```
#STOCK PREDICTION USING TWITTER SENTIMENT ANALYSIS#


              library("twitteR")
              library("httr")
              library("stringr")

              # we are using the setup_twitter_oauth function

              setup_twitter_oauth


              key         = "8FHWheFI***********"
              secret      =
"53Hum4Un4Y98gOa0PTUe9B*************************"
              mytoken     = "853727876973301765-
o*****************************"
              secrettoken =
"Ga3Wgo1seh9bakBgduk*****************************"

              # keep this order of arguments
              setup_twitter_oauth(key, secret, mytoken, secrettoken)

              tweets =  searchTwitter("@AskAnthem OR @AnthemInc OR
@ThinkAnthem OR
                                      @CareMoreHealth OR @Amerigroup OR
@AskBCBSGa OR @AnthemBusiness
                                      OR @AskEmpire OR @empirebcbs OR
#AnthemBCBS OR #bcbsAnthem", n = 1500, since = "2017-04-24", lang =
"en")

              #convert list to data frame
              tweets.df <- twListToDF(tweets)

              library("dplyr")

              tweets.nodups.df <- distinct(tweets.df, text, .keep_all =
TRUE)

              tweets.nodups.df$text <- gsub('…', '',
tweets.nodups.df$text)

              tweets.nodups.df <- plyr::rename(tweets.nodups.df,
c("created" = "Date")) #rename created to Date

              tweets.nodups.df$Date <- as.Date(tweets.nodups.df$Date)
#convert from datetime to date format

              #create text list with tweets for sentiment analysis
              tweets_text <- lapply(tweets, function(x) x$getText())

              #fix Mac encoding issue with
              tweets_text <- sapply(tweets_text,function(row)
iconv(row, to = 'UTF-8-MAC', sub = 'byte'))
```

```
              #removing duplicate tweets (retweets) from list
              tweets_nodups_text <- unique(tweets_text)

              library("NLP")
              library("tm")

              #Create tweet corpus
              r_stats_text_corpus <-
Corpus(VectorSource(tweets_nodups_text))

              #Clean up corpus in prepartion for word cloud
              #Encoding corrections for Mac
              r_stats_text_corpus <- tm_map(r_stats_text_corpus,
content_transformer(function(x) iconv(x, to='UTF-8-MAC', sub='byte')))
              r_stats_text_corpus <- tm_map(r_stats_text_corpus,
content_transformer(tolower)) #Transform all text to lower case
              r_stats_text_corpus <- tm_map(r_stats_text_corpus,
removePunctuation) #remove all punctuation
              r_stats_text_corpus <- tm_map(r_stats_text_corpus,
function(x)removeWords(x,stopwords())) #remove all stop words

              library("wordcloud")
              #Create color word cloud
              wordcloud(r_stats_text_corpus, min.freq = 10, max.words =
150, colors=brewer.pal(8, "Dark2"))

              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')

              #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")

              #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")
```

```
              #taken from https://www.r-bloggers.com/twitter-sentiment-
analysis-with-r/
              #total evaluation: positive / negative / neutral
              stat <- anthem.score.merge$score
              stat <- mutate(anthem.score.merge,
tweet=ifelse(anthem.score.merge$score > 0, 'positive',
ifelse(anthem.score.merge$score < 0, 'negative', 'neutral')))
              by.tweet <- group_by(stat, tweet, Date)
              by.tweet <- dplyr::summarise(by.tweet, number=n())

              #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))

              #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))


              #Raw plot of sentiment score versus daily change in stock
price
              plot(jitter(weekday_tweet_stock$score),
weekday_tweet_stock$Daily.Change, xlab = "Sentiment Score", ylab =
"Daily Change in Stock Price")

              #The below was modified from a LinkedIn PPT describing
sentiment analysis in R


              #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)

              #plot of % positive tweets vs daily change in stock price
with linear regression line overlaid
              plot(tweet_stock_df$percent.neg,
tweet_stock_df$Daily.Change, ylab = "Daily Change in Stock Price",
xlab = "Percent of Negative Tweets", main = "% Negative Tweets vs Daily
Stock Price Change for ANTM")
              abline(glm_model)

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

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

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

              #plot of % positive tweets vs daily change in stock price
with linear regression line overlaid
              plot(tweet_stock_df$percent.pos,
tweet_stock_df$Daily.Change, ylab = "Daily Change in Stock Price",
xlab = "Percent of Positive Tweets", main = "% Positive Tweets vs Daily
Stock Price Change for ANTM")
              abline(glm_model)
```