

latinR proposal: Flipping the sf plotting script with new geography-specific Stats

Evangeline Reynolds

2024-08-27

Contents

Status quo mapping with ggplot2's <code>geom_sf()</code>	1
A new way: geography-specific layers (when aesthetic mapping is really about <i>mapping</i>)	2

Status quo mapping with ggplot2's `geom_sf()`

ggplot2 is a beloved graphical system because of its elegance and flexibility. Usually a graph requires the user to specify three elements (and ggplot2 can take care of the rest):

1. **data** - which dataframe will be the basis of the graphic
2. **aesthetic mapping** - should variables be represented by x position, y position, color, linewidth etc, and
3. **geom**, the mark that should take on these aesthetics.

ggplot2 added more geographic viz capabilities with the addition of its `geom_sf_*`() functions. However, geographic mapping with `geom_sf` may feel a little out-of-step with ggplot2's other plot builds, as the user doesn't have to map any aesthetics to build a map! (?) Let's consider creating a choropleth of characteristic that varies with geographic area. Here's example syntax.

```
library(tidyverse)
# get shape data
nc_shapes <- sf::st_read(system.file("shape/nc.shp", package = "sf"), quiet = TRUE)

# get
my_nc_data_w_shapes <- nc_shapes %>% left_join(my_nc_data, by = "county_name")

ggplot(nc) +
  geom_sf()

last_plot() +
  aes(fill = my_var)
```

A first step is to find shape data for the areas of interest, ideally in a data frame with one column identifying the geographic area, and another list-column often named `geometry` with the geographic information included. Then the actual data of interest is joined to the geography data frame via the identifying column. Then this data can be input into `ggplot()`. To render a map, `geom_sf()` can be added with no aesthetic mapping required, because under the hood, this layer is directed to look for a column named `geometry`. But to make the map a choropleth, aesthetic mapping instruction `aes(fill = my_var)` would be required.

Whereas so many geoms *require* at least one positional aesthetic (x and y) to be declared in plot syntax, sf weirdly doesn't - as it's all managed with precomputation and under-the-hood column discovery.

A new way: geography-specific layers (when aesthetic mapping is really about *mapping*)

This talk will propose creating a Stat to allow for a workflow that's more similar to the 'classic' ggplot2 build, and ready for repeat use. The new layer that's propose would *require* a positional aesthetic to be declared by the user - but this will feel a little different from x and y, instead being a geographic aesthetic - ids of a geographic location (names, codes, abbreviations, etc). The proposed syntax example:

```
ggplot(my_nc_data) +  
  aes(county_name = county) +  
  geom_sf_county()  
  
last_plot() +  
  aes(fill = my_var)
```

A micro, but usable preparation is as follows:

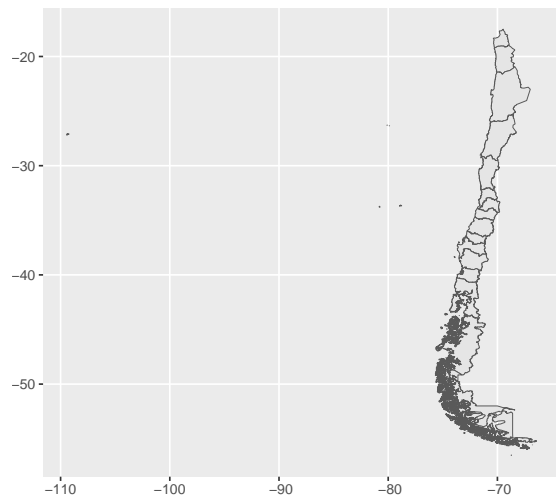
```
## some flat data we might want to make a choropleth with  
region_poblacion %>%  
  head()  
  
## # A tibble: 6 x 2  
##   region poblacion  
##   <chr>      <dbl>  
## 1 01          318724  
## 2 02          579305  
## 3 03          264856  
## 4 04          707918  
## 5 05          1721779  
## 6 06           865190  
  
## some geographic shape data  
chile_regiones <- chilemapas::generar_regiones()  
  
## Prep reference data for Stat (NEW!)  
geo_reference_chile_region <- chile_regiones |>  
  dplyr::select(region_codigo = codigo_region) |>  
  ggplot2::StatSf$compute_panel(coord = ggplot2::CoordSf) |>  
  ggplot2::StatSfCoordinates$compute_group(coord = ggplot2::CoordSf)  
  
# Flip the script... prepare compute (join) to happen in layer (NEW!)  
compute_panel_region <- function(data, ...){  
  
  data %>% inner_join(geo_reference_chile_region)  
  
}  
  
# Write a dedicated stat to do the join for you (NEW!)  
StatRegion <- ggproto(`_class` = "StatRegion",  
  `_inherit` = ggplot2::Stat,  
  compute_panel = compute_panel_region,
```

```
    required_aes = "region_codigo"  
  )
```

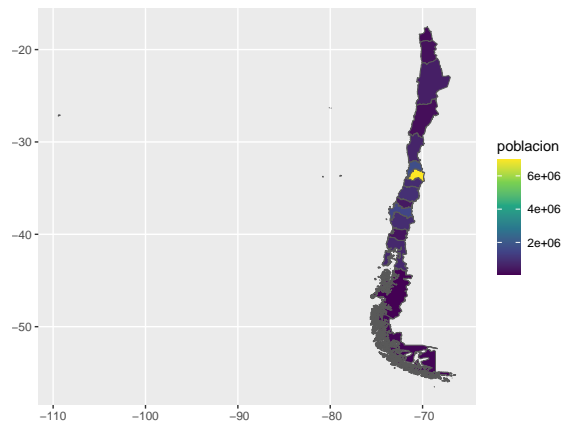
```
head(region_poblacion)
```

```
## # A tibble: 6 x 2  
##   region poblacion  
##   <chr>      <dbl>  
## 1 01          318724  
## 2 02          579305  
## 3 03          264856  
## 4 04          707918  
## 5 05         1721779  
## 6 06          865190
```

```
region_poblacion |>  
  ggplot() +  
  aes(region_codigo = region) +  
  geom_sf(stat = StatRegion, linewidth = .01)
```



```
last_plot() +  
  aes(fill = poblacion) +  
  scale_fill_viridis_c()
```



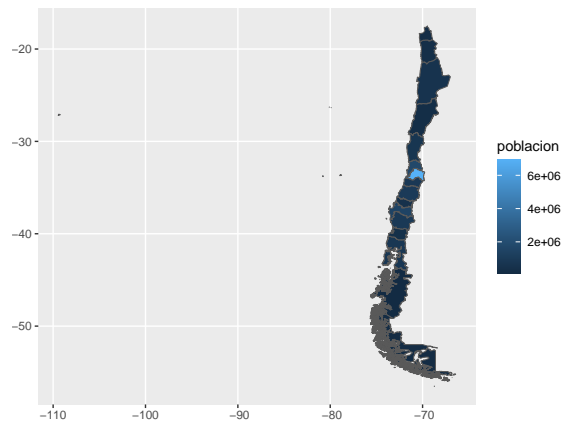
A basic user-facing function can be written like this.

```
geom_region <- function(...){ geom_sf(stat = StatRegion, ...)} # (express-type layer - currently underg
```

The question this exercise raises is, could packages like `chilemapas`, `geobrasil`, `geoAr`, `mapasPERU`, `geouy`, as well packages with data about other parts of the world, not only provide the awesome interfaces to the geographic data they do, but possibly also plotting functionality like the above `geom_region`? Then with the `Stat` and `geom_region()` from this package, you'd be able to build choropleths in a snap, without the precomputation, etc. So then getting from your flat dataframe that has region numbers and population as columns to choropleth would look like this:

```
library(chilemapas)

region_poblacion %>%
  ggplot() +
  aes(region_codigo = region) +
  geom_region() + # proposed chilemapas (?) plotting function
  aes(fill = poblacion)
```



Another, lower maintenance approach would be to provide documentation on how to build geography-specific Stats, so that organizations like Chilean universities, news organizations, etc, could use the 'flipped script' approach which might make map building more intuitive and fun.