Modeling Elon Musk's Tweet Like Count Using Linear Models

Zach Hinz, Joshua Jansen-Montoya

2022-12-09

Abstract

In our report we attempt to answer the question of what factors affect the like count that Elon Musk gets on his tweets and whether or not we can model the relationships between different Twitter metrics and the like count of a singular tweet using a linear relationship. In doing so, we map different non-numeric data to numeric values and then fit several different linear models to the data with the intention of answering these motivating questions. As a result of our analysis of these models, we found that none of our linear models provided an perfect linear model for our data with several being affected by the fact that our data set holds many outliers and points which we interpret to be more viral tweets which makes sense give Elon's frequency of tweeting and tendency towards virality. From our analysis and attempts to transform and better fit our model to the data, we decided on our Box Cox transformed linear model to be our best model due to its fulfillment of our assumptions when using a linear model and its seemingly solid fitting to our data with regards to its standard residual errors and R^2 value. We still hold many caveats for the use of the model due to the weaker predictive ability of our model. As such, we warn readers about using the model and about interpreting the coefficients as true predictors.

Introduction

In our investigation, we are interested in understanding how the like count of an individuals tweets can be affected by their other actions on the app, Twitter. More specifically, given the recent relevance of Elon Musk's purchase of Twitter and his status as an inflammatory figure who has both been on the platform for an extensive amount of time, who is currently active and who garners a large amount of attention on the platform, we thought he would be a perfect case study of this phenomenon. Our motivating question for our investigation was to understand how do different metrics of Elon's tweets e.g. their retweet count, their length, and the AI (RoBERTa) generated sentiment of the tweet which attempts to provide a metric of the emotional sentiment of the tweet. We attempted to model the amount of likes that a tweet would obtain using different linear models with the end goal of producing a linear model that can accurately predict the number of likes that Elon Musk would get on a particular tweet if we knew the values of the predictor variables for our linear models. For our analysis of our models, we will be focusing on the R^2, standard residual errors, and significance of our model along with trying to ensure that our model meets all of our required assumptions when using a linear model.

Data Set

Our data set of interest comes from the website, Kaggle.com and is the data set, "Elon Tweet Sentiment 10/28/22". This data set is the collection of different Twitter metrics on Elon's twitter usage since he first was verified on the platform in 2010. The data consists of 17,445 rows and 20 columns with the rows being our data entries and our columns being our variables. Our variables are: ..1 (numeric), Unnamed 0.1 (numeric), Unnamed 0 (numeric), Datetime (char), Tweet id (numeric), Text (char), Username (char), Location (char), Reply Count (numeric), Like Count (numeric), Retweet Count (numeric), Language (char), Twitter Access Point (char), Follower Count (numeric), Friends Count (numeric), Verified (bool), Date (numeric), Mentions (char), Sentiments (char/numeric). We take that each entry is independent inherently since we are assuming that each Tweet would not impact the following Tweets. One thing that we found noticeable in our data was that we have a number of different types of our variable that may make our linear model a bit more

complex/that we could find a way to produce a numeric quantity that encodes the information held in each entry in such a numeric variable. The variables that we chose to do this for were Text, Username, Location, Language, Twitter Access Point, Verified, Date, Mentions, and Sentiments. Therefore, we began by mapping Text to the number of characters in each tweet, or the length of the text:

```
ElonTweets_Sentiment_10_28_22 <- read.csv("~/Desktop/fall\ 22/math158/ElonTweets(Sentiment) 10-28-22.
ElonTweets_Sentiment_10_28_22 <- as.data.frame(ElonTweets_Sentiment_10_28_22)
textLength <- vector(length=length(ElonTweets_Sentiment_10_28_22$Text))
for (x in 1:length(ElonTweets_Sentiment_10_28_22$Text)) {
textLength[x] = nchar(ElonTweets_Sentiment_10_28_22$Text[x])
}
```

Further, for Mentions, we mapped if someone was mentioned (mentioned using the @ character) to 1 and if no one was mentioned, if the username was "elonmusk" to 1 and 0 if not, if the location was "Twitter HQ" to 1, 0 if not, and if the user was verified to 1 and 0 if not verified using the following code,

```
mentions <- ifelse(ElonTweets_Sentiment_10_28_22$mentions == "_", 0, 1)
userName <- ifelse(ElonTweets_Sentiment_10_28_22$Username == "elonmusk", 1, 0)
location <- ifelse(ElonTweets_Sentiment_10_28_22$location == "Twitter HQ", 1, 0)
verified <- ifelse(ElonTweets_Sentiment_10_28_22$verified == TRUE, 1, 0)
```

Now, looking at the distribution of the different languages of Elon's Tweets, we can designate a general number for all languages that occur more than 50 times (since "en" occurs 15008 times) according to the following relation: "en": 1, "de": 2, "fr": 3, "qam": 4, "qme": 5, "tl": 6, "und": 7, "zxx": 8, other: 0, giving us,

```
language <- ifelse(ElonTweets_Sentiment_10_28_22$language == "en", 1, ifelse(ElonTweets_Sentiment_10_
```

Now to conver the Date column, we can find that has occurred since we saw Elon's first text on the platform. To find the time between the first tweet ("2010-06-04") and each of his tweets, we can map each date to the time since this first tweet as follows:

```
timeSinceFirstTweet <- as.Date(as.character(ElonTweets_Sentiment_10_28_22$Date), format="%Y-%m-%d")-
                as.Date(as.character("2010-06-04"), format="%Y-%m-%d")
timeSinceFirstTweet<-as.numeric(timeSinceFirstTweet)
```

Now we can convert the column Access Point to numeric by noticing that the most common access points that Elon uses are Instagram, iPhone, Web App, and Web Client. Thus, we can map the different Twitter access points Elon has used according to the following; Instagram: 1, iPhone: 2, Web App: 3, Web Client: 4, other: 0.

```
accessPoint <- ifelse(ElonTweets_Sentiment_10_28_22$Twitter.Access.Point == "Instagram", 1,  ifelse(E
```

For our sentiment analysis, we can make a mapping from our sentiments output to a number scale from -100 to 200 where -100 - 0 are designated for negative sentiments (-100 being most extreme) 0-100 for neutrality (0 being most neutral), 100-200 being positive (200 being most positive). We do that as follows,

```
sentiments <- vector(length=length(ElonTweets_Sentiment_10_28_22$sentiment))
for (x in 1:length(ElonTweets_Sentiment_10_28_22$sentiment)) {
  if (substring(ElonTweets_Sentiment_10_28_22$sentiment[x], 3, 5) == "pos") {
    sentiments[x] = suppressWarnings(as.numeric(substring(ElonTweets_Sentiment_10_28_22$sentiment[x],
  } else if (substring(ElonTweets_Sentiment_10_28_22$sentiment[x], 3, 5) == "neu") {
   sentiments[x] = 100- 100*suppressWarnings(as.numeric(substring(ElonTweets_Sentiment_10_28_22$senti
 } else {
   sentiments[x] = -1* 100 * suppressWarnings(as.numeric(substring(ElonTweets_Sentiment_10_28_22$sent
}
}
```

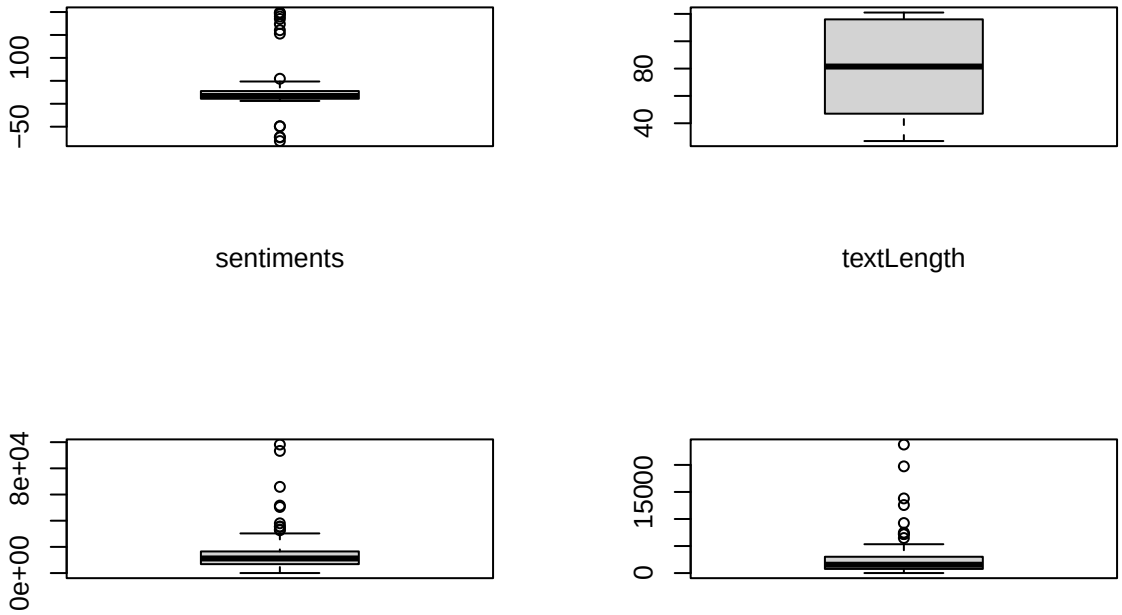
Now we will rebind all of our variables of interest into a new data frame to be used for our experiments,

leaving out ..1 (numeric), Unnamed 0.1 (numeric), Unnamed 0 (numeric), and Datetime (char) since the first three do not hold relevant information as they do not help us answer our question due to the fact that they are unlabeled, and the Datetime being left out due to the fact that the variable holds the date and time of tweet, which is another form of our Date variable that we just created a mapping for. Thus, we bind our remaining variables together to a new data frame for our analysis.

```
tweetId <- ElonTweets_Sentiment_10_28_22$Tweet.Id
replyCount <- ElonTweets_Sentiment_10_28_22$reply.count
retweetCount <- ElonTweets_Sentiment_10_28_22$retweet.count
likeCount <- ElonTweets_Sentiment_10_28_22$like.count
friendCount <- ElonTweets_Sentiment_10_28_22$Friends.Count
followerCount <- ElonTweets_Sentiment_10_28_22$Follower.Count
df <- data.frame(tweetId , replyCount , retweetCount , likeCount , friendCount , followerCount , senti
df <- na.omit(df)
```

Now that we have numeric variables that we can use to analyze using some of our different techniques, we can consider a few of the box plots for the variables that we anticipate would be significant and thus in our eventual linear model, as well as our response variable, "likeCount".

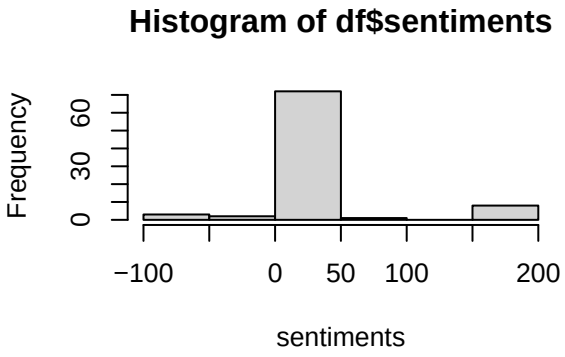
```
par(mfrow=c(2,2))
boxplot(df$sentiments, xlab = "sentiments")
boxplot(df$textLength, xlab = "textLength")
boxplot(df$likeCount, xlab = "likeCount")
boxplot(df$retweetCount, xlab = "retweetCount")
```

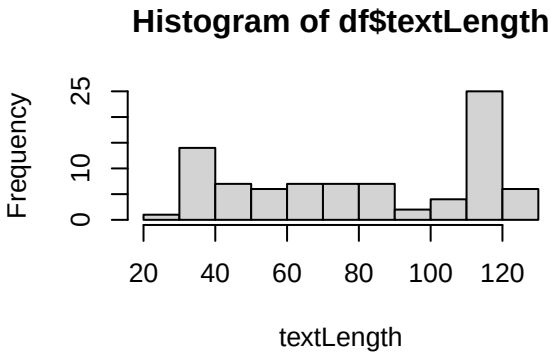


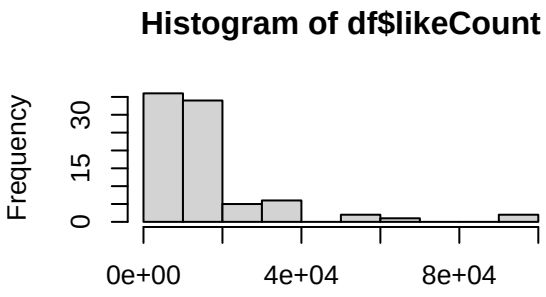
From these initial diagnostic plots, we can see that it seems like we have some points that may be extreme outliers, as noticeable by likeCount and retweetCount that may need to be removed in our later parts of analysis in order to best fit our results and that we may want to consider different model choices that would help us help reduce the impact of our outlier values as much as possible. We can also see this in the following histogram plots of the same variables.

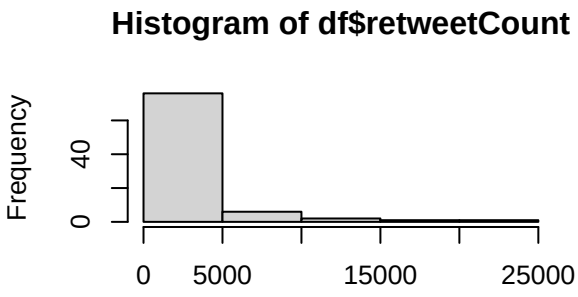
```
par(mfrow=c(2,2))
hist(df$sentiments, xlab = "sentiments")
hist(df$textLength, xlab = "textLength")
```

```
  hist(df$likeCount, xlab = "likeCount")
  hist(df$retweetCount, xlab = "retweetCount")
```









Once again, although we do not have what seems to be a large number of values that could be outliers, it appears that there are some of high magnitude that we may anticipate being high leverage points. We can also note that we do not have a clear type of distribution to our data, with some variables having a left skew. However, for all of the variables that we have in our box plots, we can see that none seem to be normally distributed.

Analysis

In our analysis, we chose to begin with an Ordinary Least Squares linear model using all the variables in our constructed data fram, df from above. Before doing so, we choose to seperate out our data into our training data and our prediction data sets. As we have ≈ 17000 different entries, we decided it would be appropriate to allocate ≈ 1000 of our entries for our prediction dataframe, dfPredict, and the remaining value for our training dataframe, dfTrain. Thus, in doing so, we obtained our first linear model, lmod,

```
  dfTrain <- df[-seq(1., NROW(df), by = 17),]
  dfPredict <- df[seq(1., NROW(df), by = 17),]
  lmod <- lm(likeCount ~ tweetId + replyCount + retweetCount + friendCount + followerCount + sentiments
summary(lmod)
```

```
##
## Call:
## lm(formula = likeCount ~ tweetId + replyCount + retweetCount +
##     friendCount + followerCount + sentiments + accessPoint +
##     textLength + mentions + location + verified + userName, data = dfTrain)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -21276.9  -2284.9   -551.7   1822.2  21220.2
##
## Coefficients: (5 not defined because of singularities)
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    4.000e+08  1.068e+09   0.374 0.709233
## tweetId        4.706e-14  1.350e-14   3.485 0.000843 ***
## replyCount     3.338e+00  2.156e+00   1.548 0.125922
## retweetCount   3.514e+00  3.140e-01  11.193  < 2e-16 ***
## friendCount           NA         NA      NA       NA
## followerCount -3.618e+00  9.664e+00  -0.374 0.709202
## sentiments     8.153e+00  1.468e+01   0.555 0.580448
## accessPoint           NA         NA      NA       NA
## textLength     1.758e+01  2.291e+01   0.767 0.445399
## mentions      -1.239e+02  3.756e+03  -0.033 0.973774
## location              NA         NA      NA       NA
## verified              NA         NA      NA       NA
## userName              NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6210 on 72 degrees of freedom
## Multiple R-squared:  0.8832, Adjusted R-squared:  0.8719
## F-statistic:  77.8 on 7 and 72 DF,  p-value: < 2.2e-16
```

In the summary of the intial model, it is noticeable that we have $0.8832 = R^2 \approxeq R^2_{adj} = 0.8719$ and a high significance to our model with a p-value of less than $2.2e^{-16}$. However, there are two intial issues with this model, the first being the fact that for several of our variables, we have NAs for their values and the second being the fact that we have a very high residual standard error which indicates that while our R^2 may be high, we seem to have a poor fit for our linear model that may step from some of the earlier mentioned points. Similarly, we seem to have a high number of highly insignificant predictor variables though we do anticipate that these may disappear when we apply our step() function later in our analysis. Now, in order to try and fix these issues, we will begin by dropping the variables that gave us NAs and then refitting our model as follows.

```
  lmod1 <- lm(likeCount ~ tweetId + replyCount + retweetCount + followerCount + sentiments + textLengt
 summary(lmod1)
```

```
##
## Call:
## lm(formula = likeCount ~ tweetId + replyCount + retweetCount +
##     followerCount + sentiments + textLength + mentions, data = dfTrain)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -21276.9  -2284.9   -551.7   1822.2  21220.2
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    4.000e+08  1.068e+09   0.374 0.709233
## tweetId        4.706e-14  1.350e-14   3.485 0.000843 ***
## replyCount     3.338e+00  2.156e+00   1.548 0.125922
## retweetCount   3.514e+00  3.140e-01  11.193  < 2e-16 ***
## followerCount -3.618e+00  9.664e+00  -0.374 0.709202
## sentiments     8.153e+00  1.468e+01   0.555 0.580448
## textLength     1.758e+01  2.291e+01   0.767 0.445399
```

```
## mentions      -1.239e+02  3.756e+03  -0.033 0.973774
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6210 on 72 degrees of freedom
## Multiple R-squared:  0.8832, Adjusted R-squared:  0.8719
## F-statistic:  77.8 on 7 and 72 DF,  p-value: < 2.2e-16
```

It is noticeable for lmod1 that all of our stats are the same as above, we just do not have the issues with the NAs. Therefore, we can feel comfortable letting this model be our base model. Now, to optimize this base model, we will begin by applying a step() function to our model to try and reduce our model by dropping some of our insignificant predictors.

Now, using the results from our step function, we find that the following linear mode is our optimal linear model and thus we will consider if this model is an improvement by using Anova to determine if we can replace our full model with this reduced model.

```
lmodRed <- lm(likeCount ~ tweetId + replyCount + retweetCount, data = dfTrain)
summary(lmodRed)
```

```
##
## Call:
## lm(formula = likeCount ~ tweetId + replyCount + retweetCount,
##     data = dfTrain)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -20921.3  -2539.4   -454.9   1313.2  21728.7
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -4.067e+04  7.975e+03  -5.099 2.43e-06 ***
## tweetId       4.894e-14  9.072e-15   5.394 7.51e-07 ***
## replyCount    3.429e+00  2.067e+00   1.659    0.101
## retweetCount  3.514e+00  3.038e-01  11.569  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6088 on 76 degrees of freedom
## Multiple R-squared:  0.8816, Adjusted R-squared:  0.8769
## F-statistic: 188.6 on 3 and 76 DF,  p-value: < 2.2e-16
```

```
anova(lmod1, lmodRed)
```

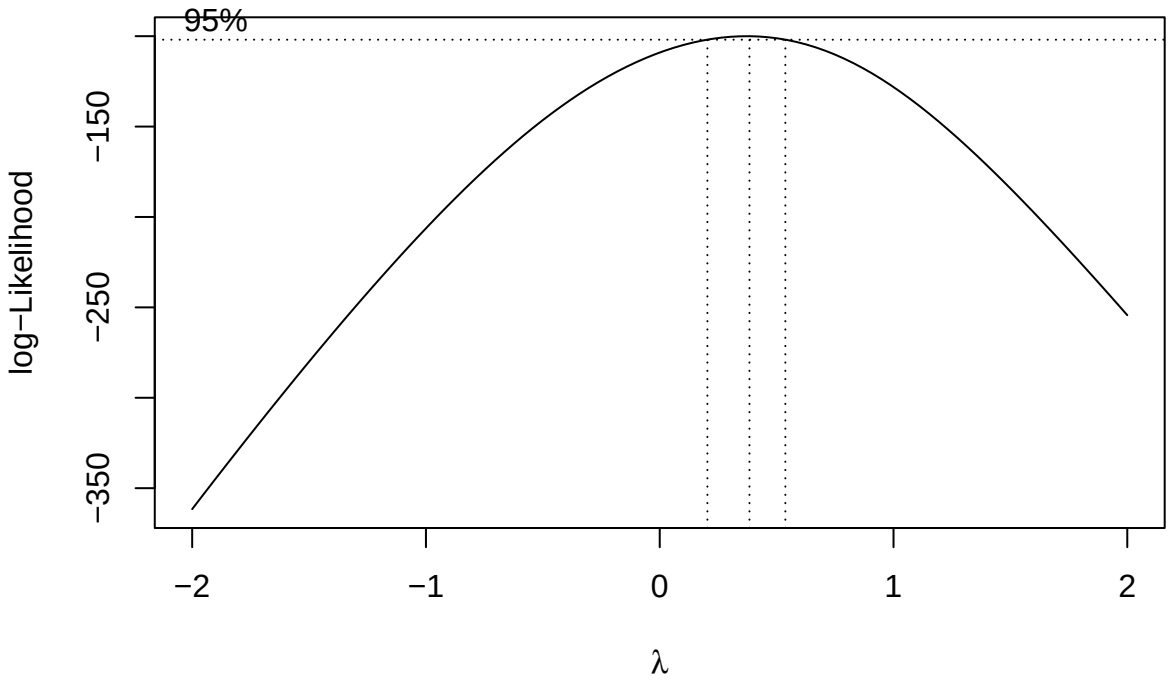
```
## Analysis of Variance Table
##
## Model 1: likeCount ~ tweetId + replyCount + retweetCount + followerCount +
##     sentiments + textLength + mentions
## Model 2: likeCount ~ tweetId + replyCount + retweetCount
##   Res.Df        RSS Df Sum of Sq      F Pr(>F)
## 1     72 2776892688
## 2     76 2816596627 -4 -39703939 0.2574 0.9043
```

However, as our p-value is greater than 0.05, it follows that we accept the null hypothesis that our full model is not necessary and that we can instead use our reduced model. Now, we will instantiate this model, lmodRed as lmod for the rest of our analysis.

```
lmod <-lm(likeCount ~ tweetId + replyCount + retweetCount, data = dfTrain)
```

Now, we will consider if there is a choice of a transformation of our response variable that would be best to use to help us obtain the best model possible. Now, we can use the Box Cox function on our model to find the proper transformation for our predictor. We find that we have a tranformation of,

```
library(MASS)
bc <- boxcox(lmod)
```




```
lambda <- bc$x[which.max(bc$y)]
lambda
```

```
## [1] 0.3838384
```

Looking at the output of our Box Cox function, we can see that we obtained a Box Cox value of 0.3434. Thus we can use this value as follows to apply the tranformation to our model and thus consider the results of the following tranformation.

```
lmodBoxCox <- lm((likeCount^{0.34}-1)/0.34 ~ tweetId + replyCount + retweetCount, data = dfTrain)
summary(lmodBoxCox)
```

```
##
## Call:
## lm(formula = (likeCount^{
##     0.34
## } - 1)/0.34 ~ tweetId + replyCount + retweetCount, data = dfTrain)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -21.9876  -5.0613   0.1857   4.8775  27.8454
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -4.229e+01  1.231e+01  -3.436 0.000959 ***
## tweetId       1.063e-16  1.400e-17   7.592 6.62e-11 ***
```

```
## replyCount    9.053e-03  3.189e-03   2.839 0.005809 **
## retweetCount  3.245e-03  4.688e-04   6.922 1.23e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.394 on 76 degrees of freedom
## Multiple R-squared:  0.8304, Adjusted R-squared:  0.8237
## F-statistic:   124 on 3 and 76 DF,  p-value: < 2.2e-16
```

Thus, analyzing our model, we can note that the Box Cox transformation indeed had an impact on our model as indicated by the fact that our residual standard error has gone down while our p-value remained highly significant. We can notice that the significance for some of our predictors has increased with reply count now becoming significant to our 0.05 threshold, but that retweetCount is no longer as highly signficant as it was in our earlier model. However, since all of our variables are significant to a degree of $\alpha = 0.05$, we have a low residual standard error and high R^2 that is approximately equal to our R^2_{adj} it follows that we can feel comfortable claiming that we can move forward with these two models. Now, we can calculate the vif numbers for our models as well as the condition numbers of our model as follows,

```
  library(faraway)
  print("vif of original model")
```

```
## [1] "vif of original model"
```

```
  vif(lmod)
```

```
##      tweetId   replyCount retweetCount
##     1.170129     3.237029     2.997477
```

```
  print("vif of boxcox model")
```

```
## [1] "vif of boxcox model"
```

```
  vif(lmodBoxCox)
```

```
##      tweetId   replyCount retweetCount
##     1.170129     3.237029     2.997477
```

```
  print("condition numbers of original model")
```

```
## [1] "condition numbers of original model"
```

```
  x <- model.matrix(lmod)[,-1]
  e <- eigen(t(x) %*% x)
  sqrt(e$val[1]/e$val)
```

```
## [1] 1.000000e+00 3.738193e+07 1.556627e+15
```

```
  print("condition numbers of Box Cox model")
```

```
## [1] "condition numbers of Box Cox model"
```

```
  x1 <- model.matrix(lmodBoxCox)[,-1]
  e1 <- eigen(t(x1) %*% x1)
  sqrt(e1$val[1]/e1$val)
```

```
## [1] 1.000000e+00 3.738193e+07 1.556627e+15
```

Our VIF numbers indicate some collinearity between our replyCount and our retweetCount variables, but not enough to where we should worry about them, as well as the fact that it appear that we have some very high condition number for both of our models. We can think that this is due to the fact that some of our predictor variables have much larger coefficients than our other coefficients. This indicates that a change in

one predictor variable would have a far more drastic impact on our output variable than a change in our other variables such as seen in our Box Cox model with tweetId vs replyCount. This is something to keep in mind with our model. Now, we can check our models to see if they meet our assumptions for using linear models on. First checking for normality of our residuals, we can check using our Shapiro-Wilk's test for our residuals to make sure that our residuals are normally distributed,

```
  shapiro.test(resid(lmod))
```

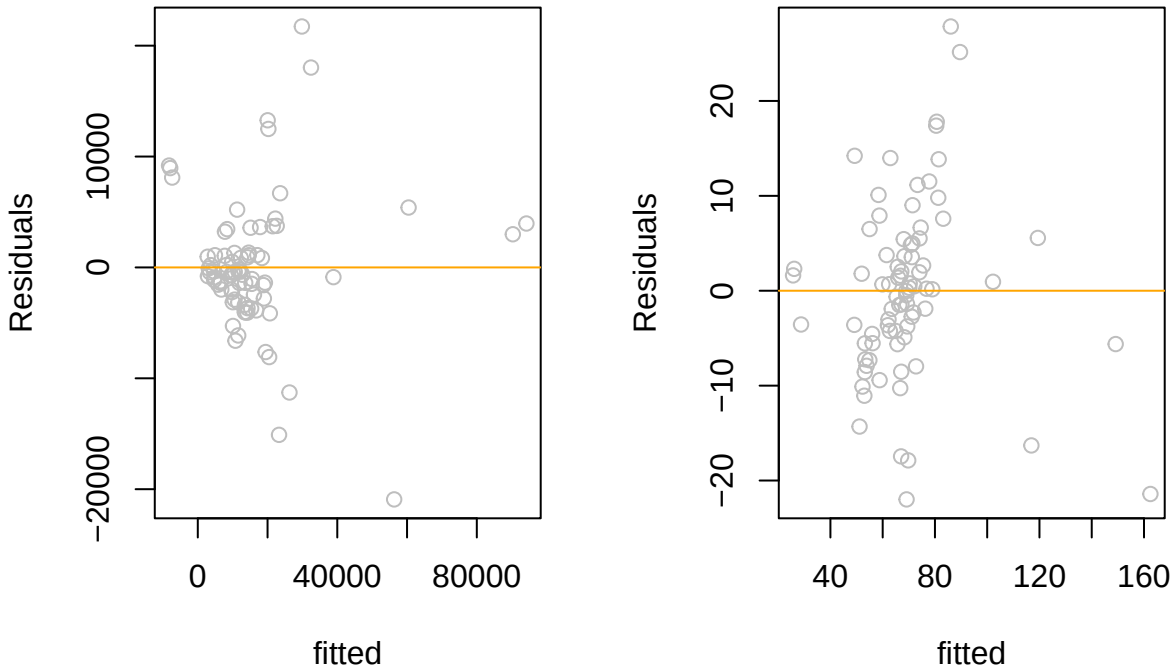
```
##
##  Shapiro-Wilk normality test
##
## data:  resid(lmod)
## W = 0.89412, p-value = 6.939e-06
```

```
  shapiro.test(resid(lmodBoxCox))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(lmodBoxCox)
## W = 0.97676, p-value = 0.1531
```

We can note here that for our two different Shapiro-Wilks tests, we can see that for our first residuals, since we have a p-value of less than 0.05, it follows that we reject our null hypothesis that the residuals are normally distributed and say that they are not normally distributed. However, since our Box Cox model has met our assumption of normal distribution of our residual values due to the p-value being greater than 0.05, we can see that our assumption of normality of the residuals is met for the Box Cox model. We will now check for equal variance/homoscedasticity and linearity among the residuals,

```
par(mfrow=c(1,2))
plot(fitted(lmod), resid(lmod), col = "grey", xlab = "fitted", ylab = "Residuals")
abline(h=mean(resid(lmod)), col = "orange")
plot(fitted(lmodBoxCox), resid(lmodBoxCox), col = "grey", xlab = "fitted", ylab = "Residuals")
abline(h=mean(resid(lmodBoxCox)), col = "orange")
```



We see that the mean of the residuals for each of our models are roughly centered around 0. The spread of the

residuals seems to be pretty even, indicating linearity, although we see that there are a handful of outliers that clearly stick out. However these outliers seem to be best minimized using our Box Cox transformed model in magnitude and in the spread of the outliers. Thus, we must consider this when using our model and when evaluating whether or not we can actually use our model as we now have problems with two of our assumptions for our linear model, lmod which is discouraging for its usage. We can now see that it seems like our best linear model is our Box Cox transformation given by the fact that the magnitude of our residuals seems to be the least of the two models and since it has met our assumptions for a linear model. Thus, we will be considering how we can further fit this particular model.

Now, we can try to further improve the fit of our model by attempting to remove some of our unusual observations. We will focus on finding the points that are outliers, high leverage points, and influential ponts, and then attempt to remove some of these problematic observations. We will first identify our outliers as follows

```
# Leverage Points
print("number of high leverage points")
```

```
## [1] "number of high leverage points"
```

```
length(which(hatvalues(lmodBoxCox) > 2 * mean(hatvalues(lmodBoxCox))))
```

```
## [1] 10
```

```
# Outliers
print("number of outliers")
```

```
## [1] "number of outliers"
```

```
length(rstandard(lmodBoxCox)[abs(rstandard(lmodBoxCox)) > 2] > 2)
```

```
## [1] 4
```

```
#Influential Points

print("number of influential points")
```

```
## [1] "number of influential points"
```

```
17440 -length(cooks.distance(lmodBoxCox) < 4 / length(cooks.distance(lmodBoxCox)))
```

```
## [1] 17360
```

We identified $ 10$ leverage points, $ 4$ outliers, and about 17360 influential points. We can refit the model using a data frame without any of the identified outliers that hold TRUE values, as an exercise.

```
newData1 = dfTrain[-c(14304, 14519)]

lmod2 <- lm((likeCount^{0.34}-1)/0.34 ~ tweetId + replyCount + retweetCount, data = newData1)

summary(lmod2)
```

```
##
## Call:
## lm(formula = (likeCount^{
##     0.34
## } - 1)/0.34 ~ tweetId + replyCount + retweetCount, data = newData1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -21.9876  -5.0613   0.1857   4.8775  27.8454
```

```
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -4.229e+01  1.231e+01  -3.436 0.000959 ***
## tweetId       1.063e-16  1.400e-17   7.592 6.62e-11 ***
## replyCount    9.053e-03  3.189e-03   2.839 0.005809 **
## retweetCount  3.245e-03  4.688e-04   6.922 1.23e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.394 on 76 degrees of freedom
## Multiple R-squared:  0.8304, Adjusted R-squared:  0.8237
## F-statistic:   124 on 3 and 76 DF,  p-value: < 2.2e-16
```

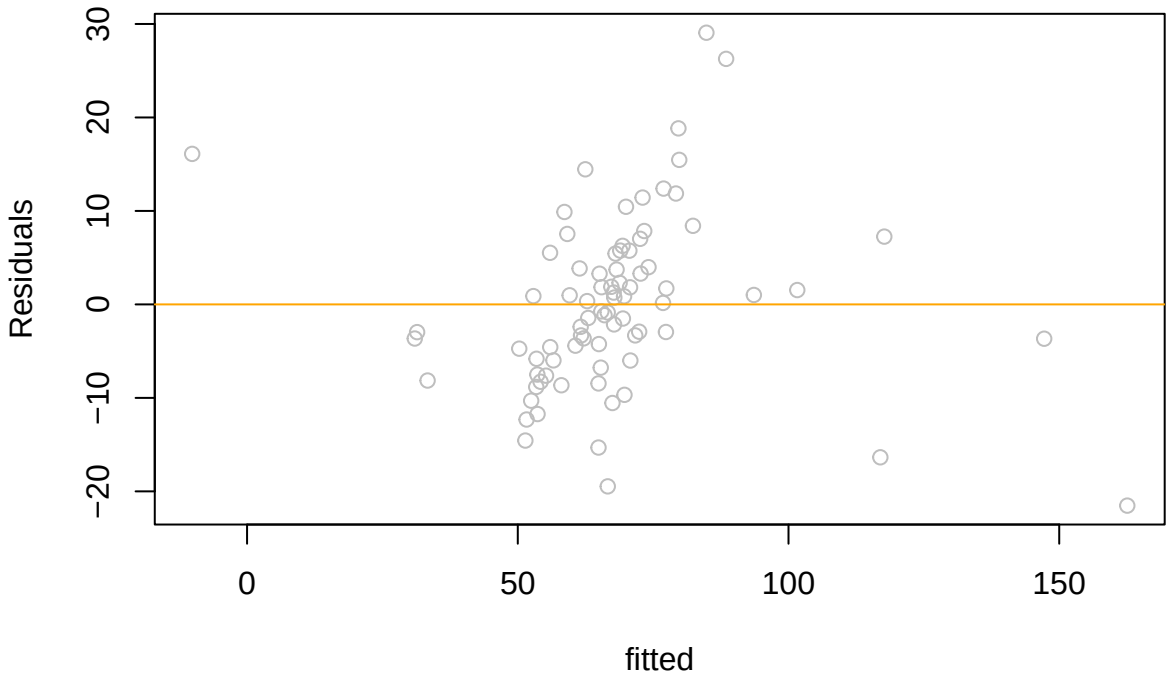
Looking at these results and comparing them with our earlier Box Cox model, we can note that there are no particular difference which we could expect given the fact that we were only removing 4 points out of 17445 total entries. We can also remove the points with the largest residual and fitted value from the model and get a better residuals graph. You can see a new graph of the residuals and fitted values below.

```
newData2 = subset(df,fitted.values(lmodBoxCox) < 120  & abs(resid(lmodBoxCox)) < 18)

newLmod <- lm((likeCount^{0.34}-1)/0.34 ~ tweetId + replyCount + retweetCount, data = newData2)

plot(fitted(newLmod), resid(newLmod), col = "grey", xlab = "fitted", ylab = "Residuals")

abline(h=0, col = "orange")
```



We can see with this that we have removed some of our most extreme values from the plot while maintaining the average of about zero on the plot, but that we subsequently seem to have new residual values that simultaneously contribute to higher residual values. Now, at this point in our analysis, we will be attempting to consider other kinds of models beyond our OLS linear model. Specifically, we will be focusing on trying PLS, Partial Component, Robust to try to construct a new model that may better fit data set and also manage to mitigate of the of issues with the non-normal distribution of our data. First, we construct our Partial Least Squares, Huber Model and our Least Trimmed Squares methods of linear models to attempt to fit a new and better

linear and to try to deal with some of the weird structures that we have to our data. Then, we can compare the predictions of these two models and our original model that come our of these models with the true values that we set aside with our dfPredict data set using the prediction function.

```
  rmse <- function(x,y) sqrt(mean((x-y)^2))
  ltsmod <- ltsreg(likeCount ~ tweetId + replyCount + retweetCount, data = dfTrain)
  library(pls)
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
  plsModel <- plsr(likeCount ~ tweetId + replyCount + retweetCount, data = dfTrain, scale=TRUE, validati
  huberModel <- rlm(likeCount ~ tweetId + replyCount + retweetCount, data = dfTrain)
  predOrig <- predict(lmodBoxCox, dfPredict)
  predPLS <- predict(plsModel, dfPredict)
  predLTS <- predict(ltsmod, dfPredict)
  predHuber <- predict(huberModel, dfPredict)
  print("Original Model: ")
```

```
## [1] "Original Model: "
```

```
  rmse(predOrig, dfPredict$likeCount)
```

```
## [1] 19291.28
```

```
  print("LTS Model:")
```

```
## [1] "LTS Model:"
```

```
  rmse(predLTS, dfPredict$likeCount)
```

```
## [1] 19599.87
```

```
  print("predPLS")
```

```
## [1] "predPLS"
```

```
  rmse(predPLS, dfPredict$likeCount)
```

```
## [1] 12305.42
```

```
  print("huber model prediction")
```

```
## [1] "huber model prediction"
```

```
  rmse(predHuber, dfPredict$likeCount)
```

```
## [1] 13064.31
```

Looking at these results, we can see that we have some issues with the prediction abilities of our models. It is noticeable that we have quite large RMSE values, each one greater than 10000. For our Box Cox predictive model, we can note that we seem to have some issues with the predictive ability of the model despite our high R^2 value and supposed good fit as indicated by such a metric and our residual standard errors. We can also note that we also have a highest RMSE values for our LTS model which implies that it may not be the best predictive model. Our Huber model does perform closely to our PLS model which indicates some predictive ability, but overall, we are not terribly impressed with the predictive abilities of any of our models.

```
  twitterPCA <- prcomp(dfTrain)
  summary(twitterPCA)
```

```
## Importance of components:
##                              PC1   PC2  PC3   PC4  PC5   PC6   PC7   PC8
## Standard deviation     8.166e+16 16853 1409 324.6 73.6 48.45 30.47 0.186
## Proportion of Variance 1.000e+00     0    0   0.0  0.0  0.00  0.00 0.000
## Cumulative Proportion  1.000e+00     1    1   1.0  1.0  1.00  1.00 1.000
##                              PC9      PC10 PC11 PC12 PC13
## Standard deviation     1.298e-15 4.468e-19    0    0    0
## Proportion of Variance 0.000e+00 0.000e+00    0    0    0
## Cumulative Proportion  1.000e+00 1.000e+00    1    1    1
```

```
  modelPCA <- lm(likeCount ~ twitterPCA$x[,1:2], dfTrain)
  summary(modelPCA)
```

```
##
## Call:
## lm(formula = likeCount ~ twitterPCA$x[, 1:2], data = dfTrain)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1198.94   -99.04    -1.60    79.11  1004.36
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)              1.588e+04  3.476e+01   456.8   <2e-16 ***
## twitterPCA$x[, 1:2]PC1 -6.750e-14  4.283e-16  -157.6   <2e-16 ***
## twitterPCA$x[, 1:2]PC2 -9.760e-01  2.075e-03  -470.2   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 310.9 on 77 degrees of freedom
## Multiple R-squared:  0.9997, Adjusted R-squared:  0.9997
## F-statistic: 1.23e+05 on 2 and 77 DF,  p-value: < 2.2e-16
```

```
  predPCA <- predict(modelPCA, dfPredict)
```

```
## Warning: 'newdata' had 6 rows but variables found have 80 rows
```

```
  suppressWarnings(rmse(predPCA, dfPredict$likeCount))
```

```
## [1] 22714
```

We can see from looking at our PCA, that all of our variance lies within our first primary component for our model. Thus, we have made the decision to use our first two PCA components in a linear model. As a result, we found that these two components were highly significant and actually the whole model had a R^2 value of 1, is generally highly significant as a whole and that we also have fairly low residual standard error values although greater than what we found in our Box Cox model. However, the predictive ability of the model appears to be worse than our Box Cox model, being the worst of our considered models.

Results

In our analysis, we tested a number of different models with the hopes of being able to find one that would appropriately answer our desired question. From our analysis, we found that our best models seemed to have been our PLS mode, our Box Cox Model, and our PCA model when considering their respective properties. The PCA model did show promise with a near 1 R^2 and a fairly low residual standard error, but due to it

having the highest RMSE value of all of our models, we decided not to use this model. The model that we decided in final was our Box Cox transformed model due to the fact that we found it had a good R^2 value with a low residual standard error, a low number of predictor variables which were all highly significant and the fact that the whole model was significant. We also felt that given the fact that our residual values for our model were relatively small and centered around zero along with meeting our assumptions of homoscedastity and normality of residuals, this model would be the best model to use. However, we can note that, as indicated by our RMSE values mentioned above in the Analysis section, it does not appear that our model does particularly well with predicting values from our data set as it appears that there are large discrepancies between the predicted values and the true values that we set aside in our dfPredict data frame. These issues with the predictive abilities of our model does provide some problems when trying to fully apply our model in a predictive setting.

Limitations of Study and Conclusion

We were able to fit a linear model that was relatively predictive for the chosen response variable and that model did conform to the assumption that the distance of the errors should follow a normal distribution among other assumptions we make to use a linear model. However, there were some issues with the predictive ability of our model as indicated by the RMSE values. We can attribute some of this to our data set. Give the structure of our data set with the virality of particular tweets, it is difficult to fully capture the features that are present in our data set and to turn them then into a fully fleshed predictive model. In the future, we would want to try more sophisticated techniques to capture our data set. It is possible as well that the information that we wish to draw out of our models to understand about the underlying question of the virality of Musk's presence on Twitter requires a non-linear model. Our general failure makes sense given that we are attempting to predict something that is fundamentally impossible to predict: whether a piece of content will be well received or go viral on social media. This data set inherently is non-normal due to the presence of virality on the internet which, while we might be able to identify some important and predictive factors, there's no way to accurately predict user engagement with internet content, particularly with such a polarizing figure like Musk and on such an ephemeral platform like Twitter. Thus, there are many instances where there is low interaction rate with his posts and other moments when his work attracts the attention of the whole platform, resulting in a massive spike in the post like count. Similarly, there are hoard of marketing campaign teams who look to replicate the influence that some of Musk's tweets have on the population with inconsistent ability as indicated by the failure of many ad campaigns and the constant need for advertising using paid ads rather than solely social media.