Tweeter sentiment analysis during COVID19 pandemic

Ivan Kozyryev

4/30/2020

Overview

The goal of my project was to determine whether the overall sentiment of the tweets with COVID19-related hashtags influenced whether the tweet would be retweeted and how many times. My initial hypothesis was that the more negative the tweet is the more retweets it will get because it has potential to stir emotions in many people. During the course of this project, as I started to learn more about tweeter, I became more broadly interested in what parameters of the tweet effect the number of retweets. Specifically, I considered the following tweet variables:

1. number of followers
2. number of friends
3. number of hashtags
4. number of user mentions
5. sentiment of the text

While there are 90 different variables associated with each tweet and many possibilities for further manipulation of the features, I chose to focus on the five above for two reasons:

- Based on my intuition about social media it seemed likely that those parameters might matter.
- I wanted to have a simpler model that can have interpretable implications about how one can spread positive sentiments and correct information on social media to a wide audience. Or alternatively, understand how *misinformation* related to the disease is effectively spread on social media.

In summary, I analyzed 50,000 tweets flagged with #COVID19 and discovered that some of the parameters in the list above play an important role in predicting whether the tweet is retweeted or not. Moreover, sentiment of the tweet has some influence on how many times a tweet is retweeted.

Exploratory initial analysis

First let's load all the libraries needed for the analysis of Twitter data. I used rtweet library to interface with the tweeter API. I chose to load libraries in a separate file for convience since I used 20 different libraries here (e.g. dplyr, readr, caret).

```r
source('Load_libraries.R')
```

As a first step, let's quickly check what topics are trending on Twitter now. I first get the current trends in the US, then sort all the topics by the tweet volume and look at the top 10 trending topics.

```r
US_trends <- get_trends("United States")

US_trends_sorted <- US_trends %>%
  group_by(trend) %>%
  summarize(tweet_vol = mean(tweet_volume)) %>%
  arrange(desc(tweet_vol))
```

```
US_trends_sorted$trend[1:10]
```

```
##  [1] "#<U+0E2A><U+0E35><U+0E48><U+0E41><U+0E2A><U+0E19><U+0E41><U+0E1F><U+0E19><U+0E1D><U+0E32><U+0E
##  [4] "Grandma Killer"       "#AEWDynamite"         "#Riverdale"
##  [7] "#WWENXT"              "#Survivor"            "SAVAGE REMIX PARTY"
## [10] "#TheMaskedSinger"
```

Many of the days recently, covid-19 related hashtags were among the top 10 trending topics on Twitter which indicates its importance in the social discussion.

Obtaining the data for tweets

Let's read *original* tweets and see what associated data is available for them. Thus, I have excluded all replies, retweets and quotes. I also only focused on the tweets in English since I am interested in analyzing the sentiment of their text. Note: Below I only download 18k tweets as an example. Previously, I have read in and saved the data for 50k tweets in the associated csv file so I don't have to wait 45 minutes every time. The limit is 18k tweets per 15 minutes.

```
covid19_st <- search_tweets("#COVID19 -filter:retweets -filter:quote -filter:replies",n = 18000, includ
```

```
## retry on rate limit...
## waiting about 13 minutes...
```

```
colnames(covid19_st)[1:25]
```

```
##  [1] "user_id"               "status_id"             "created_at"
##  [4] "screen_name"           "text"                  "source"
##  [7] "display_text_width"    "reply_to_status_id"    "reply_to_user_id"
## [10] "reply_to_screen_name"  "is_quote"              "is_retweet"
## [13] "favorite_count"        "retweet_count"         "quote_count"
## [16] "reply_count"           "hashtags"              "symbols"
## [19] "urls_url"              "urls_t.co"             "urls_expanded_url"
## [22] "media_url"             "media_t.co"            "media_expanded_url"
## [25] "media_type"
```

As we can see, there are a lot of parameters associated with each tweet but for now I am interested in only a few of those. There are 90 different parameters but I only displayed 25 as example above. The number of displayed columns can be modified to see all the available features for each tweet.

```
covid_twts <- covid19_st[,c("status_id","screen_name","text","retweet_count","followers_count","friends
```

Pre-processing the text for tweets

In order to perform analysis of the text, I first need to clean up the body of the tweet:

- Separate the text from other tweet parameters
- Remove URLs
- Only keep the upper and lower case characters

```
twt_txt <- covid_twts$text # get the tweet text
head(twt_txt)[1:3]
```

```
## [1] "Today starts #nationalnursesweek Join @ENAorg to help support these heroes everyday! #COVID19 h
## [2] "Ready to go shopping. #covid19 #covid #3d #3dprinting https://t.co/jWgTqANRWA"
## [3] "This Coronavirus humbled everyone <U+0001F4AF> #COVID19"
```

```
twt_txt_no_url <- rm_twitter_url(twt_txt)
# head(twt_txt_no_url)
twt_chrs <- gsub("[^A-Za-z]"," ", twt_txt_no_url)
head(twt_chrs)[1:3]
```

```
## [1] "Today starts  nationalnursesweek Join  ENAorg to help support these heroes everyday   COVID  "
## [2] "Ready to go shopping   covid    covid   d   dprinting"
## [3] "This Coronavirus humbled everyone     COVID  "
```

As we can see by comparing the same tweets in the "raw" form and after formatting, we are left only with letters that can be further processed and analyzed.

Pre-processing other tweet-related data

The data we read in provides the lists of the hashtags and users mentioned in each tweet, however, I was interested in looking into how the number of hashtags and mentions effects the degree of retweeting which required some initial processing.

First, lets count the number of mentions and hashtags

```
twts_tidy <- covid_twts %>%
  mutate(text = twt_chrs) %>%
  mutate(mentions = lengths(mentions_user_id)) %>%
  mutate(hashtags = lengths(hashtags)) %>%
  mutate(urls = lengths(urls_url))
```

It is important to notice that most of the tweets don't have any mentions with "NA" in place so we need to properly account for that. I first find which tweets don't have any associated mentions of users and then properly corrected the "mentions" count for those entries.

```
mentions_NAinds <- which(is.na(covid_twts$mentions_user_id)) # find which mentions are non-existent
twts_tidy$mentions[mentions_NAinds] <- 0 # correct the number of mentions entries for those indeces
```

Unlike for mentions, note that each tweet has at least one hashtag since we specifically searched for #COVID19 tweets to begin with. But we also need to account for the fact that not all tweets have listed urls:

```
urls_NAinds <- which(is.na(covid_twts$urls_url)) # find which urls are non-existent
twts_tidy$urls[urls_NAinds] <- 0 # correct the number of mentions entries for those indeces
```

Just for convinience, let's rename the columns of the tweet data set and remove some of the columns that are redundant or will not be useful for our analysis.

```
twts_tidy <- twts_tidy %>%
  dplyr::rename(id = status_id, retweets = retweet_count, friends = friends_count, followers = follower

twts_tidy <- twts_tidy %>%
  dplyr::select(-c(screen_name,mentions_user_id,urls_url))

colnames(twts_tidy)
```

```
## [1] "id"        "text"      "retweets"  "followers" "friends"   "hashtags"
## [7] "mentions"  "urls"
```

Save the cleaned data for future use if needed. Note, I have already done that for the 50k tweet download.

```
write_csv(twts_tidy,"COVID19_50k_tweets_20200503.csv")
```

Here I am reading in the data from the file instead of downloading since this saves time:

```
twts_tidy <- read_csv("COVID19_50k_tweets_20200503.csv", col_types = cols(
  id = col_character(), # otherwise "id" will be read in as col_double
  text = col_character(),
  retweets = col_double(),
  followers = col_double(),
  friends = col_double(),
  hashtags = col_double(),
  mentions = col_double(),
  urls = col_double()
))
```

Text analysis

In order to study the sentiment of individual tweets we need to unnest tokens (words in this case) and remove stop words that don't carry additional information.

```
twts_clean <- twts_tidy %>%
  unnest_tokens(word,text) %>%
  anti_join(get_stopwords(), by = "word")
```

Let's look at the most common words used in tweets associated with #COVID19

```
common_words <- twts_clean %>%
  count(word) %>%
  arrange(desc(n)) %>%
  top_n(n=25, wt = n)

common_words$word
```

```
##  [1] "covid"       "coronavirus" "s"           "cases"       "amp"
##  [6] "new"         "pandemic"    "can"         "people"      "lockdown"
## [11] "t"           "deaths"      "us"          "help"        "may"
## [16] "now"         "health"      "total"       "today"       "day"
## [21] "time"        "get"         "one"         "world"       "india"
```

While many of the top words listed have important meaning (like "us" or "people"), there additional words or letters that don't provide any useful information. In the next step, I remove unique stop words that don't add any meaning.

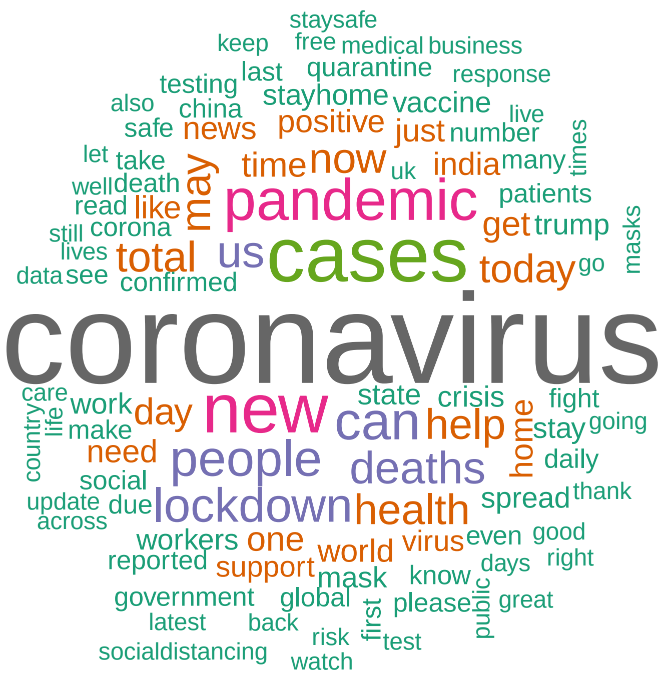
```
uniq_sw <- data.frame(word = c("s","covid","amp","t", "via", "m", "re", "don", "ve", "q", "gt", "o", "p

twts_clean <- twts_clean %>%
   anti_join(uniq_sw, by = "word")
```

Notice that I also removed "covid" since all the tweets have it because we specifically searched for #COVID19 and otherwise the wordcloud would be overwhelmed by the "covid" word (hashtag symbol and "19" were removed during the pre-processing step for the text).

Visualizing the most common words

Let's see which words are the most popular in tweets after we cleanup the text data.

```
pal <- brewer.pal(8,"Dark2")

twts_clean %>%
  count(word) %>%
  with(wordcloud(word,n,random.order = FALSE, max.words = 100, colors = pal))
```



It makes sense that "coronavirus" is the most popular word in the tweets overall.

Sentiment analysis of the #COVID19 tweets

From the wordcloud above, we see that some of the most mentioned words are positive in meaning while others are very negative in meaning. We are now ready to perform tweet sentiment analysis to see whether the *overall* tweet sentiment influences the number of retweets (and therefore potentially the influence of a tweet on other users). In the first step I used "afinn" lexicon which assigned a positive or negative score to each word. The reason I decided to use afinn lexicon is that it doesn't purely flag the word as "positive" or "negative" but assigns degrees to how positive or negative each word is. Given that tweet are very short in nature, I suspected that to have the best chance at predicting retweet degree I had to consider the degree of word sentiment.

Now lets assign a quantitative score to tweets to determine whether they are positive or negative. But first we need to determine which words from the afinn database appear in the tweets we are processing.

```
twts_afinn <- twts_clean %>%
  inner_join(get_sentiments("afinn"), by = "word")
```

Lets assign the sentiment score to each unique tweet

```
afinn_sentiment <- twts_afinn %>%
  group_by(id,retweets,followers,friends,hashtags,mentions,urls) %>%
  summarize(score = sum(value)) %>%
  arrange(desc(score))
```

Let's look at some of the most positive tweets to get inspiration for your day (and make sure that our analysis makes sense):

```
twts_tidy$text[which(twts_tidy$id == afinn_sentiment$id[1])]
```

```
## [1] "Dear God Make our day useful Our night restful Our home peaceful Our future bright Our dream tru
```

```
twts_tidy$text[which(twts_tidy$id == afinn_sentiment$id[2])]
```

```
## [1] "Hofmann s of Wakefield Award Winning Pie Box is fantastic value for money  amp  amazing quality
```

```
twts_tidy$text[which(twts_tidy$id == afinn_sentiment$id[3])]
```

```
## [1] "Another great voice of confidence in our solution  Happy to share that following the winning of
```

Those are indeed positive tweets, so it makes sense that they received high ranking (notice that I am displaying cleaned tweets with only letters remaining).

We can also look at the most negative tweets (according to the afinn lexicon). Here I displayed only the examples that don't use foul language (tweets with the most negative ranking do):

```
twts_tidy$text[which(twts_tidy$id == afinn_sentiment$id[length(afinn_sentiment$id)-1])]
twts_tidy$text[which(twts_tidy$id == afinn_sentiment$id[length(afinn_sentiment$id)-4])]
```

Those tweets do have negative sentiments and the ranking according to the afinn lexicon makes intuitive sense.

Classification analysis for binary retweet status

As a first step, lets see what factors determine whether the tweet is retweeted at all.

Logistic regression

Change the number of retweets into a binary indicator form: "N"(no retweets)/"A"(any retweet)

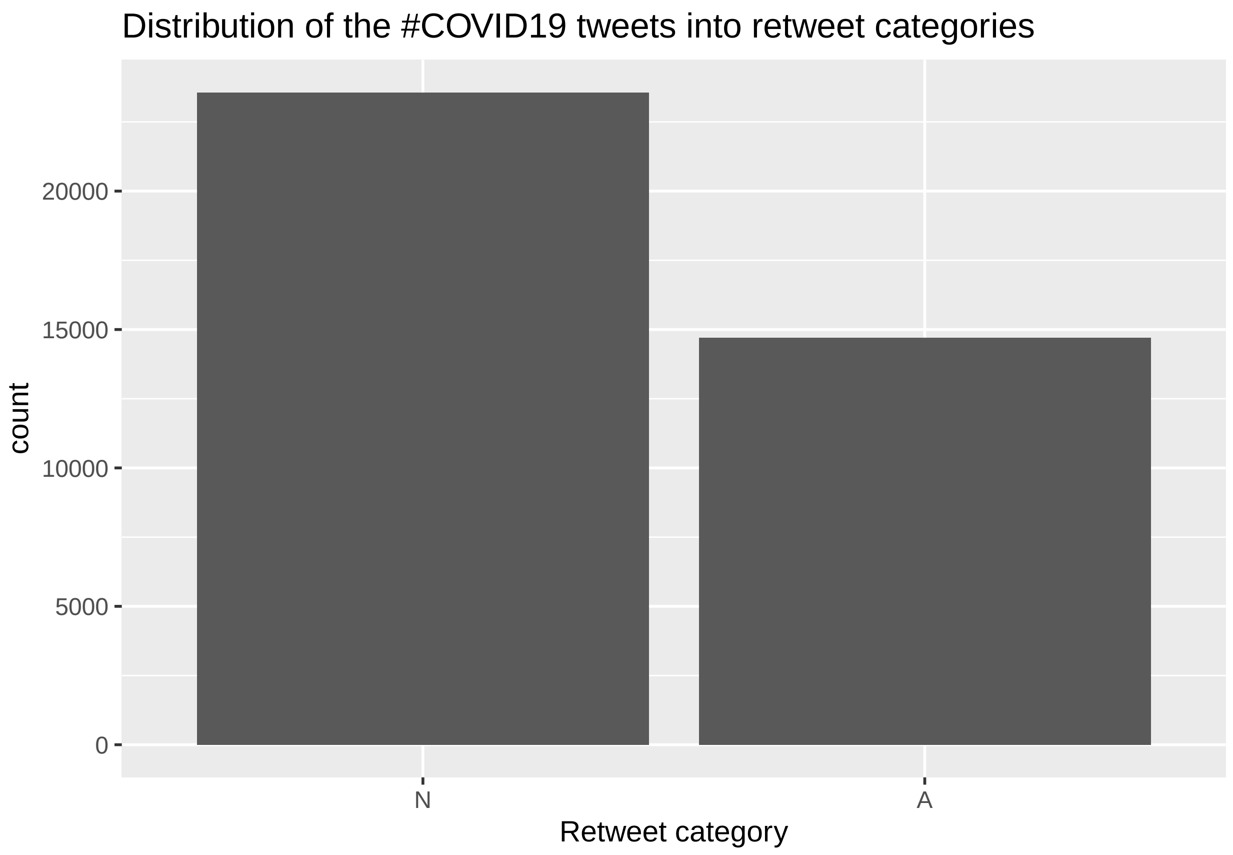
```
afinn_binary <- afinn_sentiment %>%
  mutate(retwtBi = ifelse(retweets > 0, "A", "N"))
```

Convert the retweet indicator into a ranked factor (any retweets is more than none): N < A

```
afinn_binary <- afinn_binary %>%
  mutate(retwtBi = factor(retwtBi, levels = c("N", "A")))
```

Let's look at the distribution of the tweets into the "None" and "Any" retweet categories.

```
afinn_binary %>%
  ggplot(aes(retwtBi)) +
  geom_bar() +
  xlab("Retweet category") +
  ggtitle("Distribution of the #COVID19 tweets into retweet categories")
```



As expected, most of the tweets are not retweeted at all, but a significant portion got retweeted at least ones. In fact we can see how many tweets got retweeted:

```
summaryBi <- afinn_binary %>%
  group_by(retwtBi) %>%
  summarize(n = n())
summaryBi
```

```
## # A tibble: 2 x 2
##   retwtBi     n
##   <fct>   <int>
## 1 N       23566
## 2 A       14695
```

For this given run, looks like `round(summaryBi$n[2]/sum(summaryBi$n)*100)` % of `sum(summaryBi$n)` tweets I am analyzing have been retweeted.

A brief intro to logistic regression

The probability that a tweet is retweeted is $0 < p(A) < 1$.We can model such a probability distribution with a logistic function:

$$\sigma(t) = \frac{1}{1+e^{-t}}$$

which has the following important properties:

- $\sigma(t) \to 0$ as $t \to -\infty$
- $\sigma(t) \to 1$ as $t \to \infty$

- $\sigma(t) = 1/2$ for $t = 0$

Therefore, it seems reasonable to model the probability $p(A)$ as:

$$p = \frac{\exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots \beta_k x_k)}{1 + \exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots \beta_k x_k)}$$

which is equivalent to the expression for $\sigma(t)$ above. Therefore, the odds are

$$\frac{p}{1-p} = \exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots \beta_k x_k)$$

but more importantantly $logit(p)$ is

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots \beta_k x_k$$

is linear in the fitting coefficients β_k. Alternatively, equation above can be represented as:

$\log\frac{p(A)}{p(N)} = \beta_0 + \beta^T x$

where $p(A) + p(N) = 1$.

Implementing logistic regression

Before we can we train the logistic regression model, we need to split the dataset into train and test subsets; here I used the 60/40 split ratio.

```
trBi <- sample(nrow(afinn_binary),round(nrow(afinn_binary)*0.6))
trainSet <- afinn_binary[trBi,]
testSet <- afinn_binary[-trBi,]
```

First, we fit the logit model to the training set to see which parameters are statistically significant:

```
model1LR <- glm(retwtBi ~ followers + friends + score + mentions + hashtags + urls, data = trainSet, fa
summary(model1LR)
```

```
##
## Call:
## glm(formula = retwtBi ~ followers + friends + score + mentions +
##     hashtags + urls, family = binomial(link = "logit"), data = trainSet)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -8.4904  -0.9090  -0.8682   1.3524   1.6360
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.752e-01  2.689e-02 -25.111  < 2e-16 ***
## followers    1.073e-05  5.102e-07  21.022  < 2e-16 ***
## friends      5.041e-06  1.864e-06   2.704  0.00685 **
## score        7.265e-03  3.982e-03   1.824  0.06811 .
## mentions     1.361e-01  1.064e-02  12.794  < 2e-16 ***
## hashtags    -1.146e-02  4.534e-03  -2.528  0.01146 *
## urls        -8.790e-02  2.653e-02  -3.314  0.00092 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 30659  on 22956  degrees of freedom
## Residual deviance: 28475  on 22950  degrees of freedom
## AIC: 28489
##
## Number of Fisher Scoring iterations: 10
```

From the summary table of logistic regression model (model1LR) we can see that followers and mentions variables have the highest statistical significance in determining whether the tweet will be retweeted at least once or not. The positive fit coefficient indicates that more mentions and followers lead to a higher chance to be retweeted.

In order to determine how accurate the fitted model is we now apply it to the "test" dataset and compare to the actual values:

```
prob1LR <- predict(model1LR,testSet, type = "response")
# prob1LR
cl1LR <- ifelse(prob1LR > 0.5, "A", "N")
```

Confusion matrix will provide a lot of valuable information on the model peformance. I set the positive class to "A" as we are interested in predicting whether the tweet will receive any retweets

```
confusionMatrix(factor(cl1LR, levels = c("N","A")),testSet$retwtBi, positive = "A")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    N    A
##          N 9243 4661
##          A  268 1132
##
##                Accuracy : 0.6779
##                  95% CI : (0.6705, 0.6853)
##     No Information Rate : 0.6215
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.1963
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.19541
##             Specificity : 0.97182
##          Pos Pred Value : 0.80857
##          Neg Pred Value : 0.66477
##              Prevalence : 0.37853
##          Detection Rate : 0.07397
##    Detection Prevalence : 0.09148
##       Balanced Accuracy : 0.58362
##
##        'Positive' Class : A
##
```

Notice that while the overall accuracy is 68%, Cohen's kappa value above 0.2 indicates only a fair agreement because many of the retweeted tweets are classified as "N" by mistake. Let's run the second model now keeping only statistically significant features in the model:

```r
model2LR <- glm(retwtBi ~ followers + mentions, data = trainSet, family = binomial(link = "logit"))
summary(model2LR)
```

```
##
## Call:
## glm(formula = retwtBi ~ followers + mentions, family = binomial(link = "logit"),
##     data = trainSet)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -8.4904  -0.8939  -0.8795   1.3607   1.5085
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -7.514e-01  1.651e-02  -45.50   <2e-16 ***
## followers    1.114e-05  4.897e-07   22.74   <2e-16 ***
## mentions     1.393e-01  1.060e-02   13.15   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 30659  on 22956  degrees of freedom
## Residual deviance: 28495  on 22954  degrees of freedom
## AIC: 28501
##
## Number of Fisher Scoring iterations: 10
```

We can see now that all of the variables are statistically significant as expected.

```r
prob2LR <- predict(model2LR,testSet, type = "response") # predicted probability
# pred2LR
cl2LR <- ifelse(prob2LR > 0.5, "A", "N") # predicted binary classification
confusionMatrix(factor(cl2LR, levels = c("N","A")),testSet$retwtBi, positive = "A")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    N    A
##          N 9249 4673
##          A  262 1120
##
##                Accuracy : 0.6775
##                  95% CI : (0.6701, 0.6849)
##     No Information Rate : 0.6215
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.1948
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.19334
##             Specificity : 0.97245
##          Pos Pred Value : 0.81042
##          Neg Pred Value : 0.66434
##               Prevalence : 0.37853
##           Detection Rate : 0.07318
##     Detection Prevalence : 0.09030
##        Balanced Accuracy : 0.58289
##
##         'Positive' Class : A
##
```

We can see that accuracy is in fact comparable to the full model.

One of the advantages of the ligistic regression is that it's coefficients lend to intuitive interpretation based on the log(odds) discussion above.

```
exp(coef(model2LR))
```

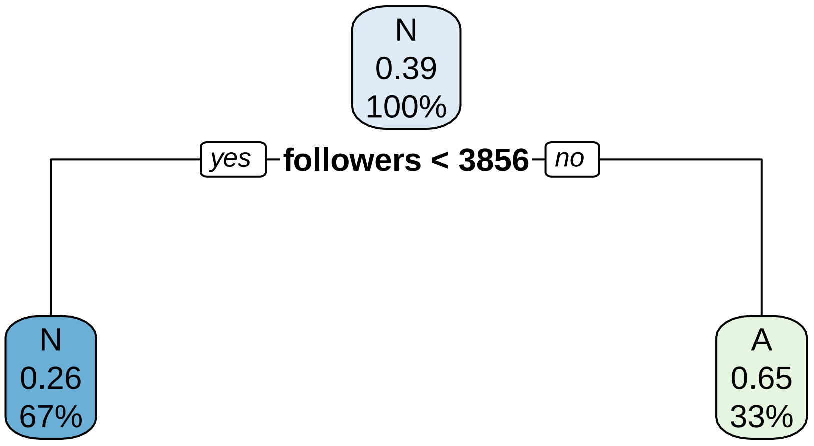
```
## (Intercept)   followers    mentions
##   0.4717209   1.0000111   1.1494963
```

We can see that mentions of other users have the largest effect on log-odds. This is something that the author of the tweet can control thus potentially increasing the odd of the tweet being retweeted

Decision tree for binary retweet indicator

The logit function assumed a linear relationship between the log(odd) and fitting coefficients. Thus it could be instructive to look at some nonlinear models. Let's use the same variables but fit the decision tree instead of logit regression

```
model1DT <- rpart(retwtBi ~ followers + friends + score + mentions + hashtags, data = trainSet, method =
rpart.plot(model1DT)
```



```
cl1DT <- predict(model1DT,testSet, type = "class")
confusionMatrix(cl1DT,testSet$retwtBi, positive = "A")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    N    A
##          N 7711 2615
##          A 1800 3178
##
##                Accuracy : 0.7115
##                  95% CI : (0.7043, 0.7187)
##     No Information Rate : 0.6215
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3695
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.5486
##             Specificity : 0.8107
##          Pos Pred Value : 0.6384
##          Neg Pred Value : 0.7468
##              Prevalence : 0.3785
##          Detection Rate : 0.2077
##    Detection Prevalence : 0.3253
##       Balanced Accuracy : 0.6797
##
##        'Positive' Class : A
##
```

As we can see from the confusion matrix, a single decision tree performs better than the logistic regression on such important parameters as accuracy and sensitivity (recall). Therefore, there is a lot of potential for random forest approach to improve the predicting power of our model.

Let's fit the random forest model to the full data:

```
model1RF <- randomForest(retwtBi ~ followers + friends + score + mentions + hashtags + urls, data = tra
```

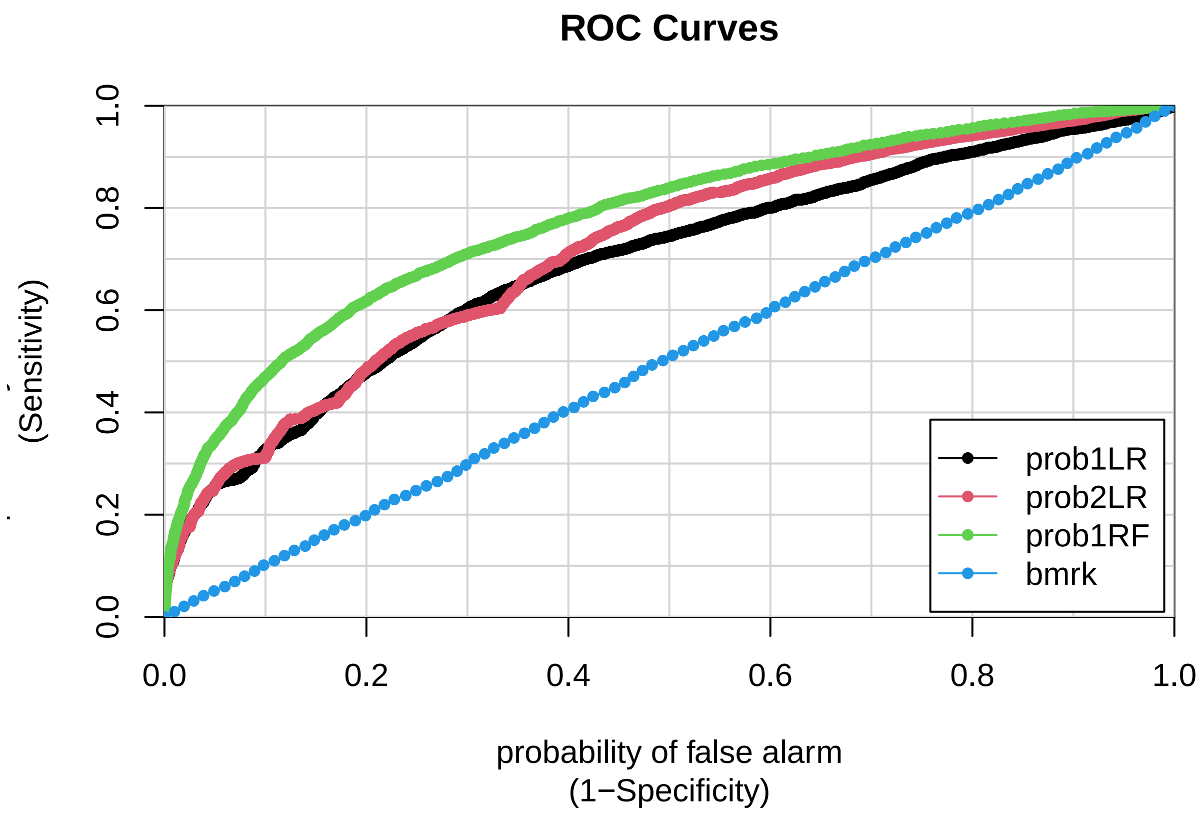
We can now calculate predicted probabilities and classes:

```
prob1RFboth <- predict(model1RF,testSet, type = "prob") # predicted probabilities
prob1RF = prob1RFboth[,"A"]
# prob1RF
cl1RF <- predict(model1RF,testSet, type = "class") # predicted binary classification
# cl1RF
confusionMatrix(cl1RF,testSet$retwtBi, positive = "A")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    N    A
##          N 8205 2724
##          A 1306 3069
##
##                Accuracy : 0.7367
##                  95% CI : (0.7296, 0.7436)
##     No Information Rate : 0.6215
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4122
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.5298
##             Specificity : 0.8627
##          Pos Pred Value : 0.7015
##          Neg Pred Value : 0.7508
##              Prevalence : 0.3785
##          Detection Rate : 0.2005
##    Detection Prevalence : 0.2859
##       Balanced Accuracy : 0.6962
##
##        'Positive' Class : A
##
```

In order to compare how different fitted model perform I will look at the ROC curves. Let's plot the ROC curves and compare the areas:

```
bmrk = seq(0,1,by=0.01)
caTools::colAUC(cbind(prob1LR,prob2LR,prob1RF, bmrk), testSet$retwtBi,plotROC = T)
```



```
##             prob1LR   prob2LR   prob1RF      bmrk
## N vs. A 0.6925732 0.7180424 0.7769357 0.5000773
```

Area under the ROC curve can be used to compare various models in order to pick the optimal one and we conclude that random forest model performs the best on this metric as well. I also plotted the benchmark line for reference.

From the area under the ROC curve we can see that Logistic Regression model 2 (LR2) (fitted only on followers and mentions) performs slightly better compared to LR1 model (fitted on all data). We can use cross-validation to try fitting different logistic regression models and see which one performs the best:

```r
set.seed(42)
# use cross-validation from the caret library
cv_model1 <- train(
  retwtBi ~ followers,
  data = trainSet,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10)
)

cv_model2 <- train(
  retwtBi ~ mentions,
  data = trainSet,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10)
)

cv_model3 <- train(
  retwtBi ~ hashtags,
  data = trainSet,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10)
)

cv_model4 <- train(
  retwtBi ~ score,
  data = trainSet,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10)
)

cv_model5 <- train(
  retwtBi ~ followers + mentions + hashtags,
  data = trainSet,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10)
)
```

Let's comapre the accuracy performance of different models now:

```r
summary(
  resamples(
    list(
      cv1 = cv_model1,
      cv2 = cv_model2,
      cv3 = cv_model3,
      cv4 = cv_model4,
      cv5 = cv_model5
    )
  )
)$statistics$Accuracy
```

```
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## cv1 0.6572300 0.6650333 0.6693534 0.6684237 0.6721114 0.6790070    0
## cv2 0.6130719 0.6143120 0.6155018 0.6162392 0.6168174 0.6226580    0
## cv3 0.6119338 0.6122004 0.6122004 0.6122316 0.6123693 0.6123693    0
## cv4 0.6119338 0.6122004 0.6122004 0.6122316 0.6123693 0.6123693    0
## cv5 0.6639094 0.6687351 0.6705139 0.6723011 0.6772331 0.6824913    0
```

We can see that the model with all three statistically significant parameters from fitting model LR2 has the best accuracy. However, using only the number of followers in the logistic regression model provides a decent accuracy as well, which indicates that this is the most important parameter for predicting whether the tweet is retweeted or not, which makes intuitive sense but it's great to see that our intuition is supported by the data as well.

Summary of the Binary Retweet Analysis

We have used Logistic Regression to identify the most important tweet variables in determining whether it is retweeted or not. Such factors as number of followers, mentions of other users and number of hashtags turned out to be statiscatilly important. We further observed that number of **followers** is the most important mentric in predicting the propensity of the tweet to be retweeted.

We can try adding more information about the tweet to begin with (remember there are 90 different variables associated with a single tweet we can get) however, personally I am interested in seeing which tweets influence many people: i.e. what causes tweet to become "viral" and be retweeted hundreds or thousands of times and potentially have a significant influence on the perception of many people about COVID19. Thus, let's actually divide tweets which have been retweet into multiple sub-categories.

Multivariate Logit

First I changed the retweets into 4 categories (CAT) that define how "popular" the tweet is:

- "N (none)":0
- "S (small)":1-10
- "M (medium)":11-100
- "L (large)":>100

While other possible divisions exist (for example I included "viral" category initially with >1,000 retweets), I wanted to make sure that there are enough samples in each category for statistically significant results from the analysis.

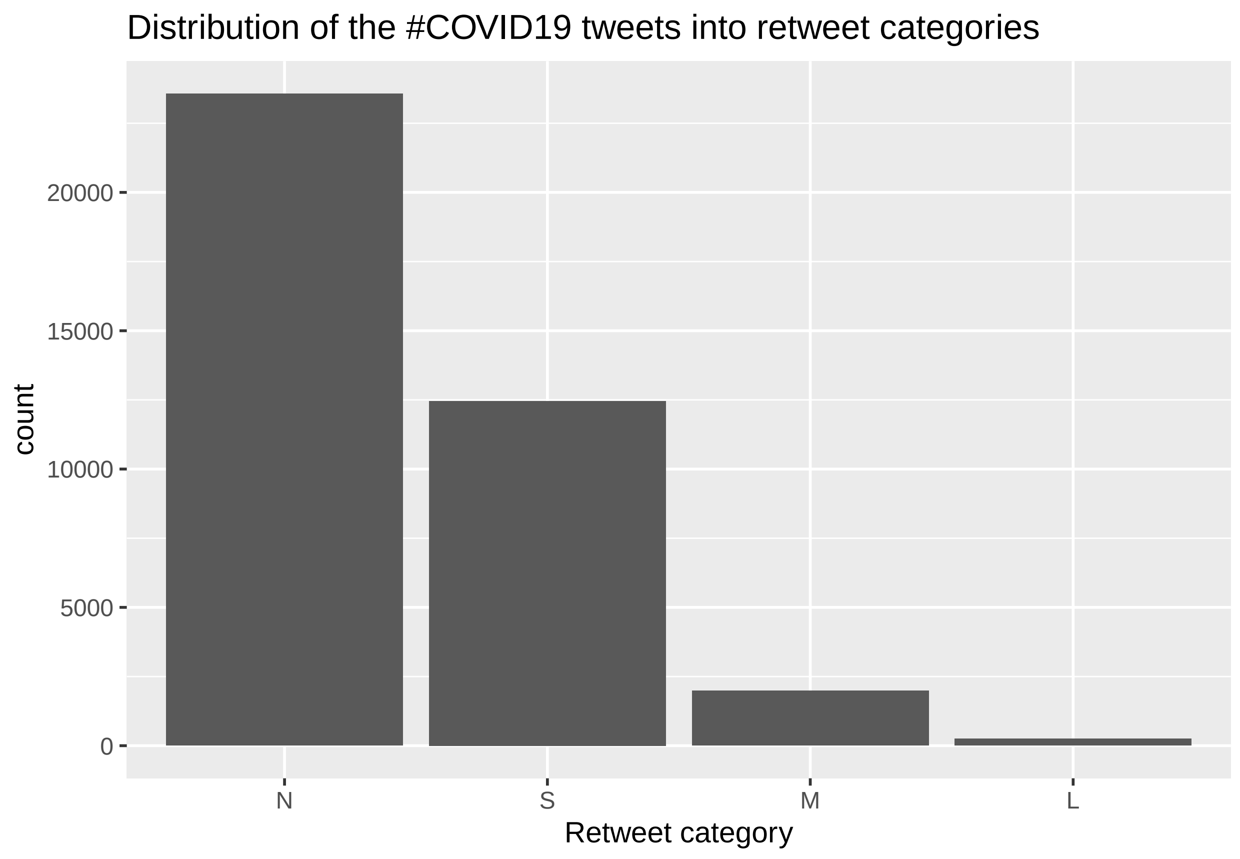
```
# afinn_five <- afinn_sentiment %>%
   # mutate(retwtCAT = ifelse(retweets == 0, "N",ifelse(retweets %in% 1:10, "S", ifelse(retweets %in% 1

# In order to do the random forest regression we cannot have any empty categories; so delete the Viral
afinn_four <- afinn_sentiment %>%
  mutate(retwtCAT = ifelse(retweets == 0, "N",ifelse(retweets %in% 1:10, "S", ifelse(retweets %in% 11:1
```

Lets convert the categories for retweets into fanked factors since there is an inherent order to them. Ranked order will be used with the multivariate logistic regression or can be used for ordinal logistic regression.

```r
# afinn_RK <- afinn_five %>%
#    mutate(retwtCAT = factor(retwtCAT,levels = c("N","S","M","L","V")))
# In case "L" category is empty use only 4 buckets
afinn_RK <- afinn_four %>%
 mutate(retwtCAT = factor(retwtCAT,levels = c("N","S","M","L")))
```

Let's see how many tweets per category there are in our sample:

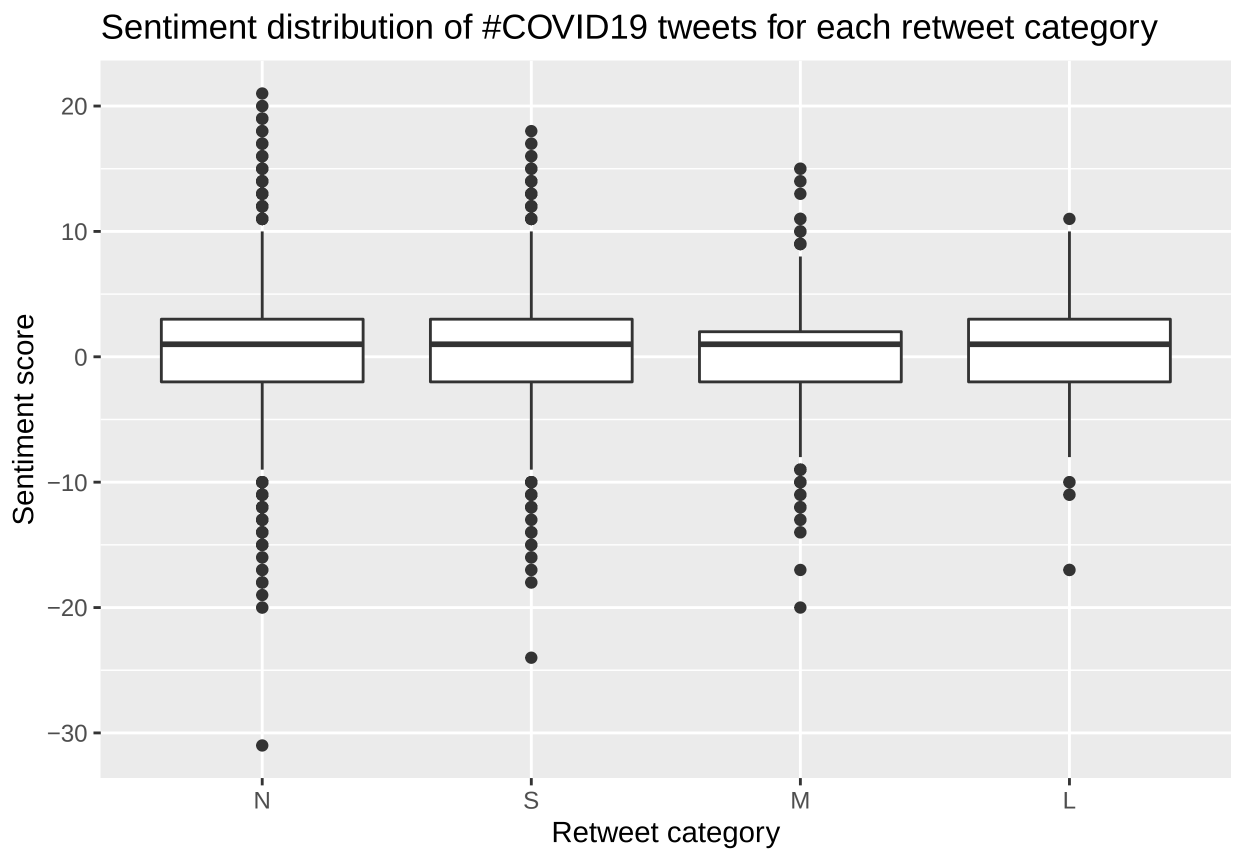
```r
afinn_RK %>%
  ggplot(aes(retwtCAT)) +
  geom_bar() +
  xlab("Retweet category") +
  ggtitle("Distribution of the #COVID19 tweets into retweet categories")
```



Most of the tweets appear to come from "N" and "S" categories which we should keep in mind when trying to predict tweets which will be retweeted more than 10 times. This makes sense intuitively since we are looking at the tweets posted recently only.

Let's display the sentiment distribution for each category

```r
afinn_RK %>%
  ggplot(aes(retwtCAT,score)) +
  geom_boxplot() +
  xlab("Retweet category") +
  ylab("Sentiment score") +
  ggtitle("Sentiment distribution of #COVID19 tweets for each retweet category")
```



Next I perform the multinomial logistic regression next since the oucome can be in one of the 4 categories: "N", "S", "M" or "L".

```
modelMLR_A <- multinom(retwtCAT ~ followers + friends + score + mentions + hashtags + urls, data = afinn
```

```
## # weights:  32 (21 variable)
## initial  value 53041.008551
## iter  10 value 37488.972642
## iter  20 value 34707.854470
## iter  30 value 31046.934501
## iter  40 value 30886.094216
## iter  50 value 30656.940765
## iter  60 value 30390.631944
## iter  70 value 30265.565570
## final  value 30249.523536
## converged
```

```
summary(modelMLR_A)
```

```
## Call:
## multinom(formula = retwtCAT ~ followers + friends + score + mentions +
##     hashtags + urls, data = afinn_RK)
##
## Coefficients:
##   (Intercept)    followers      friends       score   mentions     hashtags
## S  -0.9226479 1.039722e-05 5.168774e-06  0.006988341 0.13745099  0.002738625
## M  -2.3321342 1.085305e-05 9.723646e-06 -0.026118817 0.15472031 -0.138512992
## L  -3.8652367 1.088720e-05 9.483979e-06 -0.022171757 0.01082957 -0.234260557
##          urls
## S -0.00803076
## M -0.34796719
## L -1.00690076
##
## Std. Errors:
##    (Intercept)    followers      friends        score     mentions     hashtags
## S 7.418482e-11 3.804635e-07 1.443061e-06 2.330095e-11 3.165635e-11 3.317484e-10
## M 3.153175e-11 3.809512e-07 1.701277e-06 2.048111e-11 1.953063e-11 9.701076e-11
## L 8.632981e-12 3.813258e-07 2.214191e-06 6.169382e-12 3.925552e-12 2.037069e-11
##           urls
## S 5.356798e-11
## M 2.049137e-11
## L 3.166974e-12
##
## Residual Deviance: 60499.05
## AIC: 60541.05
```

Notice that all the fit coefficients are referenced to the "N" category:

$$\log \frac{Pr(CAT)}{Pr(N)} = \beta_0 + \beta^T x$$

where CAT is "S", "M" or "L". Let's estimate the statistical significance of the fit coefficients using the p-value for Wald's test:

```
z <- summary(modelMLR_A)$coefficients/summary(modelMLR_A)$standard.errors

z
```

```
##     (Intercept) followers  friends       score   mentions     hashtags
## S  -12437151567  27.32778 3.581812   299916569 4341972428      8255127
## M  -73961463206  28.48935 5.715499 -1275263605 7921929226  -1427810597
## L -447729084926  28.55093 4.283270 -3593837621 2758739055 -11499881155
##            urls
## S    -149917185
## M  -16981161939
## L -317937820120
```

```
p <- (1 - pnorm(abs(z), 0, 1)) * 2
p
```

```
##   (Intercept) followers      friends score mentions hashtags urls
## S           0         0 3.412193e-04     0        0        0    0
## M           0         0 1.093826e-08     0        0        0    0
## L           0         0 1.841669e-05     0        0        0    0
```

It looks like coefficients for followers, friends, sentiment score, mentions, hashtags and urls are statistically significant. Let's now look in more details at each category but first we need to add additional binary indicators to the dataset:

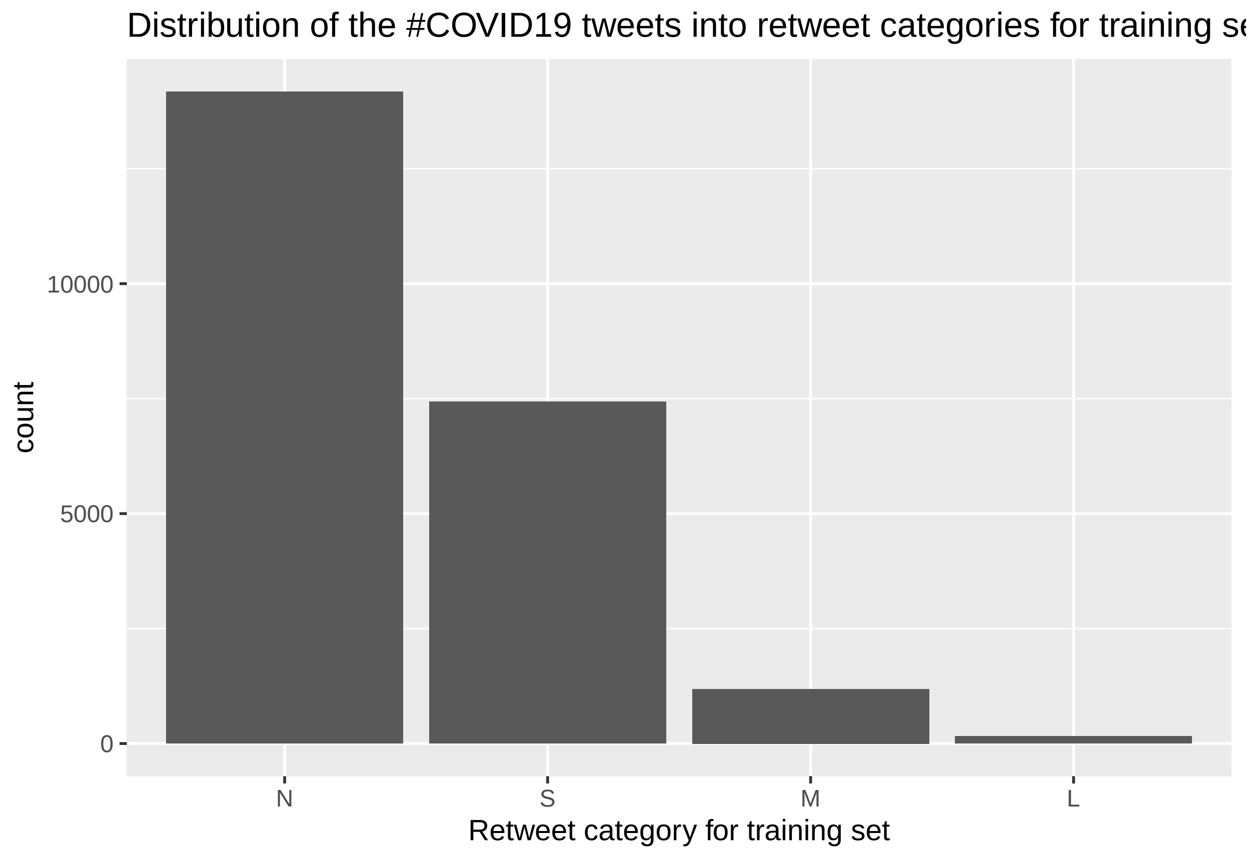
```
afinn_RK <- afinn_RK %>%
  mutate(isSmall = ifelse(retwtCAT == "S", 1, 0)) %>%
  mutate(isMedium = ifelse(retwtCAT == "M", 1, 0)) %>%
  mutate(isLarge = ifelse(retwtCAT == "L", 1, 0))
```

Split the data into train and test sets:

```
set.seed(42)
tr2 <- sample(nrow(afinn_RK),round(nrow(afinn_RK)*0.6)) # split into training and test subsets
trainRK <- afinn_RK[tr2,]
testRK <- afinn_RK[-tr2,]
```

Let's see how many tweets per category there are in our training and test samples to make sure that the distributions are representative of the total sample:

```
trainRK %>%
  ggplot(aes(retwtCAT)) +
  geom_bar() +
  xlab("Retweet category for training set") +
  ggtitle("Distribution of the #COVID19 tweets into retweet categories for training set")
```

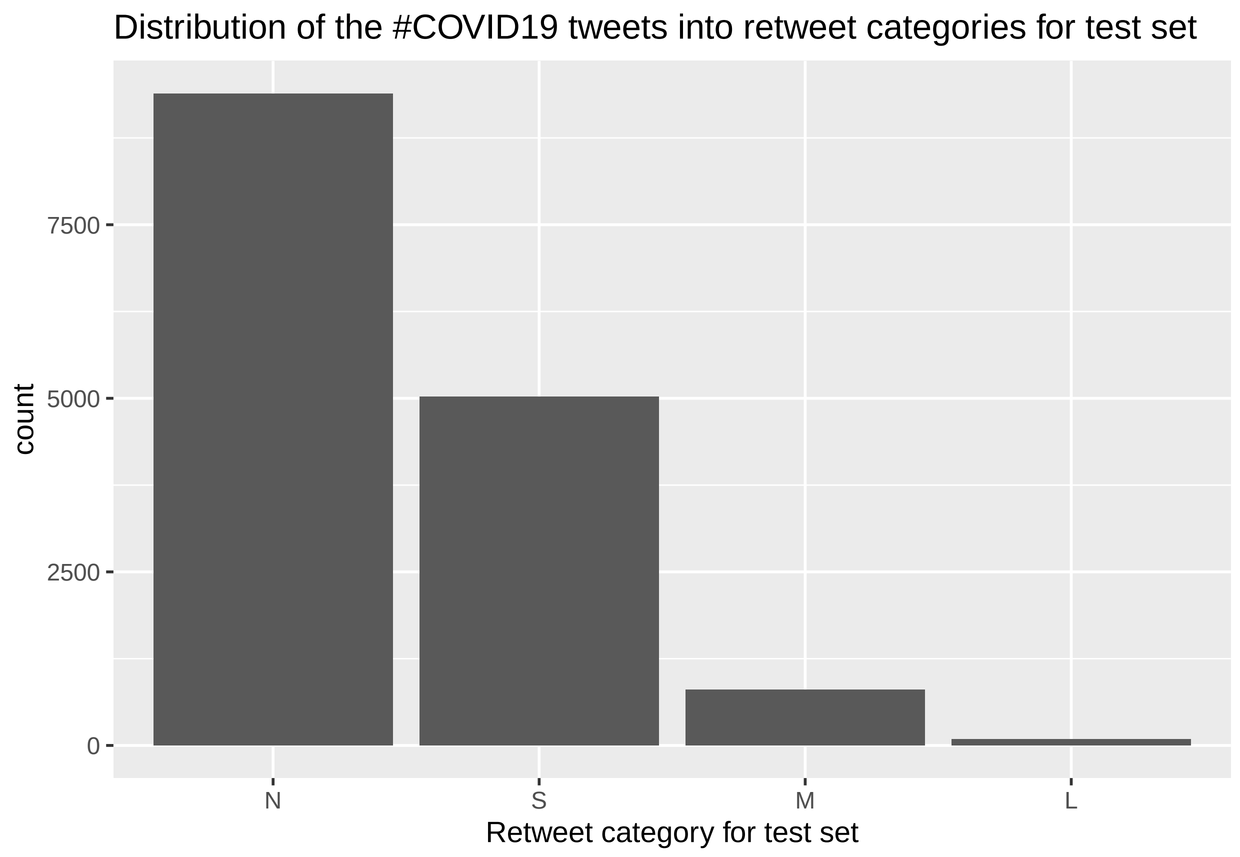


Let's also look at the exact numbers of tweets per each category:

```
trainRK %>%
  group_by(retwtCAT) %>%
  count()
```

```
## # A tibble: 4 x 2
## # Groups:   retwtCAT [4]
##   retwtCAT     n
##   <fct>    <int>
## 1 N        14176
## 2 S         7434
## 3 M         1186
## 4 L          161
```

```
testRK %>%
  ggplot(aes(retwtCAT)) +
  geom_bar() +
  xlab("Retweet category for test set") +
  ggtitle("Distribution of the #COVID19 tweets into retweet categories for test set")
```



```
testRK %>%
  group_by(retwtCAT) %>%
  count()
```

```
## # A tibble: 4 x 2
## # Groups:   retwtCAT [4]
##   retwtCAT     n
##   <fct>    <int>
## 1 N         9390
## 2 S         5022
## 3 M          800
## 4 L           92
```

As we can see from the histograms, the relative distributions look very similar, which provides strong indication that the best model according to training and benchmarking will be extendable to other future tweet analysis on this topic.

```
modelRK_LR1 <- glm(isSmall ~ followers + friends + hashtags + mentions + urls + score, data = trainRK, 
summary(modelRK_LR1)
```

```
##
```

```
## Call:
## glm(formula = isSmall ~ followers + friends + hashtags + mentions +
##     urls + score, family = binomial(link = "logit"), data = trainRK)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.0633  -0.8773  -0.8423   1.4591   1.6391
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.704e-01  2.681e-02 -32.461  < 2e-16 ***
## followers    7.885e-09  1.783e-08   0.442 0.658332
## friends      6.004e-06  1.208e-06   4.970 6.69e-07 ***
## hashtags    -4.402e-03  4.429e-03  -0.994 0.320339
## mentions     1.171e-01  1.041e-02  11.243  < 2e-16 ***
## urls         8.942e-02  2.637e-02   3.390 0.000698 ***
## score        1.162e-02  3.970e-03   2.927 0.003419 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 28913  on 22956  degrees of freedom
## Residual deviance: 28732  on 22950  degrees of freedom
## AIC: 28746
##
## Number of Fisher Scoring iterations: 4
```

Notice that friends, mentions, urls and **sentiment score** are statistically significant for predicting the tweets that get between 1-10 retweets. Positive fitted weights for those features indicate that tweets with more positive sentiments have a higher chance to be retweeted (in the "S" category).

```
modelRK_LR2 <- glm(isMedium ~ followers + friends + hashtags + mentions + urls + score, data = trainRK,
summary(modelRK_LR2)
```

```
##
## Call:
## glm(formula = isMedium ~ followers + friends + hashtags + mentions +
##     urls + score, family = binomial(link = "logit"), data = trainRK)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1015  -0.3402  -0.3010  -0.2490   3.4443
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.538e+00  6.091e-02 -41.666  < 2e-16 ***
## followers    5.408e-07  3.026e-08  17.873  < 2e-16 ***
## friends      8.558e-06  1.548e-06   5.529 3.22e-08 ***
## hashtags    -1.516e-01  1.570e-02  -9.655  < 2e-16 ***
## mentions     9.332e-02  1.896e-02   4.921 8.60e-07 ***
## urls        -2.521e-01  5.994e-02  -4.206 2.59e-05 ***
## score       -4.070e-02  8.547e-03  -4.762 1.92e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 9338.0  on 22956  degrees of freedom
## Residual deviance: 8514.2  on 22950  degrees of freedom
## AIC: 8528.2
##
## Number of Fisher Scoring iterations: 6
```

Unlike for the "S" category, all of the considered features appear to be statistically significant with different impact on the number of retweets:

- followers, friends, mentions all have positive fitted wieghts, which indicates higher chance to be in this category
- hashtags, urls and sentiment score actually have negative fitted weights which indicates that the presence of hashtags and urls leads to a lower chance to get between 11 and 100 retweets. Also notice that the negative weight for the sentiment score indicates that tweets with a negative sentiment have a higher chance to be in this category - an opposite conclusion from the "S" category.

```
modelRK_LR3 <- glm(isLarge ~ followers + friends + hashtags + mentions + urls + score, data = trainRK, 
summary(modelRK_LR3)
```

```
##
## Call:
## glm(formula = isLarge ~ followers + friends + hashtags + mentions +
##     urls + score, family = binomial(link = "logit"), data = trainRK)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0648  -0.1356  -0.1020  -0.0797   3.6410
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -4.044e+00  1.549e-01 -26.115  < 2e-16 ***
## followers    2.796e-07  2.529e-08  11.056  < 2e-16 ***
## friends      2.303e-06  1.332e-06   1.729   0.0838 .
## hashtags    -2.084e-01  4.754e-02  -4.383 1.17e-05 ***
## mentions    -1.047e-01  7.556e-02  -1.385   0.1660
## urls        -8.971e-01  1.711e-01  -5.242 1.58e-07 ***
## score       -3.336e-02  2.187e-02  -1.525   0.1272
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1918.0  on 22956  degrees of freedom
## Residual deviance: 1773.2  on 22950  degrees of freedom
## AIC: 1787.2
##
## Number of Fisher Scoring iterations: 9
```

Finally, we see that followers, hashtags and urls are statistically significant features for predicting "L" category with more followers leading to a higher probability ($w > 0$) and more hashtags and urls decreasing the probability to get many retweets ($w < 0$).

Let's look at the confusion matrices for "S", "M" and "L" categories:

```r
probS_LR1 <- predict(modelRK_LR1,testRK, type = "response")
clS_LR1 <- ifelse(probS_LR1 > 0.5, 1, 0) # predicted binary classification
confusionMatrix(factor(clS_LR1), factor(testRK$isSmall),positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 10203  4960
##          1    79    62
##
##                Accuracy : 0.6707
##                  95% CI : (0.6632, 0.6782)
##     No Information Rate : 0.6719
##     P-Value [Acc > NIR] : 0.6187
##
##                   Kappa : 0.0062
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.012346
##             Specificity : 0.992317
##          Pos Pred Value : 0.439716
##          Neg Pred Value : 0.672888
##              Prevalence : 0.328150
##          Detection Rate : 0.004051
##    Detection Prevalence : 0.009213
##       Balanced Accuracy : 0.502331
##
##        'Positive' Class : 1
##
```

```r
probM_LR2 <- predict(modelRK_LR2,testRK, type = "response")
clM_LR2 <- ifelse(probM_LR2 > 0.5, 1, 0) # predicted binary classification
confusionMatrix(factor(clM_LR2), factor(testRK$isMedium),positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 14471   756
##          1    33    44
##
##                Accuracy : 0.9484
##                  95% CI : (0.9448, 0.9519)
##     No Information Rate : 0.9477
##     P-Value [Acc > NIR] : 0.3532
##
##                   Kappa : 0.092
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.055000
##             Specificity : 0.997725
##          Pos Pred Value : 0.571429
##          Neg Pred Value : 0.950351
##              Prevalence : 0.052274
##          Detection Rate : 0.002875
##    Detection Prevalence : 0.005031
##       Balanced Accuracy : 0.526362
##
##        'Positive' Class : 1
##
```

```
probL_LR3 <- predict(modelRK_LR3,testRK, type = "response")
clL_LR3 <- ifelse(probL_LR3 > 0.5, 1, 0) # predicted binary classification
confusionMatrix(factor(clL_LR3), factor(testRK$isLarge),positive = '1')
```

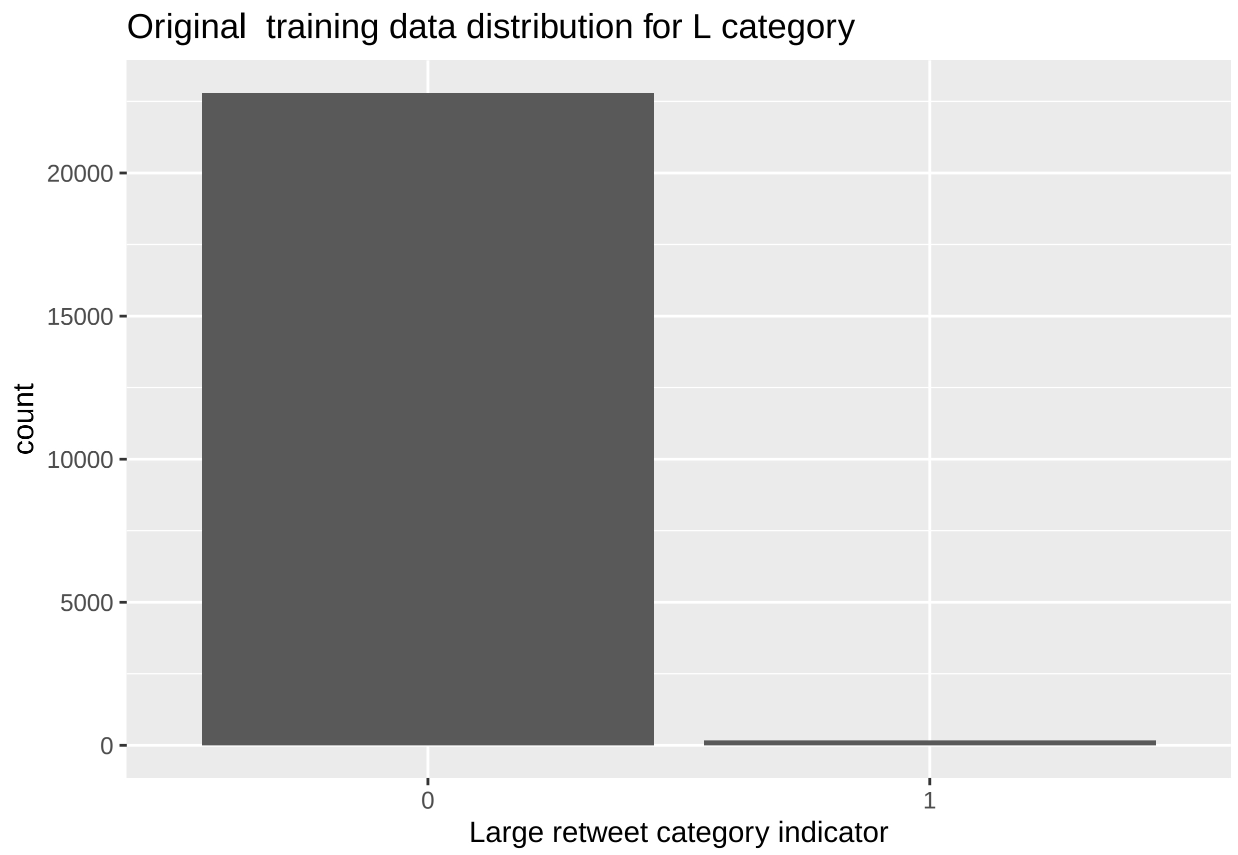
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 15212    92
##          1     0     0
##
##                Accuracy : 0.994
##                  95% CI : (0.9926, 0.9952)
##     No Information Rate : 0.994
##     P-Value [Acc > NIR] : 0.5277
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.000000
##             Specificity : 1.000000
##          Pos Pred Value :      NaN
##          Neg Pred Value : 0.993988
##              Prevalence : 0.006012
##          Detection Rate : 0.000000
##    Detection Prevalence : 0.000000
##       Balanced Accuracy : 0.500000
##
##        'Positive' Class : 1
##
```

It is important to note that while the accuracy for the logit model to predict "L" category is the highest (>99%), it is **not** a useful metric in this case because there are very little "L" category tweets (99 vs 15204 for these specific data set). Therefore, we should focus on sensitivity and specificity instead as useful metrics for predicing categories with "S", "M" and "L" indicators. My OvA (one-vs-all) logistic regression model in fact has sensitivity = 0 for the "L" class and thus is not vert useful for this test dataset.

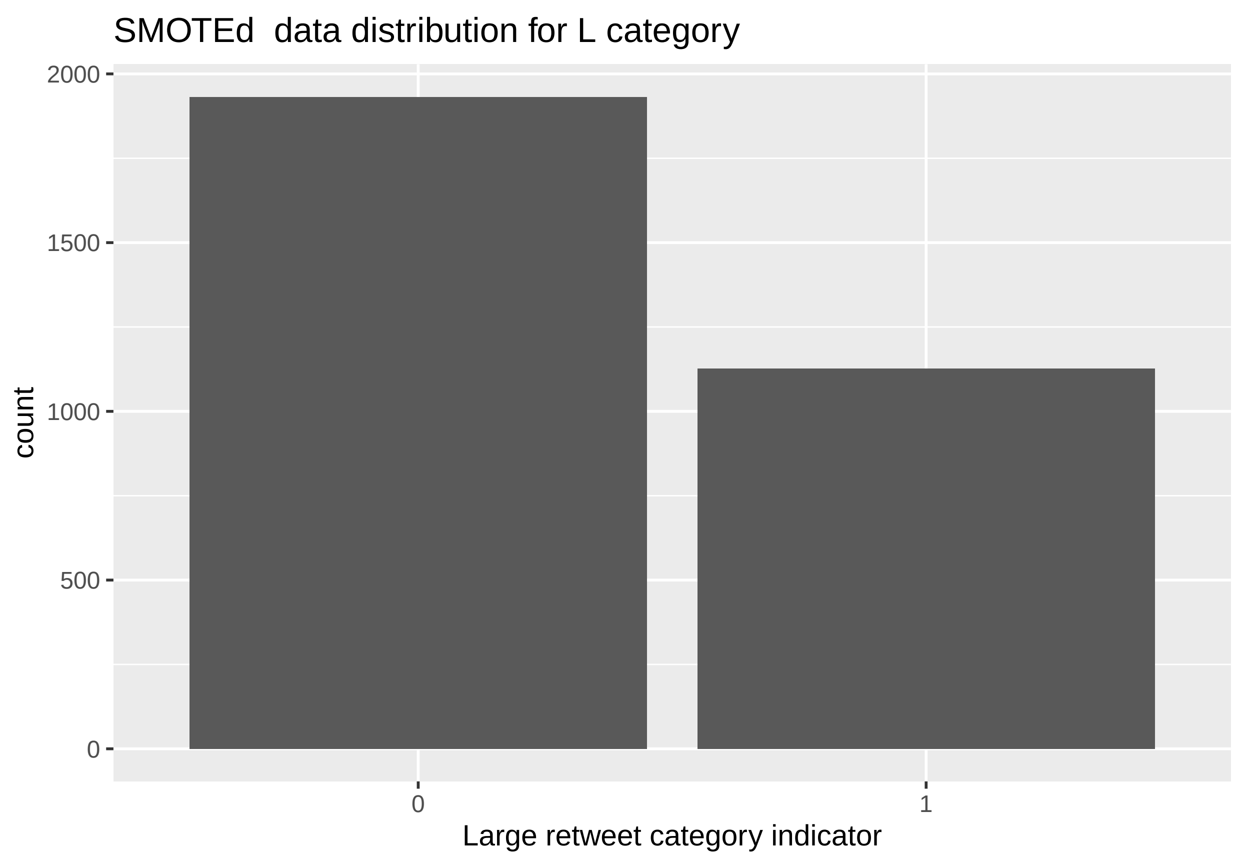
Dealing with the imbalanced class issue

Since the number of cases of "L" category (i.e. tweets that received more than 100 retweets) in the training set is less than 1%, it is not surprsing that we have trouble fitting a model that is sensitive to this category of tweets. One possibility to deal with this issue is to use the SMOTE package which undersamples the majority class and oversamples the minority class by creating synthetic minority class examples.

```r
trainLarge <- data.frame(trainRK[,c("isLarge", "followers", "friends", "hashtags", "mentions", "urls",

trainLarge$isLarge <- as.factor(trainLarge$isLarge)

trainLarge %>%
  ggplot(aes(isLarge)) +
  geom_bar() +
  xlab("Large retweet category indicator") +
  ggtitle("Original  training data distribution for L category")
```



```r
new_trainRK <- SMOTE(isLarge ~ ., trainLarge, perc.over = 600, perc.under = 200)

new_trainRK %>%
  ggplot(aes(isLarge)) +
  geom_bar() +
  xlab("Large retweet category indicator") +
  ggtitle("SMOTEd  data distribution for L category")
```



Let's train the logistic model on the new SMOTEd data set now:

```
modelRK_LR4 <- glm(isLarge ~ followers + friends + hashtags + mentions + urls + score, data = new_trainR
summary(modelRK_LR4)
```

```
##
## Call:
## glm(formula = isLarge ~ followers + friends + hashtags + mentions +
##     urls + score, family = binomial(link = "logit"), data = new_trainRK)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -5.6726  -0.8330  -0.4889   0.9928   2.3540
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.904e-01  8.791e-02   4.441 8.96e-06 ***
## followers    1.235e-06  1.130e-07  10.925  < 2e-16 ***
## friends      3.052e-05  4.148e-06   7.357 1.88e-13 ***
## hashtags    -3.413e-01  2.692e-02 -12.678  < 2e-16 ***
## mentions    -1.817e-01  4.227e-02  -4.299 1.72e-05 ***
## urls        -8.474e-01  8.749e-02  -9.685  < 2e-16 ***
## score       -4.580e-02  1.186e-02  -3.863 0.000112 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 4026.3  on 3058  degrees of freedom
## Residual deviance: 3123.6  on 3052  degrees of freedom
## AIC: 3137.6
##
## Number of Fisher Scoring iterations: 7
```

```
testLarge <- data.frame(testRK[,c("isLarge", "followers", "friends", "hashtags", "mentions", "urls", "se

testLarge$isLarge <- as.factor(testLarge$isLarge)


probL_LR4 <- predict(modelRK_LR4,testLarge, type = "response")
clL_LR4 <- ifelse(probL_LR4 > 0.5, 1, 0) # predicted binary classification
confusionMatrix(factor(clL_LR4), factor(testLarge$isLarge),positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 13704    43
##          1  1508    49
##
##                Accuracy : 0.8987
##                  95% CI : (0.8938, 0.9034)
##     No Information Rate : 0.994
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0486
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.532609
##             Specificity : 0.900868
##          Pos Pred Value : 0.031471
##          Neg Pred Value : 0.996872
##              Prevalence : 0.006012
##          Detection Rate : 0.003202
##    Detection Prevalence : 0.101738
##       Balanced Accuracy : 0.716738
##
##        'Positive' Class : 1
##
```

```
F1_Score(factor(testLarge$isLarge), factor(clL_LR4),positive = '1')
```

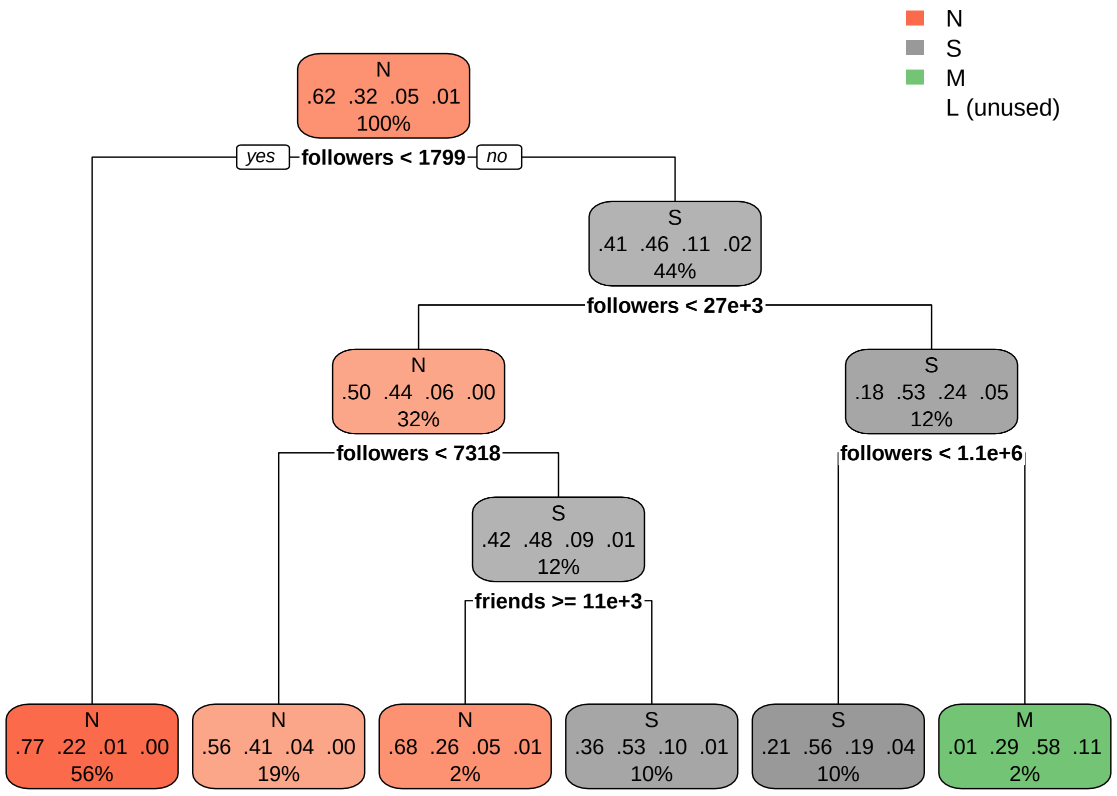
```
## [1] 0.05942996
```

Notice that our new model has correctly predicted 49 tweets with >100 retween on the test dataset it has not seen before. Compare that to the previous prediction of zero. It is important to note that now our model is more sensitive to the properties of the features that are valuable for predicting tweets with a large number of retweets. we cannot obtain such a results by simply reducing the cutoff threashold below 0.5 in the original logisitc regression model (believe me I tried).

Decision trees to determine retweet categories

I train the recursive partitioning model on the training dataset with 60% of the data we set aside before:

```
modelRK_DT <- rpart(retwtCAT ~ followers + friends + score + mentions + hashtags + urls, data = trainRK

predRK_DT <- predict(modelRK_DT,testRK, type = "class")

rpart.plot(modelRK_DT)
```



```
confusionMatrix(predRK_DT, testRK$retwtCAT)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction     N    S    M    L
##            N 8452 3154  197    6
##            S  935 1798  459   61
##            M    3   70  144   25
##            L    0    0    0    0
##
## Overall Statistics
##
##                 Accuracy : 0.6792
##                   95% CI : (0.6717, 0.6866)
##     No Information Rate : 0.6136
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                  Kappa : 0.2964
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: N Class: S Class: M Class: L
## Sensitivity            0.9001   0.3580 0.180000 0.000000
## Specificity            0.4324   0.8585 0.993243 1.000000
## Pos Pred Value         0.7157   0.5527 0.595041      NaN
## Neg Pred Value         0.7316   0.7325 0.956447 0.993988
## Prevalence             0.6136   0.3281 0.052274 0.006012
## Detection Rate         0.5523   0.1175 0.009409 0.000000
## Detection Prevalence   0.7716   0.2126 0.015813 0.000000
## Balanced Accuracy      0.6662   0.6083 0.586622 0.500000
```

We can see that follower and friends are the only two features used here. In fact it looks like a simple decision tree performs much better in predicting tweets in "S" and "M" categories.

KNN classifier

I was interested in seeing how k-nearest-neighbors classifier will perform on this multivariate classification challenge for retweet categories. However, before we can efficiently apply it on our tweet data set, I needed to transform the tweet features since they are of very diferent scales: while mentions and urls are usually a small number, followers can be in the hundreds or thousands. Moreover, sentiment score can be positive or negative while all the other variables can only take values larger than zero. Thus, I rescaled and centered the features before applying knn classification:

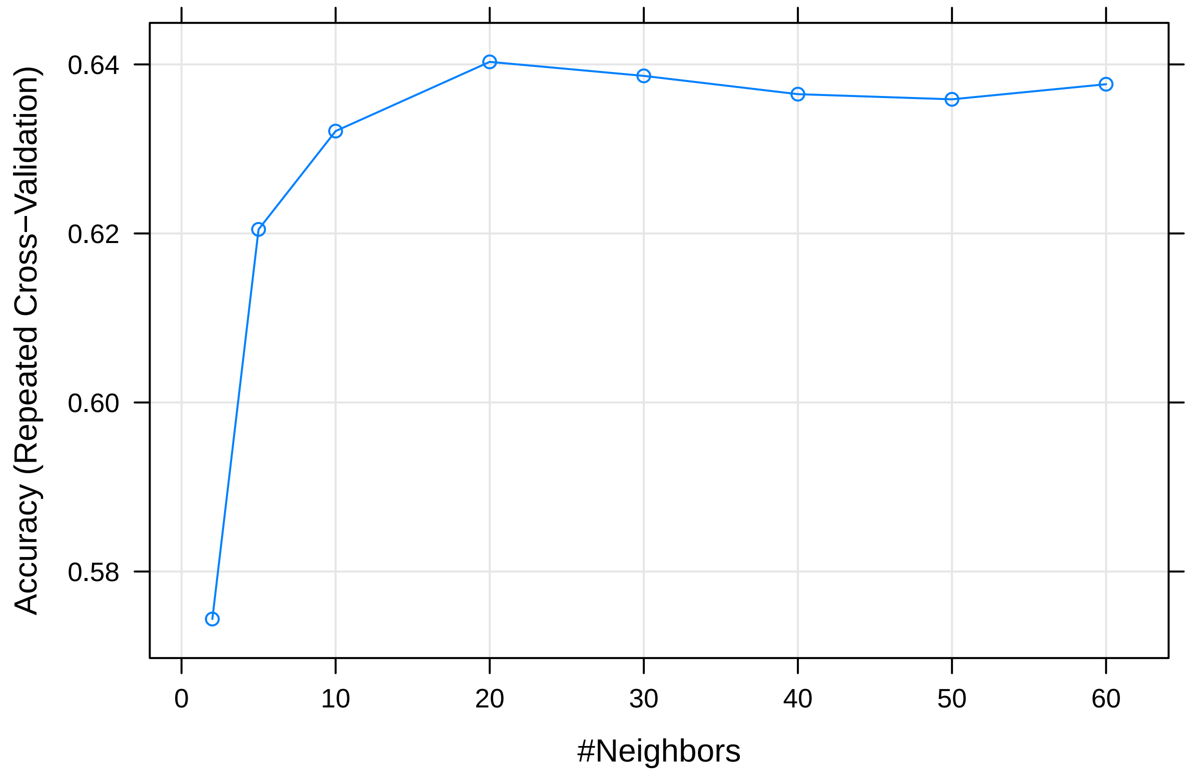
```r
trCtrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

knn_fit <- train(retwtCAT ~ followers + friends + score + mentions + hashtags + urls, data = trainRK, me
                 trControl = trCtrl,
                 preProcess = c("center","scale"),
                 tuneGrid = expand.grid(k = c(2,5,10,20,30,40,50,60)))
```

Let's look at the results now:

```r
knn_fit
```

```
## k-Nearest Neighbors
##
## 22957 samples
##     6 predictor
##     4 classes: 'N', 'S', 'M', 'L'
##
## Pre-processing: centered (6), scaled (6)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 20661, 20663, 20663, 20662, 20662, 20661, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    2  0.5743780  0.1495516
##    5  0.6204784  0.1832739
##   10  0.6321090  0.1774527
##   20  0.6402975  0.1661460
##   30  0.6386423  0.1469093
##   40  0.6364790  0.1323169
##   50  0.6358693  0.1222246
##   60  0.6376556  0.1194040
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 20.
```

```
plot(knn_fit)
```



It looks like using 30 neighbors provides the highest accuracy. However, we need to see how our model performs on the test data.

```
test_pred <- predict(knn_fit, testRK)
confusionMatrix(test_pred, testRK$retwtCAT)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    N    S    M    L
##          N 8574 3897  424   29
##          S  816 1075  252   34
##          M    0   50  124   29
##          L    0    0    0    0
##
## Overall Statistics
##
##                Accuracy : 0.6386
##                  95% CI : (0.6309, 0.6462)
##     No Information Rate : 0.6136
##     P-Value [Acc > NIR] : 9.079e-11
##
##                   Kappa : 0.1682
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: N Class: S Class: M Class: L
## Sensitivity            0.9131  0.21406 0.155000 0.000000
## Specificity            0.2645  0.89282 0.994553 1.000000
## Pos Pred Value         0.6634  0.49380 0.610837      NaN
## Neg Pred Value         0.6571  0.69932 0.955235 0.993988
## Prevalence             0.6136  0.32815 0.052274 0.006012
## Detection Rate         0.5602  0.07024 0.008102 0.000000
## Detection Prevalence   0.8445  0.14225 0.013265 0.000000
## Balanced Accuracy      0.5888  0.55344 0.574777 0.500000
```

Caret library makes it easy to try other machine learning methods. RandomForest performed well for the binary classification problem whether tweet will get retweeted or not so I excpected it to do well here to. Let's see how it performs. Notice that I chose not too rescale the features since randomforest should be robust against different scales in the variables.

```
set.seed(42)

trCtrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3, search = 'grid')

rf_fit <- train(retwtCAT ~ followers + friends + score + mentions + hashtags + urls, data = trainRK, me
                 trControl = trCtrl,
                 metric = 'Accuracy',
                 tuneGrid = expand.grid(mtry = c(1,2,3,4,5,6)))
```

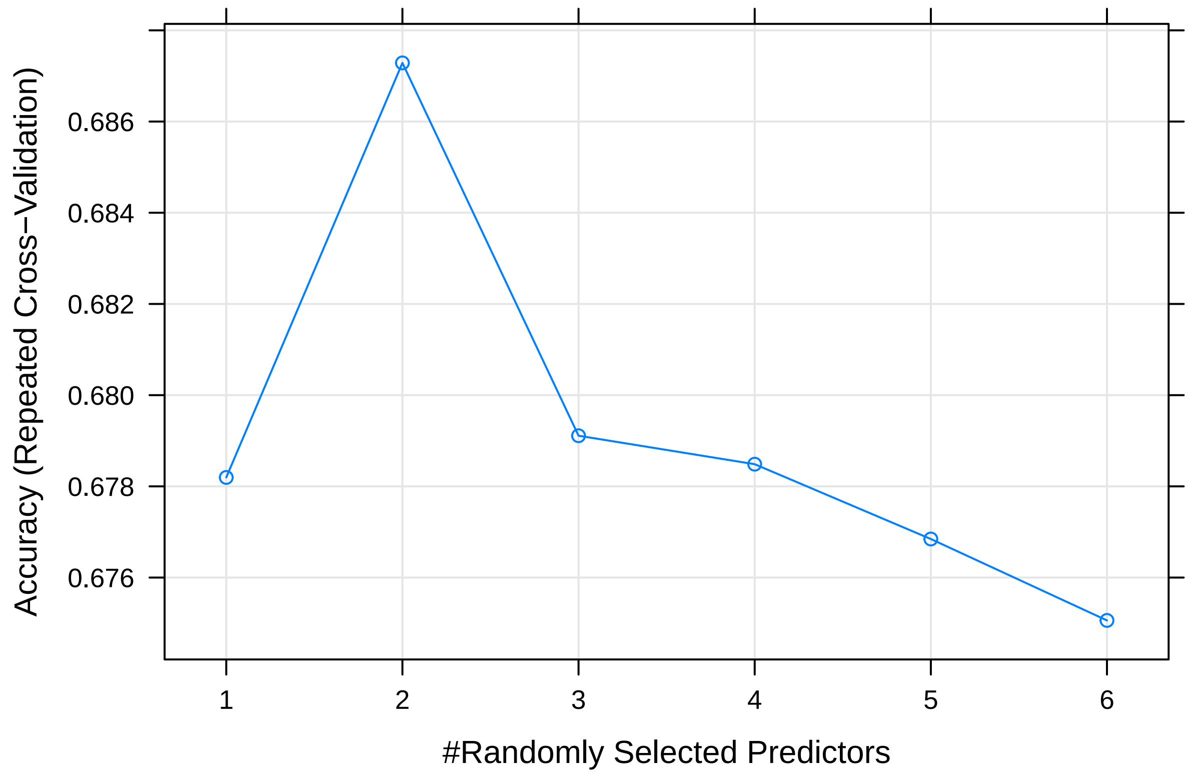
Let's look at the results:

```
rf_fit
```

```
## Random Forest
##
## 22957 samples
##     6 predictor
##     4 classes: 'N', 'S', 'M', 'L'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 20662, 20661, 20660, 20661, 20662, 20661, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   1     0.6781958  0.2686820
##   2     0.6872857  0.3287544
##   3     0.6791106  0.3223174
##   4     0.6784863  0.3227477
##   5     0.6768454  0.3203572
##   6     0.6750601  0.3178618
```

```
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```r
plot(rf_fit)
```



From the cross-validation we conclude that using 2 selected predictors performs the best. Let's see what the out-of-sample error is:

```r
test_pred <- predict(knn_fit, testRK)
confusionMatrix(test_pred, testRK$retwtCAT)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    N    S    M    L
##          N 8569 3886  419   32
##          S  821 1087  258   30
##          M    0   49  123   30
##          L    0    0    0    0
##
## Overall Statistics
##
##                Accuracy : 0.639
##                  95% CI : (0.6313, 0.6466)
##     No Information Rate : 0.6136
##     P-Value [Acc > NIR] : 4.689e-11
##
```

```
##                   Kappa : 0.1697
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: N Class: S Class: M Class: L
## Sensitivity            0.9126  0.21645 0.153750 0.000000
## Specificity            0.2667  0.89214 0.994553 1.000000
## Pos Pred Value         0.6640  0.49499 0.608911      NaN
## Neg Pred Value         0.6576  0.69980 0.955172 0.993988
## Prevalence             0.6136  0.32815 0.052274 0.006012
## Detection Rate         0.5599  0.07103 0.008037 0.000000
## Detection Prevalence   0.8433  0.14349 0.013199 0.000000
## Balanced Accuracy      0.5896  0.55429 0.574152 0.500000
```

Summary of the retweet degree prediction

By performing one-vs-all logistic regression I identified that different features are important for predicting retweet outcome classes: for example, urls, hashtags and followers are statistically significant in order to predict tweets with >100 retweets, while mentions, friends and urls are the most important features for predicting the 1-10 retweet category. While I found that other tweet parameters influence the number of retweets much more than the sentiment score, analyzing the sentiment of the tweets can still be important for understanding the public opinion and it's change over time. Below I look at the tweet sentiments using "bing" lexicon instead.

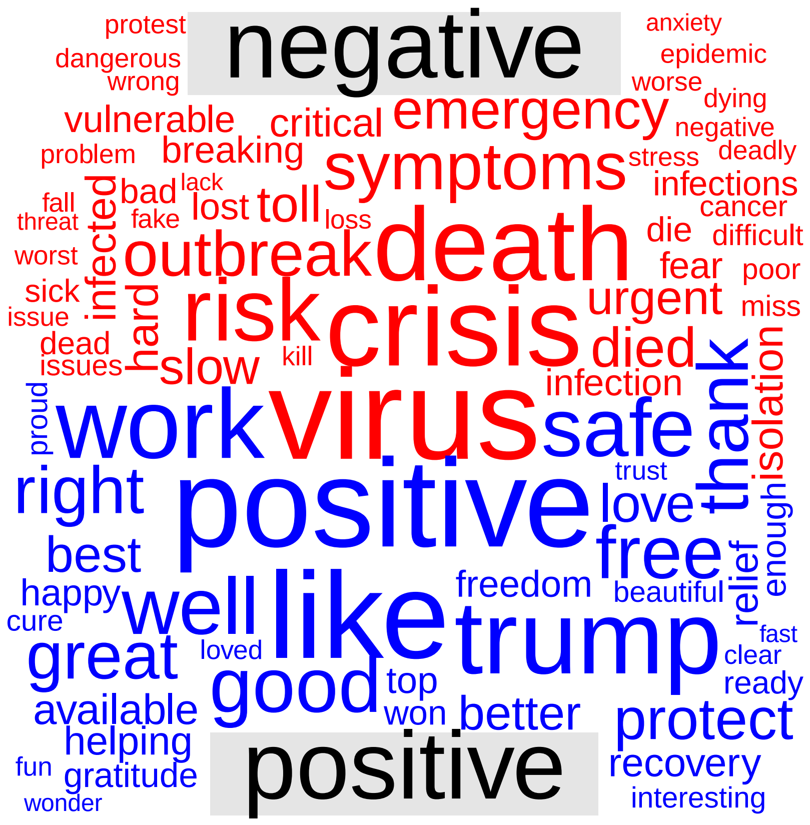
Use Bing lexicon for sentiment analysis

While "bing" lexicon only specifies whether a word is positive or negative it still can be useful for sentiment analysis

```
twts_bing <- twts_clean %>%
  inner_join(get_sentiments("bing"), by = 'word')
```

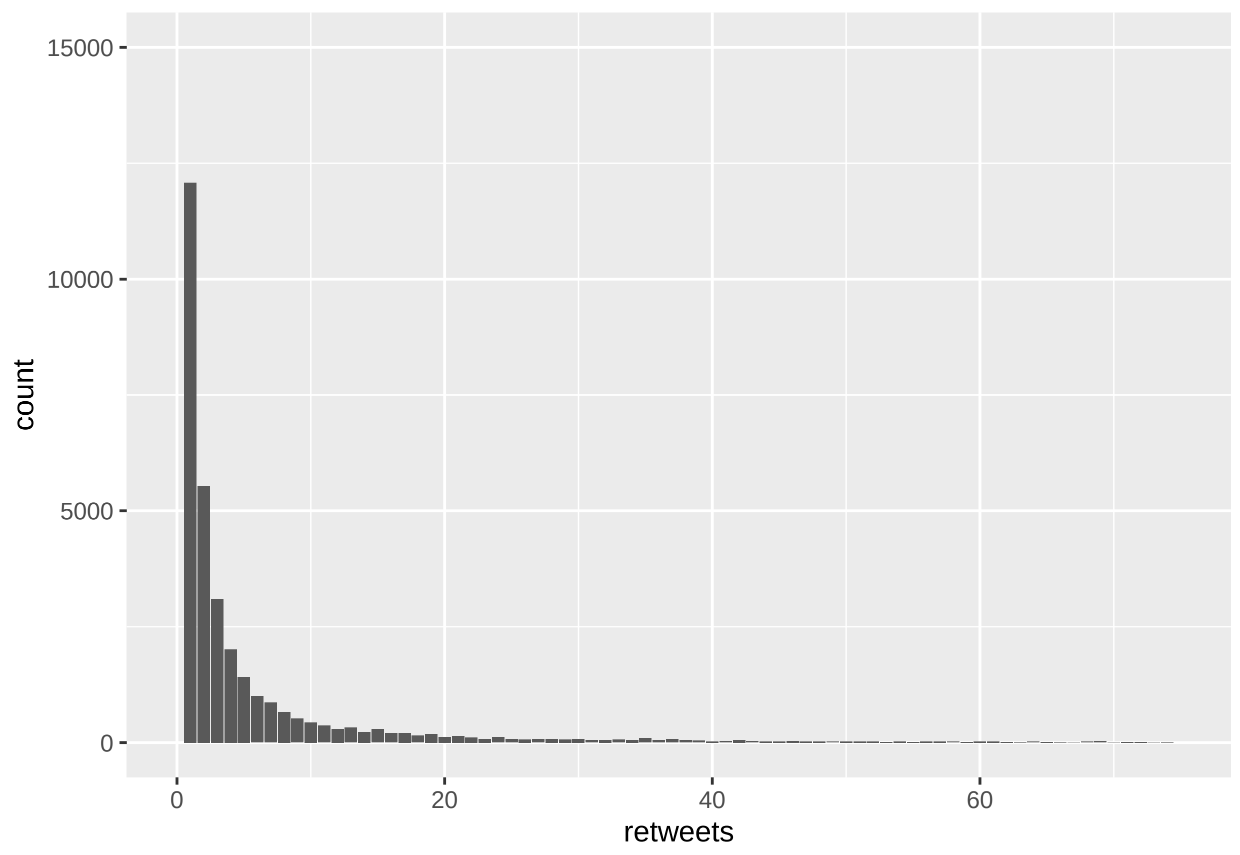
Let's look at the most popular positive and negative words:

```
twts_bing %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("red","blue"),max.words =100)
```



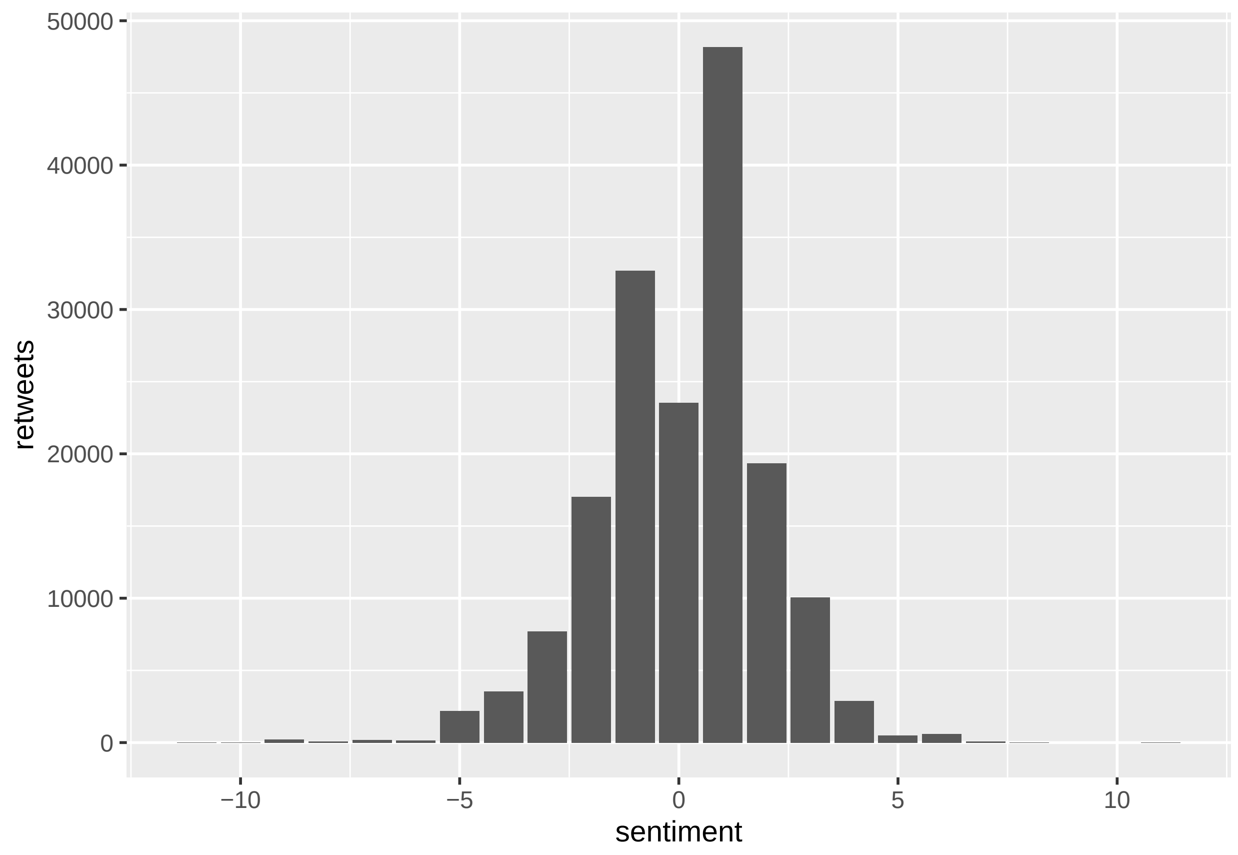
Notice that "trump" is among the most used words in positive tweets because "bing" lexicon asigns positive meaning to "trump". However, in the current situation, most likely all those tweets refer to President Trump, some in the positive context while others in the negative.

```
twts_bing %>%
  ggplot(aes(retweets)) +
  geom_bar() +
  xlim(0,75) +
  ylim(0,15000)
```

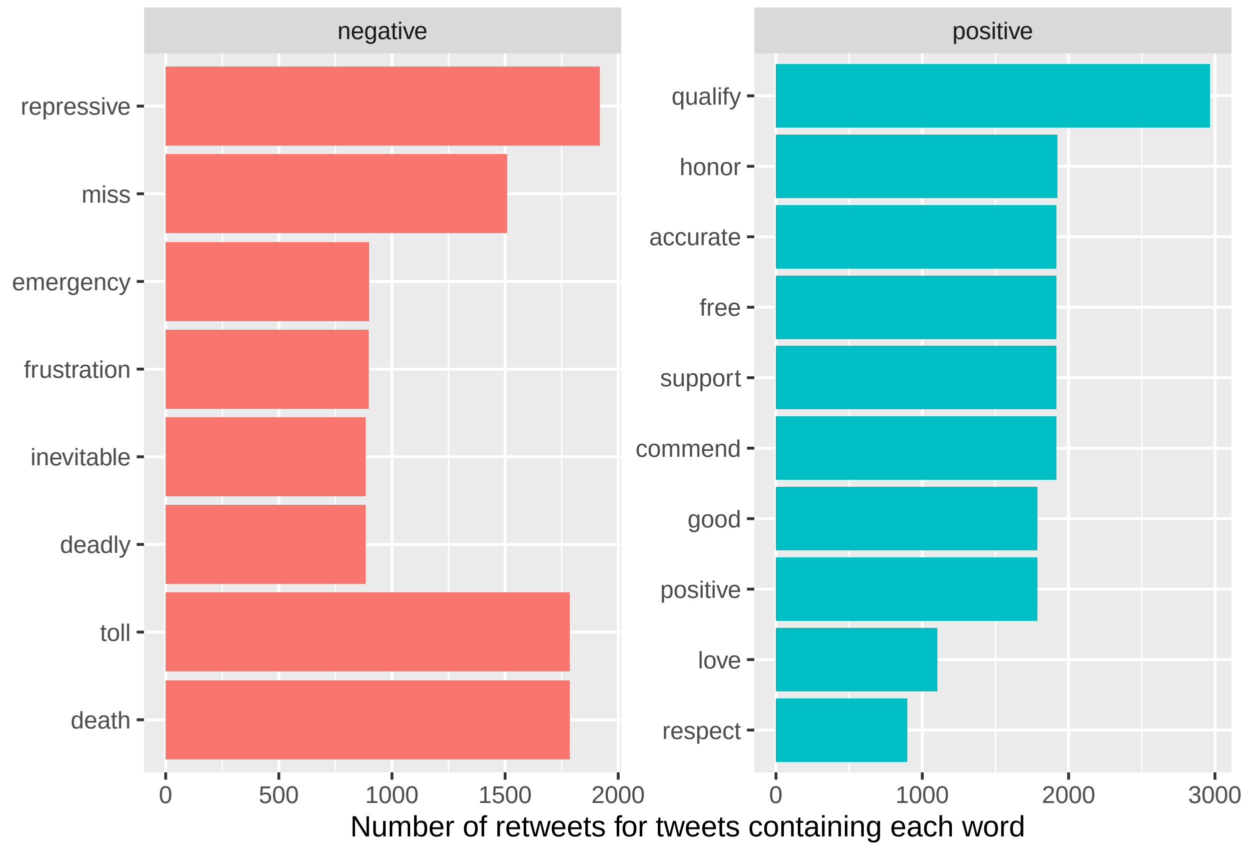


Raw distribution of the retweets is displayed above. Now let's look at the distribution of the retweets vs tweet sentiment as evaluated by the bing lexicon.

```
retwts_bing_sent <- twts_bing %>%
  group_by(id,retweets,followers) %>%
  # filter(retweets > 5) %>%
  count(id,retweets,sentiment) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment = positive - negative) %>%
  mutate(word_cnt = positive + negative)

retwts_bing_sent %>%
  ggplot(aes(sentiment,retweets)) +
  geom_bar(stat = "identity")
```

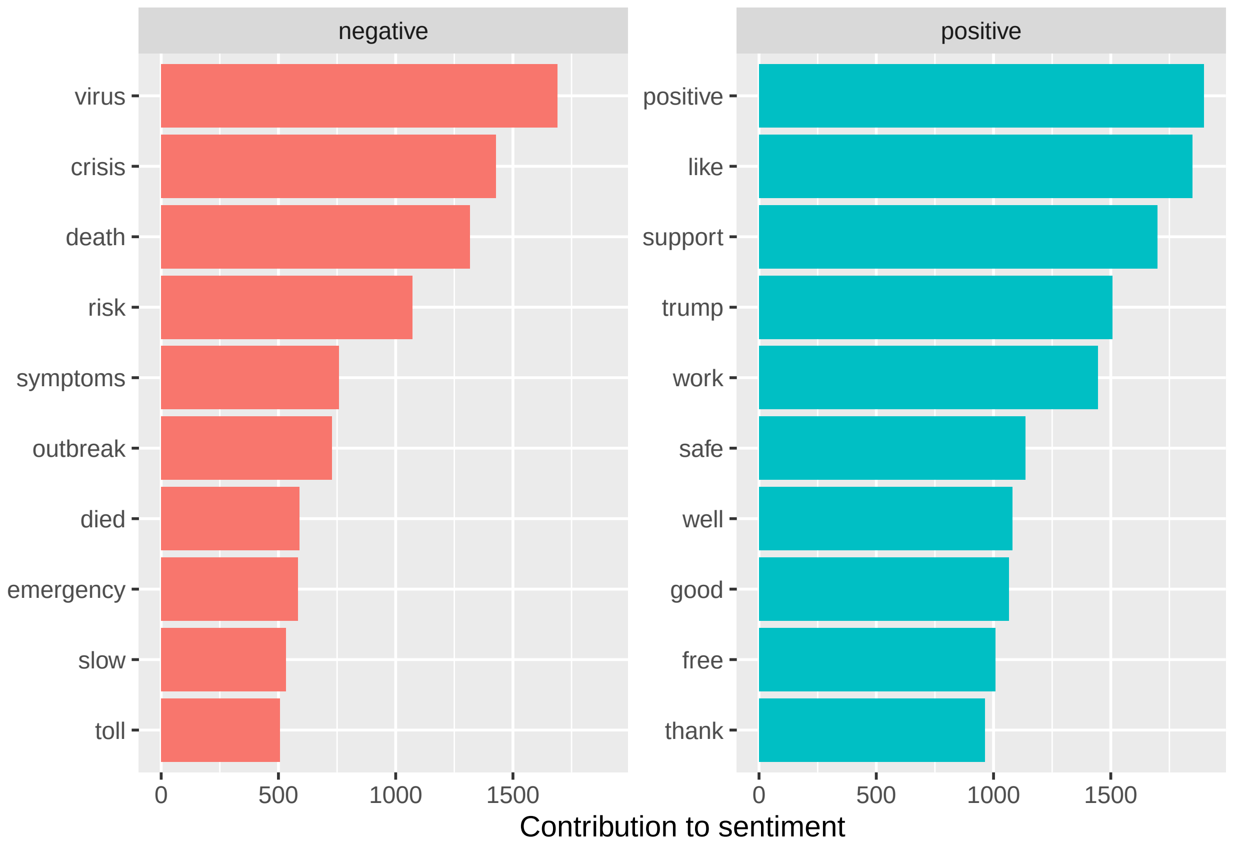


```
twts_bing %>%
  group_by(sentiment) %>%
  top_n(10, retweets) %>%
  arrange((retweets)) %>%
  ungroup() %>%
  mutate(word = factor(word, unique(word))) %>%
  ungroup() %>%
  ggplot(aes(word, retweets, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ sentiment, scales = "free", ncol = 2) +
  coord_flip() +
  labs(x = NULL,
       y = "Number of retweets for tweets containing each word")
```



```
bing_word_counts <- twts_bing %>%
  count(word, sentiment, sort = TRUE) %>%
  ungroup()

bing_word_counts %>%
  group_by(sentiment) %>%
  top_n(10) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(y = "Contribution to sentiment",
       x = NULL) +
  coord_flip()
```

```
## Selecting by n
```


negative
positive
virus
crisis
death
risk
symptoms
outbreak
died
emergency
slow
toll
positive
like
support
trump
work
safe
well
good
free
thank
0
500
1000
1500
Contribution to sentiment