Package: ggmagnify (via r-universe)

June 24, 2024

Title Create a Magnified Inset of Part of a ``Ggplot" Object

Version 0.4.1

Author David Hugh-Jones [aut, cre]

Maintainer David Hugh-Jones <davidhughjones@gmail.com>

Description Creates a magnified inset of a ggplot, with projection lines and borders around the target area and inset, plus optional shadow. Rectangular, elliptical or arbitrary regions can be magnified. Works with facets and maps. Geoms can optionally be recomputed within the inset area.

License MIT + file LICENSE

Encoding UTF-8

Language en

Roxygen list(markdown = TRUE, roclets = c(``collate", ``rd", ``namespace", ``doctest::dt_roclet"), packages = ``doctest")

RoxygenNote 7.3.1

Depends R (>= 4.1.0)

Imports cli, ggplot2 (>= 3.3.4), grid, gridExtra, gridGeometry, rlang

Suggests doctest, ggfx, hexbin, maps, rmarkdown, sf, testthat (>= 3.0.0)

URL https://github.com/hughjonesd/ggmagnify, https://hughjonesd.github.io/ggmagnify/

BugReports https://github.com/hughjonesd/ggmagnify/issues

Config/testthat/edition 3

Collate 'compute-shape-grob.R' 'stat-magnify.R' 'geom-magnify.R' 'geom-magnify-tile.R' 'ggplot-utils.R' 'helpers.R' 'inset-theme.R' 'projection.R'

Remotes thomasp85/ggforce

Repository https://hughjonesd.r-universe.dev

RemoteUrl https://github.com/hughjonesd/ggmagnify

RemoteRef v0.4.1

RemoteSha 55ca1ac0a5581695b8f04804b470424ad2f729ed

Contents

GeomMagnify 2
geom_magnify 2
grob_where 6
inset_blanks 7
inset_theme 7
rect_around 8

Index **10**

`GeomMagnify` *Internals*

Description

Internals

`geom_magnify` *Create a magnified inset of a plot*

Description

`geom_magnify()` creates a magnified inset of part of a ggplot. Optional borders are drawn around the target and inset, along with projection lines from one to the other. `from` gives the location of the target area, and `to` gives the location of the inset. Usually, these are specified as `c(xmin, xmax, ymin, ymax)`.

Usage

```
geom_magnify(
  mapping = NULL,
  data = NULL,
  stat = StatMagnify,
  position = "identity",
  ...,
  shape = c("rect", "ellipse", "outline"),
  expand = 0.1,
  aspect = c("free", "fixed"),
  axes = "",
  proj = c("facing", "corresponding", "single"),
  shadow = FALSE,
  corners = 0,
  colour = "black",
  linetype = 1,
  target.linetype = linetype,
```

```
  inset.linetype = linetype,
  proj.linetype = 2,
  alpha = 1,
  linewidth = 0.4,
  proj.fill = NULL,
  plot = NULL,
  shadow.args = list(sigma = 5, colour = "grey40", x_offset = 5, y_offset = 5),
  recompute = FALSE,
  scale.inset = 1,
  proj.combine = TRUE,
  na.rm = FALSE,
  inherit.aes = TRUE
)

geom_magnify_tile(
  mapping = NULL,
  data = NULL,
  stat = StatMagnifyTile,
  position = "identity",
  ...,
  shape = c("rect", "ellipse", "outline"),
  expand = 0.1,
  aspect = c("free", "fixed"),
  axes = "",
  proj = "facing",
  shadow = FALSE,
  corners = 0,
  colour = "black",
  linetype = 1,
  target.linetype = linetype,
  inset.linetype = linetype,
  proj.linetype = 2,
  alpha = 1,
  linewidth = 0.4,
  proj.fill = NULL,
  plot = NULL,
  shadow.args = list(sigma = 5, colour = "grey40", x_offset = 5, y_offset = 5),
  recompute = FALSE,
  scale.inset = 1,
  proj.combine = TRUE,
  na.rm = FALSE,
  inherit.aes = FALSE
)
```

Arguments

mapping, data, stat, position, ..., na.rm	See e.g. ggplot2::geom_point().
shape	Shape of the area to be magnified. "rect" for a rectangle. "ellipse" for an

	ellipse. `"outline"` for the convex hull of points in the target area, or for map polygons.
`expand`	Number. Expand the target area and inset proportionally by this amount.
`aspect`	String. `"fixed"` to fix the aspect ratio (overrides `ymax`).
`axes`	String. Which axes to plot in the inset? `""`, `"x"`, `"y"` or `"xy"`.
`proj`	String. What style of projection lines to draw? `"facing"` (the default), `"corresponding"` or `"single"`. Can be abbreviated. See below.
`shadow`	Logical. Draw a shadow behind the inset plot? Requires the "ggfx" package.
`corners`	Numeric between 0 and 1. Radius of rounded corners for the target area and inset. Only used if `shape` is `"rect"`. 0.1 is a good starting value.
`linetype`, `colour`, `alpha`, `linewidth`	Linetype, colour, alpha and linewidth for borders and projection lines.
`target.linetype`, `inset.linetype`, `proj.linetype`	Linetypes for specific components. Set to `0` for no lines.
`proj.fill`	Colour to fill between the projection lines. `NULL` for no fill. Add alpha using e.g. `scales::alpha()`. Ignored when `proj = "single"`.
`plot`	Ggplot object to plot in the inset. If `NULL`, defaults to the ggplot object to which `geom_magnify()` is added. Overrides `axes` if set. Use `inset_theme()` to give `plot` an appropriate theme.
`shadow.args`	List. Arguments to `ggfx::with_shadow()`.
`recompute`	Logical. If `TRUE`, use lims() to replot the inset. Statistics, e.g. smoothing lines, will be recomputed using only the data in the target area. If `FALSE`, use coord_cartesian() to replot the inset, keeping all the data.
`scale.inset`	Length 1 or 2 numeric. Normally, exactly the target area is shown on the inset. Sometimes you may wish to rescale the plot in the inset. Use 2 numbers to scale width and height separately.
`proj.combine`	Logical. How to draw projection lines when more than one polygon/map area is magnified? `FALSE` draws one set of projection lines for each area. `TRUE` draws a single set of lines for all the areas.
`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()`.

Details

Aesthetics:

geom_magnify understand the following aesthetics (required aesthetics are in bold):

- **from**
- **to**

`from` and `to` can be vectors of length 4, like `list(xmin, xmax, ymin, ymax)`. These specify the bottom left and top right corners of the target area to magnify, and the area for the magnified inset. The lists can optionally be named: `list(xmin = 1, xmax = 2, ymin = 3, ymax = 4)`.

Note: very early versions of ggmagnify used a different order of coordinates: `list(xmin, ymin, xmax, ymax)`.

Alternatively, `from` can be:

- A data frame of points with two columns for `x` and `y`, or a `grid::grob()` object. Points within the grob region (a polygon spanned by the data frame) will be magnified. Points should be on the same scale as the data, with `default.units = "native"` in the grob. `shape` will be ignored.
- A logical vector. Points in the data where `from` is `TRUE` will be surrounded by a rectangle, ellipse or outline.

Normally you'll set `from` and `to` in the call to `geom_magnify()`. You can specify them as aesthetics, e.g. if you want different areas per facet. If so, you may need to wrap them in a `list()` to make sure they are length one per row of data. Only the first row per panel is used. (To magnify multiple areas in one panel, use multiple calls to `geom_magnify()`.)

Shapes:

If `shape = "ellipse"` an elliptical area is magnified. This may not include all points within the target area given by `from`.

If `shape = "outline"` then a convex hull will be drawn around points in the target area. This only works if you are using `geom_point()` or some other geom with aesthetics `x` and `y`. If you are plotting a map, then `"outline"` magnifies exactly the map features selected by `from`.

Projection lines:

- `proj = "corresponding"` or `"facing"` draws projection lines from the corners of the target to the corners of the inset.
  - `"corresponding"` always projects each corner of the target to the same corner of the inset.
  - `"facing"` sometimes draws lines between facing corners, when this looks cleaner.
  - For non-rectangular insets, `"facing"` and `"corresponding"` are the same.
- `"single"` draws a single line from the midpoint of facing sides.
- To draw no lines, set `proj.linetype = 0`.

Limitations:

- `geom_magnify()` uses masks. This requires R version 4.1.0 or higher, and a graphics device that supports masking. If you are using knitr, you may have luck with the `ragg_png` device. If your device doesn't support masks, only `shape = "rect"` will work, and the plot inset will not be clipped to the panel area.
- R graphics devices are not very predictable. My current recommendations are: `ragg_png` for knitr; `cairo_pdf` for PDF output; RStudio `AGG` backend for interactive output. Your mileage may vary.
- `geom_magnify()` uses dark magic to deal with faceting. It may break with older, or newer, versions of ggplot2. If you don't need faceting, and want your code to be robust to upgrades, set `options(ggmagnify.safe_mode = TRUE)` to use slightly less magic.
- By design, `geom_magnify()` replots the original plot using new limits. It does not directly copy the target area pixels. The advantage is that you can e.g. add axes, plot points at an appropriate size, zoom in on data that's invisible in the main plot, or recompute derived graphics. If you want an exact pixel-by-pixel copy, use a different tool.
- `geom_magnify()` may break with discrete scales. This is a limitation in ggplot2 for now.
- Find a bug? Report it at https://github.com/hughjonesd/ggmagnify/issues/.

`geom_magnify_tile()`:

`geom_magnify_tile()` is a version of `geom_magnify()` which uses different aesthetics. Set `x`, `width`, `y`, `height` and `to_x`, `to_width` `to_y`, `to_height` to specify the target and inset location.

Examples

```
library(ggplot2)
ggp <- ggplot(iris, aes(Sepal.Width, Sepal.Length, colour = Species)) +
         geom_point() + xlim(c(2, 6))
from <- list(2.5, 3.5, 6, 7)
to <- list(4, 6, 5, 7)

# Basic magnification
ggp + geom_magnify(from = from, to = to)

# Convex hull of points
ggp + geom_magnify(aes(from = Species == "setosa"), to = c(3, 5, 6, 8),
                   shape = "outline")

# Order matters

# `geom_magnify()` stores the plot when it is added to it:
ggp +
  scale_color_brewer() +
  geom_magnify(from = from, to = to)

# This will print the inset without the new scale:
ggp +
  geom_magnify(from = from, to = to) +
  scale_color_brewer()

# For more examples see https://github.com/hughjonesd/ggmagnify
```

`grob_where` *Create a grob from a subset of sf data*

Description

Create a grob from a subset of sf data

Usage

```
grob_where(where, sf, crs = NULL)
```

Arguments

`where`	Logical condition to be evaluated in `sf`.
`sf`	An object of class sf.
`crs`	Optional coordinate reference system.

Value

A `grid::grob()` which you can pass to `from`.

Examples

```
library(ggplot2)
if (requireNamespace("sf", quietly = TRUE) &&
  requireNamespace("maps", quietly = TRUE)) {
  usa <- sf::st_as_sf(maps::map("state", fill=TRUE, plot = FALSE))
  texas <- grob_where(ID == "texas", usa, crs = sf::st_crs(4326))
  ggplot(usa) + coord_sf(crs = sf::st_crs(4326)) + geom_sf() +
    geom_magnify(from = texas, to = c(-90, -70, 25, 35), colour = "red",
    aspect = "fixed", expand = 0)
}
```

---

inset_blanks *Default elements to blank in the ggmagnify inset*

Description

Default elements to blank in the ggmagnify inset

Usage

```
inset_blanks(..., axes)
```

Arguments

...	Character vector of extra elements to blank.
axes	String. Which axes to plot in the inset? "", "x", "y" or "xy".

Value

A character vector.

---

inset_theme *Create a theme suitable for an inset ggplot*

Description

Use `inset_theme()` to add a suitable theme to a manually-created inset plot.

Usage

```
inset_theme(
  blank = inset_blanks(axes = axes),
  axes,
  margin = if (axes == "") 0 else 8
)
```

Arguments

blank	Character vector of theme elements to blank. See ggplot2::theme().
axes	String. Which axes to plot in the inset? "", "x", "y" or "xy".
margin	Margin around the plot. See plot.margin in ggplot2::theme().

Value

A ggplot theme object

Examples

```
library(ggplot2)
ggp <- ggplot(iris, aes(Sepal.Width, Sepal.Length, colour = Species)) +
         geom_point() + xlim(c(2, 6))
from <- list(2.5, 3.5, 6, 7)
to <- list(4, 6, 5, 7)

inset <- ggp + geom_density2d() + inset_theme(axes = "")
ggp + geom_magnify(from = from, to = to, plot = inset)
```

---

rect_around *Helper functions to find rectangles or convex hulls of data*

Description

Helper functions to find rectangles or convex hulls of data

Usage

```
rect_around(x, y, data = NULL, expand = 0)

hull_around(x, y, data = NULL, expand = 0)
```

Arguments

x, y	Unquoted names or expressions
data	A data frame
expand	Amount to expand the data around its midpoint. Default is 10 per cent.

Value

rect_around() returns a list with names xmin, xmax, ymin, and ymax. hull_around() returns a data frame with columns x and y.

Examples

```
library(ggplot2)
to <- c(2, 4.5, 6, 8)
setosas <- iris[iris$Species == "setosa", ]
rect_around(Sepal.Width, Sepal.Length, data = setosas)
hull_around(Sepal.Width, Sepal.Length, data = setosas)
```

Index

∗ **datasets**
GeomMagnify, 2

borders(), *4*

coord_cartesian(), *4*
coordinate reference system, *6*

geom_magnify, 2
geom_magnify_tile *(*geom_magnify*)*, 2
geom_point(), *5*
GeomMagnify, 2
ggfx::with_shadow(), *4*
ggplot2::geom_point(), *3*
ggplot2::theme(), *8*
grid::grob(), *5*, *6*
grob_where, 6

hull_around *(*rect_around*)*, 8

inset_blanks, 7
inset_theme, 7
inset_theme(), *4*

lims(), *4*
list(), *5*

rect_around, 8

scales::alpha(), *4*
sf, *6*