

# Package ‘ggpattern’

August 27, 2025

**Type** Package

**Title** 'ggplot2' Pattern Geoms

**Version** 1.2.1

## Description

Provides 'ggplot2' geoms filled with various patterns. Includes a patterned version of every 'ggplot2' geom that has a region that can be filled with a pattern. Provides a suite of 'ggplot2' aesthetics and scales for controlling pattern appearances. Supports over a dozen builtin patterns (every pattern implemented by 'gridpattern') as well as allowing custom user-defined patterns.

**URL** <https://github.com/trevorld/ggpattern>,  
<https://trevorldavis.com/R/ggpattern/>

**BugReports** <https://github.com/trevorld/ggpattern/issues>

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**Imports** cli, ggplot2 (>= 3.5.1), glue, grid, gridpattern (>= 1.2.2),  
lifecycle, rlang (>= 1.1.3), scales, vctrs

**Suggests** ambient, dplyr, gganimate, knitr (>= 1.47), magick, mapproj,  
maps, png, ragg (>= 1.2.0), readr, rmarkdown (>= 2.27), sf (>= 0.7-3), svglite (>= 2.1.3), testthat (>= 2.1.0), vdiff (>= 1.0.6)

**VignetteBuilder** knitr, ragg, rmarkdown

**Collate** 'a-geom-docs.R' 'aaa-ggplot2-compat-plyr.R'  
'aaa-ggplot2-ggplot-global.R' 'aaa-ggplot2-performance.R'  
'aaa-ggplot2-scale-manual.R' 'aaa-ggplot2-utilities-grid.R'  
'aaa-ggplot2-utilities.R' 'geom-.R' 'geom-rect.R' 'geom-bar.R'  
'geom-bin2d.R' 'geom-boxplot.R' 'geom-col.R' 'geom-crossbar.R'  
'geom-ribbon.R' 'geom-density.R' 'geom-histogram.R'  
'geom-polygon.R' 'geom-map.R' 'geom-sf.R' 'geom-tile.R'  
'geom-violin.R' 'ggpattern-defunct.R' 'ggpattern-deprecated.R'  
'ggpattern-package.R' 'pattern.R' 'polygon\_df.R'  
'scale-pattern-alpha.R' 'scale-pattern-auto.R'

'scale-pattern-brewer.R' 'scale-pattern-colour.R'  
 'scale-pattern-gradient.R' 'scale-pattern-grey.R'  
 'scale-pattern-hue.R' 'scale-pattern-linetype.R'  
 'scale-pattern-shape.R' 'scale-pattern-size.R'  
 'scale-pattern-iridis.R' 'scale-pattern.R' 'zxx.r' 'zzz.R'

**NeedsCompilation** no

**Author** Mike FC [aut],  
 Trevor L. Davis [aut, cre] (ORCID:  
 <<https://orcid.org/0000-0001-6341-4639>>),  
 ggplot2 authors [aut]

**Maintainer** Trevor L. Davis <trevor.l.davis@gmail.com>

**Repository** CRAN

**Date/Publication** 2025-08-27 05:10:08 UTC

## Contents

create_polygon_df . . . . .	3
draw_key_polygon_pattern . . . . .	3
geom-docs . . . . .	5
GeomRectPattern . . . . .	14
ggpattern-defunct . . . . .	15
is_polygon_df . . . . .	16
scale_continuous . . . . .	16
scale_discrete . . . . .	21
scale_pattern_alpha_continuous . . . . .	25
scale_pattern_colour_brewer . . . . .	26
scale_pattern_colour_continuous . . . . .	29
scale_pattern_colour_gradient . . . . .	30
scale_pattern_colour_grey . . . . .	33
scale_pattern_colour_hue . . . . .	34
scale_pattern_colour_viridis_d . . . . .	36
scale_pattern_identity . . . . .	38
scale_pattern_linetype . . . . .	40
scale_pattern_manual . . . . .	41
scale_pattern_shape . . . . .	43
scale_pattern_size_continuous . . . . .	45

**Index** 47

---

create_polygon_df	<i>Create a polygon_df object from the given coordinates</i>
-------------------	--

---

### Description

code using polygon\_df should not assume that the first and last point within each id are the same. i.e. they may have to manually set a final point equal to the initial point if that is what their graphics system desires

### Usage

```
create_polygon_df(x, y, id = 1L)
```

### Arguments

x, y	coordinates of polygon. not necessarily closed.
id	a numeric vector used to separate locations in x,y into multiple polygons

### Value

data.frame with x, y, id columns.

### Examples

```
df <- create_polygon_df(x = c(0, 0, 1, 1), y = c(0, 1, 1, 0))  
is_polygon_df(df)
```

---

draw_key_polygon_pattern	<i>Key glyphs for legends</i>
--------------------------	-------------------------------

---

### Description

Each geom has an associated function that draws the key when the geom needs to be displayed in a legend. These functions are called draw\_key\_\*( ), where \* stands for the name of the respective key glyph. The key glyphs can be customized for individual geoms by providing a geom with the key\_glyph argument (see [ggplot2::layer\(\)](#) or examples below.)

**Usage**

```
draw_key_polygon_pattern(data, params, size, aspect_ratio = get_aspect_ratio())

draw_key_boxplot_pattern(data, params, size, aspect_ratio = get_aspect_ratio())

draw_key_crossbar_pattern(
  data,
  params,
  size,
  aspect_ratio = get_aspect_ratio()
)
```

**Arguments**

<code>data</code>	A single row data frame containing the scaled aesthetics to display in this key
<code>params</code>	A list of additional parameters supplied to the geom.
<code>size</code>	Width and height of key in mm.
<code>aspect_ratio</code>	the geom's best guess at what the aspect ratio might be.

**Value**

A grid grob.

**Examples**

```
if (require("ggplot2")) {
  # 'stripe' pattern example
  df <- data.frame(level = c("a", "b", "c", "d"), outcome = c(2.3, 1.9, 3.2, 1))
  gg <- ggplot(df) +
    geom_col_pattern(
      aes(level, outcome, pattern_fill = level),
      pattern = 'stripe',
      fill = 'white',
      colour = 'black',
      key_glyph = draw_key_polygon_pattern
    ) +
    theme_bw(18) +
    theme(legend.position = 'none') +
    labs(
      title = "ggpattern::geom_col_pattern()",
      subtitle = "pattern = 'stripe'"
    )
  plot(gg)
}
```

**Description**

All geoms in this package are identical to their counterparts in `ggplot2` except that they can be filled with patterns.

**Usage**

```
geom_rect_pattern(  
  mapping = NULL,  
  data = NULL,  
  stat = "identity",  
  position = "identity",  
  ...,  
  linejoin = "mitre",  
  na.rm = FALSE,  
  show.legend = NA,  
  inherit.aes = TRUE  
)
```

```
geom_bar_pattern(  
  mapping = NULL,  
  data = NULL,  
  stat = "count",  
  position = "stack",  
  ...,  
  just = 0.5,  
  width = NULL,  
  na.rm = FALSE,  
  orientation = NA,  
  show.legend = NA,  
  inherit.aes = TRUE  
)
```

```
geom_bin_2d_pattern(  
  mapping = NULL,  
  data = NULL,  
  stat = "bin2d",  
  position = "identity",  
  ...,  
  na.rm = FALSE,  
  show.legend = NA,  
  inherit.aes = TRUE  
)
```

```
geom_bin2d_pattern(  
  mapping = NULL,  
  data = NULL,  
  stat = "bin2d",  
  position = "identity",  
  ...,  
  na.rm = FALSE,  
  show.legend = NA,  
  inherit.aes = TRUE  
)
```

```
geom_boxplot_pattern(  
  mapping = NULL,  
  data = NULL,  
  stat = "boxplot",  
  position = "dodge2",  
  ...,  
  outliers = TRUE,  
  outlier.colour = NULL,  
  outlier.color = NULL,  
  outlier.fill = NULL,  
  outlier.shape = 19,  
  outlier.size = 1.5,  
  outlier.stroke = 0.5,  
  outlier.alpha = NULL,  
  notch = FALSE,  
  notchwidth = 0.5,  
  staplewidth = 0,  
  varwidth = FALSE,  
  na.rm = FALSE,  
  orientation = NA,  
  show.legend = NA,  
  inherit.aes = TRUE  
)
```

```
geom_col_pattern(  
  mapping = NULL,  
  data = NULL,  
  position = "stack",  
  ...,  
  just = 0.5,  
  width = NULL,  
  na.rm = FALSE,  
  show.legend = NA,  
  inherit.aes = TRUE  
)
```

```
geom_crossbar_pattern(  
  mapping = NULL,  
  data = NULL,  
  position = "dodge",  
  ...,  
  na.rm = FALSE,  
  show.legend = NA,  
  inherit.aes = TRUE  
)
```

```
mapping = NULL,  
data = NULL,  
stat = "identity",  
position = "identity",  
...,  
fatten = 2.5,  
na.rm = FALSE,  
orientation = NA,  
show.legend = NA,  
inherit.aes = TRUE  
)
```

```
geom_ribbon_pattern(  
  mapping = NULL,  
  data = NULL,  
  stat = "identity",  
  position = "identity",  
  ...,  
  na.rm = FALSE,  
  orientation = NA,  
  show.legend = NA,  
  inherit.aes = TRUE,  
  outline.type = "both"  
)
```

```
geom_area_pattern(  
  mapping = NULL,  
  data = NULL,  
  stat = "align",  
  position = "stack",  
  na.rm = FALSE,  
  orientation = NA,  
  show.legend = NA,  
  inherit.aes = TRUE,  
  ...,  
  outline.type = "upper"  
)
```

```
geom_density_pattern(  
  mapping = NULL,  
  data = NULL,  
  stat = "density",  
  position = "identity",  
  ...,  
  na.rm = FALSE,  
  orientation = NA,  
  show.legend = NA,  
  inherit.aes = TRUE,
```

```
  outline.type = "upper"  
)
```

```
geom_histogram_pattern(  
  mapping = NULL,  
  data = NULL,  
  stat = "bin",  
  position = "stack",  
  ...,  
  binwidth = NULL,  
  bins = NULL,  
  na.rm = FALSE,  
  orientation = NA,  
  show.legend = NA,  
  inherit.aes = TRUE  
)
```

```
geom_polygon_pattern(  
  mapping = NULL,  
  data = NULL,  
  stat = "identity",  
  position = "identity",  
  rule = "evenodd",  
  ...,  
  na.rm = FALSE,  
  show.legend = NA,  
  inherit.aes = TRUE  
)
```

```
geom_map_pattern(  
  mapping = NULL,  
  data = NULL,  
  stat = "identity",  
  ...,  
  map,  
  na.rm = FALSE,  
  show.legend = NA,  
  inherit.aes = TRUE  
)
```

```
geom_sf_pattern(  
  mapping = aes(),  
  data = NULL,  
  stat = "sf",  
  position = "identity",  
  na.rm = FALSE,  
  show.legend = NA,  
  inherit.aes = TRUE,
```



```

    ...
  )

geom_tile_pattern(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  linejoin = "mitre",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_violin_pattern(
  mapping = NULL,
  data = NULL,
  stat = "ydensity",
  position = "dodge",
  ...,
  draw_quantiles = NULL,
  trim = TRUE,
  bounds = c(-Inf, Inf),
  scale = "area",
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE
)

```

## Arguments

mapping	Set of aesthetic mappings created by <a href="#">aes()</a> . If specified and <code>inherit.aes = TRUE</code> (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	<p>The data to be displayed in this layer. There are three options:</p> <p>If <code>NULL</code>, the default, the data is inherited from the plot data as specified in the call to <a href="#">ggplot()</a>.</p> <p>A <code>data.frame</code>, or other object, will override the plot data. All objects will be fortified to produce a data frame. See <a href="#">fortify()</a> for which variables will be created.</p> <p>A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code>, and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).</p>
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 the over-

	ride the default coupling between geoms and stats. The <code>stat</code> argument accepts the following:
	<ul style="list-style-type: none"> <li>• A Stat ggproto subclass, for example <code>StatCount</code>.</li> <li>• 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 "count".</li> <li>• For more information and other ways to specify the stat, see the <a href="#">layer stat</a> documentation.</li> </ul>
<code>position</code>	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: <ul style="list-style-type: none"> <li>• The result of calling a position function, such as <code>position_jitter()</code>. This method allows for passing extra arguments to the position.</li> <li>• 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 "jitter".</li> <li>• For more information and other ways to specify the position, see the <a href="#">layer position</a> documentation.</li> </ul>
<code>...</code>	Other arguments passed on to <code>layer()</code> 's <code>params</code> argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the <code>position</code> argument, or aesthetics that are required can <i>not</i> be passed through <code>...</code> . Unknown arguments that are not part of the 4 categories below are ignored. <ul style="list-style-type: none"> <li>• Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, <code>colour = "red"</code> or <code>linewidth = 3</code>. The geom's documentation has an <b>Aesthetics</b> section that lists the available options. The 'required' aesthetics cannot be passed on to the <code>params</code>. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.</li> <li>• When constructing a layer using a <code>stat_*()</code> function, the <code>...</code> argument can be used to pass on parameters to the geom part of the layer. An example of this is <code>stat_density(geom = "area", outline.type = "both")</code>. The geom's documentation lists which parameters it can accept.</li> <li>• Inversely, when constructing a layer using a <code>geom_*()</code> function, the <code>...</code> argument can be used to pass on parameters to the stat part of the layer. An example of this is <code>geom_area(stat = "density", adjust = 0.5)</code>. The stat's documentation lists which parameters it can accept.</li> <li>• The <code>key_glyph</code> argument of <code>layer()</code> may also be passed on through <code>...</code>. This can be one of the functions described as <a href="#">key glyphs</a>, to change the display of the layer in the legend.</li> </ul>
<code>linejoin</code>	Line join style (round, mitre, bevel).
<code>na.rm</code>	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
<code>show.legend</code>	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.

<code>inherit.aes</code>	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. <code>borders()</code> .
<code>just</code>	Adjustment for column placement. Set to 0.5 by default, meaning that columns will be centered about axis breaks. Set to 0 or 1 to place columns to the left/right of axis breaks. Note that this argument may have unintended behaviour when used with alternative positions, e.g. <code>position_dodge()</code> .
<code>width</code>	Bar width. By default, set to 90% of the <code>resolution()</code> of the data.
<code>orientation</code>	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 <code>orientation</code> to either "x" or "y". See the <i>Orientation</i> section for more detail.
<code>outliers</code>	Whether to display (TRUE) or discard (FALSE) outliers from the plot. Hiding or discarding outliers can be useful when, for example, raw data points need to be displayed on top of the boxplot. By discarding outliers, the axis limits will adapt to the box and whiskers only, not the full data range. If outliers need to be hidden and the axes needs to show the full data range, please use <code>outlier.shape = NA</code> instead.
<code>outlier.colour</code> , <code>outlier.size</code> , <code>outlier.stroke</code> , <code>outlier.alpha</code>	<code>outlier.color</code> , <code>outlier.fill</code> , <code>outlier.shape</code> Default aesthetics for outliers. Set to NULL to inherit from the aesthetics used for the box. In the unlikely event you specify both US and UK spellings of colour, the US spelling will take precedence.
<code>notch</code>	If FALSE (default) make a standard box plot. If TRUE, make a notched box plot. Notches are used to compare groups; if the notches of two boxes do not overlap, this suggests that the medians are significantly different.
<code>notchwidth</code>	For a notched box plot, width of the notch relative to the body (defaults to <code>notchwidth = 0.5</code> ).
<code>staplewidth</code>	The relative width of staples to the width of the box. Staples mark the ends of the whiskers with a line.
<code>varwidth</code>	If FALSE (default) make a standard box plot. If TRUE, boxes are drawn with widths proportional to the square-roots of the number of observations in the groups (possibly weighted, using the <code>weight</code> aesthetic).
<code>fatten</code>	A multiplicative factor used to increase the size of the middle bar in <code>geom_crossbar()</code> and the middle point in <code>geom_pointrange()</code> .
<code>outline.type</code>	Type of the outline of the area; "both" draws both the upper and lower lines, "upper"/"lower" draws the respective lines only. "full" draws a closed polygon around the area.
<code>binwidth</code>	The width of the bins. Can be specified as a numeric value or as a function that calculates width from unscaled x. Here, "unscaled x" refers to the original x values in the data, before application of any scale transformation. When specifying a function along with a grouping structure, the function will be called once per group. The default is to use the number of bins in <code>bins</code> , covering the range of

the data. You should always override this value, exploring multiple widths to find the best to illustrate the stories in your data.

The bin width of a date variable is the number of days in each time; the bin width of a time variable is the number of seconds.

bins	Number of bins. Overridden by <code>binwidth</code> . Defaults to 30.
rule	Either "evenodd" or "winding". If polygons with holes are being drawn (using the subgroup aesthetic) this argument defines how the hole coordinates are interpreted. See the examples in <code>grid::pathGrob()</code> for an explanation.
map	Data frame that contains the map coordinates. This will typically be created using <code>fortify()</code> on a spatial object. It must contain columns <code>x</code> or <code>long</code> , <code>y</code> or <code>lat</code> , and <code>region</code> or <code>id</code> .
draw_quantiles	If not (NULL) (default), draw horizontal lines at the given quantiles of the density estimate.
trim	If TRUE (default), trim the tails of the violins to the range of the data. If FALSE, don't trim the tails.
bounds	Known lower and upper bounds for estimated data. Default <code>c(-Inf, Inf)</code> means that there are no (finite) bounds. If any bound is finite, boundary effect of default density estimation will be corrected by reflecting tails outside bounds around their closest edge. Data points outside of bounds are removed with a warning.
scale	if "area" (default), all violins have the same area (before trimming the tails). If "count", areas are scaled proportionally to the number of observations. If "width", all violins have the same maximum width.

### Value

A `ggplot2::Geom` object.

### Pattern Arguments

Not all arguments apply to all patterns.

`pattern` Pattern name string e.g. 'stripe' (default), 'crosshatch', 'point', 'circle', 'none'

`pattern_alpha` Alpha transparency for pattern. default: 1

`pattern_angle` Orientation of the pattern in degrees. default: 30

`pattern_aspect_ratio` Aspect ratio adjustment.

`pattern_colour` Colour used for strokes and points. default: 'black'

`pattern_density` Approximate fill fraction of the pattern. Usually in range [0, 1], but can be higher. default: 0.2

`pattern_filename` Image filename/URL.

`pattern_fill` Fill colour (or `grid::pattern()/gradient fill`). default: 'grey80'

`pattern_fill2` Second fill colour (or `grid::pattern()/gradient fill`). default: '#4169E1'

`pattern_filter` (Image scaling) filter. default: 'lanczos'

`pattern_frequency` Frequency. default: 0.1

pattern\_gravity Image placement. default: 'center'  
 pattern\_grid Pattern grid type. default: 'square'  
 pattern\_key\_scale\_factor Scale factor for pattern in legend. default: 1  
 pattern\_linetype Stroke linetype. default: 1  
 pattern\_option\_1 Generic user value for custom patterns.  
 pattern\_option\_2 Generic user value for custom patterns.  
 pattern\_option\_3 Generic user value for custom patterns.  
 pattern\_option\_4 Generic user value for custom patterns.  
 pattern\_option\_5 Generic user value for custom patterns.  
 pattern\_orientation 'vertical', 'horizontal', or 'radial'. default: 'vertical'  
 pattern\_res Pattern resolution (pixels per inch).  
 pattern\_rot Rotation angle (shape within pattern). default: 0  
 pattern\_scale Scale. default: 1  
 pattern\_shape Plotting shape. default: 1  
 pattern\_size Stroke line width. default: 1  
 pattern\_spacing Spacing of the pattern as a fraction of the plot size. default: 0.05  
 pattern\_type Generic control option  
 pattern\_subtype Generic control option  
 pattern\_xoffset Offset the origin of the pattern. Range [0, 1]. default: 0. Use this to slightly shift the origin of the pattern. For most patterns, the user should limit the offset value to be less than the pattern spacing.  
 pattern\_yoffset Offset the origin of the pattern. Range [0, 1]. default: 0. Use this to slightly shift the origin of the pattern. For most patterns, the user should limit the offset value to be less than the pattern spacing.

## Examples

```

if (require("ggplot2")) {

  # 'stripe' pattern example
  df <- data.frame(level = c("a", "b", "c", "d"), outcome = c(2.3, 1.9, 3.2, 1))
  gg <- ggplot(df) +
    geom_col_pattern(
      aes(level, outcome, pattern_fill = level),
      pattern = 'stripe',
      fill = 'white',
      colour = 'black'
    ) +
    theme_bw(18) +
    theme(legend.position = 'none') +
    labs(
      title = "ggpattern::geom_col_pattern()",
      subtitle = "pattern = 'stripe'"
    )
}

```

```

plot(gg)

# 'pch' pattern example
gg <- ggplot(mtcars, aes(as.factor(cyl), mpg)) +
  geom_violin_pattern(aes(fill = as.factor(cyl),
                        pattern_shape = as.factor(cyl)),
                    pattern = 'pch',
                    pattern_density = 0.3,
                    pattern_angle = 0,
                    colour = 'black'
  ) +
  theme_bw(18) +
  theme(legend.position = 'none') +
  labs(
    title = "ggpattern::geom_violin_pattern()",
    subtitle = "pattern = 'pch'"
  )
plot(gg)

# 'polygon_tiling' pattern example
gg <- ggplot(mtcars) +
  geom_density_pattern(
    aes(
      x = mpg,
      pattern_fill = as.factor(cyl),
      pattern_type = as.factor(cyl)
    ),
    pattern = 'polygon_tiling',
    pattern_key_scale_factor = 1.2
  ) +
  scale_pattern_type_manual(values = c("hexagonal", "rhombille",
                                       "pythagorean")) +
  theme_bw(18) +
  theme(legend.key.size = unit(2, 'cm')) +
  labs(
    title = "ggpattern::geom_density_pattern()",
    subtitle = "pattern = 'polygon_tiling'"
  )
plot(gg)
}

```

## Description

Geom ggproto objects that could be extended to create a new geom.

**See Also**[ggplot2::Geom](#)

---

ggpattern-defunct      *Defunct data/functions*

---

**Description**

These data/functions are Defunct in this release of ggpattern.

1. For `magick_filter_names` use `magick::filter_types()` instead.
2. For `magick_gravity_names` use `magick::gravity_types()` instead.
3. For `magick_pattern_intensity_names` use `gridpattern::names_magick_intensity`.
4. For `magick_pattern_names` use `gridpattern::names_magick`.
5. For `magick_pattern_stripe_names` use `gridpattern::names_magick_stripe`.
6. For `placeholder_names` use `gridpattern::names_placeholder`.

**Usage**

```
calculate_bbox_polygon_df(...)  
convert_img_to_array(...)  
convert_polygon_df_to_alpha_channel(...)  
convert_polygon_df_to_polygon_grob(...)  
convert_polygon_df_to_polygon_sf(...)  
convert_polygon_sf_to_polygon_df(...)  
create_gradient_img(...)  
fetch_placeholder_img(...)  
fill_area_with_img(...)  
rotate_polygon_df(...)
```

**Arguments**

...                      Ignored

---

is_polygon_df	<i>Test if object is polygon_df or NULL</i>
---------------	---

---

**Description**

Test if object is polygon\_df or NULL

**Usage**

```
is_polygon_df(x)
```

**Arguments**

x                    object

**Value**

TRUE if object is polygon\_df or NULL

**Examples**

```
df <- create_polygon_df(x = c(0, 0, 1, 1), y = c(0, 1, 1, 0))
is_polygon_df(df)
```

---

scale_continuous	<i>Scales for continuous pattern aesthetics</i>
------------------	---

---

**Description**

Scales for continuous pattern aesthetics

**Usage**

```
scale_pattern_angle_continuous(  
  name = waiver(),  
  breaks = waiver(),  
  labels = waiver(),  
  limits = NULL,  
  range = c(0, 90),  
  trans = deprecated(),  
  guide = "legend",  
  ...,  
  transform = "identity"  
)  
  
scale_pattern_angle_discrete(..., range = c(0, 90), guide = "legend")
```



```
scale_pattern_density_continuous(  
  name = waiver(),  
  breaks = waiver(),  
  labels = waiver(),  
  limits = NULL,  
  range = c(0, 0.5),  
  trans = deprecated(),  
  guide = "legend",  
  ...,  
  transform = "identity"  
)  
  
scale_pattern_density_discrete(..., range = c(0, 0.5), guide = "legend")  
  
scale_pattern_spacing_continuous(  
  name = waiver(),  
  breaks = waiver(),  
  labels = waiver(),  
  limits = NULL,  
  range = c(0.01, 0.1),  
  trans = deprecated(),  
  guide = "legend",  
  ...,  
  transform = "identity"  
)  
  
scale_pattern_spacing_discrete(..., range = c(0.01, 0.1), guide = "legend")  
  
scale_pattern_xoffset_continuous(  
  name = waiver(),  
  breaks = waiver(),  
  labels = waiver(),  
  limits = NULL,  
  range = c(0.01, 0.1),  
  trans = deprecated(),  
  guide = "legend",  
  ...,  
  transform = "identity"  
)  
  
scale_pattern_xoffset_discrete(..., range = c(0.01, 0.1), guide = "legend")  
  
scale_pattern_yoffset_continuous(  
  name = waiver(),  
  breaks = waiver(),  
  labels = waiver(),  
  limits = NULL,
```

```
    range = c(0.01, 0.1),
    trans = deprecated(),
    guide = "legend",
    ...,
    transform = "identity"
)

scale_pattern_yoffset_discrete(..., range = c(0.01, 0.1), guide = "legend")

scale_pattern_aspect_ratio_continuous(
  name = waiver(),
  breaks = waiver(),
  labels = waiver(),
  limits = NULL,
  range = c(0.5, 2),
  trans = deprecated(),
  guide = "legend",
  ...,
  transform = "identity"
)

scale_pattern_aspect_ratio_discrete(..., range = c(0.5, 2), guide = "legend")

scale_pattern_key_scale_factor_continuous(
  name = waiver(),
  breaks = waiver(),
  labels = waiver(),
  limits = NULL,
  range = c(0.5, 2),
  trans = deprecated(),
  guide = "legend",
  ...,
  transform = "identity"
)

scale_pattern_key_scale_factor_discrete(
  ...,
  range = c(0.5, 2),
  guide = "legend"
)

scale_pattern_scale_continuous(
  name = waiver(),
  breaks = waiver(),
  labels = waiver(),
  limits = NULL,
  range = c(0.5, 2),
  trans = deprecated(),
```

```
    guide = "legend",
    ...,
    transform = "identity"
)

scale_pattern_scale_discrete(..., range = c(0.5, 2), guide = "legend")

scale_pattern_phase_continuous(
  name = waiver(),
  breaks = waiver(),
  labels = waiver(),
  limits = NULL,
  range = NULL,
  trans = deprecated(),
  guide = "legend",
  ...,
  transform = "identity"
)

scale_pattern_phase_discrete(..., range = NULL, guide = "legend")

scale_pattern_frequency_continuous(
  name = waiver(),
  breaks = waiver(),
  labels = waiver(),
  limits = NULL,
  range = NULL,
  trans = deprecated(),
  guide = "legend",
  ...,
  transform = "identity"
)

scale_pattern_frequency_discrete(..., range = NULL, guide = "legend")

scale_pattern_res_continuous(
  name = waiver(),
  breaks = waiver(),
  labels = waiver(),
  limits = NULL,
  range = NULL,
  trans = deprecated(),
  guide = "legend",
  ...,
  transform = "identity"
)

scale_pattern_res_discrete(..., range = NULL, guide = "legend")
```

```

scale_pattern_rot_continuous(
  name = waiver(),
  breaks = waiver(),
  labels = waiver(),
  limits = NULL,
  range = c(0, 360),
  trans = deprecated(),
  guide = "legend",
  ...,
  transform = "identity"
)

scale_pattern_rot_discrete(..., range = c(0, 360), guide = "legend")

```

### Arguments

name, breaks, labels, limits, range, trans, guide, ..., transform  
 See `{ggplot2}` documentation for more information on scales.

### Value

A `ggplot2::Scale` object.

### Examples

```

if (require('ggplot2')) {

  # 'stripe' pattern example
  df <- data.frame(level = c('a', 'b', 'c', 'd'),
                  outcome = c(2.3, 1.9, 3.2, 1))
  gg <- ggplot(df) +
    geom_col_pattern(
      aes(level, outcome, pattern_fill = level,
          pattern_density = outcome),
      pattern = 'stripe',
      fill = 'white',
      colour = 'black'
    ) +
    theme_bw(18) +
    theme(legend.position = 'none') +
    scale_pattern_density_continuous(range = c(0.1, 0.6)) +
    labs(
      title = 'ggpattern::geom_col_pattern()',
      subtitle = 'pattern = \'stripe\''
    )
  plot(gg)
}

```

---

scale_discrete	<i>Scales for discrete pattern aesthetics</i>
----------------	---

---

**Description**

Scales for discrete pattern aesthetics

**Usage**

```
scale_pattern_type_continuous(  
  name = waiver(),  
  breaks = waiver(),  
  labels = waiver(),  
  limits = NULL,  
  choices = NULL,  
  trans = deprecated(),  
  guide = "legend",  
  ...,  
  transform = "identity"  
)
```

```
scale_pattern_type_discrete(..., choices = NULL, guide = "legend")
```

```
scale_pattern_subtype_continuous(  
  name = waiver(),  
  breaks = waiver(),  
  labels = waiver(),  
  limits = NULL,  
  choices = NULL,  
  trans = deprecated(),  
  guide = "legend",  
  ...,  
  transform = "identity"  
)
```

```
scale_pattern_subtype_discrete(..., choices = NULL, guide = "legend")
```

```
scale_pattern_filename_continuous(  
  name = waiver(),  
  breaks = waiver(),  
  labels = waiver(),  
  limits = NULL,  
  choices = NULL,  
  trans = deprecated(),  
  guide = "legend",  
  ...,  
  transform = "identity"
```

```
)  
  
scale_pattern_filename_discrete(..., choices = NULL, guide = "legend")  
  
scale_pattern_filter_continuous(  
  name = waiver(),  
  breaks = waiver(),  
  labels = waiver(),  
  limits = NULL,  
  choices = c("lanczos", "box", "spline", "cubic"),  
  trans = deprecated(),  
  guide = "legend",  
  ...,  
  transform = "identity"  
)  
  
scale_pattern_filter_discrete(  
  ...,  
  choices = c("lanczos", "box", "spline", "cubic"),  
  guide = "legend"  
)  
  
scale_pattern_gravity_continuous(  
  name = waiver(),  
  breaks = waiver(),  
  labels = waiver(),  
  limits = NULL,  
  choices = c("center", "north", "south", "east", "west", "northeast", "northwest",  
    "southeast", "southwest"),  
  trans = deprecated(),  
  guide = "legend",  
  ...,  
  transform = "identity"  
)  
  
scale_pattern_gravity_discrete(  
  ...,  
  choices = c("center", "north", "south", "east", "west", "northeast", "northwest",  
    "southeast", "southwest"),  
  guide = "legend"  
)  
  
scale_pattern_orientation_continuous(  
  name = waiver(),  
  breaks = waiver(),  
  labels = waiver(),  
  limits = NULL,  
  choices = c("horizontal", "vertical", "radial"),
```

```
    trans = deprecated(),
    guide = "legend",
    ...,
    transform = "identity"
)

scale_pattern_orientation_discrete(
  ...,
  choices = c("horizontal", "vertical", "radial"),
  guide = "legend"
)

scale_pattern_grid_continuous(
  name = waiver(),
  breaks = waiver(),
  labels = waiver(),
  limits = NULL,
  choices = c("square", "hex"),
  trans = deprecated(),
  guide = "legend",
  ...,
  transform = "identity"
)

scale_pattern_grid_discrete(
  ...,
  choices = c("square", "hex"),
  guide = "legend"
)

scale_pattern_units_continuous(
  name = waiver(),
  breaks = waiver(),
  labels = waiver(),
  limits = NULL,
  choices = c("snpc", "cm", "inches"),
  trans = deprecated(),
  guide = "legend",
  ...,
  transform = "identity"
)

scale_pattern_units_discrete(
  ...,
  choices = c("snpc", "cm", "inches"),
  guide = "legend"
)
```

```

scale_pattern_continuous(
  name = waiver(),
  breaks = waiver(),
  labels = waiver(),
  limits = NULL,
  choices = c("stripe", "crosshatch", "circle"),
  trans = deprecated(),
  guide = "legend",
  ...,
  transform = "identity",
  na.value = "none"
)

scale_pattern_discrete(
  ...,
  choices = c("stripe", "crosshatch", "circle"),
  guide = "legend",
  na.value = "none"
)

```

### Arguments

name, breaks, labels, limits, trans, guide, ..., transform, na.value  
 See {ggplot2} documentation for more information on scales.

choices            vector of values to choose from.

### Value

A `ggplot2::Scale` object.

### Examples

```

if (require('ggplot2')) {
  gg <- ggplot(mtcars) +
    geom_density_pattern(
      aes(
        x = mpg,
        pattern_fill = as.factor(cyl),
        pattern_type = as.factor(cyl)
      ),
      pattern = 'polygon_tiling',
      pattern_key_scale_factor = 1.2
    ) +
    scale_pattern_type_discrete(choices = gridpattern::names_polygon_tiling) +
    theme_bw(18) +
    theme(legend.key.size = unit(2, 'cm')) +
    labs(
      title = 'ggpattern::geom_density_pattern()',
      subtitle = 'pattern = \'polygon_tiling\''
    )
}

```



```
    plot(gg)
  }
```

---

scale\_pattern\_alpha\_continuous  
*Alpha transparency scales*

---

## Description

See `ggplot2::scale_alpha()` for details.

## Usage

```
scale_pattern_alpha_continuous(..., range = c(0.1, 1))

scale_pattern_alpha(..., range = c(0.1, 1))

scale_pattern_alpha_discrete(...)

scale_pattern_alpha_ordinal(..., range = c(0.1, 1))
```

## Arguments

...	Other arguments passed on to <code>ggplot2::continuous_scale()</code> , <code>ggplot2::binned_scale()</code> , or <code>ggplot2::discrete_scale()</code> as appropriate, to control name, limits, breaks, labels and so forth.
range	Output range of alpha values. Must lie between 0 and 1.

## Value

A `ggplot2::Scale` object.

## Examples

```
if (require("ggplot2")) {
  # 'stripe' pattern example
  df <- data.frame(level = c("a", "b", "c", "d"), outcome = c(2.3, 1.9, 3.2, 1))
  gg <- ggplot(df) +
    geom_col_pattern(
      aes(level, outcome, pattern_fill = level, pattern_alpha = outcome),
      pattern_density = 0.6,
      pattern_size = 1.5,
      pattern = 'stripe',
      fill = 'white',
      colour = 'black',
      linewidth = 1.5
    ) +
    theme_bw(18) +
    theme(legend.position = 'none') +
```

```
scale_pattern_alpha() +  
labs(  
  title = "ggpattern::geom_col_pattern()",  
  subtitle = "pattern = 'stripe'"  
)  
plot(gg)  
}
```

---

scale\_pattern\_colour\_brewer

*Sequential, diverging and qualitative colour scales from colorbrewer.org*

---

## Description

The brewer scales provides sequential, diverging and qualitative colour schemes from ColorBrewer. These are particularly well suited to display discrete values on a map. See <https://colorbrewer2.org> for more information.

## Usage

```
scale_pattern_colour_brewer(  
  ...,  
  type = "seq",  
  palette = 1,  
  direction = 1,  
  aesthetics = "pattern_colour"  
)  
  
scale_pattern_fill_brewer(  
  ...,  
  type = "seq",  
  palette = 1,  
  direction = 1,  
  aesthetics = "pattern_fill"  
)  
  
scale_pattern_fill2_brewer(  
  ...,  
  type = "seq",  
  palette = 1,  
  direction = 1,  
  aesthetics = "pattern_fill2"  
)  
  
scale_pattern_colour_distiller(  
  ...,
```

```

    type = "seq",
    palette = 1,
    direction = -1,
    values = NULL,
    space = "Lab",
    na.value = "grey50",
    guide = guide_colourbar(available_aes = "pattern_colour"),
    aesthetics = "pattern_colour"
  )

scale_pattern_fill_distiller(
  ...,
  type = "seq",
  palette = 1,
  direction = -1,
  values = NULL,
  space = "Lab",
  na.value = "grey50",
  guide = guide_colourbar(available_aes = "pattern_fill"),
  aesthetics = "pattern_fill"
)

scale_pattern_fill2_distiller(
  ...,
  type = "seq",
  palette = 1,
  direction = -1,
  values = NULL,
  space = "Lab",
  na.value = "grey50",
  guide = guide_colourbar(available_aes = "pattern_fill2"),
  aesthetics = "pattern_fill2"
)

```

### Arguments

...	Other arguments passed on to <code>ggplot2::discrete_scale()</code> , <code>ggplot2::continuous_scale()</code> , or <code>ggplot2::binned_scale()</code> , for brewer, distiller, and fermenter variants respectively, to control name, limits, breaks, labels and so forth.
palette	If a string, will use that named palette. If a number, will index into the list of palettes of appropriate type. The list of available palettes can found in the Palettes section.
direction, type, aesthetics, values, space, na.value, guide	See <code>ggplot2::scale_colour_brewer</code> for more information.

### Details

The brewer scales were carefully designed and tested on discrete data. They were not designed to be extended to continuous data, but results often look good. Your mileage may vary.

**Value**

A `ggplot2::Scale` object.

**Palettes**

The following palettes are available for use with these scales:

**Diverging** BrBG, PiYG, PRGn, PuOr, RdBu, RdGy, RdYlBu, RdYlGn, Spectral

**Qualitative** Accent, Dark2, Paired, Pastel1, Pastel2, Set1, Set2, Set3

**Sequential** Blues, BuGn, BuPu, GnBu, Greens, Greys, Oranges, OrRd, PuBu, PuBuGn, PuRd, Purples, RdPu, Reds, YlGn, YlGnBu, YlOrBr, YlOrRd

Modify the palette through the `palette` argument.

**Note**

The `distiller` scales extend `brewer` to continuous scales by smoothly interpolating 7 colours from any palette to a continuous scale. The `fermenter` scales provide binned versions of the `brewer` scales.

**Examples**

```
if (require("ggplot2")) {
  df <- data.frame(level = c("a", "b", "c", "d"),
                  outcome = c(2.3, 1.9, 3.2, 1))
  # discrete 'brewer' palette
  gg <- ggplot(df) +
    geom_col_pattern(
      aes(level, outcome, pattern_fill = level),
      pattern = 'stripe',
      fill = 'white',
      colour = 'black'
    ) +
    theme_bw(18) +
    scale_pattern_fill_brewer()
  plot(gg)

  # continuous 'distiller' palette
  gg <- ggplot(df) +
    geom_col_pattern(
      aes(level, outcome, pattern_fill = outcome),
      pattern = 'stripe',
      fill = 'white',
      colour = 'black'
    ) +
    theme_bw(18) +
    scale_pattern_fill_distiller()
  plot(gg)
}
```

---

scale\_pattern\_colour\_continuous  
*Continuous and binned colour scales*

---

## Description

See `ggplot2::scale_colour_continuous()` for more information

## Usage

```
scale_pattern_colour_continuous(  
  ...,  
  type = getOption("ggplot2.continuous.colour", default = "gradient")  
)  
  
scale_pattern_fill_continuous(  
  ...,  
  type = getOption("ggplot2.continuous.fill", default = "gradient")  
)  
  
scale_pattern_fill2_continuous(  
  ...,  
  type = getOption("ggplot2.continuous.fill", default = "gradient")  
)
```

## Arguments

... Additional parameters passed on to the scale type  
type One of "gradient" (the default) or "viridis" indicating the colour scale to use

## Value

A `ggplot2::Scale` object.

## Examples

```
if (require("ggplot2")) {  
  df <- data.frame(level = c("a", "b", "c", "d"),  
                  outcome = c(2.3, 1.9, 3.2, 1))  
  gg <- ggplot(df) +  
    geom_col_pattern(  
      aes(level, outcome, pattern_fill = outcome),  
      pattern = 'stripe',  
      fill = 'white',  
      colour = 'black'  
    ) +  
    theme_bw(18) +  
    scale_pattern_fill_continuous()
```

```
    plot(gg)
  }
```

---

scale\_pattern\_colour\_gradient

*Gradient colour scales*

---

### Description

See `ggplot2::scale_colour_gradient()` for more information

### Usage

```
scale_pattern_colour_gradient(
  ...,
  low = "#132B43",
  high = "#56B1F7",
  space = "Lab",
  na.value = "grey50",
  guide = guide_colourbar(available_aes = "pattern_colour"),
  aesthetics = "pattern_colour"
)

scale_pattern_fill_gradient(
  ...,
  low = "#132B43",
  high = "#56B1F7",
  space = "Lab",
  na.value = "grey50",
  guide = guide_colourbar(available_aes = "pattern_fill"),
  aesthetics = "pattern_fill"
)

scale_pattern_fill2_gradient(
  ...,
  low = "#132B43",
  high = "#56B1F7",
  space = "Lab",
  na.value = "grey50",
  guide = guide_colourbar(available_aes = "pattern_fill2"),
  aesthetics = "pattern_fill2"
)

scale_pattern_colour_gradient2(
  ...,
  low = muted("red"),
  mid = "white",
```

```
    high = muted("blue"),
    midpoint = 0,
    space = "Lab",
    na.value = "grey50",
    guide = guide_colourbar(available_aes = "pattern_colour"),
    aesthetics = "pattern_colour"
)

scale_pattern_fill_gradient2(
  ...,
  low = muted("red"),
  mid = "white",
  high = muted("blue"),
  midpoint = 0,
  space = "Lab",
  na.value = "grey50",
  guide = guide_colourbar(available_aes = "pattern_fill"),
  aesthetics = "pattern_fill"
)

scale_pattern_fill2_gradient2(
  ...,
  low = muted("red"),
  mid = "white",
  high = muted("blue"),
  midpoint = 0,
  space = "Lab",
  na.value = "grey50",
  guide = guide_colourbar(available_aes = "pattern_fill2"),
  aesthetics = "pattern_fill2"
)

scale_pattern_colour_gradientn(
  ...,
  colours,
  values = NULL,
  space = "Lab",
  na.value = "grey50",
  guide = guide_colourbar(available_aes = "pattern_colour"),
  aesthetics = "pattern_colour",
  colors
)

scale_pattern_fill_gradientn(
  ...,
  colours,
  values = NULL,
  space = "Lab",
```

```

na.value = "grey50",
guide = guide_colourbar(available_aes = "pattern_fill"),
aesthetics = "pattern_fill",
colors
)

scale_pattern_fill2_gradientn(
  ...,
  colours,
  values = NULL,
  space = "Lab",
  na.value = "grey50",
  guide = guide_colourbar(available_aes = "pattern_fill2"),
  aesthetics = "pattern_fill2",
  colors
)

```

### Arguments

low, high	Colours for low and high ends of the gradient.
space, ..., na.value, aesthetics	See <code>scales::seq_gradient_pal</code> , <code>scale_colour_hue</code> , <code>ggplot2::continuous_scale</code>
guide	Type of legend. Use "colourbar" for continuous colour bar, or "legend" for discrete colour legend.
mid	colour for mid point
midpoint	The midpoint (in data value) of the diverging scale. Defaults to 0.
colours, colors	Vector of colours to use for n-colour gradient.
values	if colours should not be evenly positioned along the gradient this vector gives the position (between 0 and 1) for each colour in the colours vector. See <a href="#">rescale()</a> for a convenience function to map an arbitrary range to between 0 and 1.

### Details

`scale*_gradient` creates a two colour gradient (low-high), `scale*_gradient2` creates a diverging colour gradient (low-mid-high), `scale*_gradientn` creates a n-colour gradient.

### Value

A `ggplot2::Scale` object.

### Examples

```

if (require("ggplot2")) {
  df <- data.frame(level = c("a", "b", "c", "d"),
                   outcome = c(2.3, 1.9, 3.2, 1))
  gg <- ggplot(df) +
    geom_col_pattern(

```



```
    aes(level, outcome, pattern_fill = outcome),
    pattern = 'stripe',
    fill     = 'white',
    colour  = 'black'
  ) +
  theme_bw(18) +
  scale_pattern_fill_gradient()
plot(gg)
}
```

---

scale\_pattern\_colour\_grey

*Sequential grey colour scales*

---

### Description

Based on [gray.colors\(\)](#). This is black and white equivalent of [scale\\_pattern\\_colour\\_gradient\(\)](#).

### Usage

```
scale_pattern_colour_grey(
  ...,
  start = 0.2,
  end = 0.8,
  na.value = "red",
  aesthetics = "pattern_colour"
)
```

```
scale_pattern_fill_grey(
  ...,
  start = 0.2,
  end = 0.8,
  na.value = "red",
  aesthetics = "pattern_fill"
)
```

```
scale_pattern_fill2_grey(
  ...,
  start = 0.2,
  end = 0.8,
  na.value = "red",
  aesthetics = "pattern_fill2"
)
```

### Arguments

..., start, end, na.value, aesthetics

See `ggplot2::scale_colour_grey` for more information

**Value**

A `ggplot2::Scale` object.

**Examples**

```
if (require("ggplot2")) {
  df <- data.frame(level = c("a", "b", "c", "d"),
                  outcome = c(2.3, 1.9, 3.2, 1))
  gg <- ggplot(df) +
    geom_col_pattern(
      aes(level, outcome, pattern_fill = level),
      pattern = 'stripe',
      fill = 'white',
      colour = 'black'
    ) +
    theme_bw(18) +
    scale_pattern_fill_grey()
  plot(gg)
}
```

---

scale\_pattern\_colour\_hue

*Evenly spaced colours for discrete data*

---

**Description**

This is the default colour scale for categorical variables. It maps each level to an evenly spaced hue on the colour wheel. It does not generate colour-blind safe palettes.

**Usage**

```
scale_pattern_colour_hue(
  ...,
  h = c(0, 360) + 15,
  c = 100,
  l = 65,
  h.start = 0,
  direction = 1,
  na.value = "grey50",
  aesthetics = "pattern_colour"
)
```

```
scale_pattern_fill_hue(
  ...,
  h = c(0, 360) + 15,
  c = 100,
  l = 65,
  h.start = 0,
```

```

direction = 1,
na.value = "grey50",
aesthetics = "pattern_fill"
)

scale_pattern_fill2_hue(
  ...,
  h = c(0, 360) + 15,
  c = 100,
  l = 65,
  h.start = 0,
  direction = 1,
  na.value = "grey50",
  aesthetics = "pattern_fill2"
)

```

### Arguments

h, c, l, h.start, direction, ...	See <code>ggplot2::scale_colour_hue</code>
na.value	Colour to use for missing values
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 <code>aesthetics = c("colour", "fill")</code> .

### Value

A `ggplot2::Scale` object.

### Examples

```

if (require("ggplot2")) {
  df <- data.frame(level = c("a", "b", "c", "d"),
                  outcome = c(2.3, 1.9, 3.2, 1))
  gg <- ggplot(df) +
    geom_col_pattern(
      aes(level, outcome, pattern_fill = level),
      pattern = 'stripe',
      fill = 'white',
      colour = 'black'
    ) +
    theme_bw(18) +
    scale_pattern_fill_hue()
  plot(gg)
}

```

---

`scale_pattern_colour_viridis_d`*Viridis colour scales from viridisLite*

---

### Description

The viridis scales provide colour maps that are perceptually uniform in both colour and black-and-white. They are also designed to be perceived by viewers with common forms of colour blindness. See also <https://bids.github.io/colormap/>.

### Usage

```
scale_pattern_colour_viridis_d(  
  ...,  
  alpha = 1,  
  begin = 0,  
  end = 1,  
  direction = 1,  
  option = "D",  
  aesthetics = "pattern_colour"  
)
```

```
scale_pattern_fill_viridis_d(  
  ...,  
  alpha = 1,  
  begin = 0,  
  end = 1,  
  direction = 1,  
  option = "D",  
  aesthetics = "pattern_fill"  
)
```

```
scale_pattern_fill2_viridis_d(  
  ...,  
  alpha = 1,  
  begin = 0,  
  end = 1,  
  direction = 1,  
  option = "D",  
  aesthetics = "pattern_fill2"  
)
```

```
scale_pattern_colour_viridis_c(  
  ...,  
  alpha = 1,  
  begin = 0,  
  end = 1,
```

```

    direction = 1,
    option = "D",
    values = NULL,
    space = "Lab",
    na.value = "grey50",
    guide = guide_colourbar(available_aes = "pattern_colour"),
    aesthetics = "pattern_colour"
  )

scale_pattern_fill_viridis_c(
  ...,
  alpha = 1,
  begin = 0,
  end = 1,
  direction = 1,
  option = "D",
  values = NULL,
  space = "Lab",
  na.value = "grey50",
  guide = guide_colourbar(available_aes = "pattern_fill"),
  aesthetics = "pattern_fill"
)

scale_pattern_fill2_viridis_c(
  ...,
  alpha = 1,
  begin = 0,
  end = 1,
  direction = 1,
  option = "D",
  values = NULL,
  space = "Lab",
  na.value = "grey50",
  guide = guide_colourbar(available_aes = "pattern_fill2"),
  aesthetics = "pattern_fill2"
)

```

## Arguments

... Other arguments passed on to `ggplot2::discrete_scale()`, `ggplot2::continuous_scale()`, or `ggplot2::binned_scale()` to control name, limits, breaks, labels and so forth.

begin, end, alpha, direction, option, values, space, na.value, guide  
See `ggplot2::scale_colour_viridis_d()` for more information

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")`.

**Value**

A `ggplot2::Scale` object.

**Examples**

```
if (require("ggplot2")) {  
  df <- data.frame(level = c("a", "b", "c", "d"),  
                  outcome = c(2.3, 1.9, 3.2, 1))  
  # discrete 'viridis' palette  
  gg <- ggplot(df) +  
    geom_col_pattern(  
      aes(level, outcome, pattern_fill = level),  
      pattern = 'stripe',  
      fill = 'white',  
      colour = 'black'  
    ) +  
    theme_bw(18) +  
    scale_pattern_fill_viridis_d()  
  plot(gg)  
  
  # continuous 'viridis' palette  
  gg <- ggplot(df) +  
    geom_col_pattern(  
      aes(level, outcome, pattern_fill = outcome),  
      pattern = 'stripe',  
      fill = 'white',  
      colour = 'black'  
    ) +  
    theme_bw(18) +  
    scale_pattern_fill_viridis_c()  
  plot(gg)  
}
```

---

scale\_pattern\_identity

*Use values without scaling*

---

**Description**

Use values without scaling

**Usage**

```
scale_pattern_type_identity(..., guide = "none")
```

```
scale_pattern_subtype_identity(..., guide = "none")
```

```
scale_pattern_angle_identity(..., guide = "none")
```

```
scale_pattern_density_identity(..., guide = "none")
scale_pattern_spacing_identity(..., guide = "none")
scale_pattern_xoffset_identity(..., guide = "none")
scale_pattern_yoffset_identity(..., guide = "none")
scale_pattern_alpha_identity(..., guide = "none")
scale_pattern_linetype_identity(..., guide = "none")
scale_pattern_size_identity(..., guide = "none")
scale_pattern_shape_identity(..., guide = "none")
scale_pattern_colour_identity(..., guide = "none")
scale_pattern_fill_identity(..., guide = "none")
scale_pattern_fill2_identity(..., guide = "none")
scale_pattern_aspect_ratio_identity(..., guide = "none")
scale_pattern_key_scale_factor_identity(..., guide = "none")
scale_pattern_filename_identity(..., guide = "none")
scale_pattern_filter_identity(..., guide = "none")
scale_pattern_gravity_identity(..., guide = "none")
scale_pattern_scale_identity(..., guide = "none")
scale_pattern_orientation_identity(..., guide = "none")
scale_pattern_phase_identity(..., guide = "none")
scale_pattern_frequency_identity(..., guide = "none")
scale_pattern_grid_identity(..., guide = "none")
scale_pattern_res_identity(..., guide = "none")
scale_pattern_rot_identity(..., guide = "none")
scale_pattern_units_identity(..., guide = "none")
```

```
scale_pattern_identity(..., guide = "none")
```

### Arguments

..., guide See `ggplot2` for documentation on identity scales. e.g. `ggplot2::scale_alpha_identity()`

### Value

A `ggplot2::Scale` object.

### Examples

```
if (require('ggplot2')) {
  df <- data.frame(outcome = c(2.3, 1.9, 3.2, 1),
                  pattern_type = sample(gridpattern::names_polygon_tiling, 4))
  gg <- ggplot(df) +
    geom_col_pattern(
      aes(pattern_type, outcome, pattern_fill = pattern_type,
          pattern_type = pattern_type),
      colour = 'black',
      pattern = 'polygon_tiling',
      pattern_key_scale_factor = 1.2
    ) +
    scale_pattern_type_identity() +
    theme_bw(18) +
    theme(legend.position = 'none') +
    labs(
      x = 'level',
      title = 'ggpattern::geom_col_pattern()',
      subtitle = 'pattern = \'polygon_tiling\''
    )
  plot(gg)
}
```

---

scale\_pattern\_linetype

*Scale for line patterns*

---

### Description

Default line types based on a set supplied by Richard Pearson, University of Manchester. Continuous values can not be mapped to line types.

### Usage

```
scale_pattern_linetype(..., na.value = "blank")
```

```
scale_pattern_linetype_continuous(...)
```

```
scale_pattern_linetype_discrete(..., na.value = "blank")
```



**Arguments**

... see `ggplot2::scale_linetype()` for more information  
na.value The linetype to use for NA values.

**Value**

A `ggplot2::Scale` object.

**Examples**

```
if (require("ggplot2")) {  
  # 'stripe' pattern example  
  df <- data.frame(level = c("a", "b", "c", "d"), outcome = c(2.3, 1.9, 3.2, 1))  
  gg <- ggplot(df) +  
    geom_col_pattern(  
      aes(level, outcome, pattern_fill = level, pattern_linetype = level),  
      pattern_density = 0.6,  
      pattern_size = 1.5,  
      pattern = 'stripe',  
      fill = 'white',  
      colour = 'black',  
      linewidth = 1.5  
    ) +  
    theme_bw(18) +  
    theme(legend.position = 'none') +  
    scale_pattern_linetype() +  
    labs(  
      title = "ggpattern::geom_col_pattern()",  
      subtitle = "pattern = 'stripe'"  
    )  
  plot(gg)  
}
```

---

scale\_pattern\_manual *Create your own discrete scale*

---

**Description**

Create your own discrete scale

**Usage**

```
scale_pattern_type_manual(..., values, breaks = waiver())
```

```
scale_pattern_subtype_manual(..., values, breaks = waiver())
```

```
scale_pattern_angle_manual(..., values, breaks = waiver())
```

```
scale_pattern_density_manual(..., values, breaks = waiver())
scale_pattern_spacing_manual(..., values, breaks = waiver())
scale_pattern_xoffset_manual(..., values, breaks = waiver())
scale_pattern_yoffset_manual(..., values, breaks = waiver())
scale_pattern_alpha_manual(..., values, breaks = waiver())
scale_pattern_linetype_manual(..., values, breaks = waiver())
scale_pattern_size_manual(..., values, breaks = waiver())
scale_pattern_shape_manual(..., values, breaks = waiver())
scale_pattern_colour_manual(..., values, breaks = waiver())
scale_pattern_fill_manual(..., values, breaks = waiver())
scale_pattern_fill2_manual(..., values, breaks = waiver())
scale_pattern_aspect_ratio_manual(..., values, breaks = waiver())
scale_pattern_key_scale_factor_manual(..., values, breaks = waiver())
scale_pattern_filename_manual(..., values, breaks = waiver())
scale_pattern_filter_manual(..., values, breaks = waiver())
scale_pattern_gravity_manual(..., values, breaks = waiver())
scale_pattern_scale_manual(..., values, breaks = waiver())
scale_pattern_orientation_manual(..., values, breaks = waiver())
scale_pattern_phase_manual(..., values, breaks = waiver())
scale_pattern_frequency_manual(..., values, breaks = waiver())
scale_pattern_grid_manual(..., values, breaks = waiver())
scale_pattern_res_manual(..., values, breaks = waiver())
scale_pattern_rot_manual(..., values, breaks = waiver())
scale_pattern_units_manual(..., values, breaks = waiver())
```

```
scale_pattern_manual(..., values, breaks = waiver(), na.value = "none")
```

### Arguments

..., values, breaks, na.value

See `ggplot2` for documentation on manual scales. e.g. `ggplot2::scale_colour_manual()`

### Value

A `ggplot2::Scale` object.

### Examples

```
if (require('ggplot2')) {  
  gg <- ggplot(mtcars) +  
    geom_density_pattern(  
      aes(  
        x = mpg,  
        pattern_fill = as.factor(cyl),  
        pattern_type = as.factor(cyl)  
      ),  
      pattern = 'polygon_tiling',  
      pattern_key_scale_factor = 1.2  
    ) +  
    scale_pattern_type_manual(values = c('hexagonal', 'rhombille',  
                                         'pythagorean')) +  
    theme_bw(18) +  
    theme(legend.key.size = unit(2, 'cm')) +  
    labs(  
      title = 'ggpattern::geom_density_pattern()',  
      subtitle = 'pattern = \'polygon_tiling\''  
    )  
  plot(gg)  
}
```

---

scale\_pattern\_shape    *Scales for shapes, aka glyphs*

---

### Description

`scale_pattern_shape` maps discrete variables to six easily discernible shapes. If you have more than six levels, you will get a warning message, and the seventh and subsequent levels will not appear on the plot. Use `scale_pattern_shape_manual()` to supply your own values. You can not map a continuous variable to shape unless `scale_pattern_shape_binned()` is used. Still, as shape has no inherent order, this use is not advised..

**Usage**

```
scale_pattern_shape(..., solid = TRUE)

scale_pattern_shape_discrete(..., solid = TRUE)

scale_pattern_shape_ordinal(...)

scale_pattern_shape_continuous(...)
```

**Arguments**

...	other arguments passed to <code>discrete_scale()</code>
<code>solid</code>	Should the shapes be solid, TRUE, or hollow, FALSE?

**Details**

Scales for area or radius

**Value**

A `ggplot2::Scale` object.

**Examples**

```
if (require("ggplot2")) {
  # 'pch' pattern example
  gg <- ggplot(mtcars, aes(as.factor(cyl), mpg)) +
    geom_violin_pattern(aes(fill = as.factor(cyl),
                          pattern_shape = as.factor(cyl)),
                      pattern = 'pch',
                      pattern_density = 0.3,
                      pattern_angle = 0,
                      colour = 'black'
                    ) +
    theme_bw(18) +
    theme(legend.position = 'none') +
    scale_pattern_shape() +
    labs(
      title = "ggpattern::geom_violin_pattern()",
      subtitle = "pattern = 'pch'"
    )
  plot(gg)
}
```

---

scale\_pattern\_size\_continuous  
*Scales for area or radius*

---

## Description

Scales for area or radius

## Usage

```
scale_pattern_size_continuous(  
  name = waiver(),  
  breaks = waiver(),  
  labels = waiver(),  
  limits = NULL,  
  range = c(1, 6),  
  trans = deprecated(),  
  guide = "legend",  
  ...,  
  transform = "identity"  
)
```

```
scale_pattern_size(  
  name = waiver(),  
  breaks = waiver(),  
  labels = waiver(),  
  limits = NULL,  
  range = c(1, 6),  
  trans = deprecated(),  
  guide = "legend",  
  ...,  
  transform = "identity"  
)
```

## Arguments

name, breaks, labels, limits, trans, guide, ..., transform

See [ggplot2::scale\\_size\(\)](#) for more information

range            a numeric vector of length 2 that specifies the minimum and maximum size of the plotting symbol after transformation.

## Value

A [ggplot2::Scale](#) object.

**Examples**

```
if (require("ggplot2")) {  
  # 'circle' pattern example  
  df <- data.frame(level = c("a", "b", "c", "d"), outcome = c(2.3, 1.9, 3.2, 1))  
  gg <- ggplot(df) +  
    geom_col_pattern(  
      aes(level, outcome, pattern_fill = level,  
          linewidth = outcome, pattern_size = outcome),  
      pattern_density = 0.4,  
      pattern_spacing = 0.3,  
      pattern = 'circle',  
      fill = 'white',  
      colour = 'black'  
    ) +  
    theme_bw(18) +  
    theme(legend.position = 'none') +  
    scale_pattern_size() +  
    labs(  
      title = "ggpattern::geom_col_pattern()",  
      subtitle = "pattern = 'circle'"  
    )  
  plot(gg)  
}
```

# Index

- \* **datasets**
  - GeomRectPattern, [14](#)
- aes(), [9](#)
- borders(), [11](#)
- calculate\_bbox\_polygon\_df
  - (ggpattern-defunct), [15](#)
- convert\_img\_to\_array
  - (ggpattern-defunct), [15](#)
- convert\_polygon\_df\_to\_alpha\_channel
  - (ggpattern-defunct), [15](#)
- convert\_polygon\_df\_to\_polygon\_grob
  - (ggpattern-defunct), [15](#)
- convert\_polygon\_df\_to\_polygon\_sf
  - (ggpattern-defunct), [15](#)
- convert\_polygon\_sf\_to\_polygon\_df
  - (ggpattern-defunct), [15](#)
- create\_gradient\_img
  - (ggpattern-defunct), [15](#)
- create\_polygon\_df, [3](#)
- draw\_key\_boxplot\_pattern
  - (draw\_key\_polygon\_pattern), [3](#)
- draw\_key\_crossbar\_pattern
  - (draw\_key\_polygon\_pattern), [3](#)
- draw\_key\_polygon\_pattern, [3](#)
- fetch\_placeholder\_img
  - (ggpattern-defunct), [15](#)
- fill\_area\_with\_img (ggpattern-defunct), [15](#)
- fortify(), [9](#), [12](#)
- geom-docs, [5](#)
- geom\_area\_pattern (geom-docs), [5](#)
- geom\_bar\_pattern (geom-docs), [5](#)
- geom\_bin2d\_pattern (geom-docs), [5](#)
- geom\_bin\_2d\_pattern (geom-docs), [5](#)
- geom\_boxplot\_pattern (geom-docs), [5](#)
- geom\_col\_pattern (geom-docs), [5](#)
- geom\_crossbar\_pattern (geom-docs), [5](#)
- geom\_density\_pattern (geom-docs), [5](#)
- geom\_histogram\_pattern (geom-docs), [5](#)
- geom\_map\_pattern (geom-docs), [5](#)
- geom\_polygon\_pattern (geom-docs), [5](#)
- geom\_rect\_pattern (geom-docs), [5](#)
- geom\_ribbon\_pattern (geom-docs), [5](#)
- geom\_sf\_pattern (geom-docs), [5](#)
- geom\_tile\_pattern (geom-docs), [5](#)
- geom\_violin\_pattern (geom-docs), [5](#)
- GeomAreaPattern (GeomRectPattern), [14](#)
- GeomBarPattern (GeomRectPattern), [14](#)
- GeomBoxplotPattern (GeomRectPattern), [14](#)
- GeomColPattern (GeomRectPattern), [14](#)
- GeomCrossbarPattern (GeomRectPattern), [14](#)
- GeomDensityPattern (GeomRectPattern), [14](#)
- GeomMapPattern (GeomRectPattern), [14](#)
- GeomPolygonPattern (GeomRectPattern), [14](#)
- GeomRectPattern, [14](#)
- GeomRibbonPattern (GeomRectPattern), [14](#)
- GeomSfPattern (GeomRectPattern), [14](#)
- GeomTilePattern (GeomRectPattern), [14](#)
- GeomViolinPattern (GeomRectPattern), [14](#)
- ggpattern-defunct, [15](#)
- ggpattern-ggproto (GeomRectPattern), [14](#)
- ggplot(), [9](#)
- ggplot2::binned\_scale(), [25](#), [27](#), [37](#)
- ggplot2::continuous\_scale(), [25](#), [27](#), [37](#)
- ggplot2::discrete\_scale(), [25](#), [27](#), [37](#)
- ggplot2::Geom, [12](#), [15](#)
- ggplot2::layer(), [3](#)
- ggplot2::Scale, [20](#), [24](#), [25](#), [28](#), [29](#), [32](#), [34](#), [35](#), [38](#), [40](#), [41](#), [43–45](#)
- ggplot2::scale\_alpha(), [25](#)
- ggplot2::scale\_colour\_viridis\_d(), [37](#)
- ggplot2::scale\_linetype(), [41](#)
- ggplot2::scale\_size(), [45](#)

- gray.colors(), [33](#)
- grid::pathGrob(), [12](#)
- grid::pattern(), [12](#)
- is\_polygon\_df, [16](#)
- key glyphs, [10](#)
- layer position, [10](#)
- layer stat, [10](#)
- layer(), [10](#)
- magick\_filter\_names
  - (ggpattern-defunct), [15](#)
- magick\_gravity\_names
  - (ggpattern-defunct), [15](#)
- magick\_pattern\_intensity\_names
  - (ggpattern-defunct), [15](#)
- magick\_pattern\_names
  - (ggpattern-defunct), [15](#)
- magick\_pattern\_stripe\_names
  - (ggpattern-defunct), [15](#)
- placeholder\_names (ggpattern-defunct),
  - [15](#)
- rescale(), [32](#)
- resolution(), [11](#)
- rotate\_polygon\_df (ggpattern-defunct),
  - [15](#)
- scale\_continuous, [16](#)
- scale\_discrete, [21](#)
- scale\_pattern\_alpha
  - (scale\_pattern\_alpha\_continuous),
  - [25](#)
- scale\_pattern\_alpha\_continuous, [25](#)
- scale\_pattern\_alpha\_discrete
  - (scale\_pattern\_alpha\_continuous),
  - [25](#)
- scale\_pattern\_alpha\_identity
  - (scale\_pattern\_identity), [38](#)
- scale\_pattern\_alpha\_manual
  - (scale\_pattern\_manual), [41](#)
- scale\_pattern\_alpha\_ordinal
  - (scale\_pattern\_alpha\_continuous),
  - [25](#)
- scale\_pattern\_angle\_continuous
  - (scale\_continuous), [16](#)
- scale\_pattern\_angle\_discrete
  - (scale\_continuous), [16](#)
- scale\_pattern\_angle\_identity
  - (scale\_pattern\_identity), [38](#)
- scale\_pattern\_angle\_manual
  - (scale\_pattern\_manual), [41](#)
- scale\_pattern\_aspect\_ratio\_continuous
  - (scale\_continuous), [16](#)
- scale\_pattern\_aspect\_ratio\_discrete
  - (scale\_continuous), [16](#)
- scale\_pattern\_aspect\_ratio\_identity
  - (scale\_pattern\_identity), [38](#)
- scale\_pattern\_aspect\_ratio\_manual
  - (scale\_pattern\_manual), [41](#)
- scale\_pattern\_color\_brewer
  - (scale\_pattern\_colour\_brewer),
  - [26](#)
- scale\_pattern\_color\_continuous
  - (scale\_pattern\_colour\_continuous),
  - [29](#)
- scale\_pattern\_color\_discrete
  - (scale\_pattern\_colour\_hue), [34](#)
- scale\_pattern\_color\_distiller
  - (scale\_pattern\_colour\_brewer),
  - [26](#)
- scale\_pattern\_color\_gradient
  - (scale\_pattern\_colour\_gradient),
  - [30](#)
- scale\_pattern\_color\_gradient2
  - (scale\_pattern\_colour\_gradient),
  - [30](#)
- scale\_pattern\_color\_gradientn
  - (scale\_pattern\_colour\_gradient),
  - [30](#)
- scale\_pattern\_color\_grey
  - (scale\_pattern\_colour\_grey), [33](#)
- scale\_pattern\_color\_hue
  - (scale\_pattern\_colour\_hue), [34](#)
- scale\_pattern\_color\_identity
  - (scale\_pattern\_identity), [38](#)
- scale\_pattern\_color\_manual
  - (scale\_pattern\_manual), [41](#)
- scale\_pattern\_color\_ordinal
  - (scale\_pattern\_colour\_viridis\_d),
  - [36](#)
- scale\_pattern\_color\_viridis\_c
  - (scale\_pattern\_colour\_viridis\_d),
  - [36](#)



- scale\_pattern\_color\_viridis\_d  
(scale\_pattern\_colour\_viridis\_d),  
36
- scale\_pattern\_colour\_brewer, 26
- scale\_pattern\_colour\_continuous, 29
- scale\_pattern\_colour\_discrete  
(scale\_pattern\_colour\_hue), 34
- scale\_pattern\_colour\_distiller  
(scale\_pattern\_colour\_brewer),  
26
- scale\_pattern\_colour\_gradient, 30
- scale\_pattern\_colour\_gradient(), 33
- scale\_pattern\_colour\_gradient2  
(scale\_pattern\_colour\_gradient),  
30
- scale\_pattern\_colour\_gradientn  
(scale\_pattern\_colour\_gradient),  
30
- scale\_pattern\_colour\_grey, 33
- scale\_pattern\_colour\_hue, 34
- scale\_pattern\_colour\_identity  
(scale\_pattern\_identity), 38
- scale\_pattern\_colour\_manual  
(scale\_pattern\_manual), 41
- scale\_pattern\_colour\_ordinal  
(scale\_pattern\_colour\_viridis\_d),  
36
- scale\_pattern\_colour\_viridis\_c  
(scale\_pattern\_colour\_viridis\_d),  
36
- scale\_pattern\_colour\_viridis\_d, 36
- scale\_pattern\_continuous  
(scale\_discrete), 21
- scale\_pattern\_density\_continuous  
(scale\_continuous), 16
- scale\_pattern\_density\_discrete  
(scale\_continuous), 16
- scale\_pattern\_density\_identity  
(scale\_pattern\_identity), 38
- scale\_pattern\_density\_manual  
(scale\_pattern\_manual), 41
- scale\_pattern\_discrete  
(scale\_discrete), 21
- scale\_pattern\_filename\_continuous  
(scale\_discrete), 21
- scale\_pattern\_filename\_discrete  
(scale\_discrete), 21
- scale\_pattern\_filename\_identity  
(scale\_pattern\_identity), 38
- scale\_pattern\_filename\_manual  
(scale\_pattern\_manual), 41
- scale\_pattern\_fill2\_brewer  
(scale\_pattern\_colour\_brewer),  
26
- scale\_pattern\_fill2\_continuous  
(scale\_pattern\_colour\_continuous),  
29
- scale\_pattern\_fill2\_discrete  
(scale\_pattern\_colour\_hue), 34
- scale\_pattern\_fill2\_distiller  
(scale\_pattern\_colour\_brewer),  
26
- scale\_pattern\_fill2\_gradient  
(scale\_pattern\_colour\_gradient),  
30
- scale\_pattern\_fill2\_gradient2  
(scale\_pattern\_colour\_gradient),  
30
- scale\_pattern\_fill2\_gradientn  
(scale\_pattern\_colour\_gradient),  
30
- scale\_pattern\_fill2\_grey  
(scale\_pattern\_colour\_grey), 33
- scale\_pattern\_fill2\_hue  
(scale\_pattern\_colour\_hue), 34
- scale\_pattern\_fill2\_identity  
(scale\_pattern\_identity), 38
- scale\_pattern\_fill2\_manual  
(scale\_pattern\_manual), 41
- scale\_pattern\_fill2\_ordinal  
(scale\_pattern\_colour\_viridis\_d),  
36
- scale\_pattern\_fill2\_viridis\_c  
(scale\_pattern\_colour\_viridis\_d),  
36
- scale\_pattern\_fill2\_viridis\_d  
(scale\_pattern\_colour\_viridis\_d),  
36
- scale\_pattern\_fill\_brewer  
(scale\_pattern\_colour\_brewer),  
26
- scale\_pattern\_fill\_continuous  
(scale\_pattern\_colour\_continuous),  
29
- scale\_pattern\_fill\_discrete  
(scale\_pattern\_colour\_hue), 34

- scale\_pattern\_fill\_distiller  
(scale\_pattern\_colour\_brewer),  
26
- scale\_pattern\_fill\_gradient  
(scale\_pattern\_colour\_gradient),  
30
- scale\_pattern\_fill\_gradient2  
(scale\_pattern\_colour\_gradient),  
30
- scale\_pattern\_fill\_gradientn  
(scale\_pattern\_colour\_gradient),  
30
- scale\_pattern\_fill\_grey  
(scale\_pattern\_colour\_grey), 33
- scale\_pattern\_fill\_hue  
(scale\_pattern\_colour\_hue), 34
- scale\_pattern\_fill\_identity  
(scale\_pattern\_identity), 38
- scale\_pattern\_fill\_manual  
(scale\_pattern\_manual), 41
- scale\_pattern\_fill\_ordinal  
(scale\_pattern\_colour\_viridis\_d),  
36
- scale\_pattern\_fill\_viridis\_c  
(scale\_pattern\_colour\_viridis\_d),  
36
- scale\_pattern\_fill\_viridis\_d  
(scale\_pattern\_colour\_viridis\_d),  
36
- scale\_pattern\_filter\_continuous  
(scale\_discrete), 21
- scale\_pattern\_filter\_discrete  
(scale\_discrete), 21
- scale\_pattern\_filter\_identity  
(scale\_pattern\_identity), 38
- scale\_pattern\_filter\_manual  
(scale\_pattern\_manual), 41
- scale\_pattern\_frequency\_continuous  
(scale\_continuous), 16
- scale\_pattern\_frequency\_discrete  
(scale\_continuous), 16
- scale\_pattern\_frequency\_identity  
(scale\_pattern\_identity), 38
- scale\_pattern\_frequency\_manual  
(scale\_pattern\_manual), 41
- scale\_pattern\_gravity\_continuous  
(scale\_discrete), 21
- scale\_pattern\_gravity\_discrete  
(scale\_discrete), 21
- scale\_pattern\_gravity\_identity  
(scale\_pattern\_identity), 38
- scale\_pattern\_gravity\_manual  
(scale\_pattern\_manual), 41
- scale\_pattern\_grid\_continuous  
(scale\_discrete), 21
- scale\_pattern\_grid\_discrete  
(scale\_discrete), 21
- scale\_pattern\_grid\_identity  
(scale\_pattern\_identity), 38
- scale\_pattern\_grid\_manual  
(scale\_pattern\_manual), 41
- scale\_pattern\_identity, 38
- scale\_pattern\_key\_scale\_factor\_continuous  
(scale\_continuous), 16
- scale\_pattern\_key\_scale\_factor\_discrete  
(scale\_continuous), 16
- scale\_pattern\_key\_scale\_factor\_identity  
(scale\_pattern\_identity), 38
- scale\_pattern\_key\_scale\_factor\_manual  
(scale\_pattern\_manual), 41
- scale\_pattern\_linetype, 40
- scale\_pattern\_linetype\_continuous  
(scale\_pattern\_linetype), 40
- scale\_pattern\_linetype\_discrete  
(scale\_pattern\_linetype), 40
- scale\_pattern\_linetype\_identity  
(scale\_pattern\_identity), 38
- scale\_pattern\_linetype\_manual  
(scale\_pattern\_manual), 41
- scale\_pattern\_manual, 41
- scale\_pattern\_orientation\_continuous  
(scale\_discrete), 21
- scale\_pattern\_orientation\_discrete  
(scale\_discrete), 21
- scale\_pattern\_orientation\_identity  
(scale\_pattern\_identity), 38
- scale\_pattern\_orientation\_manual  
(scale\_pattern\_manual), 41
- scale\_pattern\_phase\_continuous  
(scale\_continuous), 16
- scale\_pattern\_phase\_discrete  
(scale\_continuous), 16
- scale\_pattern\_phase\_identity  
(scale\_pattern\_identity), 38
- scale\_pattern\_phase\_manual  
(scale\_pattern\_manual), 41

- scale\_pattern\_res\_continuous  
(scale\_continuous), 16
- scale\_pattern\_res\_discrete  
(scale\_continuous), 16
- scale\_pattern\_res\_identity  
(scale\_pattern\_identity), 38
- scale\_pattern\_res\_manual  
(scale\_pattern\_manual), 41
- scale\_pattern\_rot\_continuous  
(scale\_continuous), 16
- scale\_pattern\_rot\_discrete  
(scale\_continuous), 16
- scale\_pattern\_rot\_identity  
(scale\_pattern\_identity), 38
- scale\_pattern\_rot\_manual  
(scale\_pattern\_manual), 41
- scale\_pattern\_scale\_continuous  
(scale\_continuous), 16
- scale\_pattern\_scale\_discrete  
(scale\_continuous), 16
- scale\_pattern\_scale\_identity  
(scale\_pattern\_identity), 38
- scale\_pattern\_scale\_manual  
(scale\_pattern\_manual), 41
- scale\_pattern\_shape, 43
- scale\_pattern\_shape\_continuous  
(scale\_pattern\_shape), 43
- scale\_pattern\_shape\_discrete  
(scale\_pattern\_shape), 43
- scale\_pattern\_shape\_identity  
(scale\_pattern\_identity), 38
- scale\_pattern\_shape\_manual  
(scale\_pattern\_manual), 41
- scale\_pattern\_shape\_manual(), 43
- scale\_pattern\_shape\_ordinal  
(scale\_pattern\_shape), 43
- scale\_pattern\_size  
(scale\_pattern\_size\_continuous),  
45
- scale\_pattern\_size\_continuous, 45
- scale\_pattern\_size\_discrete  
(scale\_pattern\_size\_continuous),  
45
- scale\_pattern\_size\_identity  
(scale\_pattern\_identity), 38
- scale\_pattern\_size\_manual  
(scale\_pattern\_manual), 41
- scale\_pattern\_size\_ordinal  
(scale\_pattern\_size\_continuous),  
45
- scale\_pattern\_spacing\_continuous  
(scale\_continuous), 16
- scale\_pattern\_spacing\_discrete  
(scale\_continuous), 16
- scale\_pattern\_spacing\_identity  
(scale\_pattern\_identity), 38
- scale\_pattern\_spacing\_manual  
(scale\_pattern\_manual), 41
- scale\_pattern\_subtype\_continuous  
(scale\_discrete), 21
- scale\_pattern\_subtype\_discrete  
(scale\_discrete), 21
- scale\_pattern\_subtype\_identity  
(scale\_pattern\_identity), 38
- scale\_pattern\_subtype\_manual  
(scale\_pattern\_manual), 41
- scale\_pattern\_type\_continuous  
(scale\_discrete), 21
- scale\_pattern\_type\_discrete  
(scale\_discrete), 21
- scale\_pattern\_type\_identity  
(scale\_pattern\_identity), 38
- scale\_pattern\_type\_manual  
(scale\_pattern\_manual), 41
- scale\_pattern\_units\_continuous  
(scale\_discrete), 21
- scale\_pattern\_units\_discrete  
(scale\_discrete), 21
- scale\_pattern\_units\_identity  
(scale\_pattern\_identity), 38
- scale\_pattern\_units\_manual  
(scale\_pattern\_manual), 41
- scale\_pattern\_xoffset\_continuous  
(scale\_continuous), 16
- scale\_pattern\_xoffset\_discrete  
(scale\_continuous), 16
- scale\_pattern\_xoffset\_identity  
(scale\_pattern\_identity), 38
- scale\_pattern\_xoffset\_manual  
(scale\_pattern\_manual), 41
- scale\_pattern\_yoffset\_continuous  
(scale\_continuous), 16
- scale\_pattern\_yoffset\_discrete  
(scale\_continuous), 16
- scale\_pattern\_yoffset\_identity  
(scale\_pattern\_identity), 38

scale\_pattern\_yoffset\_manual  
(scale\_pattern\_manual), [41](#)