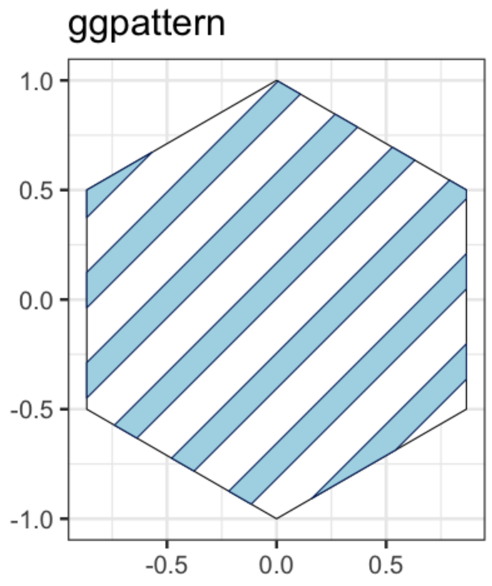
Introducing ggpattern - pattern fills for ggplot

2020-04-01

ggpattern

cool useless



`ggpattern` provides custom `ggplot2` geoms which support filled areas with geometric and image-based patterns.

Reading the articles/vignettes on the package website is probably the best way to get started.

Feature Summary

- Custom versions of (almost) all the **geoms** from ggplot2 which have a region which can be filled.
- A suite of **aesthetics** for controlling the pattern appearance (e.g. `pattern_alpha`)
- The ability to include **user-defined patterns**

Installation

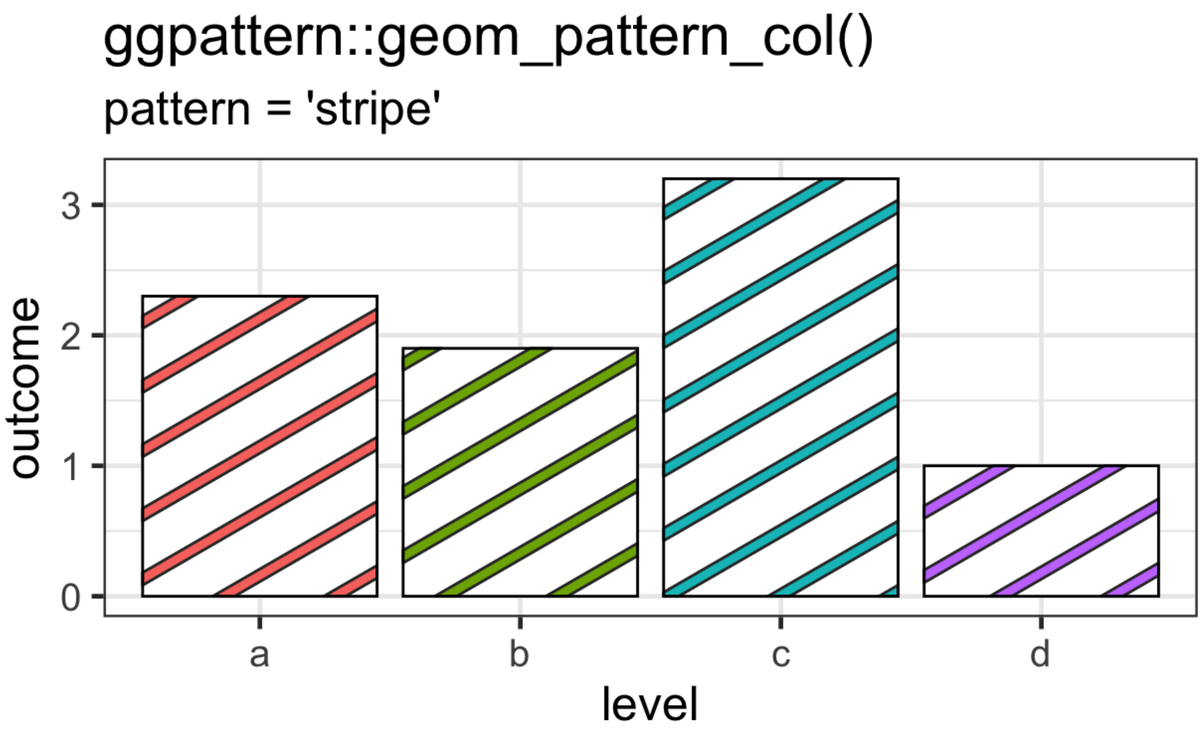
You can install the development version from GitHub with:

```
# install.packages("remotes")
remotes::install_github("coolbutuseless/ggpattern")
```

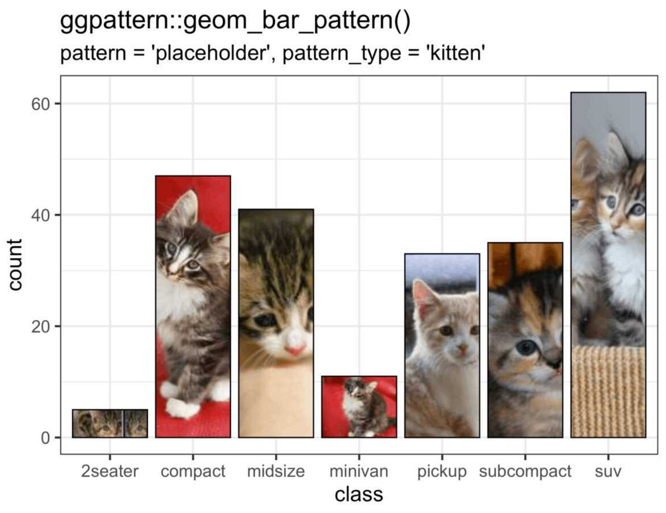
Quickstart

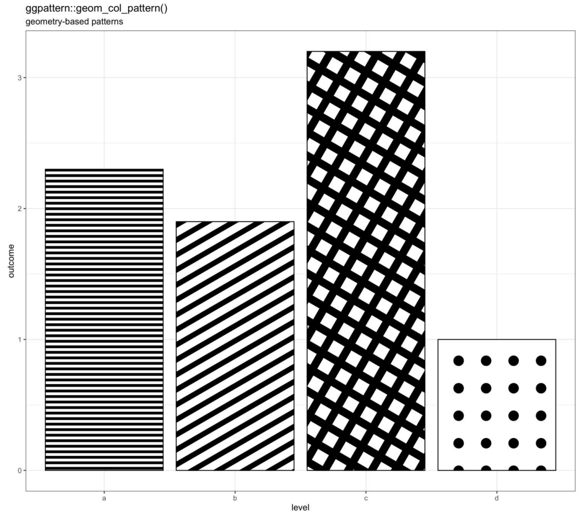
1. Take an existing plot which contains a geom with a fillable area e.g `geom_col()`.
2. Use the `{ggpattern}` version of the geom e.g. `ggpattern::geom_col_pattern()` instead of `ggplot2::geom_col()`
3. Set the aesthetic `pattern` to your choice of pattern e.g `pattern = 'stripe'`, and set other options using `pattern_*` aesthetics

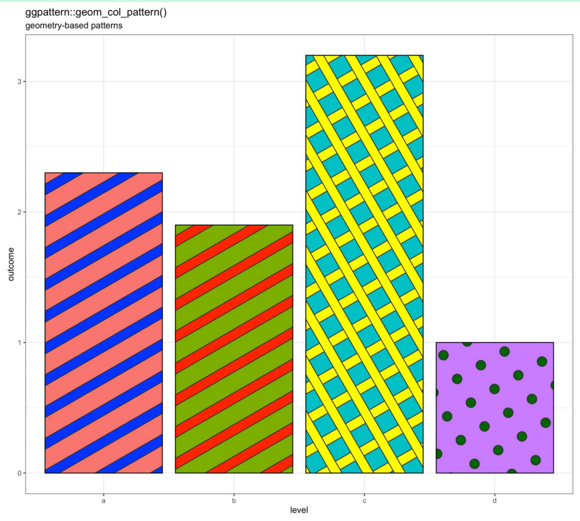
```r
df <- data.frame(level = c("a", "b", "c", 'd'), outcome = c(2.3, 1.9, 3.2, 1))

ggplot(df) +
  geom_col_pattern(
    aes(level, outcome, pattern_fill = level),
    pattern = 'stripe',
    fill    = 'white',
    colour  = 'black'
  ) +
  theme_bw(18) +
  theme(legend.position = 'none') +
  labs(
    title    = "ggpattern::geom_pattern_col()",
    subtitle = "pattern = 'stripe'"
  ) +
  coord_fixed(ratio = 1/2)
```

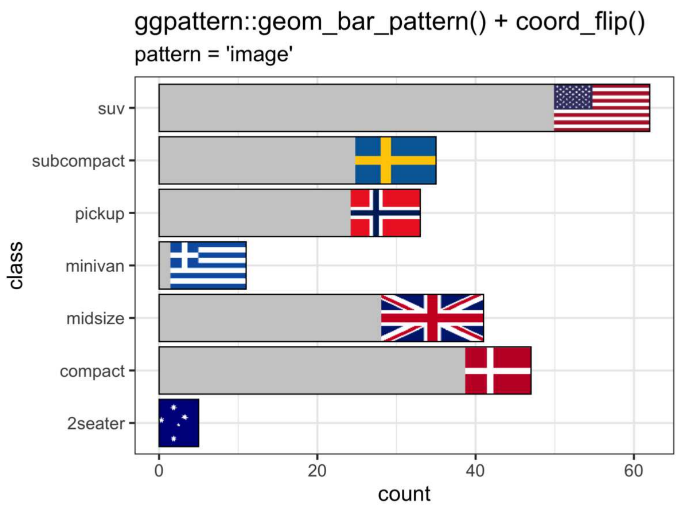


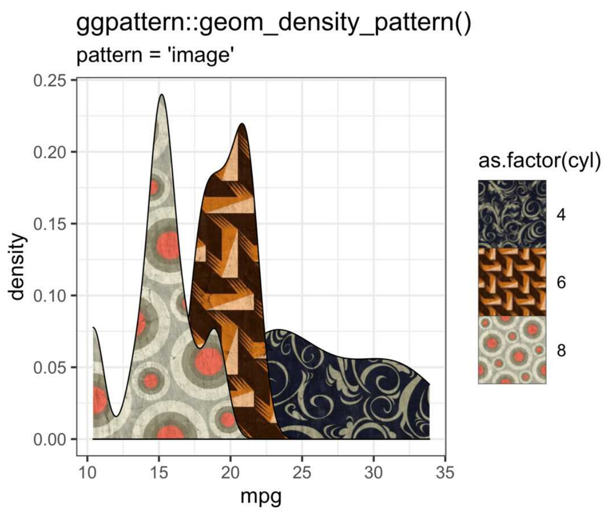
Gallery

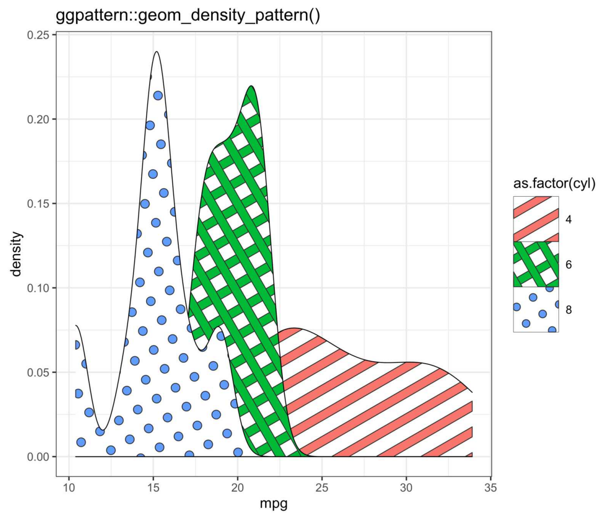


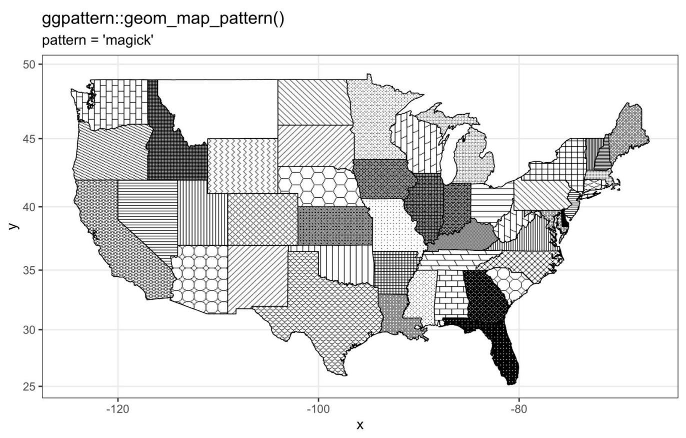


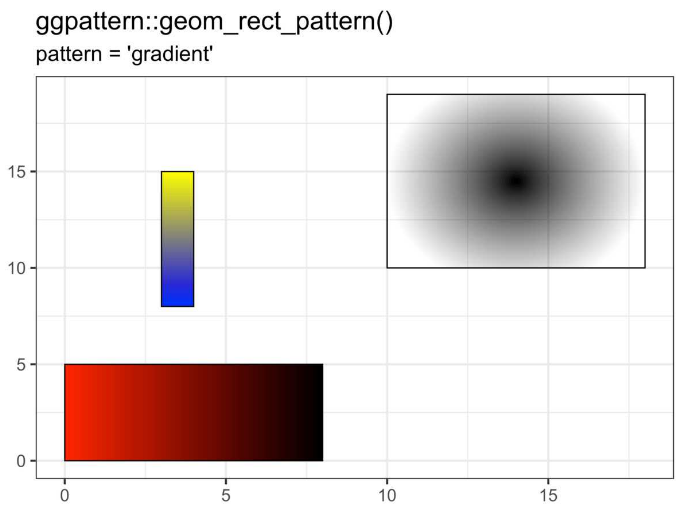












Feature Details

Available Geoms

`ggpattern` includes versions of (nearly) all geoms from `ggplot2` which could plausiblly support being filled with a pattern.

See the vignette galleries for examples of all the available geoms filled with geometry-based patterns and image-based/array-based patterns.

▸ Click to show/hide list of supported geoms

New aesthetics

To control pattern appearance, a raft of new aesthetics have been added. e.g. `pattern_alpha`, `pattern_filename`, `pattern_density`.

There are also scale functions to control each of these new aesthetics e.g. `scale_pattern_alpha_discrete`.

Not all aesthetics apply to all patterns. See the individual pattern vignettes for which aesthetics it uses, or see the first vignette on developing user-defined patterns for a table of aesthetic use by pattern, or see the individual vignettes for each pattern.

▸ Click to show/hide list of new aesthetics

User-Defined Patterns

Users can write their own pattern functions and ask `ggpattern` to use them, without having to include the pattern in the package.

See the vignettes on developing patterns (1 2, 3) for how to do this, and see the vignettes on experimental patterns to see this in action (Point filling, Hex pattern, Ambient Noise).

Vignettes

General examples

- geom gallery (geometry-based patterns) Examples of every geom filled with the geometry-based patterns (i.e. 'stripe', 'crosshatch', 'circle')
- geom gallery (array-based patterns) Examples of every geom filled with the array-based patterns (i.e. 'image', 'magick', 'gradient', 'plasma', 'placeholder')

Exploration of pattern parameters and appearance

- Geometry-based patterns
    - Common aesthetics for geometry-based patterns
    - stripes
    - crosshatch
    - circles
- Array-based patterns
    - image
    - placeholder
    - gradient
    - plasma
    - magick

Developing your own pattern

- Devloping Patterns 1 - Pattern Overview
- Devloping Patterns 2 - Geometry-based pattern
- Devloping Patterns 3 - Array-based pattern

Experimental patterns

These are patterns that aren't quite ready for prime-time. Feel free to steal the code and extend to suit your needs.

- Point filling
- Hex pattern
- Ambient Noise

Other examples

- gganimate

Limitations

- Nearly always need to use `coord_fixed()` to ensure the aspect ratio is calculated correctly. Use `pattern_aspect_ratio` to override the internal calculation, of for occasions where you can't use `coord_fixed()` because a different `coord_*()` is used.
- Legend rendering for patterns is still not great.
    - Use `pattern_key_scale_factor` to adjust legend appearance.
- The Rstudio output device can be quite slow for plots with lots of patterns. It is often faster to save directly to PNG or PDF and view that.
- Self intersecting geometry can be an issue.
- Non-linear coordinate systems have not been tested.

- Polygons with holes are not supported

ToDo

- Possibly add geoms from third-party sources e.g.
    - `geom_circle()` and `geom_voronoi()` from ggforce
- A technical vignette on how array-based patterns are implemented.