Package 'ggguides'

April 23, 2026

Title Simplified Legend and Guide Alignment for 'ggplot2'

Version 1.1.6

Description Provides one-liner functions for common legend and guide operations in 'ggplot2'. Simplifies legend positioning, styling, wrapping, and collection across multi-panel plots created with 'patchwork' or 'cowplot'.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Imports ggplot2 (>= 3.5.0), grid, gridExtra, gtable, rlang

Suggests patchwork, testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

URL https://gillescolling.com/ggguides/, https://github.com/gcol33/ggguides

BugReports https://github.com/gcol33/ggguides/issues

NeedsCompilation no

Author Gilles Colling [aut, cre]

Maintainer Gilles Colling <gilles.colling051@gmail.com>

Repository CRAN

Date/Publication 2026-04-23 10:30:06 UTC

Contents

`center_legend_title` *Center Legend Title Over Keys*

Description

Modifies a ggplot so that legend titles are centered over the key column only, rather than over the full legend width (keys + labels). This is particularly useful when legend labels are rotated, as the default centering places the title too far to the right.

Usage

```
center_legend_title(plot, position = "all")
```

Arguments

`plot`	A ggplot object.
`position`	Legend position to modify. One of `"right"`, `"left"`, `"top"`, `"bottom"`, or `"all"` (default).

Details

This function works by modifying the legend's internal gtable structure, restricting the title's column span to only the keys column. Long titles will automatically wrap to fit within the key column width, and proper spacing is added to prevent overlap with rotated labels.

The title should have `hjust = 0.5` set (done automatically by `legend_style(angle = ...)`) for proper centering.

Value

A modified gtable object that can be drawn with `grid::grid.draw()` or saved with `ggplot2::ggsave()`.

See Also

legend_style

Examples

```
library(ggplot2)

p <- ggplot(mpg, aes(displ, hwy, color = class)) +
  geom_point() +
  legend_style(angle = 45) +
  labs(color = "Vehicle Class")

# Center title over keys only (long titles wrap automatically)
# Returns a gtable - use grid::grid.draw() to render
g <- center_legend_title(p)
grid::grid.draw(g)
```

`collect_axes` *Collect Axes from Patchwork Compositions*

Description

Collects duplicate axes from multiple plots in a patchwork composition, removing redundant axis labels and ticks. This is a convenience wrapper around patchwork's axis collection functionality.

Usage

```
collect_axes(x, guides = c("collect", "keep"))

align_guides_h(x, guides = c("collect", "keep"))
```

Arguments

`x`	A patchwork object created by combining ggplot2 plots.
`guides`	How to handle guides. Either `"collect"` to combine into a single legend (default), or `"keep"` to maintain separate legends.

Details

This function applies patchwork's `axes = "collect"` layout option, which removes duplicate axes when plots are stacked or placed side-by-side. For example, in a 2x1 vertical layout, the x-axis labels on the top plot will be removed since they are redundant with the bottom plot's x-axis.

When `guides = "collect"`, legends are also merged into a single shared legend.

For cowplot: Axis alignment in cowplot requires manual use of `align_plots` with `align = "h"` or `"v"`. This function does not directly support cowplot objects.

Value

A patchwork object with collected axes.

See Also

`collect_legends`, `legend_left`, `legend_wrap`

Examples

```
library(ggplot2)

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

  # Two plots stacked vertically - x-axis is duplicated
  p1 <- ggplot(mtcars, aes(mpg, wt, color = factor(cyl))) +
    geom_point() +
    labs(y = "Weight")
  p2 <- ggplot(mtcars, aes(mpg, hp, color = factor(cyl))) +
    geom_point() +
    labs(y = "Horsepower")

  # Without collect_axes: both plots show x-axis
  p1 / p2

  # With collect_axes: removes redundant x-axis from top plot
  collect_axes(p1 / p2)

  # Keep separate legends
  collect_axes(p1 / p2, guides = "keep")
}
```

---

collect_legends *Collect Legends from Patchwork Compositions*

Description

Collects legends from multiple plots in a patchwork composition into a single legend area. This is a convenience wrapper around patchwork's guide collection functionality.

Usage

```
collect_legends(
  x,
  position = c("right", "left", "bottom", "top"),
  span = FALSE
)
```

Arguments

x	A patchwork object created by combining ggplot2 plots with operators like |, /, or +.
position	Where to place the collected legends. One of "right" (default), "left", "bottom", or "top".
span	Logical or integer vector. If TRUE, the legend will span the full height (for left/right) or width (for top/bottom). If an integer vector (e.g., c(1, 2) or 1:2), the legend spans only those rows/columns. Default is FALSE. When not FALSE, returns a gtable instead of a patchwork object.

Details

This function requires the patchwork package. If patchwork is not installed, an informative error is raised.

When span = TRUE, the function converts the patchwork to a gtable and modifies the legend cell to span all rows (for right/left positions) or all columns (for top/bottom positions). This addresses the common issue where collected legends are centered rather than spanning the full composition.

When span is an integer vector (e.g., c(1, 2)), the legend is centered between those specific rows (for right/left) or columns (for top/bottom). This is useful for attaching a legend to specific plots in a stacked layout. Row/column numbers refer to the panel positions in the patchwork (1-indexed).

For cowplot users: cowplot does not have a built-in legend collection mechanism. Consider using the lemon package for manual legend extraction and positioning.

Value

A patchwork object with legends collected, or a gtable if span is not FALSE.

See Also

collect_axes, legend_left, legend_wrap

Examples

```
library(ggplot2)

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

  p1 <- ggplot(mtcars, aes(mpg, wt, color = factor(cyl))) +
    geom_point()
```

```
  p2 <- ggplot(mtcars, aes(mpg, hp, color = factor(cyl))) +
    geom_point()

  # Without collection: each plot has its own legend
  p1 | p2

  # With collection: single shared legend
  collect_legends(p1 | p2)

  # Place collected legend on the bottom
  collect_legends(p1 | p2, position = "bottom")

  # Legend spanning full height (returns gtable, draw with grid.draw)
  gt <- collect_legends(p1 / p2, position = "right", span = TRUE)
  grid::grid.draw(gt)

  # Attach legend to specific rows (e.g., align with row 1 only)
  p3 <- ggplot(mtcars, aes(mpg, disp, color = factor(cyl))) +
    geom_point() + labs(title = "Plot 3")
  gt <- collect_legends(p1 / p2 / p3, position = "right", span = 1)
  grid::grid.draw(gt)

  # Attach legend to rows 1 and 2
  gt <- collect_legends(p1 / p2 / p3, position = "right", span = 1:2)
  grid::grid.draw(gt)
}
```

`colorbar_style` *Style Continuous Color Bar Legends*

Description

Customize the appearance of color bar legends for continuous scales. This provides a simpler interface to `guide_colourbar()`.

Usage

```
colorbar_style(
  width = NULL,
  height = NULL,
  ticks = NULL,
  ticks_length = NULL,
  frame = NULL,
  frame_linewidth = NULL,
  label = NULL,
  label_position = NULL,
  title_position = NULL,
  direction = NULL,
```

```
  reverse = NULL,
  nbin = NULL,
  aesthetic = "colour"
)

colourbar_style(
  width = NULL,
  height = NULL,
  ticks = NULL,
  ticks_length = NULL,
  frame = NULL,
  frame_linewidth = NULL,
  label = NULL,
  label_position = NULL,
  title_position = NULL,
  direction = NULL,
  reverse = NULL,
  nbin = NULL,
  aesthetic = "colour"
)
```

Arguments

width	Numeric. Width of the color bar in lines. Default is 1.
height	Numeric. Height of the color bar in lines. Default is 5.
ticks	Logical. Whether to show tick marks. Default is TRUE.
ticks_length	Numeric. Length of tick marks in lines. Default is 0.2.
frame	Logical or character. If TRUE, draws a black frame. If a color string, draws a frame in that color. If FALSE or NULL, no frame. Default is FALSE.
frame_linewidth	Numeric. Line width of the frame. Default is 0.5.
label	Logical. Whether to show labels. Default is TRUE.
label_position	Character. Position of labels: "right", "left", "top", or "bottom". Default depends on bar orientation.
title_position	Character. Position of title: "top", "bottom", "left", or "right". Default is "top".
direction	Character. Direction of the bar: "vertical" or "horizontal". Default is "vertical".
reverse	Logical. Whether to reverse the color bar. Default is FALSE.
nbin	Integer. Number of bins used to draw the color bar. Higher values give smoother gradients. Default is 300.
aesthetic	Character. Which aesthetic to apply to. Default is "colour". Common values: "colour", "color", "fill".

Details

This function simplifies common color bar customizations:

- **Size**: Use `width` and `height` to make thin/wide bars
- **Ticks**: Toggle with `ticks`, adjust length with `ticks_length`
- **Frame**: Add a border with `frame = TRUE` or `frame = "grey50"`
- **Orientation**: Use `direction = "horizontal"` for horizontal bars

The function uses the theme system internally, which is the recommended approach in ggplot2 3.5.0+.

Value

A ggplot2 guides specification.

See Also

`legend_style` for styling discrete legends, `legend_left`, `legend_bottom` for positioning.

Examples

```
library(ggplot2)

p <- ggplot(faithfuld, aes(waiting, eruptions, fill = density)) +
  geom_tile()

# Default appearance
p

# Taller, thinner bar
p + colorbar_style(width = 0.5, height = 10, aesthetic = "fill")

# Wide horizontal bar
p + colorbar_style(width = 10, height = 0.5, direction = "horizontal",
                   aesthetic = "fill")

# With frame and no ticks
p + colorbar_style(frame = "grey50", ticks = FALSE, aesthetic = "fill")

# Thin bar with frame
p + colorbar_style(width = 0.5, height = 8, frame = TRUE, aesthetic = "fill")
```

---

get_legend *Extract Legend from a ggplot*

Description

Extracts the legend (guide-box) from a ggplot object as a grob that can be used independently with grid or cowplot.

Usage

```
get_legend(plot)
```

Arguments

plot	A ggplot object.

Details

This function is useful for cowplot workflows where you want to manually position a shared legend. The extracted legend can be combined with plots using `cowplot::plot_grid()` or drawn directly with `grid::grid.draw()`.

Value

A grob containing the legend, or `NULL` if the plot has no legend.

See Also

`shared_legend`, `collect_legends`

Examples

```
library(ggplot2)

p <- ggplot(mtcars, aes(mpg, wt, color = factor(cyl))) +
  geom_point() +
  labs(color = "Cylinders")

# Extract the legend
leg <- get_legend(p)

# Draw just the legend
grid::grid.newpage()
grid::grid.draw(leg)
```

---

legend_auto_fit *Auto-fit Legend to Plot Height*

Description

Measures the legend height relative to the plot panel and automatically wraps the legend into multiple columns if it would overflow. This function must be called on a complete ggplot object, not added with `+`.

Usage

```
legend_auto_fit(plot, max_ratio = 0.95)
```

Arguments

plot	A ggplot object.
max_ratio	Maximum ratio of legend height to panel height before wrapping is triggered. Default is 0.95 (95 percent of panel height).

Details

This function builds the plot to measure actual dimensions, then rebuilds with an appropriate number of legend rows if the legend is too tall. It's particularly useful after applying `legend_style(angle = 90)` which can cause legends to exceed the plot height.

Because this requires building the plot twice, it has a small performance cost. For static plots this is negligible.

Value

A modified ggplot object with adjusted legend layout.

See Also

legend_style, legend_wrap

Examples

```
library(ggplot2)

# Legend with rotated text that might overflow
p <- ggplot(mpg, aes(displ, hwy, color = class)) +
  geom_point() +
  legend_style(angle = 90)

# Auto-fit will wrap if needed
legend_auto_fit(p)
```

---

legend_bottom *Place Legend on Bottom with Horizontal Layout*

Description

A one-liner to position the legend below the plot with horizontal layout. Optionally aligns to the full plot area rather than just the panel.

Usage

```
legend_bottom(align_to = c("panel", "plot"), by = NULL)
```

Arguments

align_to	Where to align the legend. Either "panel" (default, aligns to plot panel) or "plot" (aligns to full plot including title). Requires ggplot2 >= 3.5.0 for "plot" alignment. Ignored when by is specified.
by	Optional aesthetic name (character) to position only a specific legend. When specified, uses per-guide positioning via guide_legend(position = "bottom"). Requires ggplot2 >= 3.5.0. Common values: "colour", "fill", "size".

Value

A ggplot2 theme object (when by is NULL) or a guides specification (when by is specified).

See Also

legend_top, legend_left, legend_right, legend_horizontal

Examples

```
library(ggplot2)

# Basic usage
ggplot(mtcars, aes(mpg, wt, color = factor(cyl))) +
  geom_point() +
  legend_bottom()

# Aligned to full plot
ggplot(mtcars, aes(mpg, wt, color = factor(cyl))) +
  geom_point() +
  labs(title = "My Plot Title") +
  legend_bottom(align_to = "plot")

# Position only the colour legend at bottom
ggplot(mtcars, aes(mpg, wt, color = factor(cyl), size = hp)) +
  geom_point() +
  legend_bottom(by = "colour") +
  legend_right(by = "size")
```

---

legend_hide *Hide Specific Legends*

Description

Remove specific legends from a plot while keeping others. This is more targeted than legend_none() which removes all legends.

Usage

```
legend_hide(...)
```

Arguments

...	Aesthetic names (unquoted) to hide. Common values: `colour`, `fill`, `size`, `shape`, `linetype`, `alpha`. Note: `color` is automatically converted to `colour`.

Value

A guides specification that can be added to a plot.

See Also

`legend_select`, `legend_none`

Examples

```
library(ggplot2)

# Plot with multiple legends
p <- ggplot(mtcars, aes(mpg, wt, color = factor(cyl), size = hp)) +
  geom_point()

# Hide the size legend
p + legend_hide(size)

# Hide multiple legends
p + legend_hide(size, colour)
```

---

legend_horizontal *Set Legend Direction to Horizontal*

Description

A one-liner to arrange legend keys horizontally. Useful for legends placed at the top or bottom of a plot.

Usage

```
legend_horizontal()
```

Value

A ggplot2 theme object that can be added to a plot.

See Also

`legend_vertical`, `legend_top`, `legend_bottom`

Examples

```
library(ggplot2)

# Horizontal legend at bottom
ggplot(mtcars, aes(mpg, wt, color = factor(cyl))) +
  geom_point() +
  legend_bottom() +
  legend_horizontal()
```

`legend_inside` *Place Legend Inside the Plot Area*

Description

Position the legend inside the plot panel at specified coordinates or using convenient position shortcuts like `"topright"` or `"bottomleft"`.

Usage

```
legend_inside(
  x = NULL,
  y = NULL,
  position = NULL,
  justification = NULL,
  background = "white",
  border = NA,
  padding = 0.2,
  just = NULL
)
```

Arguments

`x`	Numeric x-coordinate in normalized 0-1 space, where 0 is left and 1 is right. Ignored if `position` is specified.
`y`	Numeric y-coordinate in normalized 0-1 space, where 0 is bottom and 1 is top. Ignored if `position` is specified.
`position`	Character shortcut for common positions. One of `"topleft"`, `"top"`, `"topright"`, `"left"`, `"center"`, `"right"`, `"bottomleft"`, `"bottom"`, `"bottomright"`. If specified, overrides `x` and `y`.
`justification`	Justification of legend relative to the anchor point. Either a character vector of length 2 (horizontal, vertical) or a single value. Common values: `c("left", "top")`, `c("right", "bottom")`, `"center"`. If `NULL`, automatically determined based on position.
`background`	Background fill color for the legend. Default is `"white"`.
`border`	Border color for the legend box. Default is `NA` (no border).

padding	Padding around legend content in cm. Default is `0.2`.
just	Deprecated. Use `justification` instead.

Value

A ggplot2 theme object that can be added to a plot.

See Also

`legend_left`, `legend_right`, `legend_top`, `legend_bottom`

Examples

```
library(ggplot2)

# Using position shortcuts (recommended)
ggplot(mtcars, aes(mpg, wt, color = factor(cyl))) +
  geom_point() +
  legend_inside(position = "topright")

ggplot(mtcars, aes(mpg, wt, color = factor(cyl))) +
  geom_point() +
  legend_inside(position = "bottomleft")

# Using coordinates
ggplot(mtcars, aes(mpg, wt, color = factor(cyl))) +
  geom_point() +
  legend_inside(x = 0.95, y = 0.95, justification = c("right", "top"))

# Custom background and border
ggplot(mtcars, aes(mpg, wt, color = factor(cyl))) +
  geom_point() +
  legend_inside(position = "topright", background = "grey95", border = "grey50")
```

---

legend_keys *Customize Legend Key Appearance*

Description

Modify the appearance of legend keys (the symbols/glyphs in the legend) without affecting the plot itself. This is a simpler alternative to using `guide_legend(override.aes = list(...))`.

Usage

```
legend_keys(
  size = NULL,
  alpha = NULL,
  shape = NULL,
```

```
  linewidth = NULL,
  linetype = NULL,
  fill = NULL,
  colour = NULL,
  color = NULL,
  stroke = NULL,
  aesthetic = c("colour", "fill")
)
```

Arguments

size	Numeric. Size of point keys.
alpha	Numeric. Alpha (transparency) of keys, between 0 and 1.
shape	Shape of point keys. Can be: • Numeric (0-25): Standard ggplot2 shape codes • Character name: "circle", "square", "diamond", "triangle", "triangle_down", "plus", "cross", "asterisk", "circle_open", "square_open", "diamond_open", "triangle_open", "circle_filled", "square_filled", "diamond_filled", "triangle_filled" Shapes 21-25 (or names ending in "_filled") support both outline (colour) and fill (fill) colors.
linewidth	Numeric. Width of line keys.
linetype	Character or numeric. Line type for line keys.
fill	Character. Fill color for filled shapes (shapes 21-25). For shapes 0-20, use colour instead.
colour, color	Character. Outline/stroke color for keys. For shapes 21-25, this controls the outline; for shapes 0-20, this is the main color.
stroke	Numeric. Stroke width for point outlines (shapes 21-25).
aesthetic	Character vector specifying which aesthetic(s) to modify. Default is c("colour", "fill") which covers most common cases. Use "all" to apply to all legends.

Details

This function wraps guide_legend(override.aes = ...) to provide a cleaner interface for common legend key modifications. It's particularly useful for:

- Making small points more visible in the legend
- Removing transparency from legend keys
- Changing symbol shapes to improve clarity
- Adding outlines to filled shapes for better visibility

Shape types:

- Shapes 0-14: Outline only (color from colour)
- Shapes 15-20: Filled solid (color from colour)

- Shapes 21-25: Outline + fill (outline from `colour`, interior from `fill`)

Important note for filled shapes (21-25):

When using filled shapes with both outline and fill colors, the behavior depends on which aesthetics are mapped in your original plot:

- **White fill, colored outline**: Works with `aes(color = var)`. Use `legend_keys(shape = "circle_filled", fill = "white")`.
- **Colored fill, black outline**: Requires `aes(color = var, fill = var)` in your plot. Then use `legend_keys(colour = "black")`.

This is because `override.aes` can only set static values; it cannot inherit from mapped aesthetics. If you only map `color` and try to override the outline to black, the fill will not have a color mapping to use.

Only non-NULL arguments are applied, so you can selectively modify specific properties.

Value

A list of ggplot2 guide specifications.

See Also

`legend_style` for styling legend text and background, `legend_order` for reordering legend entries.

Examples

```
library(ggplot2)

# Points get lost in legend - make them bigger
ggplot(mtcars, aes(mpg, wt, color = factor(cyl))) +
  geom_point(size = 1) +
  legend_keys(size = 4)

# Remove transparency from legend
ggplot(mtcars, aes(mpg, wt, color = factor(cyl))) +
  geom_point(alpha = 0.3, size = 3) +
  legend_keys(alpha = 1)

# Change shape using name
ggplot(mtcars, aes(mpg, wt, color = factor(cyl))) +
  geom_point(size = 3) +
  legend_keys(shape = "square")

# Filled shape with white fill and colored outline (shapes 21-25)
# Works because we set fill to a static color (white)
ggplot(mtcars, aes(mpg, wt, color = factor(cyl))) +
  geom_point(size = 3) +
  legend_keys(shape = "circle_filled", fill = "white", stroke = 1.5)

# Colored fill with black outline - MUST map both color AND fill in the plot
# This is a ggplot2 limitation: override.aes can only set static values,
```

```
# it cannot make fill "inherit" from color
ggplot(mtcars, aes(mpg, wt, color = factor(cyl), fill = factor(cyl))) +
  geom_point(size = 3, shape = 21, stroke = 1) +
  legend_keys(colour = "black", stroke = 1)

# Apply to fill aesthetic (e.g., for boxplots)
ggplot(mtcars, aes(factor(cyl), mpg, fill = factor(cyl))) +
  geom_boxplot(alpha = 0.5) +
  legend_keys(alpha = 1, aesthetic = "fill")
```

`legend_left` *Place Legend on the Left with Proper Alignment*

Description

A one-liner to position the legend on the left side of the plot with correct left alignment for both the key box and text labels. This goes beyond simple `legend.position = "left"` by also setting justification and box alignment.

Usage

```
legend_left(by = NULL)
```

Arguments

by	Optional aesthetic name (character) to position only a specific legend. When specified, uses per-guide positioning via `guide_legend(position = "left")`. Requires ggplot2 >= 3.5.0. Common values: `"colour"`, `"fill"`, `"size"`.

Details

When by is NULL (default), this function sets three theme elements:

- `legend.position = "left"` to place the legend on the left
- `legend.justification = "left"` to left-justify the legend box
- `legend.box.just = "left"` to left-align multiple legend boxes

When by is specified, only the legend for that aesthetic is moved, allowing different legends to be placed in different positions.

Value

A ggplot2 theme object (when by is NULL) or a guides specification (when by is specified).

See Also

`legend_right`, `legend_top`, `legend_bottom`, `legend_inside`, `legend_none`

Examples

```
library(ggplot2)

# Basic usage
ggplot(mtcars, aes(mpg, wt, color = factor(cyl))) +
  geom_point() +
  legend_left()

# Works with multiple legends
ggplot(mtcars, aes(mpg, wt, color = factor(cyl), shape = factor(am))) +
  geom_point(size = 3) +
  legend_left()

# Position only the colour legend on the left
ggplot(mtcars, aes(mpg, wt, color = factor(cyl), size = hp)) +
  geom_point() +
  legend_left(by = "colour") +
  legend_bottom(by = "size")
```

---

legend_merge *Force Legends to Merge*

Description

Force specified legends to merge together by setting them to the same order. Legends will only merge if they have matching labels (same factor levels or break values).

Usage

```
legend_merge(...)
```

Arguments

`...`	Aesthetic names (unquoted) to merge. E.g., `colour`, `fill`.

Details

ggplot2 automatically merges legends when they have the same title and matching labels. This function ensures legends have the same order value (order = 0), which is a prerequisite for merging.

If legends still don't merge after using this function, ensure:

- Both aesthetics map to the same variable
- The legends have identical titles (use `labs()`)
- The breaks/labels are identical

Value

A guides specification that can be added to a plot.

See Also

legend_split, legend_order_guides

Examples

```
library(ggplot2)

# Plot where colour and fill map to the same variable
p <- ggplot(mtcars, aes(mpg, wt, color = factor(cyl), fill = factor(cyl))) +
  geom_point(shape = 21, size = 3, stroke = 1.5) +
  labs(color = "Cylinders", fill = "Cylinders")

# Force merge (they should merge automatically if labels match)
p + legend_merge(colour, fill)
```

---

legend_none *Remove Legend from Plot*

Description

A one-liner to remove the legend from a plot. Cleaner alternative to `theme(legend.position = "none")`.

Usage

```
legend_none()
```

Value

A ggplot2 theme object that can be added to a plot.

See Also

legend_left, legend_right

Examples

```
library(ggplot2)

# Remove legend
ggplot(mtcars, aes(mpg, wt, color = factor(cyl))) +
  geom_point() +
  legend_none()
```

`legend_order` *Reorder Legend Entries*

Description

Change the order of legend entries without modifying factor levels in your data. This provides a simpler alternative to manually setting factor levels or using the `breaks` argument in scale functions.

Usage

```
legend_order(order, aesthetic = "colour")
```

Arguments

`order`	A character vector specifying the desired order of legend entries, or a function to apply to the current order (e.g., `rev`, `sort`).
`aesthetic`	Character string specifying which aesthetic's legend to reorder. Default is `"colour"`. Common values: `"colour"`, `"color"`, `"fill"`, `"shape"`, `"linetype"`, `"size"`, `"alpha"`.

Details

This function works by setting the `breaks` argument of the appropriate discrete scale. It automatically detects whether to use `scale_colour_discrete`, `scale_fill_discrete`, etc. based on the `aesthetic` argument.

When `order` is a function (like `rev` or `sort`), it will be applied to the default order of legend entries.

Value

A ggplot2 scale object that reorders the legend.

See Also

`legend_reverse` for a simpler way to reverse legend order, `legend_style` for styling legend appearance.

Examples

```
library(ggplot2)

p <- ggplot(mtcars, aes(mpg, wt, color = factor(cyl))) +
  geom_point(size = 3)

# Specify exact order
p + legend_order(c("8", "6", "4"))

# Reverse the order
p + legend_order(rev)
```

```
# Sort alphabetically/numerically
p + legend_order(sort)

# Reorder fill aesthetic
ggplot(mtcars, aes(factor(cyl), fill = factor(cyl))) +
  geom_bar() +
  legend_order(c("8", "4", "6"), aesthetic = "fill")
```

`legend_order_guides` *Control Legend Display Order*

Description

Set the display order of multiple legends. Legends with lower order values appear first (top or left).

Usage

```
legend_order_guides(...)
```

Arguments

`...`	Named arguments where names are aesthetic names and values are integer order positions. E.g., `colour = 1, size = 2`.

Details

The order value determines the position of the legend relative to others. Lower values appear first. By default, all legends have order = 0 and appear in an unspecified order.

Value

A guides specification that can be added to a plot.

See Also

`legend_merge`, `legend_split`

Examples

```
library(ggplot2)

# Plot with multiple legends
p <- ggplot(mtcars, aes(mpg, wt, color = factor(cyl), size = hp)) +
  geom_point()

# Default order
p
```

```
# Size legend first, then colour
p + legend_order_guides(size = 1, colour = 2)

# Colour legend first
p + legend_order_guides(colour = 1, size = 2)
```

`legend_reverse` *Reverse Legend Order*

Description

Reverses the order of entries in all legends. Useful when the natural data order doesn't match the desired visual order (e.g., when stacking bars).

Usage

```
legend_reverse()
```

Details

This function applies `guide_legend(reverse = TRUE)` to all common discrete aesthetics: colour, fill, shape, size, linetype, and alpha.

Value

A guides specification that can be added to a plot.

See Also

`legend_wrap`, `legend_style`

Examples

```
library(ggplot2)

# Default order
p1 <- ggplot(mtcars, aes(mpg, wt, color = factor(cyl))) +
  geom_point()

# Reversed order
p2 <- p1 + legend_reverse()
```

`legend_right` *Place Legend on the Right with Proper Alignment*

Description

A one-liner to position the legend on the right side of the plot with correct right alignment for both the key box and text labels.

Usage

```
legend_right(by = NULL)
```

Arguments

by	Optional aesthetic name (character) to position only a specific legend. When specified, uses per-guide positioning via `guide_legend(position = "right")`. Requires ggplot2 >= 3.5.0. Common values: `"colour"`, `"fill"`, `"size"`.

Details

When by is NULL (default), this function sets three theme elements:

- `legend.position = "right"` to place the legend on the right
- `legend.justification = "right"` to right-justify the legend box
- `legend.box.just = "right"` to right-align multiple legend boxes

When by is specified, only the legend for that aesthetic is moved, allowing different legends to be placed in different positions.

Value

A ggplot2 theme object (when by is NULL) or a guides specification (when by is specified).

See Also

`legend_left`, `legend_top`, `legend_bottom`, `legend_inside`

Examples

```
library(ggplot2)

# Basic usage
ggplot(mtcars, aes(mpg, wt, color = factor(cyl))) +
  geom_point() +
  legend_right()

# Position only the size legend on the right
ggplot(mtcars, aes(mpg, wt, color = factor(cyl), size = hp)) +
  geom_point() +
```

```
  legend_bottom(by = "colour") +
  legend_right(by = "size")
```

`legend_select` *Keep Only Specific Legends*

Description

Show only the specified legends and hide all others. This is the inverse of `legend_hide`.

Usage

```
legend_select(...)
```

Arguments

`...`	Aesthetic names (unquoted) to keep. All other legends will be hidden. Common values: `colour`, `fill`, `size`, `shape`, `linetype`, `alpha`.

Value

A guides specification that can be added to a plot, or `NULL` if nothing needs to be hidden.

See Also

`legend_hide`, `legend_none`

Examples

```
library(ggplot2)

# Plot with multiple legends
p <- ggplot(mtcars, aes(mpg, wt, color = factor(cyl), size = hp,
                        shape = factor(am))) +
  geom_point()

# Keep only the colour legend
p + legend_select(colour)

# Keep colour and shape, hide size
p + legend_select(colour, shape)
```

`legend_split` *Force Legends to Stay Separate*

Description

Force specified legends to remain separate by assigning different order values, preventing automatic merging.

Usage

```
legend_split(...)
```

Arguments

`...`	Aesthetic names (unquoted) to keep separate. E.g., `colour, fill`.

Details

By default, ggplot2 merges legends that have matching titles and labels. This function assigns different order values to each legend, which prevents automatic merging.

Value

A guides specification that can be added to a plot.

See Also

`legend_merge`, `legend_order_guides`

Examples

```
library(ggplot2)

# Plot where colour and fill would normally merge
p <- ggplot(mtcars, aes(mpg, wt, color = factor(cyl), fill = factor(cyl))) +
  geom_point(shape = 21, size = 3, stroke = 1.5) +
  labs(color = "Cylinders", fill = "Cylinders")

# Force separate legends
p + legend_split(colour, fill)
```

`legend_style` *Style Legend Appearance*

Description

A comprehensive one-liner to style all legend elements consistently. Instead of setting multiple theme elements separately, use this function to control text, title, keys, spacing, background, and direction in one call.

Usage

```
legend_style(
  size = NULL,
  family = NULL,
  face = NULL,
  color = NULL,
  angle = NULL,
  title_size = NULL,
  title_face = NULL,
  title_color = NULL,
  title_angle = NULL,
  title_hjust = NULL,
  title_vjust = NULL,
  title_position = NULL,
  key_width = NULL,
  key_height = NULL,
  key_fill = NULL,
  spacing = NULL,
  spacing_x = NULL,
  spacing_y = NULL,
  margin = NULL,
  background = NULL,
  background_color = NULL,
  box_background = NULL,
  box_margin = NULL,
  direction = NULL,
  byrow = NULL,
  justification = NULL,
  by = NULL
)
```

Arguments

`size`	Text size for legend labels (in points).
`family`	Font family for legend text.
`face`	Font face for legend text (`"plain"`, `"bold"`, `"italic"`, `"bold.italic"`).

color	Text color for legend labels.
angle	Rotation angle for legend labels (in degrees). Supported values: 45, -45, 90, -90. Text justification is set automatically for optimal alignment with legend keys.
title_size	Text size for legend title (in points). If NULL, inherits from size.
title_face	Font face for legend title. If NULL, inherits from face.
title_color	Text color for legend title. If NULL, inherits from color.
title_angle	Rotation angle for legend title (in degrees).
title_hjust	Horizontal justification for rotated title.
title_vjust	Vertical justification for rotated title.
title_position	Position of legend title relative to keys. One of "top", "bottom", "left", "right".
key_width	Width of legend keys. Numeric (in cm) or a unit object.
key_height	Height of legend keys. Numeric (in cm) or a unit object.
key_fill	Background fill color for legend keys.
spacing	Spacing between legend entries. Numeric (in cm) or a unit object.
spacing_x	Horizontal spacing between legend entries.
spacing_y	Vertical spacing between legend entries.
margin	Margin around entire legend. Single value (all sides) or vector of 4 values (top, right, bottom, left) in cm.
background	Legend background fill color. Use NA for transparent.
background_color	Legend background border color. Use NA for no border.
box_background	Background fill for the box containing multiple legends. Ignored when by is specified.
box_margin	Margin around the legend box. Single value or 4-vector in cm. Ignored when by is specified.
direction	Legend direction: "horizontal" or "vertical".
byrow	For multi-column legends, fill by row (TRUE) or by column (FALSE).
justification	Justification of the legend along its side. For legends on the top or bottom: "left", "center", "right", or a numeric value in [0, 1]. For legends on the left or right: "top", "center", "bottom", or a numeric value in [0, 1]. When by is specified, applies per-guide via guide_legend(theme = theme(legend.justification = ...)). When by is NULL, sets legend.justification for the whole plot. Requires ggplot2 >= 3.5.0 for per-guide use.
by	Optional aesthetic name (character) to style only a specific legend. When specified, uses per-guide theming via guide_legend(theme = ...). Requires ggplot2 >= 3.5.0. Common values: "colour", "fill", "size".

Value

A ggplot2 theme object (when by is NULL) or a guides specification (when by is specified).

See Also

legend_left, legend_wrap, legend_reverse

Examples

```
library(ggplot2)

# Simple: consistent font
ggplot(mtcars, aes(mpg, wt, color = factor(cyl))) +
  geom_point() +
  legend_style(size = 12, family = "serif")

# Styled title and keys
ggplot(mtcars, aes(mpg, wt, color = factor(cyl))) +
  geom_point() +
  legend_style(
    size = 10,
    title_size = 14,
    title_face = "bold",
    key_width = 1.5
  )

# Full styling with background
ggplot(mtcars, aes(mpg, wt, color = factor(cyl))) +
  geom_point() +
  legend_style(
    size = 11,
    title_size = 13,
    title_face = "bold",
    key_fill = "grey95",
    background = "white",
    background_color = "grey80",
    margin = 0.3
  )

# Rotated labels for long category names
ggplot(mpg, aes(displ, hwy, color = class)) +
  geom_point() +
  legend_style(angle = 45)

# Style only the colour legend
ggplot(mtcars, aes(mpg, wt, color = factor(cyl), size = hp)) +
  geom_point() +
  legend_style(title_face = "bold", background = "grey95", by = "colour") +
  legend_style(size = 10, by = "size")

# Per-legend justification: slide each legend along its side
ggplot(mtcars, aes(mpg, wt, color = factor(cyl), size = hp)) +
  geom_point() +
  legend_top(by = "colour") +
  legend_right(by = "size") +
  legend_style(by = "colour", justification = "left") +
```

```
legend_style(by = "size",   justification = "top")
```

`legend_top` *Place Legend on Top with Horizontal Layout*

Description

A one-liner to position the legend above the plot with horizontal layout. Optionally aligns to the full plot area (including title) rather than just the panel.

Usage

```
legend_top(align_to = c("panel", "plot"), by = NULL)
```

Arguments

`align_to`	Where to align the legend. Either `"panel"` (default, aligns to plot panel) or `"plot"` (aligns to full plot including title). Requires ggplot2 >= 3.5.0 for `"plot"` alignment. Ignored when by is specified.
`by`	Optional aesthetic name (character) to position only a specific legend. When specified, uses per-guide positioning via `guide_legend(position = "top")`. Requires ggplot2 >= 3.5.0. Common values: `"colour"`, `"fill"`, `"size"`.

Value

A ggplot2 theme object (when by is NULL) or a guides specification (when by is specified).

See Also

`legend_bottom`, `legend_left`, `legend_right`, `legend_horizontal`

Examples

```
library(ggplot2)

# Basic usage - aligned to panel
ggplot(mtcars, aes(mpg, wt, color = factor(cyl))) +
  geom_point() +
  legend_top()

# Aligned to full plot (useful with titles)
ggplot(mtcars, aes(mpg, wt, color = factor(cyl))) +
  geom_point() +
  labs(title = "My Plot Title") +
  legend_top(align_to = "plot")

# Position only the colour legend on top
ggplot(mtcars, aes(mpg, wt, color = factor(cyl), size = hp)) +
```

```
  geom_point() +
  legend_top(by = "colour") +
  legend_right(by = "size")
```

---

legend_vertical *Set Legend Direction to Vertical*

Description

A one-liner to arrange legend keys vertically. This is the default for legends placed on the left or right of a plot.

Usage

```
legend_vertical()
```

Value

A ggplot2 theme object that can be added to a plot.

See Also

legend_horizontal, legend_left, legend_right

Examples

```
library(ggplot2)

# Explicitly set vertical direction
ggplot(mtcars, aes(mpg, wt, color = factor(cyl))) +
  geom_point() +
  legend_right() +
  legend_vertical()
```

---

legend_wrap *Wrap Legend Entries into Columns or Rows*

Description

A one-liner to arrange legend entries in a grid layout. Works with discrete legends by applying the specified layout to all color, fill, shape, size, linetype, and alpha aesthetics.

Usage

```
legend_wrap(ncol = NULL, nrow = NULL, byrow = TRUE)
```

Arguments

ncol	Number of columns for the legend layout. If NULL, determined automatically based on nrow.
nrow	Number of rows for the legend layout. If NULL, determined automatically based on ncol.
byrow	Logical. If TRUE (default), fills by row first. If FALSE, fills by column first.

Details

This function creates a guides() specification that applies the same column/row layout to all common discrete aesthetics. At least one of ncol or nrow should be specified.

Value

A list of guide specifications that can be added to a plot.

See Also

legend_left, collect_legends

Examples

```
library(ggplot2)

# Wrap a long legend into 2 columns
ggplot(mpg, aes(displ, hwy, color = class)) +
  geom_point() +
  legend_wrap(ncol = 2)

# Wrap into 3 rows, filling by column
ggplot(mpg, aes(displ, hwy, color = class)) +
  geom_point() +
  legend_wrap(nrow = 3, byrow = FALSE)

# Combine with legend_left for left-aligned wrapped legends
ggplot(mpg, aes(displ, hwy, color = class)) +
  geom_point() +
  legend_wrap(ncol = 2) +
  legend_left()
```

---

shared_legend *Combine Plots with a Shared Legend*

Description

Combines multiple ggplot objects into a grid layout with a single shared legend. Works with base ggplot and cowplot workflows (no patchwork required).

Usage

```
shared_legend(
  ...,
  ncol = NULL,
  nrow = NULL,
  position = c("right", "left", "bottom", "top"),
  legend_from = 1,
  rel_legend_size = 0.2
)
```

Arguments

`...`	ggplot objects to combine.
`ncol`	Number of columns in the plot grid.
`nrow`	Number of rows in the plot grid. If NULL (default), computed from ncol and number of plots.
`position`	Where to place the shared legend. One of `"right"` (default), `"left"`, `"bottom"`, or `"top"`.
`legend_from`	Which plot to extract the legend from. Default is 1 (first plot). Can be an integer index or a ggplot object.
`rel_legend_size`	Relative size of the legend compared to the plot area. Default is 0.2 (20 percent).

Details

This function provides a simple way to create multi-panel plots with a shared legend without requiring patchwork. It:

1. Removes legends from all plots
2. Extracts the legend from the specified plot
3. Arranges plots in a grid
4. Attaches the legend to the specified position

For more complex layouts or patchwork users, see `collect_legends`.

Value

A gtable that can be drawn with `grid::grid.draw()` or used with `cowplot::ggdraw()`.

See Also

`get_legend`, `collect_legends`

Examples

```
library(ggplot2)

p1 <- ggplot(mtcars, aes(mpg, wt, color = factor(cyl))) +
  geom_point() + labs(title = "Plot 1", color = "Cylinders")
p2 <- ggplot(mtcars, aes(mpg, hp, color = factor(cyl))) +
  geom_point() + labs(title = "Plot 2", color = "Cylinders")
p3 <- ggplot(mtcars, aes(mpg, disp, color = factor(cyl))) +
  geom_point() + labs(title = "Plot 3", color = "Cylinders")

# Side-by-side with shared legend on right
gt <- shared_legend(p1, p2, ncol = 2, position = "right")
grid::grid.newpage()
grid::grid.draw(gt)

# 2x2 grid with legend at bottom
gt <- shared_legend(p1, p2, p3, p1, ncol = 2, nrow = 2, position = "bottom")
grid::grid.newpage()
grid::grid.draw(gt)

# Stacked with legend on left
gt <- shared_legend(p1, p2, p3, ncol = 1, position = "left")
grid::grid.newpage()
grid::grid.draw(gt)
```

Index