

Package ‘ggplotplus’

July 27, 2026

Title Universal Design-Oriented Enhancements for 'ggplot2'

Version 0.5.6

BugReports <https://github.com/MAISRC/ggplotplus/issues>

Description A collection of enhancements to 'ggplot2', with a focus on creating Universally Designed, accessible graphs easily and quickly.

Depends R (>= 4.1.0)

License MIT + file LICENSE

Encoding UTF-8

URL <https://github.com/MAISRC/ggplotplus>

RoxygenNote 7.3.3

Language en-US

Imports ggplot2 (>= 4.0.1), rlang, S7, grid, gtable, scales,
viridisLite, polyclip, ggrepel (>= 0.9.0), dplyr, stats,
grDevices

Suggests testthat (>= 3.0.0), ragg (>= 1.5.2), patchwork, cowplot

Config/testthat/edition 3

LazyData true

NeedsCompilation no

Author Alex Bajcz [aut, cre] (ORCID: <<https://orcid.org/0000-0002-5909-4676>>)

Maintainer Alex Bajcz <bajcz003@umn.edu>

Repository CRAN

Date/Publication 2026-07-27 19:30:16 UTC

Contents

add_shape_plus	2
direct_labels_plus	3
geom_jitter_plus	6
geom_point_plus	8

geom_point_plus_shapes 10

ggplotplus_shapes_list 11

ggplotplus_to_cowplot 11

ggplotplus_to_patchwork 12

gridlines_plus 13

scale_continuous_plus 15

scale_focus_plus 16

theme_plus 20

yaxis_title_plus 22

Index **24**

add_shape_plus	<i>Add a custom shape for geom_point_plus()</i>
----------------	---

Description

Registers a custom point shape for use with geom_point_plus(). Custom shapes are defined by a data frame of connected polygon coordinates and, once validated, are stored in a session-level shape registry.

Usage

```
add_shape_plus(name = NULL, shape = NULL, overwrite = FALSE, ...)
```

Arguments

name	A single character string giving the name of the new shape.
shape	A data frame with columns x, y, and piece.
overwrite	Logical. If FALSE, the default, an error is returned when name already exists in the shape registry. If TRUE, the existing shape is replaced with the newly provided shape. Useful if, e.g., you'd like to scale an existing shape up or down in size.
...	Additional arguments. Partial argument matching is supported for friendly UX.

Details

A shape must be supplied as a data frame with columns x, y, and piece. The x and y columns define the vertices of the shape, centered around 0, 0. Coordinates should generally be scaled to about +/-0.4 to match the scaling of built-in point shapes. The piece column identifies separate polygon pieces within the same shape to generate "holes," as appropriate.

For inspiration and to model inputs, see ggplotplus_shapes_list for the structure of the built-in point shapes.

Value

Invisibly returns the registered shape name.

Examples

```
test_star = data.frame(
  x = c(0.000, 0.118, 0.380, 0.190, 0.235,
        0.000, -0.235, -0.190, -0.380, -0.118),
  y = c(0.400, 0.124, 0.124, -0.047, -0.324,
        -0.153, -0.324, -0.047, 0.124, 0.124),
  piece = 1
)

add_shape_plus("test_star", test_star)
```

direct_labels_plus	<i>Add direct labels to grouped point or line plots</i>
--------------------	---

Description

`direct_labels_plus()` adds procedurally placed text labels to grouped point or line plots as an alternative to using a legend, which might be space-inefficient, off to one side away from where readers will see it, or force readers to jump their focus long distances to align groups with labels. The function is intended as a friendlier alternative to manually placing group labels via `ggplot2::annotate()`. Labels are drawn using `ggrepel::geom_label_repel()`, so they'll repel one another as well as the plotted data they're labeling (within reason).

Usage

```
direct_labels_plus(
  data,
  x,
  y,
  group,
  placement = "top",
  geometry = "point",
  adj_fact = 0,
  key_labels = NULL,
  facet_vars = NULL,
  ...
)
```

Arguments

data	A data frame containing the variables to be both plotted and labelled. Most often, this will be the same data frame as supplied to <code>ggplot2::ggplot()</code> but needn't be.
x, y	Unquoted column names giving the x- and y-coordinates of the data to be plotted and thus also labelled.

group	Unquoted column name giving the grouping variable with which to label the underlying data. Must be a single unquoted column name and not an expression.
placement	Where labels should be placed relative to each group. One of "top", "right", "bottom", or "left". For geometry = "point", this controls the target location used to choose a representative point from each group. For geometry = "line", this chooses the endpoint or extreme point to label. Experimenting with different placements to find the one that works best for a particular graph is advised!
geometry	The kind of geometry being labelled. Currently supports "point" and "line".
adj_fact	A single numeric value controlling how far label targets are adjusted toward or away from the selected edge of each group. Values are interpreted as a proportion of the group-specific x- or y-range. Positive values move targets outward (towards the plot edge); negative values move them inward. For geometry = "point", this changes the target spot used to choose a particular point from each group to label, but does not move the final label anchor away from the selected point. In practice, this is likely not particularly useful for point geometries most of the time as a result.
key_labels	Optional replacement contents for the labels. May be one of: • NULL, in which case group values are used as labels (the default behavior); • A labelling function; • a named character vector of the form <code>c("old_group_name" = "New label")</code>; or • an unnamed character vector with one label per group. Unnamed labels will be assigned to groups in alphanumeric order.
facet_vars	Optional character vector of max length 2 (or else NULL) giving one or two faceting variables you're using to facet your plot. When supplied, label locations are calculated separately within each group-by-facet combination. This is useful when adding direct labels to faceted plots.
...	Additional arguments passed along to <code>ggrepel::geom_label_repel()</code> .

Details

This function is experimental. It currently works best for ordinary scatterplots and grouped line/path plots where each group has a visually meaningful position in two-dimensional (x/y) space. It will not work for every ggplot2 geometry, statistic, coordinate system, or faceting arrangement.

For point geometries, `direct_labels_plus()` calculates one label location per group by finding the observed point closest to a group-specific target positioned towards one of the "edges" of a group's cluster of points. For "top" and "bottom" placement, the target is horizontally centered on the group's median x-value and vertically placed near the group's maximum or minimum y-value. For "left" and "right" placement, the same logic is applied with x and y reversed.

For line geometries, labels are placed at the group-specific endpoint or extreme value implied by placement: the largest x-value for "right", the smallest x-value for "left", the largest y-value for "top", and the smallest y-value for "bottom". The final label anchor may then be adjusted by `adj_fact`.

To help labels repel from plotted points or lines, the function silently adds empty-label rows at the original data coordinates. `ggrepel` does not draw these empty labels, but still uses their positions

when placing visible labels. This helps to ensure, in general, that labels neither cover each other nor the underlying data they're labelling.

Currently, the function sets defaults for the following parameters of `ggrepel::geom_label_repel()`: size (5), box.padding (0.5), max.overlaps (Inf), segment.size (1), and min.segment.length (0). However, these are overridable, if the user provides named arguments that at least partially match.

Known limitations:

- Label locations are calculated in the raw data space supplied to the function. Transformed scales, reversed axes, and non-Cartesian coordinate systems may give unexpected results. Pre-transform data rather than entering raw data and transforming later.
- `coord_flip()`, `coord_polar()`, map projections, and other coordinate transformations are not currently supported.
- The function places labels according to the data values supplied in data, not those subsequently generated by ggplot2 stat functions. For example, to label fitted smooth lines like those from `ggplot2::geom_smooth()`, feed this function fitted values outputted from such a function instead of the raw data.
- Very dense plots or plots with many groups will likely still have overlapping or poorly placed labels. `ggrepel` helps, but it cannot make a crowded plot uncrowded. Alas.
- Polygon, ribbon, area, segment, curve, and spatial geometries are not currently supported but might be in the future.

Value

A ggplot2 layer produced by `ggrepel::geom_label_repel()`.

Examples

```
ggplot2::ggplot(iris, ggplot2::aes(Sepal.Length, Sepal.Width, color = Species)) +
  ggplot2::geom_point() +
  direct_labels_plus(
    data = iris,
    x = Sepal.Length,
    y = Sepal.Width,
    group = Species,
    placement = "right",
    geometry = "point"
  ) +
  ggplot2::guides(color = "none")

line_data = ChickWeight |>
  dplyr::group_by(Diet, Time) |>
  dplyr::summarize(weight = mean(weight), .groups = "drop")

ggplot2::ggplot(line_data, ggplot2::aes(Time, weight, color = Diet)) +
  ggplot2::geom_line(ggplot2::aes(group = Diet)) +
  direct_labels_plus(
    data = line_data,
    x = Time,
    y = weight,
```

```

group = Diet,
placement = "right",
geometry = "line",
adj_fact = 0.05
) +
ggplot2::guides(color = "none")

```

geom_jitter_plus

Jittered points with ggplotplus point shapes

Description

geom_jitter_plus() is a convenience wrapper around geom_point_plus() that applies jittering to reduce overplotting. It supports the same custom shape palette and fillable point rendering as geom_point_plus() while exposing the familiar width, height, and seed arguments used by ggplot2::position_jitter().

Usage

```

geom_jitter_plus(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "jitter",
  ...,
  width = NULL,
  height = NULL,
  seed = NA,
  na.rm = FALSE,
  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. <code>~ head(.x, 10)</code>).

stat	The statistical transformation to use on the data for this layer. When using a <code>geom_*()</code> function to construct a layer, the <code>stat</code> argument can be used to override the default coupling between geoms and stats. The <code>stat</code> argument accepts the following: • A Stat ggproto subclass, for example <code>StatCount</code>. • A string naming the stat. To give the stat as a string, strip the function name of the <code>stat_</code> prefix. For example, to use <code>stat_count()</code>, give the stat as <code>"count"</code>. • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The <code>position</code> argument accepts the following: • The result of calling a position function, such as <code>position_jitter()</code>. 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 <code>position_</code> prefix. For example, to use <code>position_jitter()</code>, give the position as <code>"jitter"</code>. • For more information and other ways to specify the position, see the layer position documentation.
...	Additional arguments passed to <code>geom_point_plus()</code> , including ggplotplus-specific arguments such as <code>chosen_shapes</code> and <code>legend_title</code> . See <code>?geom_point_plus</code> for details.
width, height	Amount of horizontal and vertical jitter. Passed to <code>ggplot2::position_jitter()</code> when <code>position = "jitter"</code> .
seed	Random seed used by <code>ggplot2::position_jitter()</code> to make jittering reproducible. Defaults to NA, matching <code>ggplot2</code> .
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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders() .

Value

A ggplot2 layer.

Examples

```
ggplot2::ggplot(iris, ggplot2::aes(Species, Sepal.Length)) +
  geom_jitter_plus(
```

```

ggplot2::aes(shape = Species, fill = Petal.Length),
width = 0.15,
seed = 1,
colour = "black"
)

ggplot2::ggplot(iris, ggplot2::aes(Species, Sepal.Length)) +
  geom_jitter_plus(
    ggplot2::aes(shape = Species),
    chosen_shapes = c("plus", "flower", "lotus"),
    seed = 123
  )

```

geom_point_plus	Create and add a scatterplot layer to your ggplot2 graph with new, distinctive shapes.
-----------------	--

Description

This function behaves similarly to `ggplot2::geom_point()` except that it takes several new inputs: `shapes`, `n_shapes`, `shape_values`, `legend_title`, `key_size`, and `show_shape_scale`. These are explained below.

Usage

```

geom_point_plus(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  avail_shapes = NULL,
  n_shapes = length(avail_shapes),
  chosen_shapes = NULL,
  legend_title = NULL,
  legend_labels = NULL,
  include_shape_legend = TRUE,
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  show_shape_scale = TRUE
)

```

Arguments

<code>mapping</code>	Set of aesthetic mappings created by <code>aes()</code> , as in <code>ggplot2::geom_point()</code> .
<code>data</code>	The data to be displayed in this layer, as in <code>ggplot2::geom_point()</code> .

stat	The statistical transformation to use on the data for this layer, as in <code>ggplot2::geom_point()</code> .
position	A position adjustment to use on the data for this layer, as in <code>ggplot2::geom_point()</code> .
avail_shapes	A named list of custom shapes to be drawn in place of <code>ggplot2</code> 's standard palette of shapes. Defaults to <code>NULL</code> and is replaced internally with the palette of shapes designed specifically for use in <code>geom_point_plus()</code> . This should probably not be changed unless users have created new shapes they would like to use instead.
n_shapes	A length-1 integer corresponding to the number of distinct shapes the function is allowed to pull from the shapes palette specified to <code>avail_shapes</code> . Defaults to the length of <code>avail_shapes</code> and should probably not be changed.
chosen_shapes	A character string referring by name to elements in the current shapes registry that the function should use to allocate shapes to values, e.g. <code>c("flower", "octagon", "squircle")</code> . These are provided internally to a <code>scale_shape_manual()</code> call and are meant to circumvent the need for such a call to specify a specific subset of shapes to be used from the new shapes palette. Defaults to <code>NULL</code> , i.e., shapes are pulled from <code>shapes.list</code> in order. Numerical values will use <code>ggplot2</code> 's default shapes instead.
legend_title	A length-1 character string corresponding to the name to be used for the shape legend title (if any). This is passed internally to <code>scale_shape_manual()</code> and is meant to help circumvent the need for the user to specify any such call directly.
legend_labels	A character vector corresponding to the names to be used for the shape legend labels (if any). This is passed internally to <code>scale_shape_manual()</code> and is meant to help circumvent the need for the user to specify any such call directly.
include_shape_legend	Logical indicating whether a shape legend will be shown (one is always shown unless this is set to <code>FALSE</code> , even when shape is being mapped to a constant and thus a legend may not be appropriate).
...	Other arguments passed on to this layer()'s <code>params</code> argument, as in <code>ggplot2::geom_point()</code> .
na.rm	Logical value controlling whether missing values should be removed from the data with a warning or silently, as in <code>ggplot2::geom_point()</code> .
show.legend	Logical value controlling whether this layer should be included in the legend(s), as in <code>ggplot2::geom_point()</code> .
inherit.aes	Logical controlling whether global aesthetics specified in <code>ggplot2::ggplot()</code> should be inherited locally by this layer or not, as in <code>ggplot2::geom_point()</code> .
show_shape_scale	Logical controlling whether a call to <code>ggplot2::shape_scale_manual()</code> should be included as part of the function's operations. Generally, this should be set to <code>TRUE</code> unless shape is being mapped to a constant, in which case leaving this <code>TRUE</code> would trigger a legend that is redundant.

Details

Collectively, these inputs allow `geom_point_plus()` to access and draw several new and distinctive shapes that are designed to be more readily distinguishable from one another when shape communicates difference.

To see the special shapes available via this function run `geom_point_plus_shapes()`.

Note: As of Version 0.5.2, shapes 21-25 in R's default shapes palette are now also available via `geom_point_plus_shapes()`; these are called "circle", "square", "diamond", "triangle_up", and "triangle_down", respectively, though they can also be referred to by number.

Value

A ggplot2 layer object.

Examples

```
ggplot2::ggplot(mtcars, ggplot2::aes(wt, mpg, fill = drat)) +  
  geom_point_plus(ggplot2::aes(shape = factor(gear)), size = 5)  
ggplot2::ggplot(mtcars, ggplot2::aes(wt, mpg, fill = factor(cyl))) +  
  geom_point_plus(ggplot2::aes(shape = factor(carb)),  
    shape_values = c("squircle", "lotus", "sunburst", "octagon", "cross", "oval"),  
    size = 5, stroke = 0.4)  
ggplot2::ggplot(iris, ggplot2::aes(Petal.Width, Petal.Length, fill = Species)) +  
  geom_point_plus(ggplot2::aes(shape = Species), size = 5, alpha = 0.7)
```

`geom_point_plus_shapes`

Demo plot showing geom_point_plus() shapes

Description

A prebuilt ggplot object displaying the nine custom point shapes designed specifically for use in `geom_point_plus()`. As of Version 0.5.2, other shapes are also available, but those are not shown via this function. See the documentation for `geom_point_plus()` for details.

Usage

```
geom_point_plus_shapes()
```

Format

A ggplot object.

Value

Returns a ggplot2 graph showing the ggplotplus shapes palette.

`ggplotplus_shapes_list`*Custom shape palette for geom_point_plus()*

Description

A named list of custom point shapes used by `geom_point_plus()`. Each element contains x and y coordinates plus a piece identifier used when drawing filled shapes and holes.

Usage

`ggplotplus_shapes_list`

Format

A named list of data frames. Each data frame has columns:

x X coordinates for the shape outline.

y Y coordinates for the shape outline.

piece Integer identifier for separate polygon pieces or holes.

Examples

`names(ggplotplus_shapes_list)`

`ggplotplus_to_cowplot` *Convert a ggplotplus plot into a grob*

Description

Convenience helper for converting ggplotplus plot objects into grobs using `ggplot2::ggplotGrob()`.

Usage

`ggplotplus_to_cowplot(plot)`

Arguments

`plot` A ggplot or ggplotplus plot object.

Details

This is primarily intended for compatibility with packages such as **cowplot** that may not yet fully recognize custom S7 plot subclasses during internal dispatch.

Value

A grob object suitable for use with grid-based plotting utilities such as `cowplot::plot_grid()`.

Examples

```
p = ggplot2::ggplot(iris, ggplot2::aes(Sepal.Length, Petal.Length, colour = Species)) +
  geom_point_plus()

if(requireNamespace("cowplot", quietly = TRUE)) {

  cowplot::plot_grid(
    ggplotplus_to_cowplot(p),
    ggplotplus_to_cowplot(p)
  )
}
```

`ggplotplus_to_patchwork`

Convert a ggplotplus plot into a patchwork-compatible element

Description

Convenience helper for converting ggplotplus plot objects into patchwork-compatible wrapped elements.

Usage

```
ggplotplus_to_patchwork(plot)
```

Arguments

`plot` A ggplot or ggplotplus plot object.

Details

This helper is primarily intended for compatibility with **patchwork**, whose plot-composition operators may not yet fully recognize custom S7 plot subclasses under ggplot2 4.x.

Internally, the plot is first converted into a grob using `ggplotplus_to_cowplot()`, then wrapped using `patchwork::wrap_elements()`.

This helper requires the **patchwork** package, but ggplotplus does not import patchwork. Users only need patchwork installed if they want to compose ggplotplus plots with patchwork.

Value

A patchwork-compatible wrapped plot element.

Examples

```

if(requireNamespace("patchwork", quietly = TRUE)) {

  p1 = ggplot2::ggplot(iris, ggplot2::aes(Sepal.Length, Petal.Length, colour = Species)) +
    geom_point_plus()

  p2 = ggplot2::ggplot(mtcars, ggplot2::aes(wt, mpg, shape = factor(cyl))) +
    geom_point_plus()

  (ggplotplus_to_patchwork(p1) |
   ggplotplus_to_patchwork(p2)) +
    patchwork::plot_annotation(
      title = "ggplotplus + patchwork"
    )
}

```

gridlines_plus

*Subtle, Minimal Gridlines For When and Where They Help***Description**

Adds light and easily ignored major gridlines along only axes mapped to continuous variables. Minor gridlines are blanked. This enables the benefits of gridlines (in instances where there are some) but minimizes visual clutter and cognitive load.

Usage

```

gridlines_plus(
  color = "gray90",
  linewidth = 1.2,
  linetype = "solid",
  notx = FALSE,
  noty = FALSE,
  override_legend_alphasize = TRUE,
  enable_coaching = TRUE
)

```

Arguments

color	Gridline color. Single character string. Default: "gray90".
linewidth	Gridline width (theme line units). Single numeric. Default: 1.2.
linetype	Gridline type. Single string (e.g., "solid", "dashed").
notx, noty	Logicals indicating whether gridlines should not be drawn in a specific direction (i.e., the user wants those to be blank even when the axis is continuous). Default to FALSE (gridlines are added).

override_legend_alphasize

Logical indicating whether, for any fill, color, and/or shape legends, to override the size and/or alpha values to the ggplotplus default values (1 and 5, respectively). Defaults to TRUE.

enable_coaching

Logical indicating whether ggplotplus should perform certain automated checks for possible non-Universal graph design and provide coaching messages when these are detected. Defaults to TRUE.

Details

gridlines_plus() ignores any theme instructions to blank major panel scales in either the x or y direction via, e.g., theme_plus(panel.grid.major.y = ggplot2::element_blank()). Instead, users can set, e.g., noty == TRUE to prevent gridlines from being drawn in a specific direction, even if that scale is continuous. Similarly, gridline color, linewidth, and linetypes should be set directly in gridlines_plus() instead of via theme(). Trying to do the latter will fail without error or warning!

Under the hood, gridlines_plus() checks layer and/or global mappings to see if x and/or y are continuous. It does this by inspecting the trained panel scales. It then turns on **major** gridlines for continuous directions and explicitly blanks gridlines for other axes (as well as **all minor** gridlines). One interaction exists between gridlines_plus() and theme_plus(): If using them in tandem, the linewidth provided to gridlines_plus() will be a function of the scaling factor generated according to the export_width and export_height inputs provided to theme_plus(). That is to say that the final gridline widths may be bigger or small than those specified in gridlines_plus() as a function of your intended output size.

Value

An ggplot class object for adding to a plot with +.

Examples

```
library(ggplot2)

ggplot2::ggplot(iris, ggplot2::aes(Sepal.Length, Petal.Length)) +
  ggplot2::geom_point() +
  theme_plus() + #WE DON'T RECOMMEND USING gridlines_plus() WITHOUT ALSO USING theme_plus()
  gridlines_plus()

# Only y is continuous here (x is discrete) → y-only major gridlines
ggplot2::ggplot(mtcars, ggplot2::aes(factor(cyl), mpg)) +
  ggplot2::geom_boxplot() +
  gridlines_plus(color = "grey85", linewidth = 1, linetype = "dashed")

# Works with derived continuous axes (histogram)
ggplot2::ggplot(mtcars, aes(mpg)) +
  ggplot2::geom_histogram() +
  gridlines_plus()
```

Description

`scale_continuous_plus()` is an opinionated wrapper around `ggplot2`'s continuous x, y, colour, and fill scales. It chooses "pretty" breaks while gently expanding the scale limits so breaks generally will appear near both ends of the data range.

Usage

```
scale_continuous_plus(
  scale = NA,
  ...,
  thin.labels = FALSE,
  pad.labels = "start",
  target.breaks = 5,
  buffer_frac = 0.05,
  split_name = FALSE
)
```

Arguments

<code>scale</code>	Character string specifying which scale to modify. Options are "x", "y", "colour"/"color", and "fill".
<code>...</code>	Additional arguments passed to the corresponding <code>ggplot2</code> continuous scale function. Arguments such as <code>name</code> , <code>labels</code> , <code>guide</code> , <code>position</code> , and <code>expand</code> may be supplied. User-supplied breaks, limits, <code>n.breaks</code> , <code>trans</code> , and <code>transform</code> are ignored with a warning because this function controls those components directly.
<code>thin.labels</code>	Logical. If TRUE, every other break label is blanked to reduce crowding. Defaults to FALSE.
<code>pad.labels</code>	Character string, either "start" or "end". Used when a user-supplied label vector is shorter than the internally computed break vector by one label (which must be assessed via trial and error currently). "start" pads a blank label at the beginning; "end" pads one at the end.
<code>target.breaks</code>	Integer target number of major breaks. This is a target and not a guarantee because breaks are chosen using a "pretty" break algorithm. Default is 5.
<code>buffer_frac</code>	Numeric fraction of the data span used to decide whether a break is close enough to each endpoint. Default is 0.05.
<code>split_name</code>	Logical. If TRUE, spaces in a named name argument are replaced with line breaks. This can help long axis or legend titles fit better. Default is FALSE.

Details

This is useful because ggplot2's default continuous scales frequently will leave the ends of an axis or colorbar visually unlabeled, making it look as if an endpoint break is missing.

scale_continuous_plus() routes to one of `ggplot2::scale_x_continuous()`, `ggplot2::scale_y_continuous()`, `ggplot2::scale_colour_continuous()`, or `ggplot2::scale_fill_continuous()` based on scale.

Unlike the ggplot2 defaults, this function intentionally controls breaks and limits. If either is supplied through ..., it's ignored with a warning. Transformed scales are also not currently supported; pre-transform the data or use ggplot2's scale functions directly when a transformed scale is needed.

User-supplied label vectors are supported, but endpoint-aware breaks can sometimes create hidden outer breaks. When possible, this function pads label vectors with blank labels to align them with the computed break vector in length. If alignment is ambiguous by one label, use `pad.labels` to choose which side of the input labels vector to pad.

Value

A ggplot2 continuous scale object.

Examples

```
library(ggplot2)

ggplot(iris, aes(Sepal.Width, Sepal.Length)) +
  geom_point() +
  scale_continuous_plus(scale = "x") +
  scale_continuous_plus(scale = "y")

ggplot(iris, aes(Sepal.Width, Sepal.Length, fill = Petal.Length)) +
  geom_point(shape = 21) +
  scale_continuous_plus(scale = "x") +
  scale_continuous_plus(scale = "y") +
  scale_continuous_plus(scale = "fill")

ggplot(iris, aes(Sepal.Width, Sepal.Length)) +
  geom_point() +
  scale_continuous_plus(
    scale = "y",
    name = "Sepal length",
    labels = LETTERS[1:5],
    pad.labels = "start"
  )
```


Description

`scale_focus_plus()` is a wrapper for `ggplot2`'s `scale_fill/colour_manual` functions that helps a user quickly creates a manual color or fill scale that highlights one or more focal groups while de-emphasizing all other groups (via desaturation, by default). This is useful when a graph would ideally call attention to specific groups without removing the broader context and transparency provided by the remaining groups. This reduces cognitive load by allowing non-essential information to "fall into the background" to be readily ignored, increasing the reading rate, and also helps to signpost for the reader where the "message" is within the plot.

Usage

```
scale_focus_plus(
  aes,
  group_var,
  focal_groups,
  diff_nonfocal = FALSE,
  diff_focal = TRUE,
  gray_start = 0.35,
  gray_end = 0.65,
  focal_start = 0.75,
  focal_end = 0.25,
  custom_focal = NULL,
  custom_nonfocal = NULL,
  ...
)
```

Arguments

<code>aes</code>	A character string indicating which aesthetic should receive the focus scale. Must be one of "color", "colour", or "fill".
<code>group_var</code>	A character vector or factor containing the grouping variable mapped to <code>aes</code> . This should generally be the same vector mapped to <code>colour</code> , <code>color</code> , or <code>fill</code> in the plot.
<code>focal_groups</code>	A character vector giving the group value(s) in the <code>group_var</code> to be emphasized. All values must be present in <code>group_var</code> .
<code>diff_nonfocal</code>	Logical. If TRUE, non-focal groups are assigned different (gray/custom) color values. If FALSE, all non-focal groups receive the same (gray/custom) value. Defaults to FALSE.
<code>diff_focal</code>	Logical. If TRUE, focal groups are assigned different (viridis/custom) colors. If FALSE, all focal groups receive the same (viridis/custom) color. Defaults to TRUE.
<code>gray_start, gray_end</code>	Numeric values between 0 and 1 controlling the range of gray values used for non-focal groups when <code>custom_nonfocal</code> is not NULL. Lower values are darker and higher values are lighter. <code>gray_start</code> must be less than or equal to <code>gray_end</code> .

focal_start, focal_end	Numeric values between 0 and 1 controlling the portions of the viridis palette used for focal groups when custom_focal is not NULL. By default, scale_focus_plus() draws from the darker and lighter ends of the palette while avoiding the middle, as humans are generally more drawn to very bright and very dark colors. focal_end must be less than or equal to focal_start.
custom_focal	Optional custom color(s) for focal group(s). When diff_focal = TRUE, this must be a named character vector whose names match the unique values in focal_groups. When diff_focal = FALSE, this must be a character vector of length 1.
custom_nonfocal	Optional custom color(s) for non-focal group(s). When diff_nonfocal = TRUE, this must be a named character vector whose names match the non-focal values in group_var. When diff_nonfocal = FALSE, this must be a character vector of length 1.
...	Additional arguments passed to <code>ggplot2::scale_colour_manual()</code> or <code>ggplot2::scale_fill_manual()</code> . Do not supply values; scale_focus_plus() constructs the values argument internally. All other arguments should be accessible, including labels and name. Use these to relabel and retitle the created scale.

Details

By default, focal groups are assigned visually prominent colors from the Universal-Design-oriented viridis color palette via `viridisLite::viridis()`, while non-focal groups are assigned a shared gray. Focal groups are differentiated from one another by default; non-focal groups are not. These defaults are intended to support a common graph-design pattern: show all groups, but make the intended comparison obvious.

Note that the default values for `gray_start`, `gray_end`, `focal_start`, and `focal_end` intentionally map to different regions of the luminance (light to dark) scale so as to render the colors still distinguishable in grayscale by virtue of variance in luminance. This variance should be maintained for accessibility even if different inputs are provided.

`scale_focus_plus()` is a convenience wrapper around `ggplot2::scale_colour_manual()` and `ggplot2::scale_fill_manual()`. It does not alter the data or add any geoms. Instead, it constructs a named vector of colors from `group_var` and `focal_groups`, then passes that vector to the appropriate ggplot2 manual color/fill scale.

This function is most useful when the focal/non-focal distinction is the primary visual message, and all non-focal groups may be secondary. When individual non-focal groups must remain distinguishable, set `diff_nonfocal = TRUE` or provide `custom_nonfocal`.

Although `scale_focus_plus()` is designed for discrete data, it can also be used when the data requiring (de-)emphasis is continuous. In those cases, first create a discrete grouping variable in the data, then map that variable to colour or fill. For example, a continuous measurement could be converted to groups such as "High cover" and "Other" before calling `scale_focus_plus()`. See the examples section for an example.

Value

A ggplot2 scale object.

Examples

```
lake_dat = data.frame(
  year = rep(2012:2020, times = 5),
  lake = rep(paste("Lake", LETTERS[1:5]), each = 9),
  cpue = c(
    10, 12, 13, 14, 16, 18, 19, 21, 23,
    18, 17, 16, 16, 15, 14, 13, 13, 12,
    8, 9, 11, 13, 16, 20, 24, 27, 30,
    22, 21, 21, 20, 19, 18, 18, 17, 17,
    12, 13, 13, 15, 15, 16, 17, 18, 20
  )
)

ggplot2::ggplot(
  lake_dat,
  ggplot2::aes(x = year,
    y = cpue,
    colour = lake,
    group = lake)
) +
  ggplot2::geom_line(linewidth = 1) +
  ggplot2::geom_point(size = 2) +
  scale_focus_plus(aes = "colour",
    group_var = lake_dat$lake,
    focal_groups = c("Lake A", "Lake C"))

# One can use a shared focal color and a custom non-focal gray.
ggplot2::ggplot(
  lake_dat,
  ggplot2::aes(x = year,
    y = cpue,
    colour = lake,
    group = lake)
) +
  ggplot2::geom_line(linewidth = 1) +
  scale_focus_plus(aes = "colour",
    group_var = lake_dat$lake,
    focal_groups = c("Lake A", "Lake C"),
    diff_focal = FALSE,
    custom_focal = "#440154",
    custom_nonfocal = "gray70")

cover_dat = data.frame(
  taxon = c("Native plants",
    "Starry stonewort",
    "Eurasian watermilfoil",
    "Curly-leaf pondweed"),
  mean_cover = c(62, 28, 18, 11)
)

ggplot2::ggplot(
  cover_dat,
```

```

ggplot2::aes(x = taxon,
             y = mean_cover,
             fill = taxon)
) +
ggplot2::geom_col() +
scale_focus_plus(aes = "fill",
                 group_var = cover_dat$taxon,
                 focal_groups = "Starry stonewort") +
ggplot2::labs(x = NULL,
              y = "Mean percent cover")

# Focal groups can also be created from continuous variables. Here's how:

```

theme_plus	<i>A Universal Design-Oriented Base Ggplot2 Theme With Scalable and Overridable Defaults</i>
------------	--

Description

theme_plus() returns a ggplot2 theme designed to make publication-quality, accessible graphs easier to produce. It keeps all of ggplot2's normal behaviors (last theme wins; user overrides take precedence), but bakes in opinionated defaults with Universal Design in mind. A few knobs let you scale typography/lines, flip the legend layout, and switch the background color if desired.

Usage

```

theme_plus(
  ...,
  legend_pos = "top",
  base_font_size = 16,
  base_linewidth = 1.2,
  base_rectlinewidth = 1.2,
  line_color = "black",
  text_color = "black",
  background_color = "#FFFEFD",
  palette_discrete = "D",
  palette_continuous = "E",
  begin_discrete = 0,
  end_discrete = 0.72,
  begin_continuous = 0,
  end_continuous = 1,
  export_width = 7.25,
  export_height = 5.95,
  override_legend_alphasize = TRUE,
  enable_coaching = TRUE
)

```

Arguments

...	Optional additional theme settings passed to <code>ggplot2::theme()</code> . These are applied <i>after</i> the base theme, so the theme's defaults only "win" when no matching settings are provided by the user (same as in <code>ggplot2</code>).
legend_pos	Where to put the legend. "top" (default) creates a horizontal stripe at the top for the legend (box) when one is present; "right" uses a vertical legend at the right (<code>ggplot2</code> 's usual position) but with design modifications. "bottom" is also an option (largely same as "top").
base_font_size	Base text size (in points) for most text elements. These will scale via <code>rel()</code> . Default is 16.
base_linewidth	Baseline thickness for most line theme elements (e.g., axis lines and tick marks). Defaults to 1.2. Specific elements may use <code>rel()</code> multipliers on top of this.
base_rectlinewidth	Baseline line thickness for most rect theme elements (e.g., legend frames). Defaults to 1.2.
line_color	Default color for most line elements (axis lines, frames, etc.). Defaults to "black".
text_color	Default color for most text elements. Defaults to "black".
background_color	Background fill applied to the panel, plot, legend, and strip backgrounds. Defaults to a slightly warm white, "#FFFEFD", to reduce eyestrain.
palette_discrete, palette_continuous	Default viridis-family color palette codes ("A" through "H") to use for discrete and continuous scales, respectively.
begin_discrete, end_discrete, begin_continuous, end_continuous	Numeric values ranging between 0 and 1 for where to begin drawing colors from a viridis palette for a discrete and continuous color scale, respectively.
export_width, export_height	Length-1 numeric values indicating your intended export (most likely via <code>ggplot2::ggsave()</code>) width and height, respectively. This rescales font and line sizes internally to stay relatively appropriately for your intended export size.
override_legend_alphasize	Logical indicating whether, for any fill, color, and/or shape legends, to override the size and/or alpha values to the <code>ggplotplus</code> default values (1 and 5, respectively). Defaults to TRUE.
enable_coaching	Logical indicating whether <code>ggplotplus</code> should perform certain automated checks for possible non-Universal graph design and provide coaching messages when these are detected. Defaults to TRUE.

Details

Internally, text sizes are expressed with `rel()`, so they scale with `base_font_size`. Line/rect line thicknesses start from `base_linewidth` and `base_rectlinewidth` and scale similarly with `rel()`. The function builds a base theme, then *adds* any user overrides via `theme(...)`, so the user's preferences always take precedence.

One interaction exists between `gridlines_plus()` and `theme_plus()`: If using them in tandem, the `linewidth` provided to `gridlines_plus()` will be a function of the scaling factor generated according to the `export_width` and `export_height` inputs provided to `theme_plus()`. That is to say that the final gridline widths may be bigger or small than those specified in `gridlines_plus()` as a function of your intended output size.

Value

A `ggplot2` theme object to add with `+`.

See Also

`ggplot2::theme()`, `ggplot2::theme_gray()`, `ggplot2::theme_get()`

Examples

```
# Basic use
library(ggplot2)
ggplot(iris, aes(Sepal.Length, Petal.Length, colour = Species)) +
  geom_point() +
  theme_plus()

# Prefer the right-side legend and pure white background
ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  theme_plus(legend_pos = "right", background_color = "white")

# Scale text up and make lines a bit lighter
ggplot(iris, aes(Sepal.Length, Petal.Length)) +
  geom_point() +
  theme_plus(base_font_size = 18, base_linewidth = 1.0)

# You can still override any element normally via theme()
ggplot(iris, aes(Sepal.Length, Petal.Length)) +
  geom_point() +
  theme_plus() +
  theme(axis.line = element_line(linewidth = 0.8))

# But you could just as easily do so via theme_plus()
ggplot(iris, aes(Sepal.Length, Petal.Length)) +
  geom_point() +
  theme_plus(axis.line = element_line(linewidth = 0.8))
```

yaxis_title_plus

Relocate a Y Axis Title to Above the Y Axis on a Ggplot and Turn it Horizontal.

Description

This function relocates the y axis title of a ggplot to the top, above the y axis line and left-justified to the left edge of the y axis labels, sort of like a plot subtitle. It also orients the text horizontally for space-efficiency and easy reading. This is otherwise difficult to do using ggplot2's default styling tools.

Usage

```
yaxis_title_plus(
  location = "top",
  nudgeTopLegendDown = FALSE,
  nudgeHowMuch = 20,
  override_legend_alphasize = TRUE,
  enable_coaching = TRUE
)
```

Arguments

location	A length-1 character string matching either "top" or "bottom" for the placement of the new y axis title. Defaults to "top". "bottom" should generally only be used when the x axis labels have been moved to the top of the graph (uncommon).
nudgeTopLegendDown	A length-1 logical indicating whether a top legend (box) (if any) should be moved down to align with the relocated y axis title (where they could clip into each other). Defaults to FALSE.
nudgeHowMuch	A length-1 positive integer indicating how much to nudge the top legend (box) (if any) down, if nudgeTopLegendDown == TRUE. Defaults to 20 points as a general guess and may need adjusting.
override_legend_alphasize	Logical indicating whether, for any fill, color, and/or shape legends, to override the size and/or alpha values to the ggplotplus default values (1 and 5, respectively). Defaults to TRUE.
enable_coaching	Logical indicating whether ggplotplus should perform certain automated checks for possible non-Universal graph design and provide coaching messages when these are detected. Defaults to TRUE.

Value

An ggplot class object for adding to a plot with +.

Examples

```
#WE DO NOT RECOMMEND USING yaxis_title_plus() WITHOUT theme_plus()
ggplot2::ggplot(iris, ggplot2::aes(x=Sepal.Length, y=Petal.Length)) +
  ggplot2::geom_point() +
  theme_plus() +
  yaxis_title_plus()
```

Index

- * **datasets**
 - ggplotplus_shapes_list, 11
- add_shape_plus, 2
- aes(), 6
- annotation_borders(), 7
- direct_labels_plus, 3
- direct_labels_plus(), 3
- fortify(), 6
- geom_jitter_plus, 6
- geom_point_plus, 8
- geom_point_plus_shapes, 10
- ggplot(), 6
- ggplot2::annotate(), 3
- ggplot2::geom_smooth(), 5
- ggplot2::ggplot(), 3
- ggplot2::scale_colour_continuous(), 16
- ggplot2::scale_colour_manual(), 18
- ggplot2::scale_fill_continuous(), 16
- ggplot2::scale_fill_manual(), 18
- ggplot2::scale_x_continuous(), 16
- ggplot2::scale_y_continuous(), 16
- ggplot2::theme(), 21, 22
- ggplot2::theme_get(), 22
- ggplot2::theme_gray(), 22
- ggplotplus_shapes_list, 11
- ggplotplus_to_cowplot, 11
- ggplotplus_to_patchwork, 12
- ggrepel::geom_label_repel(), 3–5
- gridlines_plus, 13
- layer position, 7
- layer stat, 7
- scale_continuous_plus, 15
- scale_focus_plus, 16
- theme_plus, 20
- yaxis_title_plus, 22