Language Trend Analysis Across Twitter Relating to the 2016 United States Presidential Election

Luke Taylor
Dennis Walsh
Thomas Flaherty

October 20, 2016

Abstract

This first sentence of the abstract starts with a little background of the problem. The second sentence starts describing the problem with some motivation. If needed we add another sentence to increase the reader's interest and clarity. After the problem description is over, we need to describe what we have done, how we did it and what results we obtained. We include any challenges we face and solution of that challenges we made. We describe any experiment we did with their findings. We also clearly mention how our results or methods are better than any existing results or methods. Sometimes we like to mention how this work opens up some future works. Abstract should not be more than a paragraph preferably having less than 200 words.

I left in above description for now. Think about whether we need to update the abstract. I made a few minor updates already - adding presidential reference, and removing statement regarding real-time data.

Language is always evolving. Words can move in and out of use across time and between regions, carrying the topics of the day with them. Over the last few decades, the way we communicate with each other has jumped from medium to medium, seemingly with no end in sight. 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 attempt to identify different trends in the 2016 US Presidential election. These trends in turn will become interactive data products. By analyzing Twitter data, insights into emerging trends can be seen by the use of identifying keywords and their associated words. Because Twitter data has the ability to attach geographic information to the Tweets (Twitter messages), trends can be visualized on a map or over time.

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. To start our work conveniently we need to install **R**, `RStudio` and LaTeX [**lamport94**] . Once our installation is done we will configure `RStudio` to work with `knitr`. For this first install `knitr` using command `install.packages("knitr")` and include the `knitr` library by command `library(knitr)`. Once `knitr` is installed go to the `RStudio` menu `Tools > Global Options...Sweave` and change 'Weave Rnw files using' to indicate `Knitr`.

Now we are ready to create our first document using `knitr`. Go to `File > New File > R Sweave` and it will start with a new template for a document. If you save this minimal template it will be saved as a `.Rnw` file. Now we can just start filling the template with our texts. To create a human readable pdf file from `.Rnw` we just click on `Compile PDF` in `RStudio` toolbar.

PDF latex failure: If you encounter any problem such as `Running pdflatex on ...failed` it could be due to the bibliography. To solve that problem what you can do is: Go back to the folder where you saved your `.Rnw` file and find the `.tex` file that is created automatically. Now run the `.tex` file from LaTeX editor to create the pdf. Once you do this multiple times your bibliographies would be updated and you will be ready to work from `RStudio` as long as you don't change any object that has references in the file. There may be a better solution for this, but so far this worked for me.

The solution for this problem: just add `\usepackage[backend=bibtex]biblatex` in your preamble of the `.Rnw` file.

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 continous 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. (cite)

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 comed 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 100 data points and 12 variables in this data set. The variables are X, Id, InsertDateTime, CreatedAt, MessageId, UserId, PlaceId, PlaceType, PlaceName, CoordinatesType, CoordinatesCoords, 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 () {
  library(RODBC)
  dbhandle <- odbcConnect("ODBCName",
                          uid="userId",
                          pwd="passWord")

  res <- sqlQuery(dbhandle, "
     SELECT TOP (100) [Id]
    --  SELECT [Id]
      ,[InsertDateTime]
      ,[CreatedAt]
      ,[MessageId]
      ,[UserId]
      ,[PlaceId]
      ,[PlaceType]
      ,[PlaceName]
      ,[CoordinatesType]
      ,[CoordinatesCoords]
      ,[Text]
      FROM [TwitterData].[dbo].[FlattenedTweets]")
  odbcClose(dbhandle)
  return (res);
}

dataSet <- GetDataSet();
```

2.1 Preparing data

The most salient value of this data is with the comination 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 anyalysis being performed. In order to encapsulate the process, it is wrapped in a function and passed a character vector consisting of the tweet text.

```
TwitterToCorpus <- function(x) {
    # Load the NLP library
    library(tm)

#    corpusData <- as.character(x)
    corpusData <- as.character(dataSet$Text)

    #remove non-english characters
    ##enCorpusData <- iconv(corpusData, "UTF-8", "ASCII", sub="")

    #head(enCorpusData)

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

    # 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)

    # Convert special characters to spaces in the corpus
    #toSpace <- content_transformer(function (x , pattern ) gsub(pattern, " ", x))
    #myCorpus <- tm_map(myCorpus, toSpace, "/")
    #myCorpus <- tm_map(myCorpus, toSpace, "@")
    #myCorpus <- tm_map(myCorpus, toSpace, "\\|")

    # 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)

    # Remove all terms that contain the search term
    removeNonLowerAlpha <- content_transformer(function(x) gsub("[^a-z]"," ",x))
    myCorpus <- tm_map(myCorpus, removeNonLowerAlpha)

    # Remove all non a-z chars
    removeSearchTermWild <-
      content_transformer(function(x) gsub("\\b\\S*trump\\S*\\b"," ",x))
```

```
    myCorpus <- tm_map(myCorpus, removeSearchTermWild)

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

    findFreqTerms(TermDocumentMatrix(myCorpus))
   return (myCorpus)
}

myCorpus <- TwitterToCorpus(dataSet$Text)

## Loading required package:  NLP

#inspect(myCorpus)
```

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(myCorpus)
#dtm <- DocumentTermMatrix(myCorpus)
tdm <- TermDocumentMatrix(myCorpusCopy)
dtm <- DocumentTermMatrix(myCorpusCopy)

head(findFreqTerms(dtm))

## [1] "account" "achar"   "action"  "adele"   "agr"     "agua"

# 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
## account account    1
## achar     achar    1
## adele     adele    1
## agr         agr    1
## agua       agua    1
## ahi         ahi    1

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

##      term freq
## work work    2
## hoy   hoy    3
## job   job    3
## one   one    3
## uma   uma    3
## bad   bad    4

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

```
summary(dataSet)

##        X                Id                      InsertDateTime
##  Min.   :  1.00   Min.   :159433   2016-09-27 13:05:45:100
##  1st Qu.: 25.75   1st Qu.:159458
##  Median : 50.50   Median :159482
##  Mean   : 50.50   Mean   :159482
##  3rd Qu.: 75.25   3rd Qu.:159507
##  Max.   :100.00   Max.   :159532
##
```

```
##                           CreatedAt    MessageId
##  Mon Sep 26 23:47:49 +0000 2016:21   Min.   :7.806e+17
##  Mon Sep 26 23:47:47 +0000 2016:16   1st Qu.:7.806e+17
##  Mon Sep 26 23:47:50 +0000 2016:15   Median :7.806e+17
##  Mon Sep 26 23:47:46 +0000 2016:11   Mean   :7.806e+17
##  Mon Sep 26 23:47:43 +0000 2016: 9   3rd Qu.:7.806e+17
##  (Other)                       :21   Max.   :7.806e+17
##  NA's                          : 7   NA's   :7
##      UserId                     PlaceId     PlaceType
##  Min.   :5.870e+06   97bcdfca1a2dca59: 3   admin  :10
##  1st Qu.:2.888e+08   0161be1b3f98d6c3: 2   city   :79
##  Median :9.408e+08   0177894212b08f73: 2   country: 4
##  Mean   :3.093e+16   1c69a67ad480e1b1: 2   NA's   : 7
##  3rd Qu.:2.802e+09   300bcc6e23a88361: 2
##  Max.   :7.422e+17   (Other)         :82
##  NA's   :7           NA's            : 7
##           PlaceName  CoordinatesType                  CoordinatesCoords
##  Rio de Janeiro: 3   Point:13        [-103.385907,20.581306]   : 1
##  Bogot, D.C.  : 2   NA's :87        [-104.8771726,39.5807452] : 1
##  Houston       : 2                   [-117.0143058,32.6249798] : 1
##  La Plata      : 2                   [-122.33,47.61]           : 1
##  So Paulo     : 2                   [-34.92318972,-7.15089739]: 1
##  (Other)       :82                   (Other)                   : 8
##  NA's          : 7                   NA's                      :87
##                                                                                                                                  Te
##  A9 Come visit our workplace and you will see compassion in action! @ImpactMattersUs
##  Agr sim estou fdd pra dormir kkkk
##  A hoco date would be nice ??
##  A meu Deus do ceeeeeeu https://t.co/PCYf7NyH21                                                                                  :
##  A mi me gustan ls mujeres psychas que te dicen "si me entero que hablas con otra te lo pico cabron"
##  (Other)
##  NA's
```

Try to avoid putting raw output like this in your final report. Instead make a clean table as shown in table **??**. If you have to keep some raw output of your analysis please put them in a section called appendix at the end of the document. If you really believe that you have to put them here, you can do that and thats why we have this example here.

2.2 What is funny

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 Methods

This section will include the methods you are planning to use for your analysis. You should include some theoretical justification here. For example, why you think the method is applicable, what are the assumptions

about the methods, whether your data satisfies those assumption or not etc.

3.1 The model

3.2 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.

```
library(wordcloud)

## Loading required package:  RColorBrewer

library(tm)
library(SnowballC)
library(RODBC)
library(ggplot2)

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

library(plyr)
library(RColorBrewer)

# Sample set used for purpose of paper
dataSet <- read.csv('https://github.com/lbtaylor/DataScienceGroup11/raw/master/SampleTwitterData.csv')

corpusData <- as.character(dataSet)
enCorpusData <- iconv(corpusData, "UTF-8", "ASCII", sub="")

library(tm)

myCorpus <- Corpus(VectorSource(enCorpusData))
inspect(myCorpus)

## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 12
##
## [[1]]
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 5
##
## [[2]]
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 13
##
## [[3]]
## <<PlainTextDocument>>
## Metadata:  7
```

```
## Content:  chars: 301
##
## [[4]]
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 316
##
## [[5]]
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 1892
##
## [[6]]
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 1120
##
## [[7]]
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 392
##
## [[8]]
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 308
##
## [[9]]
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 392
##
## [[10]]
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 388
##
## [[11]]
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 392
##
## [[12]]
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 392
```

```
myCorpus <- tm_map(myCorpus, content_transformer(tolower))

removeURL <- content_transformer(function(x) gsub(" ?(f|ht)(tp)(s?)(://)(.*)[.|/](.*)","",x))
myCorpus <- tm_map(myCorpus, removeURL)

toSpace <- content_transformer(function (x , pattern ) gsub(pattern, " ", x))
myCorpus <- tm_map(myCorpus, toSpace, "/")
```

```
myCorpus <- tm_map(myCorpus, toSpace, "@")
myCorpus <- tm_map(myCorpus, toSpace, "\\|")

myCorpus <- tm_map(myCorpus, removeNumbers)

myExtraStopwords <- c("available","via","amp","get","com","pra","just")
myStopwords <- c(stopwords("english"), stopwords("spanish"),myExtraStopwords)
myCorpus <- tm_map(myCorpus, removeWords, myStopwords)

myCorpus <- tm_map(myCorpus, removePunctuation)

myCorpus <- tm_map(myCorpus, stripWhitespace)

myCorpusCopy <- myCorpus

myCorpus <- tm_map(myCorpus, stemDocument)

myCorpus <- tm_map(myCorpus, stripWhitespace)
myCorpus <- tm_map(myCorpus,PlainTextDocument)

tdm <- TermDocumentMatrix(myCorpus)
dtm <- DocumentTermMatrix(myCorpus)

idx <- which(dimnames(tdm)$Terms == "obama")

findFreqTerms(dtm,lowfreq = 100)

## character(0)

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.rollu
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,a
rm(allTermFreqs.tdm.rollup.df)

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

##   term freq
## 1  cna    2

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

##   term freq
## 1  cna    2
```

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

## $women
```

```
## numeric(0)

as.data.frame(fa)

## [1] women
## <0 rows> (or 0-length row.names)
```

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

Include some summary tables of the data as as shown in table **??**. Make sure you discuss about the table you have included and explain the facts it is revealing. You have to sell your table in a way that the reader will understand that this table was awesome and it reveals a fact the reader would otherwise not recognize.

Notice that we used the function `xtable()` form the **R** package `xtable` [**xtab**] to generate a pretty table. `knitr` does this using LaTeX codes generated by `xtable` and automatically put it in a nicer we and we don't have to worry about its position. Also notice how we write the caption of the table as well as refer the table **??** from the text.

```
# Creating and printing summary data table
library(xtable)

## Error in library(xtable):  there is no package called 'xtable'

summary_data <- apply(trees, 2, function(x) {
    return(c(Average = mean(x), Median = median(x), SD = sd(x), Range = range(x)))
})
print(xtable(summary_data, digits = 2, caption = paste("This table caption really",
    "describes what this table is about and what interesting facts it is revealing."),
    label = "summary-data"), caption.placement = getOption("xtable.caption.placement",
    "top"))

## Error in print(xtable(summary_data, digits = 2, caption = paste("This table caption really",
:  could not find function "xtable"
```

4.1.1 Book quality table

We can add tables that look like the tables in the book. For this we need to add package `booktabs` in the preamble of this .Rnw file. This will include a package called `booktabs` onto LaTeX. Once we add that we can now put option `booktabs = TRUE` in the **R** code as below.

```
library(knitr)
x <- head(mtcars)
kable(x,format = 'latex', booktabs = TRUE)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

4.2 Including figures

Please don't forget to add nice data plots in your documents. Plots are nice to conveying message and much better than tables. Discuss what facts the figure is revealing and refer the figure from the text as figure 1.

```
plot(trees)
```

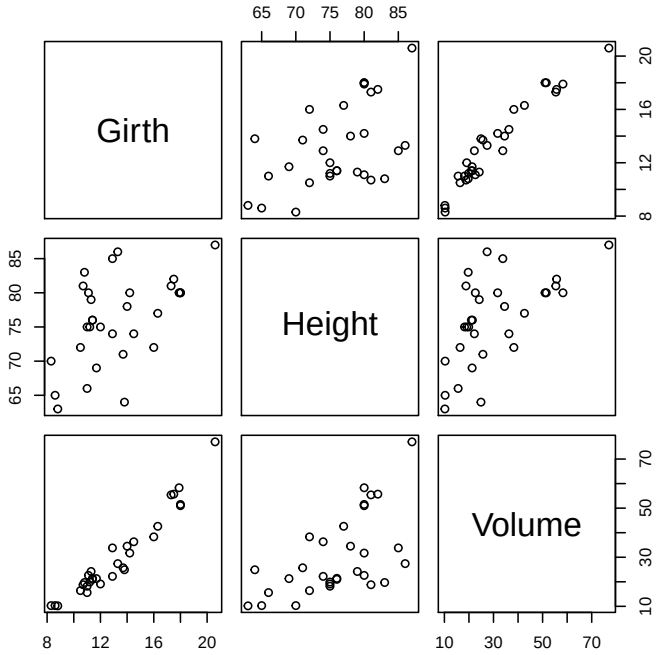

Figure 1: Awesome figure caption

4.3 How the data product works

If you build a data product you may discuss here how it works and what it provides. For data product being your main purpose, your main section may be different from just saying `Results`. You may think how you rename your sections to naturally fit in your work and the purpose.

5 Conclusion

The conclusion is an elaboration of your abstract. Here you will discuss what you have done and how. The gist of the results need to be mentioned here. It needs to be convincing and the reader will never regret

forgetting the date. Please keep it in mind that there may be readers who only read your conclusion. So, make your conclusion complete so that no reader misses anything even if they don't want to read the whole document.

Each paragraph of the conclusion may discuss one result you have found or one concept you are proposing. Discuss your findings and why it is better and how it is compared to any existing methods may exist.

Please don't forget to cite the works of others if you used it in your analysis. The citation is important for two reasons. Fist of all it acknowledges the good works other people have done which encourages them keep continue doing their good work. Second, it protects you from plagiarism which is a very nasty task everyone should avoid.

There should be one paragraph about the future direction of the work you have done. You would like to make it so fascinating that the reader would wish to be involved in this work in future.

Finally this is just a template. Your exact document may have a very different outlook. It demonstrates how you can start to write a document. Our biggest problem is to figure out where to start from. And this documents provides a guide for that. I hope it turns out to be helpful for some of the readers. If you have any comments or concern about this document please let me know so that I can improve this document.

6 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*, http://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*.

[4] Kodali, Teja. "Building Wordclouds in R". 2015. https://www.r-bloggers.com/building-wordclouds-in-r/

[5] Mitchell, Amy and Paul Hitlin. "Events Often at Odds with Overall Public Opinion". *Pew Research*. 2013. HTTP://WWW.PEWRESEARCH.ORG/2013/03/04/TWITTER-REACTION-TO-EVENTS-OFTEN-AT-ODDS-WITH-OVERALL-PUBLIC-OPINION/

[6] http://www.sthda.com/english/wiki/ text-mining-and-word-cloud-fundamentals-in-r-5-simple-steps-you-should-know

[7] http://www.slideshare.net/rdatamining/ text-mining-with-r-an-analysis-of-twitter-data

[8] Remove non-english words http://stackoverflow.com/questions/18153504/ removing-non-english-text-from-corpus-in-r-using-tm

[9] `hi`

[10] emphConverting lat and long to state value http://stackoverflow.com/questions/8751497/ latitude-longitude-coordinates-to-state-code-in-r