Package 'deeptime'

February 16, 2023

Title Plotting Tools for Anyone Working in Deep Time

Version 1.0.1

Maintainer William Gearty <willgearty@gmail.com>

Description Extends the functionality of other plotting packages like 'ggplot2' and 'lattice' to help facilitate the plotting of data over long time intervals, including, but not limited to, geological, evolutionary, and ecological data. The primary goal of 'deeptime' is to enable users to add highly customizable timescales to their visualizations. Other functions are also included to assist with other areas of deep time visualization.

URL https://github.com/willgearty/deeptime, https://williamgearty.com/deeptime/

BugReports https://github.com/willgearty/deeptime/issues

Depends R (>= 3.4)

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.2.3

LazyData true

biocViews

Imports ggplot2 (>= 3.3.0), ggnewscale, utils, ggforce, grid, gridExtra, gtable, methods, stats, lattice, rlang, scales, ggfittext, curl, cli, phytools, lifecycle

Suggests dplyr, magrittr, divDyn, gsloid, ape, paleotree, dispRity, ggtree (>= 3.6.1), testthat (>= 3.0.0), vdiffr (>= 1.0.0), knitr, rmarkdown, withr

VignetteBuilder knitr

Config/testthat/edition 3

Config/Needs/revdeps revdepcheck

NeedsCompilation no

Author William Gearty [aut, cre] (<https://orcid.org/0000-0003-0076-3262>)

Repository CRAN

Date/Publication 2023-02-16 16:40:02 UTC

R topics documented:

`coord_geo` *Transformed coordinate system with geological timescale*

Description

`coord_geo` behaves similarly to `ggplot2::coord_trans()` in that it occurs after statistical transformation and will affect the visual appearance of geoms. The main difference is that it also adds a geological timescale to the specified side(s) of the plot.

Usage

```
coord_geo(
  pos = "bottom",
  dat = "periods",
  xlim = NULL,
  ylim = NULL,
  xtrans = identity_trans(),
  ytrans = identity_trans(),
  clip = "on",
  expand = FALSE,
  fill = NULL,
  color = "black",
  alpha = 1,
  height = unit(2, "line"),
  lab = TRUE,
  lab_color = NULL,
  rot = 0,
```

```
  abbrv = TRUE,
  skip = c("Quaternary", "Holocene", "Late Pleistocene"),
  size = 5,
  lwd = 0.25,
  neg = FALSE,
  bord = c("left", "right", "top", "bottom"),
  center_end_labels = FALSE,
  dat_is_discrete = FALSE,
  fittext_args = list()
)
```

Arguments

pos	Which side to add the scale to (left, right, top, or bottom). First letter may also be used.
dat	Either A) a string indicating a built-in dataframe with interval data from the ICS ("periods", "epochs", "stages", "eons", or "eras"), B) a string indicating a timescale from macrostrat (see list here: https://macrostrat.org/api/defs/timescales?all), or C) a custom data.frame of time interval boundaries (see Details).
xlim, ylim	Limits for the x and y axes.
xtrans, ytrans	Transformers for the x and y axes. For more information see ggplot2::coord_trans().
clip	Should drawing be clipped to the extent of the plot panel? For more information see ggplot2::coord_trans().
expand	If FALSE, the default, limits are taken exactly from the data or xlim/ylim. If TRUE, adds a small expansion factor to the limits to ensure that data and axes don't overlap.
fill	The fill color of the boxes. The default is to use the color column included in dat. If a custom dataset is provided with dat without a color column and without fill, a greyscale will be used. Custom fill colors can be provided with this option (overriding the color column) and will be recycled if/as necessary.
color	The outline color of the interval boxes.
alpha	The transparency of the fill colors.
height	The height (or width if pos is left or right) of the scale.
lab	Whether to include labels.
lab_color	The color of the labels. The default is to use the lab_color column included in dat. If a custom dataset is provided with dat without a lab_color column and without fill, all labels will be black. Custom label colors can be provided with this option (overriding the lab_color column) and will be recycled if/as necessary.
rot	The amount of counter-clockwise rotation to add to the labels (in degrees).
abbrv	If including labels, whether to use abbreviations instead of full interval names.
skip	A vector of interval names indicating which intervals should not be labeled. If abbrv is TRUE, this can also include interval abbreviations.

size	Label size. Either a number as you would specify in ggplot2::geom_text() or "auto" to use ggfittext::geom_fit_text().
lwd	Line width.
neg	Set this to true if your x-axis is using negative values.
bord	A vector specifying on which sides of the scale to add borders (same options as pos).
center_end_labels	Should labels be centered within the visible range of intervals at the ends of the axis?
dat_is_discrete	Are the ages in dat already converted for a discrete scale?
fittext_args	A list of named arguments to provide to ggfittext::geom_fit_text(). Only used if size is set to "auto".

Details

Transforming the side with the scale is not currently implemented. If a custom data.frame is provided (with dat), it should consist of at least 3 columns of data. See data(periods) for an example.

- The name column lists the names of each time interval. These will be used as labels if no abbreviations are provided.
- The max_age column lists the oldest boundary of each time interval.
- The min_age column lists the youngest boundary of each time interval.
- The abbr column is optional and lists abbreviations that may be used as labels.
- The color column is also optional and lists a color for the background for each time interval.
- The lab_color column is also optional and lists a color for the label for each time interval.

If the axis of the time scale is discrete, max_age and min_age will automatically be converted to the discrete scale. In this case, the categories of the discrete axis should match the values in the name column. If the ages within dat are already discretized, you can set dat_is_discrete to TRUE to prevent this automatic conversion. This can be useful for adding a time scale where categories and time intervals are not 1:1.

pos may also be a list of sides (including duplicates) if multiple time scales should be added to the plot. In this case, dat, fill, color, alpha, height, lab, lab_color, rot, abbrv, skip, size, lwd, neg, bord, center_end_labels, and dat_is_discrete can also be lists. If these lists are not as long as pos, the elements will be recycled. If individual values (or vectors) are used for these parameters, they will be applied to all time scales (and recycled as necessary).

Examples

```
library(ggplot2)
# single scale on bottom
ggplot() +
  geom_point(aes(y = runif(1000, 0, 8), x = runif(1000, 0, 1000))) +
  scale_x_reverse() +
  coord_geo(xlim = c(1000, 0), ylim = c(0, 8)) +
  theme_classic()
```

```
# stack multiple scales
ggplot() +
  geom_point(aes(y = runif(1000, 0, 8), x = runif(1000, 0, 100))) +
  scale_x_reverse() +
  coord_geo(
    xlim = c(100, 0), ylim = c(0, 8), pos = as.list(rep("bottom", 3)),
    dat = list("stages", "epochs", "periods"),
    height = list(unit(4, "lines"), unit(4, "lines"), unit(2, "line")),
    rot = list(90, 90, 0), size = list(2.5, 2.5, 5), abbrv = FALSE
  ) +
  theme_classic()
```

`coord_geo_polar` *Polar coordinate system with geological timescale*

Description

`coord_geo_polar` behaves similarly to `ggplot2::coord_polar()` in that it occurs after statistical transformation and will affect the visual appearance of geoms. The main difference is that it also adds a geological timescale to the background of the plot.

Usage

```
coord_geo_polar(
  dat = "periods",
  theta = "y",
  start = -pi/2,
  direction = -1,
  clip = "off",
  fill = NULL,
  alpha = 1,
  lwd = 0.25,
  color = "grey80",
  lty = "solid",
  neg = TRUE,
  prop = 1
)
```

Arguments

`dat`	Either A) a string indicating a built-in dataframe with interval data from the ICS ("periods", "epochs", "stages", "eons", or "eras"), B) a string indicating a timescale from macrostrat (see list here: https://macrostrat.org/api/defs/timescales?all), or C) a custom data.frame of time interval boundaries (see Details).
`theta`	variable to map angle to (`x` or `y`)

start	Offset of starting point from 12 o'clock in radians. Offset is applied clockwise or anticlockwise depending on value of direction.
direction	1, clockwise; -1, anticlockwise
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. For details, please see coord_cartesian().
fill	The fill color of the background. The default is to use the color column included in dat. If a custom dataset is provided with dat without a color column and without fill, a greyscale will be used. Custom fill colors can be provided with this option (overriding the color column) and will be recycled if/as necessary.
alpha	The transparency of the fill colors.
lwd	Line width for lines between intervals. Set to NULL to remove lines.
color	The color of the lines between intervals.
lty	Line type for lines between intervals.
neg	Set this to true if your theta-axis is using negative values. This is often true if you are using ggtree.
prop	This is the rotational proportion of the background that the scale takes up.

Details

If a custom data.frame is provided (with dat), it should consist of at least 2 columns of data. See data(periods) for an example.

- The max_age column lists the oldest boundary of each time interval.
- The min_age column lists the youngest boundary of each time interval.
- The color column is optional and lists a color for the background for each time interval.

dat may also be a list of values and/or dataframes if multiple time scales should be added to the background. Scales will be added sequentially starting at start and going in the specified direction. By default the scales will all be equal in circular/rotational proportion, but this can be overridden with prop. If dat is a list, fill, alpha, lwd, lty, color, neg, and prop can also be lists. If these lists are not as long as dat, the elements will be recycled. If individual values (or vectors) are used for these parameters, they will be applied to all time scales (and recycled as necessary).

If the sum of the prop values is greater than 1, the proportions will be scaled such that they sum to 1. However, the prop values may sum to less than 1 if the user would like blank space in the background.

The axis.line.r, axis.text.r, axis.ticks.r, and axis.ticks.length.r ggplot2 theme elements can be modified just like their x and y counterparts to change the appearance of the radius axis. The default settings work well for a horizontal axis pointing towards the right, but these theme settings will need to be modified for other orientations. The default value for axis.line.r is element_line(). The default value for axis.text.r is element_text(size = 3.5, vjust = -2, hjust = NA). The default value for axis.ticks.r is element_line(). The default value for axis.ticks.length.r is unit(1.5, "points"). However, note that the units for this element are meaningless and only the numeric value will be used (but a unit must still be used).

Examples

```
library(ggplot2)

library(ggtree)
set.seed(1)
tree <- rtree(100)
# single scale
revts(ggtree(tree)) +
  coord_geo_polar(dat = "stages")

# multiple scales
revts(ggtree(tree)) +
  coord_geo_polar(
    dat = list("stages", "periods"), alpha = .5,
    prop = list(0.75, .25), start = pi / 4, lty = "dashed"
  ) +
  scale_y_continuous(expand = expansion(mult = c(0.02, 0.02))) +
  theme(axis.text.r = element_text(size = 3.5, hjust = .75, vjust = .75))


library(ggplot2)
library(paleotree)
data(RaiaCopesRule)
ggtree(ceratopsianTreeRaia,
       position = position_nudge(x = -ceratopsianTreeRaia$root.time)) +
  coord_geo_polar(dat = "stages")
```

`coord_trans_flip` *Transformed and flipped Cartesian coordinate system*

Description

`coord_trans_flip` behaves similarly to `ggplot2::coord_trans()` in that it occurs after statistical transformation and will affect the visual appearance of geoms. The main difference is that it also flips the x and y coordinates like `ggplot2::coord_flip()`.

Usage

```
coord_trans_flip(
  x = "identity",
  y = "identity",
  xlim = NULL,
  ylim = NULL,
  clip = "on",
  expand = TRUE
)
```

Arguments

x, y	Transformers for x and y axes or their names.
xlim, ylim	Limits for the x and y axes.
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.
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.

Examples

```
library(ggplot2)
ggplot(mtcars, aes(disp, wt)) +
  geom_point() +
  coord_trans_flip(x = "log10", y = "log10")
```

---

coord_trans_xy *Transformed XY Cartesian coordinate system*

Description

coord_trans_xy behaves similarly to ggplot2::coord_trans() in that it occurs after statistical transformation and will affect the visual appearance of geoms. The main difference is that it takes a single transformer that is applied to the x and y axes simultaneously. Any transformers produced by ggforce::linear_trans() that have x and y arguments should work, but any other transformers produced using scales::trans_new() that take x and y arguments should also work. Axis limits will be adjusted to account for transformation unless limits are specified with xlim or ylim. This only works with geoms where all points are defined with x and y coordinates (e.g., ggplot2::geom_point(), ggplot2::geom_polygon()). This does not currently work with geoms where point coordinates are extrapolated (e.g., ggplot2::geom_rect()).

Usage

```
coord_trans_xy(
  trans = NULL,
  xlim = NULL,
  ylim = NULL,
  expand = FALSE,
  default = FALSE,
  clip = "on"
)
```

Arguments

trans	Transformer for x and y axes.
xlim, ylim	Limits for the x and y axes.
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.
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.

Examples

```
# make transformer
library(ggforce)
trans <- linear_trans(shear(2, 0), rotate(-pi / 3))

# set up data to be plotted
square <- data.frame(x = c(0, 0, 4, 4), y = c(0, 1, 1, 0))
points <- data.frame(x = runif(100, 0, 4), y = runif(100, 0, 1))

# plot data normally
library(ggplot2)
ggplot(data = points, aes(x = x, y = y)) +
  geom_polygon(data = square, fill = NA, color = "black") +
  geom_point(color = "black") +
  coord_cartesian(expand = FALSE) +
  theme_classic()

# plot data with transformation
ggplot(data = points, aes(x = x, y = y)) +
  geom_polygon(data = square, fill = NA, color = "black") +
  geom_point(color = "black") +
  coord_trans_xy(trans = trans, expand = FALSE) +
  theme_classic()
```

---

disparity_through_time

Disparity through time plot using lattice

Description

Plots points on 2-D surfaces within a a 3-D framework. See `lattice::wireframe()` and `lattice::panel.cloud()` for customization options.

Usage

```
disparity_through_time(
  x,
  data,
  groups,
  pch = 16,
  col.point = c("blue"),
  scales = list(arrows = FALSE, distance = 1, col = "black", z = list(rot = 90)),
  colorkey = FALSE,
  screen = list(z = 90, x = 70, y = 180),
  aspect = c(1.5, 4),
  drape = TRUE,
  col.regions = c("white"),
  alpha.regions = c(1),
  perspective = FALSE,
  R.mat = matrix(c(1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1), 4, 4),
  par.settings = list(axis.line = list(col = "transparent"), layout.heights =
   list(top.padding = 0, main.key.padding = 0, key.axis.padding = 0, axis.xlab.padding =
   0, xlab.key.padding = 0, key.sub.padding = 0, bottom.padding = 0), layout.widths =
   list(left.padding = 0, key.ylab.padding = 0, ylab.axis.padding = 0, axis.key.padding
    = 0, right.padding = 0)),
  lattice.options = list(axis.padding = list(factor = 0)),
  ...
)
```

Arguments

`x`	a formula (most likely of the form `z ~ x * y`)
`data`	a data frame in which variables in the formula are to be evaluated
`groups`	a variable in `data` to be used as a grouping variable (this is probably the z variable)
`pch`	the point type
`col.point`	color(s) for points on surfaces
`scales`	a list specifying how the axes are drawn (see `lattice::xyplot()` for details)
`colorkey`	logical, should a legend be drawn (or a list describing the legend; see `lattice::levelplot()` for details)
`screen`	a list of the rotations that should be applied to each axis
`aspect`	a numeric vector of length 2, giving the relative aspects of the y-size/x-size and z-size/x-size of the enclosing cube
`drape`	logical, whether the surfaces should be colored based on `col.regions` and `alpha.regions`

col.regions	color(s) for surfaces
alpha.regions	alpha value(s) for surfaces
perspective	logical, whether to plot a perspective view
R.mat	a transformational matrix that is applied to the orientation of the axes
par.settings	plotting settings (see lattice::trellis.par.set())
lattice.options	lattice settings (see lattice::lattice.options())
...	Other arguments passed to lattice::wireframe()

Value

An object of class "trellis", as output by lattice::wireframe().

Examples

```
g <- data.frame(
  x = runif(100, 0, 60), y = runif(100, 0, 10),
  z = factor(rep(periods$name[1:5], each = 20),
    levels = periods$name[1:5]
  )
)
disparity_through_time(z ~ x * y,
  data = g, groups = z, aspect = c(1.5, 2),
  xlim = c(0, 60), ylim = c(0, 10), col.regions = "lightgreen",
  col.point = c("red", "blue")
)
```

---

eons *Eon data from the International Commission on Stratigraphy (v2022/10)*

Description

A dataset containing the boundary ages, abbreviations, and colors for the eons of the Geologic Time Scale. Based on The ICS International Chronostratigraphic Chart (v2022/10), by Cohen, Finney, Gibbard, and Fan.

Usage

```
eons
```

Format

A data frame with 3 rows and 5 variables:

name eon name

max_age maximum age, in millions of years

min_age minimum age, in millions of years

abbr eon name abbreviations

color the colors for each eon, according to the Commission for the Geological Map of the World

Source

https://stratigraphy.org via https://macrostrat.org/api/v2/defs/intervals?timescale=international%20eons

See Also

Other timescales: epochs, eras, periods, stages

---

epochs *Epoch data from the International Commission on Stratigraphy (v2022/10)*

Description

A dataset containing the boundary ages, abbreviations, and colors for the epochs of the Geologic Time Scale. Based on The ICS International Chronostratigraphic Chart (v2022/10), by Cohen, Finney, Gibbard, and Fan.

Usage

```
epochs
```

Format

A data frame with 34 rows and 5 variables:

name epoch name

max_age maximum age, in millions of years

min_age minimum age, in millions of years

abbr epoch name abbreviations

color the colors for each epoch, according to the Commission for the Geological Map of the World

Source

https://stratigraphy.org via https://macrostrat.org/api/v2/defs/intervals?timescale=international%20epochs

See Also

Other timescales: eons, eras, periods, stages

`eras` *Era data from the International Commission on Stratigraphy (v2022/10)*

Description

A dataset containing the boundary ages, abbreviations, and colors for the eras of the Geologic Time Scale. Based on The ICS International Chronostratigraphic Chart (v2022/10), by Cohen, Finney, Gibbard, and Fan.

Usage

```
eras
```

Format

A data frame with 10 rows and 5 variables:

name era name

max_age maximum age, in millions of years

min_age minimum age, in millions of years

abbr era name abbreviations

color the colors for each era, according to the Commission for the Geological Map of the World

Source

https://stratigraphy.org via https://macrostrat.org/api/v2/defs/intervals?timescale=international%20eras

See Also

Other timescales: eons, epochs, periods, stages

geom_phylomorpho *Plot a 2-D phylomorphospace in ggplot2*

Description

This behaves similar to `phytools::phylomorphospace()`, but is for plotting a 2-D phylomorphospace with `ggplot2::ggplot()`. This function works like any other `ggplot2` geom; it can be combined with other geoms (see the example below), and the output can be modified using scales, themes, etc.

Usage

```
geom_phylomorpho(
  tree,
  mapping = NULL,
  data = NULL,
  position = "identity",
  ...,
  seg_args = list(),
  point_args = list(),
  arrow = NULL,
  arrow.fill = NULL,
  lineend = "butt",
  linejoin = "round",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

`tree`	An object of class "phylo".
`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)`).

position	Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
...	Other arguments passed on to both ggplot2::geom_segment() and ggplot2::geom_point().
seg_args	A list of arguments passed only to ggplot2::geom_segment().
point_args	A list of arguments passed only to ggplot2::geom_point().
arrow	specification for arrow heads, as created by grid::arrow().
arrow.fill	fill colour to use for the arrow head (if closed). NULL means use colour aesthetic.
lineend	Line end style (round, butt, square).
linejoin	Line join style (round, mitre, bevel).
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. It can also be a named logical vector to finely select the aesthetics to display.
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. borders().

Details

The ancestral states are estimated using phytools::fastAnc(). The nodes are connected using ggplot2::geom_segment(), while the tips are indicated using ggplot2::geom_point().

The default expectation is that the order of the data is the same order as the tip labels of the tree (tree$tip.label). However, if this is not the case, you can map the optional label aesthetic to a column in the data that contains the tip names (see example below).

Examples

```
library(ggplot2)

library(ape)
tr <- rtree(10)
dat <- data.frame(
  x = runif(10), y = runif(10), label = tr$tip.label,
  row.names = tr$tip.label
)
ggplot(dat, aes(x = x, y = y, label = label)) +
  geom_phylomorpho(tr) +
  geom_label(size = 5)
```

get_scale_data *Get geological timescale data*

Description

This function takes a name of a geological timescale and returns data for the timescale. Valid names include those of built-in `data.frames` (`periods()`, `epochs()`, `stages()`, `eons()`, or `eras()`), partial matches of those names (e.g., "per" or "stage"), and exact matches to those hosted by Macrostrat (see full list here: `https://macrostrat.org/api/defs/timescales?all`).

Usage

```
get_scale_data(name)
```

Arguments

`name`	The name of the desired timescale.

Value

A `data.frame` with the following columns:

`name`	the names of the time intervals.
`max_age`	the oldest boundaries of the time intervals, in millions of years.
`min_age`	the youngest boundaries of the time intervals, in millions of years.
`abbr`	either traditional abbreviations of the names of the time intervals (if they exist) or custom abbreviations created with R.
`color`	hex color codes associated with the time intervals (if applicable).

ggarrange2 *Combine and arrange multiple ggplot-like objects*

Description

Arrange multiple ggplot, grobified ggplot, or geo_scale objects on a page, aligning the plot panels, axes, and axis titles.

Usage

```
ggarrange2(
  ...,
  plots = list(...),
  layout = NULL,
  nrow = NULL,
  ncol = NULL,
  widths = NULL,
  heights = NULL,
  byrow = TRUE,
  top = NULL,
  bottom = NULL,
  left = NULL,
  right = NULL,
  padding = unit(0.5, "line"),
  margin = unit(0.5, "line"),
  clip = "on",
  draw = TRUE,
  newpage = TRUE,
  debug = FALSE,
  labels = NULL,
  label.args = list(gp = gpar(font = 4, cex = 1.2))
)
```

Arguments

...	ggplot, grobified ggplot (gtable), or geo_scale objects
plots	list of ggplot, gtable, or geo_scale objects
layout	a matrix of integers specifying where each plot should go, like mat in graphics::layout(); NA or a value less than 0 or greater than the number of plots indicates a blank plot; overrides nrow/ncol/byrow
nrow	number of rows
ncol	number of columns
widths	list of requested widths
heights	list of requested heights
byrow	logical, fill by rows
top	optional string, or grob
bottom	optional string, or grob
left	optional string, or grob
right	optional string, or grob
padding	unit of length one, margin around annotations
margin	vector of units of length 4: top, right, bottom, left (as in gtable::gtable_add_padding())
clip	argument of gtable
draw	logical: draw or return a grob

newpage	logical: draw on a new page
debug	logical, show layout with thin lines
labels	character labels used for annotation of subfigures (should be in the same order as plots)
label.args	label list of parameters for the formatting of labels

Value

gtable of aligned plots

Examples

```
library(ggplot2)
p1 <- ggplot(mtcars, aes(mpg, wt, colour = factor(cyl))) +
  geom_point()
p2 <- ggplot(mtcars, aes(mpg, wt, colour = factor(cyl))) +
  geom_point() +
  facet_wrap(~cyl, ncol = 2, scales = "free") +
  guides(colour = "none") +
  theme()
ggarrange2(p1, p2, widths = c(2, 1), labels = c("a", "b"))

p3 <- ggplot() +
  geom_point(aes(y = runif(1000, 0, 8), x = runif(1000, 0, 1000))) +
  scale_x_reverse() +
  coord_geo(xlim = c(1000, 0), ylim = c(0, 8)) +
  theme_classic()
ggarrange2(ggarrange2(p1, p2, widths = c(2, 1), draw = FALSE), p3, nrow = 2)
```

---

gtable_frame2 *Decompose a ggplot gtable*

Description

Reformat the gtable associated with a ggplot object into a 7x7 gtable where the central cell corresponds to the plot panel(s), the rectangle of cells around that corresponds to the axes, and the rectangle of cells around that corresponds to the axis titles.

Usage

```
gtable_frame2(
  g,
  width = unit(1, "null"),
  height = unit(1, "null"),
  debug = FALSE
)
```

Arguments

g	gtable
width	requested width
height	requested height
debug	logical draw gtable cells

Value

7x7 gtable wrapping the plot

Examples

```
library(grid)
library(gridExtra)
library(ggplot2)
p1 <- ggplot(mtcars, aes(mpg, wt, colour = factor(cyl))) +
  geom_point()

p2 <- ggplot(mtcars, aes(mpg, wt, colour = factor(cyl))) +
  geom_point() +
  facet_wrap(~cyl, ncol = 2, scales = "free") +
  guides(colour = "none") +
  theme()

p3 <- ggplot(mtcars, aes(mpg, wt, colour = factor(cyl))) +
  geom_point() +
  facet_grid(. ~ cyl, scales = "free")

g1 <- ggplotGrob(p1)
g2 <- ggplotGrob(p2)
g3 <- ggplotGrob(p3)
fg1 <- gtable_frame2(g1)
fg2 <- gtable_frame2(g2)
fg12 <- gtable_frame2(gtable_rbind(fg1, fg2), width = unit(2, "null"),
                      height = unit(1, "null"))
fg3 <- gtable_frame2(g3, width = unit(1, "null"), height = unit(1, "null"))
grid.newpage()
combined <- gtable_cbind(fg12, fg3)
grid.draw(combined)
```

---

panel.disparity *Combined wireframe and cloud panel*

Description

Plots the provided data on 2-D surfaces within a 3-D framework. See disparity_through_time().

Usage

```
panel.disparity(x, y, z, groups, subscripts, ...)
```

Arguments

`x, y, z, groups, subscripts, ...`
: Same as for `lattice::panel.cloud()`

Value

No return value, plots the results of both `lattice::panel.cloud()` and `lattice::panel.wireframe()`.

---

periods *Period data from the International Commission on Stratigraphy (v2022/10)*

Description

A dataset containing the boundary ages, abbreviations, and colors for the periods of the Geologic Time Scale. Based on The ICS International Chronostratigraphic Chart (v2022/10), by Cohen, Finney, Gibbard, and Fan.

Usage

```
periods
```

Format

A data frame with 22 rows and 5 variables:

name period name

max_age maximum age, in millions of years

min_age minimum age, in millions of years

abbr period name abbreviations

color the colors for each period, according to the Commission for the Geological Map of the World

Source

https://stratigraphy.org via https://macrostrat.org/api/v2/defs/intervals?timescale=international%20periods

See Also

Other timescales: `eons`, `epochs`, `eras`, `stages`

scale_color_geo *Geological Time Scale color scales*

Description

Color scales using the colors in the Geological Time Scale graphics.

Usage

```
scale_color_geo(dat, ...)

scale_fill_geo(dat, ...)

scale_discrete_geo(dat, aesthetics, ...)
```

Arguments

dat	Either A) a string indicating a built-in dataframe with interval data from the ICS ("periods", "epochs", "stages", "eons", or "eras"), B) a string indicating a timescale from macrostrat (see list here: https://macrostrat.org/api/defs/timescales?all), or C) a custom data.frame of time interval boundaries (see coord_geo()).
...	Arguments passed on to ggplot2::discrete_scale scale_name The name of the scale that should be used for error messages associated with this scale. name The name of the scale. Used as the axis or legend title. If waiver(), the default, the name of the scale is taken from the first mapping used for that aesthetic. If NULL, the legend title will be omitted. labels One of: • NULL for no labels • waiver() for the default labels computed by the transformation object • A character vector giving labels (must be same length as breaks) • An expression vector (must be the same length as breaks). See ?plotmath for details. • A function that takes the breaks as input and returns labels as output. Also accepts rlang lambda function notation. limits One of: • NULL to use the default scale values • A character vector that defines possible values of the scale and their order • A function that accepts the existing (automatic) values and returns new ones. Also accepts rlang lambda function notation. na.translate Unlike continuous scales, discrete scales can easily show missing values, and do so by default. If you want to remove missing values from a discrete scale, specify na.translate = FALSE.

`na.value` If `na.translate = TRUE`, what aesthetic value should the missing values be displayed as? Does not apply to position scales where `NA` is always placed at the far right.

`drop` Should unused factor levels be omitted from the scale? The default, `TRUE`, uses the levels that appear in the data; `FALSE` uses all the levels in the factor.

`guide` A function used to create a guide or its name. See `guides()` for more information.

`super` The super class to use for the constructed scale

`aesthetics` Character string or vector of character strings listing the name(s) of the aesthetic(s) that this scale works with. This can be useful, for example, to apply colour settings to the `colour` and `fill` aesthetics at the same time, via `aesthetics = c("colour", "fill")`.

Examples

```
library(ggplot2)
df <- data.frame(
  x = runif(1000, 0, 10), y = runif(1000, 0, 10),
  color = sample(periods$name, 1000, TRUE), shape = 21
)
ggplot(df) +
  geom_point(aes(x = x, y = y, fill = color), shape = 21) +
  scale_fill_geo("periods", name = "Period") +
  theme_classic()

# cut continuous variable into discrete
df <- data.frame(x = runif(1000, 0, 1000), y = runif(1000, 0, 8))
df$color <- cut(df$x, c(periods$min_age, periods$max_age[22]), periods$name)
ggplot(df) +
  geom_point(aes(x = x, y = y, color = color)) +
  scale_x_reverse() +
  scale_color_geo("periods", name = "Period") +
  coord_geo(xlim = c(1000, 0), ylim = c(0, 8)) +
  theme_classic()
```

---

stages *Stage data from the International Commission on Stratigraphy (v2022/10)*

Description

A dataset containing the boundary ages, abbreviations, and colors for the stages of the Geologic Time Scale. Based on The ICS International Chronostratigraphic Chart (v2022/10), by Cohen, Finney, Gibbard, and Fan.

Usage

```
stages
```

Format

A data frame with 102 rows and 5 variables:

name stage name

max_age maximum age, in millions of years

min_age minimum age, in millions of years

abbr stage name abbreviations

color the colors for each stage, according to the Commission for the Geological Map of the World

Source

https://stratigraphy.org via https://macrostrat.org/api/v2/defs/intervals?timescale=international%20ages

See Also

Other timescales: eons, epochs, eras, periods

Index