Visualisation of High Frequency Data: Sentiment Analysis in Financial Markets

Barry Rafferty-Lynam
B.A.I. Engineering
Final Year Project April 2012
Supervisor: Professor Khurshid Ahmad

School of Computer Science and Statistics
O’Reilly Institute, Trinity College, Dublin 2, Ireland
Declaration

I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

________________________________________
Barry Rafferty-Lynam, April 5th 2012
Permission to Lend

I agree that the Library and other agents of the College may lend or copy this thesis upon request.

_______________________________
Barry Rafferty-Lynam, April 5th 2012
Abstract

This paper studies the processes involved in the design and production of a dynamic data graphing utility to be used for the representation of sentiment analysis in financial markets. It explores areas of research relating to the visualization of data series, sentiment fundamentals and financial analysis methods. It is an interesting combination of financial methods that bridge the gap between the quantitative study of data fundamental to the end functionality of the software and the interaction of said software with prospective end users. With the increase in computation power and the size of electronic data storage available there is a growing need to further the ability to represent vast stores of numeric data in a human readable format. This paper aims to demonstrate that data visualization via graphs is an effective way to do so.
“If I can’t picture it, I can’t understand it”

- Albert Einstein
Acknowledgements

I would like to thank my supervisor Professor Khurshid Ahmad for his support and patience throughout the year. He has been a much needed voice of reason at many times when aspects of the project seemed quite daunting.

Professor Ahmad’s PhD student, Stephen Kelly, was also an invaluable resource having just last year completed his own final year project and was familiar with the challenges of the process.

Lastly my family, to whom I owe a debt of gratitude, for their constant support and motivation over the last year and always.
# Contents

1 Introduction ........................................................................................................ 9  
   1.1 Background and Personal Motivation .......................................................... 9  
   1.2 Dataset ......................................................................................................... 9  
       1.2.1 Financial ........................................................................................... 10  
       1.2.2 Sentiment .......................................................................................... 10  
   1.3 Example Input File ...................................................................................... 11  
   1.5 Report Overview ......................................................................................... 11  

2 Data Characteristics, Financial Analysis, Visualisation .................................. 12  
   2.1 High Frequency Financial Data .................................................................... 12  
   2.2 Sentiment Analysis ...................................................................................... 13  
   2.3 Financial Analysis Methods ......................................................................... 14  
   2.4 Visualisation ............................................................................................... 17  
       2.4.1 Current Visualisation Techniques in Financial Analysis .................. 19  
       2.4.2 Visualisation Summary ...................................................................... 20  

3 Formulation of Process Methodology ............................................................... 21  
   3.1 Handling User Input ................................................................................... 21  
   3.2 Extracting Data from the Input File ............................................................. 22  
       3.2.1 Additional Data Manipulation Options .......................................... 23  
       3.2.2 Secondary User Input ...................................................................... 24  
   3.3 Generating Graphic Display ....................................................................... 24  
       3.3.1 Post Visualisation Options ............................................................. 25  
   3.4 Final Review of Process Methodology ....................................................... 26  
   3.5 Process Summary ...................................................................................... 26
4 Implementation of the Developed Program

4.1 Language Choice and Libraries Used ...............................................................27
4.2 Program Layout ..................................................................................................27
4.3 Data Handling – File Reader .............................................................................29
4.4 Data Handling - Date Management ....................................................................30
4.5 Data Handling – ‘Graph Options’ User Interface .............................................30
4.6 Generating Graph ............................................................................................32

5 Conclusion

5.1 Project Review ....................................................................................................33
5.2 Comments and Future Work ............................................................................34

Bibliography .................................................................................................................35
1 Introduction

Background and Personal Motivation

At the time of writing, the current global economic climate sees Europe still in a state of recession that began in 2007 following one of the largest financial crises to hit the global economy. The United States is in a tentative state of recovery having seen its Standard & Poor’s credit rating downgraded, for the first time ever, in August 2011. Having experienced the last five years of economic uncertainty accompanied by a constant news supply endeavoring to describe it in layman’s terms one could be forgiven for beginning to think that we are all merely passengers on a rollercoaster that is the financial markets.

However from an engineering point of view this idea is unacceptable. Some of the immediate questions that come to mind are; what needs to be studied to form an understanding of market drivers? Why is it so hard to explain this to the general public, particularly given five years to do so? If the understanding of markets, and market components comes from number crunching huge datasets what are suitable methods of aiding people form a clear mental picture of the results? This last question is fundamental to this project; how can a computer program be used to enhance an individual’s comprehension of a dataset? This project hopes to demonstrate that one such way is through the use of a graph display.

Personally, this strikes a chord as one of the main attractions of computer engineering to me was, and is, its application in the automation and simplification of processes that can aid or benefit people in everyday tasks. Whether through a reduction in the workload involved or an increase in gain provided to the user.

1.2 Dataset

The dataset used in the planning and development phase is representative of the layout and content of the datasets that are expected to be used by the end user. It comprises of quantitative information regarding sentiment analysis and financial records over an individual time series. These two components come from very different sources and will have already undergone much individual and joint processing by the time they arrive as input in this project. The financial data is quoted in periods of one minute and has been scaled up to create datasets by the hour, day, week and month. Similarly the sentiment data is scaled by the publication time of the texts parsed in its compilation. These times are variable and in order to use the financial and sentiment data together a combined time scale has been created. As much as possible this scale has strived to provide a dataset organised by minute, hour, day, week and month but at higher frequencies it is not guaranteed to be free of any missing data points for each and every minute.
However this is still an acceptable input for the program as the format is consistent with the generic format of the expected input by end users.

1.2.1 Financial
The financial components of the sample dataset are crude oil future contracts (CL1) taken from the Chicago Merchantile Exchange (CME), an exchange that trades future contracts continuously. The particular crude oil, West Texas Intermediate, is used as a benchmark in oil pricing in North America. Contracts traded continuously are typically more concerned with the relative delivery time as opposed to focusing on a specific delivery month. This means it is possible to build a historical series of contract prices that should aid the understanding of price behavior over time when visualised by the program detailed in this thesis.

1.2.2 Sentiment
The sentiment data supplied as a sample input to the program is generated from oil centric articles that have been collected autonomously via RSS feeds from the following sources:
- TradeCrudeOil.net
- Oil and Gas Journal
- Oil Drum Feed
- Washington Post Oil and Gas Feed
- Oil Marketer Feed
- Scandinavian Oil-Gas Magazine – News
- Scandinavian Oil-Gas Magazine – Financial
- Scandinavian Oil-Gas Magazine – Oil
- Oilvoice - Main Headlines
- The Street - Energy
- Seeking Alpha - Energy
- Oil Price
- Energy Today(EIA)

Links to these sites can be found at the end of the bibliography.

Once collected the articles are compiled into an ‘oil corpus’ containing only relevant oil news. The analysis methods include both natural language processing and statistical methods with a general language dictionary used for the body of the text and a specialised oil dictionary for more specific terms. Finally the polarity of the sentiment is calculated as the frequency of negative and positive terms. This is done for a selection of oil related categories such as crude oil, economy, military, or mentions of companies such as BP, Chevron, Exxon etc. These values can be seen in figure 1.4.1 in the following section.
1.3 Example Input File

From figure 1.4.1 it can be seen that the columns A and B contain time scale information. Columns D to N represent sentiment categories such as, CrudeOil, Economy, Military etc, as outlined in the previous section and from column O onwards is financial data related to oil future prices. This is the file generated when the financial and sentiment components of the dataset are compiled into one file on a single time scale.

1.5 Report Overview

The discussion in this thesis begins by detailing the research carried on relevant papers and articles in order to acquire the background knowledge and deeper understanding of the challenges involved in the visualisation of financial data. The result of this research forms the foundation necessary to actively explore possible methods for meeting these challenges.

The thesis then moves on to detailing the methodology developed in the design of a dynamic process that can be used to successfully generate graphic displays of numeric datasets found in spreadsheets files.

Following immediately on from this, the thesis documents the implementation of a java graphing application that uses the developed process as its blueprint.

Finally, the thesis concludes with a review of the project from a personal perspective and a brief discussion on how the project could be extended upon in future work or added to larger programs to provide a specific graphing utility.
2 Data Characteristics, Financial Analysis, Visualisation

2.1 High Frequency Financial Data

The original high frequency financial data is generated from tick-by-tick recording of transactions which intuitively do not occur at discrete time intervals. Ticks record and publish the value of transactions as they occur for a commodity thus follow a stochastic pattern, they can be positive negative or zero ticks depending on the upward, downward or lack of movement between consecutive prices. This is contrary to the methods of analysis applied to financial data that require fixed time intervals. Indeed the majority of published financial literature deals with much longer time periods such as day, week and month (Dacorogna, Gencay, Muller, Olsen, & Pricpet, 2001). In finance, any intraday recording is deemed high frequency and has only been realised with improvements in electronic computation and increased availability of mass data storage. For example, a frequency of minutes could contain up to 1440 data points but when compared to more traditional frequency, such as daily time intervals, is almost equivalent to the maximum number of data points in four years worth of recording. While the input data for this thesis has been augmented to a homogeneous time series it is none the less important to understand the characteristics of high frequency data that can impact, at times detrimentally, the results of any analysis applied to data constrained to static time intervals.

The defining characteristics of high frequency data are irregular temporal spacing, discrete but closely related data and daily patterns as laid out in (Engle & Russell, Analysis of High Frequency Financial Data, 2004). The irregularities in spacing can see numerous transactions per second followed by transactions separated by whole or multiple minutes. Reasoning behind this atypical pattern is the topic of many studies and publications in itself and includes sentiment analysis which will be introduced in the next section. It is worth noting that as the frequency is increased the markets are more inclined to display event driven patterns.

The leading difficulty in handling randomly spaced data points is that historically, financial analysis has been developed for specific, known time intervals and this is also the case for econometric models (Dacorogna, Gencay, Muller, Olsen, & Pricpet, 2001). In order to interpolate a stochastic model to a rigid sequence concessions must be made regarding how to treat data that occurs more or less frequently than the chosen time period. These concessions will likely see instances of data point repetition and conversely, occasions where data points are lost. In all cases interpolation will impact any future applied analysis.

The second characteristic, discreteness, acknowledges the nature of all economic data. Over traditional financial time periods, those greater than daily, prices can be seen to
rises and fall steadily, within a factor of a particular commodities standard deviation. However, as frequency of observation is increased to intraday level the variation between consecutive recordings diminishes and the plot of a series contains plateaus where consecutive points are the same. This is one illustration of where the decisions made in how to conform a stochastic series to a fixed one has impact. Figure 2.1.1 below, contains example of identical consecutive data points, or ‘zero ticks’ that are a feature of high frequency data.

![Close Price](image)

Figure 2.1.0-1 : Plot of oil future prices over a two hour period at intervals of five minutes

Finally, high frequency financial data exhibits daily patterns that follow the daily flurries of global stock markets. In tandem with the noise levels on trading floors the inter transaction period is at a minimum following market opening time and prior to close of day. With increased activity comes an increase in the behaviour of the results of analysis applied and it can be seen that volatility is greatest at these times. This is a statistic first noted in (Engle & Russell, 1998).

### 2.2 Sentiment Analysis

In 1936 John Maynard Keynes coined the term “animal spirits” in an effort to describe the emotions that influence human behaviour that are made visible in consumer confidence indexes (Keynes, 1936). In the intervening years much time has been devoted to the study of the seemingly wild, or spontaneous fluctuations of stock market prices and to identify their determinants (Cutler, Poterba, & Summers, 1989). In order to empirically analyse this behaviour focus is turned to information arrival in the market as traders use both public and private knowledge of stocks to drive their actions. Studies in this area have a common theme of the media being a main source of information that arrives in the market and have strived to quantitatively convey the role of the media in affecting market sentiment (TETLOCK, 2007).
Inferring sentiment from text is a multiphase, interdisciplinary task that requires human emotions to be identified from a psychological background. Finite lexicons of words and phrases that identify these emotions must be then be created by linguists before computational linguistics can develop statistical models that accurately portray the sentiment of a text. Only with this work completed can text be quantitatively analysed for use in engineering environment.

In trying to establish a baseline for his analysis, Tetlock asserts that media articles can be considered optimistic or pessimistic regarding a particular topic, and each has its own particular characteristics of impact. Pessimistic articles tend to generate longer lasting impact on the market whereas optimistic articles are more transient in market memory.

Building on the optimistic versus pessimistic view of sentiment, the identification of polarity is employed to give sentiment a numerical value that can then be used for quantitative analysis as seen in (Devitt & Ahmad, 2007). The use of polar values enhances the identification of turning points in sentiment which will be the object of interest for visualising data graphically in this thesis as turning points correlate to information arrival. Further application of financial analysis will endeavour to enhance these peaks to give a greater understanding of events occurring. Of significant impact is the relative occurrence of works in ‘phrases of speech’, this is to account for the context of where terms and phrases are used since positive words can be used in phrases expressing negative sentiment and visa versa, eg ‘awfully successfully’ contains a words ‘awfully’ which would usually be considered to have negative impact (Wilson, Wiebe, & Hoffman, 2005).

2.3 Financial Analysis Methods

As already seen in section 2.1 price data for a commodity typically remains close to the mean value.

A widely used statistic to measure variation is the standard deviation which shows the dispersion from the average. All variation from the mean is then considered to fall within a multiple of the standard deviation. In finance low orders of deviation are favoured. A low result for standard deviation indicates a stable series with data points lying close to the mean while a high value indicates volatility in the market. The standard deviation, is calculated as follows:

\[
\sigma = \sqrt{\frac{\Sigma(x - \bar{x})^2}{N}}
\]
Where $x$ is the current data point, $\bar{x}$ is the mean and $N$ is the population of the series. With the standard deviation known the variance of individual values from the mean can then be calculated. It is simply the square of the standard deviation:

$$\sigma^2 = \frac{\sum(x - \mu)^2}{N}$$

Where $\mu$ is the mean, $N$ is the total number of data points, $x$ is again the current data point. Finally, to categorise the location of a single value as a multiple of standard deviations from the mean, known as the $z$-score for that value, use the formula:

$$z = \frac{x - \mu}{\sigma}$$

With $x$, $\mu$ and $\sigma$ the same as above.

A second statistic synonymous with financial analysis is rate of return (ROR). It gives an analyst the percentile loss or gain that would be experienced on an investment relative to the amount of money initially invested. When used on each value of a series the rate of return better relates consecutive data points to each other making it a more appealing metric to use than the absolute price values. Return values offer normalisation not afforded by discrete prices and when simultaneously viewing multiple price series this allows relationships in data be seen over different series. Acceptable values for ROR are dependent on a traders own time scale of interest and what they consider to be acceptable levels of risk for their money. Given these personal preferences, along with the type of market being invested in there are a number of formulas available to calculate ROR. The two that this thesis deals with are arithmetic, or percentage, ROR and logarithmic or dividend ROR. The formula for the arithmetic ROR is as follows:

$$r_{\text{arith}} = \frac{p_t - p_{t-1}}{p_{t-1}}$$

Where $t$ is the time interval, $p$ is the current price and $p_{t-1}$ is the price one time interval ago. The same variables are used for the logarithmic ROR that calculates the continuously compounded return with the formula:

$$r_{\text{log}} = \ln\left(\frac{p_t}{p_{t-1}}\right)$$

For small returns, that typically occur more often in higher frequency data, the difference between arithmetic and logarithmic return is minimal and they can often be considered approximately equal. It is only as the ratio of initial value to current value grows that differences become more apparent.

In academic studies the type of ROR that is favoured is dividend return as this gives a sense of historical property to the series. For studies that require return rate over a
number of time intervals the benefit of continuously compounded rate can be highlighted by the simple example of an initial investment of $100 that undergoes a gain of 50% followed by a loss of -50%. The arithmetic ROR would result in $75 but the logarithmic result would remain at $100. In relation to this study, the benefit of using a logarithmic scale arises when representing the data graphically. As previously stated the points of change are of particular interest and a log scale heightens the variation between the peaks and trough of a series.

Due to the two categories of input data for this project, sentiment analysis polarities and records of transactions in financial markets, this is clearly an example of where multivariable analysis must be employed. Not only are there two types of variable their respective ranges are significantly different as well, oil contracts are typically in the range of $80 to $110 and sentiment polarities vary between negative and positive decimal values of +4 to -4. Coupled with that is the fact that the standard deviation for the financial input varies over time. Thus making it a heteroskedastic process. A characteristic of a heteroskedastic process is its ability to be extremely volatile. To better represent a heteroskedastic series it should be taken in sections that highlight both the areas that contain solely homoskedastic or heteroskedastic characteristics and the sections that show the transition between the two. The differences in a heteroskedastic and homoskedastic process is best represented in figure 2.3.1 below.

Figure 2.3.1: Heteroskedastic and homoskedastic processes
2.4 Visualisation

The realisation of Moore’s Law over the last forty years and its application in the growth of memory capacity has seen a tremendous increase in the sheer quantity of data available to financial analysts. There is no question that the process of data mining such quantities can only be done electronically and the conveyance of the results of data mining is an area where correct representation is fundamental to the analyst, or end user, attaining the full benefits of the computational power available (Keim, 2002) (Zhang & Zhou, 2004). In terms of quantity of data behind visible market events the iceberg analogy is relevant if a market transaction is considered to be the above water portion and the array of influencing factors the submerged 90%.

Spreadsheets have long been used to hold financial data and their graphing utilities are sufficient to provide an insight into the “big picture” of a large body of data, a readily available example is Microsoft excel (Brath & Peters, 2008). Excel’s graphing capability entails a ‘what you see is what you get’ representation of the selected data columns and as such the management of values to be graphed becomes difficult and time consuming as the desired level of analysis detail increases.

For this reason the manufacturers of financial software packages tailor the analysis capabilities to their own requirements or the envisaged requirements of potential users of the software.

As seen in the previous section outlining the analysis characteristics to be displayed by this project the visualisation requirements and dataset characteristics can be summarised in the following points:

- A multivariable high frequency dataset ordered by time.
- Individual data series with vastly different ranges.
- High frequency datasets can potentially contain thousands of data values.
- Individual series can be heteroskedastic in nature.
- The need to view small sections of the dataset is quite high.
- Ability to plot rate of return and deviations from mean should be included.

The graph desired by Professor Ahmad was specified to be in the form of a spider or radar graph. This type of graph can be seen in figure 2.4.1 below.
Spider charts are effective tools for comparing multiple series relating to various entities based on different characteristics. They are a useful way to generate a ‘feel’ for the data in question. To cater the above examples to this project's needs, the radial leg values marked ‘category’ in the figure above should be altered to denote the time scale and the example series can be used to display the data columns of the incoming spreadsheet. The choice of spider plot is further supported when the inherent limitations of the graph are considered. Spider graphs are used best for a low number of data points, as opposed to higher numbers. This is consistent with the need to allow the focusing on small sections of the dataset due to the heteroskedastic nature of financial data.

To solve the issue of large discrepancies in the range of respective series all data should be normalised between the values of zero and one. This will have the added bonus of emphasising points of change in a series and since polar sentiment analysis is used to identify information arrival, represented by changes in polarity as outlined in section 2.2, this is exactly what is needed.

To plot the rate of return and z-score values an option can be added to include these as a separate series in the graph, again furthering the illustration of points of change in the data.
2.4.1 Current Visualisation Techniques in Financial Analysis

A look at current methods of displaying financial data shows spreadsheets as the favoured format for holding datasets. The simplest aid to visualisation, shown in figure 2.4.1.1 below, is added by using in built functions of most spreadsheet platforms to perform numeric check on the data and alter the font colour or background colour of cells that are significantly higher or lower than the previous value. Additionally these values may also be extracted to a separate panel that only contains values of interest and their titles.

While this approach may be suited to analysing real time continuous data as it arrives, it is too complicated to be used in displaying trends in series. This task would be better performed by programs that generate separate graphs. Due to relentless nature of financial markets and the rate of information arrival a challenge in developing user interfaces is fitting all of the required data onto a single screen. Figure 2.4.1.2 below, shows a screen shot of a market analyst’s computer. Clearly the ability to monitor multiple sources simultaneously, whether spreadsheets or graphs is paramount.

Figure 0-2.4.1.1 :Example of a modified spreadsheet display to aid data visualisation
2.4.2 Visualisation Summary

The result of the visualisation research detailed in sections 2.4 and 2.4.1 enables the following concise features for the appearance of the developed program to be identified as desirable in a graphing component of a financial analysis software package:

- The option to graph should be a discrete feature in the interface of a larger software package.
- Higher levels of data manipulation are desired.
- The final graph produced should be in a separate window that can be scaled to thumbnail size when not required.
3 Formulation of Process Methodology

In creating a user application it is vital that before any method is theorised the requirements are tersely listed along with the expected user in figure put, if any, so that there is no ambiguity that may necessitate messy refactoring further into the project. In the case of this thesis the outlined deliverable is a computer program capable of allowing a user present it a file along with information regarding the elements of the file that are to be displayed graphically to better illustrate the behavior of each element graphed. The constraints on this are as follows:

- The user is intended to be an individual interested in the analysis of financial market behavior, particularly in relation to sentiment analysis. The program should create the user interface accordingly.
- The file type is in excel or comma separated variable (.csv) format.
- The data contained in the file must include series of sentiment polarities and stock market prices covering the same time period.
- The graph type to be generated is to be a spider, or radar plot.

With specifications defined the next step is to formulate a feasible, succinct approach that can be followed to complete this project. For this it is necessary to identify the separate processes involved in bringing it to fruition and then individually assess each component as its own task to be made functional as a stand alone system. This is fundamental to ensuring that the result will meet all the user requirements and be applicable to as broad a range of inputs as possible.

The sub processes initially identifiable are methods to:
- Take the user input for both file and desired output.
- Parse the file to correctly identify the plottable data and return the values of the elements selected for graphing.
- Realise the visualization of the data.

These three initial sub processes are then further divided, when possible, and the details of each is covered in the following sections. As appropriate, it is also good practice to review progress and address issues promptly.

3.1 Handling User Input

It becomes apparent quite rapidly that this sub process can be further reduced to a number of smaller functions. There are two separate occurrences of user input, that of the file selection and secondly the input of the choice of file elements to be graphed. This in turn presents a decision that must be made on how to allow the user to indicate the data they wish to plot.
Should the program require the user to have explicit knowledge of the contents of the file and manually enter an element to be plotted, or should the program assume the responsibility for reading the file and presenting the user with a finite list of plots that are available? The prior case reduces the amount of background computation required of the program but it presents an invitation for human error, generated by the user typing element names, to cause needless stalls in the functionality of the program. This is a frustrating characteristic of static, rigid programs that would be best avoided. A possible benefit of this approach could be that all user input can be taken once.

The alternative to this is for the program to take user input at two separate times. First the input of the file path and then have the program read the contents of the file *before* displaying a second input panel that presents the user with a list of the data elements that can be graphed. This method allows much more scope for the program to successfully detect any non graphable data and alert the user to the issue, thus preventing the user further exasperation. It also adheres to the specification that the program be able to handle any file, once in the correct format. An added factor is when background research on financial data characteristics is recalled; in particular the sheer quantity of data points present in datasets, these can be huge and as such, the likelihood for non graphable data is increased.

### 3.2 Extracting Data from the Input File

To populate the list of data elements that can be graphed it is necessary to understand that, as far as the end user is concerned, data is represented in a .csv file in much the same was as in a standard Microsoft Excel spreadsheet, in the form of a grid of rows and columns with each grid box containing one element. With this in mind it is reasonable for the program to assert three constraints of its own on the layout of the input file:

- The titles of the data elements are to appear in the first row of the file.
- There must be a column containing either date information or time information.
- The rows intended for graphing must contain decimal or integer values.

The title row will be used by the program to create the list of elements that the user may select to display on the graph. The date and time constraint is deemed reasonable as without a chronological reference a time frame could not be used for the graph, thus making the plot redundant.

Once these stipulations are met data extraction can be simplified one final time as being in one of the following processes:

- The title row identification and storage of names of the data columns.
- The time scale column(s) identification and storage.
• The identification and storage of data columns that may be plotted, which are simply any column that isn’t either date or time.

At this point in development a sense of the amount of raw data available to the program is realised and from research done, as outlined in chapter two, there are many powerful analysis techniques available to enhance the information gleaned from such data. It would be remiss not to make this additional analysis available to the user.

3.2.1 Additional Data Manipulation Options

The first addition that springs to mind is to add an option for the rate of return of a series. This will give the user a better visualization of how the series develops over time. The argument for its inclusion is further supported by the logarithmic nature of dividend returns which will make the identification of points of change to the data more apparent in a graph that potentially contains multiple series. In keeping with the effort to highlight relationships between multiple data points in a single series an option to plot the z-scores for each point should be added.

The second positive modification, again given the quantity of data that may be contained in the input file, is to offer the user a selection of time scales representative of time horizons used by markets traders (Lynch & Zumbach, 2003). For this project since sentiment analysis is a topic of interest it can be deemed sufficient to limit the highest frequency time horizon to minutes as the publication of text articles will seldom be recorded to the second. With minutes as the baseline, and default scale to be plotted, the time scale options then increase to hour, day, week and month. Although the last three mentioned do not fall under the category of high frequency data their inclusion enables the user to view the file data at both high frequency and at more traditional lower frequencies.

Finally, due to the potential size of the files, which may contain hundreds to thousands of rows, and the fact that the number of points on a single graph should be limited to preserve clarity, it would be unreasonable to expect the user to always wish to have their graph begin at the start of the file. Instead an option should be made available to allow the user enter a date as a starting point of the graph. Note, a time value is not sufficient as for large datasets repetition of time values is to be expected. The most efficient method to have the user enter a date is as text in explicit format with the program clearly stating the required format, eg ‘dd/mm/yyyy’.
3.2.2 Secondary User Input

With the parsing of the input file complete it is possible to display the user interface component where the selection of data elements to be graphed takes place along with the optional extras of plotting returns and z-score values or setting a start date for the graph. The selection made by the user in this section will not be handled by the same process as the previous input of the file path but by the graphing process itself.

3.3 Generating Graphic Display

The function of the final process is to combine the previous two, user input handling and the data extracted from the file, to apply graphic contextualization to produce an output image that meets the requirements of the user. Essentially this process entails organising and giving semblance to the stores of data harvested from the input file in conjunction with the wishes of the user for a particular compilation of the total process.

To correctly generate a plot of each individual series to be graphed the data can be processed in the following order:

- Retrieve the title of the data elements selected by the user to be graphed.
- Match the title with its set of data points. In excel these are the values contained in the column directly beneath the title.
- Identify the time or date column and set the time scale to that selected by the user.
- If any starting date has been specified by the user iterate through the data column to find the first occurrence of the date.
- Select the segment of data points to be added to the graph so as to give enough of an overview of the data trends without compromising the clarity of the graph produced by having too many data points present.

Making the above steps generic for any columns to be plotted preserves the original stipulation, made in the outline of the program desired, that it be applicable to as broad a selection of data as possible.

For the particular programming language chosen by the developer it may be possible that there is an existing body of code available which generates particular types of graph. Regardless of whether or not this is the case the vast majority of graphing methods, even a custom method created by the developer, will need the following core inputs.

- Graph title.
- The name(s) of the series to be plotted.
- Sufficient labeling for the various axis of the graph.
- Actual data points to be plotted.
These four base values should suffice to allow the generation of a basic graph and additional extras to aid and enhance the ease of interpreting the graph can be added as deemed necessary. Examples of extras are:

- Gridlines in the background to help with the comparison of the locations of separate data points.
- Joining the data points of a single series with an accurate plot line or line of best fit.
- Choosing to colour each series differently for esthetic reasons.

The client for the program should also be involved in this process of choosing which features to add.

3.3.1 Post Visualisation Options

Similar to section 3.2.1 there is an opportunity present to review the program and its functions that should not be overlooked. The research from chapter two should be recalled to identify any additional features that could be added. Since section 3.2.1 introduced the ability to start the graph from a particular date it is clear that the belief that the user may wish to iterate through a variable number of data points for a particular graph is a factor of the program functionality. An intuitive addition to make then is to allow the user to generate multiple graphs of consecutive sections of data points without having to re-run the program and go through the steps of selecting a file, the data elements to plot, the scale to use and the starting date of the graph. This implies that the user will reference between graphs so the generated graphs should appear in individual windows.

Including this modification insinuates that the expected user is interested in the analysis and comparison of financial data. Continuing in this train of thought it becomes apparent that as such the user may wish to reference the graph they plot on multiple occasions and requiring the user to input identical data multiple times is obtuse. To avoid this issue simply add the option to save the generated graph as an image file for future reference. One consideration to make for the saving option is the format of file used which in turn specifies the type of image compression used. The two most frequently encountered are JPEG and PNG. The difference between the two is that JPEG employs lossy compression while PNG files use lossless compression techniques. As the name suggests, lossy compression chooses to permanently ignore, or loose, elements of the data on the assumption that humans can make sense of images with more than 70% of data removed. Lossless compression on the other hand retains all image data and is favored when it is deemed unacceptable to remove any data that may be needed in the future such as in medical fields (Said & Pearlman, 1996). Since this program’s graph is to be interpreted visually the choice of lossless compression would seem appropriate.
3.4 Final Review of Process Methodology

With the body of the necessary methodology completed a thorough review of the outlined process should be carried out. This should be done from the perspective of the planned program user and as in the previous section the client should be involved in this process. Now is the time to ensure that all information needed by the user to understand the inputs they are required to provide is present in the user interface and that any possible constraints or areas of confusion are removed. It is advisable to have a third party, that has no prior knowledge of the program use, and is without any influence from the designer to test the program. Any difficulties encountered by the test user(s) should be addressed, and depending on the thoroughness of development shown by the developer and the collaboration with the client during development, there ideally should not be a need for major refactoring at this stage of the process. Instead it should be sufficient to add prompts which alert the user if they have entered inputs that do not align with the expected range of inputs. For example if a file selected is not in excel or .csv format, if there are no data columns present in the file that are suitable for graphing, i.e that are not of decimal or integer values, if there is no date or time column present in the graph etc. simple additions such as this will ensure smooth running of the program and negate the need to have the user rerun the program because of an unexplained error.

3.5 Process Summary

In formulating the above process the importance of producing as dynamic a process as possible was an important factor in each decision made. The result is that the finished methodology can reliably be applied to the development of any graphing program in the broad field of financial analysis. Indeed it is not overly ambitious to say that the process, with the possible exception of the ‘review’ stages, could be used to successfully produce the core elements of any basic graphing utility. In the words of the late Steve Jobs, the beauty of a device, or program, as seen from the users perspective is that “it just works”. Sadly this is often a point overlooked by program developers as they become so entrenched in the backend functionality of the code that its capabilities, no matter how great, are not readily accessible to the user.
4 Implementation of the Developed Program

The driving factor of this project was the development of a graphing utility for financial data. This was the governing element in the areas of research selected and the decisions made in the actual development of the program were based on the findings of said research. These decisions are introduced in the following sections.

4.1 Language Choice and Libraries Used

The programming language chosen for implementation was Java. Critical factors in its selection were the object orientated nature of Java making it ideal for use in the development of a dynamic program. There are a number of object orientated languages available but the added bonus of the wealth of Java API's was the deciding factor. These open source libraries of code further enhance the dynamic capability of the program by removing the need to implement complicated algorithms for program functions that are often required in many different programs such as parsing a certain file type or generating a particular type of graph. The API's used were:

- Apache POI – Java API for Microsoft Office documents, in particular POI-XSSF used with Excel documents (XSSF).
- JFreeChart 1.0.14 – a library used to in the generation of a graph and its saving as an image (jfreechart).
- JCommon 1.0.17 – a library used for certain maths methods in JFreeChart (jcommon).
- Joda Time 2.0 – a library for identifying and handling data and time inputs and outputs of different types (joda).

Without the API's the amount of programming required to read an excel input file would have been overwhelming. In fact some coding was carried out on date handling prior to the inclusion of Joda Time that quickly proved itself not to be conducive to a dynamic program. Fortunately the introduction of the API alleviated this and provided the same results which further reinforced confidence in the decision to import API's.

4.2 Program Layout

The graphing utility developed over the course of this project implemented the processes described in chapter three, the variable not accounted for in chapter three is the language to be used. This is a point that is subject to each individual developer’s personal preference and the program layout, class path etc, will vary depending on the language selected. As Java was selected in this instance along with its existing libraries the layout of the program was as follows:
The main classes in the program are the file reader, the ‘graphing options’ GUI class, and the date manager class. These perform the unseen computation that make the user interfaces perform successfully and shall be described in their own subsections. The remaining classes, the initial file selection GUI, analysis and final graphing class are less detailed. The final graphing class is responsible for calling the methods from the analysis, date manager, file reader and graph options class so will be better explained after the basic function these classes has been detailed. The analysis class is a simple class containing methods to calculate numerical computations for the standard deviation, z-scores and rate of returns. The formulae for these appeared in section 2.3.

The file selection class, seen in figure 4.2.2, is a two button user interface with an instruction to select a file from storage to use. The ‘Select File’ button triggers the built in java method of ‘Filechooser’ that opens a new window, see figure 4.2.3, that by default shows the home folder of the computer running the program. Once a file is chosen in this window the ‘Next Step’ button becomes selectable and it brings the user to the main GUI, that of the graph options interface.
4.3 Data Handling – File Reader

This class implements the process outlined in section 3.2, the parsing of the selected file and harvesting of all necessary data from it. It runs during the transition from the file selection GUI to the graphing options GUI and without it the user would not be presented with any data elements to graph. Since there are two possible input types, excel or .csv there must be two types of file readers written. Fortunately the readers are quite similar with excel in fact being a particular type of a csv file that has been specified to Microsoft’s design. This meant that the steps involved in reading either file types were the same. Essentially an ‘input spreadsheet’ object was created that contains the data points that can be referenced by row and or by column. Once the two reader methods were returning the same data the rest of the functions in the class could be developed regardless of whether an excel file or csv file was being used.

The referencing by row and column enables the reader to isolate the first row and store the column titles. Of particular interest is the column index(es) of date and or time that are integral in setting a scale for the graph. Other checks performed by iterating through each column ensure that the list of column titles that the user can select from only contain columns that are comprised of decimal or integer values.
Lastly, the reader checks to see if there is a title relating to the close price of the financial data in the graph as this will be the column that is used when calculating the rates of return and z-score. If there is not a close price column present the graphing option GUI will not display an option to plot these additional series.

4.4 Data Handling - Date Management

The data manager class makes full use of the Joda Time library to format the date and time values of the input file. Since the program is required to be dynamic the decision had to be made on how many constraints to place on the input file. One possible constraint that was originally considered was specifying the format of the date and time values in the file, for example that it had to be dd/mm/yyyy. However this was deemed too restrictive so the program was made capable of processing both the American and European formats with a number of smaller extras. In both cases the provision was made that the placement of year values could be reversed, for example, for American style date input the allowable forms are mm/dd/yyyy or yyyy/mm/dd. The processing of a time column did not need any clauses as its format could reliably be expected to be in twenty four hour format, as used in the recording of financial market transactions. With the date and time parsed they could be combined to create a timestamp value for each series value.

Once the timestamp had been identified the methods to set the scale of a series by minute, week, hour, day or month were developed. Again the Joda Time library was invaluable as it provided functions to view a timestamps as ‘minute of hour’, ‘hour of day’, ‘day of week’ etc that saved much computational power that would have been required in the processing of String data types.

4.5 Data Handling – ‘Graph Options’ User Interface

This is the largest class of the project by a considerable margin and the processes it is involved in have been seen in sections 3.1, 3.2.1, 3.2.2 and 3.3.1. It is very much the link pin of the program. The first job of the class is to generate the ‘Graph Options’ GUI, which can be seen in figure 4.5.1. Each panel of the GUI was hand coded using GroupLayout and the panels then arranged using a combination of GridLayout and BoxLayout.
When the GUI first appears, the section of the GUI below the ‘Plot’ button is not displayed. This is due to the fact that the option to cycle through a data series can only be done when there is a starting point to increment or decrement from. In this case the initial display of the graph begins at the start of the file so it is not possible to cycle backwards therefore the ‘Back’ button is not made selectable. Similarly a graph cannot be saved as an image before the graph has been generated. To cater for an input file of any size the display of columns that can be graphed is contained within a scroll pane. This measure is taken to prevent the GUI being dominated by numerous checkboxes of each column title.

Each potential source of user input, the checkboxes, drop down menu, text field and buttons all have their own action listener that will initialise parameters to be provided to the final graphing class. In the case of the checkboxes one listener is used for all the checkboxes and each time a box is selected/deselected the title is added/removed to a list of series to be graphed.
4.6 Generating Graph

The final class in the program sees the many methods developed throughout all the other classes working in the sequence, detailed in section 3.3, to give the user the data visualisation required. The end result is seen below in figure 4.6.1, the file name is displayed at the top of the graph, the radial ‘legs’ give the time axis. This could be compared to the x-axis on a traditional x-y graph. Through a process of trial and error the number of data points plotted per graph was set to twenty three. This is simply the number of points that was deemed to be the maximum number that was able to ensure clear visibility of each data point, particularly are lower values.

Figure 4.6.1: Generated graph
5 Conclusion

5.1 Project Review

This project has seen me tasked with designing and developing a graphing application for spreadsheets of financial data. The course of work has included research into financial analysis, data visualization and sentiment analysis.

From papers such as (Engle & Russell, Analysis of High Frequency Financial Data, 2004) and (Dacorogna, Gencay, Muller, Olsen, & Pricet, 2001) I developed a fundamental grounding in the characteristics of high frequency data.

Similarly, from the works of (TETLOCK, 2007), (Devitt & Ahmad, 2007) and (Wilson, Wiebe, & Hoffman, 2005) in the area of sentiment analysis. This was an interesting topic to research as the general understanding that language is an extremely emotive tool is an idea I have always been aware of but it was not an area that I had previously heard of being studied quantitatively.

The final area of research was that of visualising the data, an area I had slightly more background in. This spanned the act of researching current methods used in financial settings and the user experience from interaction with a user interface. I was fortunate enough to attend a graduate open day at the Dublin branch of Susquehanna International Group, one of the largest privately held financial institutions in the world and saw firsthand the interfaces of software packages. I was also able to talk to both the traders using them and the programmers responsible for some of their design at a meet and greet session afterwards. This was a key factor in my decision to develop the program in three separated display windows. A second reference that I had come across, before under taking the project, was the book ‘Design for Hackers’ by David Kadavy (Kadavy, 2011). While the name may be somewhat misleading to those unfamiliar with it, the book takes an interesting approach to reverse engineering successful design layout of computer interfaces in order to establish the fundamental components of a ‘nice’ visual appearance.

On reflection, I am satisfied with the developed program, that it has been built on the research I conducted to produce a java application that can reliably be used in an academic or profession environment to generate a graph that enhances the users understanding of the plotted series.
5.2 Comments and Future Work

At all times I tried to be mindful of the fact that the delivered program would be used by an independent user, reasonably computer literate, and that the ease of navigation through the steps involved should be as intuitive as possible for the user. The analysis and manipulation of the graph possible is an area that potentially can be expanded and built upon indefinitely. There are numerous papers on statistical methods that could be added but an important point I felt was the balance that must be struck between time spent by the user in specifying additional program options and the quality of additional insight gained from these options is action.

A more likely area of future work is that the program could be incorporated into a larger financial analysis software package, developed under Professor Ahmad, and as such it would be required to provide a concise graphing service. If this occurs then an immediate addition could be to offer other types of graph output. In particular, graphs that can display a larger number of data points in a single plot since current computer power and data storage space have the amount of financial data available for analysis at an all time high with no signs of decreasing.

Finally a more ambitious body of future work would be augmenting the program to explore the possibility of visualising the ratio of two commodities, eg gold and oil, with the intention of providing visual aids to the theory that there is some correlation between the volatility of metals and oil which has been highlighted over the course of the ongoing uncertainty in Iraq and more recently the Arab Spring (Hammoudeh & Yuan, 2008).

This would be a good embodiment of the observation by John Tukey;

“The greatest value of a picture is when it forces us to notice what we never expected to see.”
Bibliography


