

# Package: ggshadow (via r-universe)

October 15, 2024

**Title** Shadow and Glow Geoms for 'ggplot2'

**Version** 0.0.4

**Description** A collection of Geoms for R's 'ggplot2' library. GeomShadowLine, GeomShadowPath, GeomShadowPoint, and GeomShadowStep support drawing a shadow below lines or around points to make busy plots more aesthetically pleasing. GeomGlowLine, GeomGlowPath, GeomGlowPoint, and GeomGlowStep support adding neon glow around lines or points to get a steampunk style.

**License** GPL-2

**URL** <https://github.com/marcmenem/ggshadow/>

**BugReports** <https://github.com/marcmenem/ggshadow/issues>

**Depends** R (>= 3.4.0)

**Imports** cli, ggplot2 (>= 3.4.2), grid, lifecycle, rlang, scales, stats, vctrs

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

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.1

**Collate** 'geom-glowpath.r' 'geom-glowpoint.r' 'geom-shadowpath.r' 'geom-shadowpoint.r' 'ggplot2-utils.R' 'ggshadow-package.R' 'internal-doc.r' 'scale-shadow.r'

**Repository** <https://marcmenem.r-universe.dev>

**RemoteUrl** <https://github.com/marcmenem/ggshadow>

**RemoteRef** HEAD

**RemoteSha** 3240bf4cf71bfd59cb2b65eccb7fd4bf7b7d21d6

## Contents

|                              |    |
|------------------------------|----|
| geom_glowpath . . . . .      | 2  |
| geom_glowpoint . . . . .     | 5  |
| geom_shadowpath . . . . .    | 6  |
| geom_shadowpoint . . . . .   | 9  |
| scale_brewer . . . . .       | 11 |
| scale_colour_hue . . . . .   | 13 |
| scale_colour_steps . . . . . | 16 |
| scale_continuous . . . . .   | 19 |
| scale_gradient . . . . .     | 20 |
| scale_grey . . . . .         | 24 |
| scale_identity . . . . .     | 26 |
| scale_manual . . . . .       | 27 |
| scale_viridis . . . . .      | 29 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>33</b> |
|--------------|-----------|

---

|               |                             |
|---------------|-----------------------------|
| geom_glowpath | <i>Connect Observations</i> |
|---------------|-----------------------------|

---

### Description

Plot a glow beneath the connected lines to make it easier to read a chart with several overlapping observations. `geom_glowpath()` connects the observations in the order in which they appear in the data. `geom_glowline()` connects them in order of the variable on the x axis. `geom_glowstep()` creates a stairstep plot, highlighting exactly when changes occur.

### Usage

```
geom_glowpath(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
  arrow = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_glowline(
  mapping = NULL,
  data = NULL,
```

```

    stat = "identity",
    position = "identity",
    na.rm = FALSE,
    orientation = NA,
    show.legend = NA,
    inherit.aes = TRUE,
    ...
  )

geom_glowstep(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  direction = "hv",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)

```

## Arguments

|           |  |
|-----------|--|
| mapping   | Set of aesthetic mappings created by <code>aes()</code> or <code>aes_()</code> . 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 <code>ggplot()</code>.</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 <code>fortify()</code> 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(., 10)</code>).</p> |
| stat      | The statistical transformation to use on the data for this layer, as a string.   |
| position  | Position adjustment, either as a string, or the result of a call to a position adjustment function.  |
| ...       | Other arguments passed on to <code>layer()</code> . These are often aesthetics, used to set an aesthetic to a fixed value, like <code>colour = "red"</code> or <code>linewidth = 3</code> . They may also be parameters to the paired <code>geom/stat</code> .   |
| lineend   | Line end style (round, butt, square).  |
| linejoin  | Line join style (round, mitre, bevel).   |
| linemitre | Line mitre limit (number greater than 1).  |
| arrow     | Arrow specification, as created by <code>grid::arrow()</code> .  |

|             |   |
|-------------|---|
| na.rm       | If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.   |
| show.legend | logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.                                  |
| inherit.aes | If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. <code>borders()</code> .                     |
| orientation | The orientation of the layer. The default (NA) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting orientation to either "x" or "y". See the <i>Orientation</i> section for more detail. |
| direction   | direction of stairs: 'vh' for vertical then horizontal, 'hv' for horizontal then vertical, or 'mid' for step half-way between adjacent x-values.  |

### Details

The group aesthetic determines which cases are connected together. These functions are designed as a straight replacement to the `geom_path()`, `geom_line()` and `geom_step()` functions. To set the order of drawing, make the colour aesthetic a factor, and set the order from bottom to top.

### Value

a ggplot2 layer to add to a plot.

### Functions

- `geom_glowpath()`: Connects observations in the order in which they appear in the data.
- `geom_glowline()`: Connects observations in order of the variable on the x axis.
- `geom_glowstep()`: Creates a stairstep plot, highlighting exactly when changes occur.

### Missing value handling

`geom_glowpath()`, `geom_glowline()`, and `geom_glowstep()` handle NA as follows:

- If an NA occurs in the middle of a line, it breaks the line. No warning is shown, regardless of whether `na.rm` is TRUE or FALSE.
- If an NA occurs at the start or the end of the line and `na.rm` is FALSE (default), the NA is removed with a warning.
- If an NA occurs at the start or the end of the line and `na.rm` is TRUE, the NA is removed silently, without warning.

### Aesthetics

Adds 3 new aesthetics to `geom_path()`:

- `shadowcolour` defaults to path color, controls the color of the shadow.
- `shadowlinewidth` defaults to `linewidth`, controls the linewidth of the shadow.
- `shadowalpha` defaults to  $0.06 * \alpha$  or  $0.06$ , controls the alpha of the glow.

**See Also**

[ggplot2::geom\\_path\(\)](#), [ggplot2::geom\\_line\(\)](#), [ggplot2::geom\\_step\(\)](#): Filled paths (polygons);

**Examples**

```
# geom_glowline() is suitable for time series
library(ggplot2)
ggplot(economics_long, aes(date, value01, colour = variable)) + geom_glowline()
```

---

|                |               |
|----------------|---------------|
| geom_glowpoint | <i>Points</i> |
|----------------|---------------|

---

**Description**

The point geom is used to create scatterplots. [geom\\_glowpoint\(\)](#) is designed as a drop in replacement for [geom\\_point\(\)](#) with an added glow beneath the point to make a busy plot more aesthetically appealing or to make points stand out from the rest of the plot.

**Usage**

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

**Arguments**

|         |   |
|---------|---|
| mapping | Set of aesthetic mappings created by <a href="#">aes()</a> or <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 <code>mapping</code> 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, as a string.  |
| position    | Position adjustment, either as a string, or the result of a call to a position adjustment function.   |
| ...         | Other arguments passed on to <code>layer()</code> . These are often aesthetics, used to set an aesthetic to a fixed value, like <code>colour = "red"</code> or <code>size = 3</code> . They may also be parameters to the paired geom/stat.                   |
| na.rm       | If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.   |
| show.legend | logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.              |
| inherit.aes | If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. <code>borders()</code> . |

### Value

a layer to add to a plot.

### Aesthetics

Adds 3 new aesthetics to `geom_point()`:

- `shadowcolour` defaults to the same color as the point, controls the color of the glow
- `shadowsize` defaults to `size`, controls the size of the shadow.
- `shadowalpha` defaults to  $0.06 * \alpha$  or  $0.06$ , controls the alpha of the glow

### Examples

```
library( ggplot2 )
p <- ggplot(mtcars, aes(wt, mpg))
p + geom_shadowpoint()
```

---

geom\_shadowpath

*Connect Observations*

---

### Description

Plot a shadow beneath the connected lines to make it easier to read a chart with several overlapping observations. `geom_shadowpath()` connects the observations in the order in which they appear in the data. `geom_shadowline()` connects them in order of the variable on the x axis. `geom_shadowstep()` creates a `geom_step` plot, highlighting exactly when changes occur.

**Usage**

```
geom_shadowpath(  
  mapping = NULL,  
  data = NULL,  
  stat = "identity",  
  position = "identity",  
  ...,  
  lineend = "butt",  
  linejoin = "round",  
  linemitre = 10,  
  arrow = NULL,  
  na.rm = FALSE,  
  show.legend = NA,  
  inherit.aes = TRUE  
)
```

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

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

**Arguments**

|         |   |
|---------|---|
| mapping | Set of aesthetic mappings created by <code>aes()</code> or <code>aes_()</code> . 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    | The data to be displayed in this layer. There are three options:<br>If <code>NULL</code> , the default, the data is inherited from the plot data as specified in the call to <code>ggplot()</code> .  |

A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.

A function will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A function can be created from a formula (e.g. `~ head(.x, 10)`).

|             |   |
|-------------|---|
| stat        | The statistical transformation to use on the data for this layer, as a string.  |
| position    | Position adjustment, either as a string, or the result of a call to a position adjustment function.   |
| ...         | Other arguments passed on to <code>layer()</code> . These are often aesthetics, used to set an aesthetic to a fixed value, like <code>colour = "red"</code> or <code>linewidth = 3</code> . They may also be parameters to the paired geom/stat.                                  |
| lineend     | Line end style (round, butt, square).   |
| linejoin    | Line join style (round, mitre, bevel).  |
| linemitre   | Line mitre limit (number greater than 1).   |
| arrow       | Arrow specification, as created by <code>grid::arrow()</code> .   |
| na.rm       | If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.   |
| show.legend | logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.                                  |
| inherit.aes | If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. <code>borders()</code> .                     |
| orientation | The orientation of the layer. The default (NA) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting orientation to either "x" or "y". See the <i>Orientation</i> section for more detail. |
| direction   | direction of stairs: 'vh' for vertical then horizontal, 'hv' for horizontal then vertical, or 'mid' for step half-way between adjacent x-values.  |

### Details

The group aesthetic determines which cases are connected together. These functions are designed as a straight replacement to the `geom_path()`, `geom_line()` and `geom_step()` functions. To set the order of drawing, make the colour aesthetic a factor, and set the order from bottom to top.

### Value

a layer to add to a plot.

### Functions

- `geom_shadowpath()`: Connects observations in the order in which they appear in the data.
- `geom_shadowline()`: Connects observations in order of the variable on the x axis.
- `geom_shadowstep()`: Creates a staircase plot, highlighting exactly when changes occur.



### Missing value handling

geom\_shadowpath(), geom\_shadowline(), and geom\_shadowstep() handle NA as follows:

- If an NA occurs in the middle of a line, it breaks the line. No warning is shown, regardless of whether na.rm is TRUE or FALSE.
- If an NA occurs at the start or the end of the line and na.rm is FALSE (default), the NA is removed with a warning.
- If an NA occurs at the start or the end of the line and na.rm is TRUE, the NA is removed silently, without warning.

### Aesthetics

Adds 3 new aesthetics to `geom_path()`:

- shadowcolour defaults to white, controls the color of the shadow.
- shadowlinewidth defaults to  $2.5 * linewidth$ , controls the linewidth of the shadow.
- shadowalpha defaults to  $0.25 * alpha$  or  $0.9$ , controls the alpha of the shadow.

### See Also

`ggplot2::geom_path()`, `ggplot2::geom_line()`, `ggplot2::geom_step()`: Filled paths (polygons);

### Examples

```
# geom_shadowline() is suitable for time series
library(ggplot2)
ggplot(economics_long, aes(date, value01, colour = variable)) + geom_shadowline()

ggplot(economics_long, aes(date, value01, colour = value01,
                           group = variable, alpha=date, shadowalpha=1)) +
  geom_shadowline()
```

---

geom\_shadowpoint      *Points*

---

### Description

The point geom is used to create scatterplots. `geom_shadowpoint()` is designed as a drop in replacement for `geom_point()` with an added shadow beneath the point to make a busy plot more aesthetically appealing or to make points stand out from the rest of the plot.

**Usage**

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

**Arguments**

|             |   |
|-------------|---|
| mapping     | Set of aesthetic mappings created by <a href="#">aes()</a> or <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        | The data to be displayed in this layer. There are three options:<br>If <code>NULL</code> , the default, the data is inherited from the plot data as specified in the call to <a href="#">ggplot()</a> .<br>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.<br>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> ). |
| stat        | The statistical transformation to use on the data for this layer, as a string.  |
| position    | Position adjustment, either as a string, or the result of a call to a position adjustment function.   |
| ...         | Other arguments passed on to <a href="#">layer()</a> . These are often aesthetics, used to set an aesthetic to a fixed value, like <code>colour = "red"</code> or <code>size = 3</code> . They may also be parameters to the paired <code>geom/stat</code> .  |
| na.rm       | If <code>FALSE</code> , the default, missing values are removed with a warning. If <code>TRUE</code> , missing values are silently removed.   |
| show.legend | logical. Should this layer be included in the legends? <code>NA</code> , the default, includes if any aesthetics are mapped. <code>FALSE</code> never includes, and <code>TRUE</code> always includes. It can also be a named logical vector to finely select the aesthetics to display.  |
| inherit.aes | If <code>FALSE</code> , 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. <a href="#">borders()</a> .  |

**Value**

a layer to add to a plot.

## Aesthetics

Adds 3 new aesthetics to `geom_point()`:

- `shadowcolour` defaults to white, controls the color of the shadow.
- `shadowsize` defaults to  $1.8 * \text{size}$ , controls the size of the shadow.
- `shadowalpha` defaults to  $0.25 * \text{alpha}$  or  $0.9$ , controls the alpha of the shadow.

## Examples

```
library( ggplot2 )
p <- ggplot(mtcars, aes(wt, mpg))
p + geom_shadowpoint()
```

---

|              |   |
|--------------|---|
| scale_brewer | <i>Sequential, diverging and qualitative colour scales from colorbrewer.org</i> |
|--------------|---|

---

## 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_shadowcolour_brewer(
  ...,
  type = "seq",
  palette = 1,
  direction = 1,
  aesthetics = "shadowcolour"
)

scale_shadowcolour_distiller(
  ...,
  type = "seq",
  palette = 1,
  direction = -1,
  values = NULL,
  space = "Lab",
  na.value = "grey50",
  guide = "colourbar",
  aesthetics = "shadowcolour"
)

scale_shadowcolour_fermenter(
```

```

...,
type = "seq",
palette = 1,
direction = -1,
na.value = "grey50",
guide = "coloursteps",
aesthetics = "shadowcolour"
)

```

### Arguments

|            |   |
|------------|---|
| ...        | Other arguments passed on to <a href="#">discrete_scale()</a> , <a href="#">continuous_scale()</a> , or <a href="#">binned_scale()</a> , for brewer, distiller, and fermenter variants respectively, to control name, limits, breaks, labels and so forth.                            |
| type       | One of "seq" (sequential), "div" (diverging) or "qual" (qualitative)  |
| 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  | Sets the order of colours in the scale. If 1, the default, colours are as output by <a href="#">RColorBrewer::brewer.pal()</a> . If -1, the order of colours is reversed.   |
| 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> . |
| 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.                          |
| space      | colour space in which to calculate gradient. Must be "Lab" - other values are deprecated.   |
| na.value   | Colour to use for missing values  |
| guide      | Type of legend. Use "colourbar" for continuous colour bar, or "legend" for discrete colour legend.  |

### 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 scale object to add to a plot.

### 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.

### See Also

Other colour scales: [scale\\_colour\\_hue](#), [scale\\_colour\\_steps](#), [scale\\_gradient](#), [scale\\_grey](#), [scale\\_viridis](#)

### Examples

```
library( ggplot2 )
p <- ggplot(mtcars, aes(wt, mpg, shadowcolor=as.factor(gear)))
p + geom_shadowpoint() + scale_shadowcolour_brewer()
```

```
library( ggplot2 )
p <- ggplot(mtcars, aes(wt, mpg, shadowcolor=gear))
p + geom_shadowpoint() + scale_shadowcolour_distiller() + guides(shadowcolor='none')
```

```
library( ggplot2 )
p <- ggplot(mtcars, aes(wt, mpg, shadowcolor=as.factor(gear)))
p + geom_shadowpoint() + scale_shadowcolour_brewer()
```

---

scale\_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_shadowcolour_hue(
  ...,
  h = c(0, 360) + 15,
  c = 100,
```

```

    l = 65,
    h.start = 0,
    direction = 1,
    na.value = "grey50",
    aesthetics = "shadowcolour"
  )

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

```

## Arguments

... Arguments passed on to [ggplot2::discrete\\_scale](#)

**scale\_name** **[Deprecated]** The name of the scale that should be used for error messages associated with this scale.

**palette** A palette function that when called with a single integer argument (the number of levels in the scale) returns the values that they should take (e.g., [scales::pal\\_hue\(\)](#)).

**name** The name of the scale. Used as the axis or legend title. If [waiver\(\)](#), the default, the name of the scale is taken from the first mapping used for that aesthetic. If `NULL`, the legend title will be omitted.

**breaks** One of:

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

**labels** One of:

- `NULL` for no labels
- [waiver\(\)](#) for the default labels computed by the transformation object
- A character vector giving labels (must be same length as breaks)
- An expression vector (must be the same length as breaks). See [?plot-math](#) for details.
- A function that takes the breaks as input and returns labels as output. Also accepts rlang [lambda](#) function notation.

**limits** One of:

- `NULL` to use the default scale values
- A character vector that defines possible values of the scale and their order

- A function that accepts the existing (automatic) values and returns new ones. Also accepts rlang [lambda](#) function notation.

**expand** For position scales, a vector of range expansion constants used to add some padding around the data to ensure that they are placed some distance away from the axes. Use the convenience function [expansion\(\)](#) to generate the values for the expand argument. The defaults are to expand the scale by 5% on each side for continuous variables, and by 0.6 units on each side for discrete variables.

**na.translate** Unlike continuous scales, discrete scales can easily show missing values, and do so by default. If you want to remove missing values from a discrete scale, specify `na.translate = FALSE`.

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

**guide** A function used to create a guide or its name. See [guides\(\)](#) for more information.

**position** For position scales, The position of the axis. `left` or `right` for y axes, `top` or `bottom` for x axes.

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

**h** range of hues to use, in `[0, 360]`

**c** chroma (intensity of colour), maximum value varies depending on combination of hue and luminance.

**l** luminance (lightness), in `[0, 100]`

**h.start** hue to start at

**direction** direction to travel around the colour wheel, `1` = clockwise, `-1` = counter-clockwise

**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 `aesthetics = c("colour", "fill")`.

**Value**

a scale object to add to a plot.

**See Also**

Other colour scales: [scale\\_brewer](#), [scale\\_colour\\_steps](#), [scale\\_gradient](#), [scale\\_grey](#), [scale\\_viridis](#)

**Examples**

```
library( ggplot2 )
p <- ggplot(mtcars, aes(wt, mpg, shadowcolor=as.factor(gear)))
p + geom_shadowpoint() + scale_shadowcolour_hue()

library( ggplot2 )
p <- ggplot(mtcars, aes(wt, mpg, shadowcolor=as.factor(gear)))
p + geom_shadowpoint() + scale_shadowcolour_discrete()
```

---

scale\_colour\_steps     *Binned gradient colour scales*

---

### Description

scale\_\*\_steps creates a two colour binned gradient (low-high), scale\_\*\_steps2 creates a diverging binned colour gradient (low-mid-high), and scale\_\*\_stepsn creates a n-colour binned gradient. These scales are binned variants of the [gradient scale](#) family and works in the same way.

### Usage

```
scale_shadowcolour_steps(  
  ...,  
  low = "#132B43",  
  high = "#56B1F7",  
  space = "Lab",  
  na.value = "grey50",  
  guide = "coloursteps",  
  aesthetics = "shadowcolour"  
)  
  
scale_shadowcolour_steps2(  
  ...,  
  low = muted("red"),  
  mid = "white",  
  high = muted("blue"),  
  midpoint = 0,  
  space = "Lab",  
  na.value = "grey50",  
  guide = "coloursteps",  
  aesthetics = "shadowcolour"  
)  
  
scale_shadowcolour_stepsn(  
  ...,  
  colours,  
  values = NULL,  
  space = "Lab",  
  na.value = "grey50",  
  guide = "coloursteps",  
  aesthetics = "shadowcolour",  
  colors  
)
```

### Arguments

...     Arguments passed on to [ggplot2::binned\\_scale](#)



**name** The name of the scale. Used as the axis or legend title. If `waiver()`, the default, the name of the scale is taken from the first mapping used for that aesthetic. If `NULL`, the legend title will be omitted.

**breaks** One of:

- `NULL` for no breaks
- `waiver()` for the default breaks computed by the [transformation object](#)
- A numeric vector of positions
- A function that takes the limits as input and returns breaks as output (e.g., a function returned by `scales::extended_breaks()`). Also accepts rlang [lambda](#) function notation.

**labels** One of:

- `NULL` for no labels
- `waiver()` for the default labels computed by the transformation object
- A character vector giving labels (must be same length as breaks)
- An expression vector (must be the same length as breaks). See `?plot-math` for details.
- A function that takes the breaks as input and returns labels as output. Also accepts rlang [lambda](#) function notation.

**limits** One of:

- `NULL` to use the default scale range
- A numeric vector of length two providing limits of the scale. Use `NA` to refer to the existing minimum or maximum
- A function that accepts the existing (automatic) limits and returns new limits. Also accepts rlang [lambda](#) function notation. Note that setting limits on positional scales will **remove** data outside of the limits. If the purpose is to zoom, use the `limit` argument in the coordinate system (see `coord_cartesian()`).

**oob** One of:

- Function that handles limits outside of the scale limits (out of bounds). Also accepts rlang [lambda](#) function notation.
- The default (`scales::censor()`) replaces out of bounds values with `NA`.
- `scales::squish()` for squishing out of bounds values into range.
- `scales::squish_infinite()` for squishing infinite values into range.

**expand** For position scales, a vector of range expansion constants used to add some padding around the data to ensure that they are placed some distance away from the axes. Use the convenience function `expansion()` to generate the values for the `expand` argument. The defaults are to expand the scale by 5% on each side for continuous variables, and by 0.6 units on each side for discrete variables.

**n.breaks** The number of break points to create if breaks are not given directly.

**nice.breaks** Logical. Should breaks be attempted placed at nice values instead of exactly evenly spaced between the limits. If `TRUE` (default) the scale will ask the transformation object to create breaks, and this may result in a different number of breaks than requested. Ignored if breaks are given explicitly.

|  |                              |   |
|--|------------------------------|---|
|  | <code>right</code>           | Should the intervals be closed on the right (TRUE, default) or should the intervals be closed on the left (FALSE)? 'Closed on the right' means that values at break positions are part of the lower bin (open on the left), whereas they are part of the upper bin when intervals are closed on the left (open on the right). |
|  | <code>trans</code>           | <b>[Deprecated]</b> Deprecated in favour of <code>transform</code> .  |
|  | <code>show.limits</code>     | should the limits of the scale appear as ticks  |
|  | <code>position</code>        | For position scales, The position of the axis. <code>left</code> or <code>right</code> for y axes, <code>top</code> or <code>bottom</code> for x axes.  |
|  | <code>super</code>           | The super class to use for the constructed scale  |
|  | <code>low, high</code>       | Colours for low and high ends of the gradient.  |
|  | <code>space</code>           | colour space in which to calculate gradient. Must be "Lab" - other values are deprecated.   |
|  | <code>na.value</code>        | Colour to use for missing values  |
|  | <code>guide</code>           | Type of legend. Use "colourbar" for continuous colour bar, or "legend" for discrete colour legend.  |
|  | <code>aesthetics</code>      | 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 <code>colour</code> and <code>fill</code> aesthetics at the same time, via <code>aesthetics = c("colour", "fill")</code> .               |
|  | <code>mid</code>             | colour for mid point  |
|  | <code>midpoint</code>        | The midpoint (in data value) of the diverging scale. Defaults to 0.   |
|  | <code>colours, colors</code> | Vector of colours to use for n-colour gradient.   |
|  | <code>values</code>          | if colours should not be evenly positioned along the gradient this vector gives the position (between 0 and 1) for each colour in the <code>colours</code> vector. See <a href="#">rescale()</a> for a convenience function to map an arbitrary range to between 0 and 1.   |

## Details

Default colours are generated with `munsell` and `mns1(c("2.5PB 2/4", "2.5PB 7/10"))`. Generally, for continuous colour scales you want to keep hue constant, but vary chroma and luminance. The `munsell` package makes this easy to do using the Munsell colour system.

## Value

a scale object to add to a plot.

## See Also

[scales::seq\\_gradient\\_pal\(\)](#) for details on underlying palette

Other colour scales: [scale\\_brewer](#), [scale\\_colour\\_hue](#), [scale\\_gradient](#), [scale\\_grey](#), [scale\\_viridis](#)

**Examples**

```

library( ggplot2 )
p <- ggplot(mtcars, aes(wt, mpg, shadowcolor=gear))
p + geom_shadowpoint() + scale_shadowcolour_steps() + guides(shadowcolour='none')

library( ggplot2 )
p <- ggplot(mtcars, aes(wt, mpg, shadowcolor=gear))
p + geom_shadowpoint() + scale_shadowcolour_steps2() + guides(shadowcolour='none')

library( ggplot2 )
p <- ggplot(mtcars, aes(wt, mpg, shadowcolor=gear))
p <- p + geom_shadowpoint() + scale_shadowcolour_stepsn(colours=c('red', 'yellow'))
p + guides(shadowcolour='none')

```

---

scale\_continuous

*Continuous and binned colour scales*


---

**Description**

Colour scales for continuous data default to the values of the `ggplot2.continuous.colour` and `ggplot2.continuous.fill` options. These `options()` default to "gradient" (i.e., `scale_colour_gradient()` and `scale_fill_gradient()`)

**Usage**

```

scale_shadowcolour_continuous(
  ...,
  type = getOption("ggplot2.continuous.colour", default = "gradient")
)

scale_shadowcolour_binned(
  ...,
  type = getOption("ggplot2.binned.colour", default =
    getOption("ggplot2.continuous.colour", default = "gradient"))
)

```

**Arguments**

... Additional parameters passed on to the scale type

type One of the following:

- "gradient" (the default)
- "viridis"
- A function that returns a continuous colour scale.

**Value**

a scale object to add to a plot.

## Color Blindness

Many color palettes derived from RGB combinations (like the "rainbow" color palette) are not suitable to support all viewers, especially those with color vision deficiencies. Using `viridis` type, which is perceptually uniform in both colour and black-and-white display is an easy option to ensure good perceptive properties of your visualizations. The `colorspace` package offers functionalities

- to generate color palettes with good perceptive properties,
- to analyse a given color palette, like emulating color blindness,
- and to modify a given color palette for better perceptivity.

For more information on color vision deficiencies and suitable color choices see the [paper on the colorspace package](#) and references therein.

## See Also

[scale\\_colour\\_gradient\(\)](#), [scale\\_colour\\_viridis\\_c\(\)](#), [scale\\_colour\\_steps\(\)](#), [scale\\_colour\\_viridis\\_b\(\)](#), [scale\\_fill\\_gradient\(\)](#), [scale\\_fill\\_viridis\\_c\(\)](#), [scale\\_fill\\_steps\(\)](#), and [scale\\_fill\\_viridis\\_b\(\)](#)

## Examples

```
library( ggplot2 )
p <- ggplot(mtcars, aes(wt, mpg, shadowcolor=gear))
p + geom_shadowpoint() + scale_shadowcolour_continuous() + guides(shadowcolour='none')
```

```
library( ggplot2 )
p <- ggplot(mtcars, aes(wt, mpg, shadowcolor=gear))
p + geom_shadowpoint() + scale_shadowcolour_binned() + guides(shadowcolour='none')
```

---

scale\_gradient

*Gradient colour scales*

---

## Description

`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.

## Usage

```
scale_shadowcolour_gradient(
  ...,
  low = "#132B43",
  high = "#56B1F7",
  space = "Lab",
  na.value = "grey50",
  guide = "colourbar",
  aesthetics = "shadowcolour"
```

```

)

scale_shadowcolour_gradient2(
  ...,
  low = muted("red"),
  mid = "white",
  high = muted("blue"),
  midpoint = 0,
  space = "Lab",
  na.value = "grey50",
  guide = "colourbar",
  aesthetics = "shadowcolour"
)

scale_shadowcolour_gradientn(
  ...,
  colours,
  values = NULL,
  space = "Lab",
  na.value = "grey50",
  guide = "colourbar",
  aesthetics = "shadowcolour",
  colors
)

scale_shadowcolour_datetime(
  ...,
  low = "#132B43",
  high = "#56B1F7",
  space = "Lab",
  na.value = "grey50",
  guide = "colourbar"
)

scale_shadowcolour_date(
  ...,
  low = "#132B43",
  high = "#56B1F7",
  space = "Lab",
  na.value = "grey50",
  guide = "colourbar"
)

```

## Arguments

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

scale\_name **[Deprecated]** The name of the scale that should be used for error messages associated with this scale.

- palette** A palette function that when called with a numeric vector with values between 0 and 1 returns the corresponding output values (e.g., `scales::pal_area()`).
- name** The name of the scale. Used as the axis or legend title. If `waiver()`, the default, the name of the scale is taken from the first mapping used for that aesthetic. If `NULL`, the legend title will be omitted.
- breaks** One of:
- `NULL` for no breaks
  - `waiver()` for the default breaks computed by the [transformation object](#)
  - A numeric vector of positions
  - A function that takes the limits as input and returns breaks as output (e.g., a function returned by `scales::extended_breaks()`). Also accepts rlang [lambda](#) function notation.
- minor\_breaks** One of:
- `NULL` for no minor breaks
  - `waiver()` for the default breaks (one minor break between each major break)
  - A numeric vector of positions
  - A function that given the limits returns a vector of minor breaks. Also accepts rlang [lambda](#) function notation. When the function has two arguments, it will be given the limits and major breaks.
- n.breaks** An integer guiding the number of major breaks. The algorithm may choose a slightly different number to ensure nice break labels. Will only have an effect if `breaks = waiver()`. Use `NULL` to use the default number of breaks given by the transformation.
- labels** One of:
- `NULL` for no labels
  - `waiver()` for the default labels computed by the transformation object
  - A character vector giving labels (must be same length as breaks)
  - An expression vector (must be the same length as breaks). See `?plot-math` for details.
  - A function that takes the breaks as input and returns labels as output. Also accepts rlang [lambda](#) function notation.
- limits** One of:
- `NULL` to use the default scale range
  - A numeric vector of length two providing limits of the scale. Use `NA` to refer to the existing minimum or maximum
  - A function that accepts the existing (automatic) limits and returns new limits. Also accepts rlang [lambda](#) function notation. Note that setting limits on positional scales will **remove** data outside of the limits. If the purpose is to zoom, use the `limit` argument in the coordinate system (see `coord_cartesian()`).
- rescaler** A function used to scale the input values to the range [0, 1]. This is always `scales::rescale()`, except for diverging and n colour gradients (i.e., `scale_colour_gradient2()`, `scale_colour_gradientn()`). The rescaler is ignored by position scales, which always use `scales::rescale()`. Also accepts rlang [lambda](#) function notation.

|                 |   |
|-----------------|---|
|                 | <p>oob One of:</p> <ul style="list-style-type: none"> <li>• Function that handles limits outside of the scale limits (out of bounds). Also accepts rlang <code>lambda</code> function notation.</li> <li>• The default (<code>scales:::censor()</code>) replaces out of bounds values with NA.</li> <li>• <code>scales:::squish()</code> for squishing out of bounds values into range.</li> <li>• <code>scales:::squish_infinite()</code> for squishing infinite values into range.</li> </ul> |
|                 | <p>expand For position scales, a vector of range expansion constants used to add some padding around the data to ensure that they are placed some distance away from the axes. Use the convenience function <code>expansion()</code> to generate the values for the expand argument. The defaults are to expand the scale by 5% on each side for continuous variables, and by 0.6 units on each side for discrete variables.</p>  |
|                 | <p>trans <b>[Deprecated]</b> Deprecated in favour of transform.</p>   |
|                 | <p>position For position scales, The position of the axis. left or right for y axes, top or bottom for x axes.</p>  |
|                 | <p>super The super class to use for the constructed scale</p>   |
| low, high       | Colours for low and high ends of the gradient.  |
| space           | colour space in which to calculate gradient. Must be "Lab" - other values are deprecated.   |
| na.value        | Colour to use for missing values  |
| guide           | Type of legend. Use "colourbar" for continuous colour bar, or "legend" for discrete colour legend.  |
| 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> .   |
| 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 <code>rescale()</code> for a convenience function to map an arbitrary range to between 0 and 1.   |

## Details

Default colours are generated with `munsell` and `mns1(c("2.5PB 2/4", "2.5PB 7/10"))`. Generally, for continuous colour scales you want to keep hue constant, but vary chroma and luminance. The `munsell` package makes this easy to do using the Munsell colour system.

## Value

a scale object to add to a plot.

**See Also**

[scales::seq\\_gradient\\_pal\(\)](#) for details on underlying palette

Other colour scales: [scale\\_brewer](#), [scale\\_colour\\_hue](#), [scale\\_colour\\_steps](#), [scale\\_grey](#), [scale\\_viridis](#)

**Examples**

```
library( ggplot2 )
p <- ggplot(economics, aes(date, unemploy, shadowcolor=pce))
p + geom_shadowline() + scale_shadowcolour_gradient() + guides(shadowcolour='none')
```

```
library( ggplot2 )
p <- ggplot(economics, aes(date, unemploy, shadowcolor=pce))
p + geom_shadowline() + scale_shadowcolour_gradient2() + guides(shadowcolour='none')
```

```
library( ggplot2 )
p <- ggplot(economics, aes(date, unemploy, shadowcolor=pce))
p <- p + geom_shadowline() + scale_shadowcolour_gradientn(colours=c('red', 'green'))
p + guides(shadowcolour='none')
```

```
library( ggplot2 )
p <- ggplot(economics, aes(uempmed, unemploy, shadowcolor=as.POSIXct(date)))
p + geom_shadowpath() + scale_shadowcolour_datetime() + guides(shadowcolour='none')
```

```
library( ggplot2 )
p <- ggplot(economics, aes(uempmed, unemploy, shadowcolor=date))
p + geom_shadowpath() + scale_shadowcolour_date() + guides(shadowcolour='none')
```

---

scale\_grey

*Sequential grey colour scales*

---

**Description**

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

**Usage**

```
scale_shadowcolour_grey(
  ...,
  start = 0.2,
  end = 0.8,
  na.value = "red",
  aesthetics = "shadowcolour"
)
```



**Arguments**

...

Arguments passed on to `ggplot2::discrete_scale`

`scale_name` **[Deprecated]** The name of the scale that should be used for error messages associated with this scale.

`palette` A palette function that when called with a single integer argument (the number of levels in the scale) returns the values that they should take (e.g., `scales::pal_hue()`).

`name` The name of the scale. Used as the axis or legend title. If `waiver()`, the default, the name of the scale is taken from the first mapping used for that aesthetic. If `NULL`, the legend title will be omitted.

`breaks` One of:

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

`labels` One of:

- `NULL` for no labels
- `waiver()` for the default labels computed by the transformation object
- A character vector giving labels (must be same length as breaks)
- An expression vector (must be the same length as breaks). See `?plot-math` for details.
- A function that takes the breaks as input and returns labels as output. Also accepts rlang `lambda` function notation.

`limits` One of:

- `NULL` to use the default scale values
- A character vector that defines possible values of the scale and their order
- A function that accepts the existing (automatic) values and returns new ones. Also accepts rlang `lambda` function notation.

`expand` For position scales, a vector of range expansion constants used to add some padding around the data to ensure that they are placed some distance away from the axes. Use the convenience function `expansion()` to generate the values for the `expand` argument. The defaults are to expand the scale by 5% on each side for continuous variables, and by 0.6 units on each side for discrete variables.

`na.translate` Unlike continuous scales, discrete scales can easily show missing values, and do so by default. If you want to remove missing values from a discrete scale, specify `na.translate = FALSE`.

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

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

`position` For position scales, The position of the axis. `left` or `right` for y axes, `top` or `bottom` for x axes.

|            |       |   |
|------------|-------|---|
|            | super | The super class to use for the constructed scale  |
| start      |       | grey value at low end of palette  |
| end        |       | grey value at high end of palette   |
| 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 scale object to add to a plot.

**See Also**

Other colour scales: [scale\\_brewer](#), [scale\\_colour\\_hue](#), [scale\\_colour\\_steps](#), [scale\\_gradient](#), [scale\\_viridis](#)

**Examples**

```
library( ggplot2 )
p <- ggplot(mtcars, aes(wt, mpg, shadowcolour=as.factor(gear)))
p + geom_glowpoint() + scale_shadowcolour_grey() + guides(shadowcolour='none')
```

---

|                |                                   |
|----------------|-----------------------------------|
| scale_identity | <i>Use values without scaling</i> |
|----------------|-----------------------------------|

---

**Description**

Use this set of scales when your data has already been scaled, i.e. it already represents aesthetic values that ggplot2 can handle directly. These scales will not produce a legend unless you also supply the breaks, labels, and type of guide you want.

**Usage**

```
scale_shadowcolour_identity(..., guide = "none", aesthetics = "shadowcolour")
```

**Arguments**

|            |   |
|------------|---|
| ...        | Other arguments passed on to <a href="#">discrete_scale()</a> or <a href="#">continuous_scale()</a>   |
| guide      | Guide to use for this scale. Defaults to "none".  |
| 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> . |

**Details**

The functions `ggplot2::scale_colour_identity()`, `ggplot2::scale_fill_identity()`, `ggplot2::scale_size_identity()`, etc. work on the aesthetics specified in the scale name: `colour`, `fill`, `size`, etc. However, the functions `scale_colour_identity()` and `scale_fill_identity()` also have an optional `aesthetics` argument that can be used to define both `colour` and `fill` aesthetic mappings via a single function call. The functions `scale_discrete_identity()` and `scale_continuous_identity()` are generic scales that can work with any aesthetic or set of aesthetics provided via the `aesthetics` argument.

**Value**

a scale object to add to a plot.

**Examples**

```
library( ggplot2 )
p <- ggplot(mtcars, aes(wt, mpg, shadowcolor='red'))
p + geom_shadowpoint() + scale_shadowcolour_identity()
```

---

 scale\_manual

*Create your own discrete scale*


---

**Description**

These functions allow you to specify your own set of mappings from levels in the data to aesthetic values.

**Usage**

```
scale_shadowcolour_manual(
  ...,
  values,
  aesthetics = "shadowcolour",
  breaks = waiver()
)
```

**Arguments**

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

scale\_name **[Deprecated]** The name of the scale that should be used for error messages associated with this scale.

palette A palette function that when called with a single integer argument (the number of levels in the scale) returns the values that they should take (e.g., `scales::pal_hue()`).

|                           |   |
|---------------------------|---|
| <code>name</code>         | The name of the scale. Used as the axis or legend title. If <code>waiver()</code> , the default, the name of the scale is taken from the first mapping used for that aesthetic. If <code>NULL</code> , the legend title will be omitted.  |
| <code>labels</code>       | One of: <ul style="list-style-type: none"> <li>• <code>NULL</code> for no labels</li> <li>• <code>waiver()</code> for the default labels computed by the transformation object</li> <li>• A character vector giving labels (must be same length as <code>breaks</code>)</li> <li>• An expression vector (must be the same length as <code>breaks</code>). See <code>?plot-math</code> for details.</li> <li>• A function that takes the <code>breaks</code> as input and returns labels as output. Also accepts rlang <code>lambda</code> function notation.</li> </ul> |
| <code>limits</code>       | One of: <ul style="list-style-type: none"> <li>• <code>NULL</code> to use the default scale values</li> <li>• A character vector that defines possible values of the scale and their order</li> <li>• A function that accepts the existing (automatic) values and returns new ones. Also accepts rlang <code>lambda</code> function notation.</li> </ul>  |
| <code>na.translate</code> | Unlike continuous scales, discrete scales can easily show missing values, and do so by default. If you want to remove missing values from a discrete scale, specify <code>na.translate = FALSE</code> .   |
| <code>na.value</code>     | If <code>na.translate = TRUE</code> , what aesthetic value should the missing values be displayed as? Does not apply to position scales where <code>NA</code> is always placed at the far right.  |
| <code>drop</code>         | Should unused factor levels be omitted from the scale? The default, <code>TRUE</code> , uses the levels that appear in the data; <code>FALSE</code> uses all the levels in the factor.  |
| <code>guide</code>        | A function used to create a guide or its name. See <code>guides()</code> for more information.  |
| <code>super</code>        | The super class to use for the constructed scale  |
| <code>values</code>       | a set of aesthetic values to map data values to. The values will be matched in order (usually alphabetical) with the limits of the scale, or with <code>breaks</code> if provided. If this is a named vector, then the values will be matched based on the names instead. Data values that don't match will be given <code>na.value</code> .  |
| <code>aesthetics</code>   | 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 <code>colour</code> and <code>fill</code> aesthetics at the same time, via <code>aesthetics = c("colour", "fill")</code> .   |
| <code>breaks</code>       | One of: <ul style="list-style-type: none"> <li>• <code>NULL</code> for no breaks</li> <li>• <code>waiver()</code> for the default breaks (the scale limits)</li> <li>• A character vector of breaks</li> <li>• A function that takes the limits as input and returns breaks as output</li> </ul>  |

## Details

The functions `scale_colour_manual()`, `scale_fill_manual()`, `scale_size_manual()`, etc. work on the aesthetics specified in the scale name: `colour`, `fill`, `size`, etc. However, the functions

scale\_colour\_manual() and scale\_fill\_manual() also have an optional aesthetics argument that can be used to define both colour and fill aesthetic mappings via a single function call (see examples). The function scale\_discrete\_manual() is a generic scale that can work with any aesthetic or set of aesthetics provided via the aesthetics argument.

### Value

a scale object to add to a plot.

### Color Blindness

Many color palettes derived from RGB combinations (like the "rainbow" color palette) are not suitable to support all viewers, especially those with color vision deficiencies. Using viridis type, which is perceptually uniform in both colour and black-and-white display is an easy option to ensure good perceptive properties of your visualizations. The colorspace package offers functionalities

- to generate color palettes with good perceptive properties,
- to analyse a given color palette, like emulating color blindness,
- and to modify a given color palette for better perceptivity.

For more information on color vision deficiencies and suitable color choices see the [paper on the colorspace package](#) and references therein.

### Examples

```
library( ggplot2 )
p <- ggplot(mtcars, aes(wt, mpg, shadowcolour=as.factor(gear)))
p <- p + geom_glowpoint() + guides(shadowcolour='none')
p + scale_shadowcolour_manual(values=c('red', 'blue', 'green'))
```

---

scale\_viridis

*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_shadowcolour_viridis_d(
  ...,
  alpha = 1,
  begin = 0,
  end = 1,
  direction = 1,
```

```

    option = "D",
    aesthetics = "shadowcolour"
  )

scale_shadowcolour_viridis_c(
  ...,
  alpha = 1,
  begin = 0,
  end = 1,
  direction = 1,
  option = "D",
  values = NULL,
  space = "Lab",
  na.value = "grey50",
  guide = "colourbar",
  aesthetics = "shadowcolour"
)

scale_shadowcolour_viridis_b(
  ...,
  alpha = 1,
  begin = 0,
  end = 1,
  direction = 1,
  option = "D",
  values = NULL,
  space = "Lab",
  na.value = "grey50",
  guide = "coloursteps",
  aesthetics = "shadowcolour"
)

scale_shadowcolour_ordinal(
  ...,
  alpha = 1,
  begin = 0,
  end = 1,
  direction = 1,
  option = "D",
  aesthetics = "shadowcolour"
)

```

### Arguments

|            |  |
|------------|--|
| ...        | Other arguments passed on to <a href="#">discrete_scale()</a> , <a href="#">continuous_scale()</a> , or <a href="#">binned_scale</a> to control name, limits, breaks, labels and so forth. |
| alpha      | The alpha transparency, a number in [0,1], see argument alpha in <a href="#">hsv</a> .   |
| begin, end | The (corrected) hue in [0, 1] at which the color map begins and ends.  |

|            |   |
|------------|---|
| direction  | Sets the order of colors in the scale. If 1, the default, colors are ordered from darkest to lightest. If -1, the order of colors is reversed.  |
| option     | A character string indicating the color map option to use. Eight options are available: <ul style="list-style-type: none"> <li>• "magma" (or "A")</li> <li>• "inferno" (or "B")</li> <li>• "plasma" (or "C")</li> <li>• "viridis" (or "D")</li> <li>• "cividis" (or "E")</li> <li>• "rocket" (or "F")</li> <li>• "mako" (or "G")</li> <li>• "turbo" (or "H")</li> </ul> |
| 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> .   |
| 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.  |
| space      | colour space in which to calculate gradient. Must be "Lab" - other values are deprecated.   |
| na.value   | Missing values will be replaced with this value.  |
| guide      | A function used to create a guide or its name. See <a href="#">guides()</a> for more information.   |

**Value**

a scale object to add to a plot.

**See Also**

Other colour scales: [scale\\_brewer](#), [scale\\_colour\\_hue](#), [scale\\_colour\\_steps](#), [scale\\_gradient](#), [scale\\_grey](#)

**Examples**

```
library( ggplot2 )
p <- ggplot(mtcars, aes(wt, mpg, shadowcolour=as.factor(gear)))
p + geom_glowpoint() + scale_shadowcolour_viridis_d() + guides(shadowcolour='none')
```

```
library( ggplot2 )
p <- ggplot(mtcars, aes(wt, mpg, shadowcolour=gear))
p + geom_glowpoint() + scale_shadowcolour_viridis_c() + guides(shadowcolour='none')
```

```
library( ggplot2 )
p <- ggplot(mtcars, aes(wt, mpg, shadowcolour=gear))
```

```
p + geom_glowpoint() + scale_shadowcolour_viridis_b() + guides(shadowcolour='none')

library( ggplot2 )
p <- ggplot(mtcars, aes(wt, mpg, shadowcolour=as.factor(gear)))
p + geom_glowpoint() + scale_shadowcolour_ordinal() + guides(shadowcolour='none')
```



# Index

## \* colour scales

- scale\_brewer, 11
  - scale\_colour\_hue, 13
  - scale\_colour\_steps, 16
  - scale\_gradient, 20
  - scale\_grey, 24
  - scale\_viridis, 29
- aes(), 3, 5, 7, 10
- aes\_(), 3, 5, 7, 10
- binned\_scale, 30
- binned\_scale(), 12
- borders(), 4, 6, 8, 10
- continuous\_scale(), 12, 26, 30
- coord\_cartesian(), 17, 22
- discrete\_scale(), 12, 26, 30
- expansion(), 15, 17, 23, 25
- fortify(), 3, 5, 8, 10
- geom\_glowline (geom\_glowpath), 2
- geom\_glowpath, 2
- geom\_glowpoint, 5
- geom\_glowpoint(), 5
- geom\_glowstep (geom\_glowpath), 2
- geom\_line(), 4, 8
- geom\_path(), 4, 8, 9
- geom\_point(), 5, 6, 9, 11
- geom\_shadowline (geom\_shadowpath), 6
- geom\_shadowpath, 6
- geom\_shadowpoint, 9
- geom\_shadowpoint(), 9
- geom\_shadowstep (geom\_shadowpath), 6
- geom\_step(), 4, 8
- ggplot(), 3, 5, 7, 10
- ggplot2::binned\_scale, 16
- ggplot2::continuous\_scale, 21
- ggplot2::discrete\_scale, 14, 25, 27
- ggplot2::geom\_line(), 5, 9
- ggplot2::geom\_path(), 5, 9
- ggplot2::geom\_step(), 5, 9
- gradient scale, 16
- gray.colors(), 24
- grid::arrow(), 3, 8
- guides(), 15, 25, 28, 31
- hsv, 30
- lambda, 14, 15, 17, 22, 23, 25, 28
- layer(), 3, 6, 8, 10
- options(), 19
- RColorBrewer::brewer.pal(), 12
- rescale(), 12, 18, 23, 31
- scale\_brewer, 11, 15, 18, 24, 26, 31
- scale\_colour\_gradient(), 19, 20, 24
- scale\_colour\_gradient2(), 22
- scale\_colour\_gradientn(), 22
- scale\_colour\_hue, 13, 13, 18, 24, 26, 31
- scale\_colour\_steps, 13, 15, 16, 24, 26, 31
- scale\_colour\_steps(), 20
- scale\_colour\_viridis\_b(), 20
- scale\_colour\_viridis\_c(), 20
- scale\_continuous, 19
- scale\_fill\_gradient(), 19, 20
- scale\_fill\_steps(), 20
- scale\_fill\_viridis\_b(), 20
- scale\_fill\_viridis\_c(), 20
- scale\_gradient, 13, 15, 18, 20, 26, 31
- scale\_grey, 13, 15, 18, 24, 24, 31
- scale\_identity, 26
- scale\_manual, 27
- scale\_shadowcolour\_binned  
(scale\_continuous), 19
- scale\_shadowcolour\_brewer  
(scale\_brewer), 11

scale\_shadowcolour\_continuous  
    (scale\_continuous), 19

scale\_shadowcolour\_date  
    (scale\_gradient), 20

scale\_shadowcolour\_datetime  
    (scale\_gradient), 20

scale\_shadowcolour\_discrete  
    (scale\_colour\_hue), 13

scale\_shadowcolour\_distiller  
    (scale\_brewer), 11

scale\_shadowcolour\_fermenter  
    (scale\_brewer), 11

scale\_shadowcolour\_gradient  
    (scale\_gradient), 20

scale\_shadowcolour\_gradient2  
    (scale\_gradient), 20

scale\_shadowcolour\_gradientn  
    (scale\_gradient), 20

scale\_shadowcolour\_grey (scale\_grey), 24

scale\_shadowcolour\_hue  
    (scale\_colour\_hue), 13

scale\_shadowcolour\_identity  
    (scale\_identity), 26

scale\_shadowcolour\_manual  
    (scale\_manual), 27

scale\_shadowcolour\_ordinal  
    (scale\_viridis), 29

scale\_shadowcolour\_steps  
    (scale\_colour\_steps), 16

scale\_shadowcolour\_steps2  
    (scale\_colour\_steps), 16

scale\_shadowcolour\_stepsn  
    (scale\_colour\_steps), 16

scale\_shadowcolour\_viridis\_b  
    (scale\_viridis), 29

scale\_shadowcolour\_viridis\_c  
    (scale\_viridis), 29

scale\_shadowcolour\_viridis\_d  
    (scale\_viridis), 29

scale\_viridis, 13, 15, 18, 24, 26, 29

scales::censor(), 17, 23

scales::extended\_breaks(), 17, 22

scales::pal\_area(), 22

scales::pal\_hue(), 14, 25, 27

scales::rescale(), 22

scales::seq\_gradient\_pal(), 18, 24

scales::squish(), 17, 23

scales::squish\_infinite(), 17, 23

transformation object, 17, 22