Package ‘ggshadow’

January 22, 2021

Title Shadow and Glow Geoms for ‘ggplot2’

Version 0.0.2

Description A collection of Geoms for R's 'ggplot2' library. geom_shadowpath(), geom_shadowline(), geom_shadowstep() and geom_shadowpoint() functions draw a shadow below lines to make busy plots more aesthetically pleasing. geom_glowpath(), geom_glowline(), geom_glowstep() and geom_glowpoint() add a neon glow around lines to get a steampunk style.

Depends R (>= 3.4.0)

Imports ggplot2 (>= 3.3.0), grid, scales, rlang, glue

Suggests rmarkdown, knitr

VignetteBuilder knitr

License GPL-2

Encoding UTF-8

LazyData true

URL https://github.com/marcmenem/ggshadow/

BugReports https://github.com/marcmenem/ggshadow/issues

RoxygenNote 7.1.0

Collate 'geom-glowpath.r' 'geom-glowpoint.r' 'geom-shadowpath.r'
          'geom-shadowpoint.r' 'internal-doc.r' 'scale-shadow.r'

NeedsCompilation no

Author Marc Menem [aut, cre]

Maintainer Marc Menem <marc.menem@m4x.org>

Repository CRAN

Date/Publication 2021-01-22 08:50:03 UTC
**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**

```r
gem_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
)
```

```r
gem_glowline(
  mapping = NULL,
  data = NULL,
```
geom_glowpath

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 [aes()] or [aes_()] if specified and ‘inherit.aes = TRUE’ (the default), it is combined with the default mapping at the top level of the plot. You must supply ‘mapping’ if there is no plot mapping.
data The data to be displayed in this layer. There are three options:
If ‘NULL’, the default, the data is inherited from the plot data as specified in the call to [ggplot()].
A ‘data.frame’, or other object, will override the plot data. All objects will be fortified to produce a data frame. See [fortify()] for which variables will be created.
A ‘function’ will be called with a single argument, the plot data. The return value must be a ‘data.frame’, and will be used as the layer data. A ‘function’ can be created from a ‘formula’ (e.g. ‘~ head(.x, 10)’).
stat The statistical transformation to use on the data for this layer, 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 [layer()]. These are often aesthetics, used to set an aesthetic to a fixed value, like ‘colour = “red”’ or ‘size = 3’. 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 [grid::arrow()].
na.rm
If ‘FALSE’, the default, missing values are removed with a warning. If ‘TRUE’, missing values are silently removed.

show.legend
logical. Should this layer be included in the legends? ‘NA’, the default, includes if any aesthetics are mapped. ‘FALSE’ never includes, and ‘TRUE’ always includes. It can also be a named logical vector to finely select the aesthetics to display.

inherit.aes
If ‘FALSE’, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. [borders()].

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

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. * 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.
geom_glowpoint

See Also

[ggplot::geom_path()], [ggplot::geom_line()], [ggplot::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

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 [aes()] or [aes_()]. If specified and ‘inherit.aes = TRUE’ (the default), it is combined with the default mapping at the top level of the plot. You must supply ‘mapping’ if there is no plot mapping.

data The data to be displayed in this layer. There are three options:
If ‘NULL’, the default, the data is inherited from the plot data as specified in the call to [ggplot()].
A ‘data.frame’, or other object, will override the plot data. All objects will be fortified to produce a data frame. See [fortify()] for which variables will be created.
A ‘function’ will be called with a single argument, the plot data. The return value must be a ‘data.frame’, and will be used as the layer data. A ‘function’ can be created from a ‘formula’ (e.g. ‘~ head(.x, 10)’).
**geom_shadowpath**

**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 adjust-
ment function.

...  
Other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. 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. `[borders()]`.

**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**

```r
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 stastep plot, highlighting exactly when changes occur.
Usage

```r
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
)
```

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

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

Arguments

- **mapping**: Set of aesthetic mappings created by [aes()] or [aes_()]. If specified and ‘inherit.aes = TRUE’ (the default), it is combined with the default mapping at the top level of the plot. You must supply ‘mapping’ if there is no plot mapping.

- **data**: The data to be displayed in this layer. There are three options:
  - If ‘NULL’, the default, the data is inherited from the plot data as specified in the call to [ggplot()].
A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.

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

**stat**
The statistical transformation to use on the data for this layer, 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 `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = 'red'` or `size = 3`. 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 `grid::arrow()`.

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

**show.legend**
Logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.

**inherit.aes**
If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. `borders()`.

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

**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 stairstep 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. * shadowsize defaults to 2.5 * size, controls the size of the shadow. * shadowalpha
defaults to 0.25 * alpha or 0.9, controls the alpha of the shadow.

See Also

[ggplot::geom_path()], [ggplot::geom_line()], [ggplot::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()

Description

The point geom is used to create scatterplots. [geom_shadowpoint()] is designed as a drop in re-
placement 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 `aes()` or `aes_()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.

- **data**: The data to be displayed in this layer. There are three options:
  - If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.
  - A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.
  - A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).

- **stat**: The statistical transformation to use on the data for this layer, 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 `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. 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. `borders()`.

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 * size`, controls the size of the shadow.
- `shadowalpha` defaults to `0.25 * alpha` or `0.9`, controls the alpha of the shadow.

Examples

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

*Sequential, diverging and qualitative colour scales from color-brewer.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](https://colorbrewer2.org) for more information.

**Usage**

```r
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 [discrete_scale()], [continuous_scale()], or [binned_scale()], for 'brewer', 'distiller', and 'fermenter' variants respectively, to control name, limits, breaks, labels and so forth.
scale_brewer

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>One of seq (sequential), div (diverging) or qual (qualitative)</td>
</tr>
<tr>
<td>palette</td>
<td>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 be found in the Palettes section.</td>
</tr>
<tr>
<td>direction</td>
<td>Sets the order of colours in the scale. If 1, the default, colours are as output by RColorBrewer::brewer.pal(). If -1, the order of colours is reversed.</td>
</tr>
<tr>
<td>aesthetics</td>
<td>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(&quot;colour&quot;,&quot;fill&quot;).</td>
</tr>
<tr>
<td>values</td>
<td>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 rescale() for a convenience function to map an arbitrary range to between 0 and 1.</td>
</tr>
<tr>
<td>space</td>
<td>colour space in which to calculate gradient. Must be &quot;Lab&quot; - other values are deprecated.</td>
</tr>
<tr>
<td>na.value</td>
<td>Colour to use for missing values</td>
</tr>
<tr>
<td>guide</td>
<td>Type of legend. Use &quot;colourbar&quot; for continuous colour bar, or &quot;legend&quot; for discrete colour legend.</td>
</tr>
</tbody>
</table>

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.
scale_colour_hue

See Also

Other colour scales: scale_colour_hue, scale_colour_steps, scale_gradient, scale_viridis.

Examples

```r
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_brewer() + 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

```r
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  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::hue_pal()).

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

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)
  • A function that takes the breaks as input and returns labels as output

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

de.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.
scale_colour_steps

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()

Description

‘scheme_*_steps’ creates a two colour binned gradient (low-high), ‘scheme_*_steps2’ creates a diverging binned colour gradient (low-mid-high), and ‘scheme_*_stepsn’ creates a n-colour binned gradient. These scales are binned variants of the [gradient scale][scale_colour_gradient] 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())

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)
• A function that takes the breaks as input and returns labels as output

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 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).
• 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.

right Should values on the border between bins be part of the right (upper) bin?

trans For continuous scales, the name of a transformation object or the object itself. Built-in transformations include "asn", "atanh", "boxcox", "date", "exp", "hms", "identity", "log", "log10", "log1p", "log2", "logit", "modulus", "probability", "probit", "pseudo_log", "reciprocal", "reverse", "sqrt" and "time".

A transformation object bundles together a transform, its inverse, and methods for generating breaks and labels. Transformation objects are defined in the scales package, and are called <name>_trans (e.g., scales::boxcox_trans()). You can create your own transformation with scales::trans_new().

show.limits should the limits of the scale appear as ticks

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
low Colours for low and high ends of the gradient.
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 aesthetics = c("colour","fill").
mid colour for mid point
midpoint The midpoint (in data value) of the diverging scale. Defaults to 0.
colours 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 rescale() for a convenience function to map an arbitrary range to between 0 and 1.
colors Vector of colours to use for n-colour gradient.

details Default colours are generated with munsell and 'mnsl(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))
`scale_continuous`

```r
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**

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

```r
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](https://arxiv.org/abs/1903.06490) and references therein.
See Also

[\text{scale\_colour\_gradient}()], [\text{scale\_colour\_viridis\_c}()], [\text{scale\_colour\_steps}()], [\text{scale\_colour\_viridis\_b}()],
[\text{scale\_fill\_gradient}()], [\text{scale\_fill\_viridis\_c}()], [\text{scale\_fill\_steps}()], and [\text{scale\_fill\_viridis\_b}()]

Examples

library(\texttt{ggplot2})
\begin{verbatim}
p <- ggplot(mtcars, aes(wt, mpg, shadowcolor=gear)) p + geom_shadowpoint() + scale_shadowcolour_continuous() + guides(shadowcolour='none')
\end{verbatim}

library(\texttt{ggplot2})
\begin{verbatim}
p <- ggplot(mtcars, aes(wt, mpg, shadowcolor=gear)) p + geom_shadowpoint() + scale_shadowcolour_binned() + guides(shadowcolour='none')
\end{verbatim}

\section*{scale\_gradient}

\emph{Gradient colour scales}

\subsection*{Description}

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

\subsection*{Usage}

\begin{verbatim}
scale\_shadowcolour\_gradient( 
  ..., 
  low = "\#132B43", 
  high = "\#56B1F7", 
  space = "Lab", 
  na.value = "grey50", 
  guide = "colourbar", 
  aesthetics = "shadowcolour"
)
\end{verbatim}

\begin{verbatim}
scale\_shadowcolour\_gradient2( 
  ..., 
  low = muted("red"), 
  mid = "white", 
  high = muted("blue"), 
  midpoint = 0, 
  space = "Lab", 
  na.value = "grey50", 
  guide = "colourbar", 
  aesthetics = "shadowcolour"
)
\end{verbatim}
scale_gradient

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

colorname
```

Arguments

... Arguments passed on to \texttt{ggplot2::continuous_scale}

scale_name 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., \texttt{scales::area_pal()}).

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

breaks One of:

- \texttt{NULL} for no breaks
- \texttt{waiver()} for the default breaks computed by the \texttt{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 \texttt{scales::extended_breaks()})

minor_breaks One of:
**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)
- A function that takes the breaks as input and returns labels as output

**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 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().

**oob** One of:
- Function that handles limits outside of the scale limits (out of bounds).
- 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.

**trans** For continuous scales, the name of a transformation object or the object itself. Built-in transformations include "asn", "atanh", "boxcox", "date", "exp", "hms", "identity", "log", "log10", "log1p", "log2", "logit", "modulus", "probability", "probit", "pseudo_log", "reciprocal", "reverse", "sqrt" and "time".

A transformation object bundles together a transform, its inverse, and methods for generating breaks and labels. Transformation objects are defined in
the scales package, and are called <name>_trans (e.g., scales::boxcox_trans()).
You can create your own transformation with scales::trans_new().

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

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

mid colour for mid point

midpoint The midpoint (in data value) of the diverging scale. Defaults to 0.
colours 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 rescale() for a convenience function to map an arbitrary range to between 0 and 1.
colors Vector of colours to use for n-colour gradient.

Details
Default colours are generated with munsell and ‘mns(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_gray, 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 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::hue_pal()).

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

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)
• A function that takes the breaks as input and returns labels as output

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

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

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

Use values without scaling

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 [discrete_scale()] or [continuous_scale()]
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 ‘aesthetics = c("colour", "fill")’.

Details

The functions ‘scale_colour_identity()’, ‘scale_fill_identity()’, ‘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()
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  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::hue_pal()).
  name        The name of the scale. Used as the axis or legend title. If waiver(), the default, the name of the scale is taken from the first mapping used for that aesthetic. If NULL, the legend title will be omitted.
  labels      One of:
    • NULL for no labels
    • waiver() for the default labels computed by the transformation object
    • A character vector giving labels (must be same length as breaks)
    • A function that takes the breaks as input and returns labels as output
  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
  na.translate Unlike continuous scales, discrete scales can easily show missing values, and do so by default. If you want to remove missing values from a discrete scale, specify na.translate = FALSE.
  na.value     If na.translate = TRUE, what aesthetic value should the missing values be displayed as? Does not apply to position scales where NA is always placed at the far right.
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.

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

The super class to use for the constructed scale

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 ‘breaks’ 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 ‘na.value’.

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”).

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

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.

a scale object to add to a plot.

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](https://arxiv.org/abs/1903.06490) and references therein.

```
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'))
```
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_viridis

Arguments

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

alpha

The alpha transparency, a number in [0,1], see argument alpha in hsv.

begin

The (corrected) hue in [0,1] at which the viridis colormap begins.

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 colormap option to use. Four options are available: "magma" (or "A"), "inferno" (or "B"), "plasma" (or "C"), "viridis" (or "D", the default option) and "cividis" (or "E").

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")'.

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 rescale() 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 guides() 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, 15
  scale_gradient, 20
  scale_grey, 24
  scale_viridis, 29

coord_cartesian(), 17, 22

expansion(), 14, 17, 22, 25

geom_glowline (geom_glowpath), 2
geom_glowpath, 2
geom_glowpoint, 5
geom_glowstep (geom_glowpath), 2
geom_shadowline (geom_shadowpath), 6
geom_shadowpath, 6
geom_shadowpoint, 9
geom_shadowstep (geom_shadowpath), 6

ggplot2::binned_scale (scale_continuous), 19
RColorBrewer::brewer.pal(), 12
rescale(), 12, 18, 23, 30

scale_brewer, 11, 15, 18, 23, 25, 30
scale_colour_gradient2(), 22
scale_colour_gradientn(), 22
scale_colour_hue, 13, 13, 18, 23, 25, 30
scale_colour_steps, 13, 15, 15, 23, 25, 30
scale_continuous, 19
scale_gradient, 13, 15, 18, 20, 25, 30
scale_grey, 13, 15, 18, 23, 24, 30
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), 15
scale_shadowcolour_steps2
  (scale_colour_steps), 15
scale_shadowcolour_stepsn
  (scale_colour_steps), 15
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, 23, 25, 29
scales::area_pal(), 21
scales::boxcox_trans(), 17, 23
scales::censor(), 17, 22
scales::extended_breaks(), 16, 21
scales::hue_pal(), 14, 24, 27
scales::rescale(), 22
scales::squish(), 17, 22
scales::squish_infinite(), 17, 22
scales::trans_new(), 17, 23

transformation object, 16, 21