Twitter Equity Firm Value

CS4624: Multimedia, Hypertext, and Information Access
Blacksburg, VA 24061
5/8/2018
Client: Ziqian Song
Instructor: Dr. Edward A. Fox
By: Nathaniel Guinn, Christian Wiskur, Erik Agren, Jacob Smith, and Rohan Rane

Table of Contents

Table of Figures

Table of Tables

1. Executive Summary

This report outlines the way that the Twitter Equity team researched modern day data breaches and the way that Twitter has played a role in effecting a company's stock price following a breach. The introduction explains the importance of our research and the requirements explain the scope of our project. The design section explains the approach to each step of the project. It walks through our data collection of Twitter and stock data, how we analyzed all of this data, with a specific section on how we analyzed the stock data using the Fama French model, and lastly how we constructed our company guide. Following this is our user manual that explains all of the data files that we use in our code and that are available for future research on this project. The developers manual guides the reader through the process of setting up and running all of our scraping and analysis scripts. The lessons learned section of the document elaborates on some issues we experienced throughout the duration of the project and explains future work that could be done. This report finishes with acknowledging everyone who provided assistance, referencing all of the information used to produce our research, and an appendix of our code and reference tables.

The magnitude of the work that we did is large. We were given over seven hundred data breaches to analyze. From there we had to gather all tweets related to that event sometimes with over ninety thousand tweets scraped. After all the gathering we wanted to analyze different aspects of the Twitter information to try and find trends in companies who performed well despite a data breach.

Many of the data files that we produced aren’t present in the report because we generated over fifteen hundred but there is at least one example file to demonstrate the different inputs and outputs that our code works with.

2. Introduction

Incidents of data breaches that reveal company secrets or confidential client information can affect the firm seriously. This project records how firms use Twitter to manage the flow of information about data breach incidents. Also, it determines how users comment and spread the data breach information on Twitter. It then analyzes whether the above behaviors would have impact on firm stock performance after the data breach incidents.

For example, Equifax reported a data breach in September of 2017, which was all over the media. 143 million people were affected by this breach, and Equifax didn't release this information until 6 months after the incident occurred [1]. The stock market value of Equifax plummeted when they did announce the breach, and the company handled the entire response to the breach terribly. They tweeted out a link to a very poorly designed website and they also had multiple leadership changes before the breach was announced. We researched other companies who have gone through data breaches and determined if their social media interaction lessened the effects of the breach on the company's stock market price. We analyzed data breaches over the past 10 years and mined Twitter data from companies related to these breaches.

3. Requirements

For our project, we need to first gather tweet data before and after data breaches. Using this data, we need to look at how each firm responded to the event, for example some firms may respond to every user or make an announcement about the breach, while others may not have any activity on Twitter related to the breach. We also need to see if the firm's Twitter account had abnormal behavior after the data breach event and then compare it to their activity before the breach. Furthermore, we need to gather data on the firm's tweet data, including the firm's number of tweets, retweets, and likes. This will help us get a better idea of how much the firm used their account to handle other Twitter users, and events related to the breach.

Moreover, we also need to gather Twitter data for each data breach event, searching for tweets published during the event time using a provided keyword list. This includes many tweets, not just the firm's tweets. The goal is to analyze the topics of user discussion, classify different types of Twitter users, and identify influential users. After collecting the data, we need to analyze the stock market trends of the companies during the data breach event. Based on the firm's stock during the breach, we need to analyze companies which successfully managed the data breach and those that didn't. Ultimately, we need to come up with a proposal for how a company should handle a data breach based on our findings.

4. Design

4.1 High Level Design

Figure 1 below demonstrates the high level design of our project.

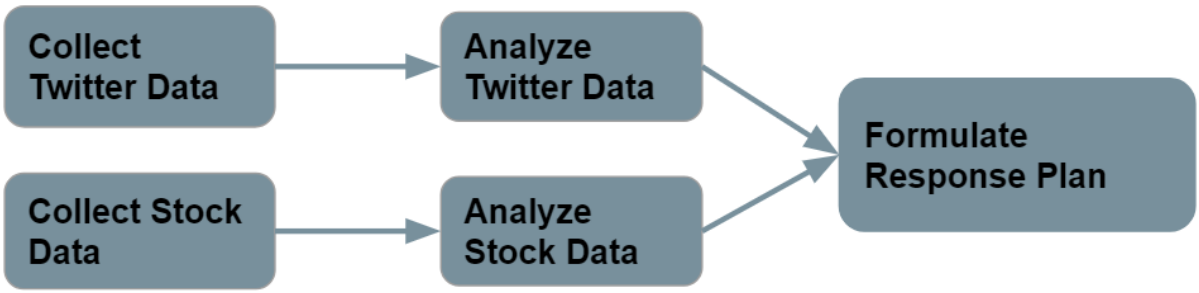

Figure 1: High Level Design

4.2 Twitter Collection

The process of collecting company and user Twitter data given in Figure 2. Refer to scrape_company_tweets.py and user_tweets.py in the Appendix.

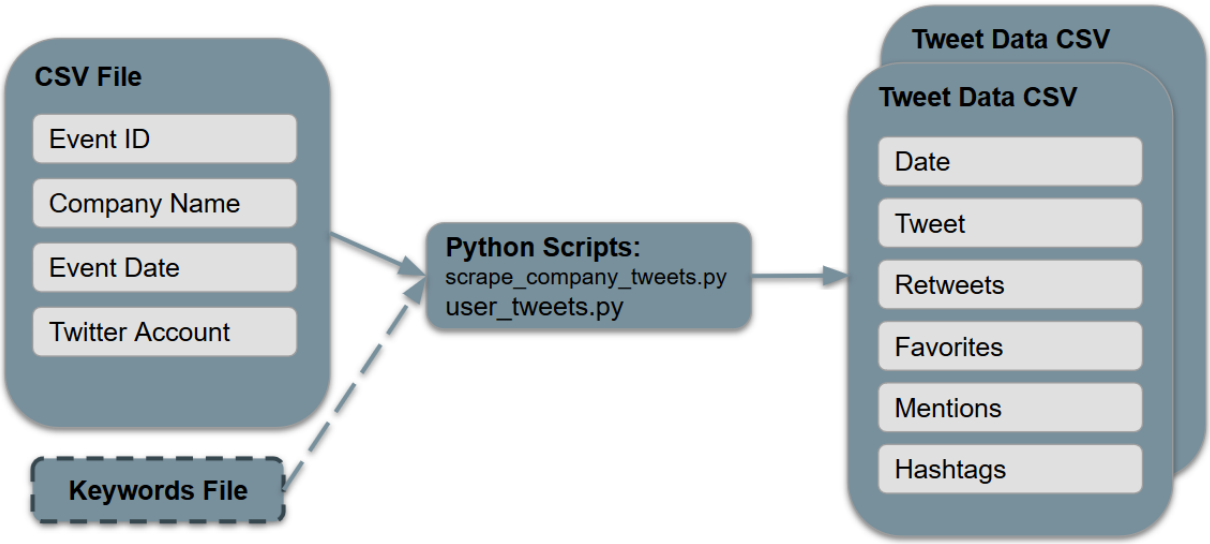

Figure 2: Tweet Collection Process

An input CSV file, which contains a list of data breaches, was processed by a Python script which returned a set of CSV files containing all relevant tweets and tweet

metadata. Each output CSV file corresponded to a data breach entry in the input CSV file.

Additional data was collected including user profile information and tweet type. Tweet type encompassed whether a tweet from a company was an announcement or reply. These data collection steps are illustrated in Figure 3 and 4. Refer to profile_scrape.py and accouncement_reply_firm.py in the Appendix.

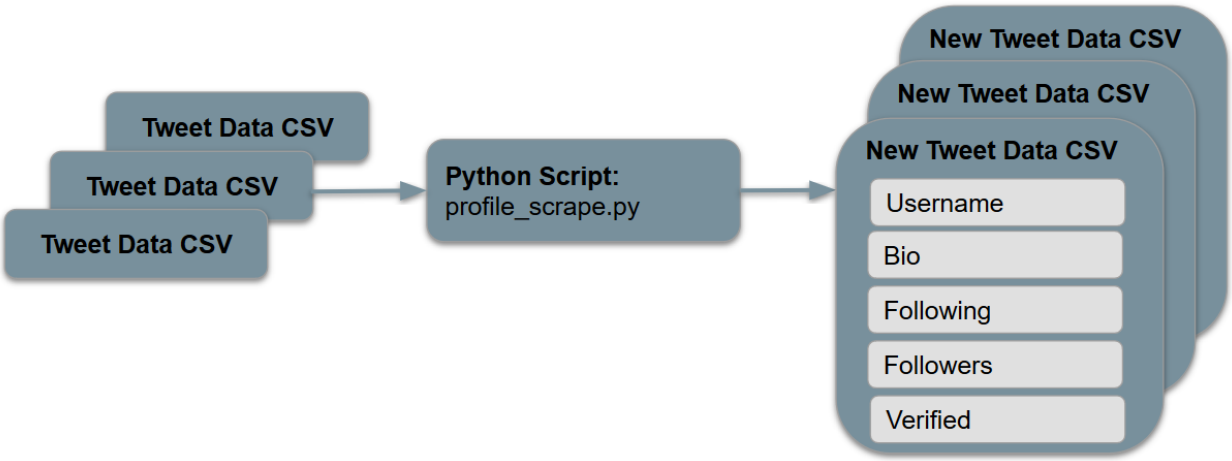

Figure 3: Profile Scraping Process

Figure 4: Announcement-Reply Determination Process

4.3 Stock Collection

Figure 5 explains how we designed our stock collection. Please refer to stockManipulation.py in the Appendix.

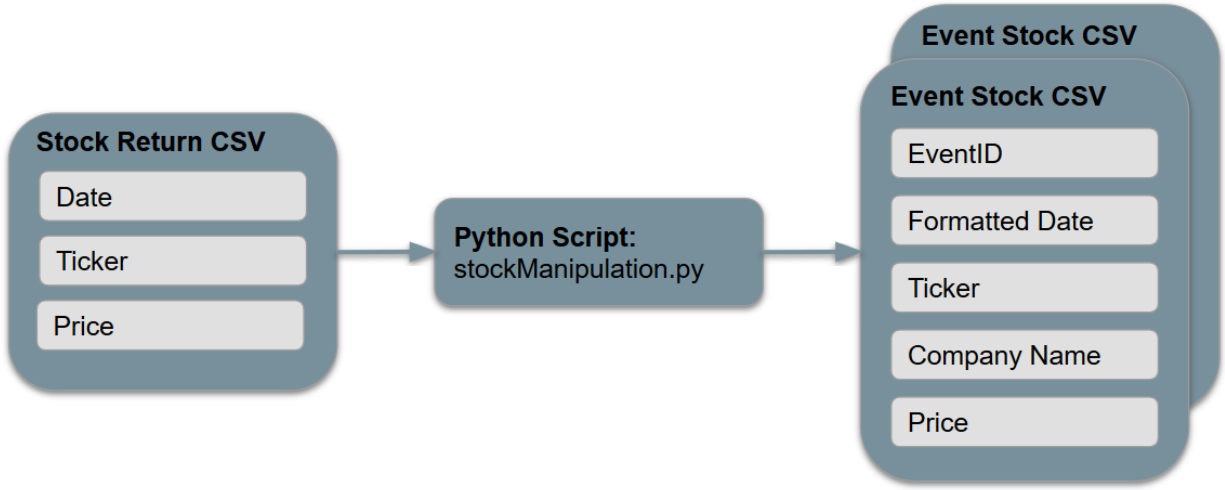

Figure 5: Stock Collection Process

4.4 Tweet Analysis

Figure 6 and Figure 7 illustrate our tweet analysis process. The process in Figure 6 determines user sentiment for a group of CSVs containing tweet data. The process in Figure 7 determines if a URL exists and the number of URLs for a group of CSVs containing tweet data. Refer to user_sentiment.py and countURLs.py in the Appendix.

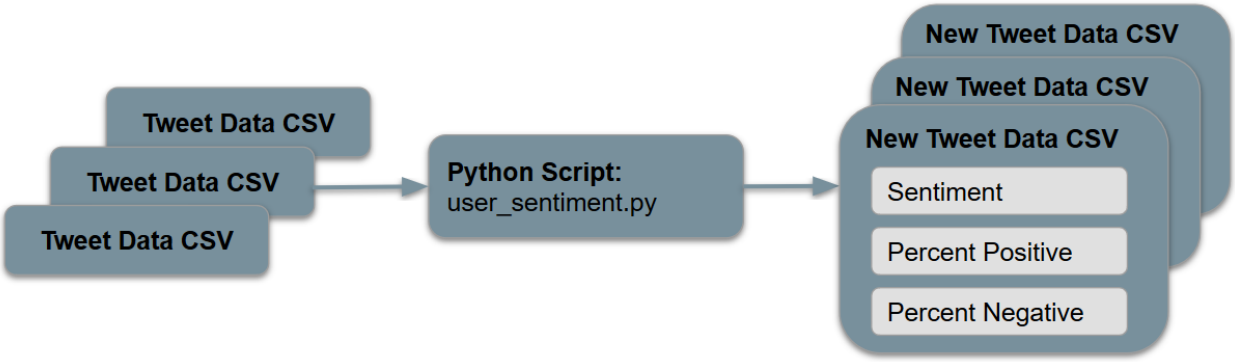

Figure 6: User Sentiment Analysis Process

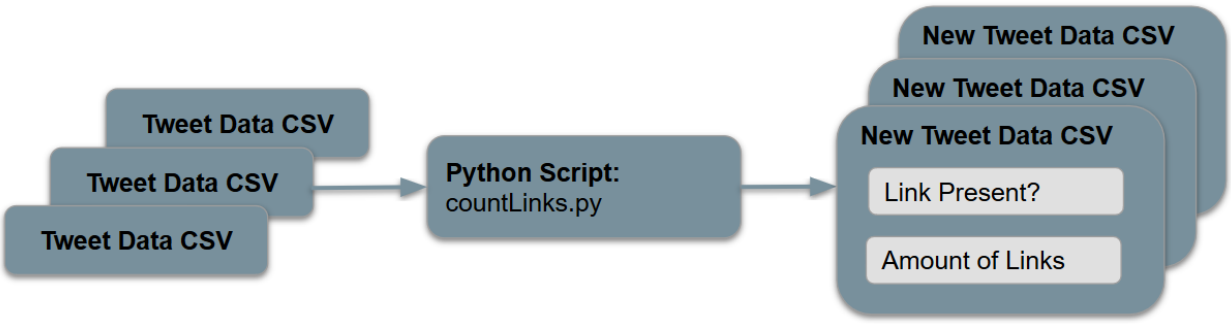

Figure 7: Link Counting Process

4.5 Fama French Model

A very popular model used to predict stock performance is the Fama French Model [2]. Our client instructed us to use this model so that is why we chose this model over other models that could also be used. Our goal of using this model was to be able to predict what the stock performance of a firm would have been had there never been a data breach, and compare that to what the stock performance actually was. The model can be seen in Figure 8 and explains each variable that makes up the model [5].

$$R_{it} - R_{ft} = \alpha_{it} + \beta_{iM}(R_{Mt} - R_{ft}) + \beta_{is}SMB_t + \beta_{ih}HML_t + \varepsilon_{it}$$

R_{it} is the total return of individual stock/portfolio i

R_{ft} is the risk free asset return

R_{Mt} is the total market portfolio return

$R_{it} - R_{ft}$ is expected excess return

$R_{Mt} - R_{ft}$ is the excess return on a market portfolio index

SMB_t is the size premium

HML_t is the value premium

Figure 8: Fama French Model

While all of the variables defined above are given by the overall stock market, the alpha and beta values are trainable variables for each particular stock. These values are formed through a similar process as linear regression over the course of 150 data points

or 150 stock return days. Once the model was trained we then used our estimated alpha and beta values to plug into the equation and the formula would then compute the stock return on the days of the breach and after the breach. We took these estimated data points and compared them to the actual stock performance on those dates. We did this to see which companies were able to minimize their stock failure after a data breach occurs.

The importance of the model in Figure 8 is that it gets rid of many confounding variables that could happen in our analysis if we just looked at which stocks fell the most. The factors represented in the model in Figure 8 take into consideration the size of the companies, different stock values, and other effects. They give us a more accurate way of predicting how much the stock changed.

5. Implementation

5.1 Twitter Component

5.1.1 Acquiring the Data

The gathering of Twitter data was accomplished using a Python script utilizing the GetOldTweets API. Two Python scripts were written, one collecting company Twitter data called scrape_company_tweets.py and another collecting user tweets based on specific keywords called keyword_tweets.py. Both scripts took an input CSV file which held data of specific breach events containing information such as the breach date, company name, company Twitter handle, and specific eventID. This input file is parsed by our script, and start and end dates for scraping are set. Company tweets are collected 120 before and 30 days after the breach event. The user tweets are collected 10 days before and 30 days after the event. The user tweets are also parsed and filtered for specific keywords given in Table 1. These collected tweets are then output to CSV files labeled with the eventID and company Twitter handle.

5.1.2 Additional Data Collection

After collecting basic tweet data through the GetOldTweets API, it was necessary to do some additional data collection. To accomplish this two Python scripts were written, profile_scrape.py and announcement_reply_firm.py. The profile_scrape.py script utilized the Requests and Beautiful Soup libraries to gather additional information on the users in the keyword tweet files that were produced by keyword_scrape.py. Specifically, it added the user’s username, bio, following count, follower count, and verified status to

each row of these files. Then, the announcement_reply_firm.py script was run on all company tweet CSV files that were produced by scrape_company_tweets.py. Using the value under the Mentions header that had been retrieved using GetOldTweets, it determined whether or not a tweet was an announcement to all users or a reply to another user's tweet. The resulting value (either Announcement or Reply) was appended by the script to the tweet's row.

5.1.3 Tweet Analysis

After collected and filtering our data, we analyzed our tweets based on two criteria. The first criteria was to check the sentiment of the tweets. The second was to count the number of URLs present in each tweet. Both these criteria were satisfied by writing Python scripts that appended to our CSVs containing Twitter data. User sentiment was calculated by using the TextBlob API [7]. A Naive Bayes analysis was conducted on each tweet, and sentiment being positive or negative was recorded. The percent positive and percent negative for each tweet was also recorded. In order to count the URLs each tweet data CSV file was input to our Python script which analyzed each row of tweets for a URL. Two columns were appended to our CSV file; one containing a value if a URL was present in the tweet, and another containing the number of links it found in the tweet.

5.2 Stock Component

5.2.1 Acquiring the Data

To gain meaningful insight into the effect of a company's responses to data breaches, we had to analyze the change in stock prices after release of information. We provided our client with a list of every company involved in data breaches since 2006 (Table 2). In a CSV file, we included each company name along with its stock ticker. Using this data, our client generated a CSV named stockReturn.csv with the previous 10 years of stock data for each company. This file included a row for every day a company's stock was traded, with attributes including company name, date, ticker, and closing price. This amounted to 1006614 rows of information.

5.2.2 Scrubbing the Data

The CSV of stock data contained far more data than necessary for our later calculations. We needed to filter down this data to only include the dates surrounding the data breach events. The formula we used to detect anomalies in stock prices, which will be discussed in the Applying Fama French section of the report, requires the stock prices of the company in a range from 120 before to 30 days after the event. The date format found in the CSV was YYYYMMDD, whereas our master CSV of data breach

events had a date format of MM/DD/YYYY. The first step in processing the data was to map all the dates in the stock CSV file to the MM/DD/YYYY format. This was accomplished within Excel, using the format cells functionality.

Next, we wrote a Python script to manipulate the data into deliverables that were in turn fed into the stock analysis formula. Using the Pandas library [4], we read in stockReturn.csv and dataBreachesActive.csv as Pandas DataFrames. Next, we create two new attributes within the data breach DataFrame - StartDate and EndDate. These columns will hold the boundary values for our timeframe for each given data breach. We iterate over the rows in dataBreachesActive.csv and use the datetime Python library to calculate the date 120 days before and 30 days after the date found in the 'Date Made Public' attribute, storing these values in new columns within the CSV.

The next step in the process was to iterate over the rows again, this time outputting a new CSV file specific to each EventId associated with a company and data breach. It would not be sufficient to create an output file for each company, because some companies experienced multiple data breaches, meaning that we need a set of 150 rows for each of these events. We temporarily filtered our stockReturn.csv to only contain the rows of information pertaining to the company involved in the current security breach. We filtered again on these rows, removing all the days that weren't within our 150 day range for the current data breach. Once we had our required rows, we removed unnecessary columns ('oldDate' and 'PERMNO'). We created a string to represent the filename using the EventId concatenated with the company's name. Finally, we generated the result as a CSV and repeated the process until every row had been processed. Each data breach row was mapped to a new CSV file, containing the desired 150 day range of stock values with each row containing columns EventId, Date, Ticker, and Name.

5.2.3 Applying Fama French

Once the stock files were collected we were able to start training our Fama French Model and fitting it to the Fama French model. We trained our Fama French Model from one hundred and fifty days before the data breach to ten days before the event. Then we wanted to analyze the predictive model from three days before the breach to three days after. We used the three and five factor model which just gets rid of the last two variables from the figure in the Design section of the report. We were then able to see how much the stock price should have been versus what it was.

5.2.4 Further Stock Analysis

The result of the Fama French file was a CSV containing numbers representing how abnormal each company's stock performed 7 days before and after each date of the

data breach event. The next step in the process was to find events in which company's stock performed abnormally poorly or abnormally well. We accomplished this through the use of a short Python script, abnormal.py, which can be found in Appendix A. We found the mean of the values 3 days after the event and subtracted the mean of the values 3 days before the event to find the change in stock abnormality, stored in the diff column of the output. The output was a CSV file named abnormalDif.csv , which contained a row for each data breach event and included company ticker, evtdate, and diff values. This table can be found in Appendix B as Table 3. Data breach events with diff values close to 0 can be interpreted as having a very small change in how abnormal their stock performed before and after the data breach event. Companies with positive values for diff had abnormal good stock performance after the data breach event when compared to their performance before the event. Lastly, companies with negative values for diff exhibited stock performance that was abnormally poor after the data breach event when compared to their performance before the event.

6. Assessment

After outputting all of our differences of stock performance abnormalities, which were explained in section 6.2.4 we had finally collected all of our data and analysis and could start preparing our company guide. We realized that we wouldn't be able to apply all of our analysis on every single data breach because the analysis would have taken weeks to complete due to the amount of data we were analyzing. Therefore we decided we were going to run our analysis on the companies that had the best and worst abnormal differences. We didn't want to pick an arbitrary number of companies so we used Z scores to narrow down our company list. After computing the mean and standard deviation of the abnormal differences we decided that the Z score that would allow for us to run our analysis would be companies 2.5 standard deviations above and below the mean. This left us with the six lowest abnormal differences and the top seven abnormal differences. The bottom six data breaches are listed in Figure 9; the top seven data breaches are in Figure 10.

ticker	evtdate	diff	Z
FRP	20-Apr-09	-0.129693	-7.786311
XRIT	11-Apr-12	-0.116467	-6.989078
HPY	20-Jan-09	-0.093973	-5.633298
INOD	13-Jan-09	-0.075830	-4.539713
EFX	7-Sep-17	-0.068093	-4.073388
DYN	21-Oct-16	-0.050297	-3.000699

Figure 9: Bottom Six Companies. Ticker is the stock ticker, evtdate is the day of the data brach, diff is the abnormal stock difference after and before the breach, and Z is the score in relation to the mean of abnormal differences.

ticker	evtdate	diff	Z
SHLD	10-Oct-14	0.083793	5.081540
RAD	19-May-17	0.069943	4.246735
ORCL	12-Nov-07	0.048193	2.935760
NFP	30-Oct-06	0.045600	2.779447
TWTR	13-Jun-16	0.044460	2.710734
PRAN	8-Mar-17	0.043583	2.657893
SEAC	8-Sep-10	0.042507	2.592997

Figure 10: Top Seven Companies. Ticker is the stock ticker, evtdate is the day of the data brach, diff is the abnormal stock difference after and before the breach, and Z is the score in relation to the mean of abnormal differences.

Once these companies were narrowed down we ran the sentiment analysis and user profile scraping on all of the tweets associated with each company. One hardship was

that any data breach before 2010 had a sparse data set. We did our best to work around this issue. The analysis for the user tweets of each data breach is in fFgure 11. If a data breach was in the top seven or bottom six but is no longer there then that means there was no Twitter data available due to the lack of tweeting around that data breach.

Date	Ticker	Diff	TotalFollowers	TotalFollowing	VerifiedUsers	TotalNeg	TotalPos	TotalNegPercent	TotalPosPercent	TotalLinkCount
20-Jan-09	HPY	-0.093973	1821848	196815	11	109	26	0.710091	0.289909	109
20-Apr-09	FRP	-0.129693	161172	89879	1	1	5	0.306338	0.693662	2
10-Oct-14	SHLD	0.083793	0	0	0	166	3	0.676407	0.323593	168
13-Jun-16	TWTR	0.044460	0	0	0	2868	6221	0.392013	0.607987	7815
21-Oct-16	DYN	-0.050297	201610966	21573015	NaN	1741	5094	0.320016	0.679691	6033
8-Mar-17	PRAN	0.043583	15811	7638	1	3	7	0.316661	0.683339	2
7-Sep-17	EFX	-0.068093	510949418	58584962	1024	173	75	0.640714	0.359286	17628
19-May-17	RAD	0.069943	0	0	0	3	15	0.139364	0.860636	17

Figure 11: The column headers explain the meaning of each. When it says Total it means all the user tweets summed together. Percentages are divided by total tweet count.

The analysis for the company tweets of each data breach is in Figure 12. The same thing applies for missing breaches in this figure as well.

Date	Ticker	Diff	TotalLinkCount	NumReplies	NumAnnouncements	TotalTweets	RatioReplyTotal
10-Oct-14	SHLD	0.083793	55	55	0	55	1
13-Jun-16	TWTR	0.044460	7	5	22	27	0.185185
21-Oct-16	DYN	-0.050297	201	108	140	248	0.435484
8-Mar-17	PRAN	0.043583	8	9	17	26	0.346154
7-Sep-17	EFX	-0.068093	230	71	1407	1478	0.0480379
19-May-17	RAD	0.069943	0	3	21	24	0.125

Figure 12: The column headers explain the meaning of each. When it says Total it means all the company tweets summed together. Percentages are divided by total tweet count.

We found some correlation between ratio of replies to total tweets and the stock performance as well as user sentiment and stock performance. The company with the best overall stock difference had the highest ratio of replies to total tweets while the company with the worst stock performance had the lowest ratio. Also when comparing the user sentiment of the bottom six to the top seven we realized that the mean of positive sentiment of the top seven was significantly higher than the bottom six. A graph showing this can be seen in Figure 13.

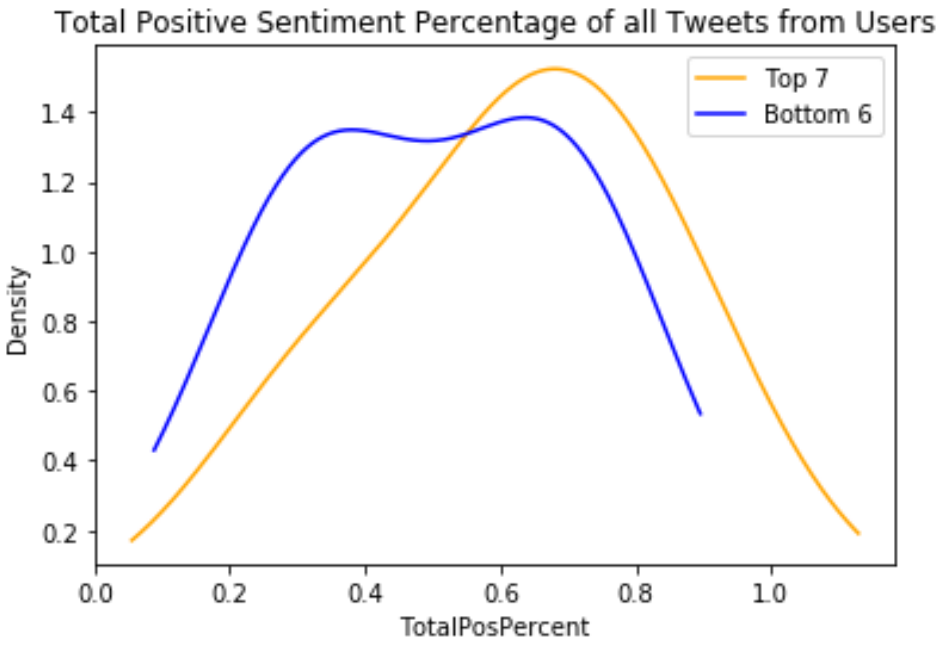

Figure 13: Graph of positive sentiment from user tweets comparing the bottom six breaches to the top seven breaches.

From these two main findings we have a few main points for companies to consider when announcing a data breach. The main focus of social media should be making replies to worried user's, instead of announcement tweets. The main way to make minimal announcements may be to make sure that company announcements are well thought out and cover any questions that could come up at a later time. Company's shouldn't hastily make announcements but should ensure that an announcement will be covering a magnitude of problems. This may also lower the number of tweets from users that are replies, which will make it easier to reply to all of their concerns. Another reason why to focus on replying, and making clear, concise, and few announcements, is to keep user sentiment positive. The reason why this can effect user sentiment may be that when a company looks to have the data breach under control and can make few announcements, then the users will believe that the company will fix the issue. Also replying to the user tweets may keep their sentiment positive because it demonstrates that the company cares about its users and fixing this issue.

7. Developer and User Manual

7.1 File Inventory

7.1.1 Tweet Data Collection Files

- requirements.txt

    - List of requirements that must be installed on your machine in order to run the GetOldTweets code.
- keywords.txt
    - List of keywords to be used and searched for in keyword_scrape.py.
    - Delimit keywords with a newline character.
- FindAccountNamesActive.csv
    - List of data breaches to be used by scrape_company_tweets.py and keyword_scrape.py.
    - Figure 14 shows the header layout for the file.

Event ID	Company Ticker Symbol	Company Name	Breach Date	Company Twitter Handle

Figure 14: Column headers for FindAccountNamesActive.csv

- scrape_company_tweets.py
    - Takes FindAccountNamesActive.csv as an input argument and outputs a CSV file for every row in FindAccontNamesActive.csv.
    - Each output CSV file contains every tweet made by the company in the row from 120 days before the breach date to 30 days after the breach date.
    - Each output CSV file row contains the tweet's date, text, number of retweets, number of favorites, mentions, and hashtags.
    - Run using "python scrape_company_tweets.py CSVFILE.csv".
- keyword_scrape.py
    - Takes FindAccountNamesActive.csv and keywords.txt as input arguments and outputs a CSV file for every row in FindAccontNamesActive.csv.
    - Each output CSV file contains every tweet within 10 days before the breach date and 30 days after the breach date that contains either the company's name and a keyword, or the company's Twitter handle and a keyword. These tweets can be from any user.
    - Each output CSV file row contains the tweet's date, text, number of retweets, number of favorites, mentions, hashtags, and ID.
    - Run using "python keyword_scrape.py CSVFILE.csv KEYWORDFILE.txt".
- announcement_reply_firm.py
    - Determines if a tweet is a reply or announcement for each tweet in the CSV files produced by scrape_company_tweets.py.

    - Runs on all CSV files in the same directory as the script. To use, place all desired CSV files in a directory with the script and run using “python announcement_reply_firm”.
    - Appends to each row in the CSV files whether the tweet is a reply or announcement.
- profile_scrape.py
    - Uses Requests and Beautiful Soup to collect data on the users who tweeted in the keyword tweet CSV files produced by keyword_scrape.py.
    - Runs on all CSV files in the same directory as the script. To use, place all desired CSV files in a directory with the script and run using “python profile_scrape.py”.
    - Appends to each row in the CSV files the username of the user who tweeted, their bio, their following count, their follower count, and whether or not they are a verified user (0 for not verified, 1 for verified).

Please refer to Figure 2 in the Design section for an illustration of the tweet data collection process.

7.1.2 Stock Data Collection Files

- DataBreachesActive.csv
    - List of data breaches to be used by stockManipulation.py
    - Row format is “Event ID”, “Company Ticker Symbol”, “Breach Date”, “Company Name”.
    - Figure 15 shows the header layout for the file.

Event ID	Ticker	Breach Date	Company Name

Figure 15: Column headers for DataBreachesActive.csv

- stockReturn.csv
    - Raw stock data file containing every stock value since 2005.
    - Figure 16 shows the header layout for the file.

PERMNO	Date	Ticker	Company Name	Stock Price

Figure 16: Column headers for stockReturn.csv

- stockManipulation.py
    - Takes DataBreachesActive.csv and stockReturn.csv as input and outputs a CSV file for every row in DataBreachesActive.csv.
    - Each output CSV file contains the stock data for the company in the row from 120 days before the breach date to 30 days after the breach date.
    - Each output CSV file row contains the Event ID, Stock Price Date, Stock Ticker Symbol, Company Name, and Stock Price
    - Run using “python stockManipulation.py”.

Please refer to Figure 3 in the Design section for an illustration of the stock data collection process.

7.1.3 Data Analysis Files

- -3to3.csv
    - Contains the stock abnormality values from 3 days before to 3 days after a company’s breach. Values were calculated using the Fama French Model. Provided to us by our client.
    - Figure 17 shows the header layout for the file.

Ticker	Breach Date	Stock Date	Abnormality

Figure 17: Column headers for -3to3.csv

- abnormalDif.csv
    - Contains the difference of the average abnormality after the breach and average abnormality before the breach for each breach using the values from -3to3.csv .
    - Figure 18 shows the header layout for the file.

Ticker	Breach Date	Difference

Figure 18: Column headers for abnormalDif.csv

- user_sentiment.py
    - Uses the TextBlob library to calculate sentiment values for every tweet in your keyword tweet CSV files produced by keyword_scrape.py.

    - Runs on all CSV files in the same directory as the script. To use, place all desired CSV files in a directory with the script and run using “python user_sentiment.py”.
    - Appends to each row in the CSV files the overall sentiment, the positive sentiment value, and the negative sentiment value.
- Company_sentiment.py
    - Uses the TextBlob library to calculate sentiment values for every tweet in your keyword tweet CSV files produced by scrape_company_tweets.py.
    - Runs on all CSV files in the same directory as the script. To use, place all desired CSV files in a directory with the script and run using “python company_sentiment.py”.
    - Appends to each row in the CSV files the overall sentiment, the positive sentiment value, and the negative sentiment value.
- abnormal.py
    - Takes -3to3.csv as input and outputs abnormalDif.csv.
    - Finds the Top 7 and Bottom 6 Data Breaches based on the Z Score.
    - Produces plots of our differences compared to sentiment and replies.
    - Run using “python abnormal.py CSVFILE.csv”.
- countURLs.py
    - Determines how many links are present in the body of a tweet. Used for the CSV files produced by both scrape_company_tweets.py and keyword_scrape.py.
    - Runs on all CSV files in the same directory as the script. To use, place all desired CSV files in a directory with the script and run using “python countURLs.py”.
    - Appends to each row in the CSV files if there is a link or not in the tweet (0 or 1), and how many links are in the tweet.

7.2 Installation Tutorial

1. Create a GitHub account if you don’t already have one.

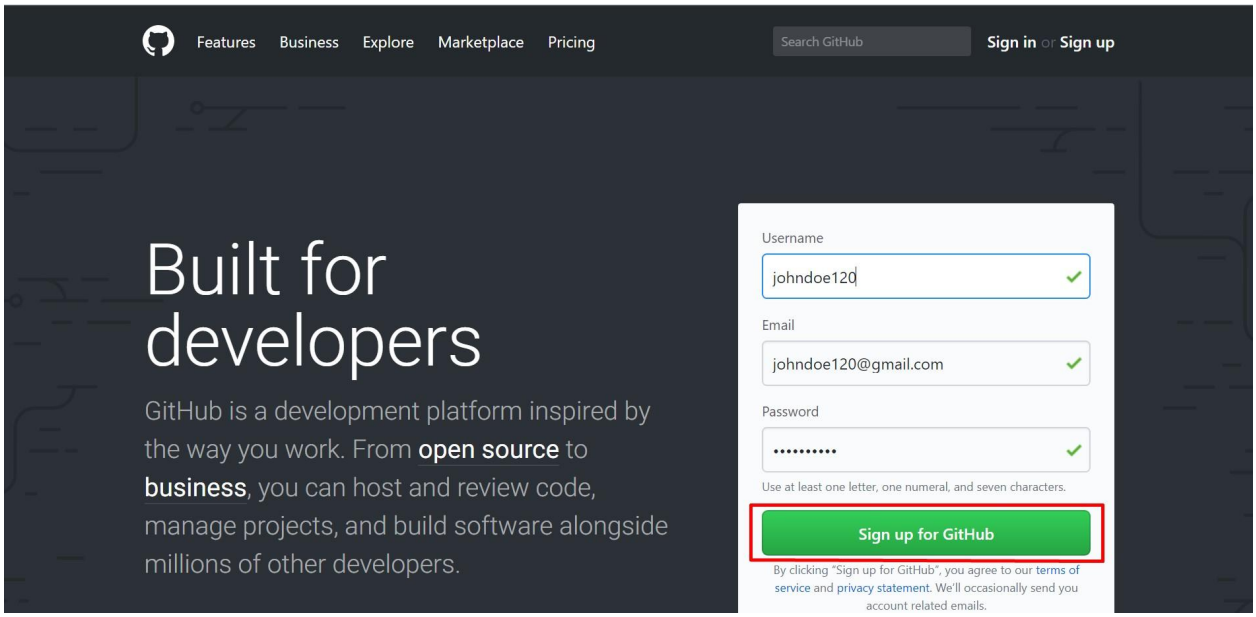

Figure 19: Github account creation page

2. Fork a copy of the GitHub repository located at https://github.com/Jefferson-Henrique/GetOldTweets-python/.

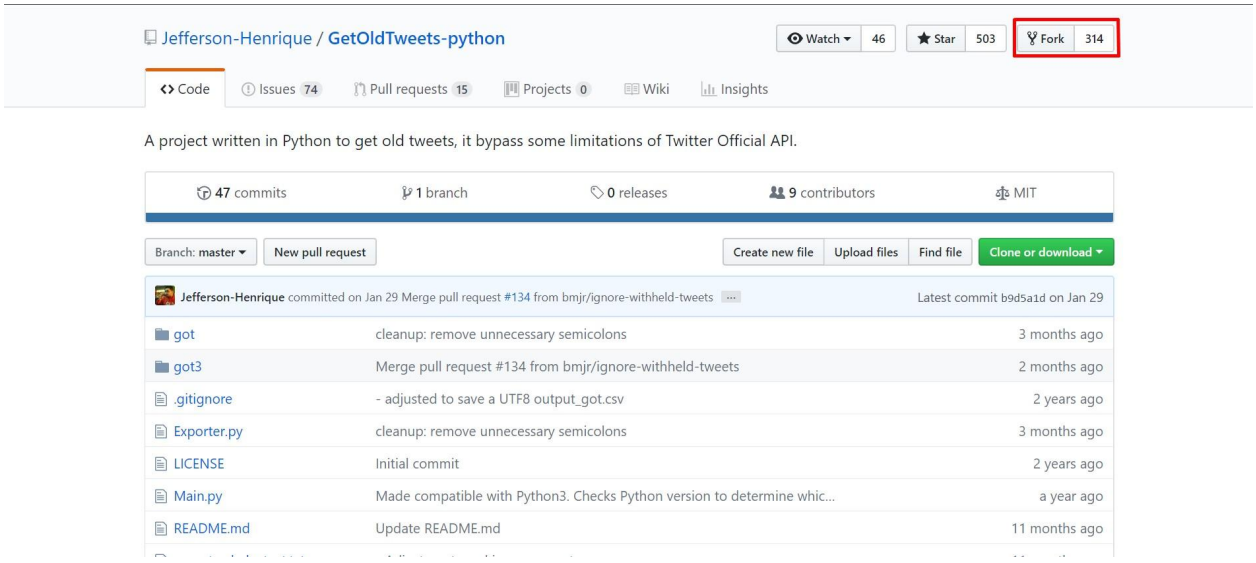

Figure 20: GetOldTweets-python API Github page

3. Clone the repository to your local machine.

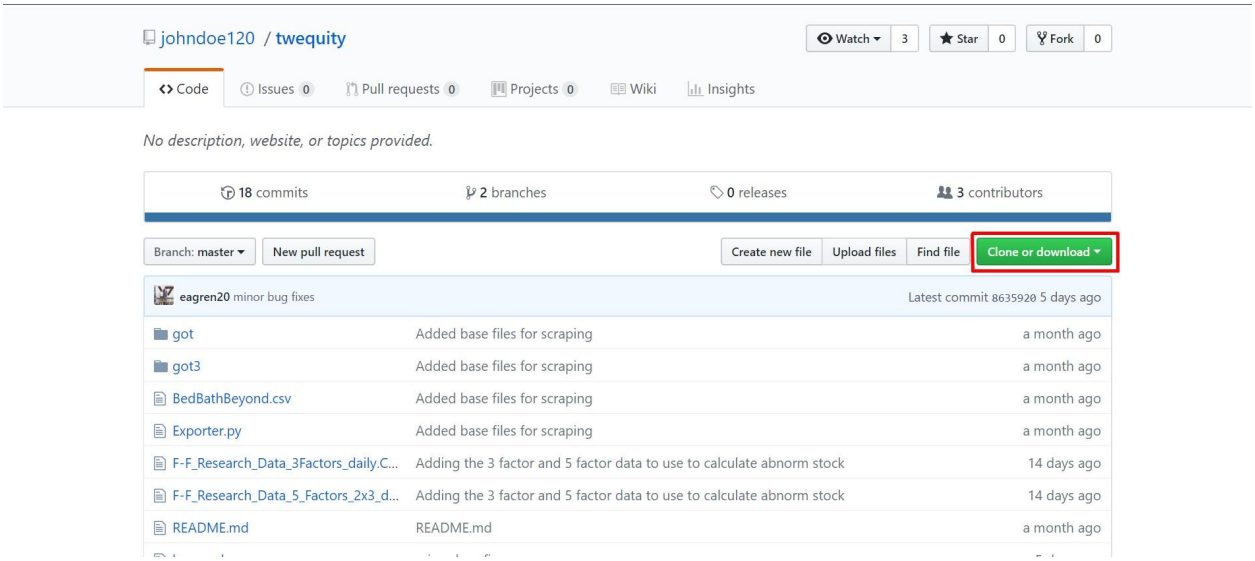

Figure 21: Cloning the Twequity repository from Github

4. Install Python on your machine if you don't already have it.

```
eagren20@DESKTOP-J84N2IU:~$ sudo apt-get install python
```

Figure 22: Installing Python using the command line

5. Add all of the files listed in the file inventory to your local repository.

Figure 23: Directory containing the required files

6. Run "pip install -r requirements.txt" on your machine.

```
eagren20@DESKTOP-J84N2IU:~$ pip install -r requirements.txt
```

Figure 24: Installing the project requirements using the command line

7. Install the packages required for the additional tweet data collection and data analysis scripts: Requests, Beautiful Soup, and TextBlob.

   Please run: "sudo pip install requests"
   "sudo apt-get install python-bs4"
   "sudo pip install -U textblob"
   "python -m textblob.download_corpora"

```
eagren20@DESKTOP-J84N2IU:~$ sudo pip install requests
eagren20@DESKTOP-J84N2IU:~$ sudo apt-get install python-bs4
eagren20@DESKTOP-J84N2IU:~$ sudo pip install -U textblob
eagren20@DESKTOP-J84N2IU:~$ python -m textblob.download_corpora
```

Figure 25: Series of commands executed to download remaining dependencies

8. You are now ready to begin running the Python scripts for both collection and analysis of the tweet/stock data.

8. Lessons Learned

8.1 Timeline

Our timeline was split into five different milestones in order to help us get the project done in a timely manner. The first milestone was to gather company/user tweet data, which went smoothly. The second milestone involved gathering company stock data for each of the data breaches. Furthermore, the third milestone included analyzing the stock prices of the companies during the event. The fourth milestone consisted of analyzing company successes and failures, while the last milestone was to come up with a guide for companies that have been breached. Overall, these milestones were very effective and helped us gain a good sense of our progress during the project. The only problem we had with our timeline was that we had new requirements added to the project later on in the semester, which hindered our time budgeting and caused us to have less time to work on the remaining milestones.

8.2 Team Member Roles

In our project, Jacob Smith was the lead editor. His responsibilities involved looking over all work and making sure that our writing was grammatically correct and relevant. Jacob also checked for any errors in our assignments and turned in all of our assignments as well. Erik Agren was the head of testing. He was in charge of writing all the Python scripts and sending the CSVs back to the team after the scripts were run. Christian was the project lead and helped in all phases of the project. He helped organize the project and constantly checked in with other team members to make sure everyone was on track. Nathaniel Guinn was the designated note taker. His responsibilities involved taking notes during group meetings so that the team could look at the notes and understand what went on during each meeting. Rohan was the presentation lead. His role involved organizing the presentations throughout the semester and making sure the presentations accurately reflected our group's current progress.

8.3 Problems and Solutions

One of the problems we encountered while scraping for data on Twitter was the scarcity of tweets around 2008. Back then, Twitter was not as popular, so most companies either didn't have a Twitter account or didn't use it to talk to customers over the social network. This makes it harder for us because there is sparse data to look at for breaches that occurred before 2010. We will have to be very cautious with our recommendations based on some of the breaches in the early 2000's based on the small amount of tweets.

Another problem we encountered was also with changes in Twitter. In 2016, Twitter changed the way mentions and replies were presented. Replying to tweets did not show up as actual tweets and in order to find the replies you have to go to the original tweet instead of having the reply show up as a tweet on the user's page. This means that if a company replied to a user, it wouldn't show up on the company's page but instead just under the original tweet. Also, mentions on Twitter worked the same way and did not show up as actual tweets on the company's page. This problem was easily solved; it just made us made changes for tweets past 2016 to account for mentions and replies to other user's tweets.

8.4 Future work

Throughout the course of the project, we used Google Drive as our data sharing platform. Our team drive stored not only our presentations but also all of our data which consisted of hundreds of CSV files. As we added more CSV files to the drive, it started

to become very slow and caused formatting issues as well. We would suggest using a different data sharing platform in order to make file sharing easier and more fluid.

Furthermore, at the end of our project when we were running scripts, it would take a very long time to look at thousands of tweets. We would suggest adding parallelization to the scripts in order to run more than one at the same time. This would save days of running scripts and since that would allow more data to be collected, we would then be able to analyze more data. Spending more time on data analysis would also help us provide a more accurate and in-depth company guide, which could help companies deal with data breaches in an effective way.

9. Acknowledgements

We would like to thank our client Ziqian Song for all of her help on the project; she can be contacted at ziqian@vt.edu. She has been instrumental with regards to training the stock data in order to predict what the stock would have been if no data breach occurred. Thanks also go to our professor Dr. Fox and our teaching assistant Jin.

10. References

1. Gressin, Seena. “The Equifax Data Breach: What to Do.” *Consumer Information*, 13 Mar. 2018, www.consumer.ftc.gov/blog/2017/09/equifax-data-breach-what-do. Accessed 18 Mar. 2018
2. Hendricks, Kevin, et al. “Article Tools.” *Management Science*, Institute for Operations Research and the Management Sciences, 14 Oct. 2015.
3. Lee, Lian Fen, et al. “The Role of Social Media in the Capital Market: Evidence from Consumer Product Recalls.” *Journal of Accounting Research*, 27 Mar. 2015.
4. Wes McKinney. Data Structures for Statistical Computing in Python, Proceedings of the 9th Python in Science Conference, 51-56 (2010)
5. Davidson, Adrian. “FAMA-FRENCH MODEL Concept and Application.” *SlidePlayer*, 10 Aug. 2017 slideplayer.com/slide/9516030/. Accessed 01 May 2018
6. Henrique, J. GetOldTweets - Python. Github. 2018. https://github.com/Jefferson-Henrique/GetOldTweets-python. Accessed 04 Feb 2018.
7. Loria, S. et al. TextBlob: Simplified Text Processing. 2018. http://textblob.readthedocs.io/en/dev/authors.html. Accessed 04 April 2018.
8. Nair, Vineeth G. *Getting Started with Beautiful Soup*. Packt Publishing Ltd, 2014.
9. Reitz, K. Requests: HTTP for Humans. 2018. http://docs.python-requests.org/en/master/. Accessed 21 April 2018

11. Appendices

Appendix A: Code

Python File, scrape_company_tweets.py

```
1.  import sys
2.  import got
3.  import csv
4.  import itertools
5.  from datetime import datetime, timedelta
6.  from dateutil import parser
7.
8.  #Sets some intial lists and variables
9.  dates = []
10. handles = []
11. eventIDs = []
12.
13. days_before = 120
14. days_after = 30
15.
16. #Checks if a input CSV file was given. If not exits the program
17. if len(sys.argv) == 1:
18.     print "Missing input CSV file"
19.     sys.exit(0)
20.
21. #Opens the CSV file and appends important data to the lists
22. with open(sys.argv[1]) as csvfile:
23.     readCSV = csv.reader(csvfile, delimiter=',')
24.     for row in readCSV:
25.         print row[3]
26.         date = datetime.strptime(row[3], "%m/%d/%Y").strftime("%Y-%m-%d")
27.         dates.append(date)
28.         handles.append(row[4])
29.         eventIDs.append(row[0])
30.
31. #Iterates over each list and scrapes Twitter using GetOldTweets API
32. for date, handle, ID in itertools.izip(dates, handles, eventIDs):
33.     event_date = parser.parse(date)
34.
35.     #Calculates the start and end date based on the event date
36.     start_date = (event_date - timedelta(days=days_before)).strftime("%Y-%m-%d")
37.     end_date = (event_date + timedelta(days=days_after)).strftime("%Y-%m-%d")
38.
39.     print handle + ":"
40.     print "Event Date: ", event_date
41.     print "Start Date: ", start_date
42.     print "End Date: ", end_date
43.
44.
```

```
    tweetCriteria = got.manager.TweetCriteria().setUsername(handle).setSince(start_date).setUntil(end_date)
    tweets = got.manager.TweetManager.getTweets(tweetCriteria)

    #Prints some statistics and creates a new CSV file to append information to.
    print "Total Tweets: ", len(tweets)
    filename = str(ID) + "_" + handle + ".csv"

    with open(filename, "w") as output:
            writer = csv.writer(output, delimiter=',')
            for t in tweets:
                row = t.date, t.text, t.retweets, t.favorites, t.mentions, t.hashtags
                writer.writerow([unicode(s).encode("utf-8") for s in row])
```

Python File, keyword_scrape.py

```
import sys
import got
import csv
import itertools
from datetime import datetime, timedelta
from dateutil import parser

dates = []
names = []
handles = []
eventIDs = []
keywords = []
tweets = []

days_before = 10
days_after = 30

if len(sys.argv) != 3:
    print "run using the following command line arguments: python keyword_scrape.py CSVFILE.csv KEYWORDFILE.txt"
    sys.exit(0)

if (not('.csv' in sys.argv[1]) or not('.txt' in sys.argv[2])):
    print "run using the following command line arguments: python keyword_scrape.py CSVFILE.csv KEYWORDFILE.txt"
    sys.exit(0)


with open(sys.argv[1]) as csvfile:
    readCSV = csv.reader(csvfile, delimiter=',')
    for row in readCSV:
        date = datetime.strptime(row[3], "%m/%d/%Y").strftime("%Y-%m-%d")
        dates.append(date)
        handles.append(row[4])
        names.append(row[2])
```

```
        eventIDs.append(row[0])

with open(sys.argv[2]) as keywordFile:
    lines = keywordFile.read().splitlines()
    for line in lines:
        keywords.append(line)

for date, handle, ID, name in itertools.izip(dates, handles, eventIDs, names):
    event_date = parser.parse(date)

    start_date = (event_date - timedelta(days=days_before)).strftime("%Y-%m-%d")
    end_date = (event_date + timedelta(days=days_after)).strftime("%Y-%m-%d")

    print handle + ":"
    print "Event Date: ", event_date
    print "Start Date: ", start_date
    print "End Date: ", end_date

    tweetCriteria = got.manager.TweetCriteria().setSince(start_date).setUntil(end_date)


    #build tweet query
    query = ''
    #add company name queries
    for keyword in keywords:
        query = query + name + ' AND ' + keyword +' OR '
    #add company handle queries
    for keyword in keywords:
        query = query + handle + ' AND ' + keyword +' OR '

    #get rid of OR at end
    query = query[:-3]
    #turn it into a list
    queries = query.split(' OR ')

    #loop through queries and collect tweets for each
    ids = set()
    noDupTweets = []
    for q in queries:
        #print 'Query: '+ q
        keywordCriteria = tweetCriteria.setQuerySearch(q)
        tweets = got.manager.TweetManager.getTweets(keywordCriteria)
        #remove duplicates
        for tweet in tweets:
            if not tweet.id in ids:
                ids.add(tweet.id)
                noDupTweets.append(tweet)

    print "Total Tweets: ", len(noDupTweets)
    filename = str(ID) + "_" + handle + "_keywords" + ".csv"
```

```
84.
85.     with open(filename, "w") as output:
86.         writer = csv.writer(output, delimiter=',')
87.         for t in noDupTweets:
88.             row = t.date, t.text, t.retweets, t.favorites, t.mentions, t.hashtags, t.id
89.             writer.writerow([unicode(s).encode("utf-8") for s in row])
```

Python File, profile_scrape.py

```
1.  import sys
2.  import csv
3.  import os
4.  import glob
5.  path = "*.csv"
6.
7.  #Checks to see if all imports and installed
8.  try:
9.      import bs4
10. except ImportError:
11.     raise ImportError('BeautifulSoup needs to be installed. Please run "sudo apt-get
    install python-bs4"')
12. except AttributeError:
13.     raise AttributeError('bs4 needs to be upgraded. Please run "pip install --upgrade
    beautifulsoup4"')
14. try:
15.     import requests
16. except ImportError:
17.     raise ImportError('Requests needs to be installed. Please run "sudo pip install
    requests"')
18.
19. #Iterates over every CSV file in the current directory
20. for fname in glob.glob(path):
21.     if (fname != 'temp.csv'):
22.         #Opens each csv file twice once to read and once to write
23.         with open(fname) as csvfile :
24.             readCSV = csv.reader(csvfile, delimiter=',')
25.             with open('temp.csv', "w") as output:
26.                 print 'file: ' + fname
27.                 #Iterates over every row of tweets in an individual CSV file
28.                 for row in readCSV:
29.                     #Pushes a request towards a Twitter API based on a Tweet ID
30.                     url = 'https://twitter.com/FalcoLombardi/status/' + row[6]
31.                     page = requests.get(url)
32.                     soup = bs4.BeautifulSoup(page.text, 'html.parser')
33.
34.                     usernameTag = soup.find('b', {'class':'u-linkComplex-target'})
35.
36.                     #Attempts to grab user information from the requested page.
37.                     #If the user information is not avaliable preset all the
    information
38.                     try:
39.                         username = usernameTag.text.encode('utf-8')
```

```
                    except AttributeError:
                        username = 'deleted'
                        bio = 'deleted'
                        following = 0
                        followers = 0
                        verified = 0
                    else:
                        url = 'https://twitter.com/' + username

                        page = requests.get(url)
                        soup = bs4.BeautifulSoup(page.text, 'html.parser')
                        bioTag = soup.find('p', {'class':'ProfileHeaderCard-bio u-dir'})
                        bio = bioTag.text.encode('utf-8')
                        followersTag = soup.find('a', {'data-nav':'followers'})
                        followingTag = soup.find('a', {'data-nav':'following'})
                        verifiedTag = soup.find('span', {'class':'ProfileHeaderCard-badges'})

                        try:
                            following = followingTag['title'].split(' ')[0]
                        except TypeError:
                            following = 0
                        try:
                            followers = followersTag['title'].split(' ')[0]
                        except TypeError:
                            followers = 0

                        verified = 1
                        if (verifiedTag is None):
                            verified = 0
                    #Writes user information containing, bio, username, following, followers, and verified status to the CSV file
                    writer = csv.writer(output, delimiter=',')
                    r = row[0], row[1], row[2], row[3], row[4], row[5], row[6], username, bio, following, followers, verified
                    writer.writerow([s for s in r])
        os.rename('temp.csv', fname)
print '- - Finished - -'
```

Python File, announcement_reply_firm.py

```
import numpy
from numpy import nan
import pandas
import glob
path = "*.csv"

#Iterates over every file in the current directory
for fname in glob.glob(path):
    table = pandas.read_csv(fname, header=None)
```

```
    #Checks if the current tweet is a Accouncement or Reply based on the current twitter data
    table[len(table.columns)] = ["Announcement" if x is nan else "Reply" for x in table[4]]
    #Writes a new csv file with the appended column
    table.to_csv(fname)
    print('Appended announcement column to', fname)
```

Python File, user_sentiment.py

```
import sys
import csv
import itertools
import re
import glob
import pandas as pd
from textblob import TextBlob
from textblob.sentiments import NaiveBayesAnalyzer

#Cleans any unwanted characters or symbols from a string input.
def clean_tweet(tweet):
    return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)", " ", tweet).split())

#Iterates over every file in the curreny directory with .csv extension
for fname in glob.glob('*.csv'):
    table = pd.read_csv(fname)
    count = 0
    #Adds new empty columns to the csv table
    table[len(table.columns)] = ""
    table[len(table.columns)] = ""
    table[len(table.columns)] = ""

    #iterates over every row in the current csv file
    for index in table.iterrows():
        #Grabs the tweet in the current row
        string = table.ix[count,1]
        if type(string) is str:
            #Runs sentiment analysis for the tweet and adds data to the newly made columns
            analysis = TextBlob(clean_tweet(string), analyzer=NaiveBayesAnalyzer())
            table.ix[count,len(table.columns)-3] = analysis.sentiment.classification
            table.ix[count,len(table.columns)-2] = analysis.sentiment.p_pos
            table.ix[count,len(table.columns)-1] = analysis.sentiment.p_neg
        count = count + 1

#Writes the new csv file
table.to_csv(fname, index=False, header=False)
```

Python File, countURLs.py

```
import glob
import re
```

```
import pandas as pd

#Iterates over the CSVs in the current directory, counts number of URLs
#in each tweet, indicates if there are > 0 tweets in one column and counts
#them in the next, by appending to the original CSV. Don't run multiple
#times on the same files, or else you'll end up with duplicate columns

def FindURL(string):
    url = re.findall('http[s]?://[ ]?(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+', string)
    return url

for fname in glob.glob('*.csv'):
    table = pd.read_csv(fname)
    count = 0
    table[len(table.columns)] = ""
    table[len(table.columns)] = ""
    for index in table.iterrows():
        string = table.ix[count,1]
    if type(string) is str:
        listOfURLs = FindURL(string)
        if len(listOfURLs) > 0:
            table.ix[count,len(table.columns)-2] = 1
        else:
            table.ix[count,len(table.columns)-2] = 0
        table.ix[count,len(table.columns)-1] = len(listOfURLs)
        count = count + 1
    table.to_csv(fname, index=False, header=False)
```

Python File, stockManipulation.py

```
#Import pandas for table manipulation
import pandas as pd
import datetime
from datetime import timedelta

#Read in the stockReturn data as stockTable
stockTable = pd.read_csv('stockReturn.csv')
#Read in the dataBreach data as dataBreaches
dataBreaches = pd.read_csv('dataBreachesActive.csv')

#Add columns to store calculated start and end dates
datesFrame = dataBreaches[['EventId', 'Ticker', 'Date Made Public', 'Name']].copy()
datesFrame['StartDate'] = ''
datesFrame['EndDate'] = ''

#Add start and end dates to every eventID
for index, row in datesFrame.iterrows():
    #Get the date the breach was made public
    tempDate = datetime.datetime.strptime(row['Date Made Public'], '%x')
    #Calculate 120 days before that date
    start = tempDate-timedelta(days=120)
```

```
    #Calculate 30 days after that date
    end = tempDate+timedelta(days=30)
    #Store these values in datesFrame
    datesFrame.set_value(index, 'StartDate', start)
    datesFrame.set_value(index, 'EndDate', end)

#Remove the row with column headers from stockTable
stockTable = stockTable[1:]
#Convert the dates in stockTable to datetime format
stockTable['formattedDate'] = pd.to_datetime(stockTable['formattedDate'])
for index, row in datesFrame.iterrows():
    print("Filtering: " + str(row['Name']))
    #Get the current company's rows from stockTable
    tempStock = stockTable[stockTable.TICKER == row.Ticker]
    #Filter the current company's rows to the dates we care about
    tempStock =
tempStock[(tempStock.formattedDate>=row['StartDate'])&(tempStock.formattedDate<=row['En
dDate'])]
    #Create an EventId column in the new table
    tempStock['EventId'] = row['EventId']
    #Rename the old stock columns
    tempStock.columns = ['PERMNO', 'oldDate', 'Ticker', 'Name', 'Price', 'Date',
'EventId']
    #Pull out only the columsn we care about
    tempStock = tempStock[['EventId', 'Date', 'Ticker', 'Name', 'Price']]
    #Convert this to a new dataFrame
    result = pd.DataFrame(tempStock)
    #Remove / values that would mess with directories
    if(type(row.Name)!=float):
        tempRowName = row.Name.replace('/', '')
    fileName = "csvs/" + str(row.EventId) + "_" + str(tempRowName) + ".csv"
    #Export to a unique csv
    result.to_csv(fileName)
```

Python File, abnormal.py

```
import pandas as pd

#Read in the data to a dataFrame
data = pd.read_csv("-3to3.csv")
#Group by company ticker and evtdate together
groups = data.groupby(["ticker", "evtdate"])
#Create the DataFrame to output to
out = pd.DataFrame(columns = ["ticker", "evtdate", "diff"])
#Use an index to append to the ouptut DataFrame
index = 0
#iterate over the values of the groups
#i contains the ticker/evtdate
#j is a dataframe of the seven days for that event
for i, j in groups:
    sumBefore = 0
    sumAfter = 0
    #Make sure the event has the right number of rows
```

```
    if j.shape == (7,4):
        #find the mean of the 3 days before
        sumBefore += float(j.iloc[0,3])
        sumBefore += float(j.iloc[1,3])
        sumBefore += float(j.iloc[2,3])
        meanBefore = sumBefore / 3
        #find the mean of the 3 days after
        sumAfter += float(j.iloc[4,3])
        sumAfter += float(j.iloc[5,3])
        sumAfter += float(j.iloc[6,3])
        meanAfter = sumAfter / 3
    #Find the difference between the means
    diff = meanAfter - meanBefore
    #Append this as a new row with the desired values
    out.loc[index] = [i[0], i[1], diff]
    index= index + 1
#Ouptut to csv
out.to_csv("abnormalDif.csv")
# Here begins our analysis from the abnormal differences
import numpy
import math
import matplotlib.pyplot
table = pd.read_csv('abnormalDif.csv')
mean = numpy.mean(table["diff"])
stdDev = numpy.std(table["diff"])
#Find all companies that have a diff value less than 2.5 standard deviations from the
mean
bottom = [x for x in table["diff"] if (x < mean - 2.5 * stdDev)]
#Find all companies that have a diff value greater than 2.5 standard deviations from
the mean
top = [x for x in table["diff"] if (x > mean + 2.5 * stdDev)]
table["Z"] = [(x-mean)/stdDev for x in table["diff"]]
bottom6 = table.sort_values("Z")[0:len(bottom)]
top7 = table.sort_values("Z",ascending=False)[0:len(top)]
#Read in all of the data for company and user tweets that has been analyzed for the
bottom 6 and top7 companies
#If any companies aren't present its because no twitter data existed for this data
breach, most likely due to
#the scarcity of tweets before 2010.
HPY_user = pd.read_csv("4_HeartlandHPY_user.csv")
HPY_user= HPY_user[:len(HPY_user)-1]
FRP_user = pd.read_csv("101_Fairpoint_user.csv")
SHLD_company = pd.read_csv("160_searsholdings.csv")
SHLD_user = pd.read_csv("160_searsholdings_keywords.csv")
TWTR_company = pd.read_csv("632_twitter.csv")
TWTR_user = pd.read_csv("632_twitter_keywords.csv")
DYN_company = pd.read_csv("647_Dyn_company.csv")
DYN_user = pd.read_csv("647_Dyn_user.csv")
DYN_user= DYN_user[:len(DYN_user)-1]
PRAN_company = pd.read_csv("665_prAna.csv")
PRAN_user = pd.read_csv("665_prAna_keywords.csv")
EFX_user= pd.read_csv("690_equifax_user.csv")
```

```
EFX_sentiment_user = pd.read_csv("690_equifax_user_sentiment_250.csv")
EFX_company = pd.read_csv("690_Equifax_company.csv")
RAD_company = pd.read_csv("699_riteaid.csv")
RAD_user = pd.read_csv("699_riteaid_keywords.csv")
#Do further analysis on all of the user tweets by summing up the values in each user tweet file
columns=["Date","Ticker","Diff","TotalFollowers","TotalFollowing","VerifiedUsers","TotalNeg","TotalPos","TotalNegPercent","TotalPosPercent","TotalLinkCount"]
UserAnalysis = pd.DataFrame(columns=columns)
row = [bottom6.iloc[2]["evtdate"],bottom6.iloc[2]["ticker"],bottom6.iloc[2]["diff"],sum([int(x.replace(",","")) for x in HPY_user["Followers"]]), sum([int(x.replace(",","")) for x in HPY_user["Following"]]), sum([int(x) for x in HPY_user["Verified"]]), sum([1 for x in HPY_user["Sent"] if x == "neg"]),sum([1 for x in HPY_user["Sent"] if x == "pos"]), sum([float(x) for x in HPY_user["neg"]])/len(HPY_user), sum([float(x) for x in HPY_user["pos"]])/len(HPY_user),int(sum([x for x in HPY_user["LinkCount"]]))]
UserAnalysis.loc[len(UserAnalysis)] = row
row = [bottom6.iloc[0]["evtdate"],bottom6.iloc[0]["ticker"],bottom6.iloc[0]["diff"],sum([int(x.replace(",","")) for x in FRP_user["Followers"]]), sum([int(x.replace(",","")) for x in FRP_user["Following"]]), sum([int(x) for x in FRP_user["Verified"]]), sum([1 for x in FRP_user["Sent"] if x == "neg"]),sum([1 for x in FRP_user["Sent"] if x == "pos"]), sum([float(x) for x in FRP_user["neg"]])/len(FRP_user), sum([float(x) for x in FRP_user["pos"]])/len(FRP_user),int(sum([x for x in FRP_user["LinkCount"]]))]
UserAnalysis.loc[len(UserAnalysis)] = row
row = [top7.iloc[0]["evtdate"],top7.iloc[0]["ticker"],top7.iloc[0]["diff"],sum([int(x) for x in SHLD_user["Followers"]]), sum([int(x) for x in SHLD_user["Following"]]), sum([int(x) for x in SHLD_user["Verified"]]), sum([1 for x in SHLD_user["Sent"] if x == "neg"]),sum([1 for x in SHLD_user["Sent"] if x == "pos"]), sum([float(x) for x in SHLD_user["neg"]])/len(SHLD_user), sum([float(x) for x in SHLD_user["pos"]])/len(SHLD_user),int(sum([x for x in SHLD_user["LinkCount"]]))]
UserAnalysis.loc[len(UserAnalysis)] = row
row = [top7.iloc[4]["evtdate"],top7.iloc[4]["ticker"],top7.iloc[4]["diff"],sum([int(x) for x in TWTR_user["Followers"]]), sum([int(x) for x in TWTR_user["Following"]]), sum([int(x) for x in TWTR_user["Verified"]]), sum([1 for x in TWTR_user["Sent"] if x == "neg"]),sum([1 for x in TWTR_user["Sent"] if x == "pos"]), sum([float(x) for x in TWTR_user["neg"]])/len(TWTR_user), sum([float(x) for x in TWTR_user["pos"]])/len(TWTR_user),int(sum([x for x in TWTR_user["LinkCount"]]))]
UserAnalysis.loc[len(UserAnalysis)] = row
row = [bottom6.iloc[5]["evtdate"],bottom6.iloc[5]["ticker"],bottom6.iloc[5]["diff"],sum([int(x.replace(",","")) for x in DYN_user["Followers"] if type(x) != float]), sum([int(x.replace(",","")) for x in DYN_user["Following"] if type(x) != float]), sum([float(x) for x in DYN_user["Verified"]]), sum([1 for x in DYN_user["Sent"] if x == "neg"]),sum([1 for x in DYN_user["Sent"] if x == "pos"]), sum([0 if math.isnan(x) else float(x) for x in DYN_user["neg"]])/len(DYN_user), sum([0 if math.isnan(x) else float(x) for x in DYN_user["pos"]])/len(DYN_user),sum([int(x) if x == 1.0 else 0 for x in DYN_user["LinkCount"]])]
UserAnalysis.loc[len(UserAnalysis)] = row
row = [top7.iloc[5]["evtdate"],top7.iloc[5]["ticker"],top7.iloc[5]["diff"],sum([int(x) for x in PRAN_user["Followers"]]), sum([int(x) for x in PRAN_user["Following"]]),
```

```
    sum([int(x) for x in PRAN_user["Verified"]]), sum([1 for x in PRAN_user["Sent"] if x ==
    "neg"]),sum([1 for x in PRAN_user["Sent"] if x == "pos"]), sum([float(x) for x in
    PRAN_user["neg"]])/len(PRAN_user), sum([float(x) for x in
    PRAN_user["pos"]])/len(PRAN_user),int(sum([x for x in PRAN_user["LinkCount"]]))]
84. UserAnalysis.loc[len(UserAnalysis)] = row
85. row =
    [bottom6.iloc[4]["evtdate"],bottom6.iloc[4]["ticker"],bottom6.iloc[4]["diff"],sum([int(
    x.replace(",","")) for x in EFX_user["Followers"] if type(x) != float]),
    sum([int(x.replace(",","")) for x in EFX_user["Following"] if type(x) != float]),
    sum([int(x) if x == 1.0 else 0 for x in EFX_user["Verified"]]), sum([1 for x in
    EFX_sentiment_user["Sent"] if x == "neg"]),sum([1 for x in EFX_sentiment_user["Sent"]
    if x == "pos"]), sum([float(x) for x in
    EFX_sentiment_user["neg"]])/len(EFX_sentiment_user), sum([float(x) for x in
    EFX_sentiment_user["pos"]])/len(EFX_sentiment_user),sum([int(x) if x == 1.0 else 0 for
    x in EFX_user["LinkCount"]])]
86. UserAnalysis.loc[len(UserAnalysis)] = row
87. row = [top7.iloc[1]["evtdate"],top7.iloc[1]["ticker"],top7.iloc[1]["diff"],0, 0,
    sum([int(x) if x == 1.0 else 0 for x in RAD_user["Verified"]]), sum([1 for x in
    RAD_user["Sent"] if x == "neg"]),sum([1 for x in RAD_user["Sent"] if x == "pos"]),
    sum([float(x) for x in RAD_user["neg"]])/len(RAD_user), sum([float(x) for x in
    RAD_user["pos"]])/len(RAD_user),sum([int(x) if x == 1.0 else 0 for x in
    RAD_user["LinkCount"]])]
88. UserAnalysis.loc[len(UserAnalysis)] = row
89. #Make a similar dataframe but just containing the four companies that have user tweet
    data in the bottom 6
90. BottomSix = pd.DataFrame(columns=columns)
91. row =
    [bottom6.iloc[2]["evtdate"],bottom6.iloc[2]["ticker"],bottom6.iloc[2]["diff"],sum([int(
    x.replace(",","")) for x in HPY_user["Followers"]]), sum([int(x.replace(",","")) for x
    in HPY_user["Following"]]), sum([int(x) for x in HPY_user["Verified"]]), sum([1 for x
    in HPY_user["Sent"] if x == "neg"]),sum([1 for x in HPY_user["Sent"] if x == "pos"]),
    sum([float(x) for x in HPY_user["neg"]])/len(HPY_user), sum([float(x) for x in
    HPY_user["pos"]])/len(HPY_user),int(sum([x for x in HPY_user["LinkCount"]]))]
92. BottomSix.loc[len(BottomSix)] = row
93. row =
    [bottom6.iloc[0]["evtdate"],bottom6.iloc[0]["ticker"],bottom6.iloc[0]["diff"],sum([int(
    x.replace(",","")) for x in FRP_user["Followers"]]), sum([int(x.replace(",","")) for x
    in FRP_user["Following"]]), sum([int(x) for x in FRP_user["Verified"]]), sum([1 for x
    in FRP_user["Sent"] if x == "neg"]),sum([1 for x in FRP_user["Sent"] if x == "pos"]),
    sum([float(x) for x in FRP_user["neg"]])/len(FRP_user), sum([float(x) for x in
    FRP_user["pos"]])/len(FRP_user),int(sum([x for x in FRP_user["LinkCount"]]))]
94. BottomSix.loc[len(BottomSix)] = row
95. row =
    [bottom6.iloc[5]["evtdate"],bottom6.iloc[5]["ticker"],bottom6.iloc[5]["diff"],sum([int(
    x.replace(",","")) for x in DYN_user["Followers"] if type(x) != float]),
    sum([int(x.replace(",","")) for x in DYN_user["Following"] if type(x) != float]),
    sum([float(x) for x in DYN_user["Verified"]]), sum([1 for x in DYN_user["Sent"] if x ==
    "neg"]),sum([1 for x in DYN_user["Sent"] if x == "pos"]), sum([0 if math.isnan(x) else
    float(x) for x in DYN_user["neg"]])/len(DYN_user), sum([0 if math.isnan(x) else
    float(x) for x in DYN_user["pos"]])/len(DYN_user),sum([int(x) if x == 1.0 else 0 for x
    in DYN_user["LinkCount"]])]
96. BottomSix.loc[len(BottomSix)] = row
```

```
97. row =
    [bottom6.iloc[4]["evtdate"],bottom6.iloc[4]["ticker"],bottom6.iloc[4]["diff"],sum([int(
    x.replace(",","")) for x in EFX_user["Followers"] if type(x) != float]),
    sum([int(x.replace(",","")) for x in EFX_user["Following"] if type(x) != float]),
    sum([int(x) if x == 1.0 else 0 for x in EFX_user["Verified"]]), sum([1 for x in
    EFX_sentiment_user["Sent"] if x == "neg"]),sum([1 for x in EFX_sentiment_user["Sent"]
    if x == "pos"]), sum([float(x) for x in
    EFX_sentiment_user["neg"]])/len(EFX_sentiment_user), sum([float(x) for x in
    EFX_sentiment_user["pos"]])/len(EFX_sentiment_user),sum([int(x) if x == 1.0 else 0 for
    x in EFX_user["LinkCount"]])]
98. BottomSix.loc[len(BottomSix)] = row
99. BottomSix
100.        #Make a similar dataframe but just containing the four companies that have user
    tweet data in the top 7
101.        TopSeven = pd.DataFrame(columns=columns)
102.        row =
    [top7.iloc[0]["evtdate"],top7.iloc[0]["ticker"],top7.iloc[0]["diff"],sum([int(x) for x
    in SHLD_user["Followers"]]), sum([int(x) for x in SHLD_user["Following"]]), sum([int(x)
    for x in SHLD_user["Verified"]]), sum([1 for x in SHLD_user["Sent"] if x ==
    "neg"]),sum([1 for x in SHLD_user["Sent"] if x == "pos"]), sum([float(x) for x in
    SHLD_user["neg"]])/len(SHLD_user), sum([float(x) for x in
    SHLD_user["pos"]])/len(SHLD_user),int(sum([x for x in SHLD_user["LinkCount"]]))]
103.        TopSeven.loc[len(TopSeven)] = row
104.        row =
    [top7.iloc[4]["evtdate"],top7.iloc[4]["ticker"],top7.iloc[4]["diff"],sum([int(x) for x
    in TWTR_user["Followers"]]), sum([int(x) for x in TWTR_user["Following"]]), sum([int(x)
    for x in TWTR_user["Verified"]]), sum([1 for x in TWTR_user["Sent"] if x ==
    "neg"]),sum([1 for x in TWTR_user["Sent"] if x == "pos"]), sum([float(x) for x in
    TWTR_user["neg"]])/len(TWTR_user), sum([float(x) for x in
    TWTR_user["pos"]])/len(TWTR_user),int(sum([x for x in TWTR_user["LinkCount"]]))]
105.        TopSeven.loc[len(TopSeven)] = row
106.        row =
    [top7.iloc[5]["evtdate"],top7.iloc[5]["ticker"],top7.iloc[5]["diff"],sum([int(x) for x
    in PRAN_user["Followers"]]), sum([int(x) for x in PRAN_user["Following"]]), sum([int(x)
    for x in PRAN_user["Verified"]]), sum([1 for x in PRAN_user["Sent"] if x ==
    "neg"]),sum([1 for x in PRAN_user["Sent"] if x == "pos"]), sum([float(x) for x in
    PRAN_user["neg"]])/len(PRAN_user), sum([float(x) for x in
    PRAN_user["pos"]])/len(PRAN_user),int(sum([x for x in PRAN_user["LinkCount"]]))]
107.        TopSeven.loc[len(TopSeven)] = row
108.        row = [top7.iloc[1]["evtdate"],top7.iloc[1]["ticker"],top7.iloc[1]["diff"],0, 0,
    sum([int(x) if x == 1.0 else 0 for x in RAD_user["Verified"]]), sum([1 for x in
    RAD_user["Sent"] if x == "neg"]),sum([1 for x in RAD_user["Sent"] if x == "pos"]),
    sum([float(x) for x in RAD_user["neg"]])/len(RAD_user), sum([float(x) for x in
    RAD_user["pos"]])/len(RAD_user),sum([int(x) if x == 1.0 else 0 for x in
    RAD_user["LinkCount"]])]
109.        TopSeven.loc[len(TopSeven)] = row
110.        TopSeven
111.        #Make a similar data frame but this time for all the company tweets
112.        columns=["Date","Ticker","Diff","TotalLinkCount","NumReplies",
    "NumAnnouncements","TotalTweets"]
113.        CompanyAnalysis = pd.DataFrame(columns=columns)
```

```
        row = [top7.iloc[0]["evtdate"],top7.iloc[0]["ticker"],top7.iloc[0]["diff"],
    sum([int(x) if x == 1.0 else 0 for x in SHLD_company["LinkCount"]]),sum([1 if x ==
    "Reply" else 0 for x in SHLD_company["Type"]]),sum([1 if x == "Announcement" else 0 for
    x in SHLD_company["Type"]]),len(SHLD_company)]
        CompanyAnalysis.loc[len(CompanyAnalysis)] = row
        row = [top7.iloc[4]["evtdate"],top7.iloc[4]["ticker"],top7.iloc[4]["diff"],
    sum([int(x) if x == 1.0 else 0 for x in TWTR_company["LinkCount"]]),sum([1 if x ==
    "Reply" else 0 for x in TWTR_company["Type"]]),sum([1 if x == "Announcement" else 0 for
    x in TWTR_company["Type"]]),len(TWTR_company)]
        CompanyAnalysis.loc[len(CompanyAnalysis)] = row
        row = [bottom6.iloc[5]["evtdate"],bottom6.iloc[5]["ticker"],
    bottom6.iloc[5]["diff"],sum([int(x) if x == 1.0 else 0 for x in
    DYN_company["LinkCount"]]),sum([1 if x == "Reply" else 0 for x in
    DYN_company["Type"]]),sum([1 if x == "Announcement" else 0 for x in
    DYN_company["Type"]]),len(DYN_company)]
        CompanyAnalysis.loc[len(CompanyAnalysis)] = row
        row = [top7.iloc[5]["evtdate"],top7.iloc[5]["ticker"],
    top7.iloc[5]["diff"],sum([int(x) if x == 1.0 else 0 for x in
    PRAN_company["LinkCount"]]),sum([1 if x == "Reply" else 0 for x in
    PRAN_company["Type"]]),sum([1 if x == "Announcement" else 0 for x in
    PRAN_company["Type"]]),len(PRAN_company)]
        CompanyAnalysis.loc[len(CompanyAnalysis)] = row
        row = [bottom6.iloc[4]["evtdate"],bottom6.iloc[4]["ticker"],
    bottom6.iloc[4]["diff"],sum([int(x) if x == 1.0 else 0 for x in
    EFX_company["LinkCount"]]),sum([1 if x == "Reply" else 0 for x in
    EFX_company["Type"]]),sum([1 if x == "Announcement" else 0 for x in
    EFX_company["Type"]]),len(EFX_company)]
        CompanyAnalysis.loc[len(CompanyAnalysis)] = row
        row = [top7.iloc[1]["evtdate"],top7.iloc[1]["ticker"],
    top7.iloc[1]["diff"],sum([int(x) if x == 1.0 else 0 for x in
    RAD_company["LinkCount"]]),sum([1 if x == "Reply" else 0 for x in
    RAD_company["Type"]]),sum([1 if x == "Announcement" else 0 for x in
    RAD_company["Type"]]),len(RAD_company)]
        CompanyAnalysis.loc[len(CompanyAnalysis)] = row
        CompanyAnalysis["RatioReplyTotal"] =
    CompanyAnalysis["NumReplies"]/CompanyAnalysis["TotalTweets"]
        #This plots the reply ratio to the difference. No strong correlation seen
        get_ipython().magic(u'matplotlib inline')
        import matplotlib.pyplot as plt
        plot = plt.scatter(x =
    CompanyAnalysis["NumReplies"]/CompanyAnalysis["TotalTweets"], y =
    CompanyAnalysis["Diff"], linewidths=2, c="g")
        plt.title("Ratio of Replies to Total Company Tweets vs Stock Difference")
        plt.xlabel("Ratio of Replies to Total Company Tweets ")
        plt.ylabel("Stock Difference")
        plot.figure.show()
        #This plot shows that the top 7 companies had a much higher mean positive
    sentiment value of user tweets
        plt.figure()
        plot = TopSeven.TotalPosPercent.plot.kde(color = "Orange")
        BottomSix.TotalPosPercent.plot.kde(color = "Blue", ax=plot)
        #The line above makes it reuse the plot
```

```
140.        plt.legend(["Top 7","Bottom 6"])
141.        plt.title("Total Positive Sentiment Percentage of all Tweets from Users")
142.        plt.xlabel("TotalPosPercent")
143.        plot.figure.show()
144.        #Smoothed out histogram
```

Appendix B: Tables

Table 1: Keywords

Keywords				
security breach	security management	security monitoring	security expenditure	information security
system security	authentication	encryption	computer virus	computer intrusion
disaster recovery	access control	cyber security	cyber attack	denial of service
hacker	hijack	infosec	breach	unauthorized access
business continuity	leakage	theft	fraud	steal

Table 2: List of Company Breaches

Ticker	EventDate	CompanyName
A	3/22/08	Agilent Technologies
AA	7/15/10	Alcoa Global Mobility Group
AACC	7/5/06	RBS National Bank, Asset Acceptance LLC
AAL	6/20/07	American Airlines
AAL	2/17/11	American Airlines
AAN	10/22/13	Aaron's
AAN	11/2/11	Aaron's
AAP	3/31/08	Advance Auto Parts
AAP	3/16/16	Advanced Auto Parts
AAPL	6/9/10	Apple Inc., AT&T
AAPL	9/1/14	Apple
AAPL	9/4/12	Apple
AAPL	2/26/14	Apple
AAPL	4/1/11	iTunes (Apple)
AAPL	2/19/13	Apple
AAPL	7/22/13	Apple Inc.

AAPL	2/16/16	Apple
AAR	7/4/10	AMR Corporation
AAR	7/2/10	AMR Corporation
ABB	9/11/17	ABB Inc.
ABM	4/21/11	ABM Industries
ABM	11/14/17	ABM Industries
ABS	8/15/14	Albertsons/AB Acquisitions LLC
ABS	4/21/07	Albertsons (Save Mart Supermarkets)
ADBE	10/4/13	Adobe, PR Newswire, National White Collar Crime Center
ADBE	5/13/13	Adobe, Washington Administrative Office of the Courts
ADBE	11/14/12	Adobe
ADP	7/6/06	Automatic Data Processing (ADP)
ADP	7/30/13	US Airways, McKesson, City of Houston, Automatic Data Processing (ADP), AlliedBarton Security Services
ADP	7/30/13	US Airways, Advanced Data Processing
ADP	12/28/11	Automatic Data Processing (ADP), A.W. Hastings'
ADP	6/17/06	Automatic Data Processing (ADP)
ADP	6/15/11	
ADP	5/5/16	ADP, LLC.
ADVS	1/10/07	Advent Software Inc.
AET	5/28/10	Aetna
AET	12/12/06	Aetna, Nationwide, WellPoint Group Health Plans, Humana Medicare, Mutual of Omaha Insurance Company, Anthem Blue Cross Blue Shield via Concentra Preferred Systems
AET	5/28/09	Aetna
AET	11/14/10	Aetna of Connecticut
AET	8/24/17	Aetna
AFBA	10/1/07	PFPC Inc., AFBA
AFL	8/22/06	AFLAC American Family Life Assurance Co.
AFL	4/19/06	Aflac
AFL	3/16/17	Aflac
AIG	6/14/06	American International Group (AIG), Indiana Office of Medical

		Excess, LLC
ALK	7/26/17	Virgin America
ALL	8/23/11	Allstate Financial
ALL	6/29/06	AllState Insurance Huntsville branch
ALSK	2/20/14	Alaska Communications
ALU	5/18/07	Alcatel-Lucent
AMCC	4/4/11	Applied Micro Circuits Corporation
AMD	1/13/13	Advanced Micro Devices (AMD), Nvidia
AMD	4/9/12	Intel, Advanced Micro Devices (AMD)
AMP	12/25/05	Ameriprise Financial Inc.
AMQ	1/30/10	Ameriquest Mortgage Company
AMTD	9/14/07	TD Ameritrade Holding Corp.
AMTD	12/1/06	TD Ameritrade
AMTD	4/20/05	TD Ameritrade
AMZN	1/29/11	Amazon.com
AMZN	9/29/17	Whole Foods
AN	5/26/14	AutoNation Toyota of South Austin
ANTM	12/12/06	Aetna, Nationwide, WellPoint Group Health Plans, Humana Medicare, Mutual of Omaha Insurance Company, Anthem Blue Cross Blue Shield via Concentra Preferred Systems
ANTM	12/12/06	Aetna, Nationwide, WellPoint Group Health Plans, Humana Medicare, Mutual of Omaha Insurance Company, Anthem Blue Cross Blue Shield via Concentra Preferred Systems
ANTM	2/5/15	Anthem
ANTM	5/13/11	Anthem Blue Cross
ANTM	11/10/14	Anthem Blue Cross
ANTM	7/31/17	Anthem
ARMK	6/6/06	ARAMARK Corporation
ARW	3/8/10	Arrow Electronics
ARW	3/8/10	Arrow Electronics
ARW	3/8/10	Arrow Electronics
ARW	3/8/10	Arrow Electronics
AV	6/3/09	Aviva
AWI	7/25/06	Armstrong World Industries,

		Deloitte & Touche
AXP	7/13/12	American Express Travel Related Services Company, Inc. (AXP)
AXP	12/29/13	American Express Company
AXP	8/14/09	American Express
AXP	3/25/14	American Express
AXP	4/7/14	American Express Company
AXP	4/1/13	Tennis Express, American Express
AXP	3/29/13	American Express
BA	7/11/14	Boeing
BA	12/13/06	Boeing
BA	4/21/06	Boeing
BA	11/15/06	Boeing, Co
BA	11/19/05	Boeing
BA	2/8/17	The Boeing Corporation
BA	2/27/17	Boeing
BAC	8/11/09	Bank of America Corp.
BAC	6/8/10	Bank of America
BAC	5/25/11	Bank of America
BAC	12/14/06	Bank of America
BAC	8/18/11	Citigroup, Inc., Bank of America, Corp.
BAC	2/13/11	Bank of America
BAC	2/25/05	Bank of America Corp.
BAC	7/17/14	Bank of America
BAC	9/23/05	Bank of America
BAC	4/12/07	Bank of America
BAC	4/7/10	Bank of America
BAC	4/28/05	Wachovia, Bank of America, PNC Financial Services Group and Commerce Bancorp
BAC	6/29/05	Bank of America
BBBY	9/25/15	Bed Bath and Beyond
BBBY	6/19/17	Bed Bath & Beyond
BBT	5/15/08	BB&T Insurance
BBY	5/6/11	Best Buy
BC	4/21/08	Brunswick Corp.
BC	2/16/07	Brunswick Corp.
BDL	5/20/11	Flanigan's
BEN	8/3/06	Franklin Templeton Investments

BGC	11/19/07	General Cable Corporation
BGS	12/6/13	B&G Foods North America, Inc., Maple Grove Farms
BHE	11/21/17	Uber
BK	3/26/08	Bank of New York Mellon
BKE	6/20/17	The Buckle Inc.
BKS	10/24/12	Barnes & Noble
BKW	2/25/12	Burger King
BLKB	6/17/09	Blackbaud Inc.
BMY	7/17/08	Bristol-Myers Squibb
BOH	3/1/13	Bank of Hawaii, First Hawaiian Bank
BPF	11/27/17	Bulletproof
BR	6/22/09	Broadridge Financial Solutions, Inc.
BRLI	8/25/14	BioReference Laboratories, Inc./CareEvolve, Inc.
BSFT	9/5/17	BroadSoft
BSX	2/8/14	Boston Scientific
BUD	7/29/08	Anheuser-Busch
C	6/9/11	Citibank
C	9/21/07	Citigroup, ABN Amro Mortgage Group
C	8/11/09	Citigroup Inc.
C	6/19/08	Citibank
C	10/14/10	Citibank
C	3/28/13	Citi
C	10/2/06	Citigroup
C	8/18/11	Citigroup, Inc., Bank of America, Corp.
C	2/24/10	Citigroup
C	7/17/13	Citigroup
C	8/9/07	Citigroup
C	7/27/10	Citigroup Inc.
C	6/6/05	Citigroup, UPS
CAKE	9/29/10	Cheesecake Factory, PGA Tour Grill, Outback Steakhouse
CAKE	9/11/10	Cheesecake Factory
CAKE	5/24/10	Cheesecake Factory
CAT	4/27/07	Caterpillar, Inc., SBA Inc.
CCI	11/25/13	Crown Castle International Corp
CELG	8/20/07	Celgene Corporation

CFR	5/19/06	Frost Bank
CHDN	9/4/12	Twinspires.com (Churchill Downs Technology Initiatives Company)
CHH	4/26/12	Choice Hotels Internationals
CHH	3/22/13	Comfort Inn and Suites
CHSCP	12/31/10	CHS, Inc.
CHSI	4/17/12	Catalyst Health Solutions, Alliant Health Plans, Inc.
CHTR	8/13/08	Charter Communications
CI	11/7/06	CIGNA HealthCare Corp
CI	12/7/06	CIGNA HealthCare Corp
CLGX	8/31/06	CoreLogic for ComUnity Lending
CMCSA	3/16/09	Comcast
CMCSA	10/3/13	Comcast Phone
CMCSA	5/20/12	Comcast
CME	11/17/13	CME Group, CME ClearPort
CMG	4/26/17	Chipotle Mexican Grill
CNC	1/26/16	Centene
CNET	7/14/14	CNET
CNQR	12/16/10	Concur Technologies Inc.
COF	3/4/14	Capital One
COF	2/12/13	J.P. Morgan Chase, Capital One
COF	5/18/10	Capitol One
COF	9/17/05	North Fork Bank (now Capital One Bank)
COF	5/9/12	Capital One Bank
COF	2/6/17	Capital One
COF	7/6/17	Spark Pay
COLB	5/21/07	Columbia Bank
CPRT	8/28/06	Copart, Inc.
CPS	10/20/09	ChoicePoint
CS	2/20/07	Credit Suisse
CSC	4/3/13	Computer Sciences Corporation
CSCO	7/10/10	Cisco Live 2010
CSCO	4/9/12	Ernst & Young LLP, Cisco Systems, Inc.
CSCO	10/25/16	Cisco
CVC	7/25/06	Cablevision Systems Corp., ACS, FedEx
CVS	2/18/09	CVS Pharmacies
CVS	7/30/14	CVS/Caremark

CVS	6/21/05	CVS
CVS	7/18/15	CVS Pharmacy, Imperial Beach
CVS	4/15/07	CVS Pharmacy
CVS	11/28/13	CVS Pharmacy, Inc., Maryland CVS Pharmacy, LLC
CVS	3/24/12	CVS Caremark
CVS	12/4/12	CVS Caremark
CVS	12/5/16	CVS Health
CVX	8/16/06	Chevron
CVX	3/9/11	Shell, Chevron
CYH	8/18/14	Community Health Systems
CYN	7/6/05	City National Bank, Iron Mountain
D	8/25/06	Dominion Resources
DBD	8/31/06	Diebold, Inc., GE Capital
DBMG	2/2/17	DBM Global
DENN	9/30/13	Denny's
DFS	2/21/14	Discover Financial Services
DFS	8/17/12	Discover Financial Services
DFS	11/11/13	Discover Financial Services
DFS	12/20/13	Discover Financial Services
DFS	9/9/06	Discover Bank
DGX	9/16/12	Quest Diagnostics
DHI	2/16/12	D.R. Horton Inc. (DHI Mortgage)
DIS	7/30/16	Disney Consumer Products and Interactive Media
DLTR	8/1/06	Dollar Tree
DNB	9/26/13	LexisNexis, Dun & Bradstreet, Kroll Background America
DNB	10/28/13	Dun & Bradstreet
DPZ	5/12/11	Domino's Pizza, KB Pizza
DPZ	6/18/08	Domino's Pizza
DRI	11/15/17	Cheddar's Scratch Kitchen
DRIV	6/4/10	Digital River Inc.
DRIV	12/22/10	Digital River Inc., SWReg Inc.
DSW	3/8/05	DSW Shoe Warehouse, Retail Ventures
DTV	10/11/06	DirecTV, Deloitte and Touche LLC
DTV	5/26/12	Direct TV
DVA	11/7/13	DaVita
DVA	3/3/08	DaVita Inc.

DXC	7/5/17	DXC Technology
DYN	10/21/16	Dyn
EBAY	5/21/14	Ebay
EFX	10/10/12	Equifax
EFX	2/11/10	Equifax
EFX	6/20/06	Equifax
EFX	5/6/16	Equifax Inc.
EFX	9/7/17	Equifax Corporation
EHTH	1/27/17	eHealth Insurance
EL	7/26/11	Este□© Lauder
EMR	5/4/12	Emerson (Funai Corporation)
ESBF	4/23/10	ESB Financial
ESRX	11/6/08	Express Scripts
ESRX	2/18/13	Express Scripts, Ernst & Young
ETFC	10/9/15	E-Trade
EV	2/8/12	Eaton Vance Management
EXEL	8/16/13	Exelixis
EXPE	11/15/06	Expedia Corporate Travel (now Egencia)
EZPW	5/8/07	EZCORP, EZPAWN
F	12/22/05	Ford Motor Co.
F	5/5/12	Ford-Motor Websites (Connect With Fiesta, Unleashfiesta)
FB	6/21/13	Facebook
FB	7/28/08	Facebook
FB	2/15/13	Facebook
FB	2/4/11	Twitter, Facebook and PayPal
FB	8/30/17	Instagram
FDX	2/4/06	FedEx
FDX	7/25/06	Cablevision Systems Corp., ACS, FedEx
FINL	3/26/13	The Finish Line, Inc.
FIRE	11/27/12	Sourcefire
FIS	7/3/07	Fidelity National Information Services/Certegy Check Services Inc.
FIS	8/26/11	Fidelity National Information Services, Inc. (FIS)
FIS	9/24/07	Fidelity National Information Services, Fidelity National Financial
FITB	4/13/06	Fifth Third Bank

FLWS	3/8/16	1-800-Flowers
FMS	2/8/07	Fresenius Medical Care Holdings Inc., Fresenius Medical Care North America (FMCNA)
FORR	12/5/07	Forrester Research
FOXA	4/16/09	Fox Entertainment Group
FRBA	10/16/06	VISA, FirstBank (1st Bank)
FRC	8/14/12	First Republic Bank
FRED	6/12/15	Fred's Inc.
FRP	4/20/09	FairPoint Communications Inc.
FSB	9/10/08	Franklin Savings and Loan
GCI	5/4/17	Gannett Co
GE	9/25/06	General Electric (GE)
GE	5/16/06	GE Money Bank, Lowe's Companies Inc.
GE	2/9/07	General Electric
GM	8/3/12	General Motors Co.
GM	3/14/06	General Motors (GM)
GM	4/16/10	General Motors
GME	6/2/17	Game Stop
GNCMA	5/24/12	General Communication Inc. (GCI)
GOOG	3/7/09	Google
GOOG	5/6/16	Google Inc.
GOOGL	5/4/17	Google Docs
GPI	7/19/06	Group 1 Automotive Inc, Weinstein Spira & Company, P.C.
GPN	3/30/12	Global Payments Inc.
GPS	9/28/07	Gap Inc.
GPS	7/16/13	Gap, Banana Republic
GPS	4/16/10	Gap Inc.
GRPN	7/2/12	Groupon
GS	7/2/14	Goldman Sachs
GS	5/18/13	Goldman Sachs, Bloomberg LP
GUID	12/20/05	Guidance Software, Inc.
GYMB	10/27/06	Gymboree
H	1/15/16	Hyatt Hotels
H	11/16/17	Hyatt Hotels
HBAN	10/27/09	FirstMerit Bank
HBAN	5/9/11	Huntington National Bank
HCSG	12/9/11	Health Care Service Corporation (HCSC)

HD	9/2/14	The Home Depot
HD	2/6/14	The Home Depot
HD	10/17/07	Home Depot
HD	12/14/10	Home Depot
HD	4/13/12	The Home Depot
HD	4/30/07	Home Depot
HD	5/24/07	Home Depot
HIG	4/6/11	Hartford Life Insurance Company
HIG	9/12/07	Hartford Life Insurance Company
HIG	10/30/07	Hartford Financial Services Group
HLT	9/25/15	Hilton Hotels
HMN	10/29/07	The Horace Mann Companies
HMN	11/12/07	The Horace Mann Companies
HNT	11/18/09	Health Net
HNT	7/2/13	Health Net, CalViva Health
HNT	4/16/10	Health Net
HOG	4/4/08	Harley-Davidson, Inc. (HOG)
HON	1/31/06	Honeywell International
HON	4/19/07	Honeywell International
HPE	8/17/07	Mercury Interactive, Hewlett-Packard
HPE	11/23/16	Hewlett Packard Enterprise Services
HPQ	12/11/08	Hewlett-Packard, Symantec
HPY	1/20/09	Heartland Payment Systems
HRB	3/23/10	H&R Block
HRB	3/23/12	H&R Block
HRB	12/22/05	H&R Block
HRB	4/8/10	H&R Block
HS	5/22/08	HealthSpring Inc.
HSBC	4/15/05	Polo Ralph Lauren, HSBC
HSBC	4/10/15	HSBC Finance Corporation
HSBC	8/9/10	HSBC Bank Nevada
HSBC	1/13/16	HSBC SBN
HSIC	3/16/07	Henry Schein, Financial Services, Inc., ChoiceHealth Leasing
HTZ	11/11/06	Hertz Global Holdings, Inc.
HUM	12/12/06	Aetna, Nationwide, WellPoint Group Health Plans, Humana Medicare, Mutual of Omaha Insurance Company, Anthem Blue Cross Blue Shield via

		Concentra Preferred Systems
HUM	5/23/14	Humana
HUM	6/3/06	Humana
HUM	10/9/15	Humana
HUM	8/18/10	Humana Inc, Matrix Imaging
IBM	5/15/07	IBM
IBM	3/15/06	Ernst & Young, IBM
IHG	9/3/13	InterContinental Mark Hopkins San Francisco
IHG	7/26/16	Kimpton Hotels
IHG	2/3/17	InterContinental Hotels Group (IHG)
IHS	2/27/13	Information Handling Services, Inc. (IHS)
ING	6/18/06	ING U.S. Financial Services, Jackson Health System
ING	10/12/10	ING
ING	6/18/06	ING U.S. Financial Services
INOD	1/13/09	Innodata Isogen, Inc.
INTC	2/10/12	Intel, Inc.
INTU	4/2/15	Intuit
INTU	5/11/17	Intuit
IR	11/6/06	Ingersoll Rand
IRM	1/17/08	GE Money , Iron Mountain
IRM	5/2/05	Time Warner, Iron Mountain Inc.
IRM	7/6/05	City National Bank, Iron Mountain
ITT	1/6/11	Marsh U.S. Consumer, Seabury and Smith, ITT Corporation
JACK	2/22/11	Jack in the Box
JIVE	9/23/16	Jive Software/Producteev
JLL	8/9/10	Jones Lang LaSalle
JPM	8/28/14	J.P Morgan Chase
JPM	12/5/13	JPMorgan Chase
JPM	7/30/11	Chase Bank
JPM	10/1/13	JP Morgan Chase
JPM	1/26/07	Chase Bank and the former Bank One, now merged
JPM	1/30/11	JP Morgan Chase, Citibank
JPM	2/12/13	J.P. Morgan Chase, Capital One
JPM	5/1/07	JP Morgan
JPM	5/1/07	JP Morgan
JPM	9/14/10	JP Morgan Chase Bank

JPM	1/20/11	Chase Bank
JPM	9/7/06	Circuit City and Chase Card Services, a division of JP Morgan Chase & Co.
JPM	6/12/10	JP Morgan Chase
JPM	8/30/05	JP Morgan Chase & Co.
JPM	3/28/13	JPMorgan Chase
JPM	1/19/10	CHASE
JWN	10/10/13	Nordstrom
KBH	1/18/07	KB Home
KBR	1/26/11	KBR, Inc.
KELYA	3/9/12	Kelly Services
KEY	5/9/12	Key Bank
KEY	11/18/06	KeyCorp
KEY	12/30/06	KeyCorp
KFY	10/12/12	Korn/Ferry International
KMB	11/2/17	Kimberly-Clark
KND	8/16/12	Kindred Healthcare Inc. (Kindred Transitional Care and Rehabilitation)
KO	1/24/14	Coca-Cola Company
KO	2/22/12	Coca-Cola Company Family Federal Credit Union
KRFT	3/3/08	Kaft Foods
KRFT	9/5/07	Affiliated Computer Services (ACS), Kraft Foods
LABL	6/16/16	Multi-Color Corporation
LCC	4/6/11	US Airways
LH	3/27/10	Laboratory Corporation of America LabCorp
LH	6/9/13	Laboratory Corporation of America (LabCorp)
LJPC	12/31/14	La Jolla Group
LLL	5/15/12	L-3 Communications Corporation
LMT	7/11/14	Lockheed Martin
LMT	5/27/11	Lockheed Martin
LNC	1/14/10	Lincoln National Corporation (Lincoln Financial)
LNC	9/16/12	Lincoln Financial Securities Corporation, Red Boat Advisor Resources
LNC	7/21/10	Lincoln National Life Insurance
LNC	7/26/11	Lincoln National Life Insurance

		Company, Lincoln Life & Annuity Company of New York
LNC	8/23/11	Lincoln Financial Group, Lincoln National Life Insurance Company, Lincoln Life and Annuity Company of New York
LNC	5/25/10	Lincoln Financial Group
LNKD	6/6/12	LinkedIn.com
LOW	5/19/14	Lowe's
LOW	5/22/14	Lowes Corporation
LOW	5/16/06	GE Money Bank, Lowe's Companies Inc.
LPLA	7/8/08	LPL Financial (formerly Linsco Private Ledger)
LPLA	10/12/07	LPL Financial
LPLA	3/9/10	LPL Financial
LPLA	8/11/10	LPL Financial
LRCX	4/14/10	Lam Research Corp.
LUX	11/26/08	Luxottica Group, Things Remembered
LVS	2/12/14	Las Vegas Sands Hotels and Casinos
LXK	2/15/08	Lexmark International
M	4/23/13	Macy's
MAR	12/28/05	Marriott International Inc.
MBI	10/7/14	Municipal Bond Insurance Association (MBIA)
MCD	8/9/11	McDonald's
MCD	8/22/08	Liberty McDonald's Restaurant
MCD	3/9/12	McDonald's
MCD	11/18/11	McDonald's
MCD	11/18/11	McDonald's
MCD	9/12/11	McDonald's
MCD	11/5/11	McDonald's
MCD	12/14/10	McDonald's, Arc Worldwide, Silverpop Systems Inc.
MCD	11/16/11	McDonald's
MCK	7/30/13	US Airways, McKesson, City of Houston, Automatic Data Processing (ADP), AlliedBarton Security Services
MCK	9/9/07	McKesson Specialty, AstraZeneca
MDB	9/5/17	MongoDB

MDT	2/8/14	Medtronic
MDT	8/2/13	Medtronic
MEET	8/18/14	MeetMe, Inc.
MET	1/24/12	Metropolitan Life Insurance Company (MetLife) of Connecticut
MET	1/25/11	MetLife
MET	8/10/10	Metropolitan Life Insurance Company (MetLife)
MGI	1/12/07	MoneyGram International
MHS	3/1/06	Medco Health Solutions
MHS	3/22/12	Medco Health Solutions, Inc.
MIK	5/11/11	Michaels Stores Inc.
MOH	5/6/14	Molina Healthcare
MOH	2/6/12	Molina Healthcare of California
MS	1/5/15	Morgan Stanley
MSFT	4/3/15	Microsoft/Xbox One
MSFT	12/26/14	Microsoft xBox
MSFT	2/22/13	Microsoft
MSG	11/22/16	The Madison Square Garden Company
MSI	5/30/05	Motorola
MTB	5/17/06	M &T Bank via contractor PFPC
MTR	2/14/12	American Stock Transfer & Trust Company, LLC, Mesa Royalty Trust
MUSA	6/9/11	Murphy USA
MUSA	9/20/13	Murphy USA
MUSA	11/6/10	Murphy USA
MWV	11/1/07	MeadWestvaco
MWW	8/23/07	Monster.com
MWW	1/23/09	Monster.com
NDAQ	7/26/13	NASDAQ OMX Group Inc.
NDAQ	7/18/13	NASDAQ.com
NDLS	5/16/16	Noodles and Company
NFLX	1/1/10	Netflix
NFLX	5/4/11	Netflix
NFP	10/30/06	National Financial Partners (NFP)
NFP	10/8/07	National Financial Partners (NFP)
NGVC	3/2/15	Natural Grocers
NLSN	2/10/14	Nielsen
NNI	7/18/06	Nelnet Inc., UPS

NOC	8/9/13	Northrop Grunman
NOC	4/19/17	Northrop Grumman Systems Corporation
NOVC	9/26/16	Novation Settlement Solutions
NSM	8/12/15	Nationstar Mortgage LLC
NTRS	7/29/14	Northern Trust Company
NTY	7/15/10	NBTY
NUAN	3/13/10	Nuance Communications Inc.
NVDA	1/13/13	Advanced Micro Devices (AMD), Nvidia
NVDA	7/13/12	Nvidia
NVDA	1/6/15	NVIDIA Corporation
NYT	1/30/13	The New York Times
NYT	8/27/13	The New York Times, Melbourne IT
OMX	2/9/06	OfficeMax
ORCL	11/11/07	Oracle Corporation, Lodestar
ORCL	8/8/16	Oracle's MICROS Point-of-Sale
OUTR	4/7/08	Redbox
OXY	1/14/09	Occidental Petroleum Corporation
PACB	9/25/14	Pacific BioSciences of California Inc.
PAET	11/17/06	Paetec Communications
PAY	3/7/17	Verifone
PBG	1/2/09	Pepsi Bottling Group
PBI	3/19/07	Pitney Bowes
PF	11/27/12	Pinnacle Foods Group, LLC
PFE	5/12/08	Pfizer
PFE	9/4/07	Pfizer
PFE	10/10/07	Wheels Inc., Pfizer
PFE	4/7/08	Pfizer Inc
PFE	8/13/07	Pfizer, Axia Ltd.
PFE	6/11/07	Pfizer
PFE	9/28/07	Pfizer
PFG	5/14/10	Principal Financial Group
PFMT	8/14/17	Performant Financial Corporation
PGR	4/6/06	Progressive Casualty Insurance
PHH	5/10/13	PHH Corporation
PJC	2/8/07	Piper Jaffrey
PKI	3/16/16	PerkinElmer, Inc.
PLAY	5/12/08	Dave & Buster's

PLNT	10/17/08	The Planet
PNC	3/19/10	PNC Financial Services Group Inc.
PNC	4/28/05	Wachovia, Bank of America, PNC Financial Services Group and Commerce Bancorp
PNX	12/4/10	Phoenix
PRA	8/11/10	ProAssurance Mid-Continent Underwriters
PRAN	3/8/17	prAna
PRU	2/6/06	Prudential Financial Inc.
PRU	3/4/13	The Prudential Insurance Company of America, Unisys
PRU	11/30/07	Prudential Financial
PSA	1/29/07	Public Storage Inc.
PSS	6/11/10	Payless Shoe Store
PULB	7/16/12	Pulaski Bank, Pulaski Financial
PWRD	4/25/12	Cryptic Studios, Perfect World
PYPL	2/4/11	Twitter, Facebook and PayPal
PZZA	11/7/05	Papa John's
QABA	12/1/05	First Trust Bank
QTM	6/17/10	Quantum Corporation
RAD	7/30/14	Rite Aid Pharmacy
RAD	9/27/12	Rite Aid Corporation
RAD	7/27/10	Rite Aid Corporation
RAD	1/12/12	RIte Aid Corporation
RAD	5/19/17	Rite Aid
RAX	5/2/12	Rackspace, Incorporating Services, Ltd.
RCII	4/25/12	Rent-A-Center, Inc.
RF	1/31/12	Regions Financial Corp., Ernst & Young
RL	4/15/05	Polo Ralph Lauren, HSBC
RL	4/28/12	Taco Bell, McDonald's, Wrigley Field, Ralph Lauren Restaurant (RL Restaurant)
ROL	3/27/13	Rollins, Inc.
ROST	8/5/10	Ross
RRD	1/28/13	RR Donnelley, UnitedHealthcare, Boy Scouts of America
RUN	2/2/17	Sunrun
S	3/11/09	Sprint
S	1/22/07	Sprint Nextel

S	12/19/06	Velocita Wireless, Sprint Nextel
S	9/2/10	Sprint
S	8/16/17	Virgin Mobile
SABR	8/7/15	Sabre Corporation
SABR	5/2/17	Sabre Corporation
SABR	5/17/17	Sabre Corporation
SAIC	3/19/07	Science Applications International Corp. (SAIC)
SAIC	7/20/07	Science Applications International Corp. (SAIC)
SAIC	2/12/05	Science Applications International Corp. (SAIC)
SAIC	1/18/08	SAIC
SBCF	3/3/11	Racetrac, Seacoast National Bank
SBH	3/5/14	Sally Beauty Supply
SBH	5/4/15	Sally Beauty Supply
SBUX	5/12/15	Starbucks
SBUX	11/3/06	Starbucks Corp.
SBUX	11/24/08	Starbucks Corp.
SCHW	4/9/10	Charles Schwab
SCHW	5/3/16	Charles Schwab
SCNB	1/12/10	Suffolk County National Bank
SCOR	6/12/13	comScore
SEAC	9/8/10	SeaChange International
SEMG	2/10/09	SemGroup LP
SFLY	11/26/14	Shutterfly/Tiny Prints/Treats/Wedding Divas
SFM	2/25/13	Sprouts
SFM	3/28/16	Sprouts Farmers Market
SHLD	10/10/14	Sears Holding Company/K-Mart
SHLD	2/28/14	Sears
SHLD	5/23/12	Sears Portrait Studio
SHLD	4/28/06	Sears, Roebuck, Company Contractor Compliance
SHLD	10/12/06	Sears Holding Corporation
SHLD	1/7/08	Sears, ManageMyHome.com
SMMF	6/22/15	Summit Financial Group
SMTC	10/8/07	Semtech
SNAP	3/4/16	Snapchat
SNE	4/27/11	Sony, PlayStation Network (PSN), Sony Online Entertainment (SOE)

SNE	11/24/14	Sony Pictures
SNE	6/6/11	Sony Pictures, Sony Corporation of America
SNE	12/26/14	Sony PlayStation
SNI	10/16/15	Scripps Network LLC. (Food.com)
SONC	9/26/17	Sonic Drive-In
SPLS	10/20/14	Staples Inc.
SPLS	2/2/12	Staples (Staples Business Depot)
SRCE	6/10/08	1st Source Bank
SRCE	11/19/10	1st Source Bank
STFGX	6/7/16	State Farm Mutual Automobile Insurance Company
STI	5/16/11	SunTrust Bank
STI	2/22/10	SunTrust Bank
STT	5/29/08	State Street Corp, Investors Financial Services
STX	3/6/16	Seagate
SVEV	3/3/10	7-Eleven
SVEV	2/24/10	7-Eleven
SVU	8/15/14	Supervalue
SWK	3/11/13	Stanley Black & Decker, Inc.
SWY	11/5/05	Safeway, Hawaii
SYMC	3/31/09	Symantec
SYMC	12/11/08	Hewlett-Packard, Symantec
SYMC	11/4/12	Symantec, ImageShack
SYNH	7/21/16	inVentiv Health, Inc.
T	6/9/10	Apple Inc., AT&T
T	8/29/06	AT&T via vendor that operates an order processing computer
T	8/30/07	AT&T
T	6/10/14	AT&T Mobility, LLC
T	10/6/14	AT&T
T	4/8/15	AT&T
T	5/25/10	AT&T/Ferrell Communication
T	5/22/08	AT&T
T	7/8/09	AT&T
T	11/21/11	AT&T
T	6/16/10	AT&T
T	2/27/10	AT&T
TAX	2/13/15	Liberty Tax Services
TAX	12/13/10	Liberty Tax Service

TD	3/13/10	TD Bank
TD	10/8/12	TD Bank
TD	3/10/11	TD Bank
TD	3/4/13	TD Bank, N.A.
TGT	12/13/13	Target Corp.
Ticker	EventDate	Company
Ticker	Date Made Public	Name
TIME	12/31/09	Time Inc., Harvard Business Review
TJG	5/29/13	TJG, Inc., Target Marketing
TJX	1/17/07	TJ stores (TJX), including TJMaxx, Marshalls, Winners, HomeSense, AJWright, KMaxx, and possibly Bob's Stores in U.S. & Puerto Rico -- Winners and HomeGoods stores in Canada -- and possibly TKMaxx stores in UK and Ireland
TM	8/4/06	Toyota
TM	8/26/16	Toyota Motor Corporation
TMUS	6/7/09	T-Mobile USA
TMUS	10/14/06	T-Mobile USA Inc.
TMUS	1/16/12	T-Mobile
TMUS	10/8/15	T-Mobile USA Inc.
TMUS	12/7/16	T-Mobile
TMUS	10/12/17	T-Mobile
TOO	11/9/17	Tween Brands, Inc.
TREE	4/22/08	LendingTree
TRI	8/11/10	Thomson Reuters
TRIP	3/24/11	TripAdvisor
TRMK	6/22/15	Trustmark Mutual Holding Company
TRU	11/30/06	TransUnion Credit Bureau, Kingman, AZ, court office
TRU	1/29/08	TransUnion, Intelenet Global Services,
TRU	3/12/12	TransUnion LLC, Manufacturers Life Insurance Company (ManuLife)
TTEC	6/21/10	TeleTech, Sony Electronics
TWC	7/28/10	Time Warner Cable
TWC	1/8/16	Time Warner Cable
TWTR	2/2/13	Twitter
TWTR	2/4/11	Twitter, Facebook and PayPal

TWTR	6/13/16	Twitter
TWTR	5/19/17	Vine
TWX	5/2/05	Time Warner, Iron Mountain Inc.
TWX	7/31/17	HBO
TWX	10/30/17	Home Box Office (HBO)
TXT	7/31/07	Textron
TYL	3/13/17	Tyler Technologies Inc.
UA	4/20/12	Under Armour Inc., PricewaterhouseCoopers
UAL	7/29/15	United Airlines
UAL	1/1/15	United Airlines
UAL	1/13/09	Continental Airlines
UBNT	8/7/15	Ubiquiti Networks Inc.
UBS	11/7/07	UBS FInancial Services
UNB	4/5/12	Union Bank
UNH	10/12/11	United Healthcare Inc., Futurity First Insurance Group
UNH	5/25/11	United Healthcare Inc.
UNH	5/18/12	UnitedHealthcare (United Health Group Plan)
UNH	1/28/13	RR Donnelley, UnitedHealthcare, Boy Scouts of America
UNH	8/6/10	United HealthGroup
UNH	8/6/10	United HealthGroup
UNH	10/11/10	UnitedHealth Group
UNH	8/6/10	United HealthGroup
UNH	8/6/10	United HealthGroup
UNP	6/16/06	Union Pacific
UPS	8/20/14	The UPS Store
UPS	7/18/06	Nelnet Inc., UPS
UPS	4/6/07	Hortica (Florists___ Mutual Insurance Company), UPS
UPS	6/6/05	Citigroup, UPS
USB	9/28/10	US Bank
USB	8/1/06	US Bank
USB	3/1/10	US Bank
V	10/16/06	VISA, FirstBank (1st Bank)
VIAB	9/20/17	Viacom
VLY	2/14/12	Valley National Bank, American Stock Transfer and Trust Company, LLC
VLY	5/27/11	Valley National Bank

VMED	10/25/07	Virgin Mobile
VRA	10/12/16	Vera Bradley
VRSN	2/2/12	VeriSign Inc.
VRSN	8/6/07	Verisign
VSTO	9/19/16	Active Outdoors
VZ	8/12/05	Verizon
VZ	8/25/06	Verizon Wireless
VZ	3/8/06	Verizon Communications
WASH	8/28/08	The Washington Trust Co.
WCC	11/3/06	Wesco
WCG	4/8/08	WellCare Health Plans Inc.
WCG	12/6/14	WellCare Health Plans
WEB	8/19/15	Web.com
WEN	7/28/10	Wendy's
WEN	1/27/16	Wendy's
WFC	9/1/06	Wells Fargo via unnamed auditor
WFC	8/12/08	Wells Fargo
WFC	8/29/06	Wells Fargo, Paymap Inc., First Horizon Home Loans, Western Union
WFC	5/5/06	Wells Fargo
WFC	10/20/11	Wells Fargo
WFC	5/25/10	Wells Fargo
WFC	7/31/17	Wells Fargo
WIN	1/27/12	Windstream
WINN	6/23/07	Winn-Dixie
WKL	7/24/06	Wolters Kluwer
WLP	2/10/10	WellPoint, Anthem/Blue Cross and Blue Shield
WLP	8/6/10	WellPoint, Inc.
WM	4/3/07	Waste Management Inc.
WM	4/3/07	Waste Management Inc.
WMB	8/1/09	Williams Cos. Inc.
WMT	9/28/07	Wal-Mart Stores Inc.
WMT	6/7/10	Wal-Mart, Sam's Club
WSBN	3/15/17	Wishbone
WSM	8/17/06	Williams-Sonoma, Deloitte & Touche
WU	8/29/06	Wells Fargo, Paymap Inc., First Horizon Home Loans, Western Union
WU	7/17/07	Western Union

WU	12/20/16	Western Union
WY	8/10/06	Weyerhaeuser Company
WYN	2/28/10	Wyndham Hotels & Resorts
WYN	2/16/09	Wyndham Hotels & Resorts
XRIT	4/11/12	X-Rite Incorporated, Pantone.com
XRX	1/23/07	Xerox
YHOO	7/12/12	Yahoo! Voices
YHOO	9/22/16	Yahoo
YHOO	12/14/16	Yahoo
YUM	11/17/17	Pizza Hut
ZEN	2/21/13	Zendesk

Table 3: Company Stock Performance Abnormalities

ticker	evtdate	diff
A	24-Mar-08	0.00504333
AA	15-Jul-10	0.01982667
AAN	2-Nov-11	-0.00987
AAN	22-Oct-13	0.00018
AAP	16-Mar-16	-0.00505
AAP	31-Mar-08	0.01875667
AAPL	1-Apr-11	0.00166333
AAPL	19-Feb-13	0.00268
AAPL	2-Sep-14	-0.0210133
AAPL	22-Jul-13	0.01950667
AAPL	26-Feb-14	0.02055
AAPL	4-Sep-12	0.00382667
AAPL	9-Jun-10	0.00244333
ABB	11-Sep-17	-0.0049267
ABM	14-Nov-17	-0.0074333
ABM	21-Apr-11	0.0007
ADBE	13-May-13	0.01789
ADBE	14-Nov-12	0.00164
ADBE	4-Oct-13	-4.33E-05
ADP	15-Jun-11	0.00157667
ADP	19-Jun-06	-0.0019167
ADP	28-Dec-11	-0.00154
ADP	30-Jul-13	-0.0066533
ADP	5-May-16	0.00108333
ADP	6-Jul-06	-0.0099833
ADVS	10-Jan-07	-0.0005133

AET	12-Dec-06	-0.00711
AET	15-Nov-10	-0.0188167
AET	24-Aug-17	-0.0017733
AET	28-May-09	-0.0162
AET	28-May-10	0.01026
AFL	16-Mar-17	0.00166667
AFL	19-Apr-06	-0.0038033
AFL	22-Aug-06	0.00742667
AIG	14-Jun-06	-0.00535
ALK	26-Jul-17	0.00122333
ALL	23-Aug-11	-0.0189367
ALL	29-Jun-06	-0.00413
ALSK	20-Feb-14	-0.00402
ALU	18-May-07	0.01125667
AMCC	4-Apr-11	0.01212
AMD	14-Jan-13	0.00981667
AMD	9-Apr-12	0.01143
AMTD	1-Dec-06	-0.00645
AMTD	14-Sep-07	0.00079667
AMZN	29-Sep-17	-0.0069133
AMZN	31-Jan-11	0.00285
AN	27-May-14	0.00248667
ANTM	31-Jul-17	0.01337667
ANTM	5-Feb-15	0.00155333
ARW	8-Mar-10	0.00969
AXP	.	0.00969
AXP	13-Jul-12	5.67E-05
AXP	14-Aug-09	0.01703333
AXP	25-Mar-14	0.00088333
AXP	30-Dec-13	0.00142333
AXP	7-Apr-14	0.00102
BA	11-Jul-14	-0.0017267
BA	13-Dec-06	-0.0042733
BA	15-Nov-06	0.00983333
BA	21-Apr-06	-0.01163
BA	27-Feb-17	0.00103
BA	8-Feb-17	-0.0046967
BAC	11-Aug-09	0.01693667
BAC	12-Apr-07	0.00463333
BAC	14-Dec-06	-0.0049067
BAC	14-Feb-11	-0.0114033
BAC	17-Jul-14	0.00620667
BAC	18-Aug-11	-0.0196733

BAC	25-May-11	-0.0037733
BAC	7-Apr-10	-0.00269
BAC	8-Jun-10	-0.00362
BBBY	19-Jun-17	-0.0063567
BBBY	25-Sep-15	-0.0120367
BBT	15-May-08	-0.0255867
BBY	6-May-11	-0.00686
BC	16-Feb-07	0.01812
BC	21-Apr-08	0.01118
BDL	20-May-11	-0.0013867
BEN	3-Aug-06	0.00275667
BGC	19-Nov-07	-0.0285967
BGS	6-Dec-13	0.0104
BHE	21-Nov-17	-0.0030733
BK	26-Mar-08	0.00447667
BKE	20-Jun-17	-0.01617
BKS	24-Oct-12	0.03563
BLKB	17-Jun-09	0.01194667
BMY	17-Jul-08	-0.02839
BOH	1-Mar-13	0.00017333
BR	22-Jun-09	-0.00973
BRLI	25-Aug-14	-0.01994
BSFT	5-Sep-17	-0.0389833
BSX	10-Feb-14	-0.0059567
BUD	29-Jul-08	-0.0058933
C	11-Aug-09	-0.0188433
C	14-Oct-10	-0.0014267
C	17-Jul-13	-0.0152867
C	18-Aug-11	0.01539333
C	19-Jun-08	-0.0204333
C	2-Oct-06	0.00968667
C	21-Sep-07	-0.0017933
C	24-Feb-10	-0.0015633
C	27-Jul-10	-0.0149233
C	28-Mar-13	-0.0033467
C	9-Aug-07	-0.0180633
C	9-Jun-11	0.02359333
CAKE	13-Sep-10	-0.0084367
CAKE	24-May-10	-0.0027933
CAKE	29-Sep-10	0.00259
CAT	27-Apr-07	-0.0039567
CELG	20-Aug-07	-0.0003967
CFR	19-May-06	0.00903667

CHDN	4-Sep-12	-0.0042433
CHH	22-Mar-13	0.00544333
CHH	26-Apr-12	-0.00446
CI	7-Dec-06	0.00250333
CI	7-Nov-06	-0.01389
CMCSA	16-Mar-09	-0.0035267
CMCSA	21-May-12	0.00170667
CMCSA	3-Oct-13	-0.0023233
CME	18-Nov-13	0.01443667
CMG	26-Apr-17	0.00514
CNC	26-Jan-16	0.03275333
CNET	14-Jul-14	0.01632667
CNQR	16-Dec-10	0.00439
COF	12-Feb-13	-0.0043833
COF	18-May-10	0.01651
COF	4-Mar-14	0.00022
COF	6-Feb-17	0.00847
COF	6-Jul-17	-0.0048433
COF	9-May-12	-0.0001733
COLB	21-May-07	0.00885
CPRT	28-Aug-06	0.00250333
CS	20-Feb-07	-0.02036
CSC	3-Apr-13	-0.00332
CSCO	12-Jul-10	0.00503667
CSCO	25-Oct-16	0.00641667
CSCO	9-Apr-12	0.01115667
CVC	25-Jul-06	0.00349333
CVS	16-Apr-07	-0.00024
CVS	18-Feb-09	0.01040667
CVS	20-Jul-15	0.00183333
CVS	26-Mar-12	-0.0126367
CVS	29-Nov-13	-0.0016833
CVS	30-Jul-14	0.00458667
CVS	4-Dec-12	0.00476
CVS	5-Dec-16	-0.0096333
CVX	16-Aug-06	0.01052667
CVX	9-Mar-11	0.00115
D	25-Aug-06	-0.0025367
DBD	31-Aug-06	-0.0037567
DENN	30-Sep-13	0.00027667
DFS	11-Nov-13	-0.0045433
DFS	17-Aug-12	0.00433667
DFS	20-Dec-13	0.00167

DFS	21-Feb-14	-0.0034367
DGX	17-Sep-12	0.01255333
DHI	16-Feb-12	-0.0163033
DIS	1-Aug-16	0.00317667
DLTR	1-Aug-06	0.02518
DNB	26-Sep-13	-0.00154
DNB	28-Oct-13	-0.0032033
DPZ	12-May-11	0.00421
DPZ	18-Jun-08	-0.0012067
DRI	15-Nov-17	-0.00471
DRIV	22-Dec-10	0.00706333
DRIV	4-Jun-10	0.00481667
DTV	11-Oct-06	0.01092667
DTV	29-May-12	-0.0063767
DVA	3-Mar-08	-0.0002167
DVA	7-Nov-13	0.02189667
DXC	5-Jul-17	0.01497333
DYN	21-Oct-16	-0.0502967
EBAY	21-May-14	-0.0104533
EFX	10-Oct-12	-0.0003133
EFX	11-Feb-10	0.00684333
EFX	20-Jun-06	-0.0039233
EFX	6-May-16	0.00227
EFX	7-Sep-17	-0.0680933
EHTH	27-Jan-17	-0.0002433
EL	26-Jul-11	0.00712333
ESRX	19-Feb-13	-0.0134033
ESRX	6-Nov-08	0.00689667
ETFC	9-Oct-15	0.01433667
EV	8-Feb-12	0.0025
EXEL	16-Aug-13	-0.00138
EXPE	15-Nov-06	0.02562333
F	7-May-12	0.00979
FB	15-Feb-13	-0.00517
FB	21-Jun-13	-0.0059133
FB	30-Aug-17	0.00166
FDX	25-Jul-06	-0.0088133
FINL	26-Mar-13	0.03099667
FIRE	27-Nov-12	0.00669667
FIS	24-Sep-07	0.00050667
FIS	26-Aug-11	0.00819333
FIS	3-Jul-07	0.00620333
FITB	13-Apr-06	0.00999333

FLWS	8-Mar-16	0.01192333
FMER	27-Oct-09	-0.0159267
FMS	8-Feb-07	-0.0136067
FORR	5-Dec-07	-0.01644
FRC	14-Aug-12	0.00156667
FRED	12-Jun-15	-0.0133767
FRP	20-Apr-09	-0.1296933
GCI	4-May-17	-0.0325033
GE	16-May-06	-0.00045
GE	25-Sep-06	0.00294333
GE	9-Feb-07	0.00898667
GM	14-Mar-06	-0.01429
GM	3-Aug-12	-0.0077533
GME	2-Jun-17	-0.00355
GNCMA	24-May-12	0.00981333
GOOG	6-May-16	-0.00524
GOOG	9-Mar-09	0.01639
GOOGL	4-May-17	-0.00787
GPI	19-Jul-06	0.00428
GPN	30-Mar-12	-0.0018167
GPS	16-Apr-10	0.00331667
GPS	16-Jul-13	0.00090333
GPS	28-Sep-07	0.00088
GRPN	2-Jul-12	-0.0313467
GS	2-Jul-14	0.00295333
GS	20-May-13	0.00217333
GYMB	27-Oct-06	-0.0165867
H	15-Jan-16	0.03979667
H	16-Nov-17	-0.00354
HBAN	9-May-11	0.00173
HCSG	9-Dec-11	0.00227
HD	13-Apr-12	0.00295333
HD	14-Dec-10	-0.0007533
HD	17-Oct-07	0.00417333
HD	2-Sep-14	-0.0049967
HD	24-May-07	-0.00672
HD	30-Apr-07	0.01304
HD	6-Feb-14	-0.00521
HIG	12-Sep-07	-0.0014367
HIG	30-Oct-07	-0.00308
HIG	6-Apr-11	0.00014667
HLT	25-Sep-15	0.00889333
HMN	12-Nov-07	-0.0277767

HMN	29-Oct-07	-0.04041
HNT	16-Apr-10	0.02267
HNT	18-Nov-09	0.001
HNT	2-Jul-13	-0.0111133
HOG	4-Apr-08	-0.0142433
HON	19-Apr-07	0.01905667
HPE	23-Nov-16	0.02445
HPQ	11-Dec-08	-0.0027133
HPY	20-Jan-09	-0.0939733
HRB	23-Mar-10	0.00134333
HRB	23-Mar-12	-0.0061167
HRB	8-Apr-10	-0.0023333
HSBC	10-Apr-15	-0.00161
HSBC	13-Jan-16	-0.0135833
HSIC	16-Mar-07	0.00319
HUM	12-Dec-06	-0.0003333
HUM	18-Aug-10	-0.0047467
HUM	23-May-14	-0.00631
HUM	5-Jun-06	0.00160667
HUM	9-Oct-15	0.01235
IBM	15-Mar-06	-0.0055833
IBM	15-May-07	0.00013
IHG	26-Jul-16	-0.0062633
IHG	3-Feb-17	0.00326333
IHG	3-Sep-13	0.00642333
IHS	27-Feb-13	0.00436667
ING	12-Oct-10	0.01059
ING	19-Jun-06	-0.00709
INOD	13-Jan-09	-0.07583
INTC	10-Feb-12	-0.00265
INTU	11-May-17	-0.00135
INTU	2-Apr-15	-0.00953
IR	6-Nov-06	-0.0004333
IRM	17-Jan-08	0.00745667
ITT	6-Jan-11	0.00499667
JACK	22-Feb-11	-0.0149733
JIVE	23-Sep-16	0.00259333
JLL	9-Aug-10	-0.00694
JPM	1-Aug-11	0.00214667
JPM	1-May-07	-0.0067
JPM	1-Oct-13	-0.0014967
JPM	12-Feb-13	0.00049
JPM	14-Jun-10	0.00851333

JPM	14-Sep-10	-0.0070867
JPM	19-Jan-10	-0.0085567
JPM	20-Jan-11	-0.0024433
JPM	26-Jan-07	-0.0054533
JPM	28-Aug-14	-0.00315
JPM	28-Mar-13	0.01004
JPM	31-Jan-11	-0.0024733
JPM	5-Dec-13	-0.00026
JPM	7-Sep-06	-0.0004667
JWN	10-Oct-13	-0.0033133
KBH	18-Jan-07	0.01010667
KBR	26-Jan-11	0.004
KELYA	9-Mar-12	-0.0033067
KEY	20-Nov-06	-0.0005667
KEY	3-Jan-07	0.00028
KEY	9-May-12	0.00112667
KFY	12-Oct-12	0.0019
KMB	2-Nov-17	0.00024667
KND	16-Aug-12	-0.0051267
KO	22-Feb-12	-0.0024867
KO	24-Jan-14	-0.0097433
LABL	16-Jun-16	-0.0157633
LCC	6-Apr-11	0.01391333
LH	10-Jun-13	-0.00039
LH	29-Mar-10	-0.0036733
LLL	15-May-12	-0.0054533
LMT	11-Jul-14	-0.0022633
LMT	27-May-11	0.00227
LNC	14-Jan-10	-0.0066633
LNC	17-Sep-12	-0.0005767
LNC	21-Jul-10	0.00273333
LNC	23-Aug-11	0.00810333
LNC	25-May-10	0.00327667
LNC	26-Jul-11	0.00739333
LNKD	6-Jun-12	0.01069
LOW	16-May-06	0.00619667
LOW	19-May-14	0.01241
LOW	22-May-14	-0.0032833
LRCX	14-Apr-10	0.00311667
LUX	26-Nov-08	0.03268
LVS	12-Feb-14	0.00161667
LXK	15-Feb-08	0.02230333
M	23-Apr-13	0.00743

MCD	12-Sep-11	0.01165
MCD	14-Dec-10	0.00587667
MCD	16-Nov-11	0.00333333
MCD	18-Nov-11	0.00389667
MCD	7-Nov-11	-0.0046033
MCD	9-Aug-11	-0.00836
MCD	9-Mar-12	0.00887
MCK	10-Sep-07	0.00842333
MCK	30-Jul-13	-0.01952
MDT	10-Feb-14	-0.00571
MDT	2-Aug-13	0.00912333
MEET	18-Aug-14	0.02469333
MET	10-Aug-10	-0.0117
MET	24-Jan-12	-0.0124433
MET	25-Jan-11	-0.0014767
MGI	12-Jan-07	0.01139
MOH	6-May-14	-0.0143133
MS	5-Jan-15	-0.00887
MSFT	22-Feb-13	-0.0012433
MSG	22-Nov-16	0.00279667
MTB	17-May-06	0.00361
MUSA	.	0.00361
MUSA	8-Nov-10	-0.0118633
MUSA	9-Jun-11	-0.0109733
MWV	1-Nov-07	-0.0023333
MWW	23-Jan-09	0.00367333
NDAQ	18-Jul-13	0.00283
NDAQ	26-Jul-13	0.00173333
NDLS	16-May-16	0.00371333
NFLX	4-Jan-10	0.00117667
NFLX	4-May-11	0.00197333
NFP	30-Oct-06	0.0456
NFP	8-Oct-07	2.67E-05
NGVC	2-Mar-15	0.00095333
NLSN	10-Feb-14	-0.01482
NNI	18-Jul-06	0.00085
NOC	19-Apr-17	-0.0015567
NOC	9-Aug-13	-0.0034633
NSM	12-Aug-15	-0.0220333
NTRS	29-Jul-14	-0.0028567
NTY	15-Jul-10	-0.00596
NUAN	15-Mar-10	-0.0034667
NVDA	13-Jul-12	0.02254

NYT	30-Jan-13	0.00142333
ORCL	12-Nov-07	0.04819333
ORCL	8-Aug-16	0.00144333
OXY	14-Jan-09	-0.00458
PACB	25-Sep-14	-0.00082
PAY	7-Mar-17	-0.01861
PBI	19-Mar-07	0.00580333
PFE	12-May-08	-0.00173
PFG	14-May-10	0.01842667
PFMT	14-Aug-17	0.01730333
PGR	6-Apr-06	0.00467333
PJC	8-Feb-07	-0.0108733
PKI	16-Mar-16	0.00293333
PNC	19-Mar-10	-0.0116867
PRA	11-Aug-10	0.00330667
PRAN	8-Mar-17	0.04358333
PSA	29-Jan-07	-0.0037633
PSS	11-Jun-10	-0.0026567
PULB	16-Jul-12	-0.0031033
PWRD	25-Apr-12	-0.00931
QTM	17-Jun-10	-0.0276767
RAD	19-May-17	0.06994333
RAD	30-Jul-14	0.03878
RAX	2-May-12	0.00695
RCII	25-Apr-12	0.00973
RF	31-Jan-12	0.0008
RRD	28-Jan-13	0.02419333
RUN	2-Feb-17	-0.0010867
S	11-Mar-09	-0.03331
S	16-Aug-17	0.00480667
SABR	17-May-17	-0.00478
SABR	2-May-17	-0.0038167
SABR	7-Aug-15	-0.0292133
SBCF	3-Mar-11	-0.0099767
SBH	5-Mar-14	0.00124
SBUX	12-May-15	0.0036
SCHW	3-May-16	0.00205667
SCHW	9-Apr-10	-0.0077233
SCOR	12-Jun-13	-0.0016333
SEAC	8-Sep-10	0.04250667
SFLY	26-Nov-14	-0.0125167
SFM	28-Mar-16	0.00188
SHLD	10-Oct-14	0.08379333

SMMF	22-Jun-15	0.00046667
SMTC	8-Oct-07	-0.00742
SNE	27-Apr-11	0.00537
SNI	16-Oct-15	-0.00464
SONC	26-Sep-17	0.00560667
SPLS	20-Oct-14	0.01032667
SRCE	10-Jun-08	0.00208333
STI	16-May-11	0.00923333
STT	29-May-08	-0.0012867
STX	7-Mar-16	0.00031667
SVU	15-Aug-14	0.00329
SWK	11-Mar-13	0.00317333
SYMC	31-Mar-09	0.00653333
T	9-Jun-10	-0.0101033
TAX	13-Feb-15	-0.0068
TD	15-Mar-10	-0.0060733
TGT	13-Dec-13	0.00054333
TJX	17-Jan-07	0.00693333
TM	26-Aug-16	0.00464333
TM	4-Aug-06	0.00099333
TMUS	12-Oct-17	0.00486667
TMUS	7-Dec-16	-0.0239933
TOO	9-Nov-17	-0.0398067
TRI	11-Aug-10	0.01231333
TRMK	22-Jun-15	0.00559333
TRU	30-Nov-06	-0.0150933
TTEC	21-Jun-10	0.0079
TWC	28-Jul-10	0.01402667
TWC	8-Jan-16	-0.0065267
TWTR	13-Jun-16	0.04446
TWTR	19-May-17	0.00046
TWX	30-Oct-17	-0.0073533
TWX	31-Jul-17	-0.0098567
TXT	31-Jul-07	0.00839333
TYL	13-Mar-17	-0.0018
UA	20-Apr-12	-0.02126
UAL	29-Jul-15	-0.01277
UBNT	7-Aug-15	-0.00469
UBS	7-Nov-07	0.0185
UNH	12-Oct-11	-1.33E-05
UNP	16-Jun-06	-0.0090233
UPS	20-Aug-14	-0.0097733
USB	28-Sep-10	0.00772

VIAB	20-Sep-17	0.00556
VMED	25-Oct-07	0.00421
VRA	12-Oct-16	-0.00441
VRSN	2-Feb-12	-0.0166433
VSTO	19-Sep-16	0.00481333
WASH	28-Aug-08	-0.0277667
WCC	3-Nov-06	-0.00251
WCG	8-Apr-08	0.01199333
WEN	27-Jan-16	0.01807333
WEN	28-Jul-10	-0.0064567
WFC	1-Sep-06	0.00381667
WFC	31-Jul-17	0.00759333
WIN	27-Jan-12	0.00783
WINN	25-Jun-07	0.01490333
WLP	10-Feb-10	-0.0059233
WM	3-Apr-07	-0.00334
WMB	3-Aug-09	-0.01485
WMT	28-Sep-07	0.01419667
WSM	17-Aug-06	-0.03001
WU	20-Dec-16	0.00245
WY	10-Aug-06	-0.00462
WYN	1-Mar-10	-0.0052933
XRIT	11-Apr-12	-0.1164667
XRX	23-Jan-07	0.01905333
YHOO	12-Jul-12	-0.00946
YHOO	14-Dec-16	-0.01637
YHOO	22-Sep-16	-0.0013767
YUM	17-Nov-17	0.00539667