Section 3: Analysis of Data

1) **Analyze your data and develop a model.**
2) **Document your process of analysis and your model.**
3) **Describe interim findings; also identify any adjustments you may decide to make to data sources (i.e., search for more or different data sets) or the model that you have developed.**
4) **Prepare a management summary ready for in-class presentation (e.g., using power point).**

To perform the sentiment analysis on the data I have used some lexicon which differentiates the negative words from the positive words. The lexicon named **Hu Liu Lexicon,** this lexicon is contributed by Hu and Bing Liu been used. These lexicon files are in working directory as .txt files. For performing the sentiment analysis, I have used the **Jeffrey Breen** approach. For the sentiment analysis I will use a function created that uses this lexicon of positive and negative words. This function will create a score for each tweet. A score of 0 indicates the tweet is neutral. A score of 1 or more indicates the tweet is positive. A score of -1 or less indicates the tweet is negative. The higher (or lower) the number indicates the relative strength of the sentiment (based on the count of words).

```
score.sentiment = function(sentences, pos.words, neg.words, .progress='none')
{
  require(plyr)
  require(stringr)

  # we got a vector of sentences. plyr will handle a list or a vector as an "l" for us
  # we want a simple array of scores back, so we use "l" + "a" + "ply" = laply:
  scores = laply(sentences, function(sentence, pos.words, neg.words) {

    # clean up sentences with R's regex-driven global substitute, gsub():
    sentence = gsub('[[:punct:]]', '', sentence)
    sentence = gsub('[[:cntrl:]]', '', sentence)
    sentence = gsub('\\d+', '', sentence)
    # and convert to lower case:
    sentence = tolower(sentence)

    # split into words. str_split is in the stringr package
    word.list = str_split(sentence, '\\s+')
    # sometimes a list() is one level of hierarchy too much
    words = unlist(word.list)

    # compare our words to the dictionaries of positive & negative terms
    pos.matches = match(words, pos.words)
    neg.matches = match(words, neg.words)

    # match() returns the position of the matched term or NA
    # we just want a TRUE/FALSE:
    pos.matches = !is.na(pos.matches)
    neg.matches = !is.na(neg.matches)

    # and conveniently enough, TRUE/FALSE will be treated as 1/0 by sum():
    score = sum(pos.matches) - sum(neg.matches)

    return(score)
  }, pos.words, neg.words, .progress=.progress )

  scores.df = data.frame(score=scores, text=sentences)
  return(scores.df)
}

#The positive and negative words lexicons are stored in a local director
#Please see appendix/reference for more information on origin
hu.liu.pos = scan('positive-words.txt', what = 'character', comment.char = ';')
hu.liu.neg = scan('negative-words.txt', what = 'character', comment.char = ';')

#Here we add some additional words that were discovered from initial review of tweets
pos.words <- c(hu.liu.pos)
neg.words <- c(hu.liu.neg, 'wait', 'waiting', 'hold', 'onhold' , 'on hold' , 'cancel','spam', 'spams', 'cancel', 'wtf')
```

Once the sentiment function and negative and positive lexicons have been created the function can be ran on the text file we created earlier to score each tweet.

```
#run the sentiment function on the text of the tweets
anthem.scores <- score.sentiment(tweets_nodups_text, pos.words, neg.words, .progress='none')

library("dplyr")

#merge the results back with the original file
anthem.score.merge <- merge(anthem.scores, tweets.nodups.df, by = 'text')

#Histogram of sentiment for all tweets
hist(anthem.score.merge$score,xlab=" ",main="Sentiment of tweets that mention Anthem BCBS", border="black",col="skyblue")
```

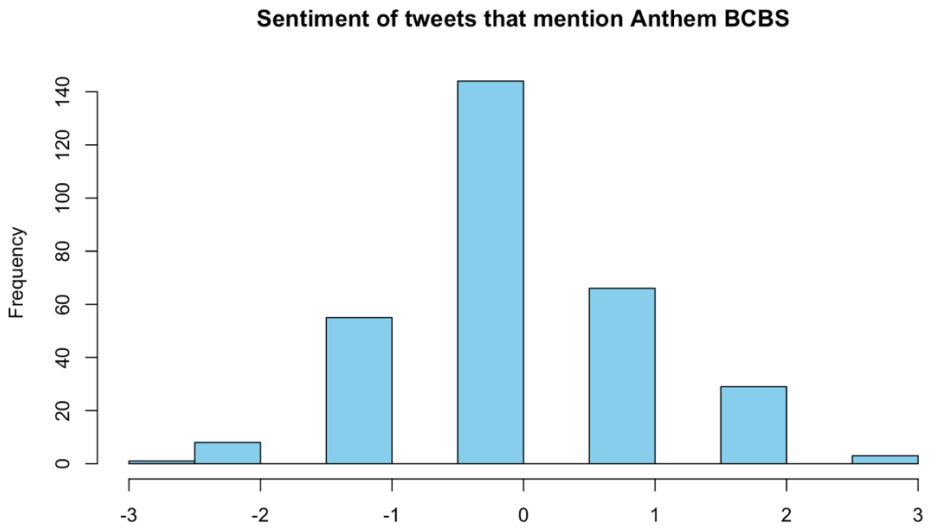

Fig 1. Screenshot showing the Sentiment Analysis Histogram

The above histogram shows the sentiments of tweets (Score Vs Frequency) and the bar with score 0 has frequency 140, which is neutral. When comparison is made between positive and negative tweets; positive has more frequency and hence the result is not optimal further I performed Scattered Plot.

```
#scatter plot of tweet date vs sentiment score
plot(anthem.score.merge$Date, anthem.score.merge$score, xlab = "Date", ylab = "Sentiment Score", main = "Sentiment of tweets that mention Anthem BCBS by Date")
```

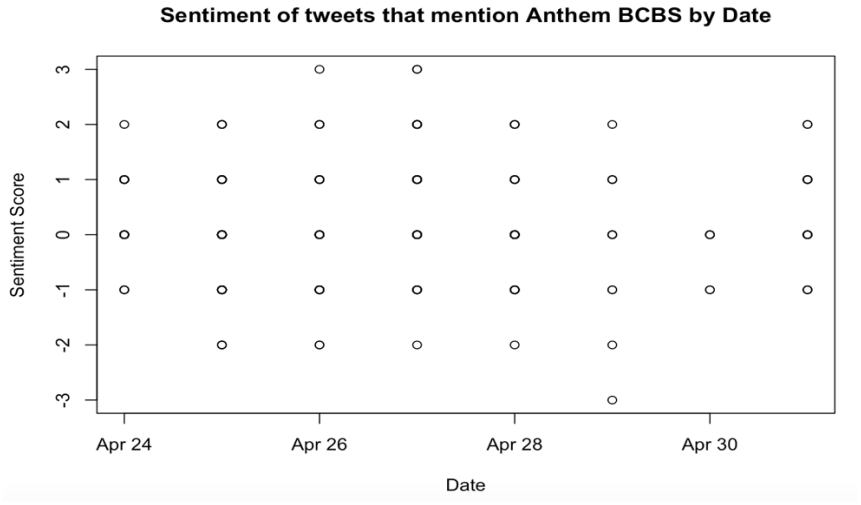

Fig 2. Screenshot showing Scatter Plot of tweet date vs sentiment score

Purely from the above histogram we see that sentiment scores look very normal with the majority of the tweets having a slightly negative (-1) sentiment score. From the scatter plot there does not appear to be much of a trend (many points are plotted on top of each other here).

```
#Sentiment (positive, negative and neutral) over time
ggplot(by.tweet, aes(Date, number)) + geom_line(aes(group=tweet, color=tweet), size=2) +
geom_point(aes(group=tweet, color=tweet), size=4) +
theme(text = element_text(size=18), axis.text.x = element_text(angle=90, vjust=1))
```

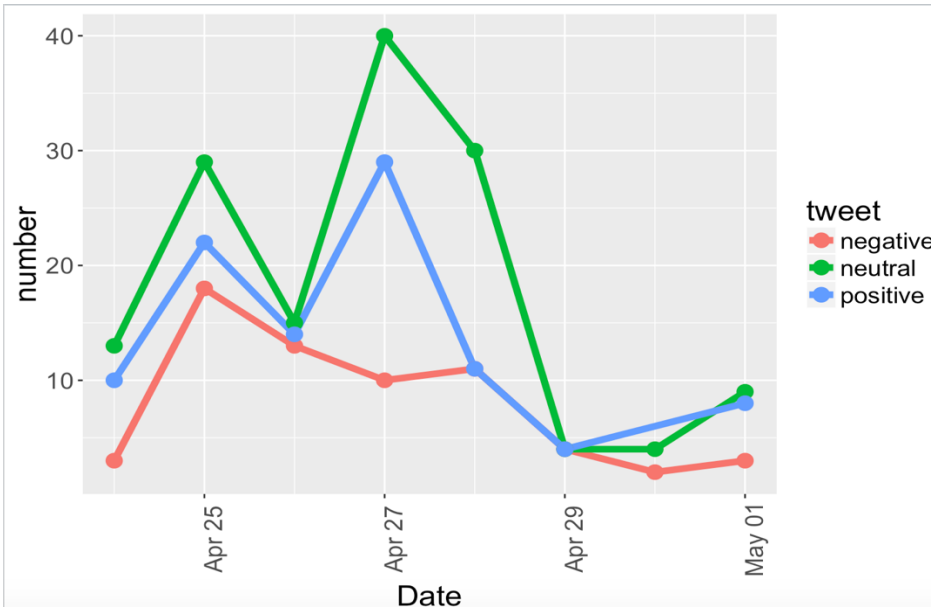

Fig 3. Screenshot showing the Sentiment over Time

Finally, from the graph of sentiment over time we can see that overall sentiment varies by date and seems to have trends over time. While sentiment on it's own is a fairly interesting analysis I wanted to further analyze and look for any correlation with stock prices.
First I read a csv file which was downloaded from Yahoo.com. This file was then cleaned a bit (date formatting) and joined with the file that contained each tweet and the associated sentiment. Finally, I eliminated any tweets that had no daily stock price change since this indicated they came from a weekend when no trading occurred.

```
#Read stock price CSV in
stock_prices <- read.csv("Anthem Stock Prices.csv")

#Format date so R knows this is a date field
stock_prices$Date <- as.Date(stock_prices$Date, "%m/%d/%y")

#Left join the sentiment analysis with the stock prices
tweet_stock <- left_join(anthem.score.merge, stock_prices, by = "Date")

#eliminate rows with no daily change
#eliminates weekend tweets
weekday_tweet_stock <- subset(tweet_stock,  !is.na(Daily.Change))
```

Now that I have the daily stock price change with each tweet and the sentiment of those tweets I can perform a simple correlation to check for association.

```
#Create indicator fields to flag tweets as positive, negative or neutral based on sentiment score
weekday_tweet_stock$pos <- as.numeric(weekday_tweet_stock$score >= 1)
weekday_tweet_stock$neg <- as.numeric(weekday_tweet_stock$score <= -1)
weekday_tweet_stock$neu <- as.numeric(weekday_tweet_stock$score == 0)

#Transform file from one row per tweet to one row per day summarizing the total positive, negative and netural tweets per day
tweet_stock_df <- ddply(weekday_tweet_stock, c('Date', 'Up.Down', 'Daily.Change'), plyr::summarise, pos.count = sum(pos), neg.count = sum(neg), neu.count = sum(neu))

tweet_stock_df$all.count <- tweet_stock_df$pos.count + tweet_stock_df$neg.count + tweet_stock_df$neu.count

#calculate the percent of tweets that were negative on each day
tweet_stock_df$percent.neg <- round((tweet_stock_df$neg.count / tweet_stock_df$all.count) * 100)

#Simple correlation
cor(tweet_stock_df$percent.neg, tweet_stock_df$Daily.Change, use = "complete")

glm_model <- glm(tweet_stock_df$Daily.Change ~ tweet_stock_df$percent.neg)
summary(glm_model)
```

```
Console ~/Documents/BUS602-PROJECT/
> #Simple correlation
>               cor(tweet_stock_df$percent.neg, tweet_stock_df$Daily.Change, use = "complete")
[1] -0.9421169
>
>               glm_model <- glm(tweet_stock_df$Daily.Change ~ tweet_stock_df$percent.neg)
>               summary(glm_model)

Call:
glm(formula = tweet_stock_df$Daily.Change ~ tweet_stock_df$percent.neg)

Deviance Residuals:
      1       2       3       4       5
 0.5214  0.6238 -0.5968 -0.9427  0.3943

Coefficients:
                            Estimate Std. Error t value Pr(>|t|)
(Intercept)                  3.30921    1.10480   2.995   0.0579 .
tweet_stock_df$percent.neg -0.24588    0.05052  -4.867   0.0166 *
```

Fig 4. Screenshot showing the Negative Correlation Statistics

The above screenshot is saying the correlation value is **-0.9421169**, which is **negative correlation**. And for every unit increase in percent of negative tweets there is decrease in 0.25 units in daily stock price change. Also P value is 0.016 (<0.05) is significant.

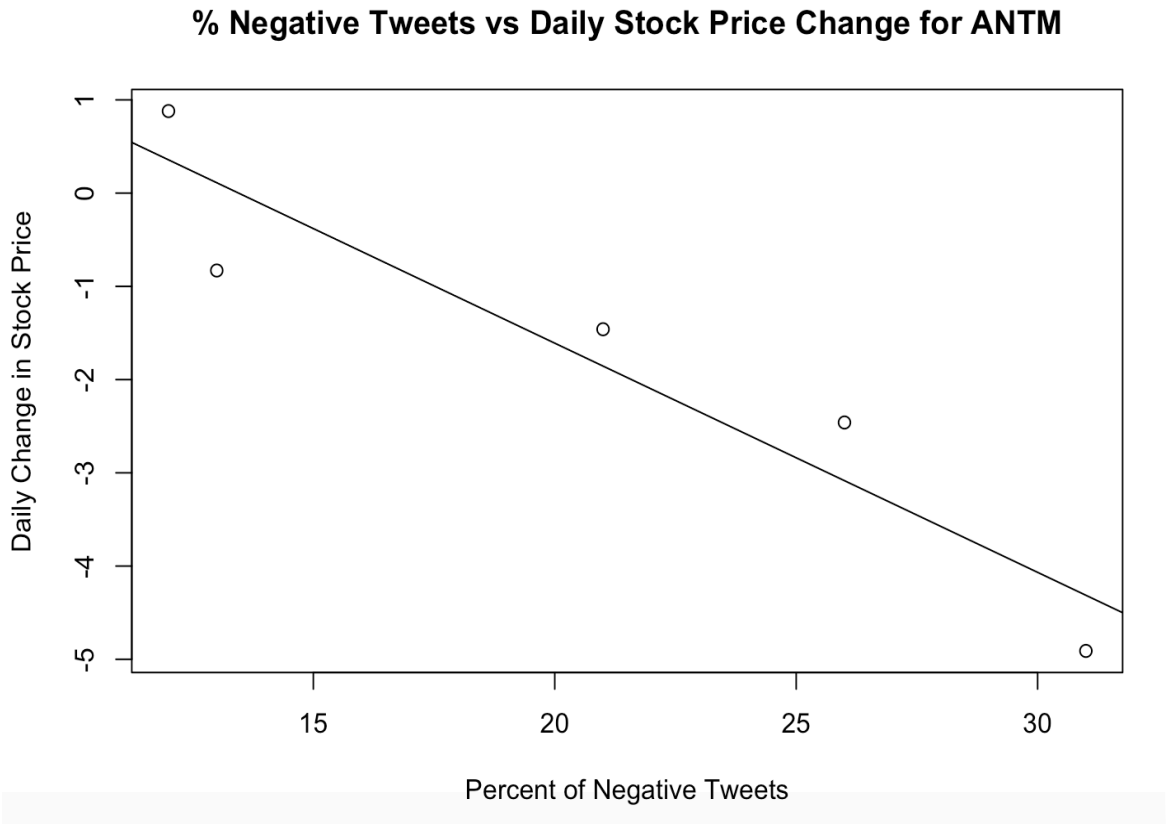

Fig 5. Screenshot showing Percent of negative tweets Vs Daily Change in Stock Price

Here dependent variable (target variable) is daily change in Stock Price and independent variable is Percent of negative tweets. From the final plot of Percent of negative tweets versus daily change in stock price we can see that logistic regression line fitted has a negative slope indicating that as the percent negative tweets increases the stock price decreases (which is negative correlation).

Hence, I would conclude negative sentiments in social media is negatively correlated with market share of company (Anthem Inc.)