Package: ggeo (via r-universe)

October 15, 2026

Type Package

Title Themes and Helpers for ggplot2

Version 1.0.0

Description This package provides helper functions for ggplot graphs and maps.

License MIT + file LICENSE

URL https://ggeo.nenuial.org

BugReports http://github.com/nenuial/ggeo/issues/

Depends R (>= 4.1)

Imports colorspace, fs, geotools, ggplot2, glue, grDevices, grid, highcharter, lifecycle, magrittr, methods, paletteer, PNWColors, prismatic, purrr, rlang, scales, showtext, stringr, sysfonts, utils, wesanderson

Suggests rnaturalearth

Remotes Nenuial/geotools

Config/Needs/website nenuial/geopkg

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libabsl-dev cmake libfreetype6-dev libgdal-dev gdal-bin libgeos-dev libglpk-dev make libicu-dev libpng-dev libuv1-dev libxml2-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev libx11-dev zlib1g-dev

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

Date/Publication 2026-10-15 19:34:43 UTC

RemoteUrl https://github.com/Nenuial/ggeo

RemoteRef HEAD

RemoteSha 2cd225644ae163f59504bb21d17f0b4f37d37685

Contents

ggeo_capitalize_title . . . 2
ggeo_coord . . . 3
ggeo_label_sci_10 . . . 7
ggeo_remove_breaks . . . 8
ggeo_remove_title . . . 8
ggeo_save . . . 9
ggeopal_center . . . 10
ggeopal_chooser . . . 11
ggeopal_hex_to_hcl . . . 11
ggeopal_to_gradient . . . 12
ggeotheme . . . 12
hc_purple_theme . . . 17

Index **19**

`ggeo_capitalize_title` *Copitalize plot title*

Description

Copitalize plot title

Usage

```
ggeo_capitalize_title(plot)
```

Arguments

`plot`	A ggplot2 object

Value

A ggplot2 object

Examples

```
ggplot(aes(x = speed, y = dist), data = cars) +
  geom_point() +
  labs(title = "Fast cars !") +
  ggeo_remove_title()
```

ggeo_coord *Return sf coord for given CRS code*

Description

A function that uses `geotools::gtl_crs_proj()` to get the CRS configuration and applies it to a ggplot2 map.

Usage

```
ggeo_coord(code, ...)
```

Arguments

`code`	A string with the CRS identifier
`...`	Arguments passed on to `ggplot2::coord_sf`

`xlim,ylim` Limits for the x and y axes. These limits are specified in the units of the default CRS. By default, this means projected coordinates (`default_crs = NULL`). How limit specifications translate into the exact region shown on the plot can be confusing when non-linear or rotated coordinate systems are used as the default crs. First, different methods can be preferable under different conditions. See parameter `lims_method` for details. Second, specifying limits along only one direction can affect the automatically generated limits along the other direction. Therefore, it is best to always specify limits for both x and y. Third, specifying limits via position scales or `xlim()`/`ylim()` is strongly discouraged, as it can result in data points being dropped from the plot even though they would be visible in the final plot region.

`expand` If `TRUE`, the default, adds a small expansion factor to the limits to ensure that data and axes don't overlap. If `FALSE`, limits are taken exactly from the data or `xlim`/`ylim`. Giving a logical vector will separately control the expansion for the four directions (top, left, bottom and right). The `expand` argument will be recycled to length 4 if necessary. Alternatively, can be a named logical vector to control a single direction, e.g. `expand = c(bottom = FALSE)`.

`crs` The coordinate reference system (CRS) into which all data should be projected before plotting. If not specified, will use the CRS defined in the first sf layer of the plot.

`default_crs` The default CRS to be used for non-sf layers (which don't carry any CRS information) and scale limits. The default value of `NULL` means that the setting for `crs` is used. This implies that all non-sf layers and scale limits are assumed to be specified in projected coordinates. A useful alternative setting is `default_crs = sf::st_crs(4326)`, which means x and y positions are interpreted as longitude and latitude, respectively, in the World Geodetic System 1984 (WGS84).

`datum` CRS that provides datum to use when generating graticules.

label_graticule Character vector indicating which graticule lines should be labeled where. Meridians run north-south, and the letters "N" and "S" indicate that they should be labeled on their north or south end points, respectively. Parallels run east-west, and the letters "E" and "W" indicate that they should be labeled on their east or west end points, respectively. Thus, label_graticule = "SW" would label meridians at their south end and parallels at their west end, whereas label_graticule = "EW" would label parallels at both ends and meridians not at all. Because meridians and parallels can in general intersect with any side of the plot panel, for any choice of label_graticule labels are not guaranteed to reside on only one particular side of the plot panel. Also, label_graticule can cause labeling artifacts, in particular if a graticule line coincides with the edge of the plot panel. In such circumstances, label_axes will generally yield better results and should be used instead.

This parameter can be used alone or in combination with label_axes.

label_axes Character vector or named list of character values specifying which graticule lines (meridians or parallels) should be labeled on which side of the plot. Meridians are indicated by "E" (for East) and parallels by "N" (for North). Default is "--EN", which specifies (clockwise from the top) no labels on the top, none on the right, meridians on the bottom, and parallels on the left. Alternatively, this setting could have been specified with list(bottom = "E", left = "N").

This parameter can be used alone or in combination with label_graticule.

lims_method Method specifying how scale limits are converted into limits on the plot region. Has no effect when default_crs = NULL. For a very nonlinear CRS (e.g., a perspective centered around the North pole), the available methods yield widely differing results, and you may want to try various options. Methods currently implemented include "cross" (the default), "box", "orthogonal", and "geometry_bbox". For method "cross", limits along one direction (e.g., longitude) are applied at the midpoint of the other direction (e.g., latitude). This method avoids excessively large limits for rotated coordinate systems but means that sometimes limits need to be expanded a little further if extreme data points are to be included in the final plot region. By contrast, for method "box", a box is generated out of the limits along both directions, and then limits in projected coordinates are chosen such that the entire box is visible. This method can yield plot regions that are too large. Finally, method "orthogonal" applies limits separately along each axis, and method "geometry_bbox" ignores all limit information except the bounding boxes of any objects in the geometry aesthetic.

ndiscr Number of segments to use for discretising graticule lines; try increasing this number when graticules look incorrect.

default Is this the default coordinate system? If FALSE (the default), then replacing this coordinate system with another one creates a message alerting the user that the coordinate system is being replaced. If TRUE, that warning is suppressed.

clip Should drawing be clipped to the extent of the plot panel? A setting of "on" (the default) means yes, and a setting of "off" means no. In most

cases, the default of "on" should not be changed, as setting clip = "off" can cause unexpected results. It allows drawing of data points anywhere on the plot, including in the plot margins. If limits are set via xlim and ylim and some data points fall outside those limits, then those data points may show up in places such as the axes, the legend, the plot title, or the plot margins.

reverse A string giving which directions to reverse. "none" (default) keeps directions as is. "x" and "y" can be used to reverse their respective directions. "xy" can be used to reverse both directions.

mapping Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.

data The data to be displayed in this layer. There are three options:
If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().
A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).

stat The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following:

- A Stat ggproto subclass, for example StatCount.
- A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count".
- For more information and other ways to specify the stat, see the layer stat documentation.

position A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:

- The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter".
- For more information and other ways to specify the position, see the layer position documentation.

na.rm If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

show.legend logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.

You can also set this to one of "polygon", "line", and "point" to override the default legend.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `annotation_borders()`.

`parse` If `TRUE`, the labels will be parsed into expressions and displayed as described in `?plotmath`.

`label.padding` Amount of padding around label. Defaults to 0.25 lines.

`label.r` Radius of rounded corners. Defaults to 0.15 lines.

`label.size` **[Deprecated]** Replaced by the `linewidth` aesthetic. Size of label border, in mm.

`border.colour,border.color` Colour of label border. When `NULL` (default), the `colour` aesthetic determines the colour of the label border. `border.color` is an alias for `border.colour`.

`text.colour,text.color` Colour of the text. When `NULL` (default), the `colour` aesthetic determines the colour of the text. `text.color` is an alias for `text.colour`.

`fun.geometry` A function that takes a `sfc` object and returns a `sfc_POINT` with the same length as the input. If `NULL`, `function(x) sf::st_point_on_surface(sf::st_zm(x))` will be used. Note that the function may warn about the incorrectness of the result if the data is not projected, but you can ignore this except when you really care about the exact locations.

`check_overlap` If `TRUE`, text that overlaps previous text in the same layer will not be plotted. `check_overlap` happens at draw time and in the order of the data. Therefore data should be arranged by the label column before calling `geom_text()`. Note that this argument is not supported by `geom_label()`.

`geom` The geometric object to use to display the data for this layer. When using a `stat_*()` function to construct a layer, the `geom` argument can be used to override the default coupling between stats and geoms. The `geom` argument accepts the following:

- A `Geom` ggproto subclass, for example `GeomPoint`.
- A string naming the geom. To give the geom as a string, strip the function name of the `geom_` prefix. For example, to use `geom_point()`, give the geom as `"point"`.
- For more information and other ways to specify the geom, see the layer geom documentation.

Value

A ggplot2 coord object

See Also

`geotools::gtl_crs_proj()`

Examples

```
rnaturalearth::ne_countries() |>
  ggplot2::ggplot() +
  ggplot2::geom_sf() +
  ggeo_coord("eqearth")
```

`ggeo_label_sci_10` *Format labels*

Description

These functions extend the scales package and allow formatting labels.

Usage

```
ggeo_label_sci_10(x)

ggeo_latex_sci_10(x)

ggeo_label_pyramid(x)

ggeo_label_abs_percent(x)
```

Arguments

x	Number to format

Details

`ggeo_label_sci_10()` is used to format numbers with a clean scientific format using a multiplier and not the *ugly* notation using the letter *e*.

`ggeo_latex_sci_10()` is used to format numbers with a clean scientific format using a multiplier and not the *ugly* notation using the letter *e* but in LaTeX format.

`ggeo_label_pyramid()` is used for population pyramids with absolute numbers. It formats the absolute number using `ggeo_label_sci_10()`.

`ggeo_label_abs_percent()` is also used for population pyramids but with relative numbers (percents). It uses `scales::percent()` and absolute numbers.

Value

A formatted string for the scales

ggeo_remove_breaks *Remove specific breaks*

Description

This function can be used to modify the breaks of a ggplot2 scale. It is specifically designed to remove the some breaks in the scale.

Usage

```
ggeo_remove_breaks(original_func, remove_list = list())
```

Arguments

original_func	The function to create the breaks. Use the break functions from the scales
remove_list	The values to remove from the scale.

Value

A list

Examples

```
ggplot2::ggplot(ggplot2::aes(x = speed, y = dist), data = cars) +
  ggplot2::geom_point() +
  ggplot2::labs(title = "Fast cars!") +
  ggplot2::scale_y_continuous(
    breaks = ggeo::ggeo_remove_breaks(scales::breaks_pretty(6), list(0)),
  )
```

ggeo_remove_title *Remove plot title*

Description

Remove plot title

Usage

```
ggeo_remove_title(plot)
```

Arguments

plot	A ggplot2 object

Value

A ggplot2 object

Examples

```
ggplot(aes(x = speed, y = dist), data = cars) +
  geom_point() +
  labs(title = "Fast cars !") +
  ggeo_remove_title()
```

`ggeo_save` *Save function*

Description

Save function

Usage

```
ggeo_save(
  plot,
  filename,
  width = geotools::gtl_options("plot_standard_width"),
  height = geotools::gtl_options("plot_standard_height"),
  dpi = 72,
  units = "cm",
  ...
)
```

Arguments

`plot`	The ggplot2 object to save (should be piped in ;)
`filename`	Path for filename (with extension!)
`width`	The plot width. Defaults to keynote width (`geotools::gtl_options("plot_standard_width")`). For full slide width use `geotools::gtl_options("plot_full_width")`.
`height`	The plot height. Defaults to keynote height (`geotools::gtl_options("plot_standard_height")`). For full slide height use `geotools::gtl_options("plot_full_height")`.
`dpi`	The DPI. Default is 72.
`units`	Units. Default is cm.
`...`	Arguments passed on to `ggplot2::ggsave`
	`device` Device to use. Can either be a device function (e.g. `png`), or one of "eps", "ps", "tex" (pictex), "pdf", "jpeg", "tiff", "png", "bmp", "svg" or "wmf" (windows only). If `NULL` (default), the device is guessed based on the `filename` extension.

`scale` Multiplicative scaling factor.

`limitsize` When `TRUE` (the default), `ggsave()` will not save images larger than 50x50 inches, to prevent the common error of specifying dimensions in pixels.

`bg` Background colour. If `NULL`, uses the `plot.background` fill value from the plot theme.

`create.dir` Whether to create new directories if a non-existing directory is specified in the `filename` or `path` (`TRUE`) or return an error (`FALSE`, default). If `FALSE` and run in an interactive session, a prompt will appear asking to create a new directory when necessary.

Examples

```
cars |>
  ggplot2::ggplot(ggplot2::aes(x = speed, y = dist)) +
  ggplot2::geom_point() -> simple_plot

ggeo_save(simple_plot, "simple_plot.png")
```

`ggeopal_center` *Return a color palette with option to center diverging palettes*

Description

Return a color palette with option to center diverging palettes

Usage

```
ggeopal_center(n, center, params)
```

Arguments

`n`	Number of colors
`center`	Where is the center (counted from the left). Use -1 for non diverging palettes.
`params`	Parameters to pass on to paletteer

Value

A vector of colors

ggeopal_chooser *Return a paletteer palette depending on*

Description

Return a paletteer palette depending on

Usage

```
ggeopal_chooser(n, params)
```

Arguments

n	Number of colors
params	Parameters to pass on to paletteer Must be a list with: • type: the type of palette, on of cont, dis or dyn • palette: the paletteer palette • dir: the palette direction

Value

A vector of colors

ggeopal_hex_to_hcl *Generate colors from a HEX color using HCL palette*

Description

Generate colors from a HEX color using HCL palette

Usage

```
ggeopal_hex_to_hcl(hex, n = 4)
```

Arguments

hex	A string HEX color
n	Number of colors wanted

Value

A string vector of HEX colors

Examples

```
ggeopal_hex_to_hcl("#222b4c") |>
  prismatic::color()

ggeopal_hex_to_hcl("#222b4c", 6) |>
  prismatic::color()
```

---

ggeopal_to_gradient *Create gradient for palette colors*

Description

Create gradient for palette colors

Usage

```
ggeopal_to_gradient(pal)
```

Arguments

pal	A vector of colors

Value

Print color vectors

Examples

```
ggeopal_to_gradient("#dd4123")
ggeopal_to_gradient("#ee950b")
ggeopal_to_gradient("#0b6b8b")
```

---

ggeotheme *ggeo plot themes*

Description

A function that returns the ggplot theme elements for the chosen theme.

Usage

```
ggeotheme(
  theme = c("ghibli_mononoke", "islamic_samarquand", "pomological_green",
    "pomological_red", "nord_blue", "swiss_red", "purple", "doc", "oc_exams"),
  main = "main",
  plot = "plot",
  ...,
  mode = c("light", "dark"),
  base = ggplot2::theme_bw()
)
```

Arguments

`theme`	Name of the theme to use. One of "ghibli_mononoke", "islamic_samarquand", "pomological_green", "pomological_red", "nord_blue", "swiss_red", "purple", "doc" or "oc_exams"
`main`	One of `main`, `main_latex` or `main_exa`
`plot`	One of `plot`, `plot_latex` or `plot_exa`

`...`	Arguments passed on to `ggplot2::theme`
	`line` all line elements (`element_line()`)
	`rect` all rectangular elements (`element_rect()`)
	`text` all text elements (`element_text()`)
	`title` all title elements: plot, axes, legends (`element_text()`; inherits from `text`)
	`point` all point elements (`element_point()`)
	`polygon` all polygon elements (`element_polygon()`)
	`geom` defaults for geoms (`element_geom()`)
	`spacing` all spacings (`unit()`)
	`margins` all margins (`margin()`)
	`aspect.ratio` aspect ratio of the panel
	`axis.title,axis.title.x,axis.title.y,axis.title.x.top,axis.title.x.bottom,axis.title` labels of axes (`element_text()`). Specify all axes' labels (`axis.title`), labels by plane (using `axis.title.x` or `axis.title.y`), or individually for each axis (using `axis.title.x.bottom`, `axis.title.x.top`, `axis.title.y.left`, `axis.title.y.right`). `axis.title.*.*` inherits from `axis.title.*` which inherits from `axis.title`, which in turn inherits from `text`
	`axis.text,axis.text.x,axis.text.y,axis.text.x.top,axis.text.x.bottom,axis.text.y.lef` tick labels along axes (`element_text()`). Specify all axis tick labels (`axis.text`), tick labels by plane (using `axis.text.x` or `axis.text.y`), or individually for each axis (using `axis.text.x.bottom`, `axis.text.x.top`, `axis.text.y.left`, `axis.text.y.right`). `axis.text.*.*` inherits from `axis.text.*` which inherits from `axis.text`, which in turn inherits from `text`
	`axis.ticks,axis.ticks.x,axis.ticks.x.top,axis.ticks.x.bottom,axis.ticks.y,axis.ticks` tick marks along axes (`element_line()`). Specify all tick marks (`axis.ticks`), ticks by plane (using `axis.ticks.x` or `axis.ticks.y`), or individually for

each axis (using axis.ticks.x.bottom, axis.ticks.x.top, axis.ticks.y.left, axis.ticks.y.right). axis.ticks.*.* inherits from axis.ticks.* which inherits from axis.ticks, which in turn inherits from line

axis.minor.ticks.x.top,axis.minor.ticks.x.bottom,axis.minor.ticks.y.left,axis.minor. minor tick marks along axes (element_line()). axis.minor.ticks.*.* inherit from the corresponding major ticks axis.ticks.*.*.

axis.ticks.length,axis.ticks.length.x,axis.ticks.length.x.top,axis.ticks.length.x.bo length of tick marks (unit). axis.ticks.length inherits from spacing.

axis.minor.ticks.length,axis.minor.ticks.length.x,axis.minor.ticks.length.x.top,axis length of minor tick marks (unit), or relative to axis.ticks.length when provided with rel().

axis.line,axis.line.x,axis.line.x.top,axis.line.x.bottom,axis.line.y,axis.line.y.lef lines along axes (element_line()). Specify lines along all axes (axis.line), lines for each plane (using axis.line.x or axis.line.y), or individually for each axis (using axis.line.x.bottom, axis.line.x.top, axis.line.y.left, axis.line.y.right). axis.line.*.* inherits from axis.line.* which inherits from axis.line, which in turn inherits from line

legend.background background of legend (element_rect(); inherits from rect)

legend.margin the margin around each legend (margin()); inherits from margins.

legend.spacing,legend.spacing.x,legend.spacing.y the spacing between legends (unit). legend.spacing.x & legend.spacing.y inherit from legend.spacing or can be specified separately. legend.spacing inherits from spacing.

legend.key background underneath legend keys (element_rect(); inherits from rect)

legend.key.size,legend.key.height,legend.key.width size of legend keys (unit); key background height & width inherit from legend.key.size or can be specified separately. In turn legend.key.size inherits from spacing.

legend.key.spacing,legend.key.spacing.x,legend.key.spacing.y spacing between legend keys given as a unit. Spacing in the horizontal (x) and vertical (y) direction inherit from legend.key.spacing or can be specified separately. legend.key.spacing inherits from spacing.

legend.key.justification Justification for positioning legend keys when more space is available than needed for display. The default, NULL, stretches keys into the available space. Can be a location like "center" or "top", or a two-element numeric vector.

legend.frame frame drawn around the bar (element_rect()).

legend.ticks tick marks shown along bars or axes (element_line())

legend.ticks.length length of tick marks in legend (unit()); inherits from legend.key.size.

legend.axis.line lines along axes in legends (element_line())

legend.text legend item labels (element_text(); inherits from text)

legend.text.position placement of legend text relative to legend keys or bars ("top", "right", "bottom" or "left"). The legend text placement might be incompatible with the legend's direction for some guides.

legend.title title of legend (element_text(); inherits from title)

legend.title.position placement of legend title relative to the main legend ("top", "right", "bottom" or "left").

legend.position the default position of legends ("none", "left", "right", "bottom", "top", "inside")

legend.position.inside A numeric vector of length two setting the placement of legends that have the "inside" position.

legend.direction layout of items in legends ("horizontal" or "vertical")

legend.byrow whether the legend-matrix is filled by columns (FALSE, the default) or by rows (TRUE).

legend.justification anchor point for positioning legend inside plot ("center" or two-element numeric vector) or the justification according to the plot area when positioned outside the plot

legend.justification.top,legend.justification.bottom,legend.justification.left,leger Same as legend.justification but specified per legend.position option.

legend.location Relative placement of legends outside the plot as a string. Can be "panel" (default) to align legends to the panels or "plot" to align legends to the plot as a whole.

legend.box arrangement of multiple legends ("horizontal" or "vertical")

legend.box.just justification of each legend within the overall bounding box, when there are multiple legends ("top", "bottom", "left", "right", "center" or "centre")

legend.box.margin margins around the full legend area, as specified using margin(); inherits from margins.

legend.box.background background of legend area (element_rect(); inherits from rect)

legend.box.spacing The spacing between the plotting area and the legend box (unit); inherits from spacing.

panel.background background of plotting area, drawn underneath plot (element_rect(); inherits from rect)

panel.border border around plotting area, drawn on top of plot so that it covers tick marks and grid lines. This should be used with fill = NA (element_rect(); inherits from rect)

panel.spacing,panel.spacing.x,panel.spacing.y spacing between facet panels (unit). panel.spacing.x & panel.spacing.y inherit from panel.spacing or can be specified separately. panel.spacing inherits from spacing.

panel.grid,panel.grid.major,panel.grid.minor,panel.grid.major.x,panel.grid.major.y,p grid lines (element_line()). Specify major grid lines, or minor grid lines separately (using panel.grid.major or panel.grid.minor) or individually for each axis (using panel.grid.major.x, panel.grid.minor.x, panel.grid.major.y, panel.grid.minor.y). Y axis grid lines are horizontal and x axis grid lines are vertical. panel.grid.*.* inherits from panel.grid.* which inherits from panel.grid, which in turn inherits from line

`panel.ontop` option to place the panel (background, gridlines) over the data layers (`logical`). Usually used with a transparent or blank `panel.background`.

`panel.widths,panel.heights` Sizes for panels (`units`). Can be a single unit to set the total size for the panel area, or a unit vector to set the size of individual panels. Using this setting overrides the aspect ratio set by the theme, coord or facets. An exception is made when the plot has a single panel and exactly one of the width *or* height is set, in which case an attempt is made to preserve the aspect ratio.

`plot.background` background of the entire plot (`element_rect()`; inherits from `rect`)

`plot.title` plot title (text appearance) (`element_text()`; inherits from `title`) left-aligned by default

`plot.title.position,plot.caption.position` Alignment of the plot title/subtitle and caption. The setting for `plot.title.position` applies to both the title and the subtitle. A value of "panel" (the default) means that titles and/or caption are aligned to the plot panels. A value of "plot" means that titles and/or caption are aligned to the entire plot (minus any space for margins and plot tag).

`plot.subtitle` plot subtitle (text appearance) (`element_text()`; inherits from `title`) left-aligned by default

`plot.caption` caption below the plot (text appearance) (`element_text()`; inherits from `title`) right-aligned by default

`plot.tag` upper-left label to identify a plot (text appearance) (`element_text()`; inherits from `title`) left-aligned by default

`plot.tag.position` The position of the tag as a string ("topleft", "top", "topright", "left", "right", "bottomleft", "bottom", "bottomright") or a coordinate. If a coordinate, can be a numeric vector of length 2 to set the x,y-coordinate relative to the whole plot. The coordinate option is unavailable for `plot.tag.location = "margin"`.

`plot.tag.location` The placement of the tag as a string, one of `"panel"`, `"plot"` or `"margin"`. Respectively, these will place the tag inside the panel space, anywhere in the plot as a whole, or in the margin around the panel space.

`plot.margin` margin around entire plot (`unit` with the sizes of the top, right, bottom, and left margins); inherits from `margin`.

`strip.background,strip.background.x,strip.background.y` background of facet labels (`element_rect()`; inherits from `rect`). Horizontal facet background (`strip.background.x`) & vertical facet background (`strip.background.y`) inherit from `strip.background` or can be specified separately

`strip.clip` should strip background edges and strip labels be clipped to the extend of the strip background? Options are `"on"` to clip, `"off"` to disable clipping or `"inherit"` (default) to take the clipping setting from the parent viewport.

`strip.placement` placement of strip with respect to axes, either "inside" or "outside". Only important when axes and strips are on the same side of the plot.

`strip.text,strip.text.x,strip.text.y,strip.text.x.top,strip.text.x.bottom,strip.text`
: facet labels (`element_text()`; inherits from `text`). Horizontal facet labels (`strip.text.x`) & vertical facet labels (`strip.text.y`) inherit from `strip.text` or can be specified separately. Facet strips have dedicated position-dependent theme elements (`strip.text.x.top`, `strip.text.x.bottom`, `strip.text.y.left`, `strip.text.y.right`) that inherit from `strip.text.x` and `strip.text.y`, respectively. As a consequence, some theme stylings need to be applied to the position-dependent elements rather than to the parent elements

`strip.switch.pad.grid,strip.switch.pad.wrap`
: space between strips and axes when strips are switched (`unit`); inherits from `spacing`.

`complete`
: set this to `TRUE` if this is a complete theme, such as the one returned by `theme_grey()`. Complete themes behave differently when added to a ggplot object. Also, when setting `complete = TRUE` all elements will be set to inherit from blank elements.

`validate`
: `TRUE` to run `check_element()`, `FALSE` to bypass checks.

`mode`
: One of `light` or `dark`

`base`
: A ggplot2 theme

Value

An object of class `ggplot2::theme()`.

Examples

```
cars |>
  ggplot2::ggplot(ggplot2::aes(x = speed, y = dist)) +
  ggplot2::geom_point() +
  ggeotheme("ghibli_mononoke")
```

`hc_purple_theme` *Highcharts themes*

Description

These functions can be used to modify the theme of a highcharts plot.

Usage

```
hc_purple_theme(hc)

hc_samarqand_theme(hc)

hc_pomogreen_theme(hc)

hc_jerusalem_theme(hc)
```

```
hc_guonian_theme(hc)

hc_web_theme(hc)

hc_dark_web_theme(hc)
```

Arguments

hc	A highcharts object

Value

A highcharts object

Examples

```
highcharter::hchart(
  cars, "point",
  highcharter::hcaes(x = speed, y = dist)
) |>
  hc_purple_theme()

highcharter::hchart(
  cars, "point",
  highcharter::hcaes(x = speed, y = dist)
) |>
  hc_samarqand_theme()

highcharter::hchart(
  cars, "point",
  highcharter::hcaes(x = speed, y = dist)
) |>
  hc_web_theme()
```

Index

aes(), *5*
annotation_borders(), *6*

element_geom(), *13*
element_line(), *13–15*
element_point(), *13*
element_polygon(), *13*
element_rect(), *13–16*
element_text(), *13–17*

fortify(), *5*

geotools::gtl_crs_proj(), *3*, *6*
ggeo_capitalize_title, 2
ggeo_coord, 3
ggeo_label_abs_percent *(*ggeo_label_sci_10*)*, 7
ggeo_label_abs_percent(), *7*
ggeo_label_pyramid *(*ggeo_label_sci_10*)*, *7*
ggeo_label_pyramid(), *7*
ggeo_label_sci_10, 7
ggeo_label_sci_10(), *7*
ggeo_latex_sci_10 *(*ggeo_label_sci_10*)*, 7
ggeo_latex_sci_10(), *7*
ggeo_remove_breaks, 8
ggeo_remove_title, 8
ggeo_save, 9
ggeopal_center, 10
ggeopal_chooser, 11
ggeopal_hex_to_hcl, 11
ggeopal_to_gradient, 12
ggeotheme, 12
ggplot(), *5*
ggplot2::coord_sf, *3*
ggplot2::ggsave, *9*
ggplot2::theme, *13*
ggplot2::theme(), *17*

hc_dark_web_theme *(*hc_purple_theme*)*, 17
hc_guonian_theme *(*hc_purple_theme*)*, 17
hc_jerusalem_theme *(*hc_purple_theme*)*, 17
hc_pomogreen_theme *(*hc_purple_theme*)*, 17
hc_purple_theme, 17
hc_samarqand_theme *(*hc_purple_theme*)*, 17
hc_web_theme *(*hc_purple_theme*)*, 17

layer geom, *6*
layer position, *5*
layer stat, *5*

margin(), *13–15*

png, *9*

scales::percent(), *7*

theme_grey(), *17*

unit(), *13*, *14*