Package: ravebuiltins (via r-universe)

June 29, 2024

Type Package

Title Builtin Modules for `RAVE`

Version 1.0.5

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

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.1

Imports methods, stats, grDevices, graphics, grid, abind, car, utils, data.table, dipsaus, threeBrain (>= 0.1.3), magrittr (>= 1.5), circular (>= 0.4-93), lmerTest (>= 3.1-0), emmeans (>= 1.4.8), lme4 (>= 1.1.26), rlang (>= 0.3.0), stringr (>= 1.3.1), shiny (>= 1.2.0), knitr, assertthat, shinyjs, shinyFiles, reshape2, digest, DT, fst, rave, raveio, rutabaga

Suggests devtools, htmltools, fastmap, lsmeans, rstudioapi (>= 0.9.0), yaml (>= 2.2.0), future (>= 0.14.0)

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

Repository https://dipterix.r-universe.dev

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

RemoteRef migrate2

RemoteSha 76e476ee2f44184695c4a939b0b068c35f9af8b3

Contents

`by_condition_welch` *Welch periodogram per condition*

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` *By Trial Plot for ERP data*

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

dev_ravebuiltins *Function to load all dev funtions and wrap them within an environment*

Description

Function to load all dev funtions and wrap them within an environment

Usage

```
dev_ravebuiltins(expose_functions = FALSE, reload = TRUE)
```

Arguments

expose_functions	logical indicating whether to expose all dev functions to the global environment
reload	logical, do you want to fast-reload the package before load the functions?

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

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` *Draw several heatmaps in a row and (optionally) a color bar*

Description

Easy way to make a bunch of heatmaps with consistent look/feel and get a colorbar. 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

`hmaps`	data to draw heatmaps
`max_zlim`	zlim that trims z value
`percentile_range`	whether to draw in percentile
`log_scale`	draw y in log scale?
`show_color_bar`	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
`useRaster, ...`	passed to image()

wide	boolean. should we use a wider margin on the left? defaults to false
PANEL.FIRST	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 image(), so if you draw within the plotting region it will overwrite the heatmap. To fix this requires editing draw_img(...) to allow for a function to be called after creation but before rendering. Don't depend on this call order, use PANEL.LAST if you want to draw things on top of the heatmap
PANEL.LAST	a function that is called after the rendering of each heat map. It is not called after the rendering of the color bar.
PANEL.COLOR_BAR	a function to adjust colorbar width
axes	vector of logicals, whether to draw axis
plot_time_range	x range, similar to xlim

See Also

layout_heat_maps

draw_img

---

easy_layout *Create a easy layout for multiple plots sharing the same x,y and legend*

Description

Provide easy ways to set plot layouts

Usage

```
easy_layout(
  K,
  nrows = 1,
  legend,
  legend_size = lcm(3),
  legend_side = 4,
  s_margin = par("mar"),
  b_margin = par("oma"),
  l_margin
)
```

Arguments

K	number of plots to be made
nrows	number of rows for the plot, default 1
legend	expression for generating legend, see "?legend"

legend_size	legend width/height, default is lcm(3)
legend_side	1 - bottom, 2 - left, 3 - top, 4 - right. Default is 4
s_margin	margins within each plots see "?par" for "mar"
b_margin	margins for the whole plot see "?par" for "oma"
l_margin	legend margin

`erp_over_time_plot` *Voltage Time Series Plot*

Description

Voltage Time Series Plot

Usage

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

Arguments

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

`get_palette` *Function to get builtin color palettes*

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

heat_map_plot *Basic Time Frequency Plot*

Description

Basic Time Frequency Plot

Usage

```
heat_map_plot(results, ...)
```

Arguments

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

Examples

```
## Not run:
rave_prepare(...)
fn = ravebuiltins:::get_module('power_explorer')
res = fn()
heat_map_plot(res$result)

## End(Not run)
```

layout_heat_maps *layout_heat_map*

Description

Create a layout so that heatmaps look nice and you have enough space for the color bar

Usage

```
layout_heat_maps(
  k,
  max_col,
  ratio = 4,
  layout_color_bar = TRUE,
  colorbar_cm = 3.5
)
```

Arguments

colorbar_cm	The default width chosen (3.5) for the color bar relies on 'lcm'. If the function detects the user is writing to a file (@seealso plotting_to_file), the width is currently forced to 3.0
k:	the number of heatmaps, excluding the color bar
max_col:	maximum number of columns before creating multiple rows
ratio:	heatmap to color bar width ratio (Default 4:1)
layout_color_bar:	whether space should be made for the color bar (Default TRUE)

---

make_image *RAVE custom image plotter*

Description

The idea here is to to separate the plotting of the heatmap from all the accoutrements that are done in the decorators. We are just plotting image(mat) Rather Than t(mat) as you might expect. The Rave_calculators know this so we can save a few transposes along the way.

Usage

```
make_image(
  mat,
  x,
  y,
  zlim,
  col = NULL,
  log = "",
  useRaster = TRUE,
  clip_to_zlim = TRUE,
  add = TRUE
)
```

Arguments

mat	z-matrix
x, y	z and y axis
zlim	value to trim zmat
col	vector of colors, color palette
log	which axis will be in log scale
useRaster	passed to image()
clip_to_zlim	whether to clip mat
add	logical, whether to overlay current plot to an existing image

`over_time_plot` *Time Series Plot*

Description

Time Series Plot

Usage

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

Arguments

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

`plot_grouped_data` *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. Note Does not handle log axes correctly param ... extra options to pass to barplot during plot creation*

Description

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. Note Does not handle log axes correctly param ... extra options to pass to barplot during plot creation

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
)
```

rave_title *Function make a title for a plot, checks par('bg') to handle dark mode*

Description

Function make a title for a plot, checks par('bg') to handle dark mode

Usage

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

Arguments

cex	the character expansion for the title (default is rave_cex.main)
font	the font type (default = 1, plain)

See Also

title

windowed_comparison_plot *By Trial Plot With Statistics*

Description

By Trial Plot With Statistics

Usage

```
windowed_comparison_plot(results, ...)
```

Arguments

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

Index