VISUAL F# STOCK MARKET ANALYSIS

USING THE .NET FRAMEWORK

Glassner, Cullen cglassne@ucsc.edu

March 16, 2014

The stock exchange is a very complicated environment, rich with time-relative data. Brokers often excuse analysis tools made by software engineers as inaccurate, stating human intuition plays a larger role in the prediction of stocks' values. Many tools have been created using more complex models and larger data sets than the tool I have created. This project is not meant to prove any correctness, rather to show the applications of functional programming, specifically the benefits of F#, when looking at large sets of data and running complex mathematical models on this data.

In the initial design of my project I was going to have an F# analytics tool back-end and a C# WPF front-end to demonstrate some benefits of using the .NET framework. Upon further investigation I learned F# can in face run C# code and libraries! As such, I was able to contain all the code in one F# program.

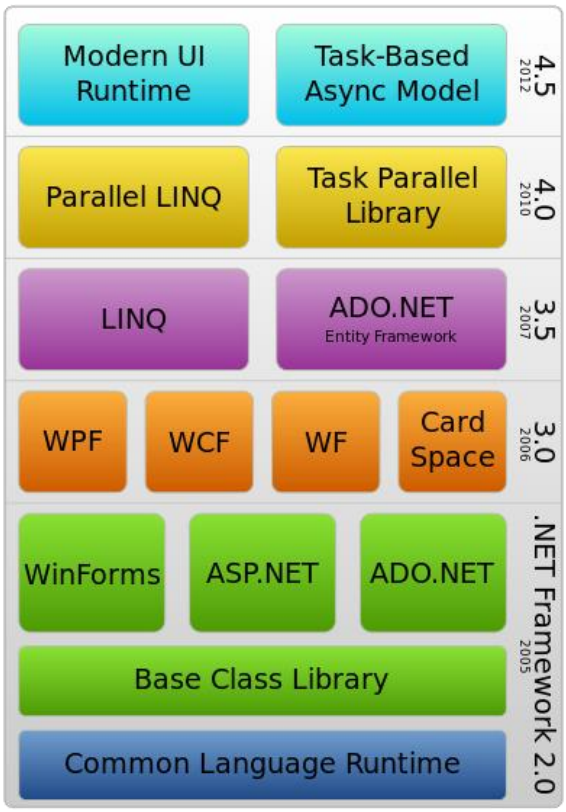
My project was created in Visual Studio 2013, however F# runs on other platforms and can actually be used to develop JavaScript and GPU code. This project would require some migration to run on these tools, so the instructions listed to run the program will assume you have a working copy of Visual Studio 2013 with F# libraries downloaded.

Code available on github

AN INTRODUCTION TO F#

This project is not meant to be an accurate predictor of trends in the New York Stock Exchange. Instead, it is meant to show some of the benefits of functional programming for such an application, and more so the benefits of F#, belonging to a larger family of languages (the .NET framework) that can easily work together and in fact run the code of another language. This gives some powerful benefits such as the convenient WPF and drawing libraries of C# running in a functional language. We also have access to objects, through C# implementations (with a few sematic changes).

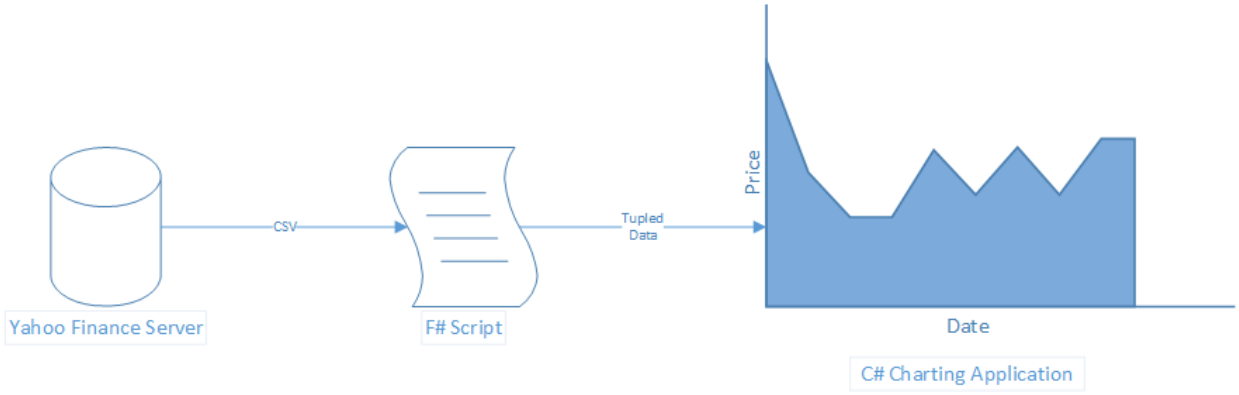
F# was developed by Microsoft and the F# Software Foundation as a strongly-typed functional language. The language also shows influences of imperative programming and object-oriented programming. F# was created in 2005 with its first stable release coming out Christmas of 2013. F# is also one of the members of the .NET framework. This framework executes code in a software environment called the Common Language Runtime. The Base Class Library sits above the CLR allowing all the .NET languages to share code, allowing for language interoperability. Below is a stack diagram of the .NET framework.



The .NET Framework Stack

ARCHITECTURE

Yahoo hosts a finance server to download stock market historical data such as opening and closing cost of a given stock, which can be downloaded via a URL request to the server. My F# script writes the URL request dynamically and downloads and parses the data set into an F# Sequence. This data can be interpreted and analyzed at this stage with F# code or passed on to the C# application for charting and display of predictions and metrics. Below is a diagram of the flow of information in this system.



Upon further coding and research I have been able to run the C# charting code in my F# script, showing off the interoperability of the .NET framework family of languages.

INTEREST OF STOCK MARKET ANALYSIS

Looking at large sets of data provides large possibilities in creating statistical models and running algorithms to analyze the data. The stock market is a great environment to test such a system on, with very well catalogue data in many different areas with many external factors playing in the change of data. This tool does not attempt to predict the stock worth or probability of a good trade as there are already tools that do such work. Instead it showcases examples for the usefulness of a functional language in such a data-rich environment. In the stock exchange it is vital to make any decisions faster than everyone else. This analysis must be as fast as possible. F# allows for parallel computation due to its lazy functional nature. This makes it great for high stakes real-time systems.

Two interesting analysis techniques are the relative strength index (RSI) and moving averages. RSI is used to show momentum and trends of a stock. It is also said to show when a stock is being overbought or oversold when the RSI is +/- 20 away (respectively) from the median 50 that the RSI oscillates around. This is actually one of the key statistical models used in many of the algorithms currently implemented in such programs. Moving averages look at an average

over a period of time. This allows a trend to be seen without all the noise of the daily stock value. It is great for using function predictions once all the calculations are completed and graphed.

Once this data is presented to an analyst (or someone at home) many options are available to make some profit. Pair trading and short selling are two examples where this information may be useful. Pair trading consist of waiting for stocks that have historically been closely coupled to diverge. At this point selling the stock that rose and buying some stock that dropped would likely be beneficial as they are probable to re-converge again. If you believe a company's value will soon drop (such as right after an IPO) you can "short sell" the stocks, where you borrow the stock from a broker just to sell the stock on the market. You get current market price for this stock and have to pay your broker at some predetermined time later, hopefully when the stock is less valuable thus turning a profit.

CONCLUSION

Functional languages have many applications in finance. F# is no exception as it has many more capabilities than other popular functional languages, leveraging the object-oriented and imperative aspects of the .NET framework when necessary. It has proven to be as fast analytical tool to run computation on large data sets and could be done much faster if I had been able to research parallel programming in F#.

REFERENCES

Herlemont, D. (2003). *Pairs Trading, Convergence Trading, Cointegration*. Retrieved from YATS Finances & Technologies: http://www.yats.com/doc/cointegration-en.html

Investopedia. (n.d.). *Short Selling: What Is Short Selling?* Retrieved from Investopedia: http://www.investopedia.com/university/shortselling/shortselling1.asp

Microsoft. (2014). *Overview of the .NET Framework*. Retrieved from MSDN: http://msdn.microsoft.com/en-us/library/zw4w595w.aspx

The F# Software Foundation. (2014). *The F# Software Foundation*. Retrieved from The F# Software Foundation: http://fsharp.org/