Pre- and Post-Debate Democratic Primary Data: Twitter, Google Trends, and Polls

Fundamentals of Computing and Data Display, Fall 2019

Zoe Padgett and Fatou Thiam

2019-12-09

Contents

Introduction

Twitter and Google Trends are becoming increasingly popular tools for social science researchers. They represent an easily accessible source of large amounts of data, which has become advantageous as survey response rates decline and costs rise. Researchers interested in predicting election results have begun looking to Twitter data to replace or supplement traditional election polls, with mixed results (Gayo-Avello 2013).

Recently, there have been studies using sentiment analysis of Twitter data to predict election outcomes in India (Salunkhe and Deshmukh 2017), to predict state-level polling results in the U.S. (Beauchamp 2017), and to predict the winners of three presidential elections in Latin America (Gaurav et al. 2013). Beauchamp (2017) found that Twitter data may be useful in making state-level campaign strategy decisions. Additionally, Kassraie, Modirshanechi, and Aghajan (2017) used Google Trends and Twitter data to predict the 2016 U.S. election outcomes with only 1% error.

We are interested in whether Twitter data and Google trends data could be used to supplement polling results, by providing real-time information to candidates while they wait for polling data to come in. For example, candidates may be interested in understanding how public opinion has shifted immediately after a debate in order to run a more agile campaign. This project is an exploratory analysis of Twitter and Google Trends data to see if pre- and post-debate polling data during the 2020 U.S. Democratic primary election aligns with these real-time sources.

Data

Using the rtweet package and the search_tweet function, we were able to gather sufficient tweets for our project and classified it as pre or post debate. The initial dataset contains a little over 1.3 million tweets which was rich enough to proceed with analysis. After removing irrelevant tweets such as tweets containing "Ukraine", we were left with 1.2 million tweets which we then divided in two groups :pre and post debate. The pre-debate dataset has 226,799 tweets and 998,780 tweets for the post-debate dataset. As expected, there was a lot more tweets after the debate than before. With Twitter having a location column, we wanted to focus our anylysis on certain regions/ states in the U.S., however the location column is an open text and not regulated. We then picked a few states to focus our analysis on.

Twitter

Below is our token to access the Twitter API and start collecting tweets. Note that these codes are fake, for security purposes.

```
create_token(
  app = "fcdd-course",
```

```
  consumer_key = "XXXXXXXXXXXXXXXXXX",
  consumer_secret = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
  access_token = "XXXXXXXXXXXXXXXXXX-XXXXXXXXXXXXXXXXXX",
  access_secret = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
)
```

Quering Tweets

Using the search_tweets function in the rtweets package, we will be quering tweets with the keywords specified below. We limited our tweets pool to November 20th and November 21st. The reasonning behind this is to get a feel post and pre debate.

```
dem <- search_tweets("#democrats OR #candidates OR #election2020 OR #BIDEN
                     OR #sanders OR #warren OR #harris
                     OR #buttigieg OR #steyer OR #yang OR #booker
                     OR #klobuchar OR #gabbard
                     OR @KamalaHarris OR @JoeBiden OR @BernieSanders
                     OR @ewarren OR @PeteButtigieg
                     OR @TomSteyer OR @AndrewYang OR @BookerCory
                     OR @amyklobuchar OR @TulsiGabbard ",  since='2019-11-20',
                     until='2019-11-21', n= 500000,
                     retryonratelimit = TRUE, verbose = TRUE)
```

Data Cleaning

Even after we specify the keywords, we know that we would still get irrelevants tweets mainly with the Ukraine issue at the time of pulling. We removed all tweets that had the word "Ukraine" in it.

```
myvars <- c("text", "location", "created_at")
df1 <- dem[myvars]
df2 <- dem2[myvars]
```

```
df1 <- df1 %>%
      mutate( ukraine = (str_detect(df1$text,
                                    regex("Ukraine",
                                          ignore_case = TRUE)))) %>%
      filter(ukraine=="FALSE") %>%
      select(-ukraine)

df2 <- df2 %>%
      mutate( ukraine = (str_detect(df2$text,
                                    regex("Ukraine",
```

```
                                                ignore_case = TRUE)))) %>%
      filter(ukraine=="FALSE") %>%
      select(-ukraine)
```

```
tweets <- rbind(df1, df2)
```

Next, we seperate the tweets by candidates in order to do the analysis at the candidate level. It's important to note that one tweets can be addressed to multiple candidates. In that case, the tweet will be found in the each of those candidate dataset. The code below exemplifies this step of the cleaning process.

```
tweets$BS <-(str_detect(tweets$text,
                        regex("#Sanders|@BernieSanders",
                              ignore_case = TRUE)))
tweets$KH <-(str_detect(tweets$text,
                        regex("#harris |@KamalaHarris",
                              ignore_case = TRUE)))
tweets$JB <-(str_detect(tweets$text,
                        regex("#biden |@JoeBiden",
                              ignore_case = TRUE)))
tweets$EW <-(str_detect(tweets$text,
                        regex("#warren |@ewarren",
                              ignore_case = TRUE)))
tweets$PB <-(str_detect(tweets$text,
                        regex("#buttigieg|@PeteButtigieg",
                              ignore_case = TRUE)))
tweets$TS <-(str_detect(tweets$text,
                        regex("#steyer | @TomSteyer",
                              ignore_case = TRUE)))
tweets$AY <-(str_detect(tweets$text,
                        regex("#yang| @AndrewYang",
                              ignore_case = TRUE)))
tweets$BC <-(str_detect(tweets$text,
                        regex("#booker | @BookerCory",
                              ignore_case = TRUE)))
tweets$AK <-(str_detect(tweets$text,
                        regex("#klobuchar | @amyklobuchar",
                              ignore_case = TRUE)))
tweets$TG <-(str_detect(tweets$text,
                        regex("#gabbard |@TulsiGabbard",
                              ignore_case = TRUE)))
```

Here we reformat the created_at column as a date & time variable in order to seperate the tweets in two groups : post and pre debate.

```
tweets <- tweets %>%
  mutate( date= as.POSIXct(created_at, tryFormats = c("%Y-%m-%d %H:%M:%OS")))

df1 <- df1 %>%
    mutate( date= as.POSIXct(created_at, tryFormats = c("%Y-%m-%d %H:%M:%OS")))

df2 <- df2 %>%
    mutate( date= as.POSIXct(created_at, tryFormats = c("%Y-%m-%d %H:%M:%OS")))
```

All the tweets received before 9 p.m. on November 20th are accounted for in the pre-debate dataset and all tweets from 11 p.m. on November 20th to the next day are in the post-debate dataset. Note that tweets during the debate(9 - 11 p.m are ignored)

```
pre_debate <- tweets %>%
              filter(date(date) == "2019-11-20" & hour(date) < 21 )

post_debate <- tweets %>%
              filter(date(date) == "2019-11-20" & hour(date) >= 23 |
                       date(date) == "2019-11-21"  )


pre_debate <- pre_debate %>% select(-date, -created_at)
post_debate <- post_debate %>% select(-date, -created_at)
```

Sentiment Analysis

Sentiment analysis of the tweets will performed for only 5 candidates.

First we need to prep the tweets by removing all non-words such as emojis and use the sentiment analysis package for analysis. Althought the sentiment analysis packae uses 5 dictionnairies, we will only look at the results from the GI dictionnary. For the pre-debate tweets sentimeent we used all the tweets from the dataset; however for the post-debate analysis, we sampled from the dataset due to volume and processing error. The code below exemplifies this process.

```
bs <- pre_debate %>%
      filter (BS == "TRUE")

usableText=str_replace_all(bs$text,"[^[:graph:]]", " ")

usableText <- tolower(bs$text)

usableText<- iconv(usableText, "UTF-8","ASCII",  sub="byte")

sentiments_bs = analyzeSentiment(as.character(usableText))
```

```
pre_sent_pos <- data.frame("Candidate" = c("Biden",
                                           "Sanders",
                                           "Warren",
                                           "Harris",
                                           "Buttigieg"),
                           "Positive" =c((sum(sentiments_jb$PositivityGI))/27477,
                                         (sum(sentiments_bs$PositivityGI))/31603,
                                        (sum(sentiments_kh$PositivityGI))/16362,
                                         (sum(sentiments_pb$PositivityGI))/27242))
```

Google Trends

This section describes gathering the Google Trends data using the package gtrendsR (Massicotte and Eddelbuettel 2019). First, we register our Google API key. Then, we pull data for each candidate from the 2 days preceding and two days following the debate (November 18 through 22). We have to pull the data in two separate blocks, because gtrendsR only allows us to use five search terms at a time. We limit the location of searches to the US. Please note that the key here is not a real API key, for security purposes.

```
register_google(key = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX")
res1 <- gtrends(c("Joe Biden", "Bernie Sanders",
                  "Elizabeth Warren", "Kamala Harris",
                  "Pete Buttigieg"), geo = "US",
                time = "2019-11-18 2019-11-22", low_search_volume = T)

res2 <- gtrends(c("Tom Steyer", "Andrew Yang", "Cory Booker",
                  "Amy Klobuchar", "Tulsi Gabbard"), geo = "US",
                time = "2019-11-18 2019-11-22", low_search_volume = T)
```

Geocoding

Next, we compile and clean the Google Trends location data and prepare it for geocoding.

```
interest_by_location1 <- as_tibble(res1$interest_by_dma)
interest_by_location2 <- as_tibble(res2$interest_by_dma)
interest_by_location <- rbind(interest_by_location1, interest_by_location2)

locations_df <- as.data.frame(interest_by_location)
locations_df$location <- as.character(locations_df$location)
```

Then, we geocode the Google trends data.

```
gc_locations <- as_tibble(mutate_geocode(locations_df, location))
```

Data Cleaning

Next, we categorize the Google Trends data into pre-debate and post-debate data, based on the date.

```
interest_over_time1 <- as_tibble(res1$interest_over_time)
interest_over_time2 <- as_tibble(res2$interest_over_time)
interest_over_time <- rbind(interest_over_time1, interest_over_time2)

interest_over_time_pre <-
  interest_over_time %>%
  filter(date < "2019-11-19") %>%
  mutate(Pre_post="Pre-debate")

interest_over_time_post <-
  interest_over_time %>%
  filter(date > "2019-11-20") %>%
  mutate(Pre_post="Post-debate")

interest_over_time_all <- rbind(interest_over_time_pre, interest_over_time_post)
```

Polls

Polling data was collected from RealClearPolitics.com, which aggregates weekly polls. We created a dataset using the RealClearPolitics average before and after the November 20 debate. Because the website changes often, and we only needed a small snapshot of the data, it was more efficient to clean the data in Excel and import to R than to use web scraping.

```
github_link <- "https://github.com/znpadgett/surv727_padgett_thiam/raw/master/Data/Proje
temp_file <- tempfile(fileext = ".xlsx")
req <- GET(github_link,
           write_disk(path = temp_file))
polling_data <- readxl::read_excel(temp_file)
```

Results

Data exploration

Because our project was exploratory in nature, we spent significant time exploring each data source.

Twitter

Create a function to tabulate the count and proportion of tweets by candidates.

Create functions

```
type_var  <- unlist(map(pre_debate, class))

freq_tab <- function(x) {
# make table with count and frequency
tab <- cbind(Count = table(x, useNA = "ifany"),
Prop = round(prop.table(table(x, useNA = "ifany")),
2))
# get the categories as variable and rearrenge
tab <- as.data.frame(tab) %>%
tbl_df() %>%
mutate(Cat = row.names(tab)) %>%
select(Cat, Count, Prop)
}
```

```
props1 <- map(pre_debate[, type_var == "logical"], freq_tab)
props2 <- map(post_debate[, type_var == "logical"], freq_tab)


vars <- unlist(map(props1, nrow))

props_tab1 <- reduce(props1, rbind)
props_tab2 <- reduce(props2, rbind)
```

```
props_tab1 <- props_tab1 %>%
mutate(Variable = rep(names(vars), vars),
       Candidate = ifelse(Variable == "BS", "Sanders",
                   ifelse(Variable == "KH", "Harris",
                   ifelse(Variable == "JB", "Biden",
                   ifelse(Variable == "EW", "Warren",
                   ifelse(Variable == "PB", "Buttigieg",
                   ifelse(Variable == "TS", "Steyer",
                   ifelse(Variable == "AY", "Yang",
                   ifelse(Variable == "BC", "Booker",
                   ifelse(Variable == "AK", "Klobuchar",
                   ifelse(Variable == "TG", "Gabbard", NA)))))))))))

props_tab2 <- props_tab2 %>%
mutate(Variable = rep(names(vars), vars),
```

```
        Candidate = ifelse(Variable == "BS", "Sanders",
                    ifelse(Variable == "KH", "Harris",
                    ifelse(Variable == "JB", "Biden",
                    ifelse(Variable == "EW", "Warren",
                    ifelse(Variable == "PB", "Buttigieg",
                    ifelse(Variable == "TS", "Steyer",
                    ifelse(Variable == "AY", "Yang",
                    ifelse(Variable == "BC", "Booker",
                    ifelse(Variable == "AK", "Klobuchar",
                    ifelse(Variable == "TG", "Gabbard", NA)))))))))))
```

A visual of the proportion of tweets pre and post debate

Graphing

Pre-debate number of tweets

```
props_tab1 %>%
  filter(Cat == "TRUE") %>%
  ggplot() +
  geom_col(mapping = aes(x = reorder(Candidate, -Count), y=Count, fill = Candidate)) +
    labs(x = "Democatric Party Canditates",
       y = "Number of Tweets",
       caption = "Source: Twiiter")
```

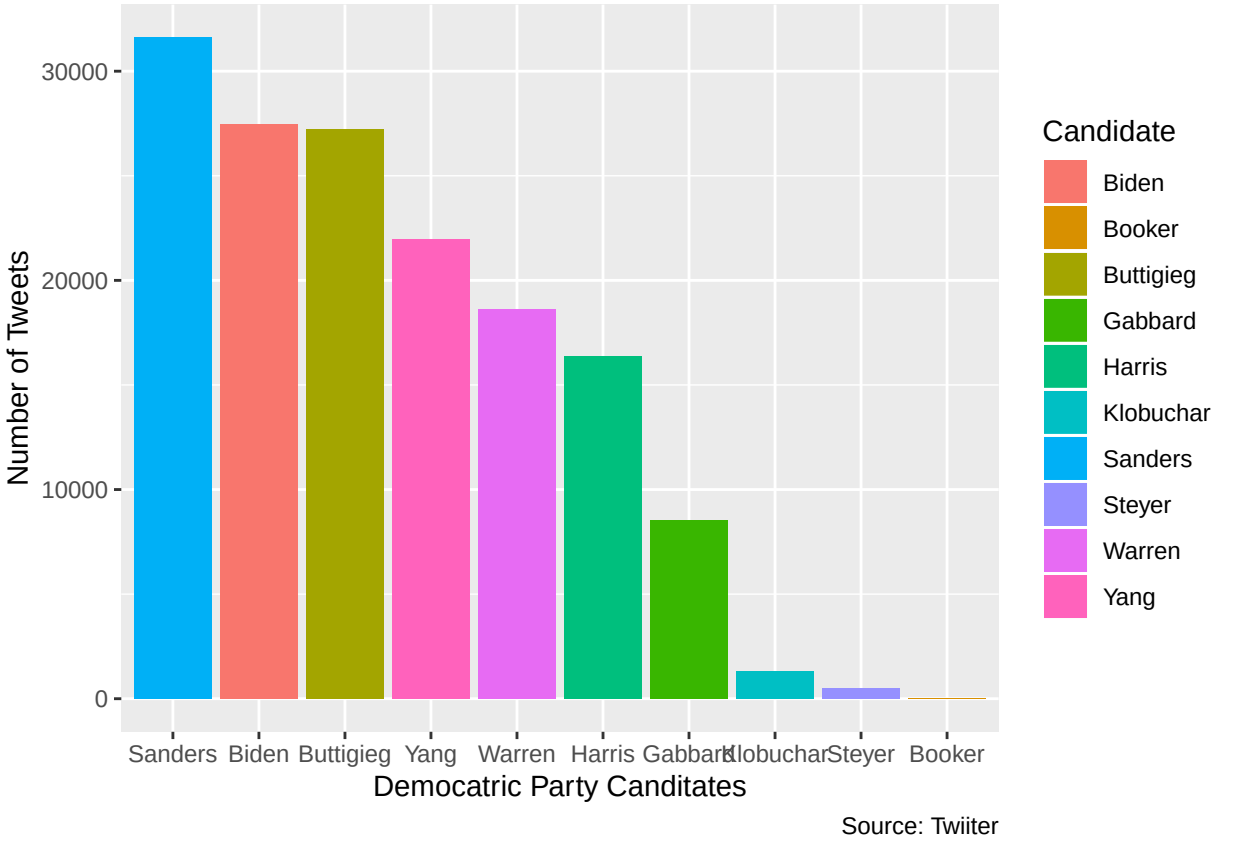

Figure 1: Pre-debate Mentions in Tweets, by Candidate

Post-debate number of tweets

```
props_tab2 %>%
  filter(Cat == "TRUE") %>%
  ggplot() +
  geom_col(mapping = aes(x = reorder(Variable, -Count), y=Count, fill = Variable)) +
    labs(x = "Democatric Party Canditates",
       y = "Number of Tweets",
       caption = "Source: Twiiter")
```

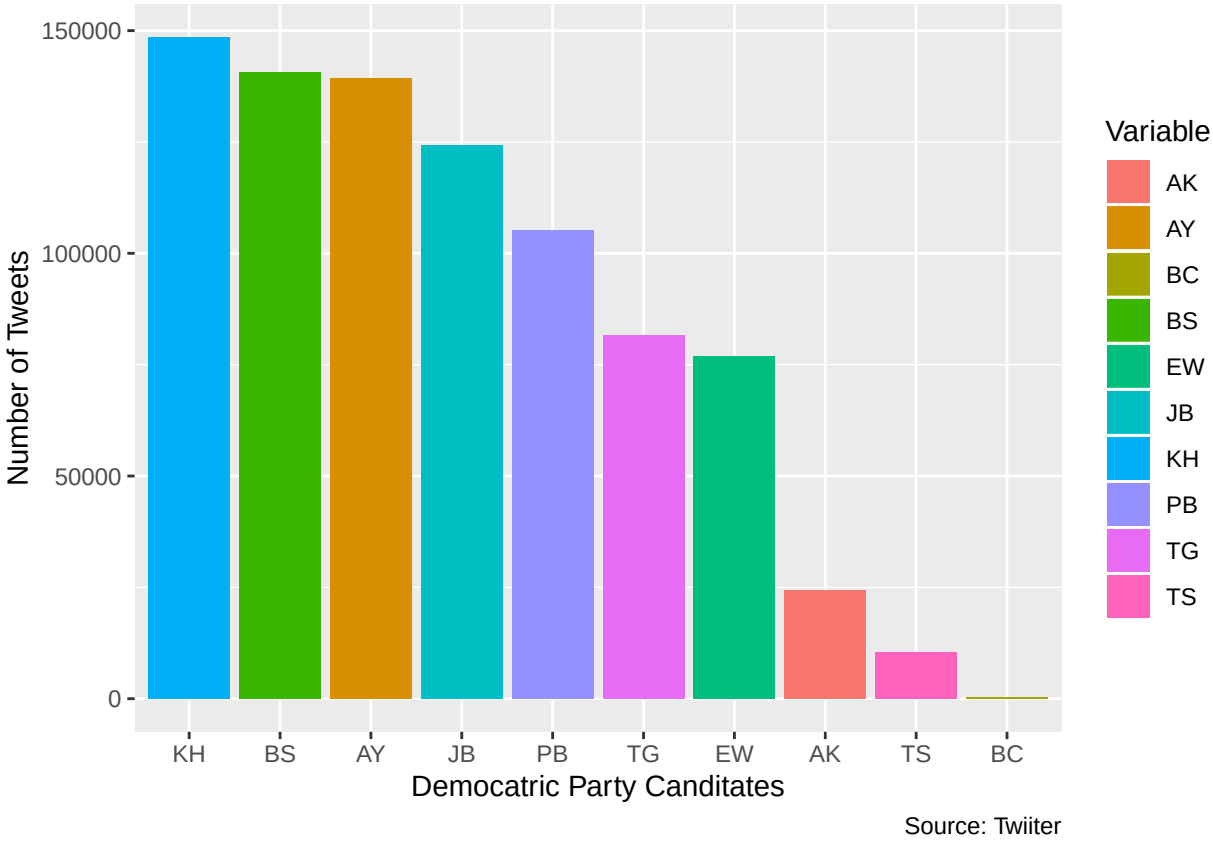

Figure 2: Post-debate Mentions in Tweets, by Candidate

Google Trends

To explore the Google Trends data, we examined the location and density of searches for each candidate. We created a map showing our results (Kahle and Wickham 2013). Because it was difficult to see the results for each candidate overlaid on one map, we added a facet wrap to show each individual candidate.

```
#get map
us <- c(left = -125, bottom = 25.75, right = -67, top = 49)
us_map <- get_stamenmap(us, zoom = 5, maptype = "toner-lite")
ggmap(us_map)
```

```
#clean data for mapping
trends_map <-
  gc_locations %>%
  group_by(keyword, location) %>%
  mutate("Hits"=sum(hits), Candidate=keyword)

#generate map
ggmap(us_map) +
  geom_point(data = trends_map, aes(x = lon, y = lat, size=Hits, color=Candidate),
             alpha = 0.2) +
  theme(axis.line=element_blank(),axis.text.x=element_blank(),
            axis.text.y=element_blank(),axis.ticks=element_blank(),
            axis.title.x=element_blank(),
            axis.title.y=element_blank()) +
  facet_wrap("Candidate", shrink=FALSE)
```

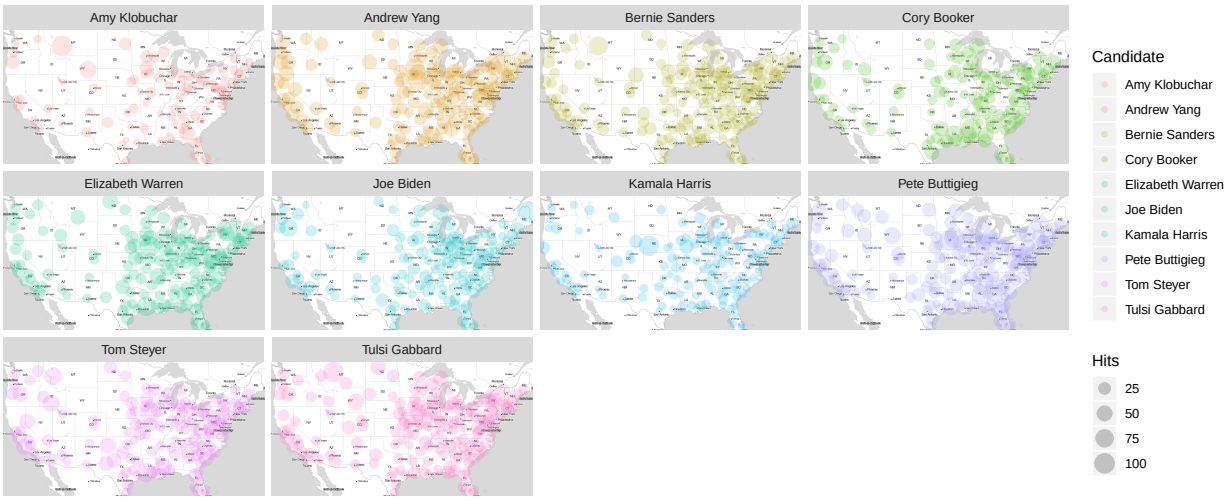

Figure 3: Map of Google Trends Hits, by Candidate

These maps show that candidates have different distributions and density of searches. Searches are more common near the coasts, and less common in the middle of the country.

We also explored the number of hits pre- and post- debate for each candidate.

```
interest_over_time_all$Pre_post <- factor(interest_over_time_all$Pre_post,
                                          levels = c("Pre-debate", "Post-debate"))

interest_over_time_all %>%
  group_by(keyword, Pre_post) %>%
  summarise(avg_hits=mean(hits)) %>%
  ggplot() +
  geom_col(mapping = aes(x=reorder(keyword, -avg_hits),
                         y=avg_hits, fill=Pre_post), color="black",
```

```
          position="dodge") +
xlab("Candidate") + ylab("Hits") +
scale_fill_discrete(name = "Timeframe") +
theme(axis.text.x = element_text(angle=35))
```

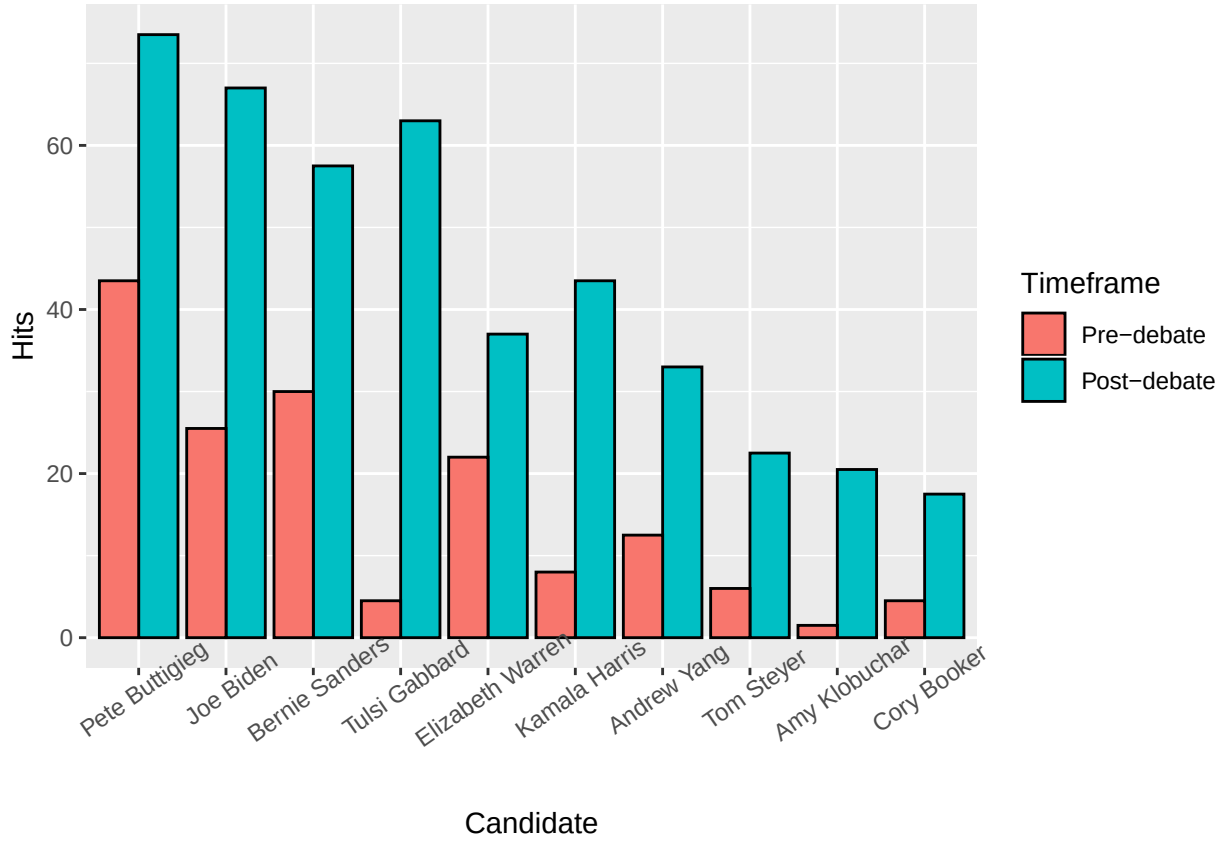

Figure 4: Pre- and Post-debate Google Trends Data, by Candidate

All candidates had higher search counts after the debate than before. This makes sense, because after a debate there is an increase in the number of news stories about candidates, which may lead to increased interest and thus, increased search activity. Additionally, users may be searching Google to learn about a candidate's debate performance if they did not watch the debate live.

Polling Data

For the polling data, we looked at the pre- and post-debate polling numbers for each candidate.

```
polling_data$Pre_post <- factor(polling_data$Pre_post, levels = c("Pre-debate", "Post-de

polling_data %>%
  group_by(Candidate, Pre_post) %>%
  filter(Candidate!="Bennet", Candidate!="Bloomberg") %>%
  ggplot() +
    geom_col(mapping = aes(x=reorder(Candidate, -Percentage), y=Percentage,
                           fill=Pre_post), color="black",   position="dodge") +
  xlab("Candidate") + ylab("Percentage") +
  scale_fill_discrete(name = "Timeframe") +
  theme(axis.text.x = element_text(angle=35))
```

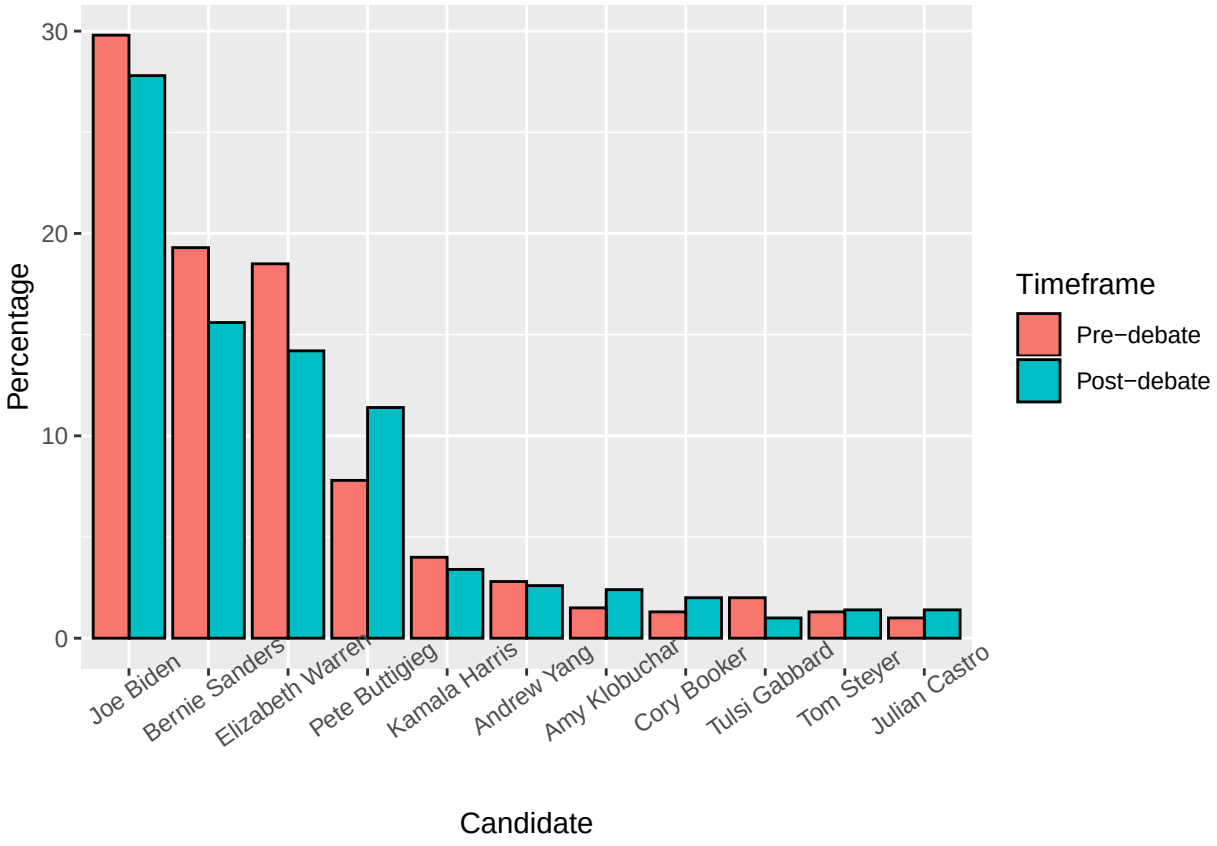

Figure 5: Pre- and Post-debate Polling Data, by Candidate

Analysis

We created a Shiny app that allows us to compare the polling, Google Trends, and Twitter sentiment data (Chang et al. 2019).

First, we cleaned up and combined the three data sources for use in the Shiny app.

```
#clean data
gtrends <-
  interest_over_time_all %>%
  mutate(Candidate=keyword, Data="GTrends") %>%
  group_by(Candidate, Pre_post, Data) %>%
  summarise(Percentage=mean(hits))

poll <-
  polling_data %>%
  select(Candidate, Pre_post, Percentage) %>%
  mutate(Data="Polling") %>%
  group_by(Candidate, Pre_post, Data)

twitter_data <- data.frame("Candidate" = c("Joe Biden", "Bernie Sanders",
                                           "Elizabeth Warren", "Kamala Harris",
                                           "Pete Buttigieg", "Joe Biden",
                                           "Bernie Sanders", "Elizabeth Warren",
                                           "Kamala Harris", "Pete Buttigieg"),
                           "Pre_post" =c("Pre-debate","Pre-debate",
                                         "Pre-debate","Pre-debate","Pre-debate",
                    "Data" = c("Twitter", "Twitter","Twitter","Twitter","Twitter",
                               "Twitter","Twitter","Twitter","Twitter","Twitter"),
                  "Percentage"=c(pre_sent_pos$Positive[1],pre_sent_pos$Positive[2],
                               pre_sent_pos$Positive[3],pre_sent_pos$Positive[4],
                               pre_sent_pos$Positive[5],post_sent_pos$Positive[1],
                               post_sent_pos$Positive[2],post_sent_pos$Positive[3],
                               post_sent_pos$Positive[4],post_sent_pos$Positive[5]))

twitter <- as_tibble(twitter_data)
twitter <-
  twitter %>%
  group_by(Candidate, Pre_post, Data) %>%
  mutate(Percentage=Percentage*100)

all_data <- rbind(poll, gtrends, twitter)

all_data$Pre_post <- factor(all_data$Pre_post, levels = c("Pre-debate",
                                                          "Post-debate"))
```

Then, we created a shiny app, which can be viewed by running the following code:

```
# Define UI
ui <- fluidPage(
```

```
  # Application title
  titlePanel("2020 Democratic Primary Candidate Data"),

  # Sidebar with a dropdown
  sidebarLayout(
    sidebarPanel(
      selectInput(inputId = "Candidate",
                  label = "Candidate",
                  choices = c("Joe Biden", "Pete Buttigieg", "Kamala Harris",
                              "Bernie Sanders", "Elizabeth Warren"),
                  selected = "Joe Biden"),
      selectInput(inputId = "Data",
                  label = "Data Type",
                  choices = c("Polling", "GTrends", "Twitter"),
                  selected = "Polling")
    ),

    # Show plot
    mainPanel(
      plotOutput(outputId = "graph")
    )
  )
)

# Define server logic
server <- function(input, output) {

  output$graph <- renderPlot({
    all_data %>%
      filter(Candidate == input$Candidate, Data == input$Data) %>%
      ggplot() +
      geom_col(mapping = aes(x=Pre_post, y=Percentage, fill=input$Candidate)) +
      ylim(0,100) +
      xlab("Timeframe") + ylab("Percentage") +
      theme(legend.position = "none")
  })
}

# Run the application
shinyApp(ui = ui, server = server)
```

When looking at the pre- and post-debate data from each source side-by-side, we can see that, while candidates who are polling higher have higher numbers of tweets and Google searches (for the most part), there is no clear relationship between the post-debate changes in polling

numbers and the post-debate changes in Twitter and Google Trends data. Regardless of polling numbers, all candidates had higher Twitter positivity ratings and higher numbers of Google searches post-debate than pre-debate.

Discussion

Our results show that neither Twitter nor Google Trends data align with pre- and post-debate changes in polling numbers There were no discernable patterns that relate the found data to the polling data. There may be many reasons why the found data did not align with the polling data. For one, the found data are not representative of U.S. voters, only Twitter users or Google searches, and the lack of availability of demographic data does not allow for weighting adjustments. Variable-to-variable comparison between two data sources may not be realistic. For example, a large number of searches and higher positivity of tweets may not actually mean that more people will vote for a candidate.

Additionally, there are many practical difficulties in using the found data, for example that the location field in the Twitter data corresponds to self-reported location, an open-ended field. Many users give incorrect or fake locations.

Overall, we don't think that this data should be used in place of survey data when it comes to election polling. However, this data could still be useful for candidates in understanding how social media users percieve them before and after a debate, or in understanding their search popularity. It is important to recognize that these measures are not an accurate proxy for voter opinion, but they could still be useful in their own right. Campaigns would also need to recognize that the target population of any inferences may on these data is not U.S. voters, or even all those living in the U.S., but is actually the population of tweets (for the Twitter data) or Google searches (for the Google Trends data). When these considerations are taken into account, we still think that this data could be useful for campaign strategy.

References

Kearney, M. W. (2018). rtweet: Collecting Twitter Data. R package version 0.6.7 Retrieved from https://cran.r-project.org/package=rtweet

Silge J, Robinson D (2016). "tidytext: Text Mining and Analysis Using Tidy Data Principles in R." *JOSS*, *1*(3). doi: 10.21105/joss.00037 (URL http://doi.org/10.21105/joss.00037), <URL:http://dx.doi.org/10.21105/joss.00037>.

D. Kahle and H. Wickham. ggmap: Spatial Visualization with ggplot2. The R Journal, 5(1), 144-161. URL http://journal.r-project.org/archive/2013-1/kahle-wickham.pdf

Garrett Grolemund, Hadley Wickham (2011). Dates and Times Made Easy with lubridate. Journal of Statistical Software, 40(3), 1-25. URL http://www.jstatsoft.org/v40/i03/.

Benoit K, Watanabe K, Wang H, Nulty P, Obeng A, Müller S, Matsuo A (2018). "quanteda: An R package for the quantitative analysis of textual data." *Journal of Open Source Software*,

3(30), 774. doi: 10.21105/joss.00774 (URL: http://doi.org/10.21105/joss.00774), <URL: https://quanteda.io>.

Stefan Feuerriegel and Nicolas Proellochs (2019). SentimentAnalysis: Dictionary-Based Sentiment Analysis. R package version 1.3-3. https://CRAN.R-project.org/package=SentimentAnalysis

Barret Schloerke, Jason Crowley, Di Cook, Francois Briatte, Moritz Marbach, Edwin Thoen, Amos Elberg and Joseph Larmarange (2018). GGally: Extension to 'ggplot2'. R package version 1.4.0. https://CRAN.R-project.org/package=GGally

H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

Gergely Daróczi and Roman Tsegelskyi (2018). pander: An R 'Pandoc' Writer. R package version 0.6.3. https://CRAN.R-project.org/package=pander

Beauchamp, N. 2017. "Predicting and Interpolating State-Level Polls Using Twitter Textual Data." *American Journal of Political Science* 61 (2).

Chang, W., J. Cheng, J. Allaire, Y. Xie, and J. McPherson. 2019. "Shiny: Web Application Framework for R."

Gaurav, M., A. Srivastava, A. Kumar, and S. Miller. 2013. "Leveraging Candidate Popularity on Twitter to Predict Election Outcome." *Proceedings of the 7th Workshop on Social Network Mining and Analysis*.

Gayo-Avello, D. 2013. "A Meta-Analysis of State-of-the-Art Electoral Prediction from Twitter Data." *Organization Studies* 31 (6): 211–48.

Kahle, D., and H. Wickham. 2013. "Ggmap: Spatial Visualization with Ggplot2." *The R Journal* 5 (1): 144–61.

Kassraie, P., A. Modirshanechi, and H. Aghajan. 2017. "Election Vote Share Prediction Using a Sentiment-Based Fusion of Twitter Data with Google Trends and Online Polls." *Proceedings of the 6th International Conference on Data Science, Technology and Applications*.

Massicotte, P., and D. Eddelbuettel. 2019. "gtrendsR: Perform and Display Google Trends Queries."

Salunkhe, P., and S. Deshmukh. 2017. "Twitter Based Election Prediction and Analysis." *International Research Journal of Engineering and Technology* 4 (10).