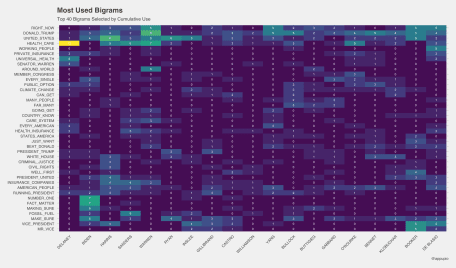
This tutorial will mainly focus on ggplot and bigrams, but it does gloss over clustering for a heatmap.



This project started a while back, tweeting the plots at the beginning of this month. Life happens I suppose. Bought a new bike, had a birthday, yaddayadda. Better late then never?

I want to preface this with the disclaimer that a phrase repeated isn’t inherently good or bad. Emphasis through repetition is sometimes needed to drive a point home.

**Required Libraries**

First step, as always, is to include the libraries we will be using.

# Always Included

library(tidyverse)

library(magrittr)

# Plotting

library(ggplot2)

library(ggthemes)

# String manipulation

library(stringr)

# Tokenization

library(quanteda)

# Heatmap rescaling

require(scales)

library(dplyr)

# Just for melt()

library(data.table)

**Define Candidates**

This appends the candidates from the second debate to the candidates of the first debate.

candidates <- c("Sanders",

"Klobuchar",

"Warren",

"Buttigieg",

"O'Rourke",

"Bullock",

"Delaney",

"Ryan",

"Williamson") %>%

append(c("Bennet",

"Gillibrand",

"Castro",

"Booker",

"Harris",

"Biden",

"Yang",

"Gabbard",

"Inslee",

"de Blasio")) %>% toupper

**Reading the Transcripts**

I ended up grabbing the transcripts from [nbcnews.com](http://nbcnews.com) and saving as a CSV file after some regex cleaning. I couldn’t find both days on their site anymore, so I am linking to the CNN transcripts

* [Night One Transcript](http://transcripts.cnn.com/TRANSCRIPTS/1907/30/se.01.html)
* [Night Two Transcript](http://transcripts.cnn.com/TRANSCRIPTS/1907/31/se.02.html)

I also kept the transcripts separate just in case I needed to refer to Night One and Night Two separately down the road. Once I read them in I noticed some white space and that I still needed to remove the ‘:’ character.

transcriptA <- read\_csv("2019-07-30.csv",col\_names = F,trim\_ws = T,quote = '"')

names(transcriptA) <- c("person","dialog")

transcriptA$person %<>% str\_replace\_all(":","")

transcriptA$dialog %<>% trimws

transcriptB <- read\_csv("2019-07-31.csv",col\_names = F,trim\_ws = T,quote = '"')

names(transcriptB) <- c("person","dialog")

transcriptB$person %<>% str\_replace\_all(":","")

transcriptB$dialog %<>% trimws

Now that we have Transcript A and B ready in and using the same column names, we can bind them.

transcript <- rbind(transcriptA,transcriptB)

If we wanted to keep our workspace clean, this would be an excellent opportunity to save the transcripts in a list (transcript$A and transcript$B). Using that method would allow you to use transcript$Full <- transcript %>% bind\_rows which looks cleaner.

**Working with Bigrams**

This part might get a little overwhelming, but essentially this chunk of code will

1. Loop through each individual candidate
   1. Subset the transcript by current candidate
   2. Loop through Dialog of current subset
      1. Return bigrams
   3. Generate frequency table of returned bigrams
   4. Add column for current candidate

The reason we are nesting an lapply instead of collapsing is to prevent the end of a sentence to be used with the beginning of a new sentence (ex: “He fell in. The boy cried” shouldn’t include the bigram “IN\_THE”). While generating n-grams on each dialog separately won’t prevent this, it will reduce occurrences.

If you want to further improve upon this code, you could split the dialog by punctuation marks c('?', '!', '.', ';').

bigrams <- lapply(unique(transcript$person),function(candidate) {

lapply(transcript %>% filter(person==candidate) %>% .[["dialog"]], function(text) {

text %>% str\_remove\_all('\\.\\.\\.') %>%

tokens(remove\_numbers = TRUE, remove\_punct = TRUE) %>%

tokens\_select(pattern = stopwords('en'), selection = 'remove') %>%

tokens\_ngrams(n = 2) %>% toupper %>% unique

}) %>% unlist %>% table %>% data.frame -> df

if(nrow(df)>0) {

df$Candidate <- candidate

return(df)

} else {

return(NULL)

}

})

names(bigrams) <- unique(transcript$person)

If you want to give the results a test, you can use

bigrams$WARREN %>% top\_n(n = 10, wt = Freq)

. Freq Candidate

1 ACROSS\_COUNTRY 3 WARREN

2 AROUND\_WORLD 5 WARREN

3 CARE\_SYSTEM 3 WARREN

4 COURAGE\_FIGHT 3 WARREN

5 DONALD\_TRUMP 8 WARREN

6 ENTIRE\_WORLD 3 WARREN

7 FIGHT\_BACK 4 WARREN

8 GOD-GIVEN\_RIGHT 3 WARREN

9 HEALTH\_CARE 7 WARREN

10 INSURANCE\_COMPANIES 3 WARREN

11 RIGHT\_NOW 6 WARREN

12 UNITED\_STATES 5 WARREN

Pretty cool, right?

**Bigrams, Extended**

Now that we have everything all nice and segmented, we will be merging everything into one big table bigram\_table to plot.

bigram\_table <- bigrams %>% bind\_rows

# Renaming, but only the first column is changed

names(bigram\_table) <- c("Gram","Freq","Candidate")

# Filter out non-candidates (announcer)

bigram\_table %<>% filter(Candidate %in% candidates)

# Create new column

bigram\_table$Repeat <- ifelse(bigram\_table$Freq>1,"Repeated","Original")

# Now some grouping to determine percentages

bigram\_table <- bigram\_table %>% group\_by(Candidate,Repeat) %>%

summarise(n = sum(Freq)) %>% mutate(Percentage = (n / sum(n))\*100)

# Label column added, but only will show repeated

bigram\_table$Label <- NA

bigram\_table$Label[bigram\_table$Repeat=="Repeated"] <- bigram\_table$Percentage[bigram\_table$Repeat=="Repeated"] %>%

round(digits = 2) %>% paste0("%")

**Plotting Originality**

ggplot(bigram\_table,

aes(x = factor(Repeat,levels=c("Repeated",

"Original")),

y = Percentage,

label = Label,

fill = Repeat)) +

geom\_bar(stat="identity") + coord\_flip() +

scale\_y\_continuous(breaks = c(0,25,50,75,100),

labels = c("0%",

"25%",

"50%",

"75%",

"100%")) +

geom\_text(nudge\_y = 15) +

theme\_fivethirtyeight() +

scale\_fill\_economist() +

facet\_wrap(~Candidate) +

labs(title = "Bigram Originality within Candidate Statements",

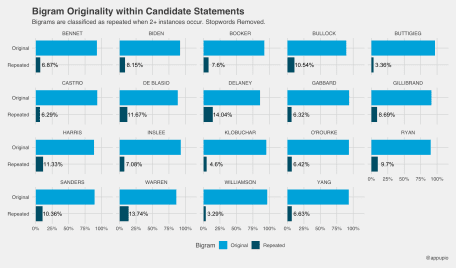
subtitle = paste0("Bigrams are classificed as repeated when ",

"2+ instances occur. ",

"Stopwords Removed."),

caption = "@appupio",

fill = "Bigram")



Cool!

**Heatmap**

**Clustering Bigrams**

This next part is going to be a lot of piping, and I am sure someone has a much better way of doing things.

First we going to overwrite bigrams table with a fresh bind\_rows call on the bigrams list.

bigram\_table <- bigrams %>%

bind\_rows %>%

select(Gram = '.', Freq, Candidate) %>%

filter(Candidate %in% candidates)

I did the part above a little different than the first time. There are a handful of ways to rename columns of a data frame. Using select is a very nice alternative.

For the next part we will want to figure out what bigrams to use. I am selecting the top 40 used the most cumulatively among all candidates. We only need a vector of the actual grams.

top\_grams <- bigram\_table %>%

group\_by(Gram) %>%

summarise(Freq = sum(Freq)) %>%

.[rev(order(.$Freq)),"Gram"] %>%

unlist %>% as.vector

To give that a test we can use

top\_grams[1:10]

Looks like what we are looking for; let’s move on.

[1] "DONALD\_TRUMP" "UNITED\_STATES"

[3] "HEALTH\_CARE" "RIGHT\_NOW"

[5] "MAKE\_SURE" "AMERICAN\_PEOPLE"

[7] "VICE\_PRESIDENT" "RUNNING\_PRESIDENT"

[9] "WHITE\_HOUSE" "INSURANCE\_COMPANIES"

Time to filter bigram\_table and convert to a matrix.

cluster\_matrix <- bigram\_table %>%

filter(Gram %in% top\_grams[1:40]) %>%

group\_by(Gram,Candidate,Freq) %>%

spread(Candidate,Freq)

cluster\_matrix[[is.na](http://is.na)(cluster\_matrix)] <- 0

# numerical columns

dat <- cluster\_matrix[,2:(ncol(cluster\_matrix))] %>% as.data.frame

row.names(dat) <- cluster\_matrix$Gram

# clustering

row.order <- hclust(dist(dat))$order

col.order <- hclust(dist(t(dat)))$order

# re-order matrix accoring to clustering

dat\_new <- dat[row.order, col.order]

# reshape into dataframe

cluster\_matrix <- melt(as.matrix(dat\_new))

names(cluster\_matrix) <- c("Gram", "Candidate","Freq")

Uff-da. Now that all of that is over, we can plot cluster\_matrix.

**Plotting the Heatmap**

Lots of ways to style the heatmap, but I am going with a viridis heatmap and including those frequency within the cells. Sometimes you also want numbers.

ggplot(cluster\_matrix,aes(x = Candidate,

y = Gram,

fill = Freq,

label = Freq)) +

geom\_tile() + scale\_fill\_viridis\_c() +

geom\_text(color="#FFFFFF",size=2) +

theme\_fivethirtyeight() +

theme(axis.text.x = element\_text(angle = 45,

hjust = 1)) +

theme(legend.position="none",

text = element\_text(size=9)) +

labs(title = "Most Used Bigrams",

subtitle = "Top 40 Bigrams Selected by Cumulative Use",

caption = "@appupio")

