ISOM3390 Winter 2021 - Take Home Exam

Lee Jae Yeol 20308109

- Task 1: Web Scraping
- Task 2: Data Wrangling
- Task 3: Data Visualization

Remember to change the `author:` field on this Rmd file to your name and SID.

In assignment 2, we used several datasets provided in the `maps` package and the data contained in **USPresidential08-16.csv** to create a Choropleth map that shows the US 2016 presidential election result at the county level (except for Alaska and Hawaii).

The 2020 presidential election result was called last month. The Democratic candidate, Joe Biden, flipped 5 states (Michigan, Wisconsin, Arizona, Pennsylvania, and Georgia) won by his incumbent counterpart, Donald Trump, in 2016, and won the election. So, it could be interesting to view how changes in county-level result contributed to the former vice president's triumph in the just-passed election.

Task 1: Web Scraping

Many major U.S.-based news websites have published county-level presidential election results (e.g., **www.nbcnews.com**, **www.usatoday**, etc.). So, we can pull the county-level result data from one of these websites to assemble a dataset for later analyses.

Note that, on these websites, the county-level result data is often organized by state and presented by different webpages. And there's often a pattern existing in the URLs of these webpages.

The following gives a list of sample URLs on **www.nbcnews.com** for webpages that present county-level results by state:

```
"https://www.nbcnews.com/politics/2020-elections/alabama-president-results"
"https://www.nbcnews.com/politics/2020-elections/arizona-president-results"
"https://www.nbcnews.com/politics/2020-elections/arkansas-president-results"
"https://www.nbcnews.com/politics/2020-elections/california-president-results"
"https://www.nbcnews.com/politics/2020-elections/colorado-president-results"
"https://www.nbcnews.com/politics/2020-elections/connecticut-president-results"
"https://www.nbcnews.com/politics/2020-elections/delaware-president-results"
"https://www.nbcnews.com/politics/2020-elections/district-of-columbia-president-results"
...
"https://www.nbcnews.com/politics/2020-elections/west-virginia-president-results"
"https://www.nbcnews.com/politics/2020-elections/wisconsin-president-results"
"https://www.nbcnews.com/politics/2020-elections/wyoming-president-results"
```

To form these URLs, you can first use the `state` dataset in `maps` to get the names of all states in the United States mainland (without Alaska and Hawaii), and then use proper `stringr` functions to concatenate them with strings that capture the common pattern (so you don't need to scrape data for Alaska and Hawaii).

```
# Form the URLs for NBC News webpages that present county-level results
# Name the result url
# You are not allowed to use any for/while/repeat loop in this chunk

state <- as_tibble(map_data("state"))
revised_statenames <- str_replace_all(unique(state$region), "( )", "-")

urls <- str_c("https://www.nbcnews.com/politics/2020-elections/", revised_statenames,"-president-results")
head(urls, n = 10)
```

```
##  [1] "https://www.nbcnews.com/politics/2020-elections/alabama-president-results"
##  [2] "https://www.nbcnews.com/politics/2020-elections/arizona-president-results"
##  [3] "https://www.nbcnews.com/politics/2020-elections/arkansas-president-results"
##  [4] "https://www.nbcnews.com/politics/2020-elections/california-president-results"
##  [5] "https://www.nbcnews.com/politics/2020-elections/colorado-president-results"
##  [6] "https://www.nbcnews.com/politics/2020-elections/connecticut-president-results"
##  [7] "https://www.nbcnews.com/politics/2020-elections/delaware-president-results"
##  [8] "https://www.nbcnews.com/politics/2020-elections/district-of-columbia-president-results"
##  [9] "https://www.nbcnews.com/politics/2020-elections/florida-president-results"
## [10] "https://www.nbcnews.com/politics/2020-elections/georgia-president-results"
```

Once you have the URLs ready, start the Web scraping process with the following skeleton code.

Drop the chunk option `eval=F` if you want to knit the .rmd file.

```
# Provide appropriate starter code
election_res_2020 <- data.frame(state_county = c(NULL), trump = c(NULL), biden = c(NULL), others = c(NULL))
rd <-rsDriver(chromever = "88.0.4324.27", verbose = F)
remdr <- rd[["client"]]

for (i in seq_along(urls)){

    # Provide your code that mimics needed interactions and extracts and parses HTML source code
    # You are not allowed to use any for/while/repeat loop within this for loop
    # You are only allowed to use *apply functions or leverage R functions' vectorization capabilities
    # whenever an iterative logic needs to be implemented
    # Some states (e.g., Arizona, D.C., etc.) have just a few counties
    # so that there's no "SHOW ALL COUNTIES" button on the corresponding webpages
    # To differentiate them between w/o a "SHOW ALL COUNTIES" button, you can test a condition like
    # length(remdr$findElements('css', ... ))!= 0
    # to check whether the current page contains a "SHOW ALL COUNTIES" button or not
    # You may want to use Sys.sleep(2), which suspends execution of R commands
    # for a specified time interval, e.g., 2 seconds in this example
    # after any browser action so as make sure dynamic contents get fully rendered before extraction
    # Points will be deducted if this loop body contains more than 60 lines (after removing these comments)

  remdr$navigate(urls[i])
  if(length(remdr$findElements(using = "css", ".jsx-1765211304.dib.button-press.founders-cond.lh-none.f4.fw
        6.ttu.clear-blue.bg-white.w-100.pv2"))!= 0) {
    loadmore <- remdr$findElement(using = "css", ".jsx-1765211304.dib.button-press.founders-cond.lh-none.f4.
        fw6.ttu.clear-blue.bg-white.w-100.pv2")
    Sys.sleep(2)
    loadmore$clickElement()
    Sys.sleep(2)
  }
  webpage <- remdr$getPageSource() %>% .[[1]] %>% read_html

  countys <- webpage  %>% html_nodes(".publico-txt.truncate.f3.pr2.pr9-m.ls-normal") %>% html_text(trim = TR
        UE) %>% str_replace("100% in$", "") %>% str_to_lower()
  state_county <- str_c(revised_statenames[i],countys, sep = ",")

  votes <- webpage %>% html_nodes(".jsx-3384420229.column-group .jsx-3437879980.dib.number.f2.founders-mono"
        ) %>% html_text(trim = TRUE) %>% str_replace_all(",", "") %>% as.numeric()
  candidates <- webpage %>% html_nodes(".jsx-4189516194.dn.dib-m.mb1")%>% html_text(trim = TRUE)

  len <- length(countys)
  if(candidates[1] == "Donald Trump"){
    election_res_2020 <- rbind(election_res_2020, data.frame(state_county = state_county, trump = c(votes[1:
        len]), biden = c(votes[(len+1):(len*2)]),
                                                 others = c(votes[(len*2+1):(length(votes))] %>% matrix(nro
        w = len, ncol= length(candidates)-2) %>% rowSums())))
  } else if(candidates[2] == "Donald Trump"){
      election_res_2020 <- rbind(election_res_2020, data.frame(state_county = state_county, biden = c(votes[
        1:len]), trump = c(votes[(len+1):(len*2)]),
                                                 others = c(votes[(len*2+1):(length(votes))] %>% matrix(nro
        w = len, ncol= length(candidates)-2) %>% rowSums())))
```

```
  }
}

# Provide your code to organize the scraped data in a tidy data frame named election_res_2020
# Export the data to a same-name csv file (your submission should include this csv file)

election_res_2020 <- as_tibble(election_res_2020)
write.csv(election_res_2020, "election_res_2020.csv")

remdr$close()
rd$server$stop()
```

```
## [1] TRUE
```

```
system("taskkill /im java.exe /f", intern=FALSE, ignore.stdout=FALSE)
```

```
## [1] 0
```

```
election_res_2020
```

```
## # A tibble: 4,593 x 4
##    state_county     trump biden others
##    <chr>            <dbl> <dbl>  <dbl>
##  1 alabama,autauga  19838  7503    429
##  2 alabama,baldwin  83544 24578   1557
##  3 alabama,barbour   5622  4816     80
##  4 alabama,bibb      7525  1986     84
##  5 alabama,blount   24711  2640    237
##  6 alabama,bullock   1146  3446     21
##  7 alabama,butler    5458  3965     65
##  8 alabama,calhoun  35101 15216    666
##  9 alabama,chambers  8753  6365    166
## 10 alabama,cherokee 10583  1624     94
## # ... with 4,583 more rows
```

`election_res_2020` that stores the scraped data should look like the following:

```
# A tibble: 4,588 x 4
   state_county     trump biden others
   <chr>            <dbl> <dbl>  <dbl>
 1 alabama,autauga  19838  7503    429
 2 alabama,baldwin  83544 24578   1557
 3 alabama,barbour   5622  4816     80
 4 alabama,bibb      7525  1986     84
 5 alabama,blount   24711  2640    237
 6 alabama,bullock   1146  3446     21
 7 alabama,butler    5458  3965     65
 8 alabama,calhoun  35101 15216    666
 9 alabama,chambers  8753  6365    166
10 alabama,cherokee 10583  1624     94
# ... with 4,578 more rows
```

As you can see, each row corresponds to one county in this data frame. The `trump` and `biden` columns store the numbers of votes received by the 2 major candidates, while the `others` column stores that received by all remaining candidates.

Note: even if you fail to get exactly the same number of rows and the same set of values displayed above, you can still move forward with your imperfect output, or even a fabricated data frame of the same structure (for example, the 2012 presidential election data contained in USPresidential08-16.csv) to complete the remaining tasks. ***Full points may still be awarded to sensical answers to the remaining problems.***

Task 2: Data Wrangling

Biden flipped 5 states that Trump won in 2016. They are Michigan, Wisconsin, Arizona, Pennsylvania, and Georgia. The following code creates a character vector for the names of the 5 states:

```
flipped_states <- c('arizona', 'georgia', 'michigan', 'pennsylvania', 'wisconsin')
```

We're going to zoom in on *counties of these 5 states* to inspect changes in vote between the 2016 and 2020 elections.

The 2016 election results of the 5 states can be found in the *USPresidential08-16.csv* file. Use the workaround indicated in assignment 2 to augment observations of interest with the `polyname` column from the `maps::county.fips` dataset. Rename this column to `state_county`. Name the resulting data frame `election_res_2016`:

```
# Provide your code to create election_res_2016
# You are not allowed to use any for/while/repeat loop in this chunk
# Use inner_join if you want to join two data frames

election_result16 <- read.csv("USPresidential08-16.csv") %>% select(fips = fips_code, ends_with("2016"), -(o
          th_2016)) %>% inner_join(as.tibble(county.fips), by = "fips") %>% select(ends_with("2016"), state_
          county = polyname)
```

```
## Warning: `as.tibble()` is deprecated as of tibble 2.0.0.
## Please use `as_tibble()` instead.
## The signature and semantics have changed, see `?as_tibble`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
election_res_2016 <- election_result16 %>% filter(str_extract(election_result16$state_county, "[A-z]+") %in%
         flipped_states) %>% as_tibble()


election_res_2016
```

```
## # A tibble: 396 x 4
##    total_2016 dem_2016 gop_2016 state_county
##         <int>    <int>    <int> <chr>
##  1      18467     6431    11112 michigan,delta
##  2       6743      904     5676 pennsylvania,fulton
##  3     284832   169169   106559 pennsylvania,delaware
##  4       3572     2695      841 georgia,hancock
##  5       3577     1186     2343 georgia,seminole
##  6      33848    16050    15871 wisconsin,sauk
##  7      96945    34436    58941 pennsylvania,washington
##  8      14757     6774     7239 michigan,leelanau
##  9       6285      619     5561 georgia,brantley
## 10       3486     1156     2158 michigan,baraga
## # ... with 386 more rows
```

The layout of `election_res_2016` should look like the following:

```
# A tibble: 396 x 4
   total_2016 dem_2016 gop_2016 state_county
        <dbl>    <dbl>    <dbl> <chr>
 1      18467     6431    11112 michigan,delta
 2       6743      904     5676 pennsylvania,fulton
 3     284832   169169   106559 pennsylvania,delaware
 4       3572     2695      841 georgia,hancock
 5       3577     1186     2343 georgia,seminole
 6      33848    16050    15871 wisconsin,sauk
 7      96945    34436    58941 pennsylvania,washington
 8      14757     6774     7239 michigan,leelanau
 9       6285      619     5561 georgia,brantley
10       3486     1156     2158 michigan,baraga
# ... with 386 more rows
```

In the 5 flipped states, there are 4 counties whose names are inconsistent between `election_res_2020` and `election_res_2016`:

In `election_res_2020`, their names are `"georgia,dekalb"`, `"michigan,st. clair"`, `"michigan,st. joseph"`, and `"wisconsin,st. croix"` (they come from NBC News webpages).

while, in `election_res_2016`, they are `"georgia,de kalb"`, `"michigan,st clair"`, `"michigan,st joseph"`, and `"wisconsin,st croix"` (they come from `maps::county.fips`).

Create a data frame that combines county-level results for the 5 states. Name the resulting data frame `election_res_1620`:

```
# Provide your code to create election_res_1620
# You are not allowed to use any for/while/repeat loop in this chunk


election_res_2020$state_county[election_res_2020$state_county == "georgia,dekalb"] = "georgia,de kalb"
election_res_2020$state_county[election_res_2020$state_county == "michigan,st. clair"] = "michigan,st clair"
election_res_2020$state_county[election_res_2020$state_county == "michigan,st. joseph"] = "michigan,st josep
        h"
election_res_2020$state_county[election_res_2020$state_county == "wisconsin,st. croix"] = "wisconsin,st croi
        x"


election_res_1620 <- election_res_2016 %>% inner_join(election_res_2020, by ="state_county") %>% arrange(sta
        te_county) %>% mutate(total_2020 = trump + biden + others, dem_2020 = biden, gop_2020 = trump) %>%
        select(state_county, ends_with("2016"), ends_with("2020")) %>% as_tibble()


election_res_1620
```

```
## # A tibble: 396 x 7
##    state_county     total_2016 dem_2016 gop_2016 total_2020 dem_2020 gop_2020
##    <chr>                 <int>    <int>    <int>      <dbl>    <dbl>    <dbl>
##  1 arizona,apache        18659    12196     5315      35183    23293    11442
##  2 arizona,cochise       43147    15291    25036      60473    23732    35557
##  3 arizona,coconino      44929    25308    16573      73346    44698    27052
##  4 arizona,gila          21398     6746    13672      27678     8943    18377
##  5 arizona,graham        11939     3301     8025      14996     4034    10749
##  6 arizona,greenlee       3243     1092     1892       3688     1182     2433
##  7 arizona,la paz         4931     1318     3381       7460     2236     5129
##  8 arizona,maricopa    1201934   549040   590465    2069475  1040774   995665
##  9 arizona,mohave        74189    16485    54656     104705    24831    78535
## 10 arizona,navajo        35409    15362    18165      51783    23383    27657
## # ... with 386 more rows
```

The layout of `election_res_1620` should look like the following:

```
# A tibble: 396 x 7
   state_county     total_2016 dem_2016 gop_2016 total_2020 dem_2020 gop_2020
   <chr>                 <dbl>    <dbl>    <dbl>      <dbl>    <dbl>    <dbl>
 1 arizona,apache        18659    12196     5315      35183    23293    11442
 2 arizona,cochise       43147    15291    25036      60473    23732    35557
 3 arizona,coconino      44929    25308    16573      73346    44698    27052
 4 arizona,gila          21398     6746    13672      27678     8943    18377
 5 arizona,graham        11939     3301     8025      14996     4034    10749
 6 arizona,greenlee       3243     1092     1892       3688     1182     2433
 7 arizona,la paz         4931     1318     3381       7460     2236     5129
 8 arizona,maricopa    1201934   549040   590465    2069475  1040774   995665
 9 arizona,mohave        74189    16485    54656     104705    24831    78535
10 arizona,navajo        35409    15362    18165      51783    23383    27657
# ... with 386 more rows
```

where the column `total_2020` contains the total numbers of votes received at the county level in the 2020 election.

Based on `election_res_1620`, please refer to the **split-apply-combine data analysis workflow** to create a data frame that summarizes the total numbers of votes received by both parties at the state level. Name the resulting data frame `election_res_1620_state`, which should have a layout that looks like the following:

```
# A tibble: 20 x 4
   state        party year      vote
   <chr>        <chr> <chr>    <dbl>
 1 Arizona      dem   2016    936250
 2 Arizona      dem   2020   1672143
 3 Arizona      gop   2016   1021154
 4 Arizona      gop   2020   1661686
 5 Georgia      dem   2016   1837300
 6 Georgia      dem   2020   2473633
 7 Georgia      gop   2016   2068623
 8 Georgia      gop   2020   2461854
 9 Michigan     dem   2016   2267373
10 Michigan     dem   2020   2804040
11 Michigan     gop   2016   2279210
12 Michigan     gop   2020   2649852
13 Pennsylvania dem   2016   2844705
14 Pennsylvania dem   2020   3459923
15 Pennsylvania gop   2016   2912941
16 Pennsylvania gop   2020   3378263
17 Wisconsin    dem   2016   1382210
18 Wisconsin    dem   2020   1630673
19 Wisconsin    gop   2016   1409467
20 Wisconsin    gop   2020   1610065
```

```
# Provide your code to create election_res_1620_state
# You are not allowed to use any for/while/repeat loop in this chunk


election_res_1620_state <- election_res_1620
name_states <- str_extract(election_res_1620_state$state_county, "[A-z]+")
str_sub(name_states,1,1) <- str_sub(name_states,1,1) %>% str_to_upper()
election_res_1620_state$state_county <- name_states


election_res_1620_state <- election_res_1620_state %>% group_by(state_county) %>% summarise("dem/2016" = sum
          (dem_2016) , "dem/2020" = sum(dem_2020), "gop/2016" = sum(gop_2016), "gop/2020" = sum(gop_2020)) %
          >% pivot_longer(cols = -(state_county), names_to = "party", values_to ="vote") %>% separate(col =
           party, into = c("party", "year"))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
election_res_1620_state
```

```
## # A tibble: 20 x 4
##    state_county party year     vote
##    <chr>        <chr> <chr>   <dbl>
##  1 Arizona      dem   2016   936250
##  2 Arizona      dem   2020  1672143
##  3 Arizona      gop   2016  1021154
##  4 Arizona      gop   2020  1661686
##  5 Georgia      dem   2016  1837300
##  6 Georgia      dem   2020  2473633
##  7 Georgia      gop   2016  2068623
##  8 Georgia      gop   2020  2461854
##  9 Michigan     dem   2016  2267373
## 10 Michigan     dem   2020  2804040
## 11 Michigan     gop   2016  2279210
## 12 Michigan     gop   2020  2649852
## 13 Pennsylvania dem   2016  2844705
## 14 Pennsylvania dem   2020  3459923
## 15 Pennsylvania gop   2016  2912941
## 16 Pennsylvania gop   2020  3378263
## 17 Wisconsin    dem   2016  1382210
## 18 Wisconsin    dem   2020  1630673
## 19 Wisconsin    gop   2016  1409467
## 20 Wisconsin    gop   2020  1610065
```

Task 3: Data Visualization

With all necessary data ready, first create 2 Choropleth maps for the 2016 and 2020 election results in one figure as shown by the first image posted on Canvas.

As you can see, only the 5 flipped states are colored. And colors represent differences in percentage of votes received by the 2 parties (dem vs. gop).

You need to use geospatial data available in `maps::state` and `maps::county` to draw states' and counties' outlines. Note that `maps::county` uses the same names for the aforementioned 4 counties as `maps::county.fips`.

Suppose that `p` is the name you to this `ggplot` object. You can directly use the code provided at the end of the following code chunk to style the plot.

Drop the chunk option `eval=F` if you want to knit the .rmd file.

```
#%>% group_by(state_county) %>% summarise("dem/2016" = sum(dem_2016) , "dem/2020" = sum(dem_2020), "gop/201
          6" = sum(gop_2016), "gop/2020" = sum(gop_2020))
# Provide your code to wrangle the data and create the plot
# You are not allowed to use any for/while/repeat loop in this chunk
# Use inner_join if you want to join two data frames

state <- as_tibble(map_data("state"))
county <- as_tibble(map_data("county"))
county <- county %>% unite(col = state_county, region, subregion, sep = ",")
election_res_1620
```

```
## # A tibble: 396 x 7
##    state_county     total_2016 dem_2016 gop_2016 total_2020 dem_2020 gop_2020
##    <chr>                 <int>    <int>    <int>      <dbl>    <dbl>    <dbl>
##  1 arizona,apache        18659    12196     5315      35183    23293    11442
##  2 arizona,cochise       43147    15291    25036      60473    23732    35557
##  3 arizona,coconino      44929    25308    16573      73346    44698    27052
##  4 arizona,gila          21398     6746    13672      27678     8943    18377
##  5 arizona,graham        11939     3301     8025      14996     4034    10749
##  6 arizona,greenlee       3243     1092     1892       3688     1182     2433
##  7 arizona,la paz         4931     1318     3381       7460     2236     5129
##  8 arizona,maricopa    1201934   549040   590465    2069475  1040774   995665
##  9 arizona,mohave        74189    16485    54656     104705    24831    78535
## 10 arizona,navajo        35409    15362    18165      51783    23383    27657
## # ... with 386 more rows
```

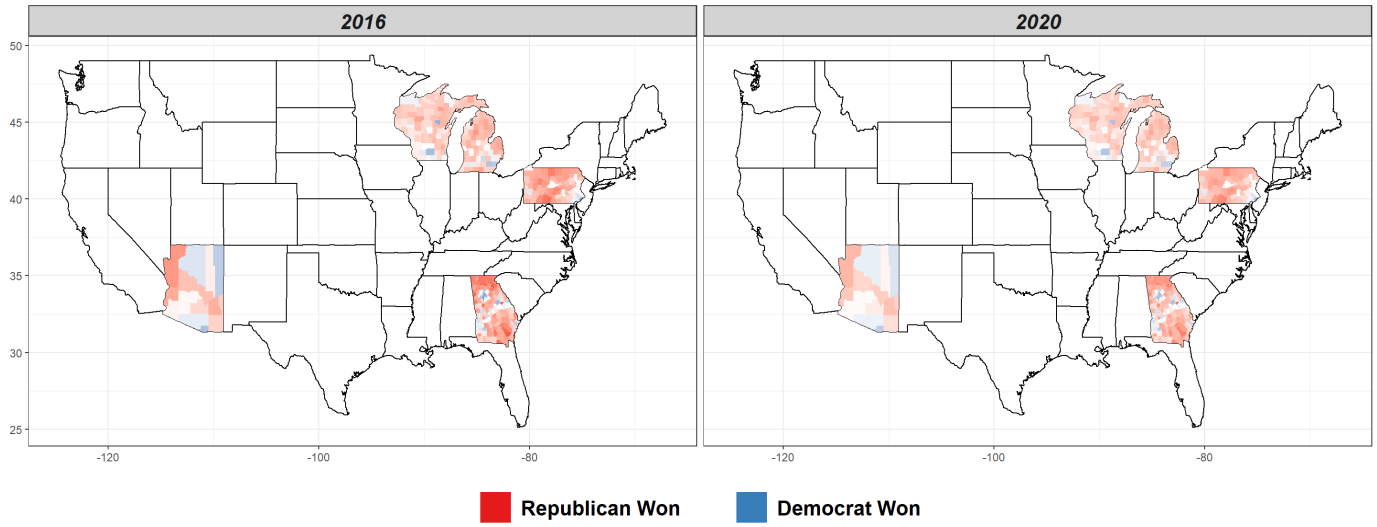
```
polygon_data <- election_res_1620  %>% mutate(percentage_change_2016 = (dem_2016 - gop_2016) / total_2016, p
          ercentage_change_2020 = (dem_2016 - gop_2016) / total_2020) %>% select(state_county, starts_with(
          "percentage")) %>%pivot_longer(cols = -(state_county), names_to = "year", values_to = "percent_cha
          nge" ) %>% inner_join(county, by = "state_county")
polygon_data$year <-str_replace(polygon_data$year, ".+_", "")
polygon_data
```

```
## # A tibble: 21,860 x 7
##    state_county   year  percent_change  long   lat group order
##    <chr>          <chr>          <dbl> <dbl> <dbl> <dbl> <int>
##  1 arizona,apache 2016           0.369 -109.  36.0    68  3160
##  2 arizona,apache 2016           0.369 -109.  35.0    68  3161
##  3 arizona,apache 2016           0.369 -109.  34.6    68  3162
##  4 arizona,apache 2016           0.369 -109.  33.8    68  3163
##  5 arizona,apache 2016           0.369 -109.  33.8    68  3164
##  6 arizona,apache 2016           0.369 -109.  33.8    68  3165
##  7 arizona,apache 2016           0.369 -109.  33.8    68  3166
##  8 arizona,apache 2016           0.369 -109.  33.8    68  3167
##  9 arizona,apache 2016           0.369 -109.  33.8    68  3168
## 10 arizona,apache 2016           0.369 -109.  33.7    68  3169
## # ... with 21,850 more rows
```

```
p <- ggplot() + geom_polygon(state, mapping = aes(long, lat, group = group),color = "black", fill = NA) + ge
          om_polygon(polygon_data, mapping = aes(long, lat, group = group, fill = percent_change)) + facet_w
          rap(~year)


p + ggtitle("Flipped States: 2016 VS. 2020 Presidential Election Wn") + theme_bw() +
  scale_fill_gradient2(name=NULL, limits = c(-1, 1),
                       low = "#e41a1c", high = "#377eb8",
                       breaks = c(-1, 1), labels = c("Republican Won        ", "Democrat Won")) +
  labs(x = NULL, y = NULL) +
  theme(legend.position = "bottom",
          strip.background = element_rect(fill="lightgray", size= 0.8),
          plot.title = element_text(size = 20, face = "bold"),
          strip.text.x = element_text(size = 16, face = "bold.italic"),
          legend.text = element_text(size = 16, face = "bold"),
          legend.spacing.x = unit(0.5, "line"),
          legend.key.size = unit(0.9, "cm")) +
    guides(fill = guide_legend(title.position = "top", title.hjust = 0.5))
```

Flipped States: 2016 VS. 2020 Presidential Election



Please also create 2 bar plots in one figure as shown by the second image posted on Canvas.

Suppose that `h` is the name you to the second `ggplot` object. You can directly use the code provided at the end of the following code chunk to style the plot.

Drop the chunk option `eval=F` if you want to knit the .rmd file.

```
# Provide your code to create the plot
# You are not allowed to use any for/while/repeat loop in this chunk

h <- ggplot(election_res_1620_state, aes(x = c(state_county), y = vote, fill = party)) + geom_bar(stat = 'id
          entity', position = "dodge")  + facet_wrap(~year)



h + ggtitle("Flipped States: 2016 VS. 2020 Presidential Election Wn") + theme_bw() +
  scale_y_continuous(breaks = c(0, 1000000, 2000000, 3000000),
                     labels = c("0", "1,000", "2,000", "3,000")) +
  scale_fill_manual(name=NULL, values = c("dem" = "#377eb8", "gop" = "#e41a1c"),
                    labels = c("gop"= "Republican Party", "dem"="Democrat Party" )) +
  labs(x = NULL, y = "No. of votesWn(in thousands)") +
  theme(legend.position = "bottom",
        strip.background = element_rect(fill="lightgray", size= 0.8),
        plot.title = element_text(size = 20, face = "bold"),
        strip.text.x = element_text(size = 16, face = "bold.italic"),
        legend.text = element_text(size = 16, face = "bold"),
        legend.spacing.x = unit(0.5, "line"),
        legend.key.size = unit(0.9, "cm"),
        axis.title.y = element_text(face="bold.italic", size=18),
        axis.text.x = element_text(size = 14, face="italic"),
        axis.text.y = element_text(size = 14, face="italic")
  ) + guides(fill = guide_legend(title.position = "top", title.hjust = 0.5))
```