ANIMATING MAPS WITH GGANIMATE

PUBLISHED MON, JAN 1, 2018 BY MICHELLE EVANS

I've recently been working on a monthly yellow fever dataset from Brazil that spans fourteen years. Overall, there are over 5600 spatial units and 168 months (that's nearly 1,000,000 rows of data for those of you keeping track). It is great to have access to so much data, but sometimes visualizing it can be a bit of a pain, especially when we are trying to look at patterns across time and space. Static maps can show the spatial patterns, time series plots can show seasonal patterns and can even be broken down to regions, but visualizing thousands of lines becomes basically uninterpretable. Luckily, in the age of computers, we are not limited to a static map, and can instead loop together maps over time using animation.

There are many ways to create animated plots in R, including interactive tools such as shiny, but as a ggplot user I'm going to focus on a package called gganimate. This package follows all the grammar of ggplot and simply

adds an aesthetic called `frame` that will looped over.

To illustrate this, let's look at the presidential election results from the 20th century, taking advantage of a dataset in the `choroplethr` package. This package also has it's own function to animate maps (choroplethr_animate), but I prefer using gganimate as it is more customizable and saves the resulting image as a gif, rather than an html file.

```
#load packages
library(choroplethr)
library(tidyverse)
#devtools::install_github("dgrtwo/gganimate")
library(gganimate)
library(maps)
library(htmltools)

#download president data
data("df_president_ts")
```

For the actual spatial data, I use the data that comes with `ggplot` which means I don't need to fortify any SpatialDataFrames into ggplot-readable objects.

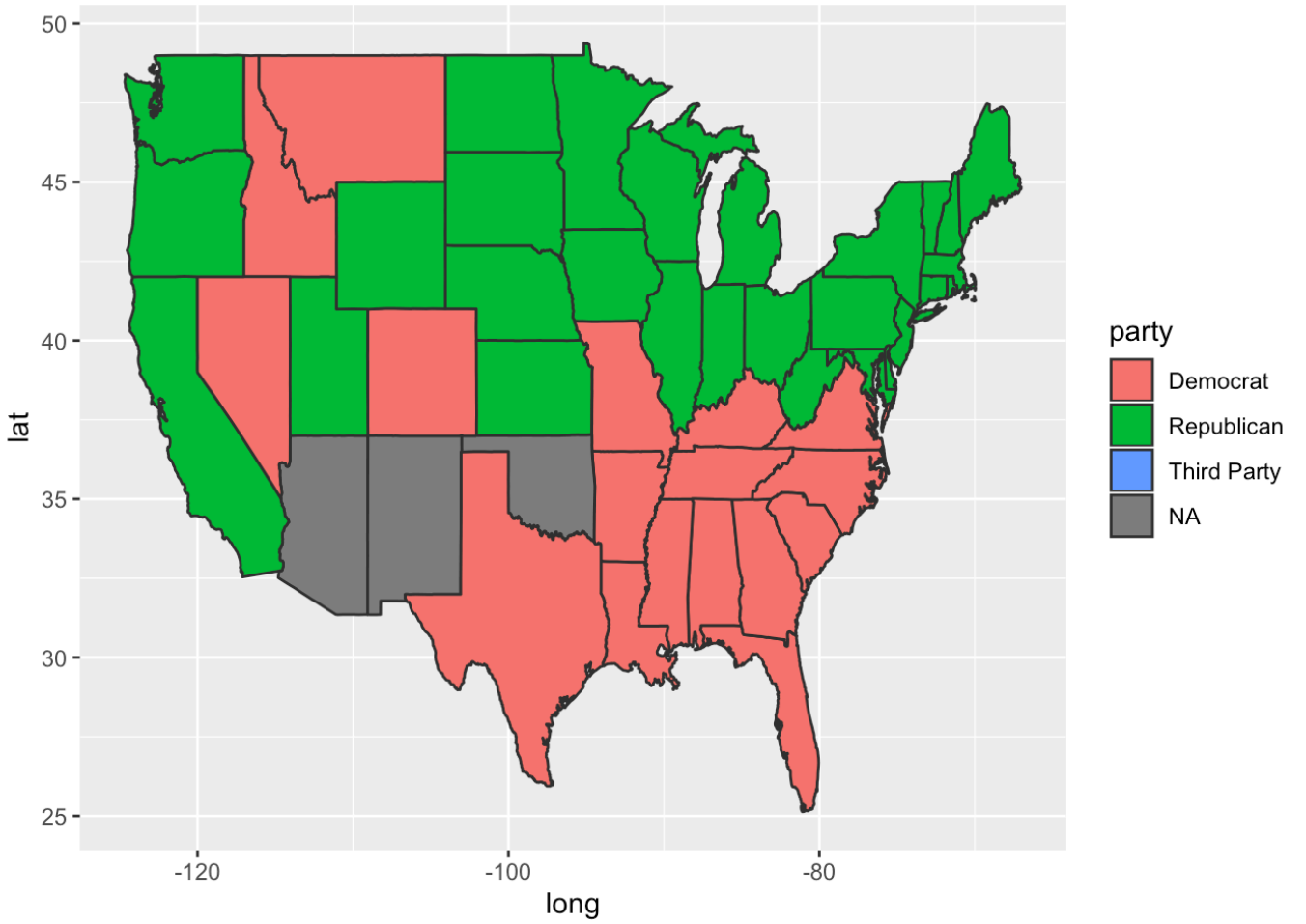
```
us <- map_data("state")
```

I'll focus only on election outcomes post-1900 to reduce the number of political parties we'll need to map. The easiest way to do this is via `dplyr` after transforming the dataframe into a long dataframe. I will also sort any candidates that aren't Democrat or Republican into a new category, Third Party.

```
elections <- df_president_ts %>%
  #gather into long data
  gather(year, winner, `1789`:`2012`) %>%
  #filter only elections after 1900
  filter(year >= 1900) %>%
  #join with state polygons
  right_join(us, by = "region") %>%
  mutate(party = case_when(
    winner %in% c("SR", "I", "AI", "PR") ~ "Third Party",
    winner == "D" ~ "Democrat",
    winner == "R" ~ "Republican"
  ))
```

Now I will set up a base map to start editing before we animate it all.

```
ggplot() +
  # polygons
  geom_polygon(data = elections, aes(x = long,
                                     y = lat,
                                     group = group,
                                     fill = party),
               color = "gray20")
```

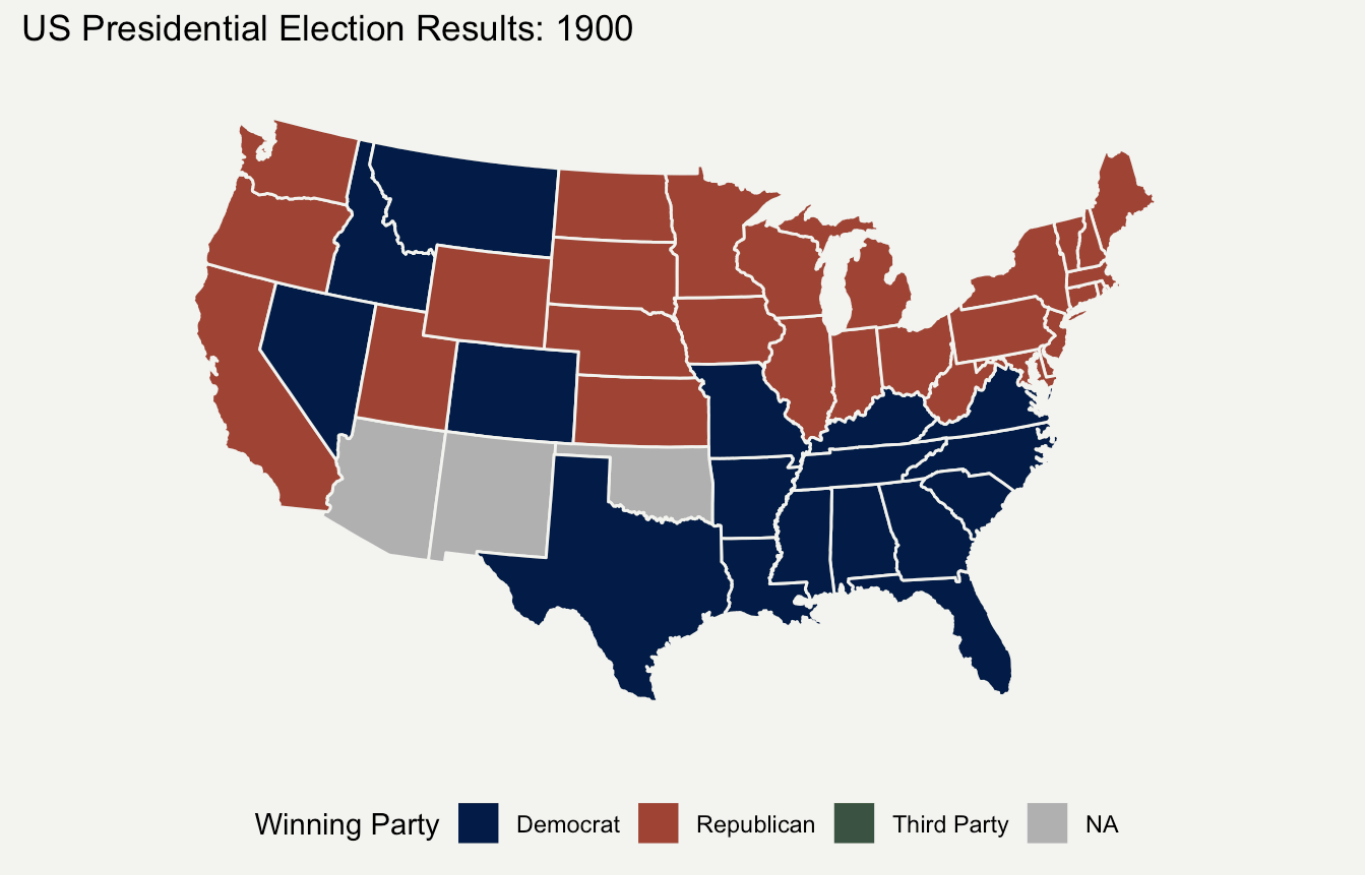


This map looks pretty horrible, so I'll edit the format and colors. Much of this is borrowed from Timo Grossenbacher's blog post on creating beautiful maps in ggplot. I like to create this as a seperate theme object which I can then add to the ggplot object.

```
theme_map <- function(...) {
  theme_minimal() +
  theme(
    axis.line = element_blank(),
    axis.text.x = element_blank(),
    axis.text.y = element_blank(),
    axis.ticks = element_blank(),
    axis.title.x = element_blank(),
    axis.title.y = element_blank(),
    panel.grid.minor = element_blank(),
    panel.grid.major = element_blank(),
```

```
    plot.background = element_rect(fill = "#f5f5f2", color = NA),
    panel.background = element_rect(fill = "#f5f5f2", color = NA),
    legend.background = element_rect(fill = "#f5f5f2", color = NA),
    panel.border = element_blank(),
    legend.position = "bottom",
    ...
  )
}
```

```
ggplot() +
  # polygons
  geom_polygon(data = elections, aes(x = long,
                                     y = lat,
                                     group = group,
                                     fill = party),
                 color = "#f5f5f2") +
  theme_map() +
  coord_map("albers", lat0=30, lat1=40) +
  scale_fill_manual(values = c("#05204A", "#A24936", "#3E5641"),
                    na.value = "gray70",
                    name = "Winning Party") +
  labs(x = NULL,
       y = NULL,
       title = "US Presidential Election Results: 1900"
       )
```



When using `gganimate`, the variable that you set as the frame aesthetic will be added to the title. This means I can then just set the title as "US Presidential Election Results:", and the year number (i.e. 1900) will be appended onto the end.

Unlike before, where I was calling data in the `geom_polygon` call, gganimate requires the data and frame aesthetic to be called in the original ggplot call.

```
p1 <- ggplot(data = elections, aes(frame = year)) +
  # polygons
  geom_polygon(aes(x = long, y = lat, group = group, fill = party),
                color = "#f5f5f2") +
  theme_map() +
```

```
  coord_map("albers", lat0=30, lat1=40) +
  scale_fill_manual(values = c("#05204A", "#A24936", "#3E5641"),
                    na.value = "gray70",
                    name = "Winning Party") +
  ggtitle("US Presidential Election Results: ")
```

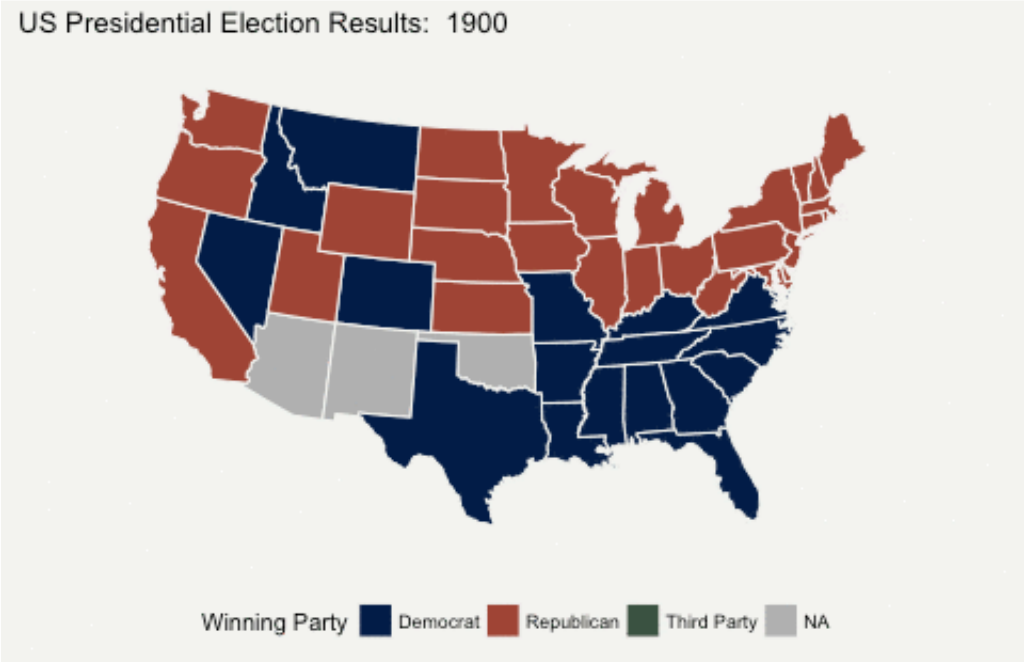
Now I just call this plot `p1` in the `gganimate` function, setting the `interval` to 1.5 seconds.

```
gganimate(p1, interval = 1.5)
```

Essentially, what this function is doing is creating these plots, saving them in virtual memory, and then combining them using Image Magick (which can also be accessed via the command line). The ease of being able to combine this within one R script, rather than having to run shell scripts from within R, makes this an efficient, and reproducible, way to animate plots.

The code above will open the image in your Image Viewer. If you want to save the image, simply include the filename in the call.

```
gganimate(p1, interval = 1.5, filename = "animationmap.gif")
```



Tagged: animation, ggplot, spatial, R

M.EVANS