You may have heard that two researchers at Clemson University, Russia , analyzed almost 3 millions tweets from the Internet Research Agency (IRA) – a “Russian troll factory”. In partnership with FiveThirtyEight, they made all of their data available as available here https://github.com/fivethirtyeight/russian-troll-tweets/. So of course, I had to read the files into R, which I was able to do with this code:

files <- c("IRAhandle\_tweets\_1.csv",  
 "IRAhandle\_tweets\_2.csv",  
 "IRAhandle\_tweets\_3.csv",  
 "IRAhandle\_tweets\_4.csv",  
 "IRAhandle\_tweets\_5.csv",  
 "IRAhandle\_tweets\_6.csv",  
 "IRAhandle\_tweets\_7.csv",  
 "IRAhandle\_tweets\_8.csv",  
 "IRAhandle\_tweets\_9.csv")  
my\_files <- paste0("~/Downloads/russian-troll-tweets-master/",files)  
  
each\_file <- function(file) {  
 tweet <- read\_csv(file) }  
  
library(tidyverse)

tweet\_data <- NULL  
for (file in my\_files) {  
 temp <- each\_file(file)  
 temp$id <- sub(".csv", "", file)  
 tweet\_data <- rbind(tweet\_data, temp)  
}

Note that this is a large file, with 2,973,371 observations of 16 variables. Let’s do some cleaning of this dataset first. The researchers, Darren Linvill and Patrick Warren, identified 5 majors types of trolls:

* Right Troll: These Trump-supporting trolls voiced right-leaning, populist messages, but “rarely broadcast traditionally important Republican themes, such as taxes, abortion, and regulation, but often sent divisive messages about mainstream and moderate Republicans…They routinely denigrated the Democratic Party, e.g. @LeroyLovesUSA, January 20, 2017, “#ThanksObama We’re FINALLY evicting Obama. Now Donald Trump will bring back jobs for the lazy ass Obamacare recipients,” the authors wrote.
* Left Troll: These trolls mainly supported Bernie Sanders, derided mainstream Democrats, and focused heavily on racial identity, in addition to sexual and religious identity. The tweets were “clearly trying to divide the Democratic Party and lower voter turnout,” the authors told FiveThirtyEight.
* News Feed: A bit more mysterious, news feed trolls mostly posed as local news aggregators who linked to legitimate news sources. Some, however, “tweeted about global issues, often with a pro-Russia perspective.”
* Hashtag Gamer: Gamer trolls used hashtag games—a popular call/response form of tweeting—to drum up interaction from other users. Some tweets were benign, but many “were overtly political, e.g. @LoraGreeen, July 11, 2015, “#WasteAMillionIn3Words Donate to #Hillary.”
* Fearmonger: These trolls, who were least prevalent in the dataset, spread completely fake news stories, for instance “that salmonella-contaminated turkeys were produced by Koch Foods, a U.S. poultry producer, near the 2015 Thanksgiving holiday.”

But a quick table of the results of the variable, account\_category, shows 8 in the dataset.

table(tweet\_data$account\_category)

##   
## Commercial Fearmonger HashtagGamer LeftTroll NewsFeed   
## 122582 11140 241827 427811 599294   
## NonEnglish RightTroll Unknown   
## 837725 719087 13905

The additional three are Commercial, Non-English, and Unknown. At the very least, we should drop the Non-English tweets, since those use Russian characters and any analysis I do will assume data are in English. I’m also going to keep only a few key variables. Then I’m going to clean up this dataset to remove links, because I don’t need those for my analysis – I certainly wouldn’t want to follow them to their destination. If I want to free up some memory, I can then remove the large dataset.

reduced <- tweet\_data %>%  
 select(author,content,publish\_date,account\_category) %>%  
 filter(account\_category != "NonEnglish")  
  
library(qdapRegex)

##   
## Attaching package: 'qdapRegex'

reduced$content <- rm\_url(reduced$content)  
  
rm(tweet\_data)

Now we have a dataset of 2,135,646 observations of 4 variables. I’m planning on doing some analysis on my own of this dataset – and will of course share what I find – but for now, I thought I’d repeat a technique I’ve covered on this blog and demonstrate a new one.

library(tidytext)  
  
tweetwords <- reduced %>%  
 unnest\_tokens(word, content) %>%  
 anti\_join(stop\_words)

## Joining, by = "word"

wordcounts <- tweetwords %>%  
 count(account\_category, word, sort = TRUE) %>%  
 ungroup()  
  
head(wordcounts)

## # A tibble: 6 x 3  
## account\_category word n  
##   
## 1 NewsFeed news 124586  
## 2 RightTroll trump 95794  
## 3 RightTroll rt 86970  
## 4 NewsFeed sports 47793  
## 5 Commercial workout 42395  
## 6 NewsFeed politics 38204

Reference code for text analysis

These tools are useful when you have multiple documents you're analyzing, such as interview text from different people or books by the same author. For my demonstration today, I'll be using (what else?) song lyrics, this time from Florence + the Machine (one of my all-time favorites), who just dropped a new album, *High as Hope*. So let's get started by pulling in those lyrics.

**library**(geniusR)

high\_as\_hope <- **genius\_album**(artist = "Florence the Machine", album = "High as Hope")

*## Joining, by = c("track\_title", "track\_n", "track\_url")*

**library**(tidyverse)

**library**(tidytext)

tidy\_hope <- high\_as\_hope %>%

**unnest\_tokens**(word,lyric) %>%

**anti\_join**(stop\_words)

*## Joining, by = "word"*

**head**(tidy\_hope)

## # A tibble: 6 x 4

## track\_title track\_n line word

## <chr> <int> <int> <chr>

## 1 June 1 1 started

## 2 June 1 1 crack

## 3 June 1 2 woke

## 4 June 1 2 chicago

## 5 June 1 2 sky

## 6 June 1 2 black

Now we have a tidy dataset with stop words removed. Before we go any farther, let's talk about the tools we're going to apply. Often, when we analyze text, we want to try to discover what different documents are about - what are their topics or themes? One way to do that is to look at common words used in a document, which can tell us something about the document's theme. An overall measure of how often a term comes up in a particular document is term frequency (TF).  
  
Removing stop words is an important step before looking at TF, because otherwise, the high frequency words wouldn't be very meaningful - they'd be words that fill every sentence, like "the" or "a." But there still might be many common words that don't get weeded out by our stop words anti-join. And it's often the less frequently used words that tell us something about the meaning of a document. This is where inverse document frequency (IDF) comes in; it takes into account how common a word is across a set of documents, and gives higher weight to words that are infrequent across a set of documents and lower weight to common words. This means that a word used a great deal in one song but very little in the other songs will have a higher IDF.  
  
We can use these two values at the same time, by multiplying them together to form TF-IDF, which tells us the frequency of the term in a document adjusted for how common it is across a set of documents. And thanks to the tidytext package, these values can be automatically calculated for us with the bind\_tf\_idf function. First, we need to reformat our data a bit, by counting use of each word by song. We do this by referencing the track\_title variable in our count function, which tells R to group by this variable, followed by what we want R to count (the variable called word).

song\_words <- tidy\_hope %>%

**count**(track\_title, word, sort = TRUE) %>%

**ungroup**()

The bind\_tf\_idf function needs 3 arguments: word (or whatever we called the variable containing our words), the document indicator (in this case, track\_title), and the word counts by document (n).

song\_words <- song\_words %>%

**bind\_tf\_idf**(word, track\_title, n) %>%

**arrange**(**desc**(tf\_idf))

**head**(song\_words)

## # A tibble: 6 x 6

## track\_title word n tf idf tf\_idf

## <chr> <chr> <int> <dbl> <dbl> <dbl>

## 1 Hunger hunger 25 0.236 2.30 0.543

## 2 Grace grace 16 0.216 2.30 0.498

## 3 The End of Love wash 18 0.209 2.30 0.482

## 4 Hunger ooh 20 0.189 2.30 0.434

## 5 Patricia wonderful 10 0.125 2.30 0.288

## 6 100 Years hundred 12 0.106 2.30 0.245

Some of the results are unsurprising - "hunger" is far more common in the track called "Hunger" than any other track, "grace" is more common in "Grace", and "hundred" is more common in "100 Years". But let's explore the different words by plotting the highest tf-idf for each track. To keep the plot from getting ridiculously large, I'll just ask for the top 5 for each of the 10 tracks.

song\_words %>%

**mutate**(word = **factor**(word, levels = **rev**(**unique**(word)))) %>%

**group\_by**(track\_title) %>%

**top\_n**(5) %>%

**ungroup**() %>%

**ggplot**(**aes**(word, tf\_idf, fill = track\_title)) +

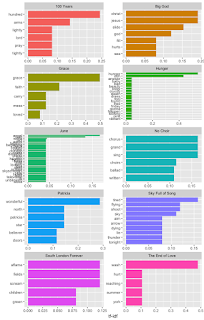
**geom\_col**(show.legend = FALSE) +

**labs**(x = **NULL**, y = "tf-idf") +

**facet\_wrap**(~track\_title, ncol = 2, scales = "free") +

**coord\_flip**()

*## Selecting by tf\_idf*

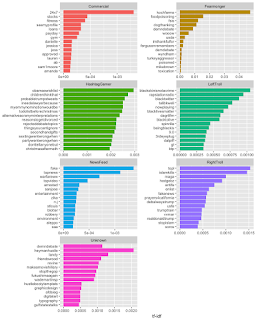
[](https://4.bp.blogspot.com/-O1JpZjrFjjI/W1yqbsavGVI/AAAAAAAANVk/CDROgMfohz4YNNJ7oHtj0PyiMu3MyHAVgCLcBGAs/s1600/Rplot.png)

Some tracks have more than 5 words listed, because of ties, but this plot helps us to look for commonalities and differences across the tracks. There is a strong religious theme across many of the tracks, with concepts like "pray", "god", "grace", and "angel" coming up in many tracks. The song "Patricia" uses many positively-valenced words like "wonderful" and "believer". "No Choir" references music-themed words. And "Sky Full of Song" references things that fly (like "arrow") and things in the sky (like "thunder").

First, I’ll conduct a TF-IDF analysis of the dataset.

tweet\_tfidf <- wordcounts %>%  
 bind\_tf\_idf(word, account\_category, n) %>%  
 arrange(desc(tf\_idf))  
  
tweet\_tfidf %>%  
 mutate(word = factor(word, levels = rev(unique(word)))) %>%  
 group\_by(account\_category) %>%  
 top\_n(15) %>%  
 ungroup() %>%  
 ggplot(aes(word, tf\_idf, fill = account\_category)) +  
 geom\_col(show.legend = FALSE) +  
 labs(x = NULL, y = "tf-idf") +  
 facet\_wrap(~account\_category, ncol = 2, scales = "free") +  
 coord\_flip()

## Selecting by tf\_idf

[](https://i2.wp.com/1.bp.blogspot.com/-SbdpFBt5MZk/W3Bd-Wvi3-I/AAAAAAAANaY/PBSzebwUl-cKMjQDOqHAKcovfHRpvdNlwCLcBGAs/s1600/tfidf_tweets1.png?ssl=1)

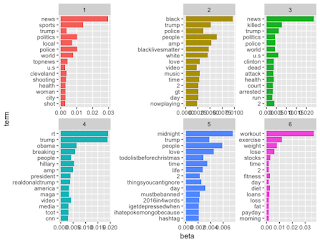
But another method of examining terms and topics in a set of documents is [Latent Dirichlet Allocation (LDA)](http://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf), which can be conducted using the R package, topicmodels. The only issue is that LDA requires a [document term matrix](https://en.wikipedia.org/wiki/Document-term_matrix). But we can easily convert our wordcounts dataset into a DTM with the cast\_dtm function from tidytext. Then we run our LDA with topicmodels. Note that LDA is a random technique, so we set a random number seed, and we specify how many topics we want the LDA to extract (k). Since there are 6 account types (plus 1 unknown), I’m going to try having it extract 6 topics. We can see how well they line up with the account types.

Library(topicmodels)

tweets\_dtm <- wordcounts %>%  
 cast\_dtm(account\_category, word, n)  
  
library(topicmodels)  
tweets\_lda <- LDA(tweets\_dtm, k = 6, control = list(seed = 42))  
tweet\_topics <- tidy(tweets\_lda, matrix = "beta")

Now we can pull out the top terms from this analysis, and plot them to see how they lined up.

top\_terms <- tweet\_topics %>%  
 group\_by(topic) %>%  
 top\_n(15, beta) %>%  
 ungroup() %>%  
 arrange(topic, -beta)  
  
top\_terms %>%  
 mutate(term = reorder(term, beta)) %>%  
 ggplot(aes(term, beta, fill = factor(topic))) +  
 geom\_col(show.legend = FALSE) +  
 facet\_wrap(~topic, scales = "free") +  
 coord\_flip()

[](https://i0.wp.com/4.bp.blogspot.com/-JAkFUPH8zic/W3BeKIg5ubI/AAAAAAAANac/ztZ8NY_jneUuH7zhoiCu49FdFqrKbZbLgCLcBGAs/s1600/LDA_tweets.png?ssl=1)