Language Trend Analysis Across Twitter in the 2016 United States Presidential Election

Luke Taylor
Dennis Walsh
Thomas Flaherty

November 27, 2016

Abstract

Language is always evolving. As the internet, and especially social media, has increased the availability and immediateness of communication to being able to share a thought with everyone in an instant, everyday language now becomes abundant data that is easier than ever to source, sort and analyze. Our objective is to explore this data and identify trends in the 2016 US Presidential election. Trends are defined as the most frequent words or topics discussed in Twitter messages (tweets) over time or by area. Extensive cleaning of data will occur so that searches for trends can occur. These trends in turn will become data products. Insights into emerging trends can be seen by identifying keywords and their associated words. Trends can be visualized over time and geography, because Twitter data has the ability to attach geographic information to the Tweets.

1 Introduction

What does a twitter analysis tell us about the 2016 Presidential election? In a Presidential election many pieces of data are analyzed to determine strategy and behavior. Polls, voting history, television reports, and newspaper reports all come into play. In recent years social media has become part of the mix. Analysis of twitter activity can provide another window into what is happening in the election and why.

Twitter is a popular and important social media platform. Twitter had 320 million monthly active users worldwide in the last quarter of 2015. 65 million of those were in the United States, or 1 in 5 Americans. Usage is highest with adults under age 50, residents of cities, and upper-income Americans (Desilver).

Twitter reflects the attitudes only of certain subsets of the United States population. Pew Research has found that "the reaction on Twitter to major political events and policy decisions often differs considerably from general public opinion" (Desilver). Twitter opinions run more negative than the general public. Twitter users are younger and lean Democratic. Twitter is broader than most public opinion surveys because those under 18, as well as those outside the United States, can participate (Mitchell and Hitlin).

Social media does not necessarily mean conversations in the public square. Often it can mean two groups, in separate houses, talking among themselves. This type of twitter conversation has been called conversation within "Polarized Crowds, where opposed groups talk about the same topic but mostly just to other group members" (Desilver). Thus it is appropriate to run separate searches on "Trump" and "Clinton" to look at what those polarized crowds are discussing. A search for "Obama" provides a contrast to the current candidates.

Research of tweets is not easily used to predict an election result, as the users of twitter represent a biased sample of the population (Gayo-Avello). But tweets can still reflect what issues and values are important to each side, how those conflicts rise and fall during the election, and the reaction to the results.

1.1 Preparing this document

This whole document is prepared using **R** [**R-base**] package `knitr` [**R-knitr**]. It is a dynamic document and reproducible any number of times for any data sets.

2 About the data

The sample data has been added to a GitHub project and is available via a web link.

The data set is a continuous json stream provided by the public Twitter sample API. This particular end point of the API provides a representative sample of data that a private individual may consume. This is contrasted by the private Twitter firehose stream which streams all of the Twitter statuses. By reserving the firehose API to a select few consumers with the knowledge and technology to utilize the firehose stream, Twitter limits any issues that would be caused by an average developer making a mistake while programming against the firehose API.

Although the sample API does not provide all of the tweets all of the time, it does provide a representative sample of data, as heavily tweeted events tend to generate representative samples (Morstatter). While a city council campaign might not show up appropriately in the twitter sample stream, a US presidential contest should.

The json data provided by the public sample streaming API contains many fields and is considered structured data. Twitter's data objects are provided here: https://dev.twitter.com/overview/api

The data used for this project comes from capturing the raw json output of sample stream.

Because of the large amount of data and the desire to analyze trends over a longer span of time than Twitter provides via its public API, it was decided to store the data in a SQL Server 2016 database. Data is parsed from JSON into a database table. The parsed data does not contain all possible fields and is indexed on the date field of the tweet.

For the purposes of this document a sample of the parsed data is stored on GitHub and used for this analysis. To reproduce the data capture and transformation techniques required to get the data to this stage, refer to appendix XXX. Once in SQL Server, the parsed data is limited to only the rows desired for analysis. There are 10000 data points and 14 variables in this data set. The variables are X, MessageId, CreatedAt, UserId, PlaceId, PlaceType, PlaceName, PlaceFullName, PlaceCountryCode, PlaceCountry, PlaceBoundingBox, CoordinatesType, CoordinatesCoordinates, Text.

Below is a function defining the to call the SQL Database which retreives the parsed data along with a command which assigns the result set to a local variable. The query can be modified to limit results to specific time frames or only to tweets that contain a textual value using the T-SQL syntax available in SQL Server 2016.

```
GetDataSet <- function (term = NA, count = 1000) {
  count <- format(count,scientific = FALSE)
  library(RODBC)
  dbhandle <- odbcConnect("MyODBCConnectionName",
                       uid="MyUserId",
                       pwd="MyPassword")
  queryParts = c("SELECT TOP (",count,") [MessageId]
                      ,[CreatedAt]
                      ,[UserId]
                      ,[PlaceId]
                      ,[PlaceType]
                      ,[PlaceName]
                      ,[PlaceFullName]
                      ,[PlaceCountryCode]
                      ,[PlaceCountry]
                      ,[PlaceBoundingBox]
                      ,[CoordinatesType]
                      ,[CoordinatesCoordinates]
                      ,[Text]
                  FROM [TwitterData].[dbo].[NewFlattenedTweets]")
  if (!is.na(term)) {queryParts <- c(queryParts," WHERE [Text] LIKE '%",term,"%'")}
  query <- paste(queryParts, collapse='')
  res <- sqlQuery(dbhandle, query)
  odbcClose(dbhandle)
  return (res);
}

# Usage
dataSet <- GetDataSet("Trump",10000);
```

2.1 Preparing data

The most salient value of this data is with the combination of the words of the tweet and the geographic information included with the tweets. In order to prepare the data for textual analysis which includes word frequency counts, word relationships, and word sentiment, it will take the form of a corpus. The corpus structure used for this analysis is part of the tm package.

Before the textual data can be used, it must be prepared by stripping out characters that interfere with english word analysis. The various transformations may or may not be desired based on the type of analysis being performed. In order to encapsulate the process, it is wrapped in a function and passed a character vector consisting of the tweet text.

```
dataSet <- read.csv('https://github.com/lbtaylor/DataScienceGroup11/raw/master/SampleTwitterData.csv')

library(tm)

## Loading required package: NLP

TwitterToCorpus <- function(twitterData, searchTerm = NULL) {
  # Load the NLP library
  library(tm)

  if (exists('searchTerm') && (is.null(searchTerm)==FALSE)){
    searchTerm <- tolower(searchTerm)
  }
  else {
    searchTerm <- 'NOTPOSSIBLE'
  }
  # Load the tweet text
  corpusData <- as.character(twitterData)
  #corpusData <- as.character(dataSet£Text)

  # Convert the character vector to a Corpus object
  myCorpus <- Corpus(VectorSource(corpusData))

  # Convert the text to lower case
  myCorpus <- tm_map(myCorpus, content_transformer(tolower))
  # Remove URL's from corpus
  removeURL <-
    content_transformer(function(x)
      gsub(" ?(f|ht)(tp)(s?)(://)(.*)[.|/](.*)", " ", x))
  myCorpus <- tm_map(myCorpus, removeURL)

  # Remove mentions (this may or may not be desired based on the type of anyalysis)
  removeMentions <-
    content_transformer(function(x)
      gsub("(\\b\\S*@|@)\\S*\\b", " ", x))
  myCorpus <- tm_map(myCorpus, removeMentions)

  # Remove numbers
  myCorpus <- tm_map(myCorpus, removeNumbers)

  # Remove common stopwords
  # Specify additional stopwords or stems as a character vector
  myExtraStopwords <-
    c("available", "via", "amp", "get", "com", "pra", "just")
  myStopwords <- c(stopwords("english"), stopwords("spanish"), myExtraStopwords)
  myCorpus <- tm_map(myCorpus, removeWords, myStopwords)

  # Remove punctuations
  myCorpus <- tm_map(myCorpus, removePunctuation)

  # Replace all characters that are not in a-z with a space
  removeNonLowerAlpha <-
    content_transformer(function(x)
      gsub("[^a-z]", " ", x))
  myCorpus <- tm_map(myCorpus, removeNonLowerAlpha)

  # Remove all terms that contain the search term
  pattern <- paste("\\b\\S*", searchTerm, "\\S*\\b", sep = '')
  removeSearchTermWild <-
    content_transformer(function(x)
      gsub(pattern, " ", x))

  myCorpus <- tm_map(myCorpus, removeSearchTermWild)


  # Eliminate extra white spaces
  myCorpus <- tm_map(myCorpus, stripWhitespace)

  return (myCorpus)
}

# Get the cleaned corpus
myCorpus <- TwitterToCorpus(dataSet$Text)

as.character(myCorpus[[550]])

## [1] " supposed homework attention keeps drifting twitter"
```

Now the corpus is ready for use. The corpus will be stemmed which associates words with their most basic root so that terms like 'women' and 'woman' will show up as the same word, 'wom.'

```
# Make a copy of the corpus
myCorpusCopy <- myCorpus

# Text stemming
#myCorpus <- tm_map(myCorpus, stemDocument)

# Stem Stuff
#myCorpus <- tm_map(myCorpus, stripWhitespace)
#myCorpus <- tm_map(myCorpus,PlainTextDocument)
#inspect(myCorpus)
```

Once the text is stemmed and tied back to a reference of the original, unstemmed corpus, it is ready to examine the frequency of words. An out of memory issue will arise when trying to transform a TermDocumentMatrix into a standard R matrix. To overcome this issue, the slam package is used. The problem stems from the fact that, in a very large data matrix where, conceptually, each word represents a row and each column represents a document (each of the tweets is considered a document) the value of that intersection is the frequency for that word in that particular document. In this case, the matrix is large and very sparse XXX (show example of sparseness ratio) XXX. The slam package allows the matrix to be "rolled up" by collapsing each column of data using a function such as sum without first converting the source data into a matrix.

```
# Word Frequency
tdm <- TermDocumentMatrix(myCorpusCopy)
dtm <- DocumentTermMatrix(myCorpusCopy)

head(findFreqTerms(dtm))

## [1] "aaaa"                   "aaaaa"                  "aaaaaa"
## [4] "aaaaaaaaa"              "aaaaaaaaaaaaaaaa"       "aaaaaaaaaaaaaaaaaaaaa"

# The following method may product the error:
# Error: cannot allocate vector of size xxx Gb
## allTermFreqs <- data.frame(unlist(rowSums(inspect(tdm[,dimnames(tdm)£Docs]))))
## colnames(allTermFreqs) <- c('freq')
## allTermFreqs£term <- rownames(allTermFreqs)
## allTermFreqs.sorted <- allTermFreqs[order(allTermFreqs£freq,allTermFreqs£term),]
## head(allTermFreqs.sorted)
## tail(allTermFreqs.sorted)

# Using the slam package, the data can be rolled up and made into a smaller matrix
library(slam)
tdm.rollup <- rollup(tdm, 2, na.rm=TRUE, FUN = sum)
allTermFreqs.tdm.rollup.matrix <- as.matrix(tdm.rollup)
rm(tdm.rollup)
allTermFreqs.tdm.rollup.df <- data.frame(
  rownames(allTermFreqs.tdm.rollup.matrix),
  allTermFreqs.tdm.rollup.matrix[,1],stringsAsFactors = FALSE)
rm(allTermFreqs.tdm.rollup.matrix)
colnames(allTermFreqs.tdm.rollup.df) <- c('term','freq')

allTermFreqs.tdm.rollup.df.sorted <- allTermFreqs.tdm.rollup.df[
  order(allTermFreqs.tdm.rollup.df$freq,allTermFreqs.tdm.rollup.df$term),]
rm(allTermFreqs.tdm.rollup.df)
```

```
head(allTermFreqs.tdm.rollup.df.sorted)

##                                      term freq
## aaaa                                 aaaa    1
## aaaaaa                             aaaaaa    1
## aaaaaaaaa                       aaaaaaaaa    1
## aaaaaaaaaaaaaaa           aaaaaaaaaaaaaaa    1
## aaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaa    1
## aaaaaaah                         aaaaaaah    1

tail(allTermFreqs.tdm.rollup.df.sorted)

##      term freq
## see   see  123
## day   day  127
## job   job  150
## love love  167
## can   can  171
## like like  196

topFreqWords <- tail(allTermFreqs.tdm.rollup.df.sorted, 10)

#allTermFreqs.tdm.rollup£term <- rownames(allTermFreqs.tdm.rollup)
#allTermFreqs.tdm.rollup.sorted <-
#  allTermFreqs.tdm.rollup[
#  order(allTermFreqs.tdm.rollup£freq,allTermFreqs.tdm.rollup£term),]
```

```
# now that we have a list of the most frequent words, we can find the words associated with them

wordAssocs <- findAssocs(tdm, topFreqWords$term, 0.07)
#sapply(wordAssocs, "[[", 1:5)
wordAssocs.top <- lapply(wordAssocs , '[' , 1:10 )

wordAssocs.df <- NULL

for (i in 1:length(wordAssocs.top)) {
  topTerm <- names(wordAssocs.top[i])
  df.new <- data.frame(topTerm = topTerm,
                       relatedTerm = names(wordAssocs.top[[i]]),
                       relatedTermAssoc = as.numeric(wordAssocs.top[[i]]),
                       row.names = NULL)
  wordAssocs.df <- rbind(wordAssocs.df, df.new)
  rm(df.new)
}

wordAssocs.df$topTerm <- factor(wordAssocs.df$topTerm)
# now the word association are in a data frame

p <- ggplot(wordAssocs.df, aes(x=relatedTerm, y=relatedTermAssoc, group = topTerm))

## Error in eval(expr, envir, enclos):  could not find function "ggplot"

p <- p + geom_point(stat="identity")
```

```
## Error in eval(expr, envir, enclos):  object 'p' not found

p <- p + facet_wrap(topTerm, scales = FALSE)

## Error in eval(expr, envir, enclos):  object 'p' not found

p <- p + coord_flip()

## Error in eval(expr, envir, enclos):  object 'p' not found

p

## Error in eval(expr, envir, enclos):  object 'p' not found
```

NOTE: THIS SECTION WILL HAVE A TABLE WITH DATA SUMMARY INFORMATION NOTE: THE STANDARD R SUMMARY IS TOO VERBOSE FOR A PAPER

```
#TODO : Display a table of terms and freqencies

#summary(dataSet)
```

After the text data has been prepared, the coordinates are then pulled from the each tweet. These will be used to show the frequency of tweets pertaining to each data set.

NOTE: THERE IS A SECTION OF CODE HERE THAT IS NOT YET WORKING

```
## Checking rgeos availability:  FALSE
##  Note:  when rgeos is not available, polygon geometry computations in maptools depend on
gpclib,
##  which has a restricted licence.  It is disabled by default;
##  to enable gpclib, type gpclibPermit()
```

```
# Prepare the data set by parsing lat long values from the bounding box and location fields
library('rjson')

dataSet[!(is.na(dataSet$PlaceBoundingBox)),'PlaceBoundingBox']

head(dataSet['PlaceBoundingBox'])
dataSet$PlaceBoundingBox <- as.character(dataSet$PlaceBoundingBox)
head(dataSet$PlaceBoundingBox)

jDat <- fromJSON(head(dataSet$PlaceBoundingBox,1))
unlist(jDat$coordinates)[1:2]

jDat <- fromJSON(dataSet$PlaceBoundingBox)

sampleSet <- head(dataSet$PlaceBoundingBox)

#boundingBoxToLatLong <-

lapply(sampleSet, function(x)  )
#dataSetflat <-

## Error:  <text>:19:32:  unexpected ')'
## 18:
## 19:  lapply(sampleSet, function(x)  )
##                                   ^
```

2.2 Data Challenges

There are many interesting things to be learned from the data. Although the source data from the twitter stream was limited by the query to English tweets, many tweets contained language other than English, including languages that require a the extended UTF character set to display. This indicates that although all twitter accounts were set to use English as the default language, a decent percentage of tweets were non-English. XXX Is it possible to figure out this percentage based on character encoding? XXX

Tweets do not always provide a simple latitude, longitude to geolocate the tweet. Some contain city information, some contain a bounding box with surrounds an area where the tweet would have come from. This creates an addition layer of work in preparing the data and is due to various personal settings Twitter users adjust to control privacy levels.

3 Data product

Twitter searches were conducted on the downloaded data based on three terms: Trump, Clinton, and Obama. The draft will focus on one search, Trump, for explanatory purposes. A series of commands are executed to turn the messy text data into relatively clean data.

```
fa <- findAssocs(tdm, "women", 0.07)
fa

## $women
##            amaze          behave          beings         cheated
##             0.38            0.38            0.38            0.38
##            cruse          engage         havest   inspirational
##             0.38            0.38            0.38            0.38
##        leadright          maples           marla       microcosm
##             0.38            0.38            0.38            0.38
##            naima           naper          powwow        societal
##             0.38            0.38            0.38            0.38
## threegenerations        athletes     generations         hosting
##             0.38            0.27            0.27            0.27
##             maga             pow       screaming         stretch
##             0.27            0.27            0.27            0.27
##           crimes           loose            push           snake
##             0.22            0.22            0.22            0.22
##       successful         certain            goes             nan
##             0.22            0.19            0.19            0.19
##             wife         married             mum           sport
##             0.19            0.17            0.17            0.17
##            human          issues          brunch            fail
##             0.15            0.15            0.14            0.14
##           called             men           wants         forever
##             0.13            0.13            0.13            0.12
##            learn           bring           girls            also
##             0.12            0.10            0.10            0.09
##           enough           trump           whole             wow
##             0.09            0.08            0.08            0.08
##          nothing
##             0.07

as.data.frame(fa)

##                  women
## amaze              0.38
## behave             0.38
## beings             0.38
## cheated            0.38
## cruse              0.38
## engage             0.38
## havest             0.38
## inspirational      0.38
## leadright          0.38
## maples             0.38
## marla              0.38
## microcosm          0.38
## naima              0.38
## naper              0.38
## powwow             0.38
## societal           0.38
## threegenerations   0.38
## athletes           0.27
## generations        0.27
## hosting            0.27
## maga               0.27
## pow                0.27
## screaming          0.27
## stretch            0.27
## crimes             0.22
## loose              0.22
## push               0.22
## snake              0.22
## successful         0.22
## certain            0.19
## goes               0.19
## nan                0.19
## wife               0.19
## married            0.17
## mum                0.17
## sport              0.17
## human              0.15
## issues             0.15
## brunch             0.14
## fail               0.14
## called             0.13
## men                0.13
## wants              0.13
## forever            0.12
## learn              0.12
## bring              0.10
## girls              0.10
## also               0.09
## enough             0.09
## trump              0.08
## whole              0.08
## wow                0.08
## nothing            0.07
```

This data will then be mapped to show it's location. Below is a rough plotting of the given data.

NOTE: THERE IS A SECTION OF CODE HERE THAT IS NOT YET WORKING NOTE: IT IS MEANT TO GENERATE A MAP OF TWEET LOCATIONS ¡¡¡¡¡¡ HEAD

```
#library(maps)
#map("world", fill=TRUE, col="white", bg="lightblue", ylim=c(-60, 90), mar=c(0,0,0,0))
#points(x.cord,y.cord, col="red", pch=16)
```

4 Results

In result section you can start with an overview of what you have found during the exploration of data.

4.1 Including tables

SECTION INCOMPLETE - WORKING ON NICE LOOKING TABLES.

```
dailyCounts <- read.csv("counts_wide.csv")
head(dailyCounts)

##         date Trump Clinton Obama
## 1 09/25/2016   113      38    26
## 2 09/26/2016  1812     735   123
## 3 09/27/2016 97267   32353  5168
## 4 09/28/2016  3119    1075   212
## 5 09/29/2016     0       0     0
## 6 09/30/2016 23140    7345  2474

library(knitr)
kable(dailyCounts, format = 'latex', booktabs = TRUE)
```

date	Trump	Clinton	Obama
09/25/2016	113	38	26
09/26/2016	1812	735	123
09/27/2016	97267	32353	5168
09/28/2016	3119	1075	212
09/29/2016	0	0	0
09/30/2016	23140	7345	2474
10/01/2016	0	0	0
10/02/2016	18268	5756	1572
10/03/2016	20900	8278	2212
10/04/2016	21032	8809	3035
10/05/2016	12901	4028	769
10/06/2016	15991	6572	2684
10/07/2016	6452	2530	1110
10/08/2016	68775	15087	2778
10/09/2016	49886	13062	2389
10/10/2016	106596	33451	5154
10/11/2016	12846	3984	812
10/12/2016	29573	9891	2658
10/13/2016	43176	13395	5456
10/14/2016	38004	11329	6090
10/15/2016	33321	9832	4003
10/16/2016	30749	8826	2696
10/17/2016	27375	8369	2793
10/18/2016	31083	9181	4428
10/19/2016	2941	861	412

4.2 Including figures

SECTION INCOMPLETE - WORKING ON PLOTS.

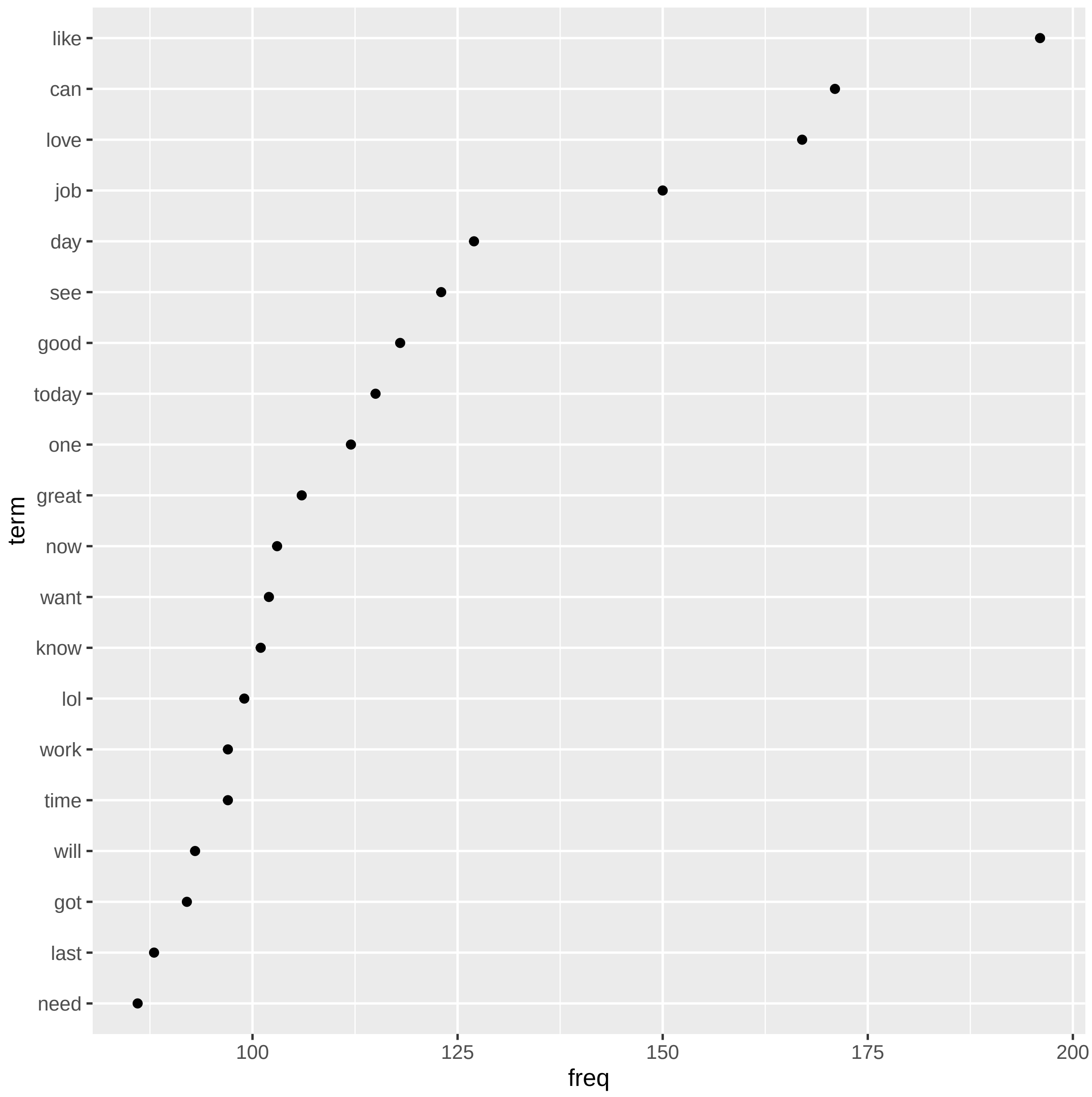
```
tail(allTermFreqs.tdm.rollup.df.sorted, 20)

##        term freq
## need   need   86
## last   last   88
## got     got   92
## will   will   93
## time   time   97
## work   work   97
## lol     lol   99
## know   know  101
## want   want  102
## now     now  103
## great great  106
## one     one  112
## today today  115
## good   good  118
## see     see  123
## day     day  127
## job     job  150
## love   love  167
## can     can  171
## like   like  196
```

```
library(ggplot2)
```

```
##
## Attaching package:  'ggplot2'
## The following object is masked from 'package:NLP':
##
##     annotate
```

```
highfreq <- tail(allTermFreqs.tdm.rollup.df.sorted, 20)
highfreq <- highfreq[order(-highfreq$freq),]
highfreq$term <- factor(highfreq$term, levels = highfreq$term[order(highfreq$freq)])
ggplot(highfreq, aes(freq, term)) +  geom_point()
```



=======

```
#library(maps)
#map("world", fill=TRUE, col="white", bg="lightblue", ylim=c(-60, 90), mar=c(0,0,0,0))
#points(x.cord,y.cord, col="red", pch=16)
```

5 Results

In result section you can start with an overview of what you have found during the exploration of data.

5.1 Including tables

SECTION INCOMPLETE - WORKING ON NICE LOOKING TABLES.

```
#dailyCounts <- read.csv("counts_wide.csv")
dailyCounts <- read.csv('https://github.com/lbtaylor/DataScienceGroup11/raw/master/counts_wide.csv')
head(dailyCounts)
```

```
##         date Trump Clinton Obama
## 1 09/25/2016   113      38    26
## 2 09/26/2016  1812     735   123
## 3 09/27/2016 97267   32353  5168
## 4 09/28/2016  3119    1075   212
## 5 09/29/2016     0       0     0
## 6 09/30/2016 23140    7345  2474
```

```
library(knitr)
kable(dailyCounts, format = 'latex', booktabs = TRUE)
```

date	Trump	Clinton	Obama
09/25/2016	113	38	26
09/26/2016	1812	735	123
09/27/2016	97267	32353	5168
09/28/2016	3119	1075	212
09/29/2016	0	0	0
09/30/2016	23140	7345	2474
10/01/2016	0	0	0
10/02/2016	18268	5756	1572
10/03/2016	20900	8278	2212
10/04/2016	21032	8809	3035
10/05/2016	12901	4028	769
10/06/2016	15991	6572	2684
10/07/2016	6452	2530	1110
10/08/2016	68775	15087	2778
10/09/2016	49886	13062	2389
10/10/2016	106596	33451	5154
10/11/2016	12846	3984	812
10/12/2016	29573	9891	2658
10/13/2016	43176	13395	5456
10/14/2016	38004	11329	6090
10/15/2016	33321	9832	4003
10/16/2016	30749	8826	2696
10/17/2016	27375	8369	2793
10/18/2016	31083	9181	4428
10/19/2016	2941	861	412

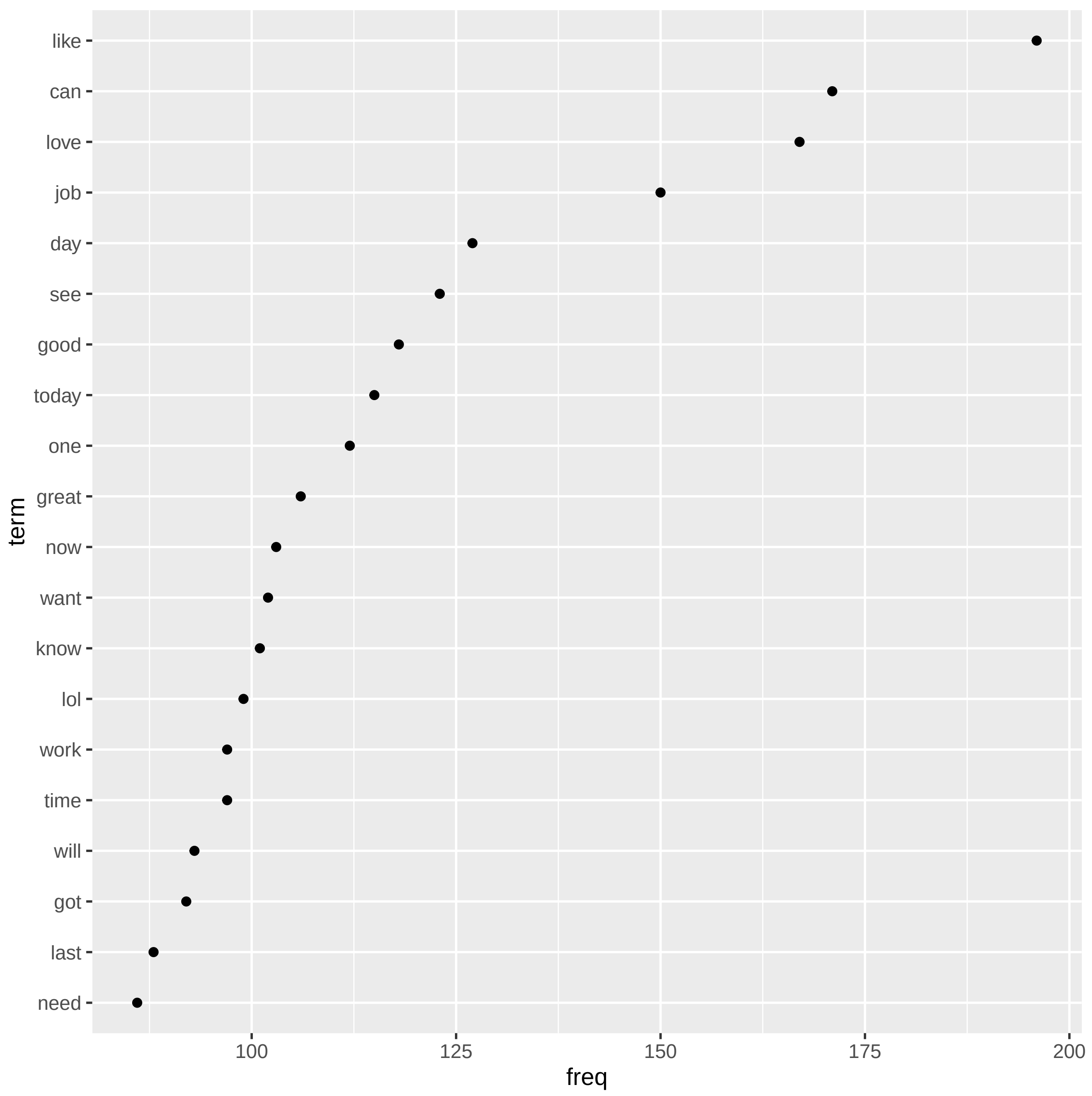
5.2 Including figures

SECTION INCOMPLETE - WORKING ON PLOTS.
¡¡¡¡¡¡¡ HEAD ¿¿¿¿¿¿¿ master =======

```
tail(allTermFreqs.tdm.rollup.df.sorted, 20)
```

```
##        term freq
## need   need   86
## last   last   88
## got     got   92
## will   will   93
## time   time   97
## work   work   97
## lol     lol   99
## know   know  101
## want   want  102
## now     now  103
## great great  106
## one     one  112
## today today  115
## good   good  118
## see     see  123
## day     day  127
## job     job  150
## love   love  167
## can     can  171
## like   like  196
```

```
library(ggplot2)
highfreq <- tail(allTermFreqs.tdm.rollup.df.sorted, 20)
highfreq <- highfreq[order(-highfreq$freq),]
highfreq$term <- factor(highfreq$term, levels = highfreq$term[order(highfreq$freq)])
ggplot(highfreq, aes(freq, term)) +  geom_point()
```



¿¿¿¿¿¿¿ dennis-friday

6 Conclusion

Initial trends that we have seen some evidence of Trump dominates twitter. Table of daily tweets mentioning Trump total more than three times tweets mentioning Clinton. Trumps count is approximately twelve times greater than Obamas count. With the negativity of twitter, domination of negativity is not necessarily a good thing.

Obama seems to be viewed more positively than either candidate. We have had mixed results with word clouds, in terms of finding things that are interesting. We will also try dot plots. Words associated with women is one of the more interesting word associations in this election. Tremendously negative for Trump, negative in a more normal way for Clinton, and mostly positive for Obama. When we find things that are interesting, we hope to also show how those aspects changed over the course of the roughly six weeks of data that we will have.

We have had the hardest time working on maps. Our hope is to show some data by state, to see if important geographical differences in the election are apparent in twitter.

7 Appendix

The SQL Server table schema

```
USE [TwitterData]
GO

/****** Object:  Table [dbo].[NewFlattenedTweets]    Script Date: 10/19/2016 10:16:05 PM ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[NewFlattenedTweets](
[MessageId] [bigint] NOT NULL,
[CreatedAt] [datetime] NOT NULL,
[UserId] [bigint] NULL,
[PlaceId] [nvarchar](350) NULL,
[PlaceType] [nvarchar](350) NULL,
[PlaceName] [nvarchar](350) NULL,
[PlaceFullName] [nvarchar](350) NULL,
[PlaceCountryCode] [nvarchar](10) NULL,
[PlaceCountry] [nvarchar](350) NULL,
[PlaceBoundingBox] [nvarchar](350) NULL,
[CoordinatesType] [nvarchar](350) NULL,
[CoordinatesCoordinates] [nvarchar](350) NULL,
[Text] [nvarchar](max) NULL,
 CONSTRAINT [PK_NewFlattenedTweets] PRIMARY KEY CLUSTERED
(
[MessageId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
    IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
    ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
```

A sample of inserting data from twitter json source file into table.

```
create table #tempTweets(Json nvarchar(max))

BULK INSERT #tempTweets --RawTweetJson
FROM 'C:\Share\Tweets_20161013_clean.json'

select count(*) from #tempTweets
select top 10 * from #tempTweets
```

```
  --SELECT MAX(LEN(JSON_VALUE([Json], '$.text'))) [Text]
  --FROM #tempTweets TT


INSERT INTO NewFlattenedTweets
(     [MessageId]
    ,[CreatedAt]
      ,[UserId]--18
    ,[PlaceId]
      ,[PlaceType]
      ,[PlaceName]
      ,[PlaceFullName]
      ,[PlaceCountryCode]
      ,[PlaceCountry]
      ,[PlaceBoundingBox]
      ,[CoordinatesType]
      ,[CoordinatesCoordinates]
      ,[Text])--545
SELECT DISTINCT
       CAST(JSON_VALUE([Json], '$.id') AS BIGINT) MessageId
  --JSON_VALUE([Json], '$.created_at') CreatedAt
  ,CONVERT(DATETIME,SUBSTRING(JSON_VALUE([Json], '$.created_at'),4,7) +
SUBSTRING(JSON_VALUE([Json], '$.created_at'),26,5) +
SUBSTRING(JSON_VALUE([Json], '$.created_at'),11,9))
CreatedAt

  ,CAST(JSON_VALUE([Json], '$.user.id') AS BIGINT) UserId
  ,CAST(JSON_VALUE([Json], '$.place.id') AS NVARCHAR(350)) PlaceId
  ,CAST(JSON_VALUE([Json], '$.place.place_type') AS NVARCHAR(350)) PlaceType
  ,CAST(JSON_VALUE([Json], '$.place.name') AS NVARCHAR(350)) PlaceName
  ,CAST(JSON_VALUE([Json], '$.place.full_name') AS NVARCHAR(350)) PlaceFullName
  ,CAST(JSON_VALUE([Json], '$.place.country_code') AS NVARCHAR(10)) PlaceCountryCode
  ,CAST(JSON_VALUE([Json], '$.place.country') AS NVARCHAR(350)) PlaceCountry
  ,CAST(JSON_QUERY([Json], '$.place.bounding_box') AS NVARCHAR(350)) PlaceBoundingBox
  ,CAST(JSON_VALUE([Json], '$.coordinates.type') AS NVARCHAR(350)) CoordinatesType
  ,CAST(JSON_QUERY([Json], '$.coordinates.coordinates') AS NVARCHAR(350)) CoordinatesCoords
  ,CAST(JSON_VALUE([Json], '$.text') AS NVARCHAR(140)) [Text]
  FROM #tempTweets TT
  WHERE JSON_VALUE([Json], '$.created_at') IS NOT NULL
  AND CAST(JSON_VALUE([Json], '$.id') AS BIGINT) NOT IN (
  SELECT MessageId
  FROM [TwitterData].[dbo].[NewFlattenedTweets]
  )

drop table #tempTweets
```

References

[1] Desilver, Drew. ”5 facts about Twitter at age 10”. *Pew Research*, urlhttp://www.pewresearch.org/fact-tank/2016/03/18/5-facts-about-twitter-at-age-10/ . 2016

[2] Gayo-Avello, Daniel. ”Don’t Turn Social Media into Another Literary Digest Poll.” *Communications of The ACM* 54.10 (2011): 121-128. *Business Source Elite.*

[3] Giachanou, Anastasia, and Fabio Crestani. ”Like It or Not: A Survey of Twitter Sentiment Analysis Methods.” *ACM Computing Surveys* 49.2 (2016): 28-28:41. *Business Source Elite*.