Effects of Earnings Announcements on Stock Options

MS&E 444: Group 4
Spring 2007

Geoffrey Evans
Pablo Jadzinsky
Yu Lau
Jianlin Wang
Ryan Williams

{gde, jadz, yulau, jianlin, ryanw1}@stanford.edu

I. Introduction and Background

EvA asked as to look from a statistical point of view at the way volatility behaves as earning announcements approach. One of EvA's strategies is to edge between options that are wrongly priced. Since it is known that volatility behaves differently around earning announcements, trading around this dates is stopped since the concept of underpriced/overpriced might have to be defined differently.

II. Fat Tails

Conventional wisdom dictates that, in general, daily stock log-returns are essentially normally distributed except around earnings announcements, which are unusually volatile periods and cause the overall distribution to have "fat tails," that is, an increased presence of high and low outliers relative to what we would expect from a normal distribution. However, inspired by a request from EvA we decided to explore the hypothesis that, even ignoring stocks' performance immediately surrounding earnings announcement dates, the distribution of daily returns is still fat-tailed. Gaining a better understanding of how earnings announcements affect the movements of stock prices would allow trading strategies to be adapted for these unusual periods so that trading does not have to be temporarily suspended.

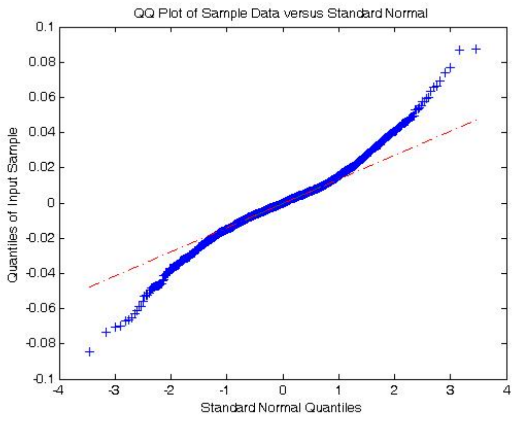
Since the hypothesis refers to the basic manner and volatility with which stocks move, we decided early on that it would be important to make sure we conducted our experiment across a diverse range of stocks in different industries and with different histories, to be sure that anything we found was indeed a more fundamental pattern than any trend that we might have encountered in one isolated sector or another. We also decided that we should just observe the net intraday changes of each stock that we examined, as a percentage of its value, rather than any finer increment of time.

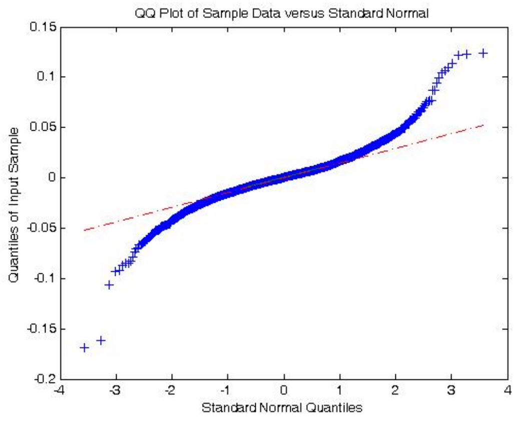
To measure how close a given stock's distribution of daily log-returns was to a normal distribution, we measured the distribution's skewness and kurtosis, and examined Q-Q plots of the sample data against a theoretical normal distribution. Skewness is defined as the third moment divided by the cube of the standard deviation, and it measures how symmetric a distribution is. Positive skewness implies that the distribution is asymmetric with a heavier right-side, while negative skewness implies that the left side of the distribution is heavier. Kurtosis is defined as the fourth moment divided by the square of the variance, and it measures how peaked a distribution is. A high kurtosis implies the distribution is strongly peaked and has fat tails, while low kurtosis implies the distribution has thin tails.

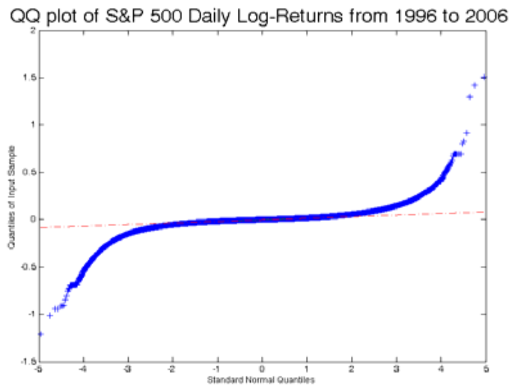
A normal distribution has skewness of 0 and kurtosis of 3, but our analysis of historical log-returns showed substantially higher kurtosis and slightly positive skewness. This implies historical log-returns are fat-tailed and have a slightly heavier right tail. More precisely, the Jarque-Bera and Lilliefors tests of normality both gave p-values of less than 0.001 for every stock we considered, indicating that the log-normal distribution is a poor model for stock returns.

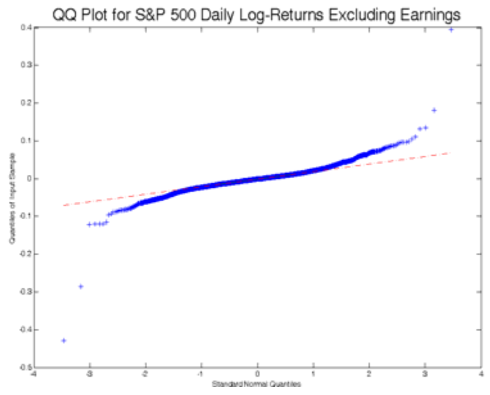
While skewness and kurtosis statistics are useful numerical tools for examining how fat-tailed a distribution is, we also used Q-Q plots to examine the relationship graphically. A Q-Q plot shows the relationship by plotting the order statistics of the historical log-returns against the quantiles of a theoretical normal distribution. If the historical data were a good match for the normal distribution, the Q-Q plot would show a linear relationship, but that was not what we found.

We tested 5 groups of 10 stocks each, from different distinct sectors of the market, and generated plots for each one showing how close the distribution of its intraday earnings was to a normal distribution, and then we repeated this experiment but omitting a 14 day window around (quarterly) earnings dates. We also repeated the experiment over all of the constituent stocks of the S&P 500. Below are some of examples of the type of phenomenon we observed:









As you can see, the original distributions are quite fat-tailed, judging by their kurtosis or the diverging tails of their Q-Q plots. However, a large measure of this fat-tailed-ness seems to disappear when we remove a moderate window of days around earnings announcements, although it is still apparent that there is something besides a normal distribution taking place here!

While this analysis of stock returns may seem somewhat boring, it is important to remember the substantial benefits of even a slightly better model of market forces. For EvA, a heightened awareness of how stock movements are usually distributed both around earnings and not, and how some common assumptions about their normality often prove to be extremely inaccurate during periods around earnings would be invaluable.

III. Characterizing Implied Volatility near Earnings

1. Moneyness and normalizing implied volatilities

In order to compare various options with different maturities and strike prices, we need a tool to normalize the options. In our project, we chose the measure of moneyness which is defined as the number of standard deviations the current price is above the at-the-money forward price. Thus, for example, moneyness is zero when the forward price of the underlying, discounted at the risk-free rate, equals the strike price. For the purpose of our project, we did not incorporate the risk-free rate in our measure of moneyness as we planned because we were short of time at the end. However, since we were only looking at 2 to 3 month time horizon, we believe incorporating risk-free rate would not have significant effects to our results.

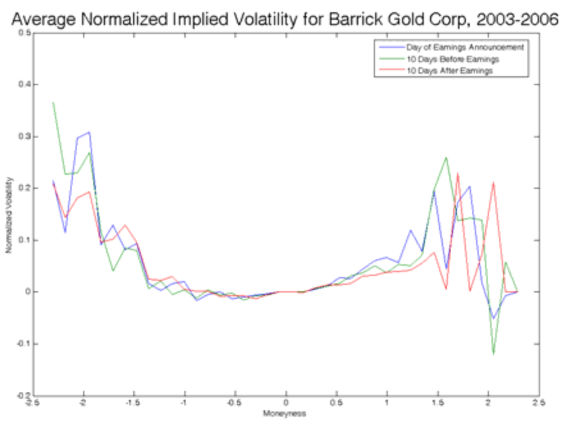
Moneyness is defined as: $m = \dfrac{\ln(S/K) + rT}{\sigma\sqrt{T}}$

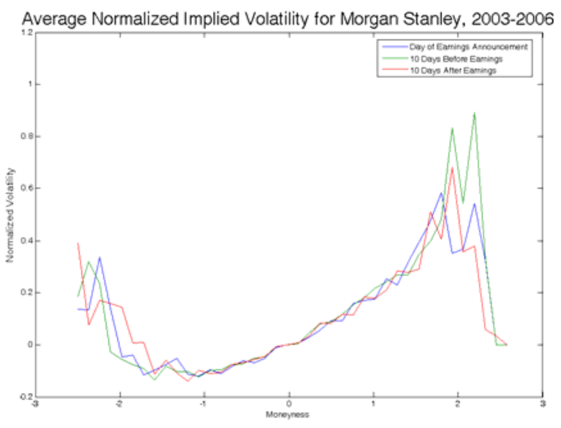
On the other hand, we would like to normalize the implied volatilities to allow us to compare results over different underlying stocks. We normalize the implied volatilities by subtracting the at-the-money volatility and dividing by the at-the-money volatility. This way, for example, two at-the-money options for different companies would have the same value after the normalization.

Normalization of implied volatility: $\tilde{\sigma} = \dfrac{\sigma - \sigma_{atm}}{\sigma_{atm}}$

2. Characterizing movements of implied volatilities around earning dates

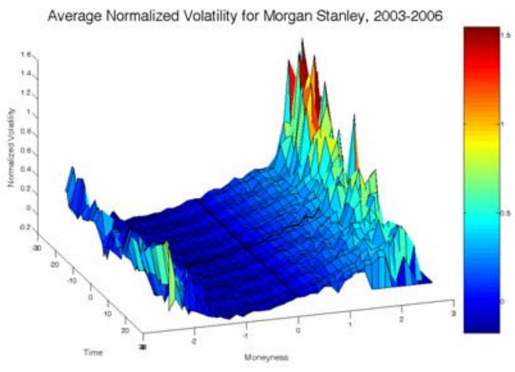
After we have done with the normalization, we developed scripts to plot implied volatility surface for an arbitrary window around various earning dates. For the purpose of our project, we looked at 30 different companies from 5 sectors for earning dates from 2003 to 2006. Our results showed, against conventional wisdom, the implied volatilities generally do not show significant fluctuation around earnings. In order to examine the effects more closely, we generated 2-D plots to show implied volatility curves 10 days before earnings announcement, at the day of earnings announcement, 10 days after earnings announcement. Below are plots we have for Morgan Stanley and Barrick Gold:

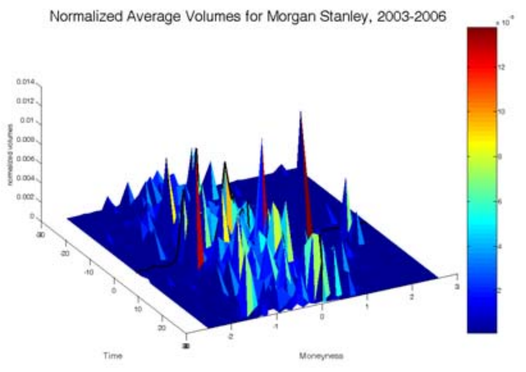




As we can see from the plots, the implied volatilities only fluctuate slightly before and after earnings. After examining these plots from a number of stocks across a variety of industries, it seems to us that the fluctuations are only due to noise rather than any meaningful statistical patterns.

Below we show an average volatility surface and an average volume surface across all earning announcements from January 2003 to December 2006 for Morgan Stanley. From generating this type of plot for the aforementioned companies we concluded that most activity is as expected centered at the ATM options. We also see the volatility smile in every case with options more than two sigmas away from the underlying price trading only occasionally.





We also calculated the standard deviations of the averages in volatility and volume but they turned out to be not very informative. The main conclusion was that the standard deviation was high anywhere where the averages where high. We thought this idea could be used to implement a trading strategy identifying options that were far away from the average according to some weighing against the standard deviation of the averages and therefore be able to buy options that were statistically underpriced or sell options that were statistically overpriced but we couldn’t find enough time to do this.

IV. Appendix - Matlab Software Packages

In order to study the fat-tail and the implied volatilities effects of stock options around their earning dates, we developed two software using Matlab: fat tail analysis package and normalized implied volatilities analysis package. For both packages, we have provided many considerations to handle the missing, inconsistent and/or unsorted data from the database and the efficiency of analysis.

(A) Fat Tail Analysis Package

In order to study the fat tail effect around earning dates for a specific stock ticker, the following data are required from WRDS (Wharton Research Data Services):

- Data of the stock daily returns
  This data may be downloaded from the CSRP database and stored in the fixed width format with three columns: PERMNO, DATE, and RETX, where RETX are the

daily returns without dividends. The data file may be named as ticker_daily_returns.txt.

- Data of the stock earning dates
  This data may be downloaded from the First Call database and stored in the fixed width format with five columns: TSYMBOL, ACTDATE, Data_Type, Periodicity, and Value, where ACTDATE is the actual earning reporting dates and periodicity is quarterly. The data file may be named as is ticker_earning_date.txt.

Both data should have consistent starting and ending dates for the software to work properly.

Table 1. Matlab Code for Fat-Tail Analysis of a Stock

(1) exclude_earning_dates.m	
Functionality:	Given the daily returns and earning dates of a stock and a time window around its earning dates, split the daily returns into two parts; one containing those data around earning dates and the other one excluding those data.
Input parameters:	• An Matlab array of earning dates • An Matlab array of daily returns • Time window around earning dates
Input files from database:	None
Outputs:	Two sorted arrays: one containing daily returns around earning dates and the other one excluding those data.
(2) main_histogram.m	
Functionality:	Entrance code
Input parameters:	• Stock ticker • Time window around earning dates
Input files from database:	• ticker_daily_returns.txt • ticker_earning_dates.txt
Outputs:	• Histogram including/excluding the time window around earning dates • Q-Q normality test results
(3) read_daily_returns.m	
Functionality:	Given a stock ticker, load the data file of its daily returns to the Matlab memory
Input parameters:	Stock ticker
Input files from database:	ticker_daily_returns.txt
Outputs:	A Matlab array storing the daily returns
(4) read_earning_dates.m	
Functionality:	Given a stock ticker, load the data file of its earning dates to the Matlab memory
Input parameters:	Stock ticker
Input files from database:	ticker_earning_dates.txt
Outputs:	A Matlab array storing the earning dates
(5) plot_daily_returns_histogram.m	
Functionality:	Given a stock ticker, plot its daily log returns including

	and excluding earning dates and perform Q-Q normality tests
Input parameters:	• Stock ticker • Time window around earning dates
Input files from database:	• ticker_daily_returns.txt • ticker_earning_dates.txt
Outputs:	• Histograms of the daily returns including and excluding earning dates • Q-Q normality test results
(6) plot_earnings_returns_histogram.m	
Functionality:	Given a stock ticker, plot its daily log returns during earning dates
Input parameters:	• Stock ticker • Time window around earning dates
Input files from database:	• ticker_daily_returns.txt • ticker_earning_dates.txt
Outputs:	Histograms of the daily returns during earning dates
(7) stringnum.m	
Functionality:	This is a utility function; given a string, return a numerical hash value for the string
Input parameters:	A Matlab string
Input files from database:	None
Outputs:	The string hash value

Table 1 lists the Matlab source code for fat-tail analysis of an individual stock. All the files listed can be viewed as black box, except main_histogram.m, which is the main entrance for the software. It is quite straightforward to modify the input parameters in main_histogram.m to perform the fat-tail analysis; for example, the following code shows how to study the fat-tail effect of stock 'GS' with a time window of 14 days around its earning dates:

```
window_size = 14;
stock_symbol = 'gs';
plot_daily_returns_histogram(stock_symbol, window_size);
```

To perform the fat-tail analysis of multiple stocks simultaneously, two data files similar to those of a single stock are downloaded; after some preprocessing by python, similar Matlab code can be developed and similar analysis can be performed just as the single stock case. For example, to study the fat-tail effect of S&P 500, the two data files can be named as snp500_daily_returns.txt and snp500_earning_dates.txt respectively; the Matlab code is:

```
window_size = 14;
descriptor = 'snp500';
plot_multi_daily_returns_histogram(descriptor, window_size);
```

Table 2 lists the Matlab source code for fat-tail analysis of multiple stocks.

Table 2. Matlab Code for Fat-Tail Analysis of Multiple Stocks

(1) exclude_multi_earning_dates.m	

Functionality:	Given the daily returns, earning dates and a time window, split the daily returns into two parts; one that contains those data around earning dates and the other excluding those data.
Input parameters:	• An Matlab array of earning dates • An Matlab array of daily returns • Time window
Input files from database:	None
Outputs:	Two sorted arrays: one containing daily returns around earning dates and the other one excluding those data.
(2) plot_multi_earnings_returns_histogram.m	
Functionality:	Given an descriptor (a user-specified string to save the downloaded data) and the time window, perform the fat-tail analysis
Input parameters:	• A descriptor string • Time window around earning dates
Input files from database:	• descriptor_daily_returns.txt • descriptor_earning_dates.txt
Outputs:	• Histogram including/excluding the time window around earning dates • Q-Q normality test results
(3) read_multi_daily_returns.m	
Functionality:	Given a descriptor, load the data file of the daily returns to the Matlab memory
Input parameters:	Descriptor string
Input files from database:	descriptor_daily_returns.txt
Outputs:	A Matlab array storing the daily returns
(4) read_multi_earning_dates.m	
Functionality:	Given a descriptor, load the data file of the earning dates to the Matlab memory
Input parameters:	Descriptor string
Input files from database:	descriptor_earning_dates.txt
Outputs:	A Matlab array storing the earning dates

(B) Implied volatilities Analysis Package

For an individual stock, there may be multiple options with different issuing dates, strike prices or expiration dates. Hence the option data may be huge even for a small date ranges. To improve the efficiency, we divide this software into two subsystems: one is for filtering the option data satisfying our specified criteria and the other one is for the statistical analysis of normalized implied volatility and volumes. The first subsystem is slow but only needs to run once; while the second subsystem is fast and may be used frequently for real analysis.

(B1) Subsystem of Filtering Option Data

Provided the earning dates, stock prices and daily option data of an individual stock during some date ranges, this system filtered out the option data based on certain criteria, calculate the moneyless value, the normalized implied volatilities and volumes of the selected options and store the chosen/calculated quantities to disk files.

The following data are required from the WRDS (Wharton Research Data Services):

- Data of the stock daily returns
  This data may be downloaded from the CSRP database and stored in the fixed width format with three columns: PERMNO, DATE, and RETX, where RETX are the daily returns without dividends. The data file may be named as ticker_daily_returns.txt.
- Data of the stock earning dates
  This data may be downloaded from the First Call database and stored in the fixed width format with five columns: TSYMBOL, ACTDATE, Data_Type, Periodicity, and Value, where ACTDATE is the actual earning reporting dates and periodicity is quarterly. The data file may be named as is ticker_earning_date.txt.
- Data of the stock options
  This data may be downloaded from the OptionMatrics database and saved as comma delimited csv file with eleven columns: date, exdate, cp_flag, strike_price, volume, open_interest, impl_volatility, optionid, index_flag, issuer, exercise_style. The data file may be named as is ticker_options.csv.

All the above data should have consistent starting and ending dates for the software to work properly. The first and last earning dates are excluded in the software.

Our option data selection criteria are:

- Only call options are considered.
- Only options that are issued before the first day of the time window around earning dates and expire after the last day of the time window around earning dates.

The chosen and calculated option data are saved to disk file with name convention of ticker_picked_call_options.data.

Table 3. Matlab Code for Option Data Filtering

(1) main_option_data.m	
Functionality:	Entrance code
Input parameters:	• Stock ticker • Time window around earning dates • A parameter to shift the earning date (pseudo earning date for exploring the earning effect)
Input files from database:	• ticker_earning_dates.txt • ticker_daily_returns.txt • ticker_options.csv
Outputs:	ticker_picked_call_options.data
(2) picked_optionids_by_earning_dates.m	
Functionality:	Based on earning dates and the option data in Matlab, select the option ids satisfies our specified criteria.
Input parameters:	Time window around earning dates
Input files:	None
Outputs:	Matlab array of selected option ids.

(2) pick_option_data.m	
Functionality:	Given a stock ticker, select the related option data with our specified criteria and calculate the related moneyless values and normalized implied volatilities and volumes
Input parameters:	• Stock ticker • Time window around earning dates • A parameter to shift the earning date (pseudo earning date for exploring the earning effect)
Input files from database:	• ticker_earning_dates.txt • ticker_daily_returns.txt • ticker_options.csv
Outputs:	ticker_picked_call_options.data
(3) read_daily_returns.m	
Functionality:	Given a stock ticker, load the data file of its daily returns to the Matlab memory
Input parameters:	Stock ticker
Input files from database:	ticker_daily_returns.txt
Outputs:	A Matlab array storing the daily returns
(4) read_earning_dates.m	
Functionality:	Given a stock ticker, load the data file of its earning dates to the Matlab memory
Input parameters:	Stock ticker
Input files from database:	ticker_earning_dates.txt
Outputs:	A Matlab array storing the earning dates
(5) read_optoins.m	
Functionality:	Give a stock ticker, read the downloaded option data from OptionMatrics to Matlab memory.
Input parameters:	Stock ticker
Input files from database:	ticker_options.csv
Outputs:	Matlab arrays for the option related data

Table 3 lists all the related Matlab source code. All the files listed can be viewed as black box, except main_option_data.m, which is the main entrance code for the subsystem. It is quite straightforward to modify the input parameters in main_option_data.m to perform the implied volatilities analysis; for example, the following code shows how to study the stock 'GS' with a time window of 30 days around its earning dates:

```
window_size = 30;
stock_symbol = 'gs';
before_earning_dates = 0;
pick_option_data(stock_symbol, window_size, before_earning_dates);
```

(B2) Subsystem of Statistical Implied Volatilities and Volume Analysis

Provided the selected option data, the moneyless quantity, normalized implied volatilities and volumes, this system plots the 3D plots of the following three quantities: dates, moneyless and normalized volatilities/volumes; the average surfaces and their standard deviation are also plotted. To better viewing, the 2D curves of normalized implied volatilities/volumes vs. moneyless for some specified dates before/after earnings dates are also plots.

The data files required fro this package is the ticker_picked_call_options.data from the first subsystem.

Table 4. Matlab Code for Statistical Implied Volatilities and Volume Analysis

(1) main_plot_options.m	
Functionality:	Given a stock ticker, perform the statistical analysis of the normalized implied volatilities and volumes
Input parameters:	Stock ticker
Input files:	ticker_picked_call_options.data
Outputs:	Statistical plots of normalized implied volatilities and volumes
(2) normalizedV1.m	
Functionality:	Calculate the normalized volatility
Input parameters:	• Implied volatility • Implied volatility of the at-the-money option
Input files:	None
Outputs:	Normalized volatility
(3) read_picked_option_data.m	
Functionality:	Given a stock ticker, read the data from ticker_picked_call_options.data to Matlab memory
Input parameters:	• Stock ticker • Flag indicating if the data is for call or put options
Input files:	ticker_picked_call_options.data
Outputs:	Matlab arrays for the related data.

Table 4 lists all the related Matlab source code. All the files listed can be viewed as black box except main_plot_options.m, which is the main entrance of this package. Using stock GS as an example, the following code shows how to modify the input parameter of main_plot_options.m and run the package:

```
stock_ticker = 'GS';
(original main_plot_options.m code)
```

Due to the huge data information, the main_plot_options.m code doesn't have clean and simple interface as the other main entrance functions from the other software. One advantage of this system design to allow users easily modify existing code or append additional code for desired plots or analysis.