Package 'ggpath'

May 8, 2026

Title Robust Image Rendering Support for 'ggplot2'

Version 1.1.1

Description A 'ggplot2' extension that enables robust image grobs in panels and theme elements.

License MIT + file LICENSE

URL https://github.com/mrcaseb/ggpath, https://mrcaseb.github.io/ggpath/

BugReports https://github.com/mrcaseb/ggpath/issues

Depends R (>= 4.1.0)

Imports cachem (>= 1.0.0), cli (>= 3.0.0), ggplot2 (>= 4.0.0), grid, magick (>= 2.7.3), memoise (>= 2.0.0), rlang (>= 0.4.11), S7 (>= 0.2.0)

Suggests covr (>= 3.5.1), rsvg (>= 2.0), testthat (>= 3.0.0), vdiffr (>= 1.0.2)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation no

Author Sebastian Carl [aut, cre, cph]

Maintainer Sebastian Carl <mrcaseb@gmail.com>

Repository CRAN

Date/Publication 2025-10-16 08:40:11 UTC

Contents

element_path *Theme Elements for Image Grobs*

Description

In conjunction with the ggplot2::theme() system, the element_ functions specify the display of how non-data components of a ggplot are drawn. Both functions call magick::image_read() to process image files from valid image URLs, local paths, raster objects, or bitmap arrays.

- element_path(): draws images as replacement for ggplot2::element_text(). Use this to replace text with images.
- element_raster(): draws images as replacement for ggplot2::element_rect(). Use this to put images in plot background.

Usage

```
element_path(
  alpha = 1L,
  colour = NA_character_,
  hjust = 0.5,
  vjust = 0.5,
  color = NULL,
  angle = 0,
  size = grid::unit(0.5, "cm")
)

element_raster(
  image_path,
  x = grid::unit(0.5, "npc"),
  y = grid::unit(0.5, "npc"),
  width = grid::unit(1, "npc"),
  height = grid::unit(1, "npc"),
  just = "centre",
  hjust = 0.5,
  vjust = 0.5,
  interpolate = TRUE
)
```

Arguments

alpha	The alpha channel, i.e. transparency level, as a numerical value between 0 and 1. 1L skips alpha channel modification for more speed.
colour, color	The image will be colorized with this color. Defaults to NA_character_ which means no change of color at all. Use the special character "b/w" to set it to black and white. For more information on valid color names in ggplot2 see https://ggplot2.tidyverse.org/articles/ggplot2-specs.html?q=colour#colour-and-fill.

hjust	A numeric vector specifying horizontal justification. If specified, overrides the just setting.
vjust	A numeric vector specifying vertical justification. If specified, overrides the just setting.
angle	The angle of the element as a numerical value between 0° and 360°.
size	The output grob size as a grid::unit. If given a numeric, cm will be applied as unit.
image_path	A file path, url, raster object or bitmap array. See magick::image_read() for further information.
x	A numeric vector or unit object specifying x-location.
y	A numeric vector or unit object specifying y-location.
width	A numeric vector or unit object specifying width.
height	A numeric vector or unit object specifying height.
just	The justification of the rectangle relative to its (x, y) location. If there are two values, the first value specifies horizontal justification and the second value specifies vertical justification. Possible string values are: "left", "right", "centre", "center", "bottom", and "top". For numeric values, 0 means left alignment and 1 means right alignment.
interpolate	A logical value indicating whether to linearly interpolate the image (the alternative is to use nearest-neighbour interpolation, which gives a more blocky result).

Details

To be able to use the functions correctly, a basic understanding of how they work is required.

element_path() can be applied wherever ggplot2::element_text() is usually used. It replaces text with an image if the text is a valid image file location or data.

element_raster() can be applied wherever ggplot2::element_rect() is usually used. A path in the sense of magick::image_read() must be explicitly specified here because it cannot read plot data. It is designed exclusively for inserting an image into the background of a plot and calls grid::rasterGrob() internally. Neither width nor height need to be specified, in which case, the aspect ratio of the image is preserved. If both width and height are specified, it is likely that the image will be distorted.

Value

An S7 object of class element.

See Also

geom_from_path(), grid::rasterGrob(), grid::unit(), magick::image_read()

Examples

```
library(ggplot2)
library(ggpath)

# compute paths of R logo file and background image file shipped with ggpath
local_r_logo <- system.file("r_logo.png", package = "ggpath")
local_background_image <- system.file("example_bg.jpg", package = "ggpath")

# create dataframe with x-y-coordinates and the above local path
plot_data <- data.frame(x = c(-1, 1), y = 1, path = local_r_logo)

# Replace title, subtitle, the caption, axis labels as well as y-axis text
# the the local image
ggplot(plot_data, aes(x = x, y = local_r_logo)) +
  theme_minimal() +
  labs(
    title = local_r_logo,
    subtitle = local_r_logo,
    x = local_r_logo,
    y = local_r_logo,
    caption = local_r_logo
  ) +
  theme(
    plot.caption = element_path(hjust = 1, size = 0.6),
    axis.text.y.left = element_path(size = 1),
    axis.title.x = element_path(),
    axis.title.y = element_path(vjust = 0.9),
    plot.title = element_path(hjust = 0, size = 2, alpha = 0.5),
    plot.subtitle = element_path(hjust = 0.9, angle = 45),
  )

# Use local image as plot background
ggplot(plot_data, aes(x = x, y = y)) +
  geom_from_path(aes(path = path), width = 0.2) +
  coord_cartesian(xlim = c(-2, 2)) +
  theme_dark() +
  theme(
    plot.background = element_raster(local_background_image),
    panel.background = element_rect(fill = "transparent")
  )
```

`geom_from_path` *ggplot2 Layer for Visualizing Images from URLs or Local Paths*

Description

This geom is used to plot images instead of points in a ggplot. It requires x, y aesthetics as well as a path.

Usage

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

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.

`...`	Other arguments passed on to `ggplot2::layer()`. These are often aesthetics, used to set an aesthetic to a fixed value. See the below section "Aesthetics" for a full list of possible arguments.
`na.rm`	If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.
`show.legend`	logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.
`inherit.aes`	If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `annotation_borders()`.

Value

A ggplot2 layer (`ggplot2::layer()`) that can be added to a plot created with `ggplot2::ggplot()`.

Aesthetics

`geom_from_path()` understands the following aesthetics (required aesthetics have no default value):

`x` The x-coordinate. Required.

`y` The y-coordinate. Required.

`path` a file path, url, raster object or bitmap array. See `magick::image_read()` for further information. Required.

`alpha = NULL` The alpha channel, i.e. transparency level, as a numerical value between 0 and 1.

`colour = NULL` The image will be colorized with this colour. Use the special character `"b/w"` to set it to black and white. For more information on valid colour names in ggplot2 see https://ggplot2.tidyverse.org/articles/ggplot2-specs.html?q=colour#colour-and-fill

`angle = 0` The angle of the image as a numerical value between 0° and 360°.

`hjust = 0.5` The horizontal adjustment relative to the given x coordinate. Must be a numerical value between 0 and 1.

`vjust = 0.5` The vertical adjustment relative to the given y coordinate. Must be a numerical value between 0 and 1.

`width = 1.0` The desired width of the image in `npc` (Normalised Parent Coordinates). The default value is set to 1.0 which is *big* but it is necessary because all used values are computed relative to the default. A typical size is `width = 0.1` (see below examples).

`height = 1.0` The desired height of the image in `npc` (Normalised Parent Coordinates). The default value is set to 1.0 which is *big* but it is necessary because all used values are computed relative to the default. A typical size is `height = 0.1` (see below examples).

Examples

```
library(ggplot2)
library(ggpath)

# compute path of an R logo file shipped with ggpath
local_image_path <- system.file("r_logo.png", package = "ggpath")

# create dataframe with x-y-coordinates and the above local path
plot_data <- data.frame(x = c(-1, 1), y = 1, path = local_image_path)

# plot images directly from local path
ggplot(plot_data, aes(x = x, y = y)) +
  geom_from_path(aes(path = path), width = 0.2) +
  coord_cartesian(xlim = c(-2, 2)) +
  theme_minimal()

# plot images directly from local path and apply transparency
ggplot(plot_data, aes(x = x, y = y)) +
  geom_from_path(aes(path = path), width = 0.2, alpha = 0.5) +
  coord_cartesian(xlim = c(-2, 2)) +
  theme_minimal()

# It is also possible and recommended to use the underlying Geom inside a
# ggplot2 annotation
ggplot() +
  annotate(
    ggpath::GeomFromPath,
    x = 0,
    y = 0,
    path = local_image_path,
    width = 0.4
  ) +
  theme_minimal()
```

`geom_lines` *ggplot2 Layer for Horizontal and Vertical Reference Lines*

Description

These geoms can be used to draw horizontal or vertical reference lines in a ggplot. They use the data in the aesthetics `x0` and `y0` to compute their `median` or `mean` and draw them as a line.

Usage

```
geom_median_lines(
  mapping = NULL,
  data = NULL,
  ...,
  na.rm = FALSE,
```

```
  show.legend = NA,
  inherit.aes = TRUE
)

geom_mean_lines(
  mapping = NULL,
  data = NULL,
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by `aes()`.
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)`).
...	Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `....` Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data. • When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the `geom` part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept. • Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept. • The `key_glyph` argument of `layer()` may also be passed on through `....` This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behavior from the default plot specification.

Value

A ggplot2 layer (ggplot2::layer()) that can be added to a plot created with ggplot2::ggplot().

Aesthetics

geom_median_lines() and geom_mean_lines() understand the following aesthetics (at least one of the x0 or y0 aesthetics is required):

x0 The variable for which to compute the median/mean that is drawn as vertical line.

y0 The variable for which to compute the median/mean that is drawn as horizontal line.

alpha = NA The alpha channel, i.e. transparency level, as a numerical value between 0 and 1.

color = "red" The color of the drawn lines.

linetype = 2 The linetype of the drawn lines.

linewidth = 0.5 The width of the drawn lines.

See Also

The underlying ggplot2 geoms ggplot2::geom_hline() and ggplot2::geom_vline()

Examples

```
library(ggplot2)

# inherit top level aesthetics
ggplot(mtcars, aes(x = disp, y = mpg, y0 = mpg, x0 = disp)) +
  geom_point() +
  geom_median_lines() +
  geom_mean_lines(color = "blue") +
  theme_minimal()

# draw horizontal line only
ggplot(mtcars, aes(x = disp, y = mpg, y0 = mpg)) +
  geom_point() +
  geom_median_lines() +
  geom_mean_lines(color = "blue") +
  theme_minimal()
```

```
# draw vertical line only
ggplot(mtcars, aes(x = disp, y = mpg, x0 = disp)) +
  geom_point() +
  geom_median_lines() +
  geom_mean_lines(color = "blue") +
  theme_minimal()

# choose your own value
ggplot(mtcars, aes(x = disp, y = mpg)) +
  geom_point() +
  geom_median_lines(x0 = 400, y0 = 15) +
  geom_mean_lines(x0 = 150, y0 = 30, color = "blue") +
  theme_minimal()
```

Index