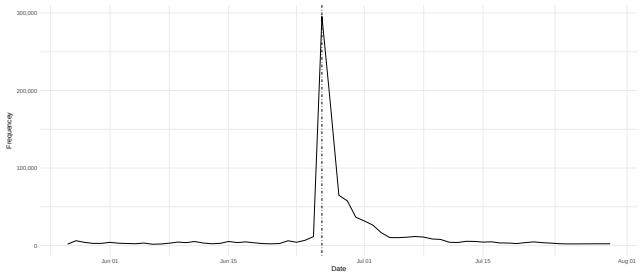
View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Caltech Authors - Main

Appendix A: Additional Descriptive Statistics

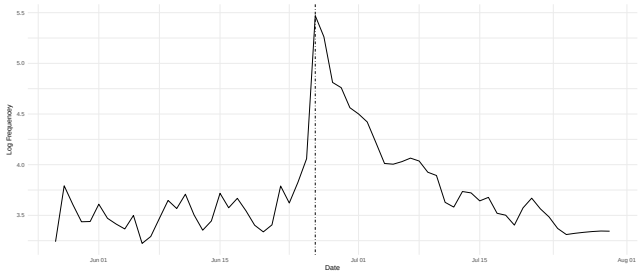
I collected the tweets analyzed in my project over a two-month time span, from May 27, 2015 to August 24, 2015. To obtain this data, I use a series of python scripts that continuously interacted with the Twitter API, using regular expressions to archive any tweet that contained one of the following issue words: **gay marriage, gay marriages, same-sex marriage, same-sex marriages, same sex marriage, same sex marriages, same-sex union, same-sex unions, same sex union, same sex unions, marriage equality, equal marriage**. During this time, I collected a total of 5,996,741 tweets. After filtering for location in the process described in the *Gathering Twitter Data* section above, I end up with 1,028,151 total tweets. In Figure A1, I plot the number of tweets I collected each day. The top half of Figure A1 plots the raw frequency of daily tweets, and it is immediately apparent that a very large number of tweets were sent on June 26, 2015, the day the Supreme Court announced their decision. This drops off quickly, although I collect a large number of tweets until early July. The bottom half of Figure A1 plots the logged frequencies in order to better visualize the entire time series.

Each state is represented in my dataset, with the number of tweets sent from each state enumerated in Table A1. One can also get a general sense of the distribution of users by looking at the heat map in Figure A2, which maps the number of tweets sent per capita using state populations recorded in the 2010 census.

Figure A1: Frequencies: May 27, 2015 to July 31, 2015



(a) Raw Frequency



(b) Log Frequency

Figure A2: Frequency of Tweets by State per Capita

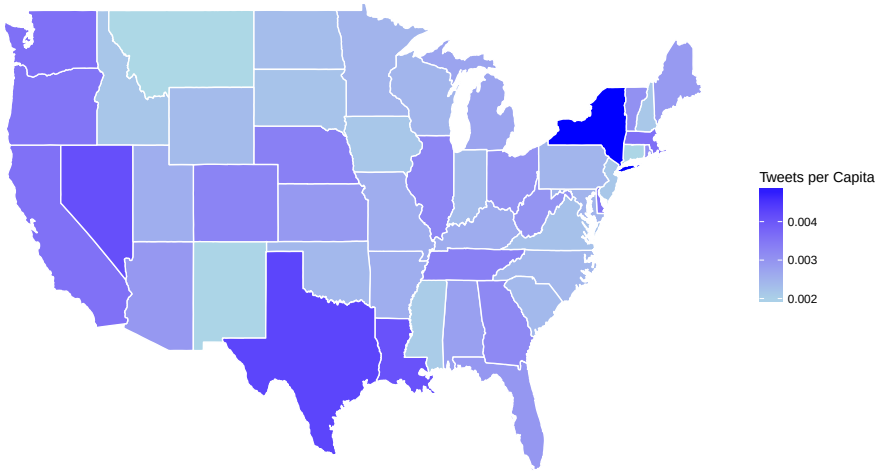


Table A1: Number of Tweets from each State

State	Number of Tweets	State	Number of Users
California	136,340	Kentucky	12,133
Texas	109,485	Nevada	11,425
New York	98,792	South Carolina	11,425
Florida	56,932	Oklahoma	9,184
Illinois	42,874	Kansas	86,25
Ohio	36,222	Arkansas	7,658
Pennsylvania	32,802	Utah	7,315
Georgia	31,259	Connecticut	7,005
Washington D.C.	29,888	Iowa	6,730
Michigan	27,755	Mississippi	6,239
Washington	25,295	Nebraska	6,203
Massachusetts	23,978	West Virginia	5,857
North Carolina	23,539	New Mexico	4,041
Tennessee	21,697	Maine	4,006
New Jersey	20,122	Hawaii	3,758
Arizona	19,372	Idaho	3,443
Louisiana	18,852	Rhode Island	3,235
Virginia	18,602	Delaware	3,189
Colorado	16,936	New Hampshire	2,909
Missouri	15,949	Alaska	2,428
Indiana	154,66	Vermont	2,001
Maryland	14,605	Montana	1,887
Wisconsin	14,391	South Dakota	1,851
Alabama	14,104	North Dakota	1,592
Oregon	14,008	Wyoming	1,351
Minnesota	13,389		

Appendix B: Preprocessing Text Data

Before running supervised training methods to estimate sentiment, I use several preprocessing scripts to manipulate and simplify the Twitter text data. First, I remove all textual information that does not inform the substance of the message, including punctuation, all forms of capitalization, and words that fail to contribute towards a sentence's meaning (such as "the, of, or").

Next, I tokenize the text, a process that splits "a string into its desired constituent parts" (Potts, 2011). My tokenizing strategy utilizes white-space to break apart a sentence into separate words. This transfers the content of a tweet into a list of individual words, ignoring the original order these words appear in the sentence. While the order of words in a sentence can absolutely contribute to the content of a message, treating each document as coming from a "bag-of-words" is a common (though at times contentious) assumption that is necessary to apply many machine learning methodologies (Grimmer & Stewart, 2013). In many situations, enough information can be gleaned from the choice of unique words to justify this assumption.

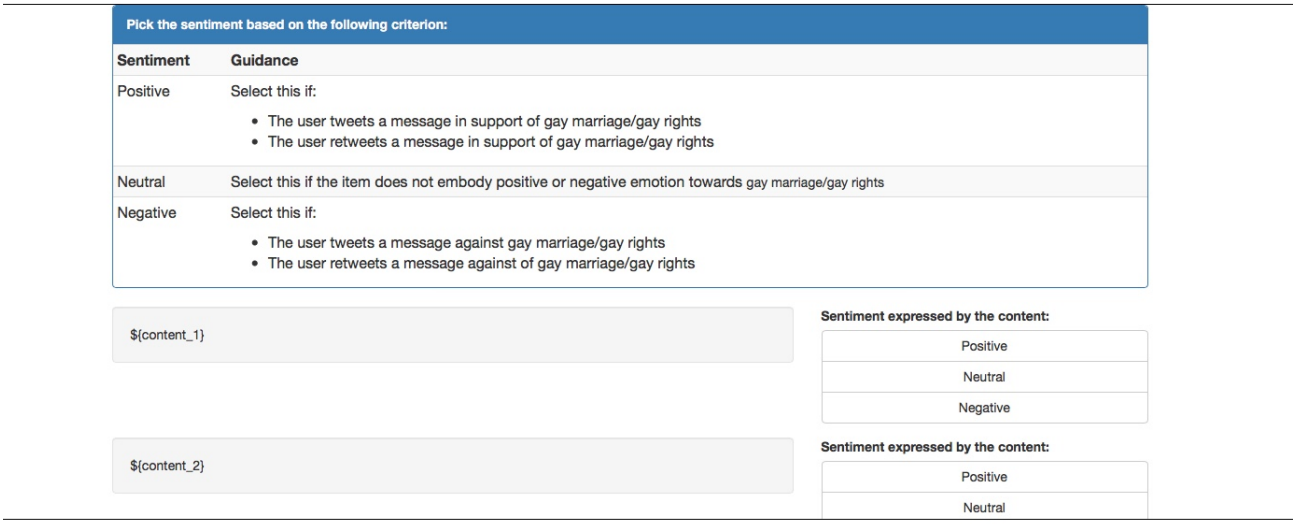
Finally, the entire dataset is transformed into a document-frequency matrix (DFM). A DFM is an $N \times J$ matrix, where N is the number of documents (in this case, tweets) and J is the number of unique features (in this case, individual words) found across all documents. Thus, if tweet n contains two instances of word j, the nj^{th} entry of the DFM is 2. With Twitter data, this represents a very sparse matrix, as the entire set of unique words J across the entire dataset can be quite large, although an individual tweet being capped at 140-characters contains a small number of individual words (while Twitter eventually increased this cap to 280-characters, this occurred after my data collection period) Thus, rather than utilizing each of the J unique features in the entire dataset, I analyze a subset of features based on how frequently the feature appears. This parameter can be tuned, but for the baseline analysis I kept a feature if it appeared at least three times throughout the dataset. In order to implement the preprocessing steps described above, this project utilized the `quanteda` R package (Benoit & Nulty, 2016). The `quanteda` package provides tools to organize and analyze string data in order to implement sentiment analysis methodologies.

Appendix C: Validating the Supervised Scoring Method

In order to train a supervised classifier, I create a set of hand-annotated tweets using Mechanical Turk. In order to label the largest number of messages in the shortest amount of time, the set of annotated tweets corresponds with the top-4,000 most repeated messages in the dataset. In total, these 4,000 tweets represent 1,895,554 total messages, and thus consists of 31.60% of all collected tweets. After stripping these 4,000 messages of usernames, hyperlinks, and punctuation, there were 3,934 unique messages in the validation set.

In order to build this hand-annotated validation set, I utilized Amazon Mechanical Turk, a crowdsourcing platform that allows a researcher to pay individuals to complete small tasks. I created a set of tasks that required Mechanical Turk users to score the sentiment of ten tweets in my validation set. I present a screen shot of the task in Figure A3.

Figure A3: Sample Mechanical Turk Task



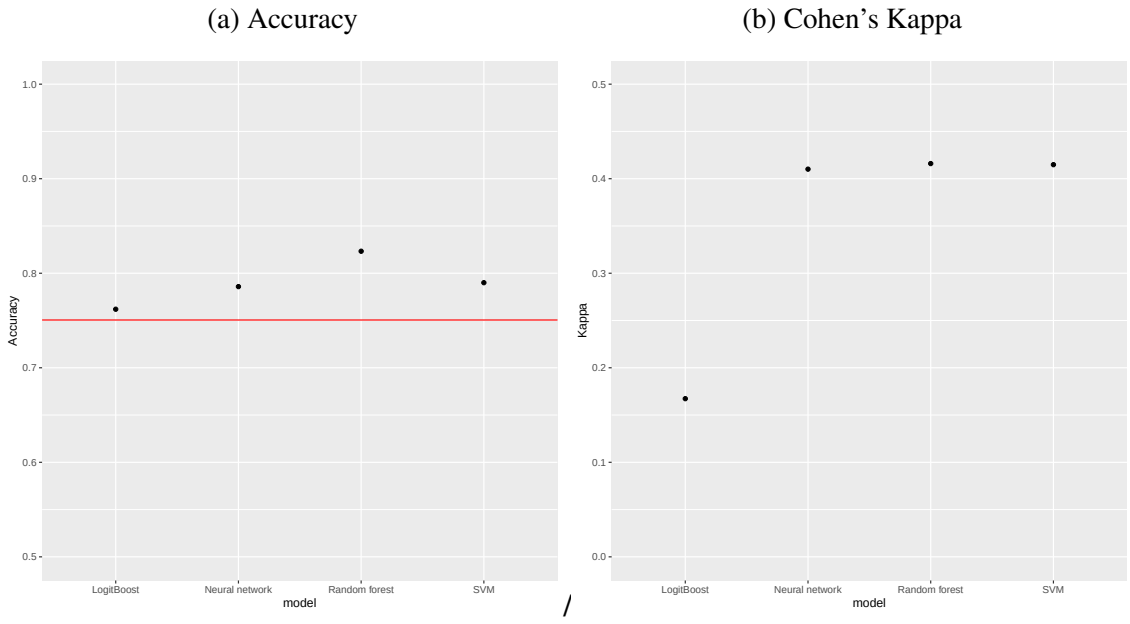
Each task was performed by three separate Mechanical Turk users in order to get a sentiment score as close to the ground truth as possible. To create a final score for each of the 3,934 unique messages, I took the majority score across the three annotations. In total, the Mechanical Turkers labeled 626 messages negative, 1,778 positive, and 1,333 neutral. Only 197 messages did not have

a majority category. In the body of the paper, I focus on using the 626 negative and 1,778 positive messages to train a binary classifier. In Appendix E, I use the neutral messages to build a three-way classifier.

In order to train a classifier, I split 10% of the training data to use a test set. This test set of tweets is not used to train the final model, and thus allows me to evaluate the performance of the model on new data.

I start by testing a number of different classifiers, including Support Vector Machines, logit-boost, neural networks, and random forest.[1] For each model, I run a 10-cross fold validation to train the classifier before evaluating performance on the left out set. The accuracy (compared with a no-information red in red) and kappa coefficient of the best performing models in each category are found in Figure A4. I find that random forest leads to the highest accuracy without sacrificing inter-rater reliability.

Figure A4: Comparing Classifiers



On choosing an overall model classification, I tune the hyper-parameters of the random forest model. Repeating 10-cross fold validation 10 times per hyper-parameter, I test which minimum

[1] I train and evaluate all classifiers with the `caret` package (Kuhn, 2008)

node size leads to the best cross validation accuracy.

For the test set, my final model accurately predicts 81.74% of the data, with 97.78% Sensitivity and 34.43% Specificity.

To better diagnose the model, I present the receiver operating characteristic (ROC) curve (with area under curve reported) in Figure A5, and the test-set confusion matrix in Table A2.

Figure A5: ROC curve

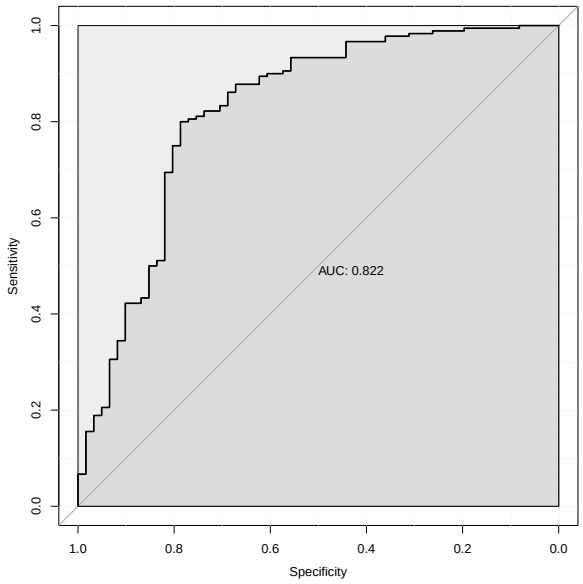


Table A2: Error Matrix:
Test Set Predictions

	Predicted Negative	**Predicted Positive**	**Total**
True Negative	21	4	25
True Positive	40	176	216
Total	61	180	

The error matrix reveals that one issue my classifier exhibits is over-predicting the positive class. While part of this issue may stem from the fact I have an unbalanced training set, visually

inspecting may of the false-positive tweets reveals that many misclassified messages are highly sarcastic in tone. While this is easy for a human reader to recognize, sarcasm is very difficult to detect in sentiment scoring algorithms.[2] Overall, this reveals that my classifier is more likely to falsely classify a negative tweet as positive, biasing all my scores upwards. Thus, as my core finding is finding a more *negative* reaction in states with a law change, this upward bias is likely attenuating my findings. Thus, this bias should not hurt the causal interpretation of my core results.

Appendix D: Neutral Tweets

An issue potentially biasing my results is the presence of a third sentiment category: neutral messages. While theoretically possible to build a third training set of *neutral* tweets and training a three-way classifier, binary classifiers tend to lead to more accurate labels. However, in a robustness check, I retrain the classifier using the neutral labels collected on Mechanical Turk. In total, this training set consists of 626 negative, 1,778 positive, and 1,333 neutral tweets.

I train this model with the same procedure described in an earlier section: leaving out 10% of the data as a test set, and doing 10-fold validation across the training set to tune over the hyperparameters in the random forest model. The confusion matrix for the best performing model on the left out test set is found in Table A3. In total, the model has an accuracy of 61.23% against a 47.06% no information rate, and Cohen's kappa coefficient 0.35.

Applying this three-way classifier to my analysis, I rerun my main model specification including with neutral labels. I score neutral messages as 0.5, in addition to scoring negative messages 0 and positive messages 1. I present the results of this robustness check in Table A4

In Table A4, I note that, across each model specification, the **Treated**$\times$**After** coefficient remains negative and statistically significant. In fact, the model including neutral labels more robustly demonstrates my core results, finding a small level of near statistical significance in model five (a null result in the binary model). This seems to provide additional evidence that the inclusion of neutral tweets biases my core results upwards, allowing me to better interpret the core results in

[2] See (Maynard & Greenwood, 2014) as an example of one attempt to address sarcasm detection in tweets.

Table A3: Multiclass Model Error Matrix:
Test Set Predictions

	Predicted Negative	**Predicted Neutral**	**Predicted Positive**	**Total**
True Negative	15	6	2	23
True Neutral	31	89	49	169
True Positive	18	39	125	182
Total	61	134	176	

Table A4: Difference-in-Difference Results: Three-Way Classifier

	Dependent variable:				
	Positive Sentiment				
	(1)	(2)	(3)	(4)	(5)
After	0.064^{***}	0.016^{***}	0.055^{***}	0.030^{***}	0.023^{***}
	(0.001)	(0.001)	(0.002)	(0.002)	(0.003)
Treated	0.001	0.002	-0.0004	0.007^{*}	0.0002
	(0.002)	(0.002)	(0.003)	(0.004)	(0.006)
Treated*After	-0.019^{***}	-0.026^{***}	-0.018^{***}	-0.016^{***}	-0.010^{*}
	(0.002)	(0.002)	(0.003)	(0.004)	(0.006)
GOP				-0.164^{***}	-0.155^{***}
				(0.001)	(0.002)
Constant	0.649^{***}	0.648^{***}	0.669^{***}	0.643^{***}	0.646^{***}
	(0.001)	(0.001)	(0.002)	(0.002)	(0.003)
Drop June 26?	No	Yes	No	No	No
Race and Gender	No	No	Yes	No	Yes
N	1,076,512	673,792	506,274	191,980	93,681
R^2	0.004	0.001	0.011	0.062	0.060
Note:					*p<0.1; **p<0.05; ***p<0.01

a causal manner.

Appendix E: Border State Analysis

As the parallel trend assumption in the difference-in-difference estimator posits that the untreated group is a good counter-factual to the treatment group, a potential criticism of my work is that I do not restrict the group of untreated states. That is, I analyze data from all fifty states, when perhaps states like California and New York do not make good counterfactuals to the states in the treatment group.

While a matching methodology represents the most rigorous way to find valid counterfactuals for users in my treated set, I do not have a rich enough set of independent variables to allow for an accurate matching procedure. However, it is possible to use the geography of the treated states to find a set of users that might represent a more valid counterfactual. Thus, I re-run my analysis with a smaller set of untreated states, restricting the untreated group to only those states that share a border with one or more treated states.[3] By restricting the untreated states in this way, I am more likely to select states with similar demographic characteristics, allowing me to further test and validate my results. The result of this robustness check is found in Table A5, which replicates the model specifications in the body of the paper.

In models one and two, the baseline models, I find a negative and statistically significant **Treated**$\times$**After** coefficient. Thus, even when restricting the untreated group to smaller set of states more likely to share characteristics with the treated set, I continue to find evidence of a causal impact. I also find this impact in model three, where I include demographic information. In models four and five, where I include the partisan labels, I find a null result. This is partially due to the upward bias of the binary classifier described in the previous appendix; with a higher probability in coding neutral messages as positive, the results are biased upwards, away from my hypothesis.

To test the impact of neutral messages on the border states, I rerun the analysis with the three-

[3]The bordering states include: Oklahoma, Kansas, New Mexico, Colorado, Wyoming, Montana, Minnesota, Iowa, Wisconsin, Illinois, Indiana, Alabama, Florida, South Carolina, North Carolina, Virginia, West Virginia, Pennsylvania.

Table A5: Border States with Binary Classifier

	Dependent variable:				
	Positive Sentiment				
	(1)	(2)	(3)	(4)	(5)
After	0.015^{***}	-0.053^{***}	0.023^{***}	-0.033^{***}	-0.037^{***}
	(0.002)	(0.003)	(0.003)	(0.005)	(0.007)
Treated	0.007^{**}	0.008^{**}	0.015^{***}	-0.009	0.0004
	(0.003)	(0.003)	(0.004)	(0.007)	(0.010)
Treated*After	-0.013^{***}	-0.015^{***}	-0.022^{***}	0.00001	0.005
	(0.003)	(0.004)	(0.005)	(0.007)	(0.011)
GOP				-0.212^{***}	-0.205^{***}
				(0.003)	(0.004)
Constant	0.800^{***}	0.797^{***}	0.813^{***}	0.833^{***}	0.807^{***}
	(0.002)	(0.002)	(0.003)	(0.005)	(0.007)
Drop June 26?	No	Yes	No	No	No
Race and Gender	No	No	Yes	No	Yes
N	580,896	362,785	272,415	102,474	49,974
R^2	0.0001	0.003	0.005	0.055	0.052
Note:					*p<0.1; **p<0.05; ***p<0.01

way classifier described in Appendix D: *Neutral Tweets*. In this model, I code the dependent variable as 0 for negative tweets, 0.5 for neutral tweets, and 1 for positive tweets. I present the results in Table A6.[18]

Table A6: Border States with Three-Way Classifier

	Dependent variable:				
	Positive Sentiment				
	(1)	(2)	(3)	(4)	(5)
After	0.065^{***}	0.015^{***}	0.066^{***}	0.026^{***}	0.031^{***}
	(0.002)	(0.002)	(0.003)	(0.003)	(0.005)
Treated	0.014^{***}	0.015^{***}	0.024^{***}	0.008	0.024^{***}
	(0.002)	(0.003)	(0.003)	(0.005)	(0.006)
Treated*After	-0.021^{***}	-0.025^{***}	-0.028^{***}	-0.011^{**}	-0.019^{***}
	(0.003)	(0.003)	(0.004)	(0.005)	(0.007)
Republican				-0.175^{***}	-0.160^{***}
				(0.002)	(0.003)
Constant	0.637^{***}	0.635^{***}	0.647^{***}	0.647^{***}	0.621^{***}
	(0.002)	(0.002)	(0.003)	(0.003)	(0.005)
Drop June 26?	No	Yes	No	No	No
Race and Gender	No	No	Yes	No	Yes
N	607,695	378,437	286,118	106,761	52,115
R^2	0.003	0.0003	0.012	0.072	0.064
Note:					*p<0.1; **p<0.05; ***p<0.01

Here, I find that all the results are extremely similar to Table A5, but with a negative and significant **Treated**×**After** coefficient in models 4 and 5. This robustness check helps confirm the results in the main section of my paper.

Appendix F: Checking for Bot Accounts

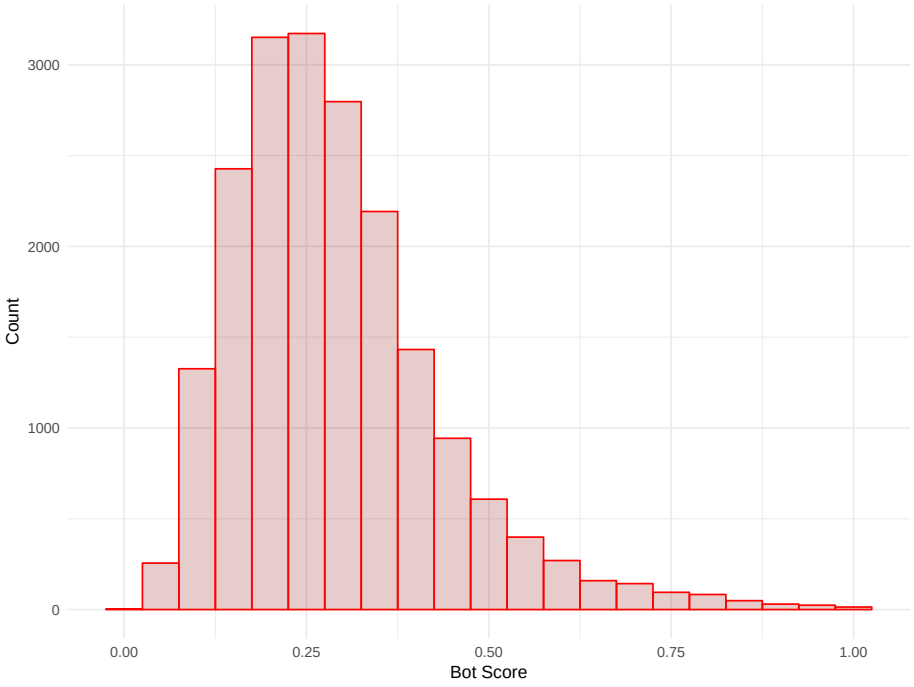
Detecting 'bot' accounts is the subject of many machine learning papers, with researchers focusing on different techniques to determine whether messages are sent by humans or automated programs (e.g. Wang, 2010; Jajodia et al., 2012; Ferrara, Varol, Davis, Menczer, & Flammini, 2016). Given the discussions in the wake of the 2016 U.S. election regarding automated systems disseminating "fake news" on social media platforms, it is important to consider whether or not my dataset is filled with 'bot' accounts biasing my results.

To get a sense of how many likely bot accounts are present in my dataset, I pull a sample of 30,000 random users. To figure out how likely these 30,000 users are 'bot' accounts, I utilize the `Botometer` publicly available API.[4] The `Botometer` API interacts with the Twitter API, pulling over one thousand features from the user's Twitter profile to compare against a collection of 15,000 manually verified bot accounts and 16,000 verified human accounts (Varol, Ferrara, Davis, Menczer, & Flammini, 2017). The classifier then runs an ensemble method using random forests, AdaBoost, logistic regression, and decision trees to determine the likelihood a given user is human or a 'bot.' The classifier outputs a likelihood from zero to one; the closer the bot score is to one, the more likely the account is run by an automated program. I present the distribution of classification scores from 30,000 randomly selected users in Figure A6.

Figure A6 demonstrates that the majority of users are likely human, with a mean bot score of 0.29 with a standard deviation of 0.14 across the sample. Only a small number of users are likely bots, with only 9.2% of users with a bot score greater than 0.5 and 1.3% of users with a bot score greater than 0.75. While important to note `Botometer` represents only one approach to detecting bots, this preliminary analysis shows little evidence that bots drive my results.

[4] https://botometer.iuni.iu.edu

Figure A6: Histogram of Twitter Bot Likelihood



References

Benoit, K., & Nulty, P. (2016). quanteda: Quantitative analysis of textual data [Computer software manual]. Retrieved from `http://github.com/kbenoit/quanteda` (R package version 0.9.1-11)

Ferrara, E., Varol, O., Davis, C., Menczer, F., & Flammini, A. (2016). The rise of social bots. *Communications of the ACM*, *59*(7), 96–104.

Grimmer, J., & Stewart, B. M. (2013). Text as data: The promise and pitfalls of automatic content analysis methods for political texts. *Political Analysis*, *21*, 267-297.

Jajodia, S., Wang, H., Gianvecchio, S., & Chu, Z. (2012, 11). Detecting automation of twitter accounts: Are you a human, bot, or cyborg? *IEEE Transactions on Dependable and Secure Computing*, *9*, 811-824.

Kuhn, M. (2008). Building predictive models in r using the caret package. *Journal of Statistical Software, Articles*, *28*(5), 1–26. Retrieved from `https://www.jstatsoft.org/v028/i05` doi: 10.18637/jss.v028.i05

Maynard, D., & Greenwood, M. (2014, 01). Who cares about sarcastic tweets? investigating the impact of sarcasm on sentiment analysis. *Proceedings of LREC*, 4238-4243.

Potts, C. (2011). Sentiment symposium tutorial [Computer software manual]. Retrieved from `http://sentiment.christopherpotts.net/index.html`

Varol, O., Ferrara, E., Davis, C. A., Menczer, F., & Flammini, A. (2017). Online human-bot interactions: Detection, estimation, and characterization. *arXiv:1703.03107*.

Wang, A. H. (2010). Detecting spam bots in online social networking sites: A machine learning approach. In *Ifip annual conference on data and applications security and privacy* (pp. 335–342).