

#TwitterCritic: Sentiment Analysis of Tweets to Predict TV Ratings



TWEETS
4

FOLLOWERS
13



+ Follow

Isabel Litton

@iLitton

Joined September 2011

Tweet to Isabel Litton

Who to follow · Refresh · View all



Cal Poly @calpoly

+ Follow



SAS Software @SASsoftware

+ Follow



Python Software @TheP...

+ Follow

Find friends

Trends · Change

#SeniorProject

19K Tweets about this trend

#LearnByDoing

26.7K Tweets about this trend

#Statistics

55.2K Tweets about this trend

Tweets

Tweets & replies



Isabel Litton @iLitton · June 2015

A Senior Project presented to the Faculty of the Statistics Department



Isabel Litton @iLitton · June 2015

California Polytechnic State University, San Luis Obispo



Isabel Litton @iLitton · June 2015

In Partial Fulfillment of the requirements for the Degree Bachelor of Science



Isabel Litton @iLitton · June 2015

Advisor: Rebecca Ottesen



Back to top ↑

#Abstract

Twitter has rapidly become one of the most popular sites of the Internet. It functions not just as a microblogging service, but as a crowdsourcing tool for listening, promotion, insight and much more. From the perspective of TV networks, tweets capture the real time reactions of viewers, making them an ideal indicator of a show's ratings. This paper predicts Internet Movie Database (IMDB) television ratings by text mining Twitter data.

Tweets for five television shows were downloaded over a period of several months utilizing a SAS macro. Television show data, such as rating, show title, episode title, and more were retrieved through the Python package IMDBpy. Overall, there were four to seven episodes for each show, with approximately 1,000 to 100,000 tweets per episode.

Tweets were cleaned through a series of Perl-derivative regular expressions in SAS and Python. Once the data were cleaned as much as possible, both SAS and Python were used to score each tweet for sentiment analysis based on the AFINN dictionary. PROC SQL was used to join the datasets as the data were transferred from each program.

Sentiment analysis was used to determine the attitude or emotion of each tweet in order to properly capture the audiences' natural reactions. Reviews are written by a select minority of reviewers, while tweets can be written by anyone. The tweets might be more honest than an actual review because users are not writing tweets in the same setting that they would write a review.



#Table of Contents

- Introduction 4**
- Background Information (Data Sources) 5**
- Overall Process 7**
 - Cleaning Process 7
 - Scoring Process 12
 - Analysis 16
 - Multiple Linear Regression..... 16
 - Multivariate Adaptive Regression Splines (MARS) 18
- Visualizations..... 19**
- Conclusion 22**
- Appendix 23**
 - Other Cleaning Methods Considered..... 23
 - Model Conditions 24
- Acknowledgements 26**
- Resources 27**

#Introduction

Ever since the beginning of television criticism, reviews were made by professional critics who were generally not connected to the public audience. The rise of social media and networking sites, notably Facebook and Twitter, now allow us to solicit the opinions of the online community instead of relying solely on professional critics. Statisticians can leverage the data available through crowd sourcing to discover how the general public feels. This project harnessed the power of Twitter, IMDB, and sentiment analysis to investigate whether crowd sourcing of the Twitter community could accurately predict the IMDB critic ratings of certain television shows. Essentially, sentiment analysis was performed on tweets downloaded for five different television shows to predict their IMDB ratings. This paper discusses the entire process of sentiment analysis of Tweets, starting from the downloading and cleaning of the data to the end results of the analyses.

#Data Sources

The data and their sources are summarized in the following table:






DATA	SOURCE
Twitter 	SAS 
IMDB 	Python 
Sentiment Dictionary AFINN-111	Text File From Google 

Table 1. Project Data and Sources

Twitter data were accessed using a SAS macro called “%Tweetomatic”, which uses an automated approach to download the JSON formatted tweets. This macro was written as a summer research project and was developed over the course of two summers. “%Tweetomatic” combines batch processing with SAS code that allows users to leave their computer unattended while the data is downloading. It utilizes PROC HTTP to access Twitter’s API, and can be run with Base SAS versions 9.2 and above. The macro’s algorithm analyzes the rate at which tweets are posted to make the automated retrieval process possible. More details about this macro can be found in the paper provided in the Resources section.

IMDB data were downloaded using Python, specifically through a package called “IMDBPy”, which allows users to directly access the IMDB database. The Python script retrieves the show title, episode title, rating, number of critics who rated that episode, and more and writes this information to a CSV file. The main limitation with this script is that users must directly look up a show’s ID to be used in the script beforehand. The TV show’s IMDB ID can be easily looked up through the following steps:

1. Import the IMDB package by calling the code: `import imdb`

2. Call the IMDB function to gain access to the package's functions and to create an instance of the IMDB class: `imdb.IMDB()`
3. Call the following line of code: `print(search_movie("Television Show Title"))` to display a list of all movies or shows matching the title provided.

Since many movies and shows share the same title, it's difficult for the program to select the exact show a user wants without the identification number.

Words contained in the tweets were scored based on values provided by the AFINN-111 list, which can be found through a simple Google search. The list was published in the Technical University of Denmark by Finn Årup Nielsen. Each word in the list is assigned an integer ranging between -5 to +5 based on its valence.

Information about the shows selected for this project is outlined in the following table.

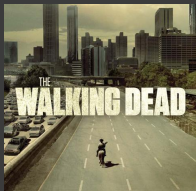



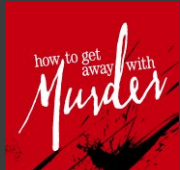
SHOWS				
The Walking Dead 	NCIS: LA 	New Girl 	Arrow 	How To Get Away With Murder 
13 episodes	15 episodes	16 episodes	17 episodes	9 episodes
213,988 tweets	7,044 tweets	12,742 tweets	171,038 tweets	229,034 tweets

Table 2. TV Show Information

The five shows were selected for their differences in genres, times, and networks. The tweets accounted for in Table 2 were posted starting midnight of the day the show aired until the actual show time. Due to the timing of the project and deadline restrictions, no full seasons were captured for any of the shows. The downloading process started in October and concluded in April.

#Overall Process

The overall process involves two main parts: cleaning and analysis. The cleaning process involved deleting and replacing text with regular expressions, SAS, Python, and SQL. The analysis included multiple regression and multivariate adaptive regression splines performed solely in SAS.

Cleaning Process

Tweets are unstructured text, which make them difficult to score accurately. Although each tweet is only 140 characters, they're filled with links, acronyms, emoticons, misspelled words, slang words, and much, much more. The final cleaning process involved three parts. Initial cleaning by deleting certain words such as "&" and "http://". Replacing acronyms with their actual words, for example, LOL would be replaced by laugh out loud. Replacing the emoticon encoded values with their actual meaning so that a smiley face was represented by the word "smile". This step was critical for the sentiment analysis scoring process because the scoring file links only to words.

The cleaning process is shown with the following example of what a tweet undergoes at each step. Figure 1 illustrates the tweet that will be used in the example. Although this tweet was made for the purpose of this example, it is representative of the challenges involved in scoring unstructured text messages.



Figure 1. Example of What a Tweet Looks Like On Twitter.com

The example tweet shown in Figure 1 is how the data appear on the Twitter website. However, Figure 2 reveals how tweets appear once they're read into SAS.

```
omg h8 @MzKatieCassidy!! \ud83d\udc4e but so so much \ud83d\udc98
\ud83d\udc98 \ud83d\udc98 for @amellywood #Arrow
```

Figure 2. Example of What a Tweet Looks Like After Its Downloaded

It may be easy to classify this tweet just by reading it, however, having the computer score it is challenging. In order for the program to score tweets as accurately as possible, code was written to remove certain words that do not have a sentiment value and to replace acronyms and emoticons with their meaningful translations. The cleaning process begins with some data manipulation before trying to delete or replace

any words. In the following figures, boxes colored in blue represent steps completed in SAS, while boxes in grey represent steps accomplished with Python.

Step 1: Unroll each tweet into individual words so that each row/line is a single word instead of the entire tweet.

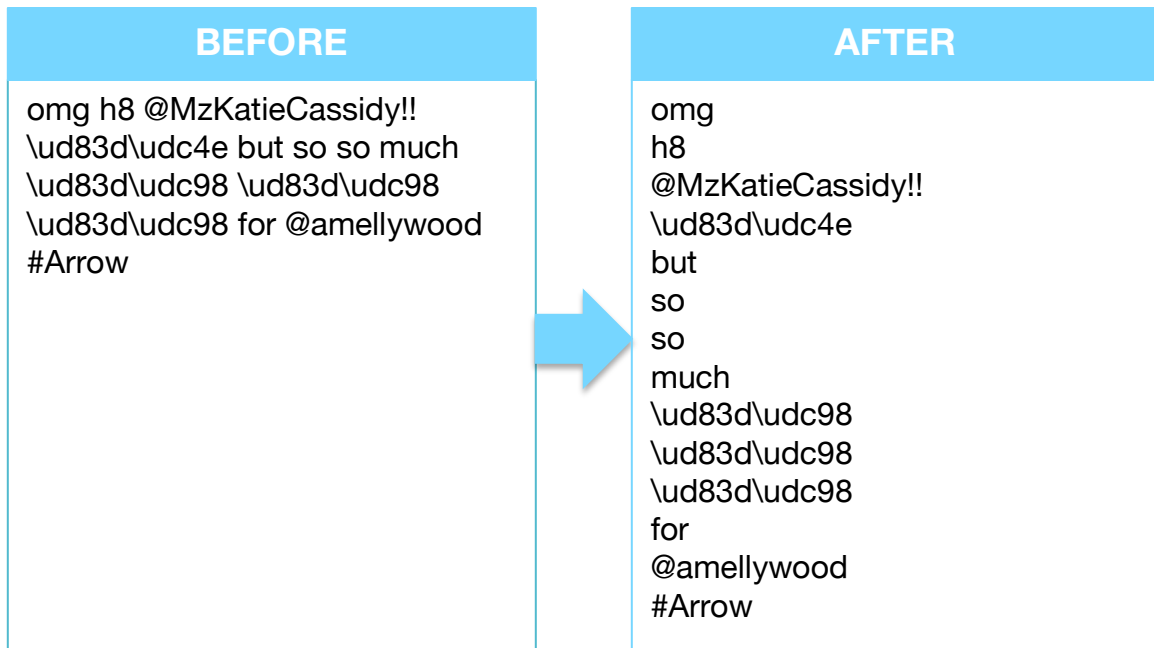


Figure 3. Diagram of Unrolling a Tweet

Tweets were separated into individual words in order to make string matching easier. Instead of scanning through the entire tweet for multiple phrases to delete, the program matches the phrases directly to each word. Handling the words this way streamlines the next step. It should be noted that the actual tweet data sets consisted of many records and they had to be split into smaller, more manageable data sets to be unrolled. Data sets were processed in partitions to handle the larger datasets with over 50,000 tweets that took a long time to unroll. Partitioning the data optimized the unrolling procedure, which streamlined the initial cleaning outlined in the next step.

Step 2: Delete certain words that hold no sentiment value

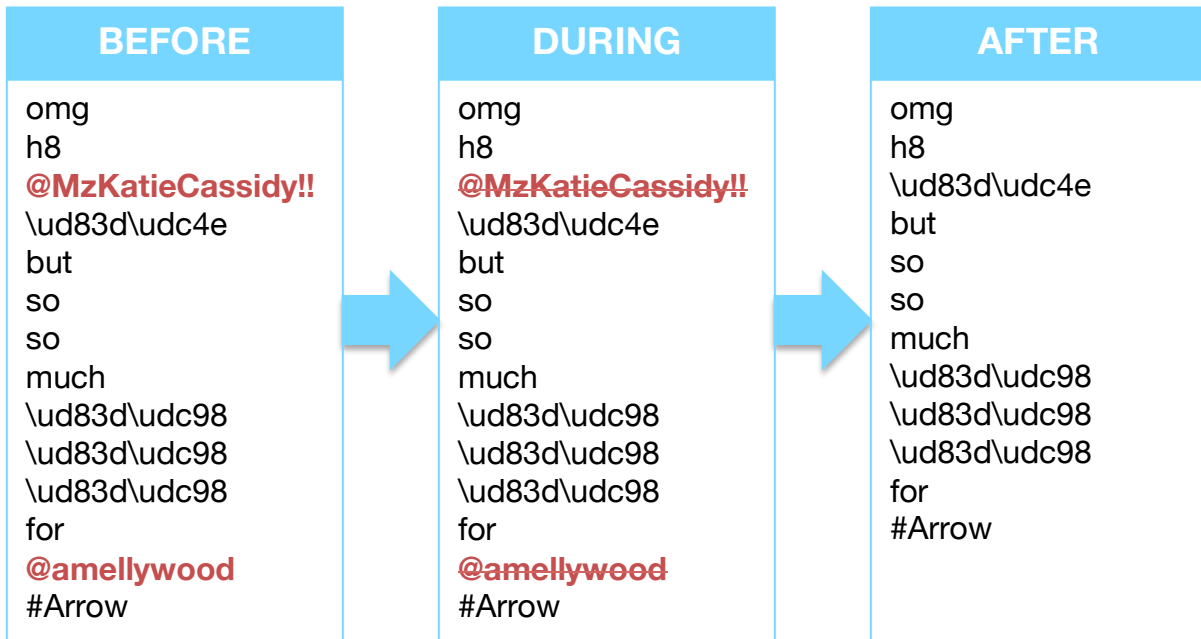


Figure 4. Initial Cleaning With Regular Expressions

Initial cleaning was performed through regular expressions within SAS. Regular expressions are basically pattern matching with strings. Similar to the example tweet, many posts often contain URL links and usernames, which do not contain any sentiment value. Perl code allows easy matching of words that start with “@”, contain “http”, and any other phrases that are not meaningful to score. Using regular expression also accommodates the variation in the usernames and web addresses by allowing SAS to look for key strings but ignore various patterns in the string.

Step 3: Select unique words and write them to a file

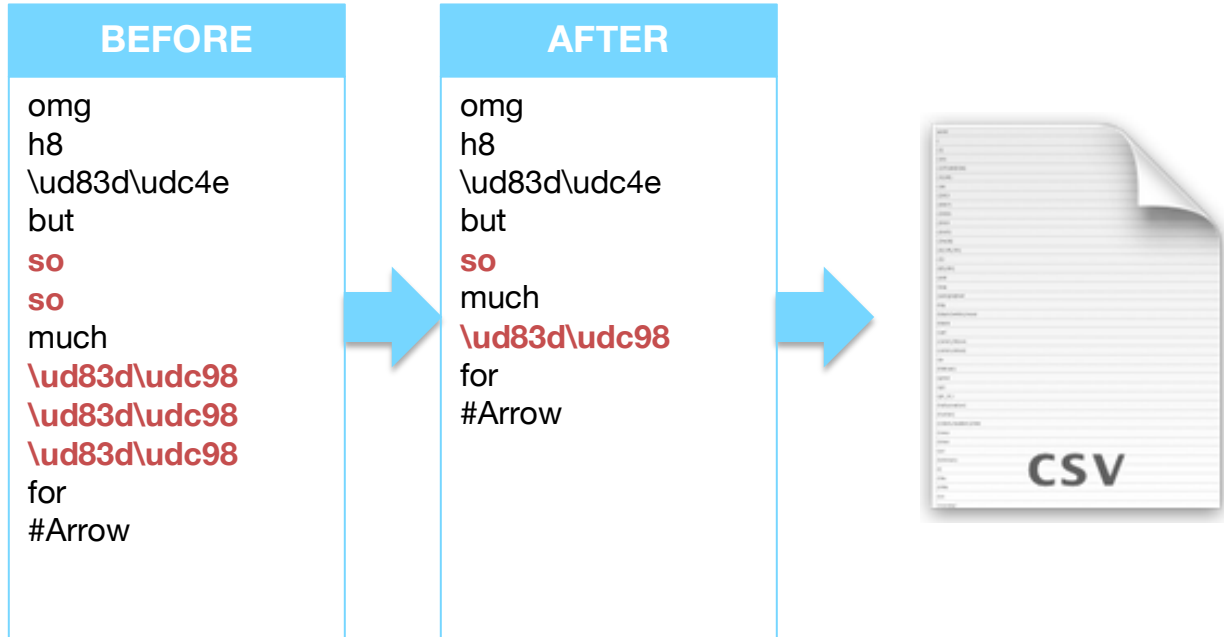


Figure 5. Writing Unique Words to a CSV File

Once each tweet was split into individual words, the number of observations increased considerably to the point where it was difficult to process due to lengthy processing time. Certain shows had approximately 100,000 tweets for just one episode and when unraveled, this resulted in millions of words. These episodes led to unreasonable processing times and sometimes crashed the program due to insufficient memory. To bypass the memory errors, an algorithm was developed to output distinct words to a file with a simple SQL query. Referring back to the example, Figure 5 illustrates that the text “so” and “\ud83d\udc98” would only be output once despite appearing multiple times in the original tweet. This resulted in a smaller overall word data set to be used for further cleaning and scoring.

Step 4: Replace acronyms with their meanings

BEFORE	AFTER	
<p>omg h8 \ud83d\udc4e but so much \ud83d\udc98 for #Arrow</p>	<p>Uncleaned</p> <p>omg omg omg h8 \ud83d\udc4e but so much \ud83d\udc98 for #Arrow</p>	<p>Cleaned</p> <p>oh my god hate \ud83d\udc4e but so much \ud83d\udc98 for #Arrow</p>

Figure 6. Replacing Acronyms With Meaningful Words

The CSV file containing unique words from all tweets was then read into Python to translate the acronyms into English words. Combining regular expressions and dictionary look up tables, Python replaced acronyms with their actual meanings. As shown above, “omg” would be replaced with three separate words “oh”, “my”, and “god”. Once all acronyms were properly translated, the next step was run to handle the emoticons.

Step 5: Replace emoticons with their meanings

BEFORE	AFTER	
<p>oh my god hate \ud83d\udc4e but so much \ud83d\udc98 for #Arrow</p>	<p>Uncleaned</p> <p>omg omg omg h8 \ud83d\udc4e but so much \ud83d\udc98 for #Arrow</p>	<p>Cleaned</p> <p>oh my god hate bad but so much heart for #Arrow</p>

Figure 7. Replacing Emoticons With Meaningful Words

Similar to the previous step, Python translated what each encoded emoticon actually meant in plain words. In this example, the “thumbs down” emoticon is translated to “bad” and the “red heart” emoticon is replaced with the word “heart”. Compared to the unstructured text from the original example tweet, the data is finally in a meaningful form that the computer can make sense of for sentiment scoring.

Scoring Process

Step 6: Score each newly cleaned word

BEFORE		AFTER		
Uncleaned	Cleaned	Unclean	Clean	Score
omg	oh	omg	oh	0
omg	my	omg	my	0
omg	god	omg	god	0
h8	hate	h8	hate	-4
...\udc4e	bad	...\udc4e	bad	-3
but	but	but	but	0
so	so	so	so	0
much	much	much	much	0
...\udc98	heart	...\udc98	heart	+1
for	for	for	for	0
#Arrow	#Arrow	#Arrow	#Arrow	0




Figure 8. Scoring Each Cleaned Word

An additional Python function was developed to score each cleaned word based on the AFINN dictionary. Once the cleaned words were assigned an integer from -5 to 5, the original word, cleaned word, and sentiment score were written to a CSV file. Words that did not match any sentiment words were assigned a value of 0. The scored data was then merged back with the original data consisting of all words unrolled as shown previously in Figure 5.

Step 7: Join original data with newly cleaned and scored data

BEFORE				
Original Unclean		Unclean	Clean	Score
omg	+	omg	oh	0
h8		omg	my	0
\ud83d\udc4e		omg	god	0
but		h8	hate	-4
so		...\udc4e	bad	-3
so		but	but	0
much		so	so	0
\ud83d\udc98		much	much	0
\ud83d\udc98		...\udc98	heart	+1
\ud83d\udc98		for	for	0
for		#Arrow	#Arrow	0
#Arrow				



DURING				
Original Unclean		Unclean	Clean	Score
omg	↔	omg	oh	0
h8		omg	my	0
\ud83d\udc4e		omg	god	0
but		h8	hate	-4
so		...\udc4e	bad	-3
so		but	but	0
much		so	so	0
\ud83d\udc98		much	much	0
\ud83d\udc98		...\udc98	heart	+1
\ud83d\udc98		for	for	0
for		#Arrow	#Arrow	0
#Arrow				



AFTER	
Clean	Score
oh	0
my	0
god	0
hate	-4
bad	-3
but	0
so	0
so	0
much	0
heart	+1
heart	+1
heart	+1
for	0
#Arrow	0

Figure 9. Joining Original Data With Cleaned Data Based On Uncleaned Words

SQL queries were used to combine the original unrolled data with the condensed scored data joined by original word and uncleaned word. Joining the data sets on the uncleaned word not only allows us to score the original data, but also replaces the messy words with the newly cleaned ones. Figure 9 demonstrates this joining process by starting with the original, uncleaned data, adding in the newly scored data, and ending with a table of just the cleaned words and their corresponding scores. To further clarify in the preceding example– while “omg” appears only once in the original data, it matches the three “omg” observations in the uncleaned column in the scored data set. Therefore, the final table will have the corresponding cleaned words “oh”, “my”, and “god”. Similarly, although “so” appears only once in the scored data set, it will show up twice in the final table because it occurred twice in the original data. Now that the original data is scored, all that is left in the cleaning process is to roll the words back into tweets and total their scores.

Step 8: Reroll individual words back into one tweet and total the scores

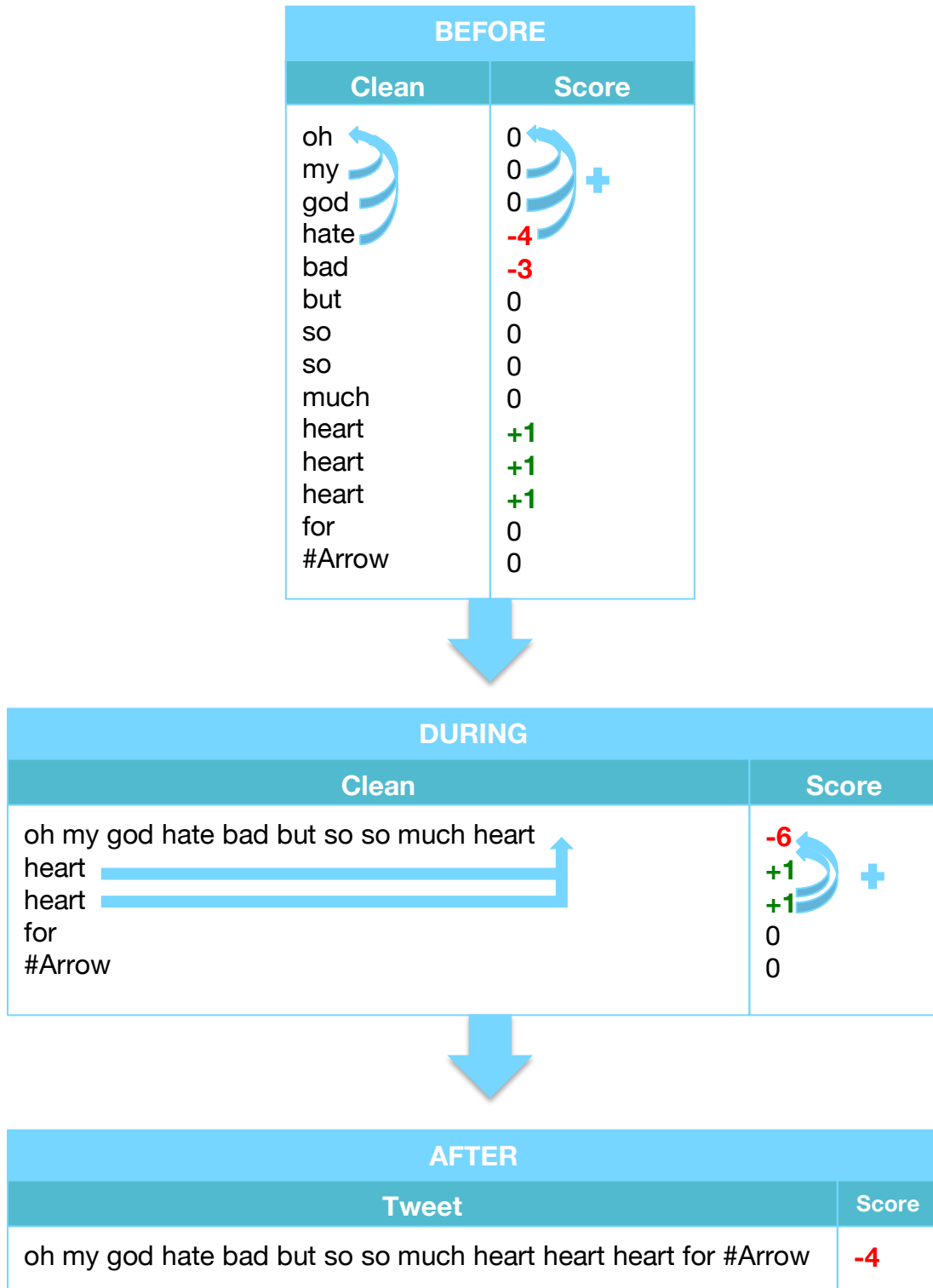


Figure 10. Recombining individuals words into each tweet and totaling their scores

In this final step the individual words get concatenated to the first word so that each row becomes a tweet again. As each word was appended, their scores were accumulated to compute the total. The end result was a tweet with slightly different words, due to the cleaning, and a total score. This step concludes the cleaning process and at this point the tweets were ready to be analyzed.

Analysis Process

An observation in the pre-analytic data consisted of a tweet, its score, show title, episode number, and the rating. Using individual scores as a predictor became a problem because the rating would be the same for tweets of the same episode. Since there are multiple observations for a single episode, the data almost seems to fit a repeated measures design, however, it does not fit exactly. Each tweet did not come from every episode of every show violating the definition of a repeated measures design. Therefore, in the final analytic data set the tweets were collapsed into specific statistics such as total score, mean score, and the standard deviation of scores for every episode to account for the multiple observations. Two different types of analyses were performed using these predictors, multiple regression and multivariate adaptive regression splines (MARS).

(Note: Vote count represents the number of critics who rated the episode on IMDB.)

Analysis #1: Multiple Regression

The following list represents all the models considered:

- Rating = ShowTitle + VoteCount + TotalScore
- Rating = ShowTitle + VoteCount + MeanScore + SDScore
- Rating = ShowTitle + VoteCount + MeanScore

Backwards elimination stepwise regression was used to select a final model with show title, mean score, and vote count. Table 11 reveals that all predictors are significant in this model based on the p-values. In this model 64.16% of the variability in the IMDB ratings of shows could be explained by the model with title of the show, number of critic ratings, and mean score as the predictors.

Source	F-value	P-value	R-Squared
ShowTitle	4.27	0.0428	0.6416
VoteCount	13.81	<0.0001	
MeanScore	7.06	0.0100	

Table 3. Type III Model ANOVA Results for Model With ShowTitle, VoteCount & MeanScore

Table 4 shows the parameter estimates for each predictor and confirms that mean score, and vote count are significant predictors after accounting for the other variables in the model. The shows *New Girl* and *The Walking Dead* are significant compared to the reference show *Arrow*. The show *Arrow* was randomly selected to be the reference group and does not hold any special meaning. Every one unit increase in the mean sentiment score and every additional critic rating is associated with a decrease of 0.22 and increase of 0.0001 in rating respectively. When changing from the show *Arrow* to *New Girl*, there is an associated decrease of 0.66 in the IMDB rating of the show. Lastly, when going from the show *Arrow* to *The Walking Dead*, there is an associated decrease in the predicted IMDB rating by 0.90.

Source	Estimate	t Value	P-value
MeanScore	-0.22	-2.07	0.0428
VoteCount	0.0001	2.66	0.0100
ShowTitle: <i>How to Get Away with Murder</i>	-0.06	-0.28	0.7780
ShowTitle: <i>NCIS: LA</i>	-0.18	-0.82	0.4151
ShowTitle: <i>New Girl</i>	-0.66	-3.14	0.0026
ShowTitle: <i>The Walking Dead</i>	-0.90	-4.29	<0.0001

Table 4. Parameters Estimates for Multiple Regression Model with MeanScore, VoteCount & ShowTitle

Analysis #2: MARS

MARS is a nonparametric regression that fits curved lines based on calculated splines. This model is more flexible and combines model selection with basis functions. This analysis uses the generalized cross validation (GCV) as an approximation to assess model performance. All the models fit for the multiple regression analysis were also fit using MARS, which ended up with the same final model as the multiple regression including show title, vote count, and mean score. Variable importance was calculated based on the square root of the GCV from a submodel minus the square root of the GCV from the selected model scaled to 100. The submodel is formed by removing all basis functions that have a certain variable removed. Based on variable importance, the number of critics has the largest importance, while mean sentiment score has the lowest. In other words, the contribution of the number of critics is the largest after accounting for the other variables in the model. Lastly, the R^2 squared of 0.6992 is similar to the one found through multiple regression which reveals that a majority of the variability in the ratings can be explained by this model.

Functional Component	Variable Importance	R-Squared
VoteCount	100.00	0.6992
ShowTitle	35.33	
MeanScore	1.82	

Table 5. Variable Importance for MARS Model with MeanScore, VoteCount & ShowTitle

#Visualizations

The following figures display simple scatter plots of each predictor by show.

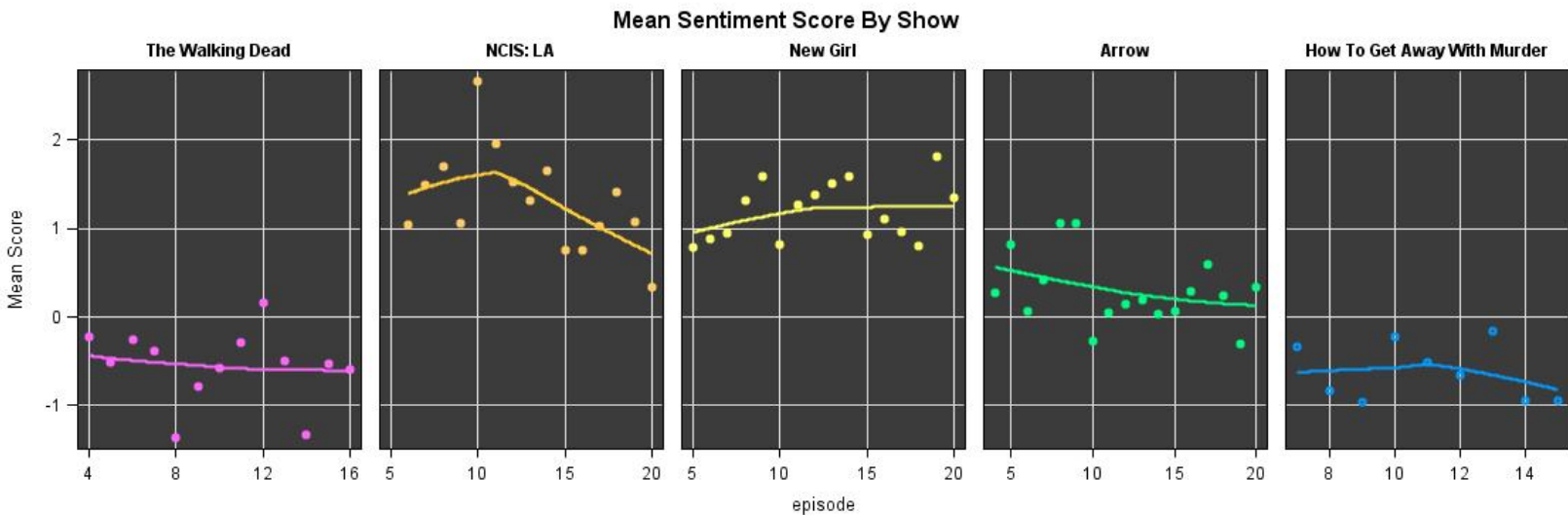


Figure 11. Scatterplots of Mean Sentiment Score By Show

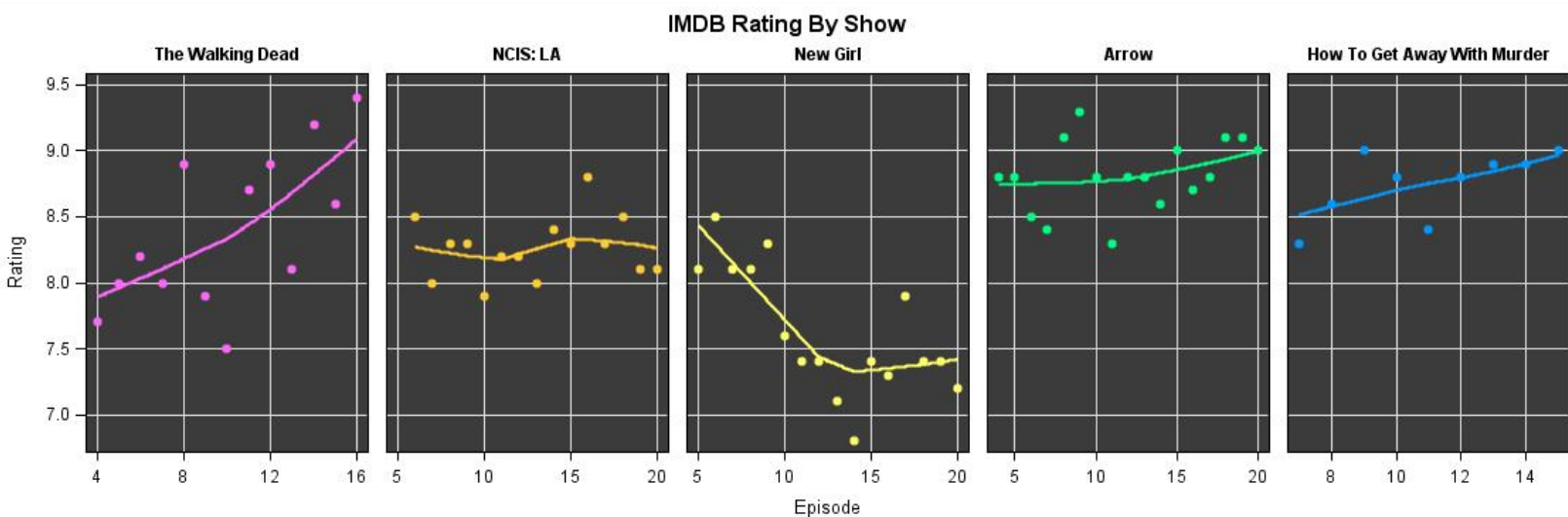


Figure 12. Scatterplots of IMDB Rating By Show

NCIS:LA and *Arrow* (orange and green cells) did not experience drastic variation between the mean score and rating by episode compared to other shows. *New Girl* (yellow cell), however, appears to have the largest discrepancy between mean score and rating. The ratings plummeted around episode 10, but the mean sentiment score consistently stayed around 1 to 2. The mean score appears to even increase later in the season, while ratings continued to get worse. This large discrepancy between mean score and rating for *New Girl* is difficult to explain, however, there appears to be

a logical reason for the disparity for *The Walking Dead* and *How To Get Away With Murder* (pink and blue cells). Both *The Walking Dead* and *How To Get Away With Murder* consistently have negative mean sentiment scores, but the ratings appear to be on the rise. Both shows had a substantial amount of tweets containing swear words compared to the other shows. For example, one *The Walking Dead* tweet received a score of -98 because it only contained one swear word (to the reader's imagination) repeated over and over. Unfortunately, the AFINN-111 dictionary assigns swear words the most extreme sentiment values of -4 or -5, which may not be the correct context of the word given opposite connotations used by the younger generation who are also more likely to tweet.

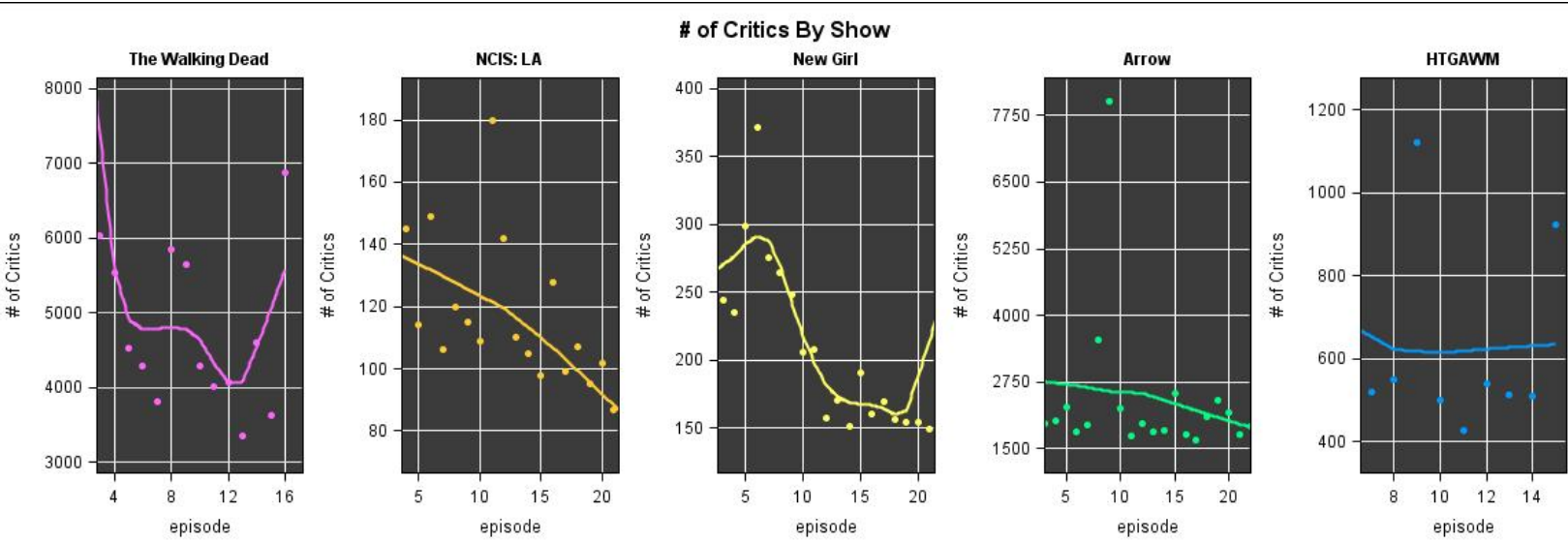


Figure 13. Scatterplots of the Number of Critics Who Rated Each Episode By Show

Lastly, the scatterplot shown in figure 12 displays vote count for each episode by the show to provide some perspective about the magnitude of the number of critics. *NCIS: LA* and *New Girl* have the lowest number of critics per episode, while *The Walking Dead* and *Arrow* have the largest. For almost all of the shows, there appears to be one episode early in the season that has a considerably higher number of critics than the rest of the episode. It is surprising that this does not occur for the season finales of the shows.

#Conclusion

Crowdsourcing Twitter data allows us to capture real time reactions from the online community, especially the instant feedback for television shows. These tweets represent the unfiltered, candid thoughts of users that might be more honest than an official review because they're capturing users' natural reactions. Sentiment analysis on tweets for five different shows with two different types of statistical model was performed. Out of all possible predictor variables included in the model, mean score, show title, and vote count were the only significant predictors for rating after accounting for all other variables in the model.

The sentiment analysis and models discussed in this paper only scratch the surface of what can be done with this data. Some future steps that warrant exploration include:

- investigating more cleaning methods such as stemming
- comparing multiple regression and MARS models with cross validation
- comparing tweets before, during, and after the airing of an episode
- using SAS Text Miner to form text topics
- examining the geolocation of tweets by show
- downloading more data for more shows

#Appendix

Other Cleaning Methods Considered

In addition to the cleaning process described above, various other options were explored. Ultimately, these cleaning methods were not included in the final process because of their inaccuracy. While some of these other methods were more efficient, the main cleaning process used for this project was the most accurate out of these three.

Checking For Embedded Sentiment Words

A common problem speculated to occur was missed sentiment words due to hashtag phrases. Since many hashtag phrases do not contain spaces, it would be difficult to accurately score these phrases if they contained any sentiment words. A SAS macro to separate any sentiment words embedded in blocks of text was written so that the scoring program would be able to capture these hidden words. For example, if the word with the embedded sentiment word was: “#lovethisshow”, the macro would separate this phrase into “#love” and “thisshow”. Now the sentiment word “#love” could be properly accounted for in the scoring process. Just as a side note, the scoring code would still be able to assign the phrase “#love” a sentiment value despite the preceding “#”. This method was not implemented in the final process because the occurrence of sentiment words embedded in hashtags was surprisingly low. The problem of embedded sentiment words only occurred in less than .01% of all words in the tweets for the various episodes that were tested. In addition, when this macro was included in the code that would unroll each tweet it caused the program to run unnecessarily longer than it needed to be. Therefore, for the sake of efficiency and after discovering that the problem was not as common as previously believed, this method was not incorporated into the final process.

Fuzzy String Matching

Another cleaning method that was investigated was fuzzy string matching through a Python package called “FuzzyWuzzy”. This package includes functions that will match strings based similarities based on distances, token sets, and sorts. The partial string similarity attempts to account for inconsistent length strings through what the developers call “best partial”. The site SeatGeek has an in-depth explanation, but the following example will be used for the the context of this study. The following function will compare the two strings provided and assign a value of how similar they are and can be used for identifying substrings of the given string. For example consider the code: `fuzz.partial_ratio(“amazing”, “ahmazing”) = 85`

This function does a great job of capturing slang words that look similar to sentiment words but do not match exactly. The use of fuzzy string matching was not included in the final processes because it ended up over scoring words. Many of the scores greater than 100 did not match the sentiment word at all. For example, matching the word “brilliant” and “ill” resulted in a value of 100, but these words obviously have opposite sentiment score. Since so many words resulted in erroneous scores, this method was excluded because in order to maximize scoring accuracy.

Model Conditions

Analysis #1: Multiple Regression

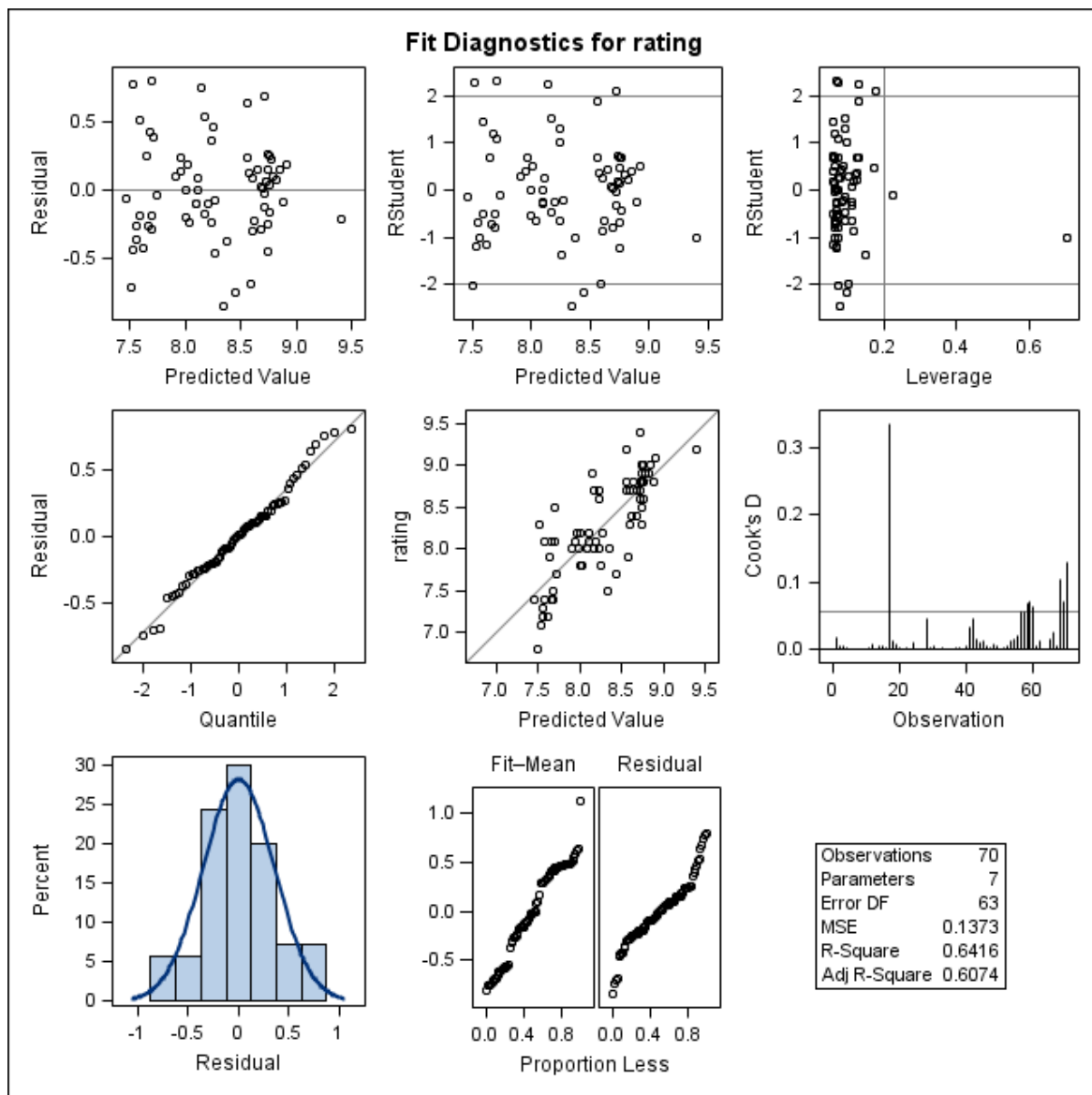


Figure 14. Plots to Check Conditions for Multiple Regression Models in Analysis #1

Linearity:

Since the points do not appear to form any patterns in the Residual by Predicted Value plot, the linearity condition does not appear to be violated.

Normality:

Since the distribution of the residuals appears to be approximately normal in the Percent by Residual plot, the condition does not appear to be violated.

Equal Variance:

The equal variance does not appear to be violated since there is no fanning shape or pattern in the Residuals by Predicted Value plot.

#Acknowledgments



Isabel Litton @iLitton · now

140 characters isn't enough to thank
[@RebeccaOttesen](#) for her endless guidance
and support! I don't know what I'd without
her! ❤️ [#thankful](#)



#Resources

- **ADAPTIVEREG Procedure:**
<https://support.sas.com/documentation/onlinedoc/stat/121/adaptivereg.pdf>
- **AFINN Dictionary:**
http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010
- **Fuzz String Matching:** <http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/>
- **IMDBPy:** <http://imdbpy.sourceforge.net/support.html#documentation>
- **Tweetomatic:** http://wuss.org/proceedings14/53_Final_Paper_PDF.pdf
- **Using Sentiment Analysis to Predict Popular TV Series:** <http://www.r-bloggers.com/using-sentiment-analysis-to-predict-ratings-of-popular-tv-series/>