Methods

The research process is done in an open-source programming language called 'R', and most of the code is written in RStudio. This project is indebted to the hard work of R programmers who designed the development environment that this project is designed in, and also those who developed the packages that the analysis uses as tools. R, while relatively new at this time, proves to be one of the most useful tools available to the contemporary social science researcher – especially those employing digital methods.

One reason that R is so useful is it's being open-source on all levels. This has effects across the entire research process. For example, analysis in R often involves imputing data into functions and getting some output from it. In some closed-sourced programs, the actual processing of the data that occurs by these functions is masked to the user. In R the user can inspect the source code of that function, and determine exactly what is happening to their data as it passes through said function. The benefits in this case are many. The user has a very clear idea of what is happening with their research, but it also empowers the user to have total control over that function. Another benefit to using open source languages is the ability to freely share packages between users, users who often create packages for their own use but decide to share them gratis to anybody else who may need them. This benefit brushes against the benefits of using open source software in general, an interesting topic but one that is well beyond the scope of this work.

The language itself when used in a good IDE might seem challenging to some users who primarily use graphic analysis tools, but once the initial hump of discomfort is overcome R is a monstrously powerful tool for researchers. The most important reason that R is so powerful is because the baseline packages can be augmented with anything developers can come up with, meaning that most research methods can be done in R, and often times it is much easier. The following methods that are used in this analysis could have been done with a suite of research tools, but because R has such a strong base and a very active community of developers the project never needed to move away from it.

Data Collection

The first stage of research was to gather 'data'. The term data will be used hereafter, but must be qualified first to clarify that it does not represent numerical data in the strictest sense at this point. The form of the data at this stage was a 'tweet' and the corresponding meta-data from Twitter. This data had 16 different variables when collected from Twitter, the most important of which are: who wrote the tweet, the text of the tweet, if the tweet was favourited, screen name who it was a reply to if it was a reply, the date and time it was created down to the second, the device used to publish the tweet, if it was a re-tweet, who originally tweeted it if it was a re-tweet, how many times it had been re-tweeted, and the latitude and longitude of where that tweet was published. This level of data collection was made possible by the public facing API (application programming interface) that Twitter provides for researchers and developers. This API allows a user to develop an application which uses the data directly from Twitter, the level of which depends on which API the user has access to. Due to the limited resources on this project, the public facing Search API was used – although there are far more powerful API's available to developers and researchers willing to pay a fee. The problem with this level of access is that a users access to Twitter data is limited in time going back a week and 180 calls on data per 15 minute window, a limitation that is explored later in this work.  
 This research project did not involve access to this API directly however, instead this project depends on a package for R named “Twitter” (Gentry, 2015). This package was created for this accessing the Twitter Search API in R. It should be made clear that this package does not have any tools for analysis, it is just a tool made to collect tweets. As mentioned before, the API was limited in the number of calls it can make in a window, and this 180 calls per 15 minutes might sound like plenty, but each command in the data collection stage takes more than 1 call, meaning the early iterations of this project brushed up against this rate limit often. One method to solve this problem is to write all the commands into a script, and run that whole script in R, enabling the program to keep making commands if it was rate limited until the program entered a new window.

An example of part of this script:

*write.csv(twListToDF(searchTwitter("Harper",n=2000, since="2015-09-28", until="2015-09-29")),file="Harper0928.csv")*

Starting from the inside brackets and moving out there are 4 arguments: the search term, the number of tweets to collect, the dates of the earliest tweets to pull, and the dates from the late tweets to pull. This innermost bracket is what gets passed of first function of the Twitter package, the 'searchTwitter' function which searches Twitter based on user specified information. That 'searchTwitter' function is argument for the second function, the 'twListToDF' function which transforms the output of the 'searchTwitter' function into a data-frame, which is the main way that data is handled in R. This data-frame transforms what is essentially a series of vectors into the familiar column and row format that many are familiar with. Finally, that data-frame is then passed through the 'write.csv' function from the base packages in R, which saves the data frame as a. .csv file with the name specified in the second argument of the function.

This same type of function was used for each search term, on each day of the data collection period. There were 14 search terms in total: Harper, Trudeau, Mulcair, Elizabeth May, Conservative, Liberal, NDP, Green Party, #cdnpoli, #elxn42, #elxn2015, Canadian Politics, Canada, #oct19. Some search terms were added partway through the data collection period, as a response to the trending topics on Twitter regarding the election. The earliest data is from September 28th, and the latest data is from November 1st. The average N for each of these functions was roughly 4000 tweets to be collected for that search term on that day, though that is sometimes is high as 6000 and as low as 100. Since there are 14 terms it often came to be that each day there was anything from 50-60 thousand tweets being collected each day over 30 days, the total number of tweets analyzed in this project comes in at a little over 1.5 million. As mentioned earlier there is a limit on how far back the Twitter API lets a user collect data from, for this reason the event that was observed has to be something occurring near the time of data collection. Using this API it is impossible to go back and research 'historical' tweets, a topic for later on in this project.

It is important to discuss the decision process involved with defining the search parameters. The two arguments of the searchTwitter function that define what gets collected that enable the researcher any real control over were the arguments for the search term, and for the number of tweets to be collected. This number could be much higher than 4000 tweets, but higher than 4000 meant being rate limited far more often. This means that the data collection process would be very slow process, if the data got collected at all without being the program crashing. This problem is also coupled with the potential consequence of Twitter revoking access to the API or the application being blacklisted when the rate limit gets abused. For these reasons, a reasonable N is between 4000-5000 tweets per search term per day. That N does not guarantee that there is always going to be that many tweets on that day for that search term, in many cases the actual number of tweets fulfilling the date argument fell short of the N argument, resulting in data-frames that were short in cases relative to other days. For this reason, not every day can be directly analyzed and needed to be normalized in the processing portion of the project.   
 Since the subject in question was the 2015 Canadian Federal Election, it is wise to include as many search terms related to that as resources allow for. The major parties: Liberal, Conservative, NDP, and Green Party. The leaders of those parties: Harper, Trudeau, Mulcair, and Elizabeth May. These search terms had to be short enough to not exclude important tweets, but specific enough to include only those which pertained to the election. “Liberal” for example would search for any tweet including the word “liberal”, in upper or lower case. 'Liberal' was chosen rather than “liberal party” because that would exclude tweets that might have left out the word 'party'. The balance is trying to avoid being too general, this search term would all pull tweets out that have the word “liberalized” which might have nothing to do with the election for example. The same logic has to be applied, but in the case of the Green Party and Elizabeth May it was not possible to just have the last name of the leader and 'Green' since those would be too vague of a search term. For this reason, it might be said that there is a fundamental difference in Twitter data starting at this point, but the line has to be drawn someplace. This practice will be discussed in later sections of the project.

The dates that selected were based on the 2015 Election campaign, and the date of the election. The data actually begins after the start of the campaign, but the start does serve as a good point to lead towards the election date. This is a project on the narrative of the election, and so the data does not necessarily need to be begin on the same day as the start of this historically long campaign. It would have been more beneficial to start on that day, but due to project limitations this was not possible. It is also because of those limitations that it is not possible to confidently say much about the start of the campaign narrative, there is only confidence in the early days and the wind up period before the election. The data also extends beyond the date of the election, not strictly at the end of the campaign. This is to try to capture the ending sentiments and the wind down period after the election. The range of time that this data has effectively captures the narrative of the 2015 election, but of course this is one area that could be seen as a limitation on the end results.

Once all the Twitter data was collected, it is then concatenated each. .csv file which represented an individual search term on a single day into one main .csv file for each search term spanning the whole data collection process. This was one of the only processes not done inside R, but rather using a simple Linux command. The end result of this data collection stage was 14 different main files each containing over 100 thousand tweets worth of Twitter data, spanning over the course of the month of October.

Data Processing

There are a number of tutorials[[1]](#footnote-2) using R to analyze Twitter data based on sentiment, this so called sentiment analysis was the basis for this analysis. Not everything from all of these sources is used on this project, but part of the research process was discovering uses of R for this and these tutorials are indispensable.

One function that used from another source is Jeffery Breen's sentiment scoring function as seen here:

*score.sentiment <- function(sentences, pos.words, neg.words,.progress='none')*

*{*

*require(plyr)*

*require(stringr)*

*scores <- laply(sentences, function(sentence, pos.words, neg.words){*

*sentence <- gsub('[[:punct:]]', "", sentence)*

*sentence <- gsub('[[:cntrl:]]', "", sentence)*

*sentence <- gsub('\\d+', "", sentence)*

*sentence <- tolower(sentence)*

*word.list <- str\_split(sentence, '\\s+')*

*words <- unlist(word.list)*

*pos.matches <- match(words, pos.words)*

*neg.matches <- match(words, neg.words)*

*pos.matches <- !is.na(pos.matches)*

*neg.matches <- !is.na(neg.matches)*

*score <- sum(pos.matches) - sum(neg.matches)*

*return(score)*

*}, pos.words, neg.words, .progress=.progress)*

*scores.df <- data.frame(score=scores, text=sentences)*

*return(scores.df)*

*}*

There are 3 main parts to this function. The first section dealing with formatting the text so as to remove punctuation and change everything to lower case. The next part is splitting each tweet apart into word while still keeping them inside the same tweet wrapper. The last part is comparing each word to a list of words called a lexicon, in this case 'pos.words'. The score that each tweet receives is the sum of the positive matches minus the sum of the positive matches. This is a fairly simple algorithm, whose function is to just describe how many words from a list of words are found in any given tweet. This sentiment score represents the first dimension that these tweets are analyzed on.

The list of words for sentiment is actually called the 'Subjectivity Lexicon' from Theresa Wilson , Jaynce Weibe, and Paul Hoffman all from the University of Pittsburgh who used it as a part of the paper describing this type of phrase level analysis described earlier. The actual word list itself comes off of Jeffery Breen's github, the same lexicon he uses for his sentiment analysis. The process that Wilson,Wiebe and Hoffman describe is much more involved that a Bayesian matching algorithm, but due to a lack of resource it could not be implemented in this research.

Sentiment is not the only metric that these tweets are scored on, there are many other dimensions of analysis that can be explored. These other dimensions include: economics, healthcare, climate, and foreign policy. Each dimension required its own lexicon, and without having the benefit of existing lexicons it is a requirement to write them. The lexicons are more or less just words pertaining to that topic, and the creation of these lexicons was little more than modifying existing glossaries and indexes of these words to fit the algorithm. This is another area where it is important to find a proper balance and having to draw an arbitrary line. Like the search term definitions, the researcher runs the risk of being either too inclusive or too exclusive in this definition of what might be an 'economic' word. Being too inclusive means a type I error is more common and being too exclusive means its a higher chance of a type II error. There is no perfect lexicon, and it is a relatively small problem since all tweets regardless of search term are passed through the same lexicon, and so any bias present in one will be present in the other. This means on the whole, things might seem more 'economic' than another for example, but there will be no bias between search terms.

There is also the problem of some words having the potential to belong to many different lexicons and being double counted. Rather than picking and choosing which words should go where and imbuing the analysis with my own bias, this analysis includes such words in both lexicons and accepts the risk of double counting a few tweets. This is a problem that is wholly unavoidable, and while it does present a limitation on this work, that limitation is the product of the methodology itself. To do this type of analysis at all means that there will be some new types of unavoidable limitations, and so rather than scrap the whole work it should just simply be noted that there is these types of issues.

The sentiment scoring function proved useful not just for analyzing sentiment, but also for scoring the tweets on all the above dimensions with some slight modifications. The measurement of sentiment can either be positive or negative, which is reflected in the score function, but how focused on 'healthcare' a tweet is cannot be negatively focused, and so that score must only be positive. As such the scoring function is just equal to how many matches each tweet has on the lexicon of whichever dimension is being tested for.

The parsing code is available for review outside of this section as it is too long to document here, but is to be described in short detail. The first part of the code imports the data set to be analyzed, and does some preliminary formatting as found in this tutorial[[2]](#footnote-3). One major function of the preliminary formatting is that it deletes the duplicate tweets, but not the re-tweets which should serve to reinforce the narrative present in the tweets. This is a conscious decision to leave re-tweets in the analysis as they are part of the narrative as well, and while they do not necessarily represent separate ideas from the original tweet they are none the less important.

Next is the definition of the scoring function for each of the election themes, and the loading of each themes corresponding lexicon. After all has been defined, the data is passed through each scoring function, each tweet is given a score for each dimension as it passes through each function and those scores are tallied up into one final data-frame. This data frame has the columns for each possible permutation of the dimension scores and the date, and a number of tweets that fit those parameters. This number of tweets varies from day to day, sometimes due to the specified N and sometimes due to the variance of tweets for that day, and so at this stage these numbers were normalized to represent a proportion of all tweets for that day. That data-frame with aggregate data is saved, one for each of the search terms tested for. It is this aggregate data that goes on to be visualized and then analyzed in the next stage.

Data Visualization

The visualization stage of the project is the stage in which the most creative liberties can be taken. Before this stage, the data is still in a fairly cumbersome format, while it is not as hard to manage as 1.5 million tweets it is still a lot of numbers to keep track of. The visualization stage is an attempt to format this aggregate data into something more manageable, to understand the story that the data tells us in a single glance. One of R's biggest strengths that it provides to users, and an area with some of the most active development, is in visualization.

Part of digital research is going to have to be the publication and communication of the work being done. This means that there has to be tools available to researchers to accomplish this, and there is no better tool for sharing information than the internet. There are tools available across the board for self-publishing figures and interactive charts, but none as integrated directly into the work as Shiny is to R. Shiny is a platform for making digital interactive web-apps, and is available for the most commonly used R environment, RStudio. What Shiny allows for is the free publication of a researcher's data, directly from the researcher and directly from the workspace that it was developed in. This means that for very large projects, the early stages can be published and researchers can provide daily updates with a few clicks.

This section on visualization depends heavily on the work of Hadley Wickham, an R developer who has made much of the power of R accessible to users. The package that Wickham developed that was most useful for this project was his 'ggplot2' package, a data visualization tool for R. Like Shiny, there are many tools available to researchers which enable similar tasks to be accomplished, but none are as fast or as integrated as the packages are to R. The visualization technique that I used for this project is a simply bubble chart, wherein the size of the bubble corresponds to the proportion of tweets that day fitting the co-ordinates of the plot. The Y-axis in these plots was always the sentiment score, and the X-axis was user-specified using a pull-down menu which selected any of the other election dimensions. All of these plots rendered in 'ggplot2' were then passed through another package called 'plotly', which enables interactivity such as zoom and pan with the graphs.

The final product of this visualization is a website in which a user can visit and explore the data for themselves. While this project has its own conclusions, that does not mean that curious others can't visit the website and explore the Twitter data for themselves. It is this that represents a fundamental shift between traditional methodology and digital methodology, wherein the entire process can be built upon by others (provided the tools stay available). This entire project is intended to be public, and so at any stage in the future anybody with the desire can audit the project to find and fix flaws, or to build on it themselves, or to simply explore it for themselves and make their own conclusions.

1. [https://sites.google.com/site/miningTwitter/questions/sentiment/sentiment](https://sites.google.com/site/miningtwitter/questions/sentiment/sentiment), [http://www.r-bloggers.com/Twitter-sentiment-analysis-with-r/](http://www.r-bloggers.com/twitter-sentiment-analysis-with-r/), [http://thinktostart.com/create-a-wordcloud-with-your-Twitter-data/](http://thinktostart.com/create-a-wordcloud-with-your-twitter-data/), https://jeffreybreen.wordpress.com/2011/07/04/Twitter-text-mining-r-slides/ [↑](#footnote-ref-2)
2. [http://analyzecore.com/2014/04/28/Twitter-sentiment-analysis/](http://analyzecore.com/2014/04/28/twitter-sentiment-analysis/) - This tutorial is one of the more fundamental works for the parsing segment of this project, and so this along with the prior tutorials and Jeffery Breen deserve credit for their influence on this project. Much of the code I used is just slightly modified versions of what I found there and so these authors/developers/educators deserve the credit for the parsing section of the analysis. None of the work used is proprietary, but it is still important to give credit where it is due. [↑](#footnote-ref-3)