Simple Sentiment Analysis for Twitter

Using 16,000 tweets directed at or written by the top four 2016 presidential candidates post-Iowa Caucus: Hillary Clinton, Ted Cruz, Bernie Sanders, and Donald Trump, we will create plots to analyze the sentiment of Twitter users related to the four candidates.

Global parameters

Set working directory by pointing to the location on your computer where you have stored the files. Below, we have chosen to Save the folder "RAnalysis" on the Desktop on a Mac. It contains all the other R scripts, texts, notebooks, and results. If you have branched the Github, simply note where you have save the folder. If you are on a PC, you will need to use an absolute path such as "C:Users:XXX."

Hint: Can't figure out your syntax? Click Session > Set Working Directory > Choose Directory, then select the Text_Analysis directory in which you are working. This will set your working directory in the console below while you are working here, but make sure to copy the path into the "setwd" command below to keep the directory constant if you close this script and reopen later. ***

```
setwd("~/Desktop/R/Text_Analysis/")

#Source Multiplot Function
source("RScripts/multiplot.R")
```

Include necessary packages for notebook

R's extensibility comes in large part from packages. Packages are groups of functions, data, and algorithms that allow users to easily carry out processes without recreating the wheel. Some packages are included in the basic installation of R, others created by R users are available for download. Make sure to have the following packages installed before beginning so that they can be accessed while running the scripts. By calling library(packagename) they will be automatically loaded and ready to use.

The first four packages are used to render the RNotebook you are currently viewing:

knitr - Creates a formatted report from the script provided

formatR - Works with knitr to keep the R code tidy and more readable. Here we use it with the global opts_chunk$set(tidy.opts=list(width.cutoff=50), tidy=TRUE) following the knitr library to keep the code lines from running off the pdf page.

markdown - a package used to render textual documents from plain text to others such as R and XML

rmarkdown - similar to markdown but specifically to render R documents

The next four packages are used within the sentiment analysis script:

twitteR - this is a fun little package that allows the user to search twitter for tweets containing certain words/phrases or by certain people, check out this post for information on setting up an API key to access tweets

plyr - this package splits, combines, and applies data (the laply function used below while instantiating the algorithm comes with this package)

stringr - eases string operations (we will use str_split also while instantiating the algorithm, which is included with the stringr package).

ggplot2 - a great package that renders a range of plots to display information, we will use this package to create our ultimate plots

```
library(knitr)
opts_chunk$set(tidy.opts=list(width.cutoff=50), tidy=TRUE)
library(formatR)
library(markdown)
library(rmarkdown)
library(twitteR)
library(plyr)
```

```
##
## Attaching package: 'plyr'

## The following object is masked from 'package:twitteR':
##
##     id
```

```
library(stringr)
library(ggplot2)
```

Load data

This each .RData object is a curated set of tweets grabbed from Twitter using the TwitteR package. The tweets orginially return as a list, which was then saved as the .RData object you see here. We load the .RData files into our environment using readRDS and save them into a variable we can use later.

Hint 1: The names of the variables can be anything (happiness, rainbows, cats), but usually, you want to name them using terms to indicate what the variable holds. Here we use clinton.tweets to represent the tweets associated with @HillaryClinton on Twitter, and so on for the others. The period (.) between "clinton" and "tweets" does not serve any purpose other than to separate the two terms for ease of reading.

Hint 2: Another popular naming convention is to list the type of object with the name, for example clinton.tweets.l since this object is a list. This convention helps when manipulating the data later.

```
clinton.tweets <- readRDS("~/Desktop/R/Text_Analysis/data/twitter/tweetsclinton.RData")
cruz.tweets <- readRDS("~/Desktop/R/Text_Analysis/data/twitter/tweetssanders.RData")
sanders.tweets <- readRDS("~/Desktop/R/Text_Analysis/data/twitter/tweetstrump.RData")
trump.tweets <- readRDS("~/Desktop/R/Text_Analysis/data/twitter/tweetscruz.RData")
```

Inspect Data

Before preparing the data for analysis, let us first see what we are working with. For tutorial purposes, we will look at the first: clinton.tweets, but feel free to use the same commands to inspect any of the variables we have created. This step is a good idea to ensure you know what you are ultimately analyzing. For example, if you find that the length is zero, you may have to go back and reload or re-grab the data.

Class describes the type of object. In this case, we grabbed tweets using the TwitteR package, which results in a list:

```
class(clinton.tweets)
```

```
## [1] "list"
```

Length shows how many elements the object has. We grabbed 4,000 tweets (specified while using the TwitteR package), so our list is made up of 4,000 elements. An element here is a tweet!

```
length(clinton.tweets)
```

```
## [1] 4000
```

We can also look at individual elements - Lets see what the first (most recent) tweet is...

```
clinton.tweets[1]
```

```
## [[1]]
## [1] "NOControl07: Coin flip \"wins\" by @HillaryClinton doesn't change how happy I am for @BernieSan
```

Or perhaps you would like to see the first few elements:

```
head(clinton.tweets)
```

```
## [[1]]
## [1] "NOControl07: Coin flip \"wins\" by @HillaryClinton doesn't change how happy I am for @BernieSan
##
## [[2]]
## [1] "ItsAlinajo: @JAPITTER ROFLMAAAAO!! Thanks Jeff! Coins for donors! :-) @DJROBBZ_BLOG @Teddyshous
##
## [[3]]
## [1] "terryshumaker: @billclinton in Laconia, NH: Progressive reformers in our history like @HillaryC
##
## [[4]]
```

```
## [1] "watizy: @CNN @HillaryClinton it looks more like fraud, winning six coin toss. Yeah right."
##
## [[5]]
## [1] "Eric_Bailon: RT @RuFreeman: Perspective for the Dems on #IowaCaucusresults  :  ) #Bernie2016 @H
##
## [[6]]
## [1] "OhCarouu: RT @HillaryClinton: Text CONGRATS to 47246 to tell Hillary you'll be by her side for
```

There are many more ways to inspect parts of data (check out the CRAN), but these quick checks are helpful while manipulating and trimming the data pre-analysis.

Extract Text

Since we grabbed our data from Twitter, there is metadata included in the tweet list such as retweet, favorite, and reply information. For sentiment analysis, we just need the text. The following line extracts the text from each tweet in the list and saves the new list of just text as [candidate_name].text.

```
clinton.text = laply(clinton.tweets, function(t) t$getText())
cruz.text = laply(cruz.tweets, function(t) t$getText())
sanders.text = laply(sanders.tweets, function(t) t$getText())
trump.text = laply(trump.tweets, function(t) t$getText())
```

Now we are all set! Let's load in some postivity (and negativity, I guess...)!

Loading the Opinion Lexicons to Determine Sentiment

This is an essential step for sentiment analysis. These text documents from Hu and Liu, 2004* are filled with positive and negative words, respectively. The algorithm we will write next will check these documents to score each word in the tweet. If the algorithm runs across the word "love" in a tweet, it will check the positive-words.txt file, find "love" is included, and score the word with a +1. More on that in a second...

```
lex.pos = scan("~/Desktop/R/Text_Analysis/data/opinionLexicon/positive-words.txt",
    what = "character", comment.char = ";")
lex.neg = scan("~/Desktop/R/Text_Analysis/data/opinionLexicon/negative-words.txt",
    what = "character", comment.char = ";")
```

Add words relevant to our corpus using the combine c() function:

```
pos.words = c(lex.pos, "win", "prove", "beat", "endorse",
    "endorses", "exciting", "vote", "wins", "support",
    "supports", "help", "winner")
neg.words = c(lex.neg, "lose", "losing", "defeat",
    "halt")
```

Implement the sentiment scoring algorithm

Here is where the magic happens - let's create the algorithm! To begin a function which will iterate over all of the elements in the specified object, we need to give it a name - score.sentiment sounds good. First, we set the name of our function equal to function(). Then, we fill the parenthesis with our arguments. Here, we want an argument for the tweets (our data we gathered), our positive words, and our negative words.

This function takes in an argument (tweets for us), normalizes all of the text (including removing emojis which is important for tweets), splits the tweets into separate words to analyze, compares the tweet words to the positive and negative dictionaries, returns a TRUE if the word is in the dictionary and FALSE if it is not, the trues and falses are summed up for each tweet (this is the score of the tweet), and finally, the tweet's score and its text are returned as a data frame for ease of plotting.

Hint: You may notice some odd expressions in the first few lines of the scores function: they are called regular expressions (or, regex). Regular expressions allow the user to match on certain patterns throughout the entire text. In this case, we are matching on punctuation, control characters (or invisible, non-printing characters), and digits, in the effort to strip out unnecessary characters that do not provide sentiment valence. Check out this CRAN page for more information about using regex in R.

```
score.sentiment = function(tweets, pos.words, neg.words,
    .progress = "none") {

    # figure out the score for each tweet specifically
    scores = laply(tweets, function(tweet, pos.words,
        neg.words) {
        # normalize tweet text
        tweet = gsub("[[:punct:]]", "", tweet)
        tweet = gsub("[[:cntrl:]]", "", tweet)
        tweet = gsub("\\d+", "", tweet)

        # REMOVE EMOJIS!
        tweet = iconv(tweet, "ASCII", "UTF-8", sub = "")

        tweet.lower = tolower(tweet)

        # split list into separate words
        word.list = str_split(tweet.lower, "\\s+")
        words = unlist(word.list)

        # compare our words to the dictionaries of positive
        # and negative terms using match function
        pos.matches = match(words, pos.words)
        neg.matches = match(words, neg.words)

        # match returns a position of the matched term or
        # NA, but we just want the TRUE/FALSE, not NA
        pos.matches = !is.na(pos.matches)
        neg.matches = !is.na(neg.matches)
```

```
        # the score of each tweet is the sum of the
        # positive matches minus the sum of the negative
        # matches
        score = sum(pos.matches) - sum(neg.matches)

        return(score)
    }, pos.words, neg.words, .progress = .progress)

    # compile the scores and text of tweets into a data
    # frame for plotting
    scores.df = data.frame(score = scores, text = tweets)
    return(scores.df)
}
```

Algorithm Testing

Whenever you create a function (or algorithm), it is best to test it on some sample data to ensure it behaves as you expect. Let's create a sample list of emotional sentences and save it as "sample".

```
sample = c("This ice cream is the best! I love this flavor!",
    "I am so angry at the terrible weather today. Frustrated.",
    "Wow, spectacular, I wish I could be as perfect as you.")
```

We already did the hard part by building our score.sentiment function earlier. Here, we just need to tell the algorithm what to use as arguments! We need to put our sample data in as the "tweets" argument, pos.words as the "pos.words"" argument, and neg.words as the "neg.words" argument!

Lets also save the result of running the algorithm as an object called sample.result.

```
sample.result = score.sentiment(sample, pos.words,
    neg.words)
```

Did it work?

```
sample.result
```

```
##   score                                                text
## 1     2          This ice cream is the best! I love this flavor!
## 2    -3 I am so angry at the terrible weather today. Frustrated.
## 3     3   Wow, spectacular, I wish I could be as perfect as you.
```

It worked! Now we can be sure our algorithm is behaving as we would expect. Lets score our Twitter data since we are confident in our function-creating abilities...

Scoring Twitter Data

Again, we have already created our function, we just need to tell it what to analyze. We will use our [candidate].text objects as the "tweets" argument and save each as an object called [candidate].result to plot a little later.

```
clinton.result = score.sentiment(clinton.text, pos.words,
    neg.words)
cruz.result = score.sentiment(cruz.text, pos.words,
    neg.words)
sanders.result = score.sentiment(sanders.text, pos.words,
    neg.words)
trump.result = score.sentiment(trump.text, pos.words,
    neg.words)
```

Lets peek at the results to see if it is still working...

```
head(clinton.result)
```

```
##   score
## 1     2
## 2     0
## 3     2
## 4     2
## 5     0
## 6     0
##
## 1                       Coin flip "wins" by @HillaryClinton doesn't change how happy I am for @Bernie
## 2  @JAPITTER ROFLMAAAAO!! Thanks Jeff! Coins for donors! :-) @DJROBBZ_BLOG @Teddyshouse2II @richdoll
## 3           @billclinton in Laconia, NH: Progressive reformers in our history like @HillaryClinton g
## 4                                                          @CNN @HillaryClinton it looks more like
## 5 RT @RuFreeman: Perspective for the Dems on #IowaCaucusresults  :  ) #Bernie2016 @HillaryClinton @S
## 6 RT @HillaryClinton: Text CONGRATS to 47246 to tell Hillary you'll be by her side for the next part
```

```
head(cruz.result)
```

```
##   score
## 1     2
## 2     0
## 3     0
## 4     1
## 5     1
## 6     0
##
## 1                         Coin flip "wins" by @HillaryClinton doesn't change how happy I am for @Bern
## 2                                   This Revolution Will Be Televised!  @SusanSarandon @BernieSande
## 3
## 4   RT @BernieSanders: What last night proved is that every single contribution matters and every vo
## 5   RT @BernieSanders: The American people are ready to stand up and fight for fundamental change on
## 6 RT @BernieSanders: Blitzer: "The debate in New York City that they want. Are you accepting that?"\
```

```
head(sanders.result)
```

```
##   score
## 1     0
## 2     2
## 3     0
## 4     2
## 5    -1
## 6     0
##
## 1             RT @foxandfriends: "@realDonaldTrump is actually the only person bothering to talk to
## 2                          2016 Nobel Peace Prize Nominees Include @realDonaldTrump?!? https://t.co/
## 3                                                                                WATCH LIVE: @Reall
## 4                 RT @realDonaldTrump: The media has not covered my long-shot great finish in Iowa
## 5                         Ron Fournier's name calling and veiled slaps at @realDonaldTrump hurts the
## 6 RT @politicalpolls5: Let's not stop now. We need to keep the #TrumpTrain going. @realDonaldTrump @
```

```
head(trump.result)
```

```
##   score
## 1    -1
## 2     0
## 3     1
## 4    -3
## 5    -1
## 6     1
##
## 1   RT @oreillyfactor: The Carson campaign says @tedcruz played dirty tricks in Iowa. @RealBenCarson
## 2   RT @steve0423: Mainstream media lost it when @tedcruz gave glory to God, yet if he were yelling a
## 3
## 4 Rubio is likable\nuntil his mask comes off\nand he's hurling rapid-fire untrue accusations at @ted
## 5                 @GarryRSmith and @RepWendyNanney Rep Burns and @kevinbryantsc kicking off the @ted
## 6  RT @FoxBusiness: .@tedcruz wins Republican #IowaCaucus, tells supporters "our rights come from ou
```

Plotting Twitter Data

To visualize the sentiment of these particular tweets, we will use the plotting system ggplot2 and its quick plot (qplot) functionality. The qplot function takes an object, which is the score from our tweets, then you can specify other parts of the plot. For example, we have titled our plots "Sentiment of [candidate's twitter handle] on Twitter," labeled the x-axis "Valence of Sentiment (Tweet Score)," and labeled the y-axis "Count (Tweets)." These were all saved as [candidate].plot objects to use later.

Below, we impose the "Black and White" theme on the plots

```r
# Plotting Twitter Data
clinton.plot = qplot(clinton.result$score, xlim = (c(-7,
    7)), main = "Sentiment of @HillaryClinton on Twitter",
    xlab = "Valence of Sentiment (Tweet Score)", ylab = "Count (Tweets)")
cruz.plot = qplot(cruz.result$score, xlim = (c(-7,
    7)), main = "Sentiment of @tedcruz on Twitter",
    xlab = "Valence of Sentiment (Tweet Score)", ylab = "Count (Tweets)")
sanders.plot = qplot(sanders.result$score, xlim = (c(-7,
    7)), main = "Sentiment of @BernieSanders on Twitter",
    xlab = "Valence of Sentiment (Tweet Score)", ylab = "Count (Tweets)")
trump.plot = qplot(trump.result$score, xlim = (c(-7,
    7)), main = "Sentiment of @realDonaldTrump on Twitter",
    xlab = "Valence of Sentiment (Tweet Score)", ylab = "Count (Tweets)")


multiplot(clinton.plot, sanders.plot, trump.plot, cruz.plot,
    cols = 2)
```

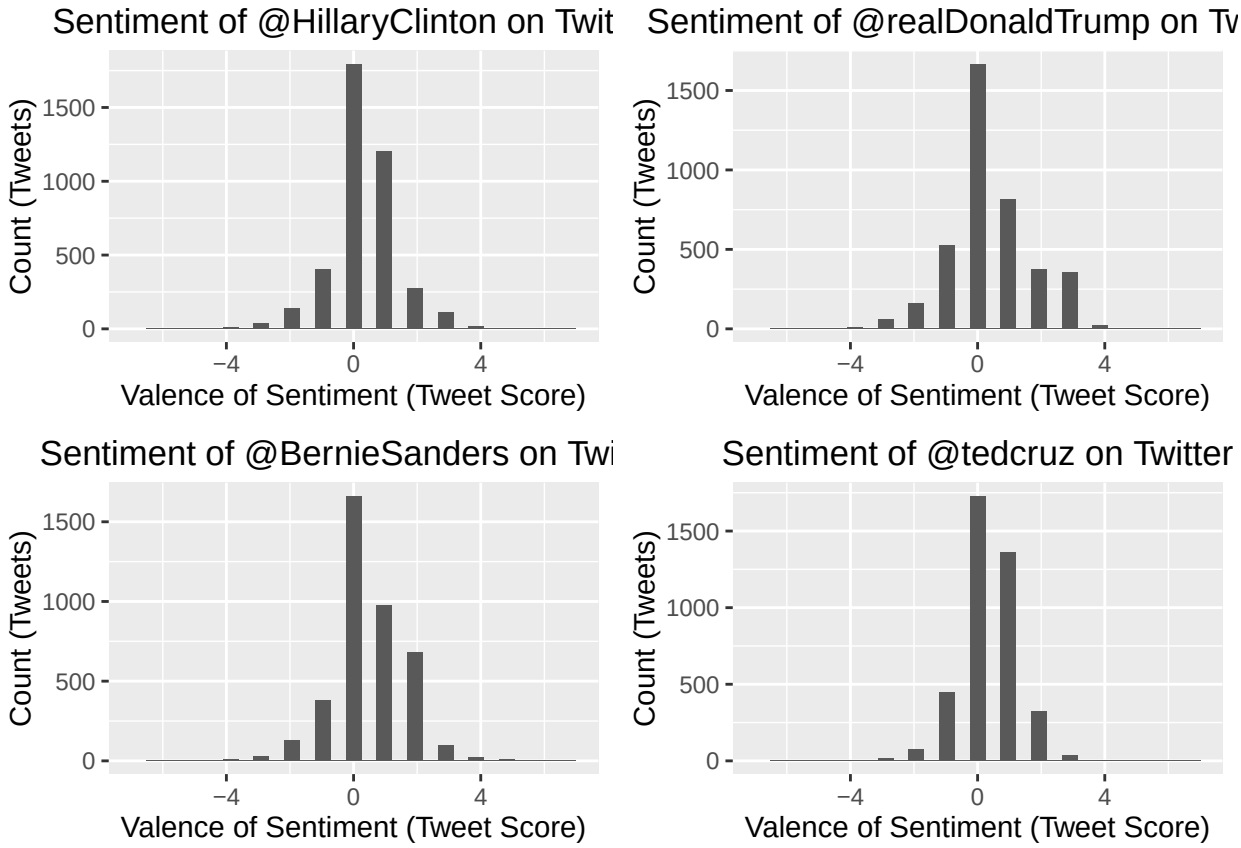
```
## Loading required package: grid
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 1 rows containing non-finite values (stat_bin).
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 1 rows containing non-finite values (stat_bin).
```



Voila!

Acknowledgements: This algorithm was adapted from Jeffrey Breen's Mining Twitter for Airline Consumer Sentiment article. You can find it here: http://www.inside-r.org/howto/mining-twitter-airline-consumer-sentiment.

Reference: Liu, Minqing Hu and Junsheng Cheng. "Opinion Observer: Analyzing and Comparing Opinions on the Web." Proceedings of the 14th International World Wide Web conference (WWW-2005), May 10-14, 2005, Chiba, Japan.