CES 632 Project 1:

Data Mining Tweets on the 2020 Presidential Election

Chrisina Giagnoni

September 17, 2020

Abstract

We will collect status updates, called "Tweets" from the social media platform Twitter, and discuss the full process of obtaining Tweets, exploring the data, determining our hypotheses about the data, cleaning the data, visualization of the data, and finally draw some conclusions about our findings. We will use the 2020 Presidential election to narrow down our key word search, so each Tweet collected should pertain to the election. R software will be used to explore the features and content of the collected data. Once we have an idea of how the raw data was structured, we came up with three hypotheses: the distribution of Tweet frequency would vary by day and time; if the words in the Tweet text and hashtag text would reflect current election news at the time; and if Donald Trump was the most retweeted or quoted Twitter user in the realm of election Tweets. Keeping the hypotheses in mind, we decided on what information to keep and what kind of visualizations we wanted to use. Based on what we learned in data exploration and our hypotheses we could begin data cleaning and preprocessing. We broke down the preprocessing of the data into five basic steps, covered the R packages and code used to get the data in a form that is best for each visualization technique, as well as explain the purpose of each preprocessing step. In the visualization section we present three types of visualization each meant to offer insight into each of our hypotheses: a histogram of the Tweet frequencies; bar charts for the word frequencies in Tweets and hashtags; and word clouds of the most retweeted and quoted users from the historical and new data sets. Based on our visualizations, we determined that our hypothesis about Tweet distribution appeared to be incorrect, since Tweets were distributed uniformly over the time intervals. We cautiously accept our second and third hypotheses to be backed up by our visualizations, but we did find some interesting anomalies in the data.

3.1 Problem description

Twitter is a worldwide social media platform where users "Tweet" "status updates". There are three main types of Tweet: an original Tweet in which the content is new; a retweet where a user just shares another user's Tweet; or a quote Tweet which is similar to a retweet but contains new comments from the user. Tweets can include more than just text. They can include media such as videos and pictures; or non-text non-media content such as emojis. Since Twitter is a worldwide platform languages from around the world can be seen in Tweets, however Americans are the predominant nationality on Twitter (Clement, 2020). As mentioned in the New York Times, Twitter is well-known as a political tool in the U.S., stating in the article titled "Twitter is a big deal in politics. That doesn't make it right.",

> It's hard to overstate the role Twitter now plays in politics.
>
> It's the president's favorite form of communication. It's where public officials make statements, where activists pressure politicians and where reporters announce their latest scoops. It's where the "conventional wisdom" forms and where our national political narrative is created.
>
> And it's totally unrepresentative of America.

(Lerer, 2019). The NYT article brings up some interesting points, like Donald Trump is a frequent Tweeter, and we can keep those in mind as we explore, process, and analyze the data (Lerer, 2019). Also, from a practical standpoint, the 2020 Presidential Election we would generate the required 500+ MB of data relatively quickly. Thus, we made the choice to research the looming 2020 presidential election from a snapshot of two separate data sets collected on two different days. The first data set will be referred to as the "historical data" collected in August, and the second data set collected in September will be referred to as the "new data". Both data sets were collected using the keywords to "presidential election", "2020 election", "Trump re-election", "Biden 2020". After collecting the data we will use R to explore, make hypotheses about the data, preprocess the data, then use visualization tools to draw conclusions about our hypotheses.

3.2 Data Gathering

To gather data, we first applied for a developer account on Twitter to get the API key and secret, as well as the access token and secret. Then we edited the `tweets.py` file downloaded from Blackboard by adding the keys and secrets obtained from Twitter, and changed the keywords to “presidential election”, “2020 election”, “Trump re-election”, “Biden 2020” to match the keywords in the historical data set on the 2020 election.

On Saturday September 5, 2020 from 15:15 to 18:49 PM UTC we collected 571.6 MB of data in JSON format, but compressed to 312.6 MB after parsing to CSV. The historical data was 510.3 MB in JSON and 293.6 MB in CSV, so the data we collected is similar in size to the historical data. To parse the data from JSON to CSV we downloaded the `tweetparser.py` script, and used it to create our two CSV files, which were then ready to explore in R.

3.3 Data exploration

Before we can begin preprocessing we must first explore the data. We want to check the dimensions of our data: how many features are present? How many observations of individual Tweets are there? We need to find how much of the data is missing or not applicable. We also want to figure out what information is under what feature. Once we have a good idea of what features we want to keep, we can take a closer look at the raw data to get an idea of what hypotheses we want to make, what kind of visualizations we want to use, and needs to happen in the preprocessing phase.

We read in the historical and new data sets in R so we could begin looking at individual features and observations. We referenced the Twitter data dictionary to determine what the features were (Twitter, N.D.), and found that the majority of the data could be categorized by “parent objects” listed as follows:

Tweet object - contains information about the Tweet, including the tweet text

User object - contains information about the user who Tweeted

Entities object - contains common things included in Tweets like links, user mentions, etc

Extended entities object - contains information about media in the Tweet

Geo objects - contains information about locations

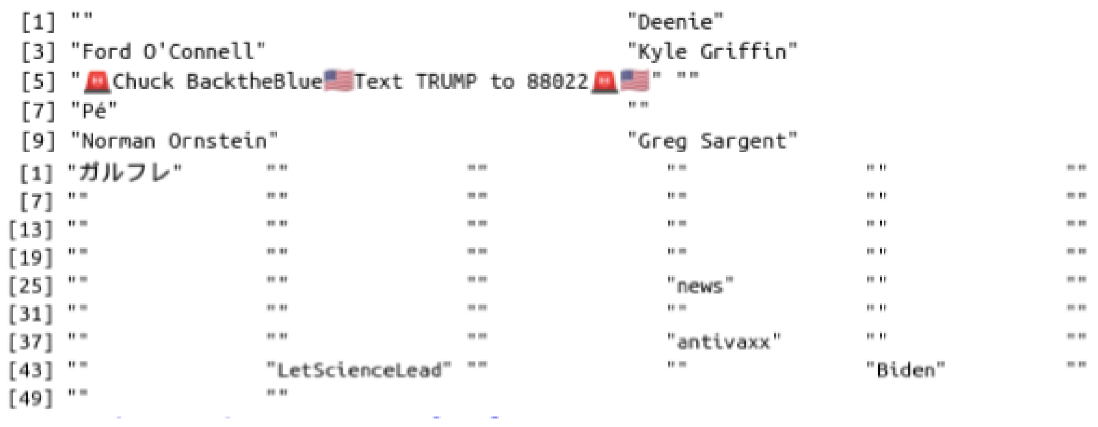
Each of these “parent objects” has a subset of other data characteristics called “child objects” so most of the features look like `parent_object.child_object`. There are some objects that are not contained within a “parent object”, but we found to be useful. The two most useful stand-alone features were `lang` which tells the language of the Tweet, and `created_at` tells the date and time the Tweet was created.

By looking at the first and last entries in the `created_at` column we found that the historical data was collected on Friday, August 14, 2020 from 15:31 to 18:10 UTC and the new data was collected Saturday September 5, 2020 from 15:15 to 18:49 PM UTC.

Then we checked the size of the data sets, and found that the historical data had 71,662 observations and 998 varibles. The new data had 70,340 observations and 991 varibles. Since there are nearly 1000 features in both data sets, it is reasonable to assume that many of the columns are not useful for our purposes. So our first task is to reduce the number of features of the data set to the features which will help us with our problem described in 3.1. We used R to get a table of `NA` counts, and found that both data sets had 78 features that were entirely made up of `NA`s. The first two features, contributors and coordinates were both etirely made up of `NA`. So we knew that we could at least reduce our data by 78 variables. As a general snapshot, we calculated that the proportion of `NA` entries is nearly 40% for both data sets.

We also want to note that blanks are distinct from `NA`. For example, not every Tweet is a retweet or quote, so instead of `NA` where the original Tweeter’s user name would be there is a blank space `""`. Similarly, not every Tweet has hashtags, so we would see a `""` instead of an `NA`, both of these can be seen in Figure 1.

We checked what percentage of the data is either blank, `""` or `NA`. About 85% of both data sets is either blank or `NA`. Our hypotheses would determine what we do with the blank spaces or `NA`. For example, if we want to know if Donald Trump is retweeted more than anyone else, the blank spaces would just be ignored, since we only care about the subset of data that is retweets. However, blanks can also also offer information, like the frequency of retweets, since a blanks in the retweeted user name would indicate that Tweet is original or a quote Tweet.



```
 [1] ""                                              "Deenie"
 [3] "Ford O'Connell"                                "Kyle Griffin"
 [5] "🚨Chuck BacktheBlue🇺🇸Text TRUMP to 88022🚨🇺🇸" ""
 [7] "Pé"                                            ""
 [9] "Norman Ornstein"                               "Greg Sargent"
 [1] "ガルフレ"        ""               ""              ""              ""              ""
 [7] ""               ""               ""              ""              ""              ""
[13] ""               ""               ""              ""              ""              ""
[19] ""               ""               ""              ""              ""              ""
[25] ""               ""               ""              "news"          ""              ""
[31] ""               ""               ""              ""              ""              ""
[37] ""               ""               ""              "antivaxx"      ""              ""
[43] ""               "LetScienceLead" ""              ""              "Biden"         ""
[49] ""               ""
```

Figure 1: Retweeted user names and hashtag text data snapshots.

Since Twitter is used around the world, it is also reasonable to assume that there are different languages spoken, and the data includes a column for languages, `lang`. We used the `ggplot2` package in R to create a bar chart of the languages used in the Tweets we collected, shown in Figure 2. English is the dominating language followed by undetermined, then French and Spanish. During the preprocessing phase, we will want to remove any observations that contain non-English words for a few reasons. First, English dominates all other languages, so any other non-English words would likely be ignored anyway, and second, to reduce the size of the data set, and third, it will make cleaning the text itself easier since we will want to remove any special characters, e.g. letters with accents.

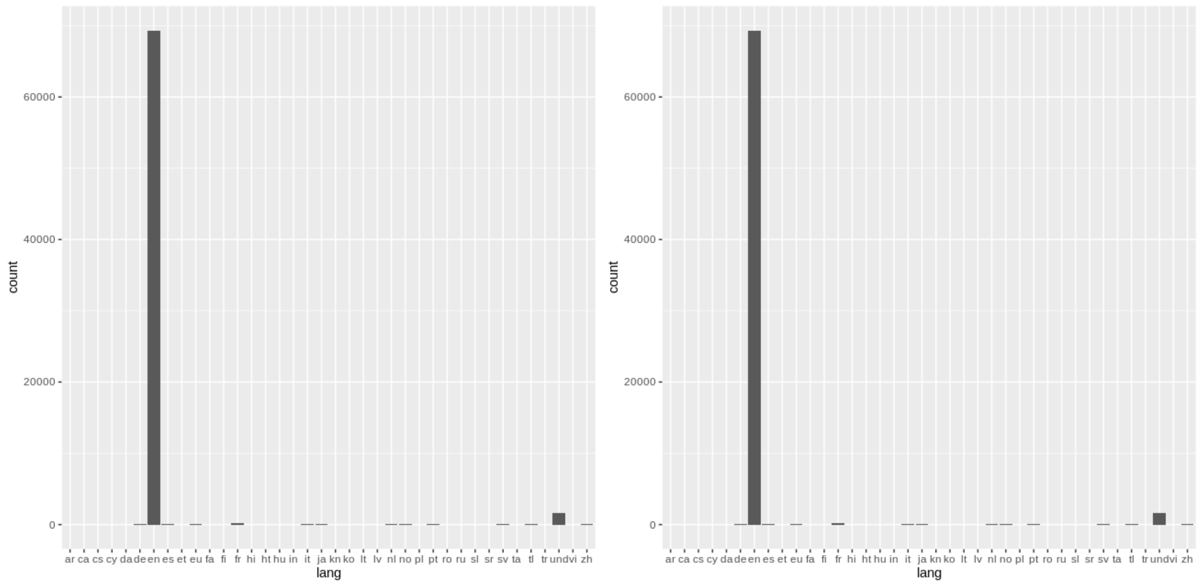


Figure 2: Bar chart of languages in Tweets: historical data left, new data right

Finally, we determined that the most useful and features to keep were

time - the time in UTC (to be extracted from `created_at`)

retweeted_status.user.screen_name - if the Tweet is a reTweet, this is the user being retweeted

quoted_status.user.screen_name - if the Tweet is a quoted Tweet, this is the user being quoted

text - the original text of the Tweet, this does not include quoted text

entities.hashtags.hashtags.text - the Tweet hashtags

After deciding what features to keep, we took a closer look at what the features we have not already explored looks like. We noted that while some features contained many blank spaces, none of the features contained any `NA`.

Since both data sets were collected on a single day, we do not need the date component, and the time feature will to be extracted from `created_at`. It can be a challenge working with time in R, so we will keep that in mind during the preprocessing and visualization phases. We will likely have to convert the data to numeric after extracting as a string.

The feature **text** is exceptionally noisy. It contains whether or not it is a retweet or quote with RT at the begining of the text, which we will want to remove. It also contains redundant data such as mentions with `@user` and hashtags with `#text` that need to removed from the data. Tweet text also includes non-text noise such as media, emojis, URLs starting with `http`, and indications of a new line with `\n`. Finally, we will want to remove punctuation, captialization, and stop words such as "the" and "a" - any common English words that do not contain any information.

We choose **retweeted_status.user.screen_name** and **quoted_status.user.screen_name** over the user's name since they contain the same information, but the screen name tends to be more tidy than the user name. Like the text, we will need to remove any emojis and capitalization, but otherwise they are generally tidy. Our hypotheses will determine what we do with the blank spaces.

The **entities.hashtags.hashtags.text** is the least noisy text feature, where we will only need to remove capitalization. Like with the screen names, we will determine what to do with the blank spaces after making our hypotheses.

Then we will be able to put the desired, cleaned columns into a new data frame which will be ready to use in visualization techniques or ready for a few more steps of preprocessing for a specific visualization.

3.4 Hypothesis

During data exploration, we noted that the historical data was collected Friday, August 14, 2020 from 15:31:42 to 18:10:04 UTC and the new data was collected on Saturday, September 5, 2020 from 15:15:01 to 18:47:46 UTC. This presented an interesting opportunity to compare Tweet frequency on two different days during the same time period. Since general working hours for most people are 9 AM to 5 PM, we would hypothesize that Friday mornings would be a more active Tweet time, and there would more tweets later in the day on Saturdays. Twitter activity could be useful to people who use the social media platform as a form of advertising, marketing or promotion since they would want to promote during these active times. We will also need to keep in mind that this is only a small snapshot of two days, and not even a full 24 hours worth of Tweet information. Furthermore, we only collected data using the keywords based around the election. Thus this information may be useful to someone promoting a political campaign, planning a future campaign.

According to a 2018 survey "Around seven-in-ten adult Twitter users in the U.S. (71%) get news on the site", and if only a small amount of news is spread on Twitter this could give a very small view of what is happening in the world to these users (Hughes, & Wojcik 2019). Additionally, since anyone can share virtually anything, one would think that it would be the perfect setting for "fake news". This could have an effect on public views on current events, and in our data, will give us insight into a users' viewpoint of 2020 Presidential election news. Thus, our our second hypothesis we want to look at the content of Tweets and hashtags. We hypothesize that hashtags and text words change based on news of the day and are going to handle the tweet text separately than the hashtags and see if they each reflect headlines around the dates the data was collected. We want to focus on English only text and hashtags.

Lerer claimed in the New York Times article, "Twitter is a big deal in politics. That doesn't make it right.", that Twitter is the President's favorite form of communication (2019). Trump has even gone as far as occassionally firing people via Twitter (Jeong, 2018). It is interesting to note that most Twitter users do not tweet very often and only "80% of all tweets from American adults come from the top 10% of tweeters" (Hughes, & Wojcik, 2019). Thus, if Twitter is Trump's favorite form of communication, according to Huges &

Wojcik, his voice should be one of the 10%, and his Tweets would take up a large chunk of real estate on that corner of the internet. So our third hypothesis is that Trump is retweeted and quoted more than any other Twitter user, and this has not changed between the historical and new data. For this we will combine the data from both the retweeted and quoted user screen names to find out whether or not Trump is the most retweeted or quoted Twitter user. Following hypothesis 2, this is important to note because if most information on Twitter is from a small sample of the population, this can skew the news the the 71% of Twitter users get on any given day. Thus, if our hypothesis proves true, the Twitter users that get news from Twitter may get most of it from the President.

3.5 Preprocessing

During the preprocessing phase, we want to clean the data so that it is ready for visualization in the following section. We will keep in mind our hypotheses, goals, and what we learned during data exploration to help guide our process. We will discuss each of the following five steps in their own section.

1. Remove any rows with non-English words.
2. Extract times from `created_at`.
3. Clean the text.
4. Put the cleaned data into a new tibble with the desired features.
5. Extract desired features into new tables or vectors appropriate for each visualization technique.

Recall the features we want to keep are **time**, **text**, **entities.hashtags.hashtags.text**, **quoted__status.user.screen__name** and **retweeted__status.user.screen__name**. Each feature had different problems, so each will need to be cleaned in a potentially different way. We have already read in the historical and new data sets in R, so now we need to load the packages that will help us with data cleaning.

```
#load packages for cleaning
library(tidyverse)
library(tidytext)
library(tm)
library(dplyr)
library(SnowballC)
library(stringr)
library(chron)
```

For our purposes, packages `stringr` (Wickham, 2019) and `chron` (James, et.al., 2020) are for dealing with the time data. Packages `tidyverse`(Wickham, 2019), `tidytext` (De Queiroz, et. al., 2020), `tm` (Feinerer & Hornik, 2019), and `snowballC` (Bouchet-Valat, 2020) are all for text cleaning, and `dplyr` (Wickham, 2020) is used for general data organization and cleaning.

Step 1.

We want to eliminate all observations with non-English words. To do this in R, we will use a common technique where we create an entirely new data frame by extracting only the desired rows and columns from an existing data set. We simply pick out a new name to assign the data frame, then assign the selected rows and columns to the new name. To select rows and columns we need to use R's syntax 'df__name[row__numbers__or__names, column__numbers__or__names]. To select all rows or columns, we simply leave the space blank. To select a row based on a condition from a column we need to know what the column type is. For example, is the data in the column numeric, characters, or factors? Once we figure that out we use the appropriate logical syntax to select items based on that condition.

To apply the technique, we are creating new data frames called `cl.election` and `cl.etweets` for the historical and new data, respectively, by selecting the rows that correspond to `en` under the `lang` column. We checked that `lang` is considered a character. Then `election$lang=="en"` tells the computer to only keep rows with the character string `"en"` for English in the language column. The blank after that tells the computer that we are keeping all variables. Thus this technique only changes the number of observations and does nothing to the number of variables.

```
#remove non english rows
cl.election=election[(election$lang=="en"),]
cl.etweets=etweets[(etweets$lang=="en"),]
```

Removing the non-English rows reduces the number of rows by only a few thousand observations in both cases. This is summarized in the following table which gives the number of rows in each data set before and after this preprocessing step, as well as give the number of rows removed.

	Historical Data	New Data
Before	71662	70340
After	69274	66337
Removed	2388	4003

Considering both data sets originally had over 70,000 observations, this does not change the data size very much, but will help with further cleaning and interpretation of our results. To summarize, there are 69,274 English observations in the partially cleaned historical data set and 66,337 English observations in the partially cleaned new data set. We do want to note that this has not removed every non-English word in the data set. It is entirely possible that someone wrote “bonjour, what a beautiful day” in a Tweet marked `en`, dispite the French word “bonjour” being in the Tweet text itself.

Removing the non-English observations is an important preprocessing step since our hypotheses are only concerned with English text. We can further justify this by noting that so few Tweets are non-English we would want to treat them separately regardless since they are such a small subset of our Tweet data as we saw in section 3.3. Furthermore, some of the following techniques will only work on English words, notibly removing English stop words in step 5.

Step 2

Similar to step 1, we are using a technique that creates a new data frame. In this case, we are just going to “write over” the partially cleaned data by naming our new data sets the same thing. We want to extract time from the `created_at` column in the partially cleaned historical and new data sets, since if we do not use the partially cleaned sets we will re-introduce the non-English rows into our data sets. In this case, we are not extracting desired rows, but extracting desired information from one column, every row.

Typically in R, we can use the `as.time` command to convert a character string in a certain format to a time variable. However, the Twitter time data, `created_at` is a character string in the format “day mon date HH:MM:SS +0000 year”, e.g. “Fri Aug 14 15:31:42 +0000 2020”, which is not a typical format. So in this case, the `as.time` command will not work. To work around this requires three steps: extracting the time as HH:MM:SS as a string, then converting it to time with the `chron` package in R, and finally converting it to a numeric variable. Converting to numeric allows us to create a histogram using the built-in command `hist` during the visualization phase. Histograms are only for numeric data, so we have to coerce it into a continuous, numeric variable. Since the times overlap, `hist` gives us the option to overlay the two bar charts so we can see both frequencies at the same time. R simply takes time, converts it to an hour decimal and then divides that result by 24, so it is easy to go back and forth between colon notation and numeric decimal time.

We have included a flow chart showing the conversion process from `election$created_at` to `cl.election&ntime`

`created_at` $\Longrightarrow$ extract HH:MM:SS $\Longrightarrow$ convert to colon time $\Longrightarrow$ convert to numeric

For example, to get from `Fri Aug 14 15:31:42 +0000 2020` to `0.6470139` by hand we would

Fri Aug 14 15:31:42 +0000 2020 $\Longrightarrow$ 15:31:42 $\Longrightarrow 15 + \frac{31}{60} + \frac{42}{3600} = 15.52833 \Longrightarrow 15.52833/24 = 0.6470139.$

Luckily, the few lines of code below do all of that work for us in a matter of seconds.

```r
#extract individual times
cl.election$time=str_extract(cl.election$created_at, "[0-9]{2}:[0-9]{2}:[0-9]{2}")
#convert string to time
cl.election$time=chron(times=cl.election$time)
#convert to numeric
cl.election$ntime=as.numeric(cl.election$time)
#extract individual times
cl.etweets$time=str_extract(cl.etweets$created_at, "[0-9]{2}:[0-9]{2}:[0-9]{2}")
#convert string to time
cl.etweets$time=chron(times=cl.etweets$time)
#convert to numeric
cl.etweets$ntime=as.numeric(cl.etweets$time)
```

Time is a strange variable since it could be considered a discrete factor (like it is in the raw data) or a continuous numeric variable (which we coerced out of the raw data). This technique is necessary for creating the histogram, since histograms only make sense for numeric variables. In this case, we wanted to use a histogram since they are great for setting an even "bin width" and we want to look at the frequency of Tweets over ten-minute intervals, since it made the most sense over our 4-ish hour time window in which we observed these election Tweets.

Step 3

Similarly to steps 1 and 2, we are going to continue to alter the entire data set. This time we are removing or changing noisy characters and strings. We will be using `gsub` which uses regular expressions to find strings to be replaced in a vector or data frame. The code in all will look something like `gsub(thing_to_replace, thing_to_replace_with, data)`. We also used the `iconv` command which converts character encodings, e.g. remove emojis. The `tolower` command simply changes any uppercase letters to lowercase, and `removePunctuation` removes punctuation. Like `gsub` we tell the command where to find the data we want cleaned, but both of `tolower` and `removePunctuation` do not need to be told what to replace the item with.

To apply these techniques, the order in some cases matters, for example, we first want to remove any non-text objects except punctuation. We need to keep the punctuation for removal of hashtags and mentions. We are using `gsub` and regular expressions to remove URLs and non-English characters such as emojis. In the columns we are concerned with, `text` is the only one that contains URLs, so we use the gsub command to remove anything starting with `http` or `https` by replacing the URLs with blanks, denoted `""` in the code. The Tweet text also contains indications of new lines with `\n`. We will use the same function to tell R to replace any `\n` with blanks, `""`. Next, we need to remove any non-English non-numeric (0-9) characters with the `sapply` (this just applies the `iconv` to everything) and `iconv` functions which will remove and replace any non-ASCII characters. This needs to be done on the Tweet text and user screen names. To remove mentions which are `@user_screen_name`, indications of retweet which are `RT`, and hashtags which are `#text`. We will use the `gsub` command again to replace any mentions, `RT`, or hashtags from the Tweet text. Then we can lowercase all letters with the `tolower` command. As a final clean up, we want to remove any punctuation using the `removePunctuation` command.

```r
#remove all urls
cl.election$text = gsub("http.*","",cl.election$text)
cl.election$text = gsub("https.*","",cl.election$text)
cl.etweets$text = gsub("http.*","",cl.etweets$text)
cl.etweets$text = gsub("https.*","",cl.etweets$text)

#remove all \n from text
cl.election$text <- gsub("\n","",cl.election$text)
cl.etweets$text <- gsub("\n","",cl.etweets$text)

#remove non-alpha non-numeric characters
```

```r
cl.election$text = sapply(cl.election$text,function(row) iconv(row, "latin1", "ASCII", sub=""))
cl.etweets$text = sapply(cl.etweets$text,function(row) iconv(row, "latin1", "ASCII", sub=""))

cl.election$retweeted_status.user.name = sapply(cl.election$retweeted_status.user.name,function(row)
  iconv(row, "latin1", "ASCII", sub=""))
cl.etweets$retweeted_status.user.name = sapply(cl.etweets$retweeted_status.user.name,function(row)
  iconv(row, "latin1", "ASCII", sub=""))

cl.election$quoted_status.user.screen_name = sapply(
  cl.election$quoted_status.user.screen_name,function(row)
    iconv(row, "latin1", "ASCII", sub=""))
cl.etweets$quoted_status.user.screen_name = sapply(
  cl.etweets$quoted_status.user.screen_name,function(row)
    iconv(row, "latin1", "ASCII", sub=""))

#remove all mentions
cl.election$text <- gsub("@[[:alpha:]]*","",cl.election$text)
cl.etweets$text <- gsub("@[[:alpha:]]*","",cl.etweets$text)

#remove all RT from text
cl.election$text <- gsub("RT","",cl.election$text)
cl.etweets$text <- gsub("RT","",cl.etweets$text)

#remove hashtags text from text
cl.election$text=gsub("#[A-Za-z0-9]+|@[A-Za-z0-9]+|\\w+(?:\\.\\w+)*/\\S+", "", cl.election$text)
cl.etweets$text=gsub("#[A-Za-z0-9]+|@[A-Za-z0-9]+|\\w+(?:\\.\\w+)*/\\S+", "", cl.etweets$text)

#lowercase all text
cl.election$text=tolower(cl.election$text)
cl.etweets$text=tolower(cl.etweets$text)

cl.election$retweeted_status.user.name=tolower(cl.election$retweeted_status.user.name)
cl.etweets$retweeted_status.user.name=tolower(cl.etweets$retweeted_status.user.name)

cl.election$quoted_status.user.screen_name=tolower(cl.election$quoted_status.user.screen_name)
cl.etweets$quoted_status.user.screen_name=tolower(cl.etweets$quoted_status.user.screen_name)


cl.election$entities.hashtags.hashtags.text=tolower(cl.election$entities.hashtags.hashtags.text)
cl.etweets$entities.hashtags.hashtags.text=tolower(cl.etweets$entities.hashtags.hashtags.text)

#remove punctuation
cl.election$text=removePunctuation(cl.election$text)
cl.etweets$text=removePunctuation(cl.etweets$text)
```

Without looking at the individual Tweets, there is not much to see here, we've simply removed any noise from the text objects: **text**, **hashtags.hashtags.text**, **retweeted_status.user.name**, and **quoted_status.user.screen_name**. By noise in this case, we mean URLs, emoojis, indications of new lines, mentions, indications of retweets and new lines, hashtags in the Tweet text, changed all letters to lowercase, and removed any punctuation.

To understand why this step is so important, we want to recall that our second and third hypotheses are only concerned with specific words, and not the way the words appear. To look at the content inside of a Tweet text, we only want to keep the Tweet text written by the user. Things like indications of new lines, retweets, mentions, URLs, punctuation, and emojis are unwanted in our word analysis. We also need to remove redundant information, like the hashtags, since we are considering that information separately in the hashtag text. Furthermore, we want `[emoji]biden` to be interpreted by R as the same as `biden`. The same things mentioned for text also applies to the hashtag text, except it didn't have things like mentions and URLs. For all text components, we needed to lowercase all letters. This is important since we want the software to read the words the same regardless of capitalization. Without this step, the software would consider `Biden` to be different from `biden` which would effect our counts since `biden` will be split into categories based on capitalization. This would hinder our evaluation and interpretation of our output since we would not get accurate counts on plain word usage.

Step 4

The next technique is creating new tibbles - a type of data frame - which contains only our desired information. In R, we choose a name for our tibble, use the `tibble` command from the `tm` package to name our desired features and put them into a tibble.

Now we put the partially cleaned data into new "tibbles" and name them `new.election` and `new.etweets` for the historical and new data, respectively. We have chosen names for each of the features that are easy to remember and make sense. We took the extracted numeric time column and named it `time.created`. For the retweeted and quoted user screen names we labeled those `rt.user` and `qu.user`. Then the original Tweet text and hashtag text features were named text and hashtag, respectively.

```
new.election=tibble(time.created= cl.election$ntime,
                    rt.user= cl.election$retweeted_status.user.screen_name,
                    qu.user=cl.election$quoted_status.user.screen_name,
                    text=cl.election$text,
                    hashtag=cl.election$entities.hashtags.hashtags.text)
new.etweets=tibble(time.created= cl.etweets$ntime,
                   rt.user=cl.etweets$retweeted_status.user.screen_name,
                   qu.user=cl.etweets$quoted_status.user.screen_name,
                   text=cl.etweets$text,
                   hashtag=cl.etweets$entities.hashtags.hashtags.text)
```

We used the `str` command to look at the structure of the new data frames. The cleaned historical data set has 69,274 observations and the cleaned new data has 66,337 observations. Both have the 5 features: `time.created`, `rt.user`, `qu.user`,`text`, and `hashtag`` which is the hashtag text`. The structure also tells us that time is the only numeric variable, and all of the text variables are treated as characters.

Tibbles are a type of data frame that work best with the packages for our final few steps. It makes finalizing preprocessing much smoother. We could have gotten by without this step, but it is nice to have all of our cleaned data and desired features in one easy-to-use tibble.

Step 5

Each visualization technique requires one or two more preprocessings step, but they are unique to each technique. For our three visualizations, we have chosen histogram, bar chars, and word clouds. To use `hist` with our numeric time data, we will not extract the column here since it will be wast to extract the column in the process of creating the histograms. Both of the visualizations with words require that we extract individual words into their own data frame, and for each part we followed different step-by-step tutorials which are referenced with each technique.

To get the text from the tweets and hashtags ready for a bar chart we relied on *Text Mining with R* by Julia Silge and David Robinson to walk us through the steps. Included in the tidy text section was a flow chart describing the process.

Since we already did a lot of the work in steps 1-4, we only needed to unnest the tokens and remove stop words. Unnesting is a process in which the software extracts each word into its own observation. This creates

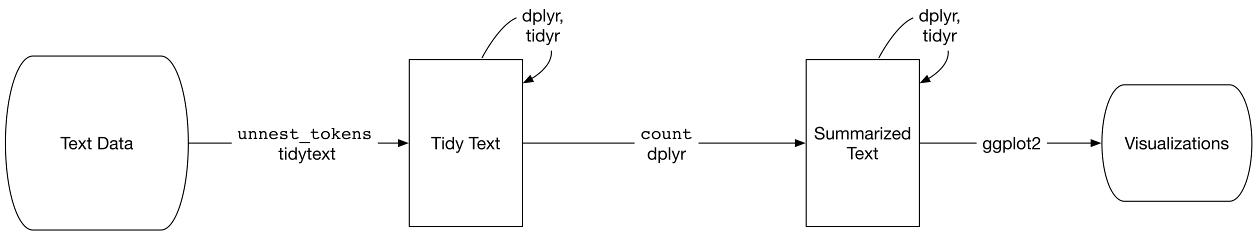


Figure 3: Flow chart of preprocessing for visualization with a bar chart (Silge, & Robinson, 2020).

a very long vector of single words with their frequency. The second step in this process is removing common words like “the”, “at”, and “in” called “stop words”. After that, the new data frame is ready to be made into bar charts.

To apply this technique, the following code creates a new data frame in to put the unnested words and their frequency, `dplyr` selects the column we want to unnest, and the `unnest_tokens` simply unnests the words into the new data frame. We can see the most frequent words after unnesting the historical data Tweet text in the following table.

```
#break the text into individual tokens tweet this creates a new data frame
new.election.text <- new.election %>%
  dplyr::select(text) %>%
  unnest_tokens(word, text)
```

Word	n
the	53617
to	43498
of	22733
is	21664
in	17305
and	16597

We can see in the table that if we were to stop preprocessing here, our visualizations would have very little meaning. All of the most frequently used words are “stop words”, which we will remove in the next step. Then in the following table we can see how removing stop words changed our most frequently used words drastically.

```
#remove stop words
data("stop_words")
new.election.text <- new.election.text %>%
  anti_join(stop_words)
```

Word	n
trump	16219
election	14134
president	11590
postal	9934
biden	9807
2020	7542

We will do the same thing for the new tweet text and both hashtag text features.

```
#break the text into individual tokens tweet this creates a new data frame
#we did this for each of the four variables
```

```
new.etweets.text <- new.etweets %>%
  dplyr::select(text) %>%
  unnest_tokens(word, text)
#remove stop words
data("stop_words")
new.etweets.text <- new.etweets.text %>%
  anti_join(stop_words)
```

This step is vital to coax bar charts out out the data. Extracting the text and unnesting it allows us to look at the frequency of each individual word. Otherwise when we look at the text, R would not know to look at individual words. Removing stop words is important since we are only looking at meaningful words, which will help us understand what is in the Tweets and hashtags and how they relate to the news of that day.

To determine the most retweeted/quoted user we thought a word cloud would illustrate the information in an eye-catching way. However, word clouds in R require the most preprocessing steps. For the word cloud we referenced STHDA's tutorial on word clouds in R (STHDA, N.D.). The first step is creating a corpus, which similar to unnesting, separates all of the unique words into a vector. The corpus then needs to be turned into a term document matrix which counts the frequency of the words. From there we turn the term document into a matrix, sort the matrix so that the counts are in decreasing order, and finally put it all together in a new data frame. This final data frame is what R will use to create the word cloud.

We needed to load in the packages `SnowballC`, `wordcloud`, and `RColorBrewer`. We first combined the retweeted and quoted user screen names into one vector using the combine command `c()`, and then to reduce the length of the vector we removed any blank rows with the command `-wich(user=="")`. Before removing the blanks for the historical `user` vector it had 138,548 observations and after it only had 79,025. For the new data the `user` vector was 132,674 names long, and was reduced to 78,938 after removing blanks. Creating word clouds on my machine took a lot of memory, so it was very important to reduce the size of the vector as much as possible. After that we had to make a corpus with the `Corpus` command which is similar to unnesting. Then we had to coerce the corpus into a document term matrix, change it to a vector, sort it by word frequency and then put it back into a data frame. Then it is finally ready to be turned into a word cloud.

```
library(SnowballC)
library(wordcloud)
library(RColorBrewer)
#combine retweeted user name and quoted user name for historical data
user=c(new.election$rt.user, new.election$qu.user)
#delete blanks
user=user[-which(user == "")]
docs <- Corpus(VectorSource(user))
dtm <- TermDocumentMatrix(docs)
m <- as.matrix(dtm)
v <- sort(rowSums(m),decreasing=TRUE)
d <- data.frame(word = names(v),freq=v)
#do it again for the new data
```

Note that we did not have to remove any stop words here because the type of text was different. The retweeted and quoted user screen names would be a single word in the vector. These steps were necessary since creating the word cloud required the words be put into a corpus, counted and then put into a new data frame.

3.6 Vsualization

In this section, we discuss our three hypotheses from section 3.4 and how the visualizations we have created relate. We will explain why each visualization was chosen as an appropriate answer to each hypothesis, how it was made, and try to make any inferences about our hypotheses based on these visualizations. Recall that we are using a histogram to visualize the time data, bar charts for the Tweet and hashtag text, and word clouds for most retweeted and quoted Twitter user. In the histogram, we came to the conclusion that the Tweet frequency was actually uniformly distributed, contrary to our hypothesis. The bar charts revealed that in general, our hypothesis was correct: Tweet text reflects current political news, however the hashtags showed us some interesting and unexpected anomalies in the data. The word clouds had Donald Trump's name at the top, but not always number one, in the most retweeted/quoted Twitter user.

A histogram was an good visualization (see Figure 4) to look at the frequency of tweets over time since we could overlay the two days the different data sets were collected and bin the time in 10-minute intervals, which was appropriate for the approximately 3-4-hour time window. Since both data sets were collected over the same time period it gave us the unique opportunity to overlay the histograms for clear visual inspection of both data sets at the same time. In the preprocessing section we converted time into a continuous numeric variable, so it is ready to go into the `hist` command. We start by setting the starting and ending points of the time window with the first and last time points in the newer data set since it covers a longer interval. Then we create two histograms and save them under `histhist` for the historical data and `newhist` for the new data. We break the time interval into approximately 10 minute intervals. Finally we can use the plot function to set up the historical data histogram and add the new data histogram. We set the colors to be transparent so both histograms were visible in the same space. We also added a legend for clarity.

Again, each bar represents the number of tweets in the ten minute time range. The horizontal axis gives time in UTC and was used using the conversion process described in step 2 of preprocessing. The vertical axis tells the Tweet frequency. The historical data is shown in the transparent red, and the new data is shown in transparent blue. We originally hypothesized that Friday morning (historical data) would be a more active Tweet time, and Saturday afternoon (new data) would be more active in the afternoon. Without a statistical test like a goodness-of-fit or permutation test, we can not draw any hard conclusions, but visual inspection we would lean towards our hypothesis being wrong. Both charts are remarkably uniform in distribution. There are no very large spikes or trends in Tweet frequency on these days over these times. While we did not make this hypothesis, it is worth noting that the historical data was a consistently more busy election Tweet time.

```r
#set lower and upper bounds on the x-axis
#the new data was taken over a longer period of time
a=min(new.etweets$time.created)
b=max(new.etweets$time.created)
#create a histogram of the historical data
histhist<-hist(as.numeric(new.election$time.created),
     xlim = c(a,b),
     ylim = c(0,4000),
     breaks = 22, #approximately every 10 min
     xaxt = "n",
     plot = FALSE)
#create a histogram of the new data
newhist<-hist(as.numeric(new.etweets$time.created),
     xlim = c(a,b),
     ylim = c(0,4000),
     breaks = 22 , #approximately every 10 min
     xaxt = "n",
     plot = FALSE)
#plot both histograms together
plot(histhist,
     col  = adjustcolor('red', alpha=0.3),
     main="Tweet frequency over time",
     xlab = "Time in UTC",
     ylab = "Tweet Frequency",
     xlim = c(a,b),
     ylim = c(0,4000),
     xaxt = "n")
plot(newhist,
     col  = adjustcolor('blue', alpha=0.3),
     xlim = c(a,b),
     ylim = c(0,4000),
     xaxt = "n",
     add=TRUE)
axis(side=1, at=c(a,a+((b-a)/4),a+((b-a)/2),a+(3*(b-a)/4), b), labels=c("15:15","16:08","17:01","17:50"
legend(.75,4000,legend=c("historical data","new data"),
       col=c(adjustcolor('red', alpha=0.3),adjustcolor('blue', alpha=0.3)),
       lty=1, lwd=10,cex=.75)
```

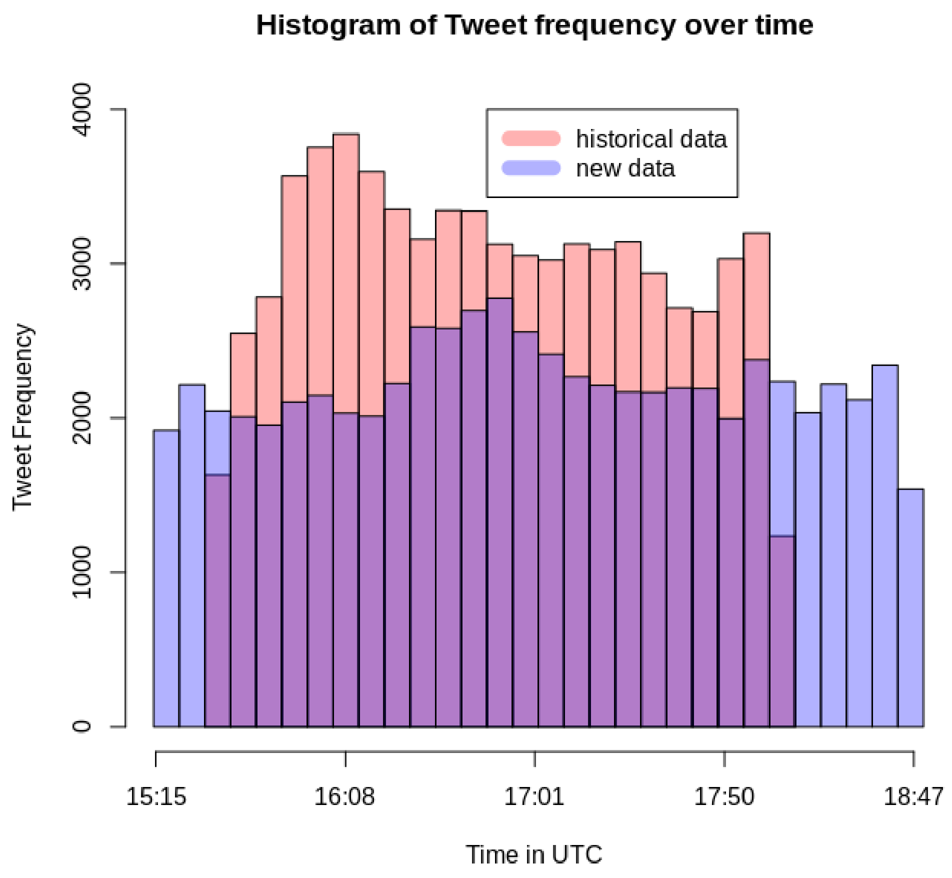


Figure 4: Histogram of Tweet frequency over 10 minute intervals drom 15:15 to 18:47 UTC

Our next visualization (see Figure 5) hoped to answer whether or not Tweet and hashtag content about the election changed. We used guidence from *Text Mining with R* to create the following four bar charts (Silge & Robinson, 2020). A bar chart is good for visualizing frequencies of different categories, so it would work well in seeing what the top 15 words in Tweet and hashtag texts were. Bar charts not only tell us which words appear more, but how much more in comparison with other words. This nice feature of bar charts is especially useful when we look at the hashtag text. This visualization technique takes the cleaned, unnested word data, sorts, and pulls out the top 15 words, then uses `ggplot` to plot the bar chart (Wickham, et.al. 2020). Each bar corresponds to a word given on the vertical axis and frequency given on the horizontal axis.

Note that "trump", "biden", "president", "2020", and "election" are at the top, which makes sense since they were our search keywords, so if we were to go back and clean the data again, we might want to remove those common words with the stop words. As far as the structure of the Tweet text data in general shows that the top two to three words hold a significant lead over the other top 12 words, then the frequency tapers off a little bit more slowly. This is a big diffrence between the Tweet text and the hashtag text. In the hastag text, the top word is used much more than even the second word, then quickly tapers off so that all following words do not have such a stark difference in frequency.

In the historical data, we see that the "postal service" and "clinesmith" were hot topics at the time. A quick web search shows that on August 14th 2020 one of the the top election headlines was about Trump sabotaging the USPS to effect the 2020 Presidential election outcome, which was very clearly reflected in the bar charts with "postal" and "savetheusps" towards the top (WinCalendar, n.d.).

In the new data, we see that "atlantic" and "communist" were new top words, with "joe" likely referring to Joe Biden. Interestingly, clarity on the "joe" and "communist" will come when we look at the hashtags from that data next. The hashtags in the historical data mostly reflect what we saw in the text bar chart. The hashtags observed in the new data has a very interesting feature: `beijingbiden` absolutely dominates. Keep in mind the scale when comparing the two hashtag bar charts; `beijingbiden` is hashtagged over 2000 times, whereas in the historical data, the top hastag `breaking` does not even make it to 1000. Furthermore `beijingbiden` is hashtagged more than 4 times more than the next most popular hashtag: `texas`, and only a few ranks lower `chinawantsbiden` appears. This explains the `joe` and `commuinist` that appeared in the Tweet text. The context being the conception that Joe Biden is a communist that is "soft on China". The bar charts show a change in Tweet text and hashtag text between the historical and new data, backing the hypothesis made in 3.4. A search of September 5, 2020 brings up top headlines about Trump insulting military veterans which was published in *The Atlantic*, and between the Tweet text and hashtag text, indications of this headline are seen with `Atlantic` and `trumphatesvets`, but are greatly overshadowed by words about Biden and China, which do not appear in any major headlines of that day (WinCalendar, n.d.).

While we might be tempted to dismiss "beijingbiden" as an outlier, we could argue that this anomaly is an important attribute of the data to be studied further. Who is making these Tweets about "beijingbiden" and why? A website Beijing Biden comes up as a top web search shows that a company called America First Action, Inc. has taken the time to put together an website about Biden's alleged relationship with China and his communist ties. It is quite the feat for them to produce such a strong presence in the new Twitter data set. Attributes like these are important in catching fake Tweets and fake Twitter accounts (Timberg & Dwoskin, 2018).

Based on the visualization, I would say that in general, we do have a reflection on the keywords from current events and news at the time. However, there are cases where either real or fake Twitter accounts are able to overpower those keyowrds and Tweets at the time. These attributes tell us quite a bit about how social media can be manipulated by groups to influence what words appear most frequently. If we took a closer look at who created these Tweets and who retweets them, we could learn quite a bit about what groups are effected by such social media influences.

```r
library(ggplot2)
# plot the top 15 words
#this same code was used for each of the four vectors we extracted in step 5 of preprocessing
new.election.text %>%
  count(word, sort = TRUE) %>%
  top_n(15) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(x = word, y = n)) +
  geom_col() +
  xlab(NULL) +
  coord_flip() +
  labs(x = "Unique words",
       y = "Count",
       title = "Count of unique words found in novel tweet text")
```

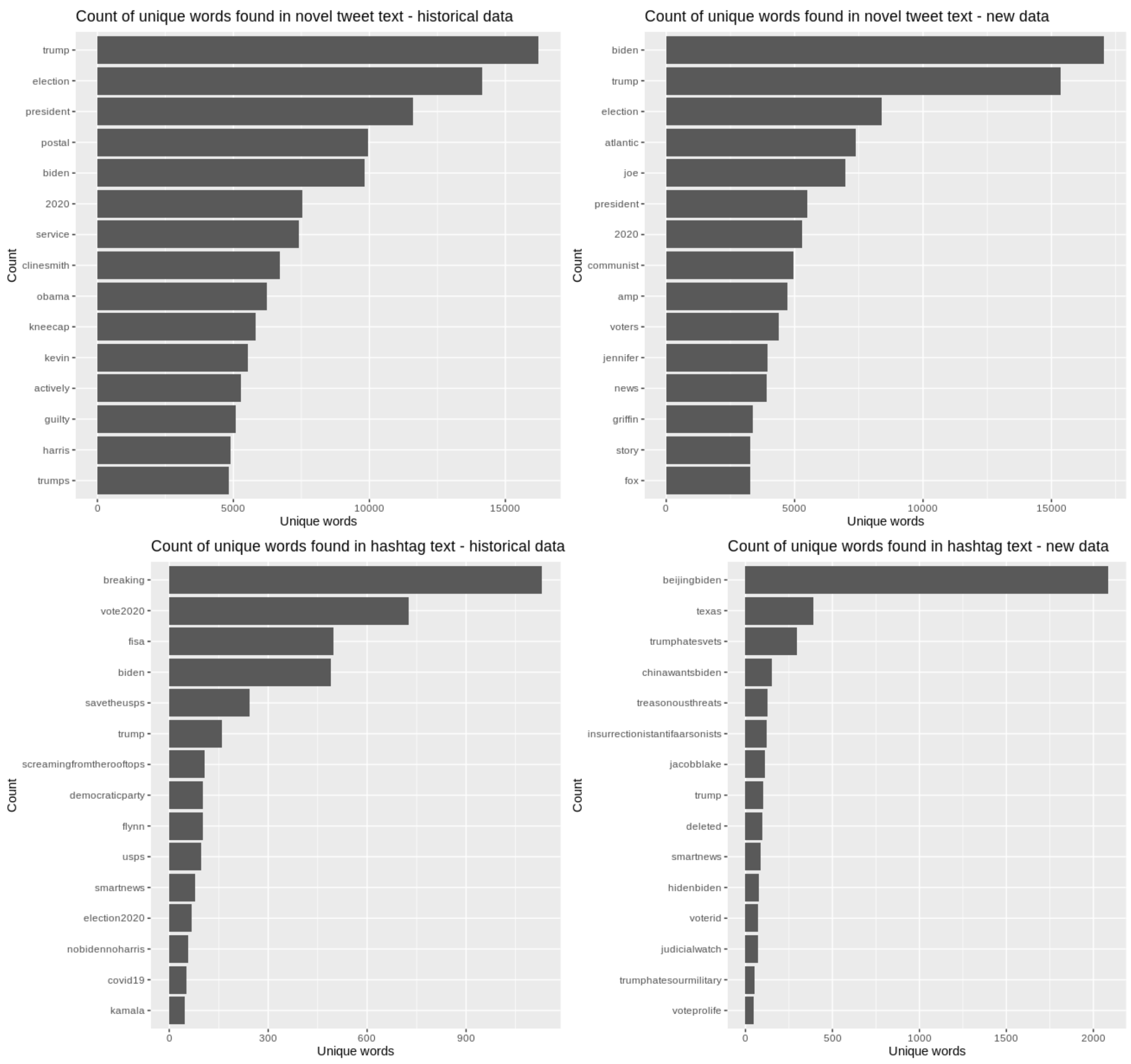


Figure 5: Barcharts for top 15 words in Tweet and hashtag text

Our last visualization technique (see Figure 6) is the most visually exciting, full of color and varying text sizes which cohesively and quickly gives broad information. Since word clouds are great at conveying information quickly, we thought it was the perfect visualization for the top retweeted/quoted user. We would immediately see if Donald Trump was number one based on the color and size of his Twitter user name. If his name wasn't the biggest and boldest, the size and color would still tell us whether or not he was towards the top in the most retweeted and quoted Twitter users.

We created a wordcloud of who gets retweeted or quoted the most and relied heavily on STHDA's tutorial on word clouds in R to get the pleasing results (STHDA, n.d.). The packages we needed were `wordcloud` (Fellows, 2018) to create the wordcloud, and `RColorBrewer` (Neuwirth, 2014) to make the differenct colors. Most of the work to create the word cloud was done in preprocessing, and all that was left to do was decide on colors, sizes, and how many words we wanted in the word cloud. We set a seed so the color palette and layout of the word clouds would be reproducible, then took the cleaned data frame from section 3.5 and put it into the `wordcloud` command. We set the max words to 100 since it output enough words to make the cloud full but not so many that it was too visually noisy. We also used the recommended color palette from STHDA. Each word in the word cloud is a user in the top 100 most retweeted or quoted users on that particular day. The size and color of the word relative to other words ranks it higher or lower; the bigger the word, the more it appeared in the data. In our word clouds, the largest words appear the most frequently and have the color grey. The first word cloud shows the top 100 retweeted/quoted users from the historical data, and the second word cloud is the top 100 retweeted/quoted users from the new day.

We can see that "realdonaldtrump", Trump's Twitter user name, is in the top 5 user names, but not number one in the historical data set. However, in the new data we see that "realdonaldtrump" is retweeted significantly more than nearly any of the other top 100 users. Since the historical data was only taken over a three hour period we would not automatically discard the hypothesis that Trump is the most retweeted/quoted Twitter user, but we do see that at any given point he could be temporarily succeeded by another user. While our findings didn't show Trump as the top retweeted/quoted user of the 2020 election at all times, our word clouds suggest that over time he likely maintains a position at the top. Similar to our other hypotheses and conclusions, we would want to collect more data and apply a statistical test of significance to justify our conclusions.

```r
library(wordcloud)
library(RColorBrewer)
set.seed(1234)
 wordcloud(words = d$word, freq = d$freq, min.freq = 1,
                   max.words=100, random.order=FALSE, rot.per=0.35,
                   colors=brewer.pal(8, "Dark2"))
 wordcloud(words = ed$word, freq = ed$freq, min.freq = 1,
          max.words=100, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))
```

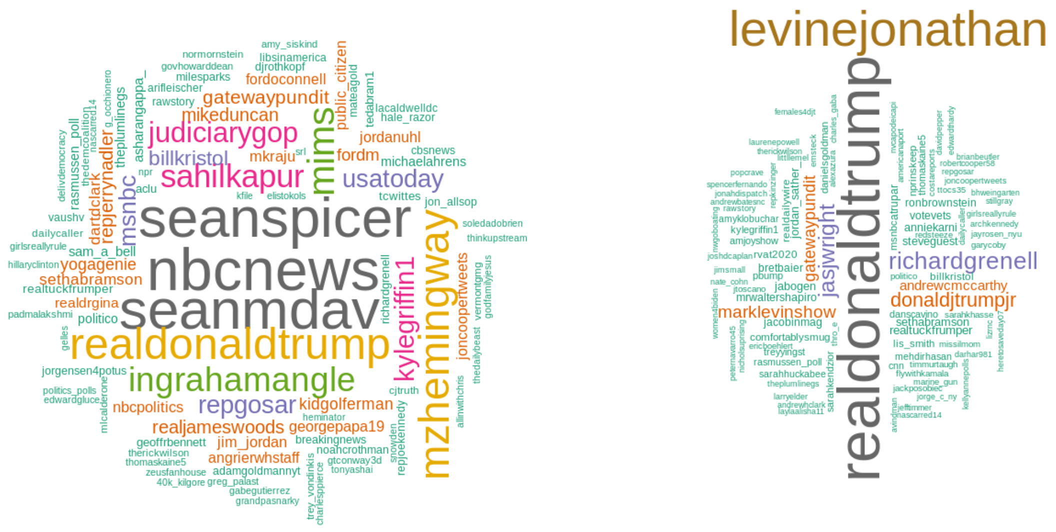


Figure 6: Retweeted/quoted user screen name word cloud, historical data on the left, new data on the right

3.7 Conclusion

In section 3.4 we introduced hypotheses: the distribution of Tweet frequency varied over time; election news would be reflected in Tweet text and hashtag text; and that Donald Trump is the most retweeted and quoted user on Twitter (within our search parameters). In section 3.5 we preprocessed the data and got it ready for visualization by removing all non-English observations and extracting time as a numeric variable. Next, we cleaned the text data by removing noise like emojis and capitalization so the important information - the words themselves - would be represented accurately and uniformly. Finally we moved the cleaned and desired features and observations to a new data frame. From there we were able to put the data in the form appropriate for each visualization: a histogram for the time and Tweet frequency hypothesis, bar charts for the Tweet and hashtag text, and word clouds to see if Trump was the most retweeted/quoted Twitter user. The visualizations yielded mixed results for each hypothesis, but all offered insight into whether or not the hypotheses were true, and prompted further questions.

We found that the Tweet frequency over two different days, but over the same time interval was remarkably uniform. Ideally, we would collect information about Tweet frequency over different days and time periods to get a full picture of how Tweet frequency is distributed. Such information is important for individuals, companies, and organizations so they know the best time to get information to the public. Tweets still make their way to Facebook, Instagram and other social media platforms in the form of screenshots and copy/paste text - dispite never having been on Twitter, I still see Tweets make their way into my small corner of the internet.

In the bar charts, we saw the political news of that day reflected in the word frequency, and we can say that our hypothesis could . Trump's USPS scandal appeared to be the number one election news in the historical data, while Trump's veteran comments were reflected in the new data. It is interesting to note that Trump is the most talked-about candidate, whether that's a good thing or not may be revealed during the election in November. Even though the top words about Trump were not positive based on our results, will we see that any publicity is good publicity? Will this help him win a second term? This is a study that could certainly be extended, and even looking back at previous elections - does name frequency predict the winner? The bar charts in many ways prompted more questions than answered.

The hashtag bar charts yeilded arguably the most interesting results. Seeing "beijingbiden" hashtagged more than double over any other word in both the historical and new data was unanticipated. The influence a seemingly small group (I had never heard of America First Action, Inc. until this project) could have on a global public forum can not be denied and should not be ignored. In section 3.4 we noted that most Tweets only came from a small group of Twitter users, and a large group of Twitter users get their news from Tweets. This means that many people's views are influenced by a small, and likely non-representative sample of the world population. Users should be aware of these findings so that hopefully they keep that in mind when digesting information from Twitter. The "news" they get is likely biased, and provides only a narrow view of the world at large.

The word clouds were good at immediately telling multiple stories at once. Who was retweeted or quoted the most at that time? In the historical data we saw that while Trump was not number one, he still heald a place near the top. In the new data Trump outshined everyone else as the most dominant voice in Twitter with more retweets and quotes than any other user - and by a significant amount. Again, this is such an important idea since so many users get news from the site, so when the President Tweets something - true or false - it appears to spread more than any other information on Twitter (in the political realm). Like the small groups influencing Twitter trends, this has important social consequences. Public opinion swayed by potentially false information has the potential to effect everyone in the country in a dangerous way. Again, users should be aware of these facts in order to formulate fact-based and balanced opinions and views of the world.

Overall, to really draw strong conclusions about our three hypotheses, we would want to collect more data over different days and time periods and apply more rigorus statistical tests and methods. This does not mean the information we inferred from our visualizations is useless, on the contrary, it gave insight into other hypotheses that are important to answer. One of the most intriguing new questions about how small groups can influence the information on Twitter, and thus on the internet as well. Socially and politically, these are important questions to answer since it effects the way people vote, and how real policy effects real change, for better or worse.

References

Bouchet-Valat, M., (2020, April 1).*SnowballC: Snowball Stemmers Based on the C 'libstemmer' UTF-8 Library.* Retrieved from https://cran.r-project.org/web/packages/SnowballC/index.html

Clement, J. (2020). Leading countries based on number of Twitter users as of July 2020. *Statista.* https://www.statista.com/statistics/242606/number-of-active-twitter-users-in-selected-countries/

De Queiroz, G., Fay, C., Hvitfeldt, E., Keyes O., Misra, K., Mastny, T., Erickson, J., Robinson, D., Silge, J. (2020, July 11). *tidytext: Text Mining using 'dplyr', 'ggplot2', and Other Tidy Tools.* Retrieved from https://cran.r-project.org/web/packages/tidytext/index.html

Fellows, I., (2018, August 24). *wordcloud: Word Clouds.* Retrieved from https://cran.r-project.org/web/packages/wordcloud/index.html

Feinerer, I., Hornik, K. (2019, December 12). *tm: Text Mining Package.* Retrieved from https://cran.r-project.org/web/packages/tm/index.html

Hughes, A., & Wojcik, S. (2019, August 2). 10 facts about Americans and Twitter. *Pew Research Center.* https://pewrsr.ch/2M9oPsZ

James, D, Hornik, K, Grothendieck, G. (2020 August 18). *chron: Chronological Objects which can Handle Dates and Times.* Retrieved from https://cran.r-project.org/web/packages/chron/index.html

Jeong, S. (2018, March 13). Trump fires secretary of state via tweet. *The Verge.* https://www.theverge.com/2018/3/13/17113950/trump-state-department-rex-tillerson-fired-tweet-twitter

Lerer, L. (2019, April 11). Twitter is a big deal in politics. That doesn't make it right. *The New York Times.* https://www.nytimes.com/2019/04/11/us/politics/on-politics-twitter-democrats.html

Neuwirth, E., (2014, December 7). *RColorBrewer: ColorBrewer Palettes.* Retrieved from https://cran.r-project.org/web/packages/RColorBrewer/index.html

Timberg, C. & Dwoskin, E. (2018, July 6). Twitter is sweeping out fake accounts like never before, putting user growth at risk. *The Washington Post.* https://www.washingtonpost.com/technology/2018/07/06/twitter-is-sweeping-out-fake-accounts-like-never-before-putting-user-growth-risk/

Text mining and word cloud fundamentals in R: 5 simple steps you should know. (n.d.). *Statistical Tools for High-Throughput Data Analysis (STHDA).* http://www.sthda.com/english/wiki/text-mining-and-word-cloud-fundamentals-in-r-5-simple-steps-you-should-know

Silge, J., & Robinson, D. (2020). *The tidy text format. In Text mining with R* [ebook edition]. O'Reilly. https://www.tidytextmining.com/tidytext.html

Wickham, H. (2020, August 18). *dplyr: A Grammar of Data Manipulation.* Retrieved from https://cran.r-project.org/web/packages/dplyr/index.html

Wickham, H., Chang, W., Henry, L., Lin Pedersen, T., Takahashi, K., Wilke, C., Woo, K., Yutani, H., Dunnington, D. (2020, june 19). *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics.* Retrieved from https://cran.r-project.org/web/packages/ggplot2/index.html

Wickham, H/ (2019, February 10). *stringr: Simple, Consistent Wrappers for Common String Operations.* Retrieved from https://cran.r-project.org/web/packages/stringr/index.html

Wickham, H. (2019, November 21). *tidyverse: Easily Install and Load the Tidyverse.* Retrieved from https://cran.r-project.org/web/packages/tidyverse/index.html

America First Action, Inc. (n.d.). *Beijing Biden.* September 16, 2020, https://beijingbiden.com/

Twitter. (n.d.) *Data dictionary. Developer.* https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/overview/intro-to-tweet-json

Saturday, September 5, 2020. (n.d.). *WinCalendar.* https://www.wincalendar.com/Calendar/Date/September-5-2020

Friday, August 14, 2020. (n.d.). *WinCalendar.* https://www.wincalendar.com/Calendar/Date/August-14-2020