

An Analysis of the COVID-19 Pandemic's Affect on Word Sentiment Using the R Package *metamessengr*

Alexander Robertson, Tobius Koch

BIS 620

<https://github.com/albobson/metamessengr>

Introduction

The words that we use in conversation are a reflection of our feelings and attitudes towards a given subject. Sentiment analysis, also known as opinion mining, is a field of study that seeks to understand what our word choices imply about our affective states. Sentiment analysis allows us to quantitatively assess qualitative text information for use in spatial visualization and hypothesis testing.

There are many ways to perform sentiment analysis, the most straightforward being analyzing the sentiments associated with individual words of a text. This form of analysis implies that the overall sentiment of a set of words is the sum of their individual associated sentiments. There are many lexicons of sentiment associated with individual words, some of which are binary (positive/negative), others are explicit to the associated emotion (anger/sadness/etc.). For this project, we used the AFFIN lexicon developed by Finn Årup Nielsen, which ranks each word in the English language on a scale of -5 to +5¹.

In 2011, Facebook released its messaging application, Facebook Messenger, as a standalone application². As of 2017, Messenger had over 1.6 million users³, making it one of the most popular messaging applications in the world. Facebook allows users to download their own information⁴, which allows individuals to analyze their own messaging history. The average Facebook user would have a difficult time understanding and analyzing this data, as cleaning and sorting it is not a trivial task. In this paper, we describe an R package that we developed for this purpose, *metamessengr*, and show how it can be used to visualize and statistically test texting data. Given that Messenger is a private application, we assume that sentiments may be expressed more freely in conversations. We used the functions we developed to test whether the sentiment of our messages in Facebook group chats had a significant decrease after the COVID-19 pandemic began in the United States of America.

Materials and Methods

Metamessengr

Data Importation

Messenger data can be downloaded in either HTML or JSON file formats. While difficult for the average user to parse, JSON file formats are much easier for data processing. The function `mess_selection()` was created to select all of the sender information, message content, and timestamp of the message for each message downloaded. The function extracts the filenames for each conversation downloaded. Those file names are then iterated over using a for loop and the `fromJSON()` function from the *jsonlite* package to extract the total information from each message. The *dplyr*

and *purrr* packages are then used to select the desired information. This information is then stored in a new dataframe, which the user can assign to a new object.

Data Processing

The results of the data importation can be quite messy and need to be cleaned before data analysis. This can either be done in two steps, using the `clean_mess_text()` and `clean_mess_time()` functions, or in one step using the `clean_mess_all()` function. These have been separated into three functions to give the user more control over the data cleaning process. We recommend the use of the `clean_mess_all()` function for most instances.

The `clean_mess_all()` function makes the data ready for analysis in one step for the user. Stop words are words that are so commonly used, they offer very little information to the text (for example: “the”, “a”, “that”, etc.). A dataset of these words was acquired from the *tidytext* package, and was removed from the content column. The `custom_clean` argument was created which allows a user to input a character vector of words that they would additionally like to be removed from the analysis (“https” for example). All words are then formatted to be lowercase. A new column is added to the data frame called “convo”, which pastes the name of the sender and the name of the conversation, allowing one to elucidate what one sent to different group chats. Another new column was created called “length”, which records the number of characters in each message. Messenger has a maximum character limit per message of 640 words. Because of this, any messages with a “length” greater than 640 were set as NA, as the associated message was wrong. For use in further analysis, a new column is created at the beginning of the function which preserves the “uncleaned” data.

The `clean_mess_all()` function additionally formats the time in a more usable manner. Using functions from the package *chron* and functions from base R, we converted the timestamp of each message into a date, hour of the day, and time of day that the message was sent.

Data Summaries

Three functions were created to easily give the user general summaries of their newly cleaned data: `sender_sum()`, `group_sum()` and `top_words()`. The `sender_sum()` function breaks down the data by sender and group, and indicates how many messages and the total number of characters each person sent to a specific group. The `group_sum()` function outputs the summary of each group by indicating how many messages and total characters were sent.

The `top_words()` function utilizes the Grady Augmented lexicon⁵, which is a lexicon that contains all words in the English language, as well as proper nouns. This lexicon is used to select just words in the English dictionary. Then the data is analyzed for the number of times each individual in the data set used each word, and the

frequency of that word in the individuals total messaging history. Since this dataset is longer than realistically readable by a human, an additional argument was provided, `num`, which allows a user to indicate how many of the top words for each individual in the dataset are listed.

Sentiment Lexicon Joining

The `mess_sentiment()` function was written to easily associate each word in the dataset with a sentiment. This function utilizes the `unnest_tokens()` and the `get_sentiments()` functions from the *tidytext* package. As stated previously, this analysis was performed using the AFFIN lexicon, which was sourced through the *tidytext* package, but many other lexicons could be associated with this function using the `lexicon` argument. This function creates a new column which is a list of every word from every message, and associates its AFFIN score from -5 to +5.

“Pre” and “Post” Event Association

Since we are interested in seeing the difference in sentiment before and after an event-time, the function `mess_cut_off()` was created. This function adds a column to the dataset which describes the message as pre- or post-event. The event-time is indicated in the argument, `time`, which is set to roughly the first day of the COVID-19 pandemic in the United States of America.

Data Plotting

All data was plotted using the *ggplot* package in R. *Ggplot* is one of the most commonly used graphical tools used for research and easy syntax makes it ideal for analysis.

Statistical Analyses

For our proof of concept, we looked at Tobias’s data and subset the top 12 senders for the final analysis. This was done for two primary reasons; to avoid Facebook preset messages, spam messages, and Facebook marketplace interactions, as well as to focus on closer relationships that have enough data. This additionally ensured there was a personal connection to the user which both parties would be more likely to speak candidly.

Normality of the data was tested by visual inspection of a histogram (Figure 1). Although it is clear that zero is overrepresented in the data, previous work considers social media data normally distributed⁶. This phenomenon is due in large part to the fact that the majority of words used in natural language are considered neutral and receive a score of zero. Given this fact, the data appear normal. (Figure 1)

A two-tailed t-test was initially run to assess whether there was a significant difference in the means between the “pre” and “post” event sentiment scores. The t-test

was run on the whole data set as well as for the user in order to provide more personal insight.

The sender of the message may modify the type of language used when talking to the user and vice versa. In order to account for the sender, a two-tailed ANOVA test was run with sender as a covariate

In order to measure the effect estimate of “pre” to “post” pandemic sentiment we conducted a linear regression on the sentiment using the “pre” and “post” event as our primary independent variable with “pre” as our reference group (Table 2). The ANOVA test found that sender was a statistically significant interaction term and was included in our model as an interaction term with the user as the reference.

Results

The two way T-test comparing “pre” and “post” pandemic sentiment in Facebook messenger data was insignificant ($p = 0.8113$, with a “pre” and “post” mean of 0.12 and 0.113 respectively) and the user specific T-test was also insignificant ($p = 0.06$, with a “pre” and “post” mean of 0.18 and 0.10 respectively). However, the two way ANOVA test with sender as the interaction term was highly significant ($p < 0.001$).

In the linear regression, the unadjusted estimate was insignificant (-0.01 CI(0.02,-0.04) $p = 0.79$). The adjusted crude estimate of sentiment in Facebook messenger data “post” pandemic compared to “pre” with a decrease of -0.09 ($p = 0.03$, CI $(-0.05, -0.013)$) when adjusted for sender and the user as the reference group.

These results indicate that there was in fact a slight decrease in messenger word sentiment post pandemic when controlling for the sender. Given that the unadjusted estimate was null, sender of the message appears to be modifying the effect of sentiment with the user’s. This implies that different message senders will alter the sentiment of their message based on the recipient (the user).

Discussion

This project demonstrates the functionality of our R package, *metamessengr*, and shows how it can be utilized to understand our relationship with language and other individuals. *Metamessengr* makes it easy to select and clean Facebook Messenger data, which can be further mined for additional insights and information. Built into the package are functions which allow a user to see general summaries of their information. Table 1 shows the results of the `sender_sum()` package, which is easy to understand and gives the user insight into their messaging history.

After using the functions to clean the data and add the sentiment scores, one is left with a usable dataset for further analysis. Figure 2 shows a plot of the total message sentiment of each message over time, colored by pre-pandemic and post-pandemic timestamps. Other visualizations can easily be performed using ggplot or other plotting

packages. Figure 3 shows a simple *ggplot* boxplot of the total sentiment expressed by each sender before and after the pandemic. Similarly, Figure 4 is a plot of the individual senders and their estimated sentiment post event as compared to the user. The function `top_words()` was used to generate Figure 5 with simple *ggplot* syntax. Future updates to the package will include functions to create these plots.

Currently, there are two functions in the package which create plots of the data, `plot_senti_time()` (Figure 6), which plots each sender's sentiment over time, and `plot_mess_dens()` (Figure 7) which plots each sender's sentiment density.

More interesting is the tools provided to analyze and plot the data. Utilization of this tool could provide users insight into trends into their mental health by analyzing the language that they use every day in general as well as insights into interpersonal relationships with those they are messaging with. This level of detail is not available in social media posts which are not directed at any one particular person. This tool could tell whether or not the users language is statistically negative with family compared to friends or acquaintances compared to sexual partners. Understanding the sentiment of our language can be an important tool in recognizing potential problems that may be occurring as a result of prolonged negativity which can be a sign associated with depression and anxiety. When developed further with more rigorous statistical methodology, this tool could even one day help alert the user of warning signs that indicate that they are at risk for certain negative sequelae.

While there are many analytical tools to assess publicly available posts on social media, few have looked at personal messages. By its very nature, public posts are generally crafted with forethought, and public perception is taken into consideration when wording the message. This is in contrast to private messaging, where users are likely to be less guarded and be more instinctive with their messaging and thus could provide a more unbiased dataset.

Limitations and Concluding Remarks

In this paper, we described the functionality of our newly developed R package, *metamessngr*, and the ways that these tools could be applied to gain insight into one's own Messenger data. We showed that through using our functions, one could assess the effects that an event has on the sentiment expressed in messages shared on Facebook Messenger. Future improvements and added functionalities will make this tool useful and interesting for any user to gain insight into their own data.

There are a few major limitations and caveats that should be considered with this analysis. In a relatively simple sentiment analysis such as this, words can be easily taken out of context. The form of sentiment analysis we conducted breaks down each word individually without taking into account the previous or subsequent words. This can be particularly problematic with words considered to be profanity. "Ass" is considered to be a highly negative (-4) word using the *afinn* method but is often used in

regular lexicon with a positive connotation meaning such as “bad ass”. This is one of many examples. This may bias our results downward and differentially impact sentiment among users that preferentially utilize less formal use of language. The degree to which it impacts sentiment, particularly on a private messaging platform used predominantly by younger members of the population could warrant further research. As seen by Figure X of the user, 3 of the top 25 words are considered profanity. In the future, we plan on adding functionality which will assess the level of profanity expressed by each user.

Additionally, the analysis that we present only specifies the sender of the message, with the user always being either the recipient or member of a group in which the sender has sent the message too. Our analysis doesn’t distinguish who the user is sending the message too. Distinguishing this relationship could provide more insight into whether the sentiment is modified by who the user is sending the message to. With the tools that we have created, this would certainly be possible, but we did not have time to perform this analysis.

Lastly, by looking just at Facebook Messenger data, we are potentially biasing our data towards younger, more internet savvy individuals. The way that this group talks, especially amongst themselves, is likely not representative of the entire population.

Going forward, we plan to add additional functionality to our package by looking at bigrams (two-word sentiment analysis) as well as other forms of visualization. Due to the constraint of time on this project, we did not have time to delve into using the *testthat* package for our functions (though not for a lack of trying), though we plan on adding such testing in the future. Additionally, we ran into a problem of privacy and anonymity when creating package vignettes and example data. Since the conversations that we have with friends often contain very personal information, it is difficult to create an example dataset for users to play with. Randomly generating a dataset would remove any ability to meaningfully analyze the data, so we are at a loss for what to do on this front. Please let us know if you have any ideas or suggestions, as we plan to continue to improve this package.

Sources

1. Nielsen, F. Å. A new ANEW: Evaluation of a word list for sentiment analysis in microblogs.

Proceedings of the ESWC2011 Workshop on ‘Making Sense of Microposts’: Big things come in small packages 718 in CEUR Workshop Proceedings 93-98. (2011).

2. Facebook Launches Standalone iPhone/Android Messenger App (And It’s Beluga).

TechCrunch

<https://social.techcrunch.com/2011/08/09/facebook-launches-standalone-mobile-messenger-app-and-it%E2%80%99s-beluga/>.

3. Facebook Messenger has 1.2 billion users and is now twice the size of Instagram - Vox.

<https://www.vox.com/2017/4/12/15263312/facebook-messenger-app-billion-users>.

4. How do I download a copy of my information on Facebook? | Facebook Help Center.

<https://www.facebook.com/help/212802592074644>.

5. GradyAugmented function - RDocumentation.

<https://www.rdocumentation.org/packages/qdapDictionaries/versions/1.0.7/topics/GradyAugmented>.

6. Karn, A., Shrestha, A., Pudasaini, A., Mahara, B. & Jaiswal, A. Statistic-Based Sentiment Analysis of Social Media Data. **2**, 28–32 (2018).

Figures

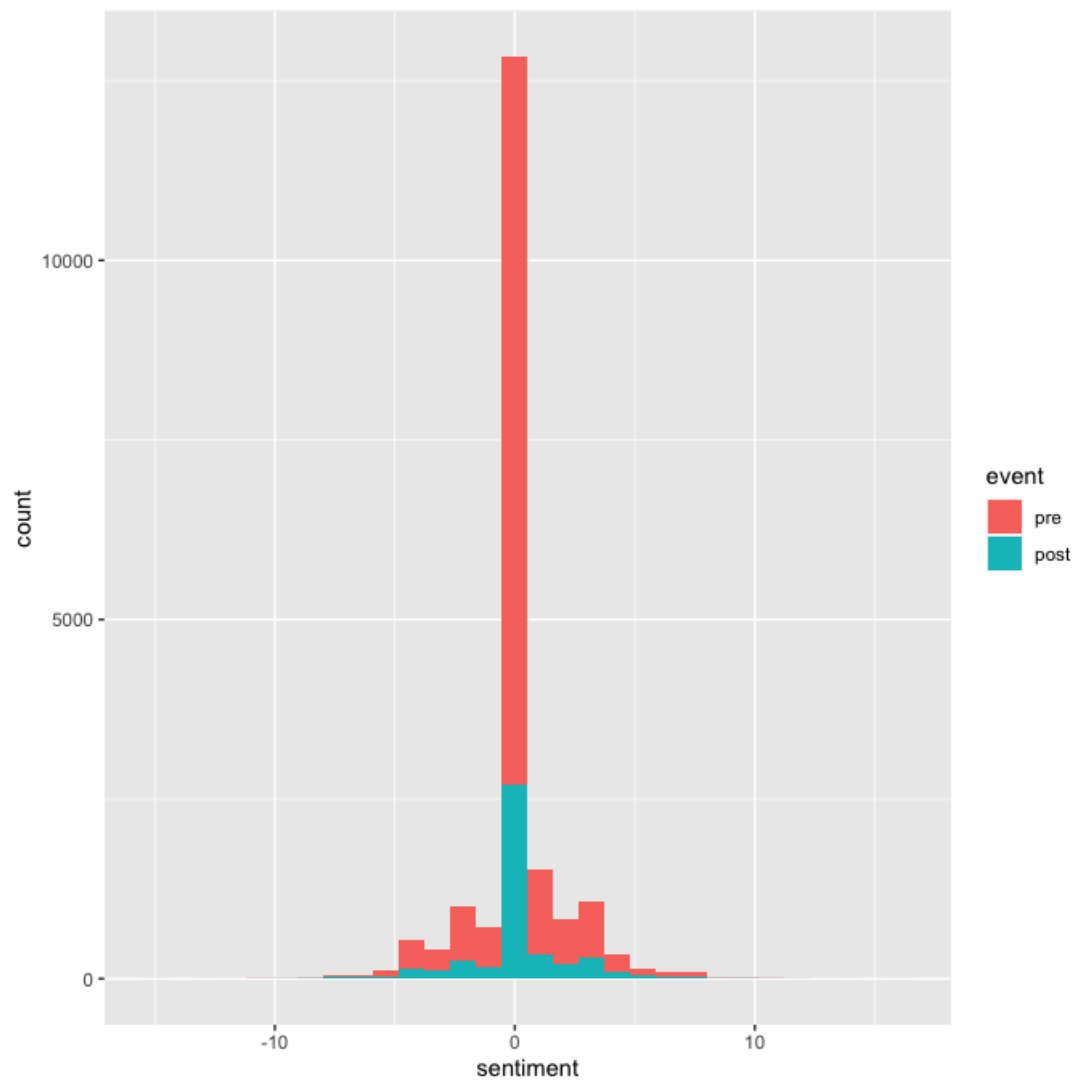


Figure 1. Histogram Density Plot of Sentiment Pre and Post COVID-19 Pandemic

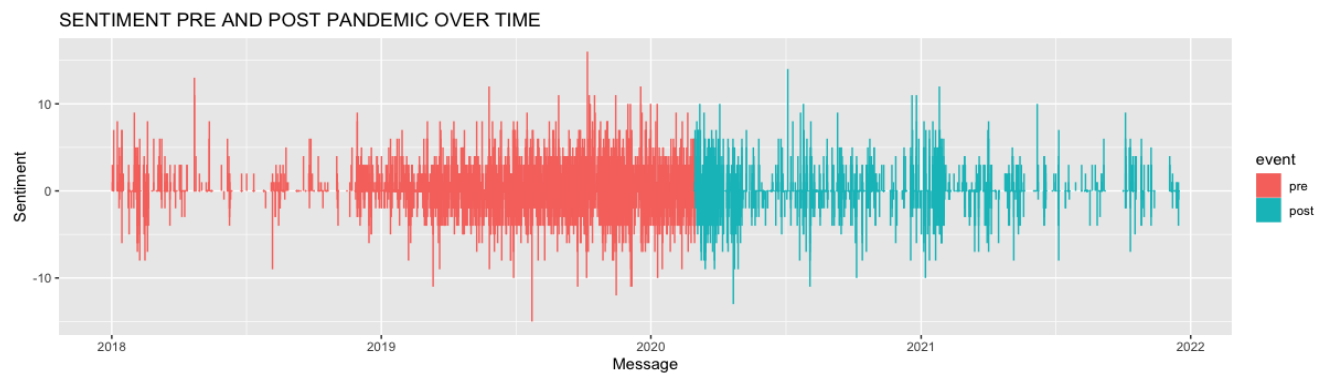


Figure 2. Sentiment Over Time, Pre and Post COVID-19 Pandemic

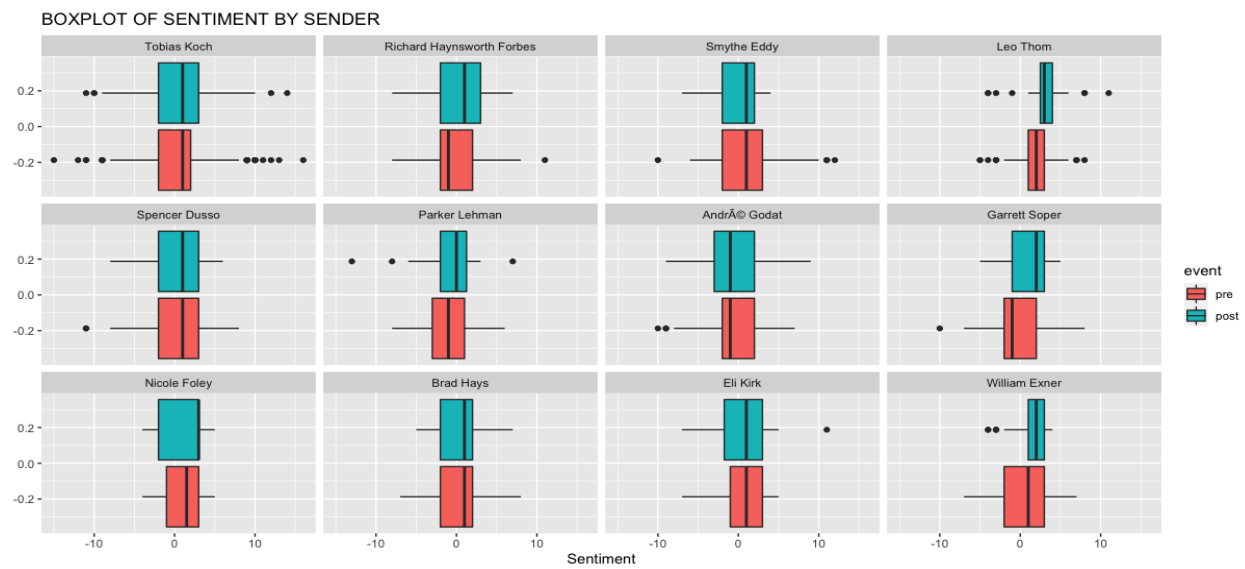


Figure 3. Boxplot of Sentiment for Each Sender

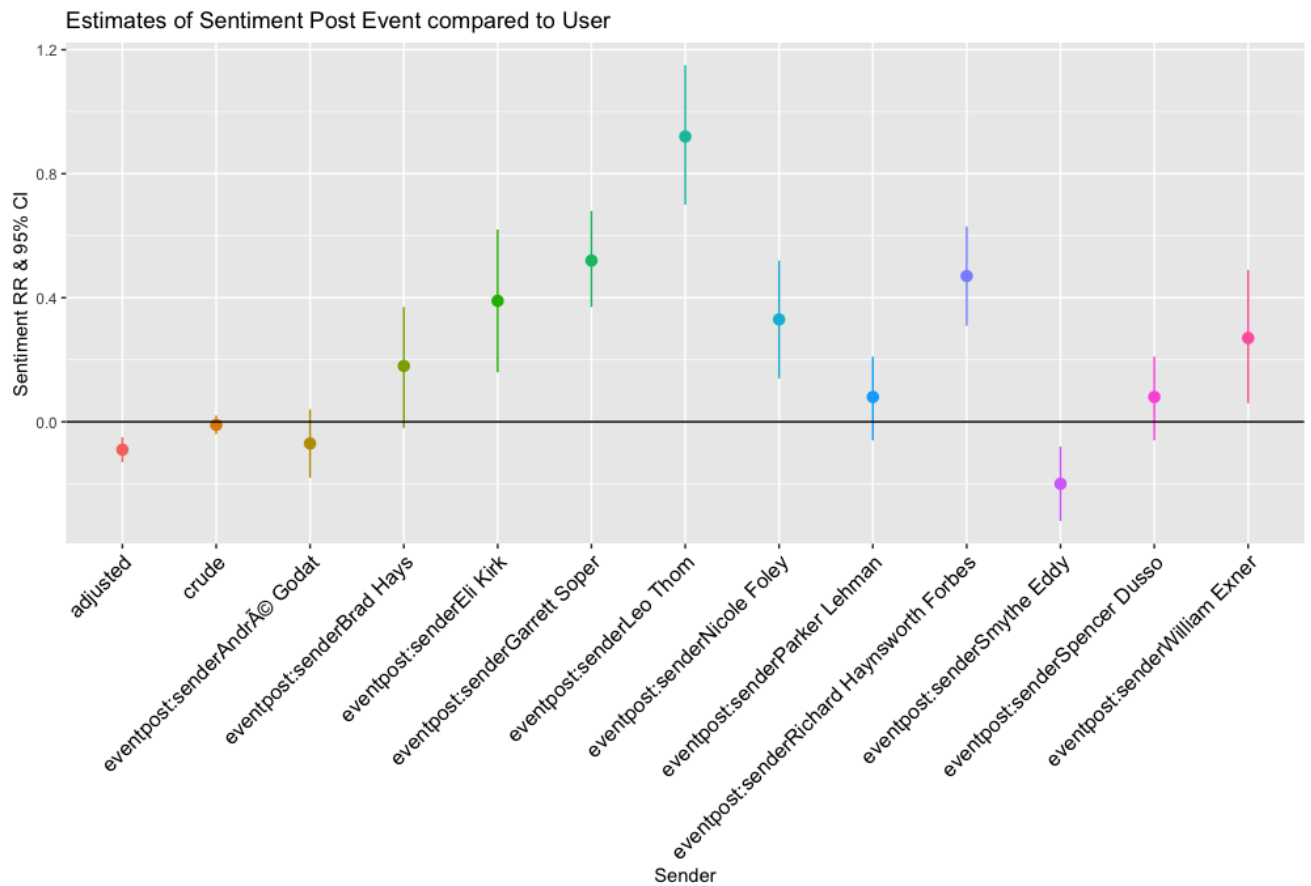


Figure 4. Estimates of Sentiment Post Event Compared to User

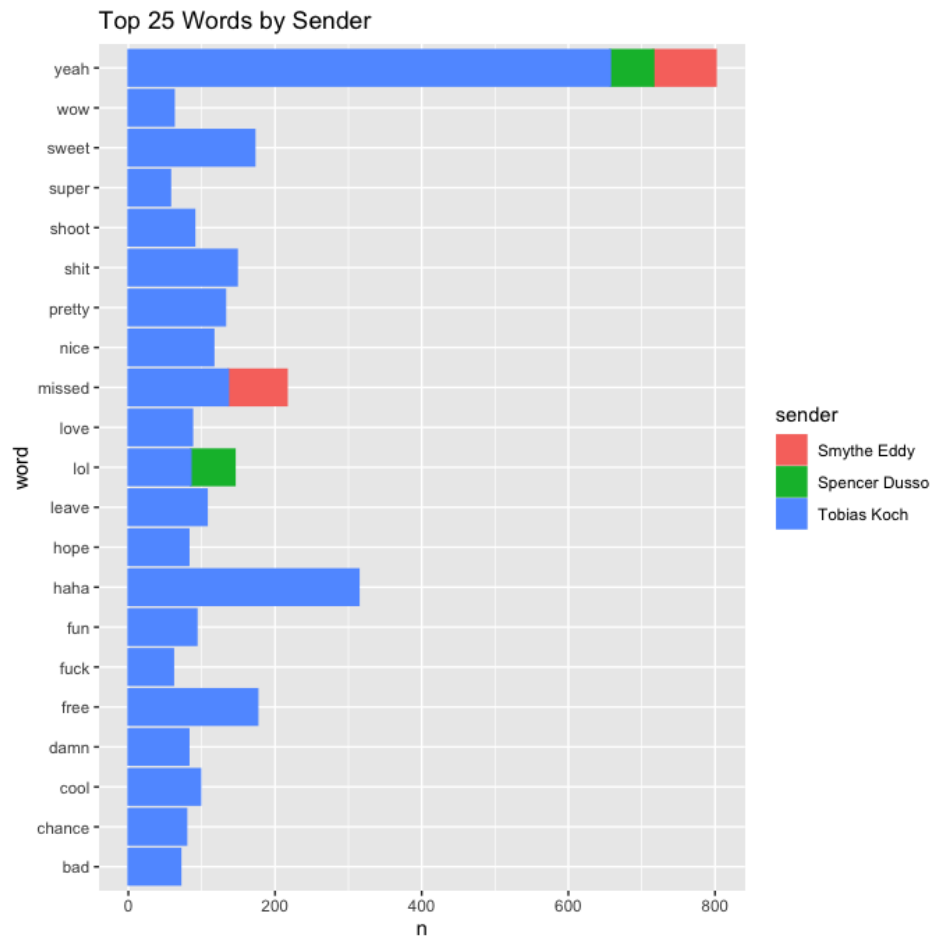


Figure 5. Top 25 Words by Sender

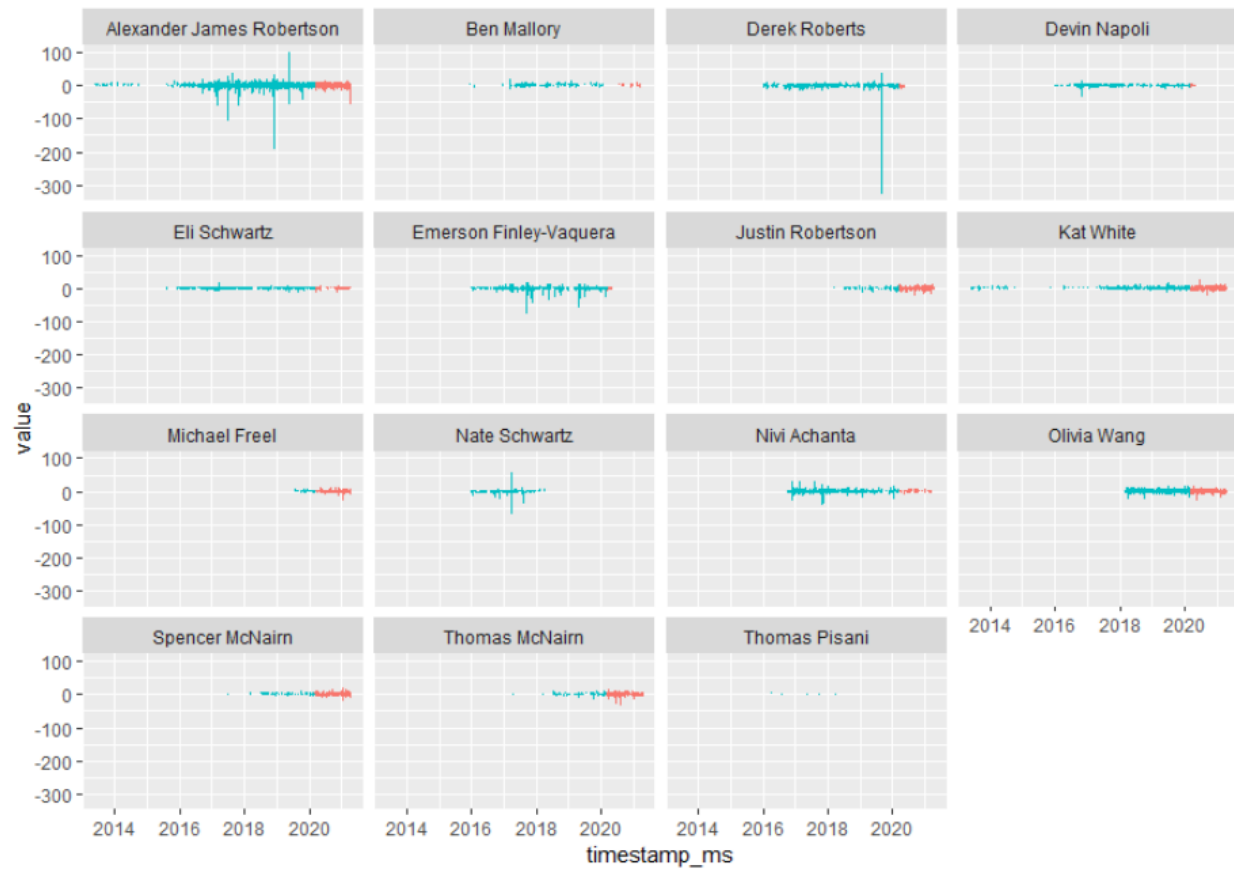


Figure 6. Results of using the `plot_senti_time()` function

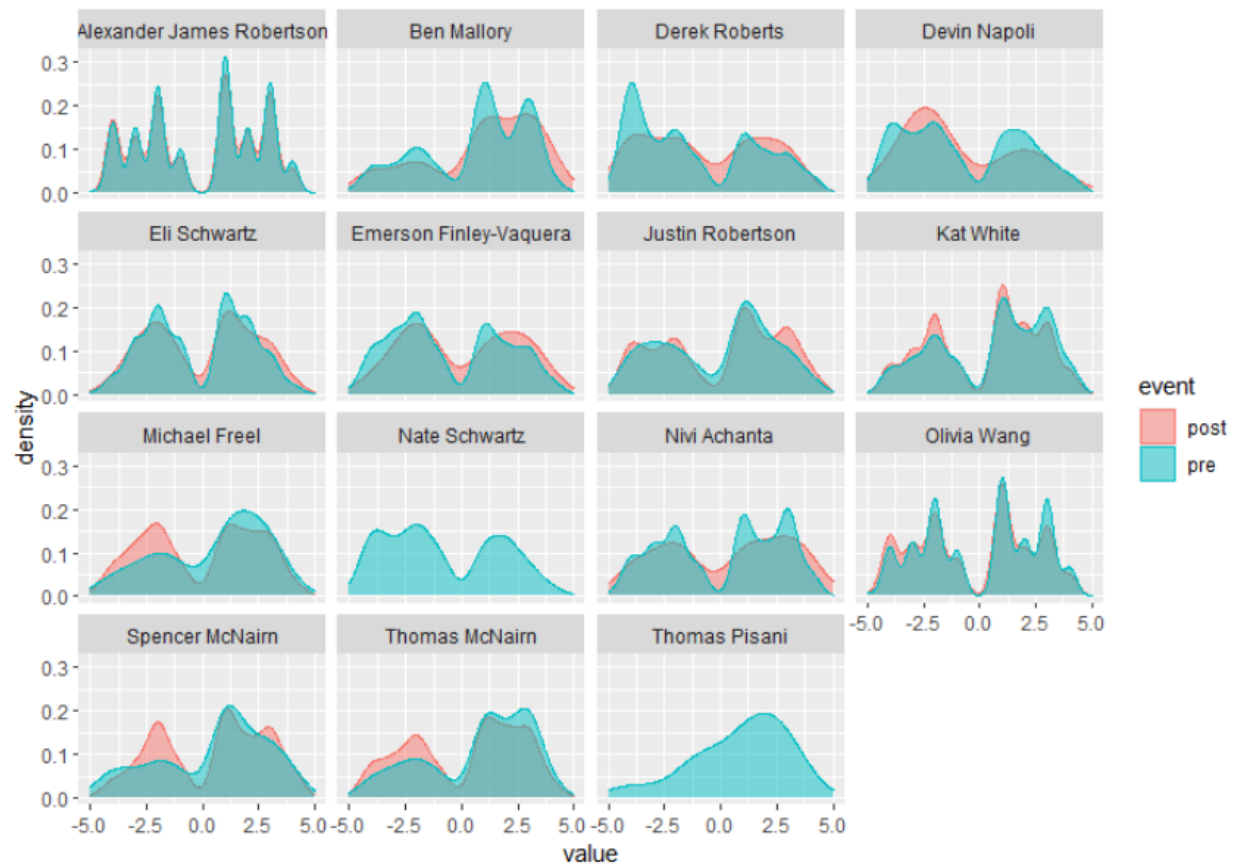


Figure 7. Results of using the `plot_mess_dens()` function


Rank	sender	messages	Words
1	Tobias Koch	9807	548205
2	Smythe Eddy 	1470	65071
3	Andre Godat	1452	74454
4	Parker Lehman	1423	46868
5	Garrett Soper	1376	40151
6	Spencer Dusso	1361	47030
7	Brad Hays	636	27588
8	Richard Haynsworth Forbes	584	26956
9	Leo Thom	525	30085
10	William Exner	460	17559
11	Eli Kirk	399	18696
12	Nicole Foley	396	15690

Table 1. Results of `sender_sum()` function

Term	estimate	ucl	lcl	p.value
crude	-0.01	0.02	-0.04	0.79
adjusted	-0.09	-0.05	-0.13	0.03
Richard Haynsworth Forbes	0.47	0.63	0.31	0
Smythe Eddy	-0.2	-0.08	-0.32	0.11
Leo Thom	0.92	1.15	0.7	0
Spencer Dusso	0.08	0.21	-0.06	0.58
Parker Lehman	0.08	0.21	-0.06	0.58
Andre Godat	-0.07	0.04	-0.18	0.52
Garrett Soper	0.52	0.68	0.37	0
Nicole Foley	0.33	0.52	0.14	0.08
Brad Hays	0.18	0.37	-0.02	0.37
Eli Kirk	0.39	0.62	0.16	0.09
William Exner	0.27	0.49	0.06	0.19

Table 2. Results of Linear Model