to build the panel layout. Set FALSE when the caller manages the layout.

Value

Called for its side effect of drawing the heatmaps; returns NULL invisibly.

erp_over_time_plot	<i>Voltage Time Series Plot</i>
--------------------	---------------------------------

Description

Voltage Time Series Plot

Usage

```
erp_over_time_plot(results, ...)
```

Arguments

results	results returned by module
...	other parameters passed to module output

expand_heatmap	<i>Interpolate a heatmap palette to a given number of colors</i>
----------------	--

Description

Expands a short heatmap palette into a smooth color ramp of ncolors entries, suitable for passing to image or to a threeBrain viewer.

Usage

```
expand_heatmap(pal, results, ncolors = 101, space = "Lab", ...)
```

Arguments

pal	a character vector of colors, or the name of one of the builtin palettes returned by <code>get_heatmap_palette</code>
results	optional RAVE results object; when supplied, ncolors and the palette name are read from it instead
ncolors	number of colors in the returned ramp (default 101)
space	color space used for interpolation, passed to <code>grDevices::colorRampPalette</code> (default 'Lab')
...	further arguments passed to <code>grDevices::colorRampPalette</code> , for example bias

Value

A character vector of ncolors hex color strings.

See Also

`get_heatmap_palette`

Examples

```
expand_heatmap(get_heatmap_palette('BlueWhiteRed'), ncolors = 101)
```

get_heatmap_palette	<i>Builtin diverging heatmap palettes</i>
---------------------	---

Description

Returns one of the builtin diverging color palettes used for **RAVE** heatmaps. Most are adapted from <https://colorbrewer2.org>. Each palette has an odd number of colors so that the middle entry is the neutral midpoint.

Usage

```
get_heatmap_palette(pname, get_palettes = FALSE, get_palette_names = FALSE)
```

Arguments

pname	palette name, one of the values returned by this function with <code>get_palette_names = TRUE</code> . If missing, all palettes are returned. An unrecognized name warns and falls back to a random palette.
get_palettes	ignored, kept for backward compatibility
get_palette_names	whether to return the available palette names instead of a palette; only used when pname is missing

Value

A character vector of hex colors carrying a name attribute; or, when pname is missing, the full named list of palettes, or a character vector of palette names if get_palette_names = TRUE.

See Also

[expand_heatmap](#)

Examples

```
get_heatmap_palette(get_palette_names = TRUE)
get_heatmap_palette('BlueWhiteRed')
```

get_palette	<i>Function to get builtin color palettes</i>
-------------	---

Description

Function to get builtin color palettes

Usage

```
get_palette(pname, get_palettes = FALSE, get_palette_names = FALSE)
```

Arguments

pname	palette name
get_palettes	ignored
get_palette_names	whether to get palette names

make_label_with_count	<i>Append the electrode count to each label</i>
-----------------------	---

Description

Collapses a vector of electrode labels to its unique values and appends the number of electrodes carrying each label, as Label[n].

Usage

```
make_label_with_count(lbl)
```

Arguments

lbl	character vector of per-electrode labels, with repeats
-----	--

Value

A character vector of the unique labels with counts appended. The result carries a map attribute giving, for each element of lbl, the index of its label in the returned vector.

See Also

[remove_count_from_label](#)

Examples

```
make_label_with_count(c('G 1', 'G 1', 'S 2'))
```

over_time_plot	<i>Time Series Plot</i>
----------------	-------------------------

Description

Time Series Plot

Usage

```
over_time_plot(results, ...)
```

Arguments

results	results returned by module
...	other parameters passed to module output

plot_grouped_data	<i>Plot grouped data as points, means and error bars</i>
-------------------	--

Description

This doesn't do any decoration, it's designed for use with `rutabaga::create_frames`. Note that we're using `barplot` to set the x- and y-range of the plot. Does not handle log axes correctly.

Usage

```
plot_grouped_data(
  mat,
  yvar,
  xvar,
  gvar = NULL,
  types = c("jitter points", "means", "ebar polygons"),
  layout = c("grouped", "overlay"),
  draw0 = TRUE,
  draw0.col = "black",
  ylim = NULL,
  col = NULL,
  ...,
  plot_options = NULL,
  jitter_seed
)
```

Arguments

<code>mat</code>	a <code>data.frame</code> or <code>data.table</code> holding the data to plot
<code>yvar</code>	name of the column holding the response values
<code>xvar</code>	name of the column defining positions along the x axis
<code>gvar</code>	name of the grouping column, or <code>NULL</code> for no grouping
<code>types</code>	character vector selecting which layers to draw; any of 'jitter points', 'means' and 'ebar polygons'
<code>layout</code>	whether groups are drawn side by side ('grouped') or on top of one another ('overlay')
<code>draw0</code>	logical, whether to draw a reference line at zero
<code>draw0.col</code>	color of the zero reference line (default black)
<code>ylim</code>	y axis limits; computed from the data when <code>NULL</code>
<code>col</code>	vector of colors, one per group; defaults to the active palette
<code>...</code>	extra options to pass to <code>barplot</code> during plot creation
<code>plot_options</code>	optional named list of graphical parameters such as <code>pch</code> and <code>lwd</code> , overriding the defaults
<code>jitter_seed</code>	optional integer seed making the point jitter reproducible across redraws

Value

Called for its side effect of drawing the plot; returns `NULL` invisibly.

rave_title	<i>Draw a plot title that respects the RAVE dark mode</i>
------------	--

Description

Wraps `graphics::title`, defaulting the color to the current foreground so titles stay legible when `par('bg')` is dark, and shrinking the text when the plot is being written to a file.

Usage

```
rave_title(main, cex = rave_cex.main, col, font = 1, adj = 0.5, ...)
```

Arguments

<code>main</code>	the title text, passed to <code>graphics::title</code>
<code>cex</code>	the character expansion for the title (default is <code>rave_cex.main</code>)
<code>col</code>	title color; defaults to the current foreground color
<code>font</code>	the font type (default = 1, plain)
<code>adj</code>	horizontal justification, 0 left, 0.5 centered, 1 right
<code>...</code>	further arguments passed to <code>graphics::title</code>

Value

Called for its side effect of drawing the title; returns the value of `graphics::title` invisibly.

See Also

[title](#)

remove_count_from_label	<i>Strip the electrode count from a label</i>
-------------------------	---

Description

Removes the trailing [5] / [10] electrode count that [make_label_with_count](#) appends to **FreeSurfer** labels.

Usage

```
remove_count_from_label(str)
```

Arguments

<code>str</code>	character vector of labels
------------------	----------------------------

Value

The labels with any trailing bracketed count removed.

See Also

[make_label_with_count](#)

Examples

```
remove_count_from_label('G_temp_sup[5]')
```

```
remove_gyrus_sulcus_labels
```

Strip anatomical region prefixes from a label

Description

Removes the leading GS , G and S gyrus/sulcus markers from **FreeSurfer** anatomical labels.

Usage

```
remove_gyrus_sulcus_labels(str)
```

Arguments

str character vector of labels

Value

The labels with gyrus/sulcus prefixes removed.

See Also

[remove_hemisphere_labels](#)

Examples

```
remove_gyrus_sulcus_labels('G temp_sup')
```

`remove_hemisphere_labels`*Strip hemisphere prefixes from a label*

Description

Removes the leading L / R hemisphere markers so that labels are compared consistently across hemispheres. Kept in one place so every caller strips them the same way.

Usage

```
remove_hemisphere_labels(str)
```

Arguments

`str` character vector of labels

Value

The labels with hemisphere prefixes removed.

See Also

[remove_gyrus_sulcus_labels](#)

Examples

```
remove_hemisphere_labels('L G_temp_sup')
```

Index

by_condition_welch, [2](#)
by_trial_erp_map, [2](#)

draw_cut_point, [3](#)
draw_many_heat_maps, [3](#)

erp_over_time_plot, [5](#)
expand_heatmap, [5](#), [7](#)

get_heatmap_palette, [6](#), [6](#)
get_palette, [7](#)

make_label_with_count, [7](#), [10](#), [11](#)

over_time_plot, [8](#)

plot_grouped_data, [8](#)

rave_title, [10](#)
remove_count_from_label, [8](#), [10](#)
remove_gyrus_sulcus_labels, [11](#), [12](#)
remove_hemisphere_labels, [11](#), [12](#)

title, [10](#)