Scraping Data from Twitter and Data Visualization

Brian Hsu
November 26, 2017

Introduction

Motivation

The motivation behind this post was to delve deeper into R, beyond what was laid out by the framework of the course. Hopefully, this exploration into both R and actual, real life data will be both informative and entertaining. The final portion, involving the creation of a word cloud, produces a result akin to something that might be seen on social media.

The purpose of this post is to gain some experience using other packages beyond the ones introduced in the Stat 133 course to produce some hopefully interesting results. After scraping the data using the rtweet, we will attempt to clean it and export it, using methods learned in this course. Additionally, we will create graphs using ggplot, and attempt to create a word cloud visualization out of the tweets. We will attempt to look at the tweets of our President, Donald Trump.

Requirements

This post will provide requirements and commands to install requisite packages immediately before the packages are used. However, for convenience, the required packages are listed here, followed by the commands to install them.

Required Packages

- rtweet
- ROAuth
- dplyr
- tm
- SnowballC
- wordcloud
- RColorBrewer
- formattable
- ggplot2

Commands To Install:

```
install.packages("rtweet")
install.packages("ROAuth")
install.packages("dplyr")
install.packages("tm")
install.packages("SnowballC")
install.packages("wordcloud")
install.packages("RColorBrewer")
install.packages("formattable")
install.packages("ggplot2")
```

In addition to the above packages, the data scraping section of this post requires Twitter API keys. However, as an alternative to obtaining said keys and to doing the data scraping, this is a link to a csv containing the results of the data scraping section: Trump Tweets CSV

Because the csv is available, the data scraping section is therefore optional; after downloading the csv, you can skip directly to the "Data Visualization" section and proceed from there. If the data scraping section is skipped, only the last 7 packages are required, namely "dplyr", "tm", "SnowballC", "wordcloud", "RColorBrewer", "formattable", and "ggplot2".

Setup

Scraping the data requires access to the Twitter API, and thus requires API keys. However, obtaining these can be a minor hassle, so I will provide a link to a csv containing scraped data below, after the data scraping section. Alternatively, it is possible to scrape your own data by registering a Twitter application, then subsequently following the steps below.

Data Scraping

To begin scraping the data, we require a few additional packages. To install them, run the following commands:

```
# Installing needed packages
install.packages("rtweet")
install.packages("ROAuth")
```

The 'twitteR' package is used to search and scrape the tweets. The 'ROAuth' package allows us to authenticate our API keys.

After installing, the packages need to be loaded.

```
# Loading packages
library(rtweet)
library(ROAuth)
```

We now need to set up our API keys. API stands for Application Programming Interface. Essentially, using our API keys, we can interact with Twitter's servers to scrape data. To get API keys, you need to register for a Twitter account and create a Twitter Application. If you do not wish to do so, there should be a link to a csv containing already scraped data below.

After obtaining the API keys, you need to authenticate with Twitter to begin scraping. To do so, we will need to create a twitter token.

```
# Setting up API key
appname = "YOUR APP NAME GOES HERE" # Put appname here
key = "YOUR API KEY GOES HERE" # Put API key here
secret = "YOUR API SECRET GOES HERE" # Put API secret here
twitter_token = create_token(
  app = appname,
  consumer_key = key,
  consumer_secret = secret) #Creates twitter token, used for rtweet
```

After executing the above commands, your browser should appear. You should click authenticate on the popup that appears.

Now that we have authenticated with twitter, we can now begin to use the rtweet package to scrape data from Twitter!

To begin with, we will get some recent tweets from President Trump's twitter using the get_timeline function.

```
# Gets tweets from @realDonaldTrump's timeline
trump_tweets = get_timeline("realDonaldTrump", n = 5000)
```

We now have a data frame, named trump_tweets, which we know how to manipulate using methods learned from class.

If you take a look at the data, you will see that the data frame is massive. However, there are many columns we do not need. Before exporting to a csv, we should strip the data frame of unnecessary columns. To perform this stripping, we will use the dplyr package.

To install dplyr, you can run

```
# Installs dplyr
install.packages("dplyr")
```

Then, to load the package run this command:

```
# Loads dplyr
library(dplyr)
```

Now, we want to select the columns we need. We will only be dealing with some of the numbers, and the actual text of the tweet itself, so we need not worry about many of the other columns. We are losing a fair amount of possibly interesting data; however, for the sake of simplicity, we move onward with only a few select columns.

```
trump_tweets = select(trump_tweets,
                      status_id,
                      created_at,
                      user_id,
                      screen_name,
                      text,
                      source,
                      favorite_count,
                      retweet_count)
```

Now that we have somewhat condensed our data, we can now export the csv.

```
write.csv(trump_tweets, "../output/TrumpTweets.csv")
```

Now that we have finally scraped our data, we can move on to some analysis.

Data Visualization

CSV Download

The CSV obtained from the previous section can be downloaded here: Trump Tweets CSV

To load the csv, run the following command.

```
trump_tweets = read.csv("../output/TrumpTweets.csv") # Replace the path to TrumpTweets.csv if necessary.
```

Word Cloud

The first thing we are going to do is create a word cloud. A word cloud is a form of data visualization that attempts to highlight the most used words in a corpus, or a body, of text.

To begin with, we want to create a file containing the text from all the files we obtained from the data scraping section. We also need a few additional packages.

```
# Installing packages
install.packages("tm")
install.packages("SnowballC")
install.packages("wordcloud")
install.packages("RColorBrewer")
```

```
# Loading packages
library("tm")
library("SnowballC")
library("wordcloud")
library("RColorBrewer")
```

Using the 'tm' package, we will create a body of text consisting of the raw text of all of Trump's tweets.

```
tweet_text = Corpus(VectorSource(trump_tweets$text)) # Creates a text corpus
```

Now that we have the body of text, we need to perform some cleaning on it.

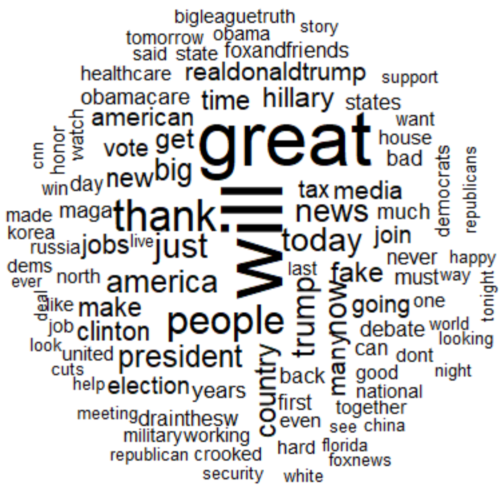
```
tweet_text = tm_map(tweet_text, content_transformer(tolower)) # sets to lower case
tweet_text = tm_map(tweet_text, removePunctuation) # Remove punctuation
tweet_text = tm_map(tweet_text, PlainTextDocument) # Convert to text doc
tweet_text = tm_map(tweet_text, removeWords, stopwords('english')) # Removes commonly used words.
tweet_text = tm_map(tweet_text,  stripWhitespace) # Removes excess whitespace
```

Because some of the tweets contains URL, we need to remove them using regex. We will do the same thing with ampersands.

```
removeURL = function(x) gsub("http[[:alnum:]]*", "", x)
removeAMP = function(x) gsub("amp", "", x)
tweet_text = tm_map(tweet_text, content_transformer(removeURL))
tweet_text = tm_map(tweet_text, content_transformer(removeAMP))
```

Now we can plot the wordcloud, using the 'wordcloud' package

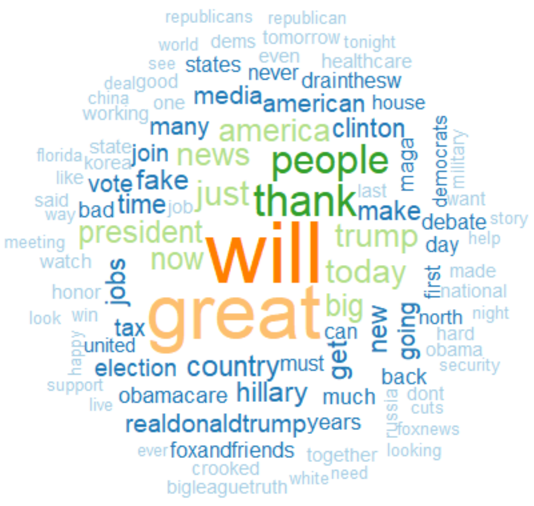
```
wordcloud(tweet_text, max.words = 100, min.freq = 1, random.order = FALSE, rot.per = .1)
```



Some interesting observations can be made from this word cloud -- firstly, the words "will" and "great" are the largest. Much of Trump's platform stems from his promise to "Make America Great Again", in his words, so it appears that the word cloud confirms that he is sticking to that.

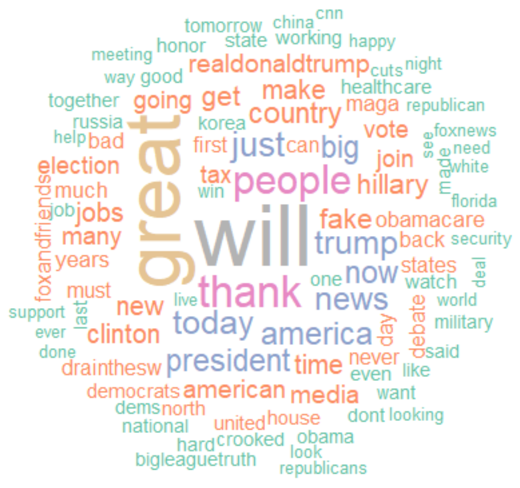
From an aesthetic perspective, the word cloud is not fantastic looking. We can make some changes to improve that. Using the RColorBrewer package, we can specify a color palette to be used in the word cloud.

```
wordcloud(tweet_text, max.words = 100, rot.per = .1, random.order = FALSE, min.freq = 1, colors = brewer.pa
```



To use a different color palette, you can just change the parameter within brewer.pal to something else. For example:

```
wordcloud(tweet_text, max.words = 100, rot.per = .1, random.order = FALSE, min.freq = 1, colors = brewer.pa
```



Scatterplot and Table

Now that we have a word cloud, we want to create perhaps more numerical, qualitative visualizations -- a table and a graph. First, we want to convert the corpus to a TermDocumentMatrix, which contains the frequencies of the words in one column and the words themselves in another. We then convert the term document to a matrix, which we then sort then convert to a data frame.

```
termDoc = TermDocumentMatrix(tweet_text)
matrix = as.matrix(termDoc)
sorted = sort(rowSums(matrix), decreasing = TRUE)
tweet_frequencies = data.frame(words = names(sorted), freq = sorted)
```

Table

Now, simply using the "head" function, we can output a table containing the most popular words.

```
head(tweet_frequencies, n = 10) # Returns 10 most frequently used words.
```

```
##                 words freq
## will             will  624
## great           great  545
## thank           thank  283
## people         people  242
## just             just  195
## today           today  184
## america       america  183
## trump           trump  180
## news             news  177
## president president  166
```

Alternatively, to output a nicer looking table, we can use the "formattable" library. As usual, to install, run the following command:

```
install.packages("formattable")
```

Then, we can load the formattable package.

```
library(formattable)
```

Now we can use it to output a table. We use the slice command to ensure that the table is not too large.

```
formattable(slice(tweet_frequencies, 1:10))
```

words	freq
will	624
great	545
thank	283
people	242
just	195
today	184
america	183
trump	180
news	177
president	166

Graph

Now, we can create a graph.

Just like in class, we're going to use ggplot. To install, do the following command:

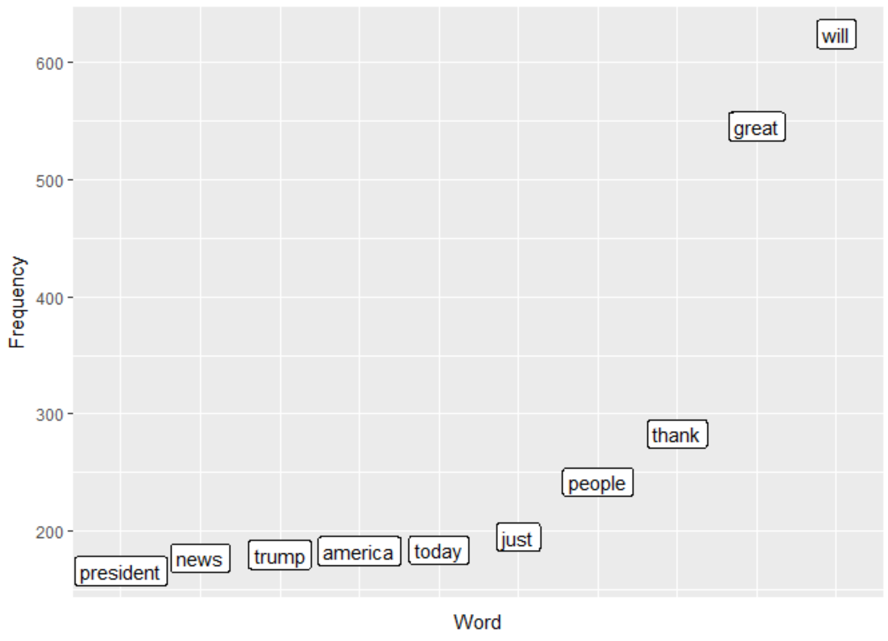
```
install.packages("ggplot2")
```

Next, we need to load the library.

```
library(ggplot2)
```

Now, using the ggplot command and others, we can create a simple scatter plot of the first 10 most commonly used words

```
ggplot(slice(tweet_frequencies, 1:10), aes(x = reorder(words, freq), y = freq)) +
  geom_point() + # Adds points to graph
  geom_label(aes(label=words)) + # Adds labels to graph
  theme(axis.text.x=element_blank(), axis.ticks.x=element_blank()) + # Removes x axis labels
  xlab("Word") + ylab("Frequency")
```



It seems like the two most used words are great and will, an observation repeated from the word cloud. However, this time, we can see concrete numbers for the number of times they have been used. Word clouds served us the information more directly, with the larger words being evidently used far more; however, the table and the scatter plot yielded more accurate results.

Conclusion/Reflection

Take Home Message

The purpose of my post was to gain experience in using an unfamiliar R package, and to produce actual, concrete results in it. In this I was successful -- I was actually surprised by rtweet's ease of use. The most troublesome part was signing up for API keys; otherwise, rtweet had a few simple commands that produced data that could easily be manipulated using methods learned from class. The documentation for many of the packages was clear and relatively simple as well, letting me create something interesting without too much extraneous reading or headache. Another interesting draw from this project was the suitability of different types of data -- I saw that both a scatter plot and a wordcloud drew identical conclusions on the same data, however, a scatter plot provided more concrete numbers while a wordcloud produced a more aesthetically pleasing plot.

Perhaps the central take home message is that it is possible, and perhaps even relatively easy, to use R to scrape and analyze data, and to produce aesthetically pleasing graphs that I believe to be acceptable for mass consumption. Throughout this post, I learned more about R and gained some independence -- no longer did I have a lab to follow, or instructions. I struck out on my own, and tried to produce a result I was interesting in with some success.

My post could be expanded on in several ways. There are a number of analyses to be done regarding Trump's tweets -- perhaps I could do something regarding the post times, as I know our president is infamous for tweeting at odd hours of the morning. Furthermore, I could do an analysis on the words itself, perhaps using some sort of heuristic to judge his words. The possibilities are near-endless, and learning to scrape data on my own from a real life data set was an invaluable skill.

References

"Text Mining and Word Cloud Fundamentals in R : 5 Simple Steps You Should Know." Text Mining and Word Cloud Fundamentals, www.sthda.com/english/wiki/text-mining-and-word-cloud-fundamentals-in-r-5-simple-steps-you-should-know.

"How to Use R to Scrape Tweets." R-bloggers, https://www.r-bloggers.com/how-to-use-r-to-scrape-tweets-super-tuesday-2016.

"Package 'tm'." R Documentation, https://cran.r-project.org/web/packages/tm/tm.pdf.

"twitteR." Geoffjentry's Github, https://github.com/geoffjentry/twitteR.

"Building Wordclouds in R." datascience+, https://datascienceplus.com/building-wordclouds-in-r/.

"R gsub Function." Endmemo, http://www.endmemo.com/program/R/gsub.php.

"Twitter Analytics using R." Credera, https://www.credera.com/blog/business-intelligence/twitter-analytics-using-r-part-1-extract-tweets/

"GGplot Function Reference." Tidyverse, http://ggplot2.tidyverse.org/reference/.

"How to Write CSV in R." Rprogramming.net, http://rprogramming.net/write-csv-in-r/.