Wisconsin Gubernatorial Results, 1990-2018

In this project, we map ward-level results of each Wisconsin gubernatorial election since 1990.

Part 1: Load required packages

To execute our code, four packages are needed (beyond the packages included with base R). We also load the `here` package to provide portability across users.

```r
# Load packages
library(here)
library(scales)
library(sf)
library(tidyverse)
library(tigris)
```

Part 2: Load data

To get our ward-level data, we will read in three files with election data from the Wisconsin Legislative Technology Services Bureau. In doing do, we request the ward-level data using the ward boundaries (and accompanying data) as of 2018. To take advantage of the `sf` data format, we will use the `sf::st_read` function. Additionally, we will use the `tigris` package to request county boundaries to provide county outlines in our maps to help the viewer orient oneself within the state.

```r
# Store URL as separate objects to help with PDF formatting
url_1990_2000 <- paste0('https://opendata.arcgis.com/datasets/',
                        '44c7cd59ac4b4390abdfd28815f8b1ad_0.geojson')
url_2002_2010 <- paste0('https://opendata.arcgis.com/datasets/',
                        '336c768bad3041de89a80011015e0a3e_0.geojson')
url_2012_2020 <- paste0('https://opendata.arcgis.com/datasets/',
                        '99cddbb661bd42d3b9e37fd6ef6251a5_0.geojson')

# Read in Wisconsin ward-level data using 2018 ward definitions
wi_1990_2000 <- sf::st_read(url_1990_2000)
wi_2002_2010 <- sf::st_read(url_2002_2010)
wi_2012_2020 <- sf::st_read(url_2012_2020)
wi_counties <- tigris::counties(state = 'WI',
                                cb = TRUE,
                                resolution = '500k',
                                year = 2018,
                                class = 'sf')
```

Part 3: Transform data

The data provided by the Legislative Technology Services Bureau includes raw vote totals within each ward. For the purposes of these maps, we will convert the data into percentages for the gubernatorial races. We segment the vote into three groups: Democratic, Republican, and other. To combine all other (non-Democratic, non-Repupblican) votes, we assign the percent of the vote going to "Other" as the vote share not going to the Democratic or Republican candidate.

```r
# Transform data
wi_1990_2000 <- wi_1990_2000 %>%
  dplyr::mutate(pct_gov_dem_1990 = GOVDEM90 / GOVTOT90,
                pct_gov_dem_1994 = GOVDEM94 / GOVTOT94,
                pct_gov_dem_1998 = GOVDEM98 / GOVTOT98,
                pct_gov_rep_1990 = GOVREP90 / GOVTOT90,
                pct_gov_rep_1994 = GOVREP94 / GOVTOT94,
                pct_gov_rep_1998 = GOVREP98 / GOVTOT98,
                pct_gov_other_1990 = 1 - pct_gov_dem_1990 - pct_gov_rep_1990,
                pct_gov_other_1994 = 1 - pct_gov_dem_1994 - pct_gov_rep_1994,
                pct_gov_other_1998 = 1 - pct_gov_dem_1998 - pct_gov_rep_1998)
wi_2002_2010 <- wi_2002_2010 %>%
  dplyr::mutate(pct_gov_dem_2002 = GOVDEM02 / GOVTOT02,
                pct_gov_dem_2006 = GOVDEM06 / GOVTOT06,
                pct_gov_dem_2010 = GOVDEM10 / GOVTOT10,
                pct_gov_rep_2002 = GOVREP02 / GOVTOT02,
                pct_gov_rep_2006 = GOVREP06 / GOVTOT06,
                pct_gov_rep_2010 = GOVREP10 / GOVTOT10,
                pct_gov_other_2002 = 1 - pct_gov_dem_2002 - pct_gov_rep_2002,
                pct_gov_other_2006 = 1 - pct_gov_dem_2006 - pct_gov_rep_2006,
                pct_gov_other_2010 = 1 - pct_gov_dem_2010 - pct_gov_rep_2010)
wi_2012_2020 <- wi_2012_2020 %>%
  dplyr::mutate(pct_gov_dem_2012 = GOVDEM12 / GOVTOT12,
                pct_gov_dem_2014 = GOVDEM14 / GOVTOT14,
                pct_gov_dem_2018 = GOVDEM18 / GOVTOT18,
                pct_gov_rep_2012 = GOVREP12 / GOVTOT12,
                pct_gov_rep_2014 = GOVREP14 / GOVTOT14,
                pct_gov_rep_2018 = GOVREP18 / GOVTOT18,
                pct_gov_other_2012 = 1 - pct_gov_dem_2012 - pct_gov_rep_2012,
                pct_gov_other_2014 = 1 - pct_gov_dem_2014 - pct_gov_rep_2014,
                pct_gov_other_2018 = 1 - pct_gov_dem_2018 - pct_gov_rep_2018)
```

Part 4: Create maps

Because the underlying structure of each gubernatorial election's data set is the same, we can pull the code to generate a map into a reusable function. The function below uses a data set and year as inputs and produces a graph for the specified year as its output. By using a function to generate the map, we ensure that formatting and labels are consistent across years.

```r
# Define function to create maps
create_gubernatorial_map <- function(data, year) {

  field_name <- sprintf('pct_gov_dem_%i', year)
  graph_title <- sprintf('%i Gubernatorial Vote Share by Ward', year)
  caption_title <- paste0('Source: Wisconsin State Legislature,',
                          ' Legislative Technology Services Bureau, ©2019')


  result <- data %>%
    dplyr::select(CNTY_NAME, dplyr::matches(field_name), geometry) %>%
    ggplot() +
    geom_sf(aes_string(color = field_name, fill = field_name), size = 0) +
    geom_sf(data = wi_counties, color = 'black', fill = NA) +
    coord_sf(datum = NA) +
    scale_fill_gradient2('% Democratic',
                         limits = c(0, 1),
                         low = muted('red'),
                         mid = 'white',
                         midpoint = 0.5,
                         high = muted('blue'),
                         labels = scales::percent,
                         guide = guide_colorbar(
                           direction = 'horizontal',
                           barheight = unit(2, units = 'mm'),
                           barwidth = unit(100, units = 'mm'),
                           title.position = 'top')) +
    scale_color_gradient2(limits = c(0, 1),
                          low = muted('red'),
                          mid = 'white',
                          midpoint = 0.5,
                          high = muted('blue'),
                          guide = FALSE) +
    labs(title = graph_title,
    caption = caption_title) +
    theme_classic() +
    theme(axis.text = element_blank(),
          legend.position = 'bottom',
          legend.title.align = 0.5,
          plot.caption = element_text(hjust = 0,
                                      face = 'italic',
                                      color = 'dark gray'),
          plot.title = element_text(hjust = 0.5,
                                    size = 18,
                                    face = 'bold'),
          plot.margin = grid::unit(c(0.5, 0.5, 0.5, 0.5), 'in'))
  result
}

# Store each election's graph as a separate object
wi_gov_1990 <- create_gubernatorial_map(wi_1990_2000, 1990)
wi_gov_1994 <- create_gubernatorial_map(wi_1990_2000, 1994)
wi_gov_1998 <- create_gubernatorial_map(wi_1990_2000, 1998)
wi_gov_2002 <- create_gubernatorial_map(wi_2002_2010, 2002)
wi_gov_2006 <- create_gubernatorial_map(wi_2002_2010, 2006)
wi_gov_2010 <- create_gubernatorial_map(wi_2002_2010, 2010)
wi_gov_2012 <- create_gubernatorial_map(wi_2012_2020, 2012)
wi_gov_2014 <- create_gubernatorial_map(wi_2012_2020, 2014)
wi_gov_2018 <- create_gubernatorial_map(wi_2012_2020, 2018)
```

Part 5: Generate output file

Having created objects containing each of the nine maps, we can write all nine of these maps to a single output file. The output file destination is set using the `here` package to allow for portability across users.

```
# Specify desired output path and file name
file_dest <- here::here('wi_gov_results', 'output', 'wi_gov_results.pdf')

# Create output file
pdf(file = file_dest, width = 8.5, height = 11)
wi_gov_1990
wi_gov_1994
wi_gov_1998
wi_gov_2002
wi_gov_2006
```

```
wi_gov_2010
wi_gov_2012
wi_gov_2014
wi_gov_2018
dev.off()
```