Package 'deeptime'

August 19, 2024

Title Plotting Tools for Anyone Working in Deep Time

Version 2.0.0

Maintainer William Gearty <willgearty@gmail.com>

Description Extends the functionality of other plotting packages (notably 'ggplot2') 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://williamgearty.com/deeptime/, https://github.com/willgearty/deeptime

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

Depends R (>= 3.4)

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

LazyData true

Imports ggplot2 (>= 3.5.0), utils, ggforce, grid, gridExtra, gtable, methods, stats, lattice, rlang (>= 1.1.0), scales, ggfittext, curl, cli, lifecycle, grImport2

Suggests geomtextpath, phytools, dplyr, divDyn, gsloid, ape, palaeoverse, paleotree, dispRity, ggtree (>= 3.6.1), testthat (>= 3.0.0), vdiffr (>= 1.0.0), knitr, rmarkdown, withr, rsvg, ggpattern, ggrepel, rmacrostrat, svglite

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 2024-08-19 07:00:43 UTC

Contents

`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,
  alpha = 1,
  height = unit(2, "line"),
  bord = c("left", "right", "top", "bottom"),
  lwd = 0.25,
  color = "black",
  lab = TRUE,
  lab_color = NULL,
  rot = 0,
  family = "sans",
  fontface = "plain",
  size = 5,
  skip = c("Quaternary", "Holocene", "Late Pleistocene"),
  abbrv = TRUE,
  neg = FALSE,
  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? 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 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.
alpha	The transparency of the fill colors.

height	The height (or width if pos is left or right) of the scale.
bord	A vector specifying on which sides of the scale to add borders (same options as pos).
lwd	Line width.
color	The outline color of the interval boxes.
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).
family	The font family to use for the labels. There are only three fonts that are guaranteed to work everywhere: "sans" (the default), "serif", or "mono".
fontface	The font face to use for the labels. The standard options are "plain" (default), "bold", "italic", and "bold.italic".
size	Label size. Either a number as you would specify in ggplot2::geom_text() or "auto" to use ggfittext::geom_fit_text().
skip	A vector of interval names indicating which intervals should not be labeled. If abbrv is TRUE, this can also include interval abbreviations.
abbrv	If including labels, should the labels be abbreviated? If TRUE, the abbr column will be used for the labels. If FALSE, the name column will be used for the labels. If "auto", the abbreviate() function will be used to abbreviate the values in the name column. Note that the built-in data and data retrieved via get_scale_data() already have built-in abbreviations. However, using the "auto" option here will create new unique abbreviations based on only the intervals that are being plotted. In many cases, this may result in abbreviations that are shorter in length because there are fewer similar interval names to abbreviate.
neg	Set this to TRUE if your x-axis is using negative values.
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, alpha, height, bord, lwd, color, lab, lab_color, rot, family, fontface, size, skip, abbrv, neg, 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

[Deprecated]

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 = "on",
  fill = NULL,
  alpha = 1,
  lwd = 0.25,
  color = "grey80",
  lty = "solid",
  lab = FALSE,
  abbrv = TRUE,
  skip = c("Quaternary", "Holocene", "Late Pleistocene"),
  neg = TRUE,
  prop = 1,
  textpath_args = list()
)
```

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.
lab	Whether to include labels. Requires the geomtextpath package.
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.
neg	Set this to true if your theta-axis is using negative values. This is usually true if you are using ggtree.
prop	This is the rotational proportion of the background that the scale takes up.
textpath_args	A list of named arguments to provide to geomtextpath::geom_textpath(). Only used if lab is set to TRUE. Useful arguments include color (font color), family (font family), fontface, hjust (radial adjustment), and size (font size).

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 abbr column is optional and lists abbreviations that may be used as labels.
- 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, color, lty, lab, abbrv, skip, neg, prop, and textpath_args can also be lists (N.B. textpath_args would be a list of 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.

coord_geo_polar manually generates the r axis, meaning it does not support changing the guide features of ggplot v. 2.5.0 or later. However, the deeptime.axis.line.r, deeptime.axis.text.r, deeptime.axis.ticks.r, and deeptime.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 deeptime.axis.line.r is element_line(). The default value for deeptime.axis.text.r is element_text(size = 3.5, vjust = -2, hjust = NA). The default value for deeptime.axis.ticks.r is element_line(). The default value for deeptime.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).

Care must be taken when adding labels to plots, as they are very likely to overlap with the plot under the default settings. The textpath_args argument can be used to adjust the settings for the plotting of the labels. See geomtextpath::geom_textpath() for details about the available arguments. Also note that the curvature of the labels may vary based on the distance from the origin. This is why abbrv is set to TRUE by default.

Life cycle

This function is soft-deprecated in favor of coord_geo_radial() as of **deeptime** version 1.1.0. There is currently no plan to remove this function, but users are strongly encouraged to migrate to the new function for enhanced polar functionality.

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(deeptime.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_geo_radial *Enhanced polar coordinate system with geological timescale*

Description

coord_geo_radial behaves similarly to ggplot2::coord_radial() 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. coord_geo_radial is similar to coord_geo_polar() but has more options related to the polar coordinate plotting that are inherited from ggplot2::coord_radial() (e.g., end, r_axis_inside, inner.radius). Furthermore, unlike coord_geo_polar, coord_geo_radial uses the ggplot2 internals to draw the r and theta axes, gridlines, etc. This means that users can tweak the guide and theme settings for these features (see examples).

Usage

```
coord_geo_radial(
  dat = "periods",
  theta = "y",
  start = -0.5 * pi,
  end = 1.25 * pi,
  expand = TRUE,
  direction = 1,
  r_axis_inside = NULL,
  inner.radius = 0.05,
  fill = NULL,
  alpha = 1,
  lwd = 0.25,
  color = "grey80",
  lty = "solid",
  lab = FALSE,
  abbrv = TRUE,
  skip = c("Quaternary", "Holocene", "Late Pleistocene"),
  neg = TRUE,
  prop = 1,
  textpath_args = list(),
  clip = "off",
  rotate_angle = FALSE
)
```

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.
end	Position from 12 o'clock in radians where plot ends, to allow for partial polar coordinates. The default, NULL, is set to start + 2 * pi.
expand	If TRUE, the default, adds a small expansion factor the the limits to prevent overlap between data and axes. If FALSE, limits are taken directly from the scale.
direction	1, clockwise; -1, anticlockwise
r_axis_inside, rotate_angle	**[Deprecated]**
inner.radius	A numeric between 0 and 1 setting the size of a inner.radius hole.
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.
lab	Whether to include labels. Requires the geomtextpath package.
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.
neg	Set this to true if your theta-axis is using negative values. This is usually true if you are using ggtree.
prop	This is the rotational proportion of the background that the scale takes up.
textpath_args	A list of named arguments to provide to geomtextpath::geom_textpath(). Only used if lab is set to TRUE. Useful arguments include color (font color), family (font family), fontface, hjust (radial adjustment), and size (font size).
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().

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 abbr column is optional and lists abbreviations that may be used as labels.
- 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, color, lty, lab, abbrv, skip, neg, prop, and textpath_args can also be lists (N.B. textpath_args would be a list of 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.

Care must be taken when adding labels to plots, as they are very likely to overlap with the plot under the default settings. The textpath_args argument can be used to adjust the settings for the plotting of the labels. See geomtextpath::geom_textpath() for details about the available arguments. Also note that the curvature of the labels may vary based on the distance from the origin. This is why abbrv is set to TRUE by default.

Examples

```
library(ggplot2)

library(ggtree)
set.seed(1)
tree <- rtree(100)
# single scale
revts(ggtree(tree)) +
  coord_geo_radial(dat = "stages") +
  scale_y_continuous(guide = "none", breaks = NULL) +
  theme_gray()

# multiple scales
revts(ggtree(tree)) +
  coord_geo_radial(
    dat = list("stages", "periods"), alpha = .5,
    prop = list(0.75, .25), start = pi / 4, end = 2 * pi, lty = "dashed"
  ) +
  scale_y_continuous(expand = expansion(mult = c(0.02, 0.02)),
                     guide = "none", breaks = NULL) +
  theme_gray()


library(ggplot2)
library(paleotree)
data(RaiaCopesRule)
ggtree(ceratopsianTreeRaia,
       position = position_nudge(x = -ceratopsianTreeRaia$root.time)) +
  coord_geo_radial(dat = "stages") +
  scale_y_continuous(guide = "none", breaks = NULL) +
  theme_classic()
```

`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.

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.

Details

This coordinate system 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()). Furthermore, when used with ggplot2 3.5.0 and later, some transformation edge cases may cause problems with rendering axis lines. This includes not currently support "capping" axes. I hope to support all of these geoms, edge cases, and features in the future.

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 (v2023/06)*

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 (v2023/06), 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 (v2023/06)*

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 (v2023/06), 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 (v2023/06)*

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 (v2023/06), 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

facet_grid_color *Lay out panels in a grid with colored strips*

Description

facet_grid_color behaves similarly to ggplot2::facet_grid() in that it forms a matrix of panels defined by row and column faceting variables. The main difference is that it also allows the user to specify the background colors of the individual facet strips using the colors argument. If you have only one variable with many levels, try facet_wrap_color().

Usage

```
facet_grid_color(
  rows = NULL,
  cols = NULL,
  scales = "fixed",
  space = "fixed",
  shrink = TRUE,
  labeller = "label_value",
  colors = stages,
  as.table = TRUE,
  switch = NULL,
  drop = TRUE,
  margins = FALSE,
  axes = "margins",
  axis.labels = "all"
)
```

Arguments

rows, cols	A set of variables or expressions quoted by vars() and defining faceting groups on the rows or columns dimension. The variables can be named (the names are passed to labeller). For compatibility with the classic interface, rows can also be a formula with the rows (of the tabular display) on the LHS and the columns (of the tabular display) on the RHS; the dot in the formula is used to indicate there should be no faceting on this dimension (either row or column).
scales	Are scales shared across all facets (the default, "fixed"), or do they vary across rows ("free_x"), columns ("free_y"), or both rows and columns ("free")?
space	If "fixed", the default, all panels have the same size. If "free_y" their height will be proportional to the length of the y scale; if "free_x" their width will be proportional to the length of the x scale; or if "free" both height and width will vary. This setting has no effect unless the appropriate scales also vary.
shrink	If TRUE, will shrink scales to fit output of statistics, not raw data. If FALSE, will be range of raw data before statistical summary.

`labeller` A function that takes one data frame of labels and returns a list or data frame of character vectors. Each input column corresponds to one factor. Thus there will be more than one with `vars(cyl, am)`. Each output column gets displayed as one separate line in the strip label. This function should inherit from the "labeller" S3 class for compatibility with `labeller()`. You can use different labeling functions for different kind of labels, for example use `label_parsed()` for formatting facet labels. `label_value()` is used by default, check it for more details and pointers to other options.

`colors` Specifies which colors to use to replace the strip backgrounds. Either A) a function that returns a color for a given strip label, B) the character name of a function that does the same, C) a named character vector with names matching strip labels and values indicating the desired colors, or D) a data.frame representing a lookup table with columns named "name" (matching strip labels) and "color" (indicating desired colors). If the function returns

`as.table` If `TRUE`, the default, the facets are laid out like a table with highest values at the bottom-right. If `FALSE`, the facets are laid out like a plot with the highest value at the top-right.

`switch` By default, the labels are displayed on the top and right of the plot. If `"x"`, the top labels will be displayed to the bottom. If `"y"`, the right-hand side labels will be displayed to the left. Can also be set to `"both"`.

`drop` If `TRUE`, the default, all factor levels not used in the data will automatically be dropped. If `FALSE`, all factor levels will be shown, regardless of whether or not they appear in the data.

`margins` Either a logical value or a character vector. Margins are additional facets which contain all the data for each of the possible values of the faceting variables. If `FALSE`, no additional facets are included (the default). If `TRUE`, margins are included for all faceting variables. If specified as a character vector, it is the names of variables for which margins are to be created.

`axes` Determines which axes will be drawn. When `"margins"` (default), axes will be drawn at the exterior margins. `"all_x"` and `"all_y"` will draw the respective axes at the interior panels too, whereas `"all"` will draw all axes at all panels.

`axis.labels` Determines whether to draw labels for interior axes when the `axes` argument is not `"margins"`. When `"all"` (default), all interior axes get labels. When `"margins"`, only the exterior axes get labels and the interior axes get none. When `"all_x"` or `"all_y"`, only draws the labels at the interior axes in the x- or y-direction respectively.

Examples

```
library(ggplot2)
df <- data.frame(x = 1:10, y = 1:10, period = c("Permian", "Triassic"))
ggplot(df) +
  geom_point(aes(x, y)) +
  facet_grid_color(cols = vars(period), colors = periods)
```

`facet_wrap_color` *Wrap a 1d ribbon of panels into 2d with colored strips*

Description

`facet_wrap_color` behaves similarly to `ggplot2::facet_wrap()` in that it wraps a 1d sequence of panels into 2d. The main difference is that it also allows the user to specify the background colors of the individual facet strips using the `colors` argument. This is generally a better use of screen space than `facet_grid_color()` because most displays are roughly rectangular.

Usage

```
facet_wrap_color(
  facets,
  nrow = NULL,
  ncol = NULL,
  scales = "fixed",
  shrink = TRUE,
  labeller = "label_value",
  colors = stages,
  as.table = TRUE,
  drop = TRUE,
  dir = "h",
  strip.position = "top",
  axes = "margins",
  axis.labels = "all"
)
```

Arguments

`facets`	A set of variables or expressions quoted by `vars()` and defining faceting groups on the rows or columns dimension. The variables can be named (the names are passed to `labeller`). For compatibility with the classic interface, can also be a formula or character vector. Use either a one sided formula, `~a + b`, or a character vector, `c("a", "b")`.
`nrow, ncol`	Number of rows and columns.
`scales`	Should scales be fixed (`"fixed"`, the default), free (`"free"`), or free in one dimension (`"free_x"`, `"free_y"`)?
`shrink`	If `TRUE`, will shrink scales to fit output of statistics, not raw data. If `FALSE`, will be range of raw data before statistical summary.
`labeller`	A function that takes one data frame of labels and returns a list or data frame of character vectors. Each input column corresponds to one factor. Thus there will be more than one with `vars(cyl, am)`. Each output column gets displayed as one separate line in the strip label. This function should inherit from the "labeller" S3 class for compatibility with `labeller()`. You can use different

	labeling functions for different kind of labels, for example use `label_parsed()` for formatting facet labels. `label_value()` is used by default, check it for more details and pointers to other options.
`colors`	Specifies which colors to use to replace the strip backgrounds. Either A) a function that returns a color for a given strip label, B) the character name of a function that does the same, C) a named character vector with names matching strip labels and values indicating the desired colors, or D) a data.frame representing a lookup table with columns named "name" (matching strip labels) and "color" (indicating desired colors). If the function returns
`as.table`	If `TRUE`, the default, the facets are laid out like a table with highest values at the bottom-right. If `FALSE`, the facets are laid out like a plot with the highest value at the top-right.
`drop`	If `TRUE`, the default, all factor levels not used in the data will automatically be dropped. If `FALSE`, all factor levels will be shown, regardless of whether or not they appear in the data.
`dir`	Direction: either `"h"` for horizontal, the default, or `"v"`, for vertical.
`strip.position`	By default, the labels are displayed on the top of the plot. Using `strip.position` it is possible to place the labels on either of the four sides by setting `strip.position = c("top", "bottom", "left", "right")`
`axes`	Determines which axes will be drawn in case of fixed scales. When `"margins"` (default), axes will be drawn at the exterior margins. `"all_x"` and `"all_y"` will draw the respective axes at the interior panels too, whereas `"all"` will draw all axes at all panels.
`axis.labels`	Determines whether to draw labels for interior axes when the scale is fixed and the `axis` argument is not `"margins"`. When `"all"` (default), all interior axes get labels. When `"margins"`, only the exterior axes get labels, and the interior axes get none. When `"all_x"` or `"all_y"`, only draws the labels at the interior axes in the x- or y-direction respectively.

Examples

```
library(ggplot2)
df <- data.frame(x = 1:10, y = 1:10, period = c("Permian", "Triassic"))
ggplot(df) +
  geom_point(aes(x, y)) +
  facet_wrap_color(vars(period), colors = periods)
```

`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	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".
...	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(). Note that phytools is not necessarily installed with deeptime, but it is required to use this function. Following the estimation of the ancestral states, 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)
```

---

geom_points_range *Display points and their range*

Description

This geom is like ggplot2::geom_pointrange() in that it draws points and lines. However, unlike ggplot2::geom_pointrange(), this geom takes in sets of x-y points and calculates the ranges/intervals based on those. It then plots both the original points and the ranges using ggplot2::geom_linerange(). In cases where not all points are connected (because of grouping due to aesthetics), the background_line argument can be used to add lines that span the entire point range for each x or y category.

Usage

```
geom_points_range(
  mapping = NULL,
  data = NULL,
  stat = "points_range",
  position = "identity",
  ...,
  na.rm = FALSE,
  orientation = NA,
  background_line = NULL,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_points_range(
  mapping = NULL,
  data = NULL,
  geom = "points_range",
  position = "identity",
  ...,
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

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 the 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".

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".

... — Arguments passed on to both ggplot2::geom_linerange() and ggplot2::geom_point().

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

orientation — The orientation of the layer. The default (NA) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting orientation to either "x" or "y". See the *Orientation* section for more detail.

background_line — A named list of aesthetic values to use for plotted line segments that span the entire y or x range for each x or y category. The default aesthetics will be used for any aesthetics that are not specified in the list. This can be useful if the plotted groups of points don't overlap but you want a continuous line connecting all points for a given x or y category. If NULL (the default), no line segments will be plotted.

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().

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".

Aesthetics

geom_points_range() understands the following aesthetics (required aesthetics are in bold):

- **x**
- **y**
- size

- color/colour
- fill
- shape
- alpha
- group
- linetype
- linewidth

Computed variables

These are calculated by the 'stat' part of layers and can be accessed with delayed evaluation. `stat_points_range()` provides the following variables, some of which depend on the orientation:

- `after_stat(ymin)` *or* `after_stat(xmin)`
  the minimum extent of the point range
- `after_stat(ymax)` *or* `after_stat(xmax)`
  the maximum extent of the point range

Orientation

This geom treats each axis differently and, thus, can thus have two orientations. Often the orientation is easy to deduce from a combination of the given mappings and the types of positional scales in use. Thus, ggplot2 will by default try to guess which orientation the layer should have. Under rare circumstances, the orientation is ambiguous and guessing may fail. In that case the orientation can be specified directly using the `orientation` parameter, which can be either `"x"` or `"y"`. The value gives the axis that the geom should run along, `"x"` being the default orientation you would expect for the geom.

Examples

```
library(ggplot2)

library(palaeoverse)
data(tetrapods)
tetrapod_names <- tetrapods$accepted_name[1:50]
beds_sampled <- sample.int(n = 10, size = 50, replace = TRUE)
occdf <- data.frame(taxon = tetrapod_names, bed = beds_sampled)
ggplot(occdf, aes(y = reorder(taxon, bed, min), x = bed)) +
  geom_points_range()
```

`geo_pattern` *Get a FGDC geologic plotting pattern*

Description

Retrieve a single geologic pattern as defined in the FGDC Digital Cartographic Standard for Geologic Map Symbolization by the U.S. Geological Survey and the Geologic Data Subcommittee (GDS) of the Federal Geographic Data Committee (FGDC).

Usage

```
geo_pattern(
  code,
  scale = 2,
  col = NULL,
  fill = NULL,
  alpha = NULL,
  bg = "white"
)

geo_grob(code, col = NULL, fill = NULL, alpha = NULL, bg = "white")
```

Arguments

`code`	The number corresponding to the pattern to return. Strings and numbers are permitted.
`scale`	The visual scale of the pattern (higher values mean the pattern is more zoomed in).
`col`	The color to use for the lines of the pattern.
`fill`	The color used to fill various closed shapes (e.g., circles) in the pattern.
`alpha`	The transparency to use for the fill of the pattern.
`bg`	The background color to use for the pattern.

Details

For specific codes, see the "pattern numbers" in the full pattern chart for valid `code` values. Daven Quinn has also assembled more accessible documentation of the map patterns/codes and lithology patterns/codes. `rmacrostrat::def_lithologies()` can also be used to look up pattern codes for various lithologies (see the "fill" column). Note that codes associated with color variants (e.g., "101-M") are supported but will result in the default color variant instead (usually black and white, e.g., "101-K").

These patterns were originally processed and optimized by Daven Quinn and are hosted on GitHub.

Value

`geo_grob()` returns a grob object with a single instance of the desired pattern. `geo_pattern()` returns a GridPattern object with a repeated instance of the desired pattern.

See Also

FGDC patterns: grid.pattern_geo(), scale_fill_geopattern()

Examples

```
library(grid)
# Get a generic igneous pattern
pattern1 <- geo_pattern(code = "313-K")
# Get the pattern for a sandstone
pattern2 <- geo_pattern(code = "607")

# plot the two patterns
grid.newpage()
grid.draw(rectGrob(gp = gpar(fill = pattern1)))
grid.newpage()
grid.draw(rectGrob(gp = gpar(fill = pattern2)))
```

---

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). Note that the colors in the built-in data.frames are according to the Commission for the Geological Map of the World. If you would like to obtain custom Macrostrat colors that are better for mapping, you should specify the full name of a timescale (e.g., "international periods") and set true_colors to FALSE. Note that these colors only vary for the Precambrian.

Usage

```
get_scale_data(name, true_colors = TRUE)
```

Arguments

name	The name of the desired timescale.
true_colors	Return original international time scale colors? (as opposed to custom Macrostrat plotting colors)

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)
```

`grid.pattern_geo` *Plot an individual FGDC pattern using grid*

Description

This function can be used to plot a single geologic pattern as defined in the FGDC Digital Cartographic Standard for Geologic Map Symbolization by the U.S. Geological Survey and the Geologic Data Subcommittee (GDS) of the Federal Geographic Data Committee (FGDC). The pattern is plotted on the existing canvas (i.e., use `grid::grid.newpage()` to make a new canvas).

Usage

```
grid.pattern_geo(params, boundary_df, aspect_ratio, legend = FALSE)
```

Arguments

`params`	A list of pattern parameters to customize the plotted pattern (see "Details").
`boundary_df`	A `data.frame` consisting of three columns: "x" (x-coordinates), "y" (y-coordinates), and "id" (polygon group ID). This `data.frame` defines the boundary (as a closed polygon) of the plotted pattern.
`aspect_ratio`	Unused.
`legend`	Unused.

Details

The following `params` are accepted:

`pattern_alpha` Alpha transparency for pattern. default: 1

`pattern_colour` Color used for strokes and points in the pattern. default: 'black'

`pattern_fill` Color used to fill various closed shapes (e.g., circles) in the pattern. default: `NA`

`pattern_scale` Scale. default: 2

`pattern_type` Code for the FGDC pattern to use. See `geo_pattern()` for more details. default: "101"

`fill` Color used for the background. default: "white"

Warning

Pattern fills are not supported on all graphics devices. Not all devices are under active development, and such devices are unlikely to add support for new features (such as pattern fills). The new features have only been implemented on a subset of graphics devices so far: `cairo_pdf()`, `cairo_ps()`, `x11(type="cairo")`, `png(type="cairo")`, `jpeg(type="cairo")`, `tiff(type="cairo")`, `svg()`, and `pdf()`. Although there is no support yet for `quartz()` or `windows()`, almost all of the graphics devices above will work on all major platforms. Further, the ragg and svglite packages contain graphics devices that support patterns. When using a graphics device where patterns are not supported, closed shapes will be rendered with a transparent fill. Note that, at least on Windows

machines, the default device in RStudio and in the knitr package is png(), which does not support patterns. In RStudio, you can go to 'Tools > Global Options > General > Graphics' and choose the 'Cairo PNG' device from the dropdown menu to display patterns. Similar issues may arise when using RStudio on other operating systems.

See Also

FGDC patterns: geo_pattern(), scale_fill_geopattern()

Examples

```
# use the function directly to make a hexagon with the pattern
library(grid)
x <- 0.5 + 0.5 * cos(seq(2 * pi / 4, by = 2 * pi / 6, length.out = 6))
y <- 0.5 + 0.5 * sin(seq(2 * pi / 4, by = 2 * pi / 6, length.out = 6))
grid.newpage()
grid.pattern_geo(params = list(pattern_type = "633", pattern_scale = 4),
                 boundary_df = data.frame(x, y, id = 1))

# use the function via ggpattern by specifying `pattern = 'geo'`
library(ggplot2)
library(ggpattern)
df <- data.frame(trt = c("a", "b", "c"), outcome = c(2.3, 1.9, 3.2))
ggplot(df, aes(trt, outcome)) +
  geom_col_pattern(aes(color = trt, pattern_type = trt), pattern = 'geo',
  pattern_color = "black", fill = "white", pattern_fill = "white") +
  scale_pattern_type_manual(values = c("101", "313", "634")) +
  scale_color_viridis_d() +
  theme(legend.key.size = unit(1.5, 'cm'))
```

---

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)
```

---

guide_geo *Geological timescale axis guide*

Description

guide_geo behaves similarly to ggplot2::guide_axis() in that it modifies the visual appearance of the axis. The main difference is that it adds a geological timescale instead of an axis.

Usage

```
guide_geo(
  dat = "periods",
  fill = NULL,
  alpha = 1,
  height = unit(2, "line"),
  bord = c("left", "right", "top", "bottom"),
  lwd = 0.25,
  color = "black",
  lab = TRUE,
  lab_color = NULL,
  rot = 0,
  family = "sans",
  fontface = "plain",
  size = 5,
  skip = c("Quaternary", "Holocene", "Late Pleistocene"),
  abbrv = TRUE,
  neg = FALSE,
  end_labels = "center",
  dat_is_discrete = FALSE,
  fittext_args = list(),
  theme = NULL,
  title = waiver(),
  order = 0,
  position = waiver()
)
```

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).
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.
alpha	The transparency of the fill colors.
height	The height (or width if pos is left or right) of the scale.
bord	A vector specifying on which sides of the scale to add borders (same options as pos).
lwd	Line width.
color	The outline color of the interval boxes.
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).

family — The font family to use for the labels. There are only three fonts that are guaranteed to work everywhere: "sans" (the default), "serif", or "mono".

fontface — The font face to use for the labels. The standard options are "plain" (default), "bold", "italic", and "bold.italic".

size — Label size. Either a number as you would specify in ggplot2::geom_text() or "auto" to use ggfittext::geom_fit_text().

skip — A vector of interval names indicating which intervals should not be labeled. If abbrv is TRUE, this can also include interval abbreviations.

abbrv — If including labels, should the labels be abbreviated? If TRUE, the abbr column will be used for the labels. If FALSE, the name column will be used for the labels. If "auto", the abbreviate() function will be used to abbreviate the values in the name column. Note that the built-in data and data retrieved via get_scale_data() already have built-in abbreviations. However, using the "auto" option here will create new unique abbreviations based on only the intervals that are being plotted. In many cases, this may result in abbreviations that are shorter in length because there are fewer similar interval names to abbreviate.

neg — Set this to TRUE if your x-axis is using negative values.

end_labels — How should labels for intervals at the ends of the guide be treated? "center", the default, centers the labels within the visible part of the label. "clip" removes the labels if their midpoint is beyond the axis limits. "keep" plots the labels in the midpoint of the full interval.

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".

theme — A theme object to style the guide individually or differently from the plot's theme settings. The theme argument in the guide overrides, and is combined with, the plot's theme.

title — A character string or expression indicating a title of guide. If NULL, the title is not shown. By default (waiver()), the name of the scale object or the name specified in labs() is used for the title.

order — A positive integer of length 1 that specifies the order of this guide among multiple guides. This controls in which order guides are merged if there are multiple guides for the same position. If 0 (default), the order is determined by a secret algorithm.

position — Where this guide should be drawn: one of top, bottom, left, or right.

Details

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.

Since this guide only plots the timescale and not ticks or an axis line, a call to this function should almost always be combined with a call to `ggplot2::guide_axis()` within a call to `ggplot2::guide_axis_stack()` (see Examples). Note that in most cases this has the same end result as a single call to `coord_geo()`; however, there are some use cases in which this may be more useful or allow for more customization. For example, users may wish to combine this guide in unique ways with other guides. Further, since `coord_geo()` doesn't work with radial/fan phylogenies (and `coord_geo_radial()` is quite different visually), this guide can be used to achieve the look of `coord_geo()` on a radial/fan phylogeny.

Examples

```
library(ggplot2)
# reproduce the coord_geo() appearance
ggplot() +
  geom_point(aes(y = runif(1000, 0, 8), x = runif(1000, 0, 1000))) +
  scale_x_reverse(guide = guide_axis_stack(guide_geo(), "axis",
                  spacing = unit(0, "npc"))) +
  coord_cartesian(xlim = c(1000, 0), ylim = c(0, 8)) +
  theme_classic()

# the coord_geo() look on a radial phylogeny
library(ggtree)
library(paleotree)
data(RaiaCopesRule)
ggtree(ceratopsianTreeRaia,
       position = position_nudge(x = -ceratopsianTreeRaia$root.time)) +
  coord_geo_radial(dat = "stages", fill = c("grey80", "grey95"),
                   end = 1.25 * pi) +
  guides(
    r = guide_axis_stack(guide_geo(rot = -90, neg = TRUE,
                                   height = unit(0.5, "line")),
                         "axis", spacing = unit(0, "npc"))
```

```
) +
scale_y_continuous(guide = "none", breaks = NULL) +
theme_classic()
```

`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 (v2023/06)*

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 (v2023/06), 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 **[Deprecated]** 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 includes the levels in the factor. Please note that to display every level in a legend, the layer should use `show.legend = TRUE`.

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

`call` The `call` used to construct the scale for reporting messages.

`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()
```

`scale_fill_geopattern` *Geologic pattern fill scale*

Description

Fill scale using the FGDC Digital Cartographic Standard for Geologic Map Symbolization. Fill values should correspond to specific pattern codes (see "Details").

Usage

```
scale_fill_geopattern(na.value = "grey50", ...)
```

Arguments

`na.value` The aesthetic value to use for missing (NA) values. May be either a color or a GridPattern object (such as that returned by `geo_pattern()`).

`...` Arguments passed on to `ggplot2::discrete_scale`

`scale_name` **[Deprecated]** 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.

`breaks` One of:

- `NULL` for no breaks
- `waiver()` for the default breaks (the scale limits)
- A character vector of breaks
- A function that takes the limits as input and returns breaks as output. Also accepts rlang lambda function notation.

`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`.

`drop` Should unused factor levels be omitted from the scale? The default, `TRUE`, uses the levels that appear in the data; `FALSE` includes the levels in the factor. Please note that to display every level in a legend, the layer should use `show.legend = TRUE`.

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

`call` The `call` used to construct the scale for reporting messages.

Details

For specific codes, see the "pattern numbers" in the full pattern chart for valid `code` values. Daven Quinn has also assembled more accessible documentation of the map patterns/codes and lithology patterns/codes. `rmacrostrat::def_lithologies()` can also be used to look up pattern codes for various lithologies (see the "fill" column). Note that codes associated with color variants (e.g., "101-M") are supported but will result in the default color variant instead (usually black and white, e.g., "101-K").

These patterns were originally processed and optimized by Daven Quinn and are hosted on GitHub.

Warning

Pattern fills are not supported on all graphics devices. Not all devices are under active development, and such devices are unlikely to add support for new features (such as pattern fills). The new features have only been implemented on a subset of graphics devices so far: `cairo_pdf()`, `cairo_ps()`, `x11(type="cairo")`, `png(type="cairo")`, `jpeg(type="cairo")`, `tiff(type="cairo")`, `svg()`, and `pdf()`. Although there is no support yet for `quartz()` or `windows()`, almost all of the graphics devices above will work on all major platforms. Further, the ragg and svglite packages contain graphics devices that support patterns. When using a graphics device where patterns are not supported, closed shapes will be rendered with a transparent fill. Note that, at least on Windows machines, the default device in RStudio and in the knitr package is `png()`, which does not support patterns. In RStudio, you can go to 'Tools > Global Options > General > Graphics' and choose the 'Cairo PNG' device from the dropdown menu to display patterns. Similar issues may arise when using RStudio on other operating systems.

See Also

FGDC patterns: `geo_pattern()`, `grid.pattern_geo()`

Examples

```
library(ggplot2)
vals <- c("101", "313", "603", "733")
ggplot(mpg, aes(factor(cyl), fill = vals[factor(cyl)])) +
  geom_bar() +
  scale_fill_geopattern(name = NULL)
```

`stages` *Stage data from the International Commission on Stratigraphy (v2023/06)*

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 (v2023/06), 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