Project 3

Diego Correa

9/8/2020

Overview

Loading data

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

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
#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':    735 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  "9/13/2020" "9/13/2020" "9/13/2020" "9/13/2020" ...
##  $ 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  0.761 -1.979 -2.057 -3.26 0.646 ...
##  $ projected_zscore: num  0.664 -1.364 -1.602 -2.665 0.718 ...
##  $ projected_hi    : num  1.336 -0.692 -0.93 -1.992 1.39 ...
##  $ projected_lo    : num  -0.00856 -2.0362 -2.27472 -3.33678 0.04533 ...
##  $ category        : chr  "stock market" "spending" "manufacturing" "jobs" ...
##  $ indicator       : chr  "S&P 500" "Personal consumption expenditures" "Industrial production" "Nonf
##  $ timestamp       : chr  "10:33:03 13 Sep 2020" "10:33:03 13 Sep 2020" "10:33:03 13 Sep 2020" "10:33
##  $ 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)

str(dfEI)
```

```
## 'data.frame':    735 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: "2020-09-13" "2020-09-13" ...
##  $ 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  0.761 -1.979 -2.057 -3.26 0.646 ...
##  $ projected_zscore: num  0.664 -1.364 -1.602 -2.665 0.718 ...
##  $ projected_hi    : num  1.336 -0.692 -0.93 -1.992 1.39 ...
##  $ projected_lo    : num  -0.00856 -2.0362 -2.27472 -3.33678 0.04533 ...
##  $ category        : chr  "stock market" "spending" "manufacturing" "jobs" ...
##  $ indicator       : chr  "S&P 500" "Personal consumption expenditures" "Industrial production" "Non
##  $ timestamp       : POSIXct, format: "2020-09-13 10:33:03" "2020-09-13 10:33:03" ...
##  $ simulations     : int  40000 40000 40000 40000 40000 40000 40000 40000 40000 40000 ...
```

Connecting and storing to db

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

summary(con)
```

```
## <MySQLConnection:0,0>
##   User:   root
##   Host:   35.225.135.85
##   Dbname: DATA607
##   Connection type: 35.225.135.85 via TCP/IP
##
## Results:
```

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

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

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

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

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

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

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

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

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

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

```
dbListTables(con)
```

```
## [1] "Economic_Index"                 "Presidential_EV_Probabilities"
## [3] "Presidential_National_Toplines" "Presidential_Scenario_Analysis"
## [5] "Presidential_State_Toplines"
```

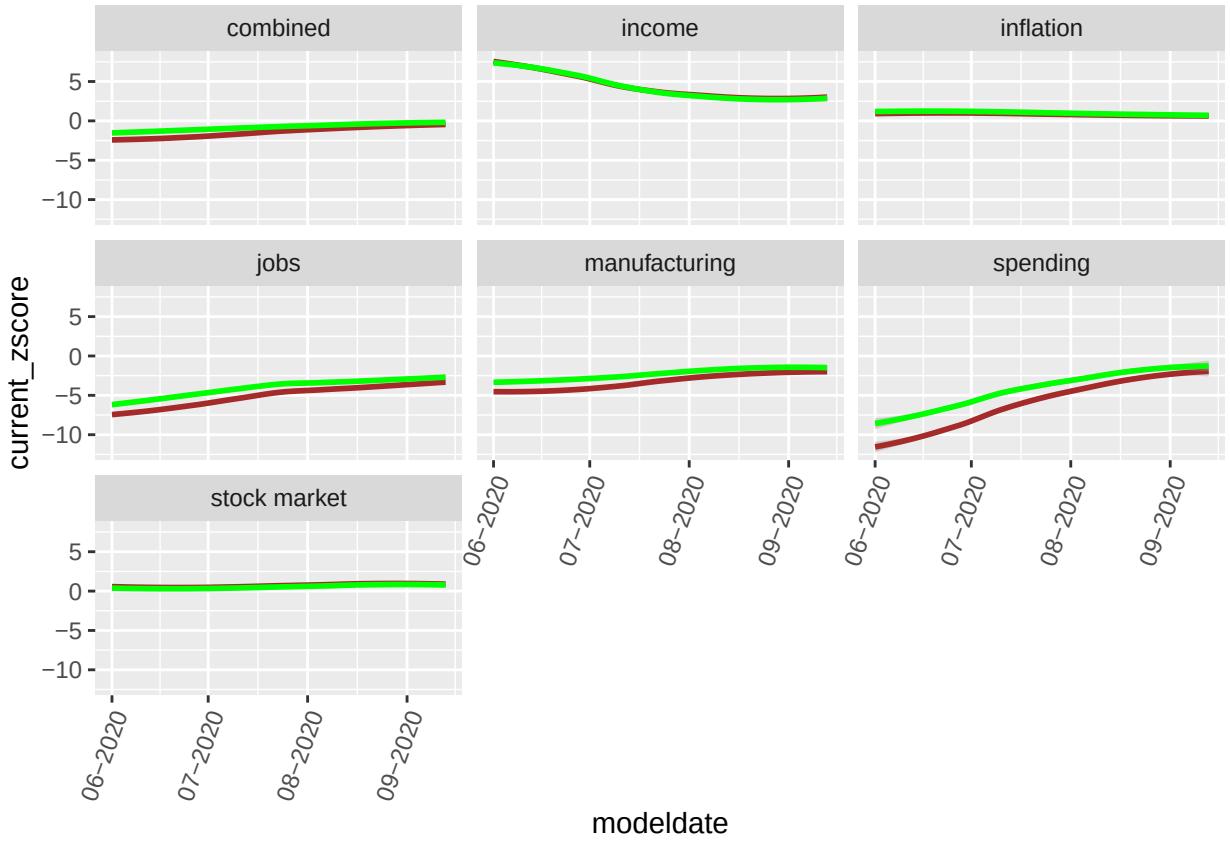
Economic Index

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:

```
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.

```
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: "2020-09-13" "2020-09-13" ...
##  $ 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  0.00175 0.00178 0.00287 0.0017 0.00103 ...
##  $ evprob_chal   : num  0.0 2.5e-05 0.0 2.5e-05 2.5e-05 0.0 2.5e-05 0.0 0.0 2.5e-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: "2020-09-13 10:33:03" "2020-09-13 10:33:03" ...
##  $ 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.

```
str(dfPSA)
```

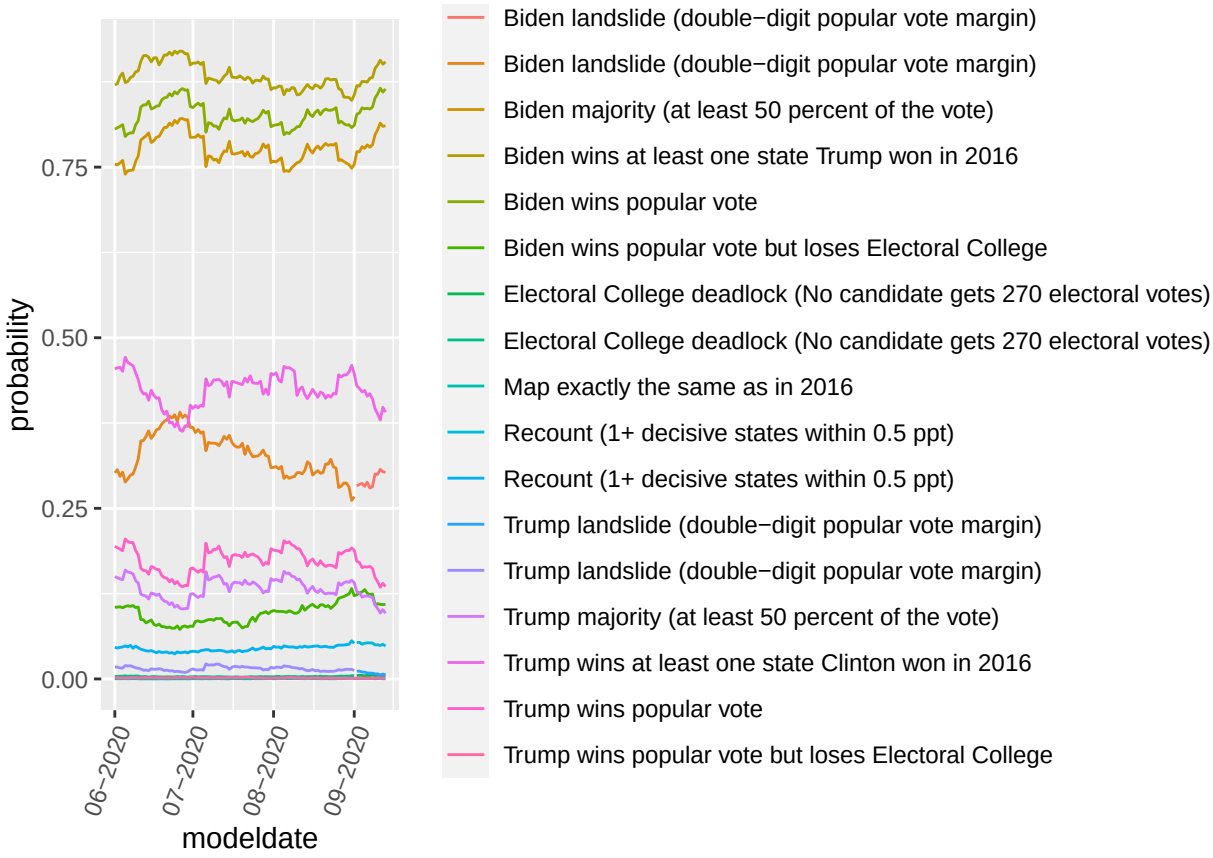
```
## 'data.frame':    1365 obs. of  12 variables:
##  $ cycle               : int  2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 ...
##  $ branch              : chr  "President" "President" "President" "President" ...
##  $ model               : chr  "polls-plus" "polls-plus" "polls-plus" "polls-plus" ...
##  $ modeldate           : Date, format: "2020-09-13" "2020-09-13" ...
##  $ 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  0.81073 0.09625 0.10944 0.00131 0.86455 ...
##  $ scenario_description: chr  "Biden majority (at least 50 percent of the vote)" "Trump majority (at
##  $ timestamp           : POSIXct, format: "2020-09-13 10:33:03" "2020-09-13 10:33:03" ...
##  $ simulations         : int  40000 40000 40000 40000 40000 40000 40000 40000 40000 40000 ...
```

```
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')
```




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

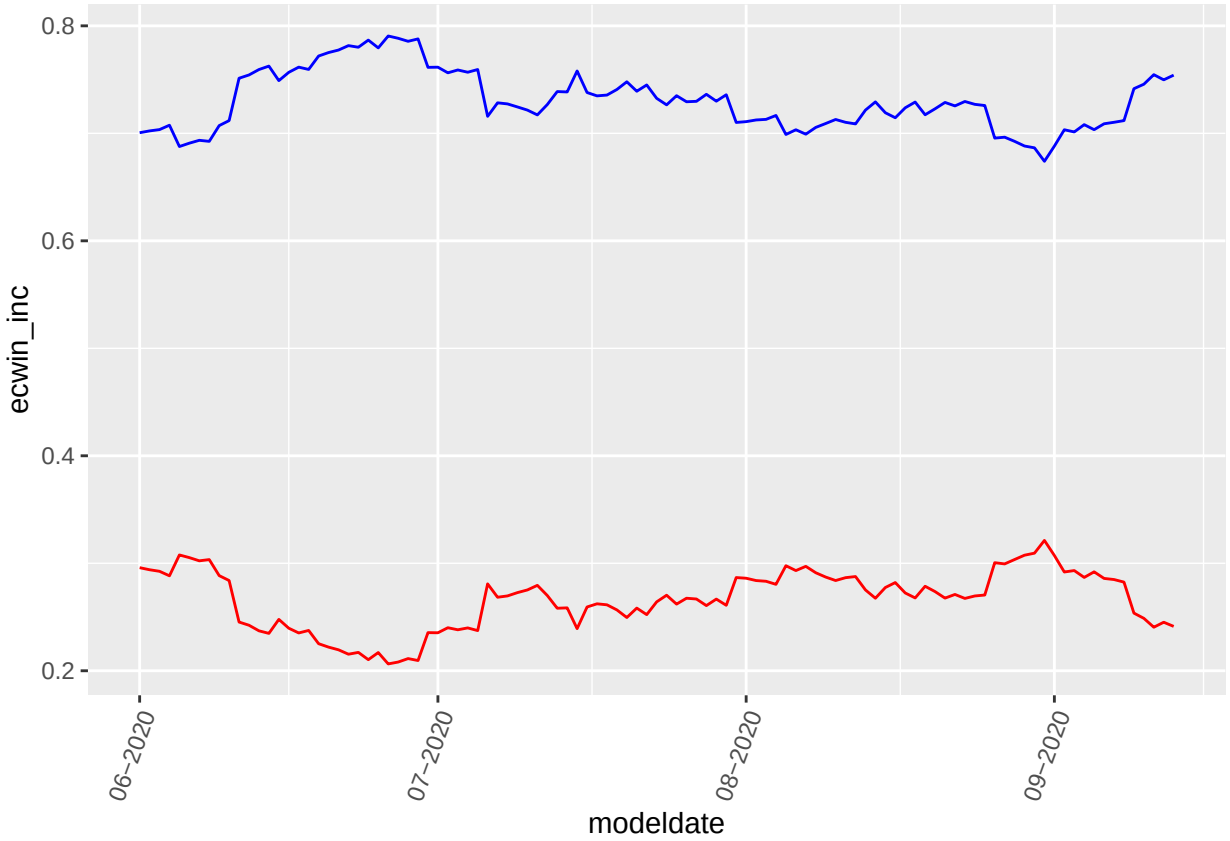
Presidential National Toplines

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')
```



State Toplines

```
res1 <- dbGetQuery(con, 'select modeldate, state, state_turnout, winstate_inc,
                  winstate_chal, tipping from Presidential_State_Toplines;')

res2 <- dbGetQuery(con, 'select state, avg(state_turnout) AVG_Turnout from Presidential_State_Toplines
                   group by state order by AVG_Turnout desc;')
res2
```

```
##                  state AVG_Turnout
## 1           California  14250000.0
## 2              Florida  10137500.0
## 3                Texas   9473148.5
## 4             New York   7698649.9
## 5         Pennsylvania   6268720.1
## 6                 Ohio   5665407.5
## 7             Illinois   5630481.5
## 8             Michigan   5033399.1
## 9       North Carolina   4997414.0
## 10             Georgia   4309311.9
## 11            Virginia   4217843.5
## 12          New Jersey   3914075.2
## 13          Washington   3569748.0
## 14       Massachusetts   3339868.9
## 15           Minnesota   3057890.0
## 16           Wisconsin   3044985.1
## 17            Colorado   2922639.5
```

```
## 18             Missouri   2839082.6
## 19              Indiana   2821712.2
## 20              Arizona   2811835.5
## 21             Maryland   2811448.6
## 22            Tennessee   2599010.1
## 23       South Carolina   2212056.4
## 24              Alabama   2141237.5
## 25               Oregon   2107866.0
## 26            Louisiana   2012434.9
## 27             Kentucky   1937932.2
## 28          Connecticut   1637886.5
## 29                 Iowa   1565090.0
## 30             Oklahoma   1458930.5
## 31                 Utah   1225680.0
## 32               Nevada   1221675.4
## 33          Mississippi   1216265.4
## 34               Kansas   1206029.1
## 35             Arkansas   1152891.9
## 36             Nebraska    869104.5
## 37           New Mexico    814759.9
## 38        New Hampshire    765577.6
## 39                Maine    757608.5
## 40                Idaho    736384.0
## 41        West Virginia    686594.2
## 42              Montana    519365.5
## 43         Rhode Island    473773.2
## 44             Delaware    458656.2
## 45               Hawaii    431654.2
## 46                 ME-1    395316.1
## 47         South Dakota    390269.0
## 48                 ME-2    357086.2
## 49         North Dakota    344012.9
## 50 District of Columbia    328520.1
## 51              Vermont    321932.4
## 52               Alaska    316091.9
## 53                 NE-2    300398.9
## 54                 NE-1    288211.0
## 55                 NE-3    278142.4
## 56              Wyoming    257585.0
```