Final Project
COM EM 747
Bruce Mallory
https://github.com/BruceMallory/747_FinalProject

Introduction

For this project I have:

A. Downloaded articles from the *NYTimes* that discuss the "For the People Act" (H.R.1).
B. Calculated metrics about each article, including:
    a. Article sentiment scores (using three sentiment dictionaries),
    b. Word frequencies within each article.
C. Selected target words (e.g., "Manchin," "filibuster") for analysis.
D. Written visualization functions to display:
    a. article sentiment over time,
    b. within article word-frequency over time.
E. Presented several interesting observations based on these visualizations.

A. Downloading articles

I used *Nexis/Uni* to download articles from the *NYTimes* that included the phrase "For the People Act" from November 1st, 2020, to September 1st, 2021. As I did this, I used the *Nexis/Uni* API to identify and remove duplicate articles that appeared in different versions of the paper (sometimes with different titles).

The articles were downloaded in .RTF format, and I converted them to .txt format. As I translated. between formats, I cleaned up the files by:

- removing explanatory verbiage (e.g., photo captions, author descriptions, live links),
- removing all articles that were daily summaries,
- removing duplicate articles that the API didn't flag,
- adding "Document: Type" tags where they were missing.

B. Calculating metrics

Because of *Nexis/Uni*'s limit on the number of articles in a download, I downloaded articles in two-month intervals. I then wrote several functions that I used to import my data, from the .txt files, into an R data frame.
(https://github.com/BruceMallory/747_FinalProject/blob/main/wrangling_functions.R)

As I imported the articles (n=113), I only ingested the title of the article, the date of the article, and the body of the article. I ignored the section titles, bylines, copyright, and other information about the article that was included in the downloaded .RTF file.

My functions created two data frames (`Article_sntmts` and `Article_wdfrq`).

For the word frequency data frame, I tokenized the body of each article by word, removed stop words, and then calculated the proportion each word was of the total words in the article.

```
Article_sntmts                113 obs. c
  $ Article                : int  1 2 3
  $ Type                   : chr  "Op-Ed
  $ Date                   : POSIXct, fo
  $ prop_matched_affin: num  0.1475
  $ prop_matched_bing : num  0.153
  $ prop_matched_nrc  : num  0.292
  $ affin_score            : num  -0.041
  $ bing_score             : num  -0.041
  $ nrc_score              : num  0.0295
  $ nrc_positive_score: num  0.165
  $ nrc_negative_score: num  0.1357
  $ nrc_fear_score         : num  0.0649
  $ nrc_anger_score        : num  0.0649
  $ nrc_joy_score          : num  0.0354
  $ title_score            : num  -0.375
```

```
Article_wdfrq              41636 obs.
  $ Article    : num  1 1 1 1 1 1 1 1
  $ Type       : chr  "Op-Ed" "Op-Ed"
  $ Date       : POSIXct, format: "20
  $ title      : chr  "Democracy's Ne
  $ word       : chr  "trump" "electi
  $ n          : int  10 7 6 5 5 4 4
  $ proportion: num  0.0225 0.0158 0
```

For the sentiment data frame, I calculated the proportion of words in the article that matched the given sentiment dictionary. Though I have no comparison, I was struck by how few of the words in an article matched any of the sentiment dictionaries. And I noticed that the nrc dictionary matched significantly more words than the other dictionaries.

```
> summary(Article_sntmts$prop_matched_affin)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
0.06145 0.09804 0.11048 0.11258 0.12811 0.19298 
> summary(Article_sntmts$prop_matched_bing)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
0.05063 0.11581 0.12971 0.13077 0.14758 0.20714 
> summary(Article_sntmts$prop_matched_nrc)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.1538  0.2316  0.2476  0.2487  0.2661  0.3613 
```

For the affin dictionary, which assigns words a score between -5 (negative sentiment) and +5 (positive sentiment) I calculated the mean sentiment of all the words that were matched in the article (`affin_score`).

For the bing dictionary, which assigns a binary positive/negative sentiment to each matched word, I assigned values (positive=+1, negative=-1) and calculated the mean sentiment of all the matched words (`bing_score`).

For the nrc dictionary, which also tags words as positive/negative, I also calculated a mean sentiment (`nrc_score`).

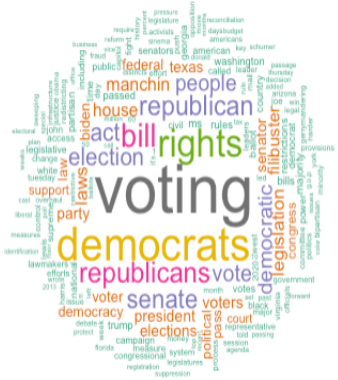
Additionally, the nrc dictionary tags words with specific emotions (e.g., fear, anger). And I calculated the mean number of words in each article that were tagged by fear, anger, or joy (`nrc_fear_score, nrc_anger_score, nrc_joy_score`).

As I was working with the coding of the sentiment scores, I noticed several words (e.g., "trump" and "congress") which were given positive sentiment values that did not make sense in the context of a newspaper article about politics. I removed the words I noticed from the sentiment lexicons, but I did not spend any time examining the validity of the sentiment lexicons in the context of political articles. Further, I did not attempt to validify my measure of an article's sentiment (the average sentiment score of matched words) with any other measure of that article's sentiment. I did not read any of the articles, and I have reservations about the validity and/or usefulness of any of the sentiment scores that I calculated.

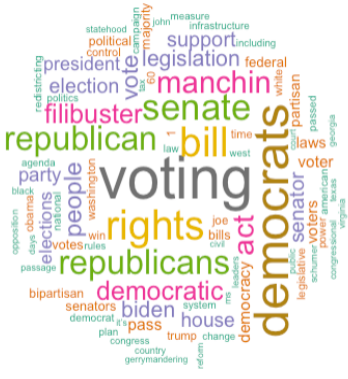
C. Focus words

There were approximately 9,000 unique words used in the 113 articles that I am attempting to analyze. Aside from focusing on words that made sense in the context of my investigation, I looked at words most frequently used. And at words used in the top quartile of articles measured by a particular sentiment score.

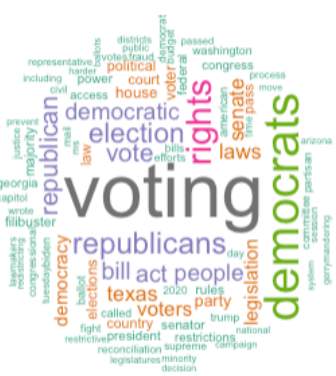
Initially I tried a word cloud visualization. Though interesting as "eye candy," I found the visualization difficult to use as I searched for words to focus my investigation around.



all words



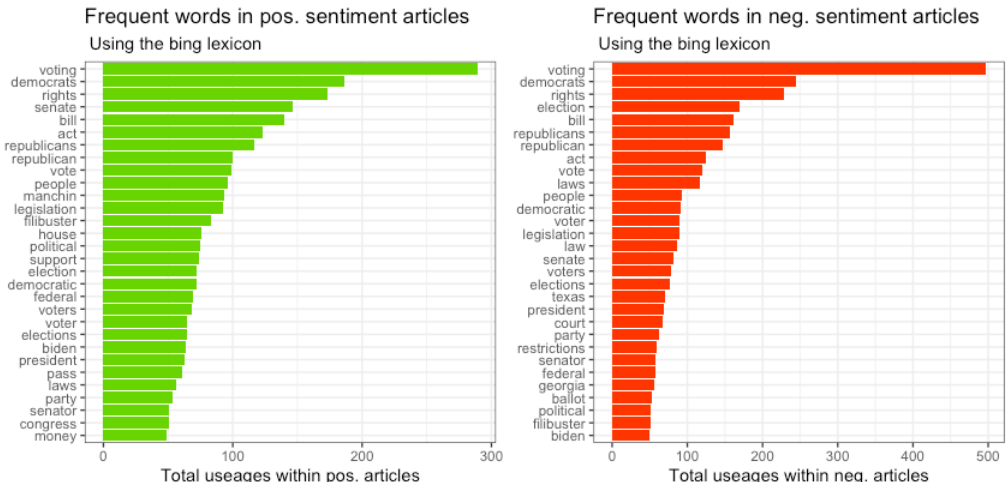
words from positive `affin_sent` articles



words from negative `affin_sent` articles

I then worked on writing a function to display bar graphs of the 30 most frequent words in articles in the top and bottom quartiles of articles based on a given sentiment score.
https://github.com/BruceMallory/747_FinalProject/blob/main/filtering_functions.R

This visualization was more useful, since it allowed me to scan the two lists and compare the frequency of use within positively and negatively sentiment scored articles.

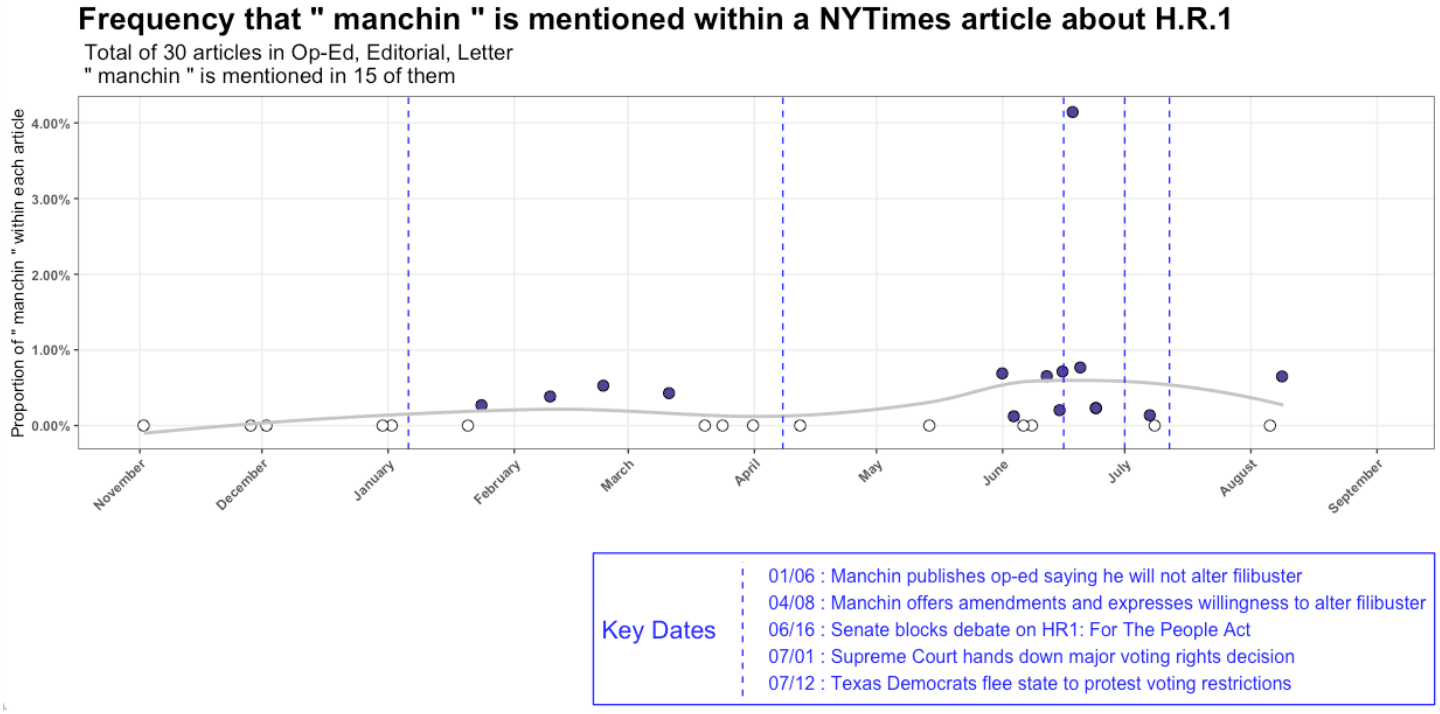


D. Visualization functions

I wrote two functions to display my data.
https://github.com/BruceMallory/747_FinalProject/blob/main/display_functions.R

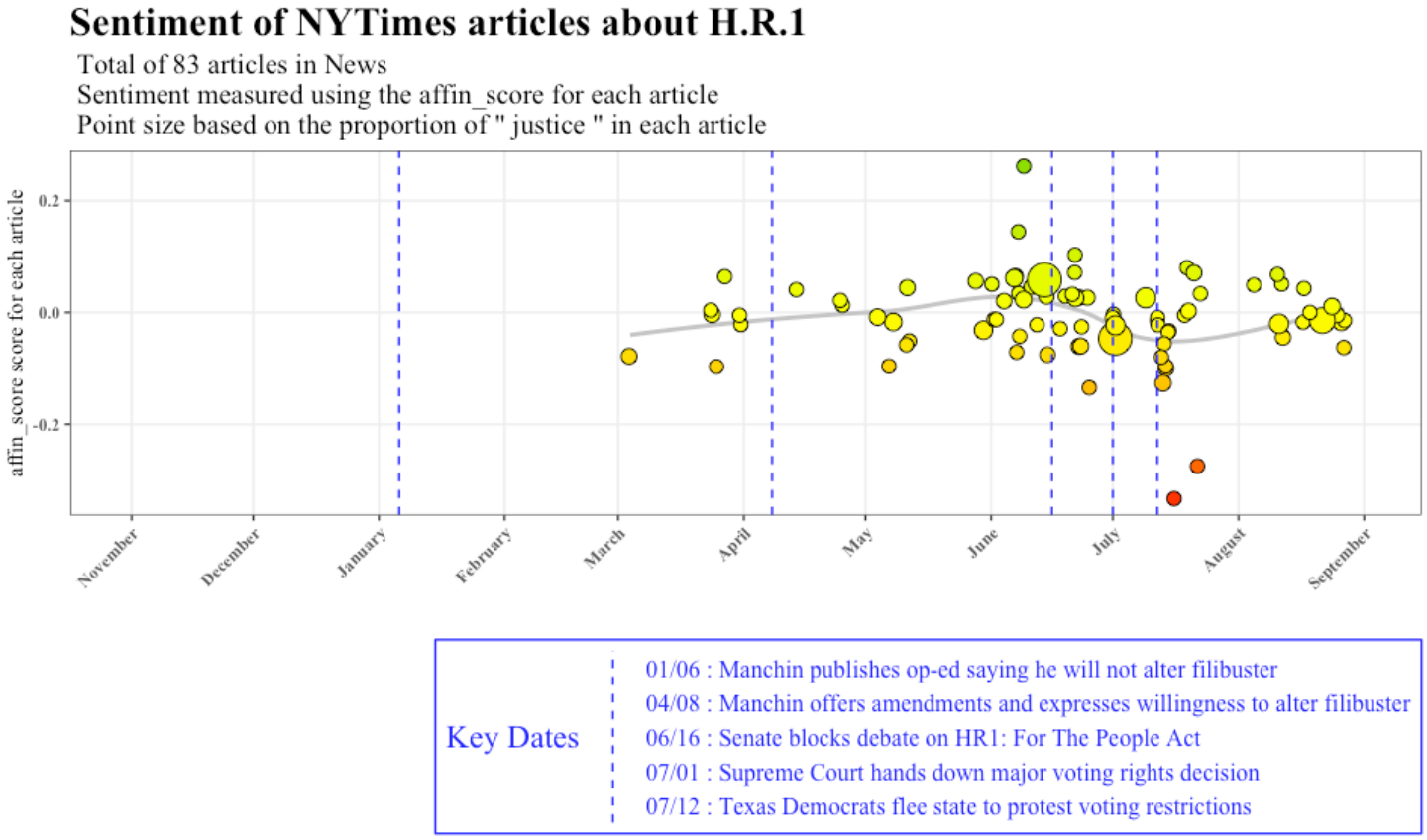
(1) A function that displays word frequency over time - with key events overlayed.
This function takes two inputs:
a) the `type` of article ("Op-Ed", "Editorial", "Letter", and/or "News"),
b) the `target_word` whose frequency within each article is being displayed.

Example: `wdfrq_over_time(c("Op-Ed", "Editorial", "Letter"),"manchin")`



(2) A function that displays article sentiments over time - with key events overlayed.
This function takes three inputs:
a) the `type` of article (either "Op-Ed", "Editorial", "Letter", and/or "News"),
b) the `sent_dict` to be used,
c) a `target_word` whose frequency within each article will be displayed.

Example: `sntmts_over_time(c("News"), "affin_score", "justice")`



E. Interesting observations

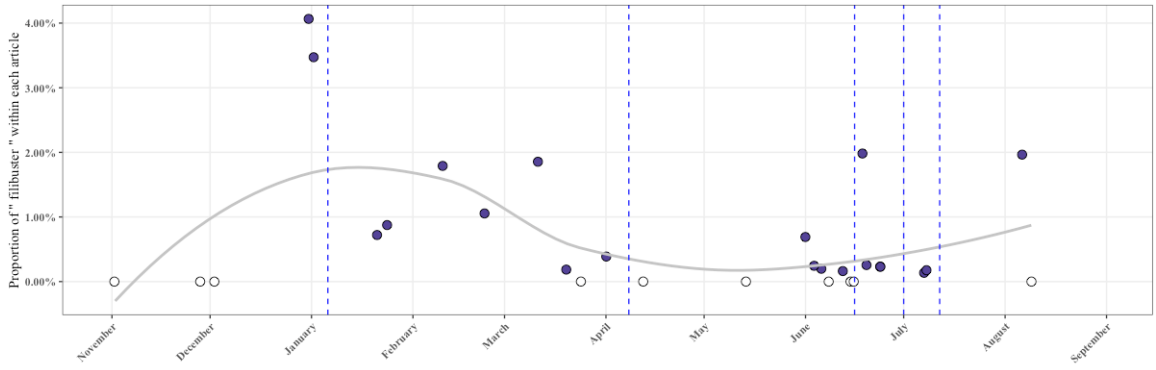
Because I was able to subset my articles by type, I was able to see that all the articles in the *NYTimes* that included the phrase "For the People Act" and were classified as "News" were written after March 1st. While the 30 opinion articles ("Op-ed," "Editorial," and "Letters") in my sample were distributed across the time frame that I was sampling.

In looking at the frequency of the word "filibuster" in these two subsets of articles (news vs. opinion), the peaks for frequency occurred in different locations. For the opinion articles, 'filibuster' frequency peaked just prior to Senator Manchin publicly stating that he would not alter the filibuster, then trended downward, followed by moving upward after the Republican filibuster of H.R.1. Which is a different pattern than in the news articles where the peak of 'filibuster' usage was just prior to the Republican's filibuster.

My observation is based on the three visualizations on the next page.

Frequency that " filibuster " is mentioned within a NYTimes article about H.R.1

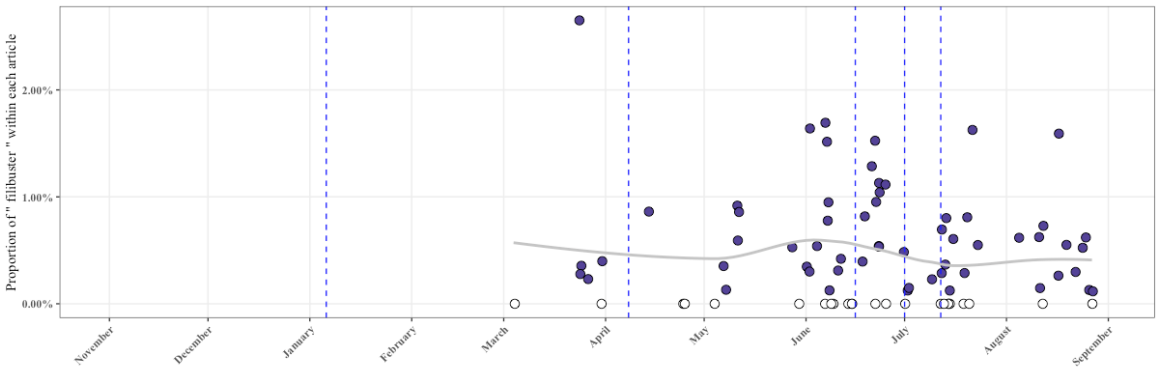
Total of 30 articles in Op-Ed, Editorial, Letter
" filibuster " is mentioned in 20 of them



Key Dates
- 01/06 : Manchin publishes op-ed saying he will not alter filibuster
- 04/08 : Manchin offers amendments and expresses willingness to alter filibuster
- 06/16 : Senate blocks debate on HR1: For The People Act
- 07/01 : Supreme Court hands down major voting rights decision
- 07/12 : Texas Democrats flee state to protest voting restrictions

Frequency that " filibuster " is mentioned within a NYTimes article about H.R.1

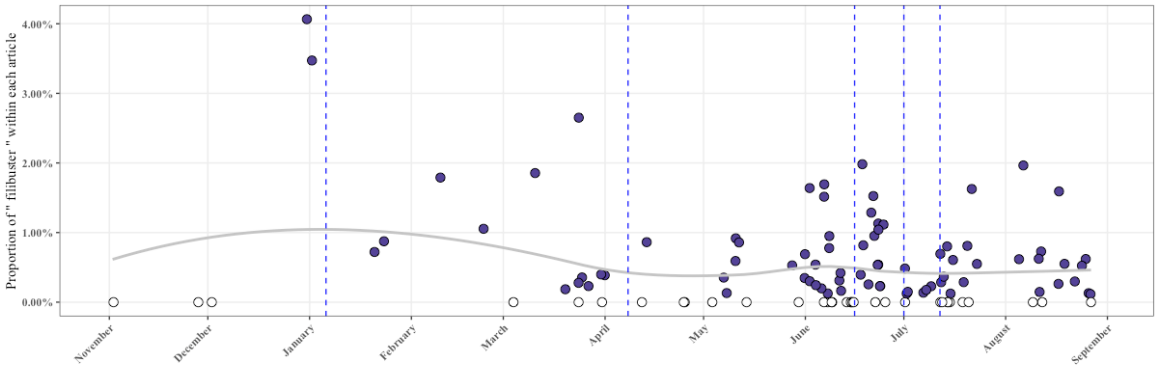
Total of 83 articles in News
" filibuster " is mentioned in 59 of them



Key Dates
- 01/06 : Manchin publishes op-ed saying he will not alter filibuster
- 04/08 : Manchin offers amendments and expresses willingness to alter filibuster
- 06/16 : Senate blocks debate on HR1: For The People Act
- 07/01 : Supreme Court hands down major voting rights decision
- 07/12 : Texas Democrats flee state to protest voting restrictions

Frequency that " filibuster " is mentioned within a NYTimes article about H.R.1

Total of 113 articles in Op-Ed, Editorial, Letter, News
" filibuster " is mentioned in 79 of them

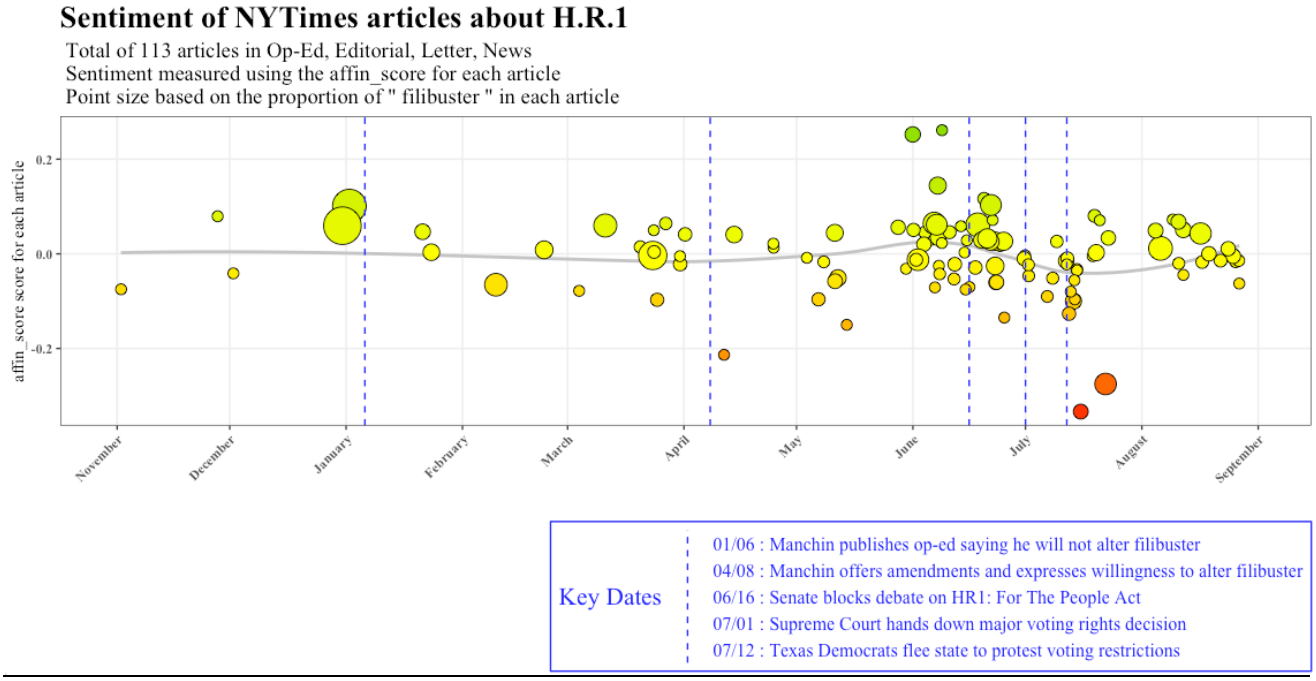


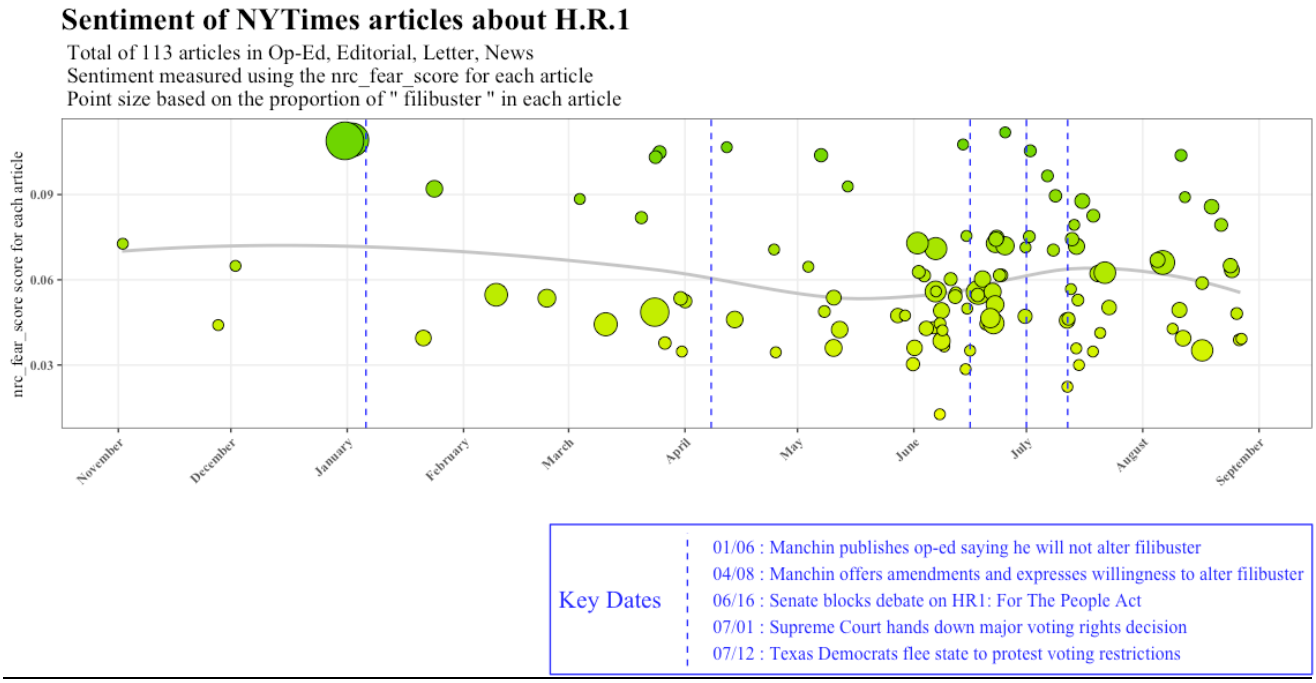
Key Dates
- 01/06 : Manchin publishes op-ed saying he will not alter filibuster
- 04/08 : Manchin offers amendments and expresses willingness to alter filibuster
- 06/16 : Senate blocks debate on HR1: For The People Act
- 07/01 : Supreme Court hands down major voting rights decision
- 07/12 : Texas Democrats flee state to protest voting restrictions

E. Interesting observations (cont.)

In looking at article sentiment over time (using the affin lexicon and my mean sentiment per word score) my visualizations show a positive peak just prior to the Republicans filibuster followed by a dip in sentiment. But this pattern does not repeat itself when I used the nrc lexicon and looked at the frequency of words in each article that had been tagged as "fear" words. In this visualization fear subsides prior to the Republican filibuster, then increases afterwards.

In my sentiment visualizations, I used the size of the dots to represent the frequency of a target word. Looking at the dot sizes, the 'filibuster' word frequency pattern described above is not as clear as in the word frequency visualization shown on the previous page. But these visualizations do clearly highlight the two articles that were written at the beginning of January that had a high frequency of 'filibuster' usage.





F. Conclusion

I believe that the visualizations that I was able to code, are constructed in a manner that presents a great deal of information in a clean and understandable format. But in wrangling the data for these visualizations and in building the visualizations, it has become quite clear to me that the validity of text mining and sentiment analysis in the context of articles about a political event is suspect.

I had the following observations:

(1) The bag-of-words approach to attaching a sentiment to an article ignores may of the aspects of an article that convey sentiment. Using bag-of-words, the sentence "Democrats happy with result" (which would have 1 positive word and a positive sentiment), would have a radically different sentiment score than the sentence "Democrats not sad with result" (which would have 2 negative words and a negative sentiment). Even though both sentences are saying the same thing.

(2) The title of an article often gives more information about the sentiment of an article, than the collection of words in the body of the article. For this reason, I tried the bag-of-words approach just on the titles – but that didn't give me noticeably different results.

(3) The `unnest_tokens()` function advertises a parameter that will tokenize text by sentence. And there is a package (`sentimentr`) which will calculate sentece level sentiments (paying attention to the effect of negation within a sentence). But in a brief attempt at sentence level tokenization and sentiment analysis, I was not successful.

(4) The sentiment lexicons were validified on a corpus that were not articles about politics, and as such there are clear contextual differences in how words are perceived (e.g., "trump").

(5) In tokenizing by words, `unnest_tokens()`drops capitalization, which meant that I was unable to distinguish between "Democratic" (as in the party), and "democratic" (the political system).

And, most significantly, in building and using my visualizations I was quite aware of the fishing expedition that my visualizations encouraged. By coding in the ability to investigate different target words, different sentiment lexicons, and different sub-sets of article types I gave myself a wealth of possible visualizations. With that wealth of possible visualizations, I moved from what-does-the-data-tell-me into what-patterns-can-I-find. And with the patterns being represented by a LOESS regression line, there is a strong possibility that a visual dip in the smoothed line is caused by a very small number of articles. As such, I believe that my visualizations are prone to showing insignificant patterns and thus encourage over interpretation.

All of this leads me to the conclusion that the text mining techniques that I've implemented for this project have not led me any closer to understanding how newspaper articles portrayed the progression of the "For the People's Act" over time.