Project 3

Diego Correa

9/8/2020

Overview

Loading data

```
library(tidyverse, quietly = TRUE)

#Presidential National Toplines
url1 <- ’https://projects.fivethirtyeight.com/2020-general-data/presidential_national_toplines_2020.csv
#Presidential State Toplines
url2 <- ’https://projects.fivethirtyeight.com/2020-general-data/presidential_state_toplines_2020.csv’
#Presidential EV Probabilities
url3 <- ’https://projects.fivethirtyeight.com/2020-general-data/presidential_ev_probabilities_2020.csv’
#Presidential Scenario Analysis
url4 <- ’https://projects.fivethirtyeight.com/2020-general-data/presidential_scenario_analysis_2020.csv
#Economic Index
url5 <- ’https://projects.fivethirtyeight.com/2020-general-data/economic_index.csv’

#Saving data into variables
dfPNT <- read.csv(file = url1)
dfPST <- read.csv(file = url2)
dfPEP <- read.csv(file = url3)
dfPSA <- read.csv(file = url4)
dfEI <- read.csv(file = url5)
```

Observing & Cleaning

```
str(dfEI)
```

```
## ’data.frame':    749 obs. of  15 variables:
##  $ cycle           : int  2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 ...
##  $ branch          : chr  "President" "President" "President" "President" ...
##  $ model           : chr  "polls-plus" "polls-plus" "polls-plus" "polls-plus" ...
##  $ modeldate       : chr  "9/15/2020" "9/15/2020" "9/15/2020" "9/15/2020" ...
##  $ candidate_inc   : chr  "Trump" "Trump" "Trump" "Trump" ...
##  $ candidate_chal  : chr  "Biden" "Biden" "Biden" "Biden" ...
##  $ candidate_3rd   : logi  NA NA NA NA NA NA ...
##  $ current_zscore  : num  0.846 -1.979 -2.057 -3.26 0.603 ...
##  $ projected_zscore: num  0.749 -1.377 -1.608 -2.682 0.673 ...
##  $ projected_hi    : num  1.409 -0.717 -0.948 -2.022 1.333 ...
##  $ projected_lo    : num  0.0891 -2.0368 -2.2681 -3.3417 0.0131 ...
##  $ category        : chr  "stock market" "spending" "manufacturing" "jobs" ...
##  $ indicator       : chr  "S&P 500" "Personal consumption expenditures" "Industrial production" "Non
##  $ timestamp       : chr  "07:06:02 15 Sep 2020" "07:06:02 15 Sep 2020" "07:06:02 15 Sep 2020" "07:0
##  $ simulations     : int  40000 40000 40000 40000 40000 40000 40000 40000 40000 40000 ...
```

```
# str(dfPNT)
# str(dfPST)
# str(dfPEP)
# str(dfPSA)
```

When we look at the dataframes, we see the the modeldate and the timestamp fields are stored as character types. Additionally, the timestamp field contains multplie whitespaces. Before dumping the data into the MySQL database, we need to clean up the fields.

As the same process needs to be done to each data frame, a function was created to address preform the necessary cleansing mentioned above. We do this with the help of the lubridate package.

```
library(lubridate, quietly = TRUE)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
datetime.this <- function(df){
  df$modeldate <- mdy(df$modeldate)

  df$timestamp <- str_replace_all(df$timestamp, ' {2}', ' ')

  x <- data.frame(str_split(df$timestamp, ':| ', n = 4, simplify = TRUE))

  x <- x %>%
    mutate(date = dmy(X4))

  x <- x %>%
    mutate(timestamp = as_datetime(paste(date, as.integer(X1), as.integer(X2), as.integer(X3),
                                         sep = ' ')))

  df$timestamp <- x$timestamp

  return(df)
}


dfEI <- datetime.this(dfEI)
dfPST <- datetime.this(dfPST)
dfPNT <- datetime.this(dfPNT)
dfPEP <- datetime.this(dfPEP)
dfPSA <- datetime.this(dfPSA)
```

```r
str(dfEI)
```

```
## 'data.frame':    749 obs. of  15 variables:
##  $ cycle           : int  2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 ...
##  $ branch          : chr  "President" "President" "President" "President" ...
##  $ model           : chr  "polls-plus" "polls-plus" "polls-plus" "polls-plus" ...
##  $ modeldate       : Date, format: "2020-09-15" "2020-09-15" ...
##  $ candidate_inc   : chr  "Trump" "Trump" "Trump" "Trump" ...
##  $ candidate_chal  : chr  "Biden" "Biden" "Biden" "Biden" ...
##  $ candidate_3rd   : logi  NA NA NA NA NA NA ...
##  $ current_zscore  : num  0.846 -1.979 -2.057 -3.26 0.603 ...
##  $ projected_zscore: num  0.749 -1.377 -1.608 -2.682 0.673 ...
##  $ projected_hi    : num  1.409 -0.717 -0.948 -2.022 1.333 ...
##  $ projected_lo    : num  0.0891 -2.0368 -2.2681 -3.3417 0.0131 ...
##  $ category        : chr  "stock market" "spending" "manufacturing" "jobs" ...
##  $ indicator       : chr  "S&P 500" "Personal consumption expenditures" "Industrial production" "Non
##  $ timestamp       : POSIXct, format: "2020-09-15 07:06:02" "2020-09-15 07:06:02" ...
##  $ simulations     : int  40000 40000 40000 40000 40000 40000 40000 40000 40000 40000 ...
```

Connecting and storing to db

```r
#storing the data in google cloud database
library(RMySQL, quietly = TRUE)
con <- dbConnect(MySQL(),
                 user = 'root',
                 host = '35.225.135.85',
                 dbname = 'DATA607')

dbWriteTable(con, 'Presidential_National_Toplines', dfPNT, row.names = FALSE, overwrite = TRUE)
```

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

```r
dbWriteTable(con, 'Presidential_State_Toplines', dfPST, row.names = FALSE, overwrite = TRUE)
```

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

```r
dbWriteTable(con, 'Presidential_EV_Probabilities', dfPEP, row.names = FALSE, overwrite = TRUE)
```

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

```r
dbWriteTable(con, 'Presidential_Scenario_Analysis', dfPSA, row.names = FALSE, overwrite = TRUE)
```

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

```r
dbWriteTable(con, 'Economic_Index', dfEI, row.names = FALSE, overwrite = TRUE)
```

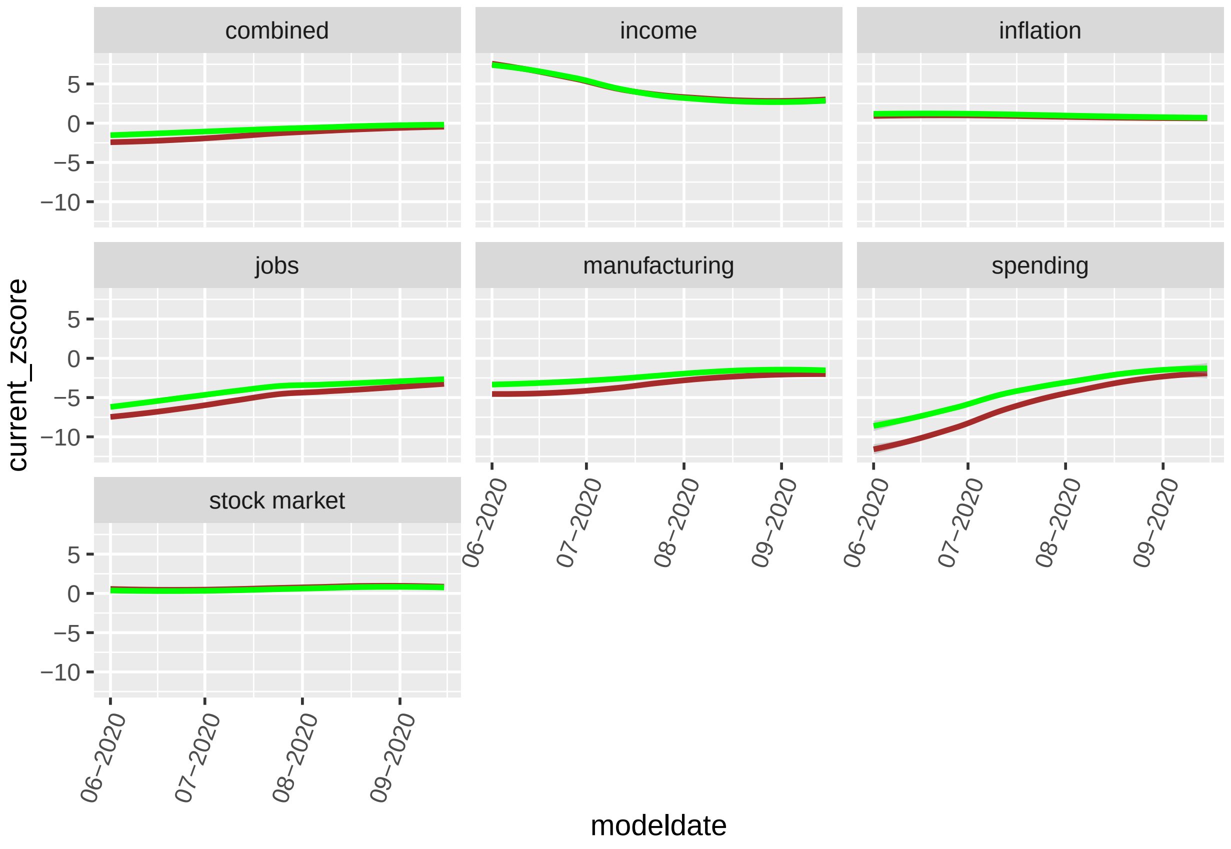
```
## [1] TRUE
```

Economic Index

The Economic Index table contains economic indicators that serve as inputs to the forecast. For more information on these indicators, see this post. The economic indexes were collected from the Federal Reserve Bank Of St. Louis and the stock prices data from Yahoo Finance. This sheet contains the following additional columns:

```r
res <- dbGetQuery(con, 'select modeldate, category, current_zscore, projected_zscore
                  from Economic_Index;')
res$modeldate <- ymd(res$modeldate)

current <- ggplot(data = res) +
  geom_smooth(mapping = aes(x = modeldate, y = current_zscore), color = 'brown') +
  geom_smooth(mapping = aes(x = modeldate, y = projected_zscore), color = 'green') +
  facet_wrap(~category) +
  theme(axis.text.x = element_text(angle = 70, hjust = 1))
current + scale_x_date(date_labels = '%m-%Y')
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Presidential EV Probabilities table

The Presidential EV Porbabilities table contains the forecasted chances of every possible Electoral College outcome. This table contains only one most recent day's electoral college simulations.

```r
str(dfPEP)
```

```
## 'data.frame':    539 obs. of  13 variables:
##  $ cycle          : int  2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 ...
##  $ branch         : chr  "President" "President" "President" "President" ...
##  $ model          : chr  "polls-plus" "polls-plus" "polls-plus" "polls-plus" ...
##  $ modeldate      : Date, format: "2020-09-15" "2020-09-15" ...
##  $ candidate_inc  : chr  "Trump" "Trump" "Trump" "Trump" ...
##  $ candidate_chal: chr  "Biden" "Biden" "Biden" "Biden" ...
##  $ candidate_3rd  : logi  NA NA NA NA NA NA ...
##  $ evprob_inc     : num  0.00162 0.00185 0.0029 0.00175 0.00115 ...
##  $ evprob_chal    : num  0.0 0.0 2.5e-05 0.0 0.0 2.5e-05 0.0 2.5e-05 2.5e-05 7.5e-05 ...
##  $ evprob_3rd     : logi  NA NA NA NA NA NA ...
##  $ total_ev       : int  99 98 97 96 95 94 93 92 91 90 ...
##  $ timestamp      : POSIXct, format: "2020-09-15 07:06:02" "2020-09-15 07:06:02" ...
##  $ simulations    : int  40000 40000 40000 40000 40000 40000 40000 40000 40000 40000 ...
```

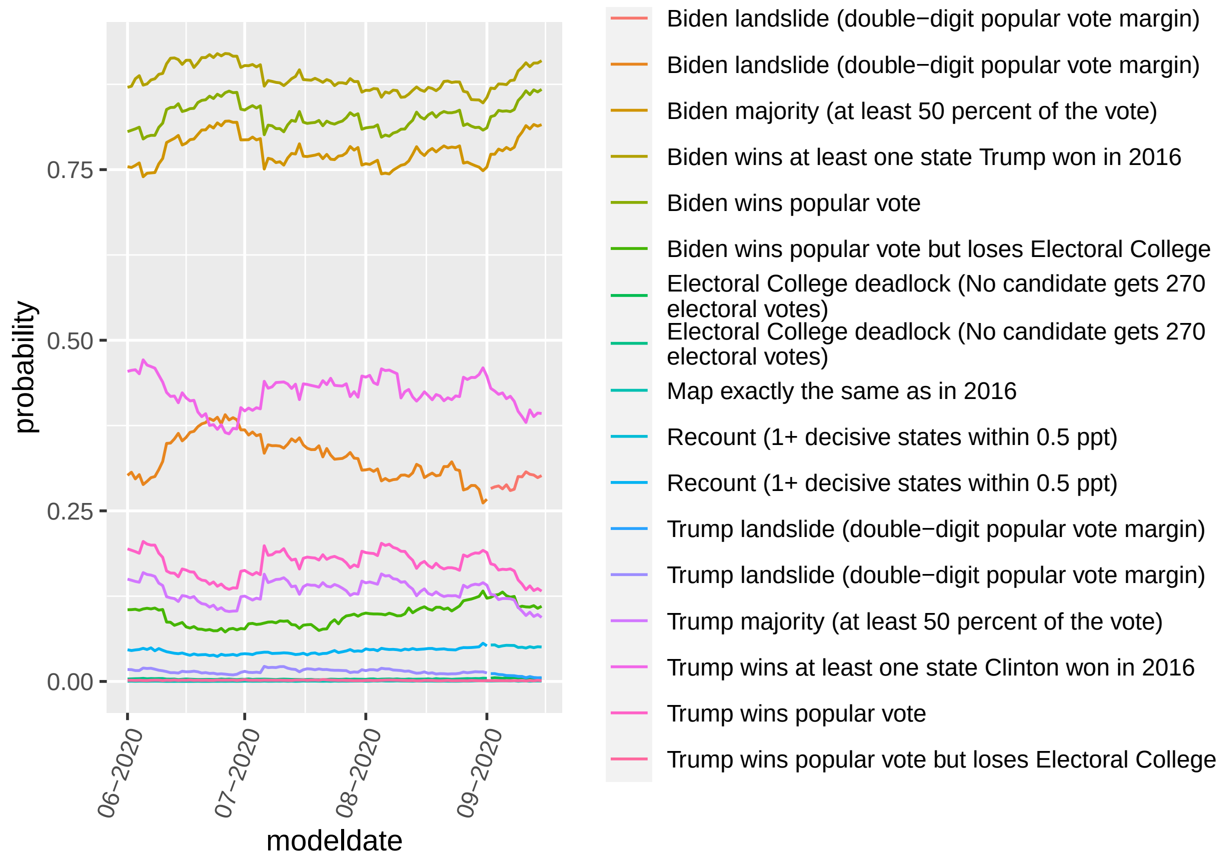
Presidential Scenario Analysis

The Presidential Scenario Analysis contains the forecasted chances of various possible election outcome scenarios.

```r
str(dfPSA)
```

```
## 'data.frame':    1391 obs. of  12 variables:
##  $ cycle               : int  2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 ...
##  $ branch              : chr  "President" "President" "President" "President" ...
##  $ model               : chr  "polls-plus" "polls-plus" "polls-plus" "polls-plus" ...
##  $ modeldate           : Date, format: "2020-09-15" "2020-09-15" ...
##  $ candidate_inc       : chr  "Trump" "Trump" "Trump" "Trump" ...
##  $ candidate_chal      : chr  "Biden" "Biden" "Biden" "Biden" ...
##  $ candidate_3rd       : logi  NA NA NA NA NA NA ...
##  $ scenario_id         : int  9 8 6 5 4 3 2 1 14 13 ...
##  $ probability         : num  0.81588 0.0938 0.11017 0.00115 0.86762 ...
##  $ scenario_description: chr  "Biden majority (at least 50 percent of the vote)" "Trump majority (at
##  $ timestamp           : POSIXct, format: "2020-09-15 07:06:02" "2020-09-15 07:06:02" ...
##  $ simulations         : int  40000 40000 40000 40000 40000 40000 40000 40000 40000 40000 ...
```

```r
res <- dbGetQuery(con, 'select modeldate, scenario_description, probability from
                  Presidential_Scenario_Analysis;')

res$modeldate <- ymd(res$modeldate)

probScenario <- ggplot(data = res) +
  geom_line(mapping = aes(modeldate, y = probability, color = scenario_description)) +
  theme(axis.text.x = element_text(angle = 70, hjust = 1))

probScenario + scale_x_date(date_labels = '%m-%Y') + scale_color_discrete(labels = function(x) str_wrap
```



```
#not very pretty to look at
```

State Toplines

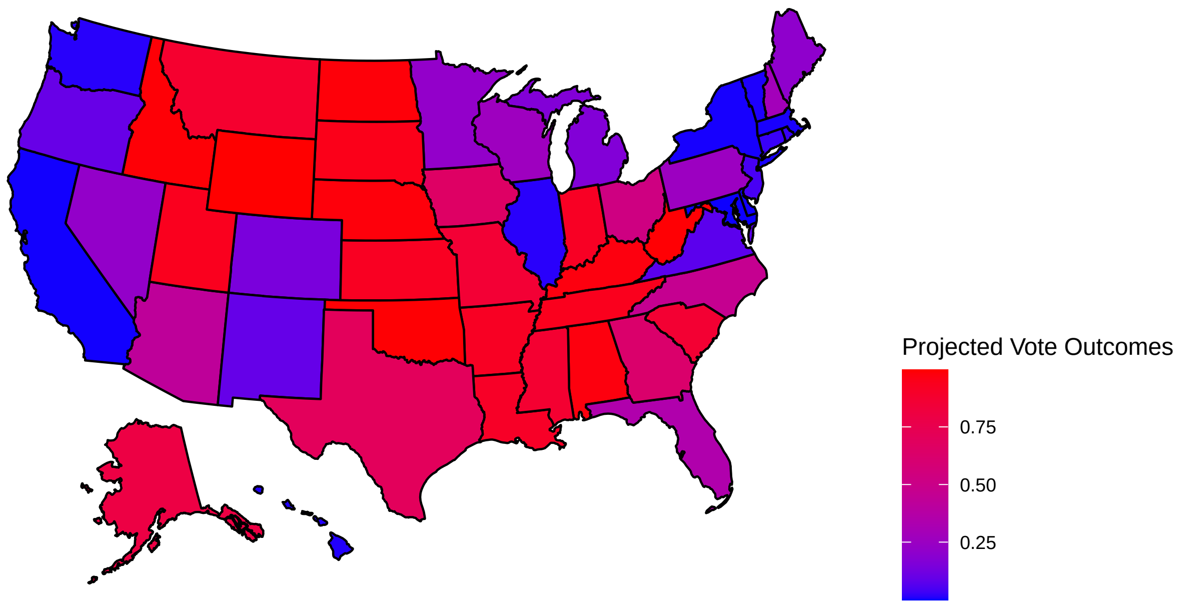
Looking at red and blue states

```
res <- dbGetQuery(con, 'select state, avg(winstate_inc) Inc from Presidential_State_Toplines group by st

dfStateWin <- res

library(usmap)

plot_usmap(data = dfStateWin, values = 'Inc') +
  scale_fill_continuous(
    low = 'blue', high = 'red', name = 'Projected Vote Outcomes'
  ) + theme(legend.position = 'right')
```

```
## Warning: Use of `map_df$x` is discouraged. Use `x` instead.
```

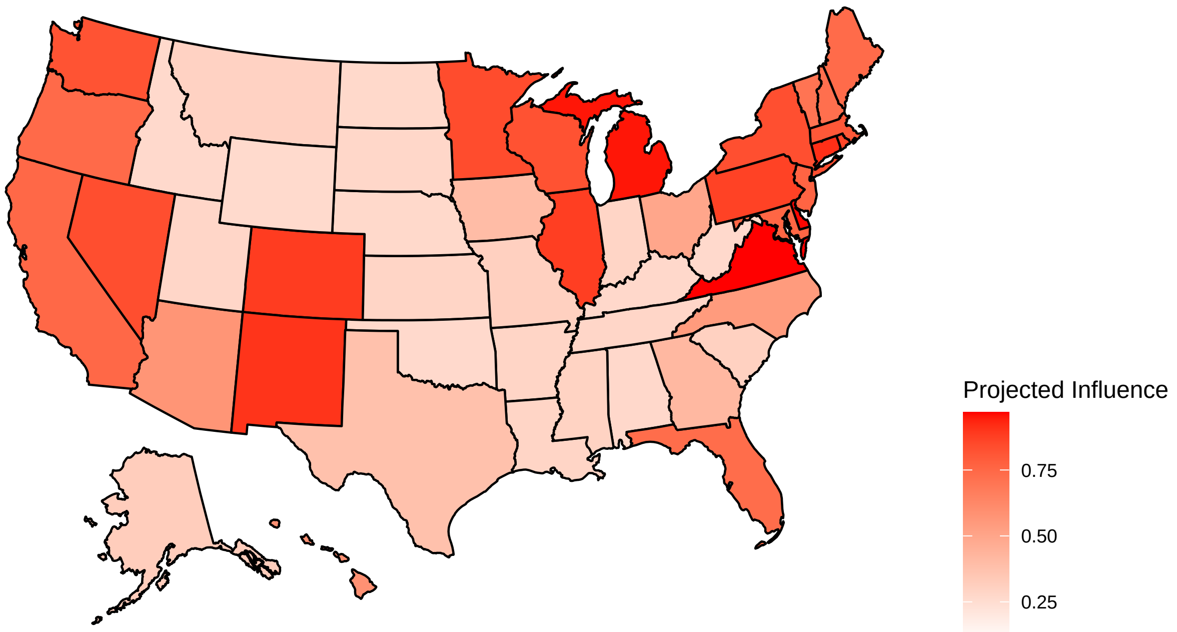
```
## Warning: Use of `map_df$y` is discouraged. Use `y` instead.
```

```
## Warning: Use of `map_df$group` is discouraged. Use `group` instead.
```



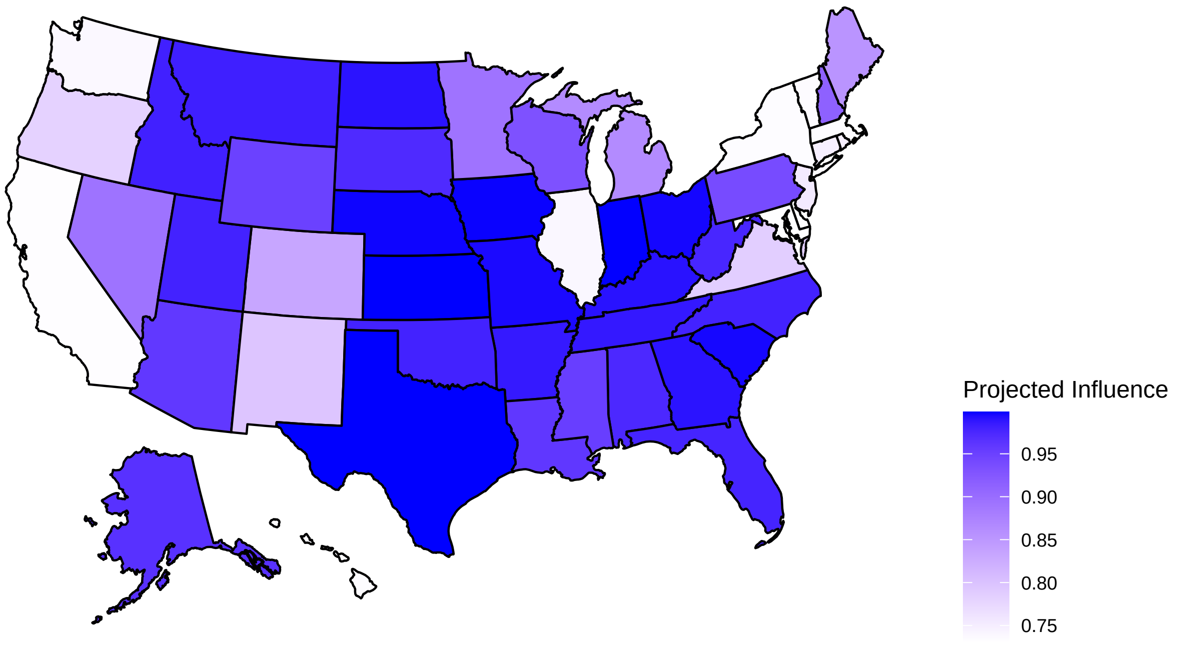
```
res <- dbGetQuery(con, 'select state, avg(win_EC_if_win_state_inc
) Inc, avg(win_EC_if_win_state_chal) Chal from Presidential_State_Toplines group by state;')

dfStateInfluence <- res

plot_usmap(data = dfStateInfluence, values = 'Inc') +
  scale_fill_continuous(
    low = 'white', high = 'red', name = 'Projected Influence'
  ) + theme(legend.position = 'right')
```

```
## Warning: Use of `map_df$x` is discouraged. Use `x` instead.

## Warning: Use of `map_df$y` is discouraged. Use `y` instead.

## Warning: Use of `map_df$group` is discouraged. Use `group` instead.
```



```r
plot_usmap(data = dfStateInfluence, values = 'Chal') +
  scale_fill_continuous(
    low = 'white', high = 'blue', name = 'Projected Influence'
  ) + theme(legend.position = 'right')
```

```
## Warning: Use of `map_df$x` is discouraged. Use `x` instead.

## Warning: Use of `map_df$y` is discouraged. Use `y` instead.

## Warning: Use of `map_df$group` is discouraged. Use `group` instead.
```



separating red and blue states based on average probability of winning state

```
# Obtaining the average
res1 <- dbGetQuery(con, 'select state, round(avg(winstate_inc), 4) AVGWinStateInc,
                   case
                   when avg(winstate_inc) > 0.5 then "Red"
                   when avg (winstate_inc) < 0.5 then "Blue"
                   else "Tie"
                   end "Color"
                   from Presidential_State_Toplines group by state;')

redStates <- res1 %>%
  filter(Color == 'Red')
redStates <- data.frame(redStates$state)
colnames(redStates) <- 'states'

blueStates <- res1 %>%
  filter(Color == 'Blue')
blueStates <- data.frame(blueStates$state)
colnames(blueStates) <- 'states'

dbWriteTable(con, 'redStates', redStates, overwrite = TRUE)
```

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

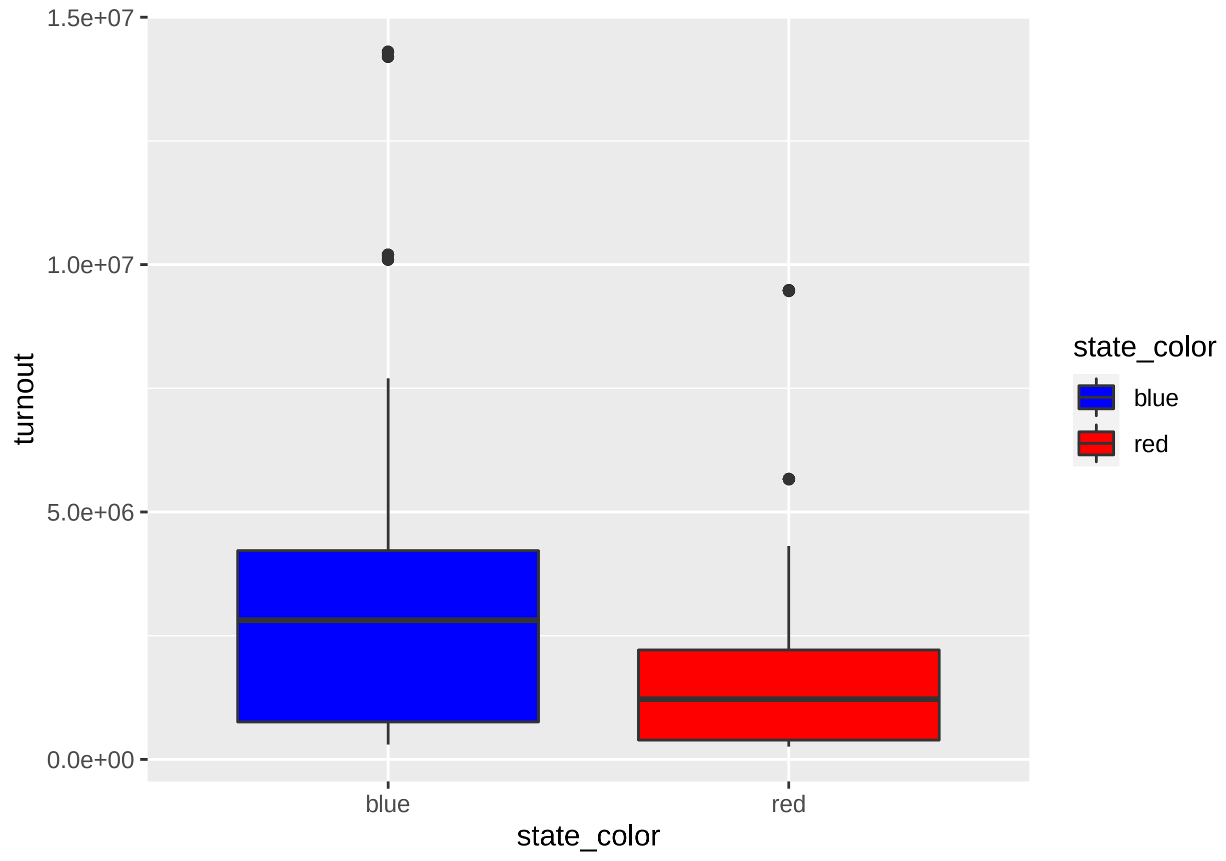
```r
dbWriteTable(con, 'blueStates', blueStates, overwrite = TRUE)
```

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

```r
res2 <- dbGetQuery(con, 'select state, state_turnout, "red" from Presidential_State_Toplines
                          where state in (select states from redStates);')
redTO <- res2
names(redTO) <- c('state', 'turnout', 'state_color')

res3 <- dbGetQuery(con, 'select state, state_turnout, "blue" from Presidential_State_Toplines
                          where state in (select states from blueStates);')
blueTO <- res3
names(blueTO) <- c('state', 'turnout', 'state_color')

df1 <- rbind(redTO, blueTO)

bp <- ggplot(data = df1, aes(x = state_color, y = turnout, fill = state_color)) +
  geom_boxplot()

bp + scale_fill_manual(values = c('blue', 'red'))
```

Voter Turnout by State

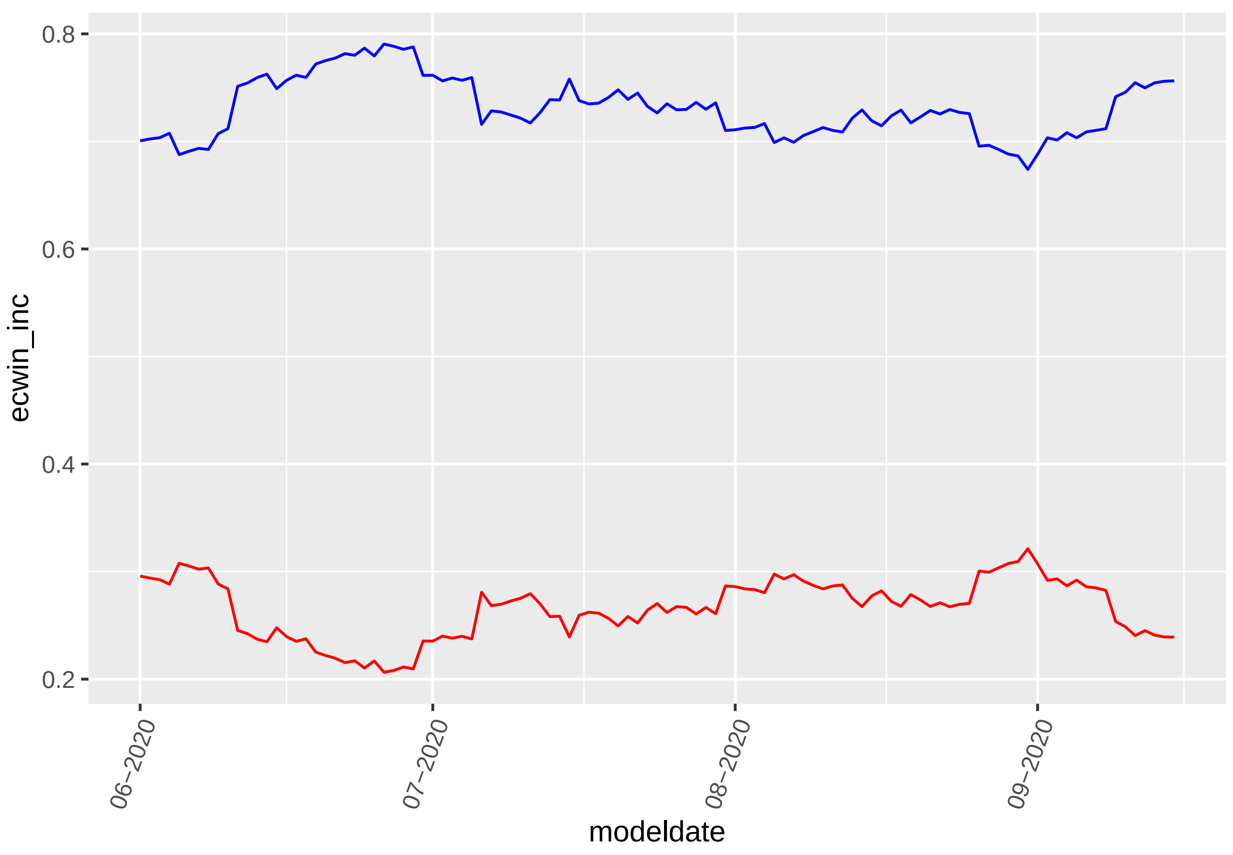
```
## Warning: Removed 5432 rows containing non-finite values (stat_boxplot).
```



Presidential National Toplines

contains the final national topline on each day.Chances to win electoral votes (Biden in blue, Trump in red)

```
res <- dbGetQuery(con, 'select modeldate, ecwin_inc, ecwin_chal from
                  Presidential_National_Toplines;')
res$modeldate <- ymd(res$modeldate)


probEWin <- ggplot(data = res) +
  geom_line(mapping = aes(x = modeldate, y = ecwin_inc), color = 'red') +
  geom_line(mapping = aes(x = modeldate, y = ecwin_chal), color = 'blue') +
  theme(axis.text.x =  element_text(angle = 70, hjust = 1))

probEWin + scale_x_date(date_labels = '%m-%Y')
```



0.8
0.6
0.4
0.2
ecwin_inc
06–2020
07–2020
08–2020
09–2020
modeldate