Fundamentals of Computing and Data Display

Term paper template

Author

2019-12-06

Contents

TO DO: - write intro, results, and discussion sections - save the following datasets to github and write code in designated chunks to bring them in: props__tab1 props__tab2 gc__locations interest__over__time__all pre__sent__pos post__sent__pos - create presentation

Introduction

This section outlines the research idea. We can also cite related work here (Wickham 2014; Baumer, Kaplan, and Horton 2017).

Note that compiled term paper (the PDF) is supposed to be more text-centered than the RMarkdown documents we used in class, i.e. the text sections are more detailed and big or redundant code chunks can be hidden.

Data

This section describes the data sources and the data gathering process.

Twitter

Below is our token to access the Twitter API and start collecting tweets.

```r
create_token(
  app = "fcdd-course",
  consumer_key = "7y8Xu96DRkBoLsdP8XbQJXTn9",
  consumer_secret = "MCM7gcAfTJKwHks2yIRZaDZcODRQCJBusiYoZtobj4Xv07bvpV",
  access_token = "967157671353864192-DvdUZcJN1ocpqx74YN2GwyejeIMR1Da",
  access_secret = "KkCoMdH1vuivbWgH6IdDaDuiScNcxr389caRbhG2L2kXW"
)
```

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 #ha
                     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 ",  s
```

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.

```
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)))
```

```
df1$BS <-(str_detect(df1$text, regex("#Sanders|@BernieSanders", ignore_case = TRUE)))
df1$KH <-(str_detect(df1$text, regex("#harris |@KamalaHarris",  ignore_case = TRUE)))
df1$JB <-(str_detect(df1$text, regex("#biden |@JoeBiden", ignore_case = TRUE)))
df1$EW <-(str_detect(df1$text, regex("#warren |@ewarren", ignore_case = TRUE)))
df1$PB <-(str_detect(df1$text, regex("#buttigieg|@PeteButtigieg", ignore_case = TRUE)))
```

```r
df1$TS <-(str_detect(df1$text, regex("#steyer | @TomSteyer", ignore_case = TRUE)))
df1$AY <-(str_detect(df1$text, regex("#yang| @AndrewYang", ignore_case = TRUE)))
df1$BC <-(str_detect(df1$text, regex("#booker | @BookerCory", ignore_case = TRUE)))
df1$AK <-(str_detect(df1$text, regex("#klobuchar | @amyklobuchar", ignore_case = TRUE)))
df1$TG <-(str_detect(df1$text, regex("#gabbard |@TulsiGabbard", ignore_case = TRUE)))
```

```r
df2$BS <-(str_detect(df2$text, regex("#Sanders|@BernieSanders", ignore_case = TRUE)))
df2$KH <-(str_detect(df2$text, regex("#harris |@KamalaHarris",  ignore_case = TRUE)))
df2$JB <-(str_detect(df2$text, regex("#biden |@JoeBiden", ignore_case = TRUE)))
df2$EW <-(str_detect(df2$text, regex("#warren |@ewarren", ignore_case = TRUE)))
df2$PB <-(str_detect(df2$text, regex("#buttigieg|@PeteButtigieg", ignore_case = TRUE)))
df2$TS <-(str_detect(df2$text, regex("#steyer | @TomSteyer", ignore_case = TRUE)))
df2$AY <-(str_detect(df2$text, regex("#yang| @AndrewYang", ignore_case = TRUE)))
df2$BC <-(str_detect(df2$text, regex("#booker | @BookerCory", ignore_case = TRUE)))
df2$AK <-(str_detect(df2$text, regex("#klobuchar | @amyklobuchar", ignore_case = TRUE)))
df2$TG <-(str_detect(df2$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.

```r
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)

```r
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.

```r
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))

##

bs_post <- post_debate %>%
      filter (BS == "TRUE")

n.sample = 31603
bs_sample = bs_post[sample(1:nrow(bs_post), n.sample, replace=FALSE),]

usableText <- tolower(bs_sample$text)

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

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

```r
kh <- pre_debate %>%
      filter (KH == "TRUE")

usableText <- tolower(kh$text)

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

sentiments_kh = analyzeSentiment(as.character(usableText))

##

kh_post <- post_debate %>%
      filter (KH == "TRUE")

n.sample = 16362
kh_sample = kh_post[sample(1:nrow(kh_post), n.sample, replace=FALSE),]

usableText <- tolower(kh_sample$text)

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

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

```r
jb <- pre_debate %>%
      filter (JB == "TRUE")

usableText <- tolower(jb$text)
```

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

sentiments_jb = analyzeSentiment(as.character(usableText))

##
jb_post <- post_debate %>%
      filter (JB == "TRUE")

n.sample = 27477
jb_sample = jb_post[sample(1:nrow(jb_post), n.sample, replace=FALSE),]

usableText <- tolower(jb_sample$text)

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

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

```r
ew <- pre_debate %>%
      filter (EW == "TRUE")

usableText <- tolower(ew$text)

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

sentiments_ew = analyzeSentiment(as.character(usableText))

##
ew_post <- post_debate %>%
      filter (EW == "TRUE")

n.sample = 18628
ew_sample = ew_post[sample(1:nrow(ew_post), n.sample, replace=FALSE),]

usableText <- tolower(ew_sample$text)

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

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

```r
pb <- pre_debate %>%
      filter (PB == "TRUE")

usableText <- tolower(pb$text)

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

sentiments_pb = analyzeSentiment(as.character(usableText))

##

pb_post <- post_debate %>%
      filter (PB == "TRUE")
```

```
n.sample = 27242
pb_sample = pb_post[sample(1:nrow(pb_post), n.sample, replace=FALSE),]

usableText <- tolower(pb_sample$text)

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

sentiments_pb_post = 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))
```

```
post_sent_pos <- data.frame("Candidate" = c("Biden", "Sanders", "Warren", "Harris", "Buttigieg"),
                        "Positive" =c((sum(sentiments_jb_post$PositivityGI))/27477,
                                      (sum(sentiments_bs_post$PositivityGI))/31603,
                                     (sum(sentiments_kh_post$PositivityGI))/16362,
                                      (sum(sentiments_pb_post$PositivityGI))/27242))
```

Google Trends

This section describes gathering the Google Trends data using the package gtrends. 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 gtrends only allows us to use five search terms at a time. We limit the location of searches to the US.

```
register_google(key = "AIzaSyChFy9VXwWjDdXWWxWY2Qce_cRgaE6bNVM")
res1 <- gtrends(c("Joe Biden", "Bernie Sanders", "Elizabeth Warren", "Kamala Harris", "Pete Buttigieg")
                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 =
                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, 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/Project%20polling%20
temp_file <- tempfile(fileext = ".xlsx")
req <- GET(github_link,
           write_disk(path = temp_file))
polling_data <- readxl::read_excel(temp_file)
```

Results

This section presents the main results.

Data exploration

The results section may have a data exploration part, but in general the structure here depends on the specific project.

Twitter

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

Create functions

```r
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)
}
```

```r
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)
```

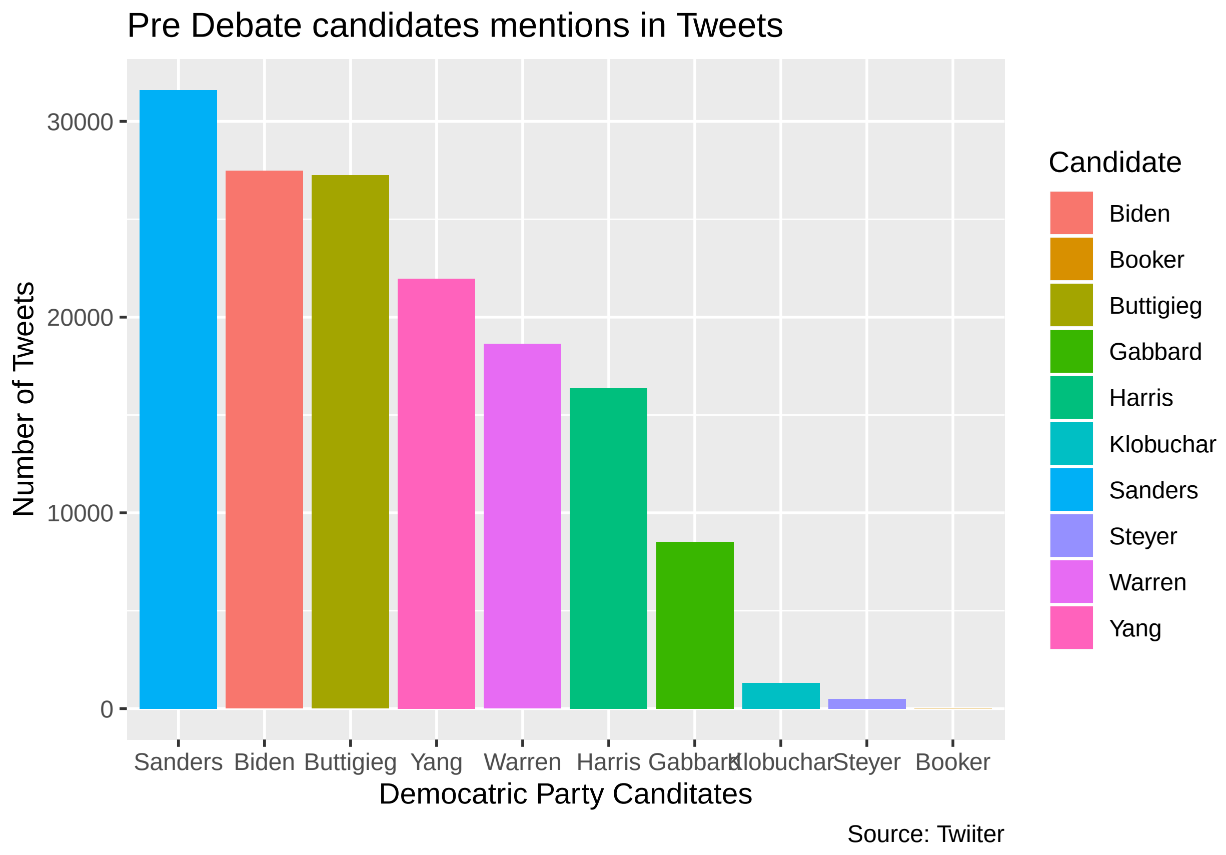
```r
props_tab1 <- props_tab1 %>%
mutate(Variable = rep(names(vars), vars),
       Candidate = ifelse(Variable == "BS", "Sanders",
                   ifelse(Variable == "KH", "Harris",
                   ifelse(Variable == "JB", "Biden",
                   ifelse(Variable == "EW", "Warren",
                   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

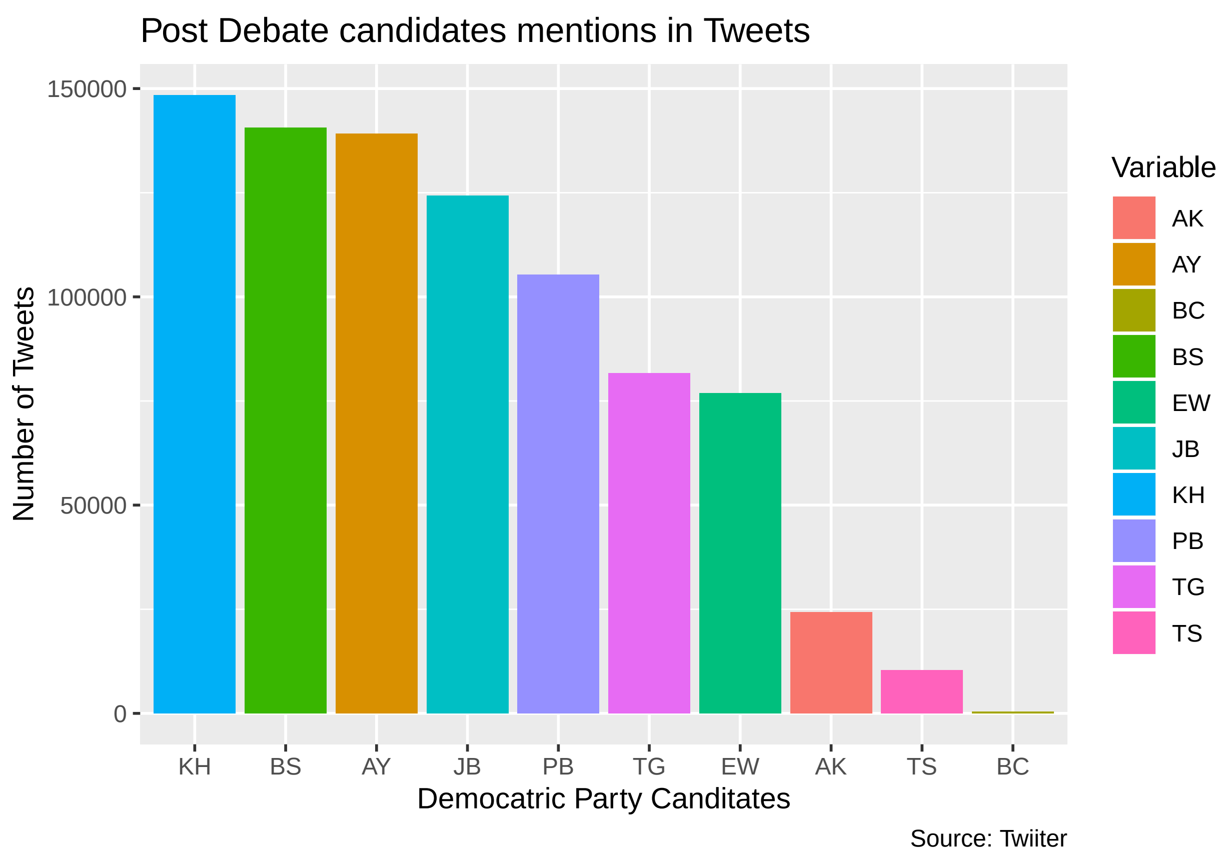
Pre-debate number of tweets

```r
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",
       title = "Pre Debate candidates mentions in Tweets",
       caption = "Source: Twiiter")
```



Post-debate number of tweets

```r
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",
       title = "Post Debate candidates mentions in Tweets",
       caption = "Source: Twiiter")
```



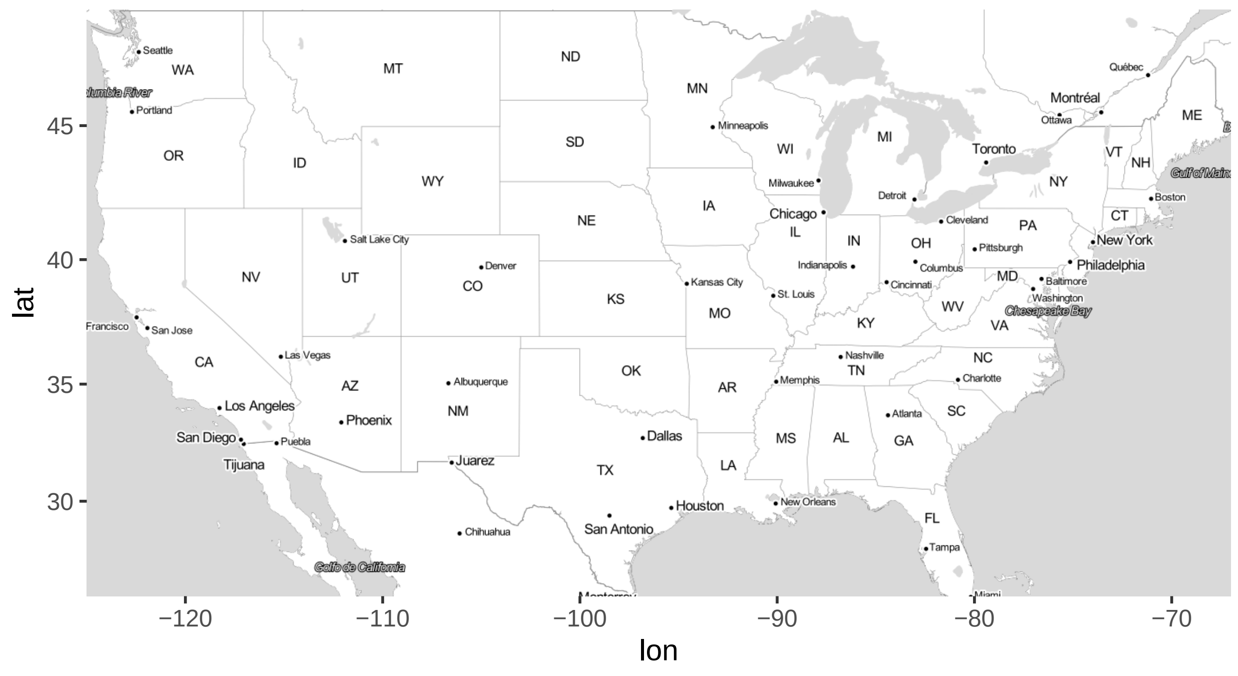
Google Trends

To explore the Google Trends data, we examined the location and density of searches for each candidate. We created a map showing our results. Because it was difficult to see the results for each candidate, we created a shiny app that allows users to toggle between candidates.

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

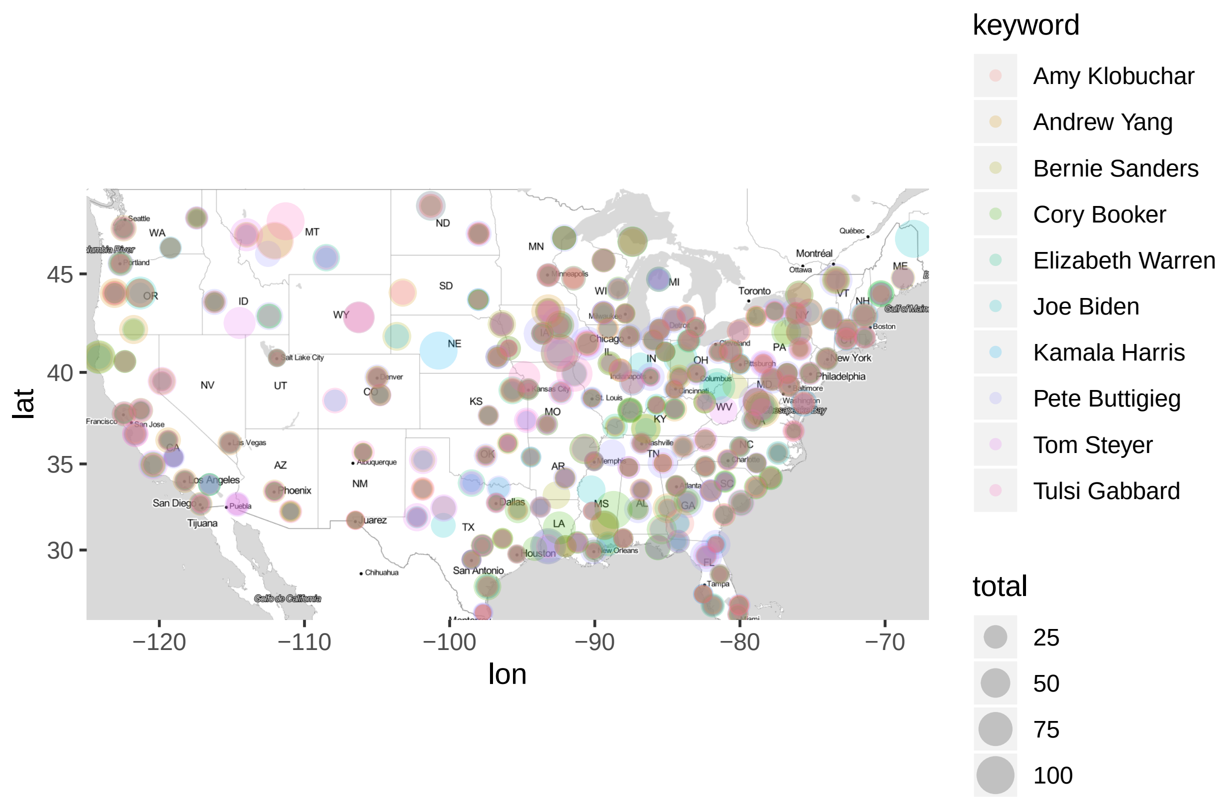
```
## Source : http://tile.stamen.com/toner-lite/5/4/10.png

## Source : http://tile.stamen.com/toner-lite/5/5/10.png

## Source : http://tile.stamen.com/toner-lite/5/6/10.png

## Source : http://tile.stamen.com/toner-lite/5/7/10.png

## Source : http://tile.stamen.com/toner-lite/5/8/10.png

## Source : http://tile.stamen.com/toner-lite/5/9/10.png

## Source : http://tile.stamen.com/toner-lite/5/10/10.png
## Source : http://tile.stamen.com/toner-lite/5/4/11.png

## Source : http://tile.stamen.com/toner-lite/5/5/11.png

## Source : http://tile.stamen.com/toner-lite/5/6/11.png

## Source : http://tile.stamen.com/toner-lite/5/7/11.png

## Source : http://tile.stamen.com/toner-lite/5/8/11.png

## Source : http://tile.stamen.com/toner-lite/5/9/11.png

## Source : http://tile.stamen.com/toner-lite/5/10/11.png

## Source : http://tile.stamen.com/toner-lite/5/4/12.png

## Source : http://tile.stamen.com/toner-lite/5/5/12.png

## Source : http://tile.stamen.com/toner-lite/5/6/12.png

## Source : http://tile.stamen.com/toner-lite/5/7/12.png

## Source : http://tile.stamen.com/toner-lite/5/8/12.png

## Source : http://tile.stamen.com/toner-lite/5/9/12.png

## Source : http://tile.stamen.com/toner-lite/5/10/12.png

## Source : http://tile.stamen.com/toner-lite/5/4/13.png

## Source : http://tile.stamen.com/toner-lite/5/5/13.png

## Source : http://tile.stamen.com/toner-lite/5/6/13.png

## Source : http://tile.stamen.com/toner-lite/5/7/13.png

## Source : http://tile.stamen.com/toner-lite/5/8/13.png

## Source : http://tile.stamen.com/toner-lite/5/9/13.png

## Source : http://tile.stamen.com/toner-lite/5/10/13.png
```

```r
ggmap(us_map)
```



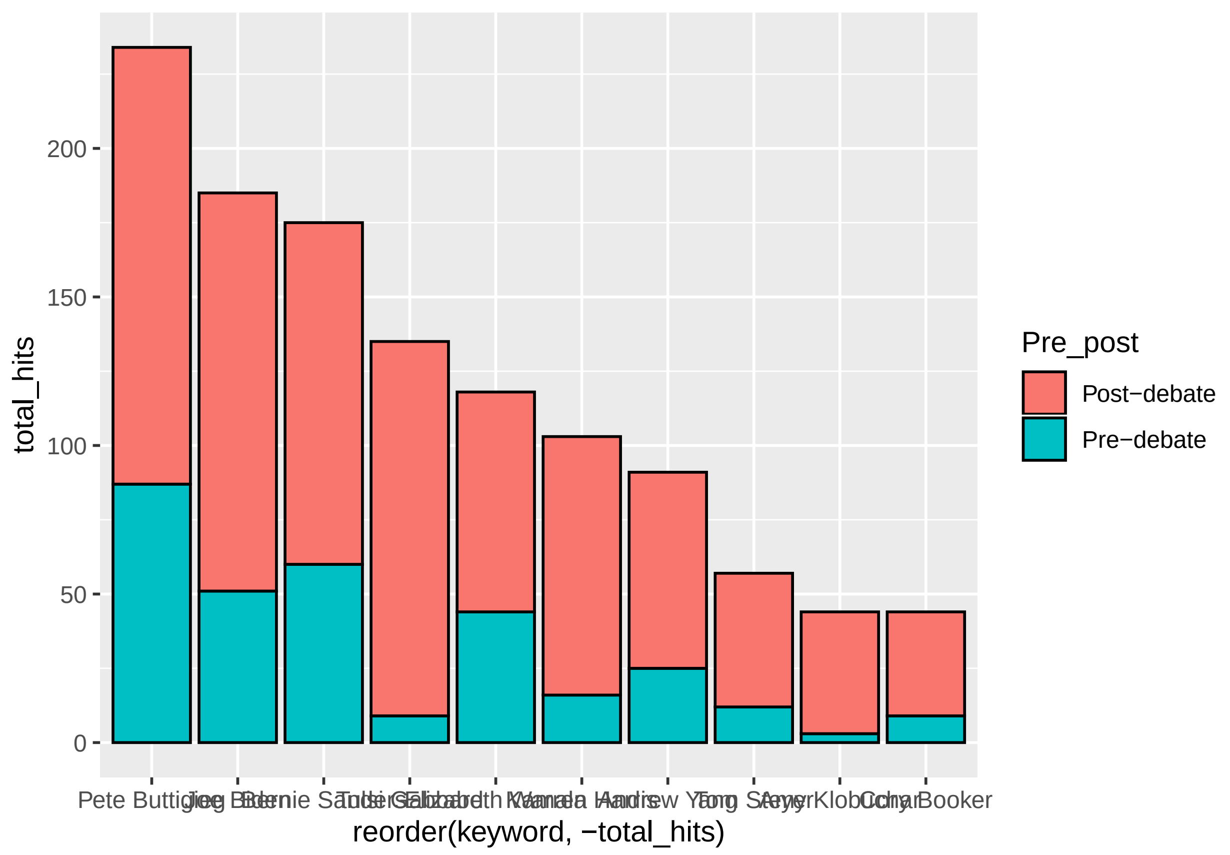
```r
#clean data for mapping
trends_map <-
  gc_locations %>%
  group_by(keyword, location) %>%
  mutate(total=sum(hits))

#generate map
ggmap(us_map) +
  geom_point(data = trends_map, aes(x = lon, y = lat, size=total, color=keyword), alpha = 0.2)
```

```
## Warning: Removed 717 rows containing missing values (geom_point).
```



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

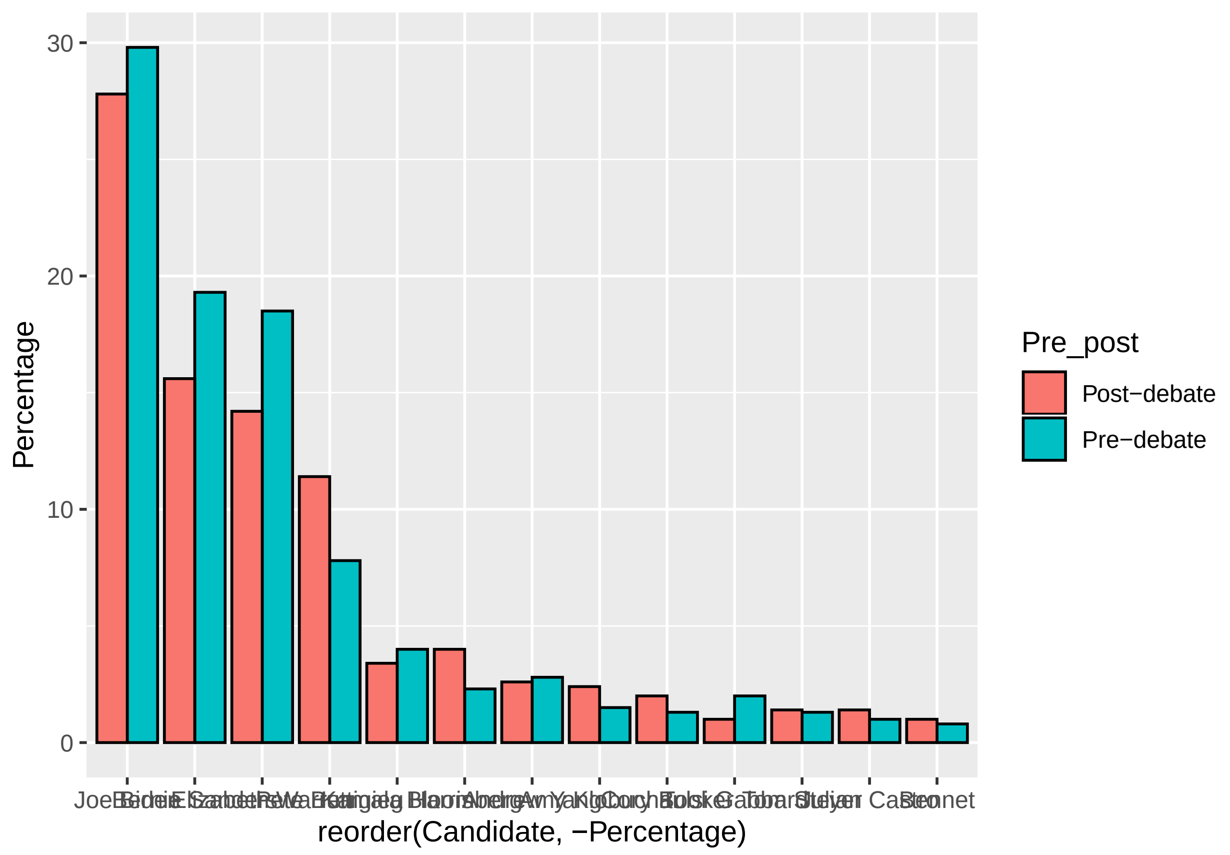
```
interest_over_time_all %>%
  group_by(keyword, Pre_post) %>%
  summarise(total_hits=sum(hits)) %>%
  ggplot() +
  geom_col(mapping = aes(x=reorder(keyword, -total_hits), y=total_hits, fill=Pre_post), color="black")
```



Polling Data

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

```
polling_data %>%
  group_by(Candidate, Pre_post) %>%
  ggplot() +
    geom_col(mapping = aes(x=reorder(Candidate, -Percentage), y=Percentage, fill=Pre_post), color="blac
```



Analysis

We created a Shiny app that allows us to compare the polling, Google Trends, and Twitter sentiment data. 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-debat
                   "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)
```

```
## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector
```

Then, we created a shiny app, which can be viewed by accessing the following site: ______________________.

```
# 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", "Elizab
                  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)
  })
}

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

Shiny applications not supported in static R Markdown documents

```
# What happens here depends on the specific project
```

```
# What happens here depends on the specific project
```

Discussion

This section summarizes the results and may briefly outline advantages and limitations of the work presented.

References

Baumer, Benjamin S., Daniel T. Kaplan, and Nicholas J. Horton. 2017. *Modern Data Science with R*. Chapman & Hall/CRC Press.

Wickham, Hadley. 2014. "Tidy Data." *Journal of Statistical Software* 59 (10): 1–23.