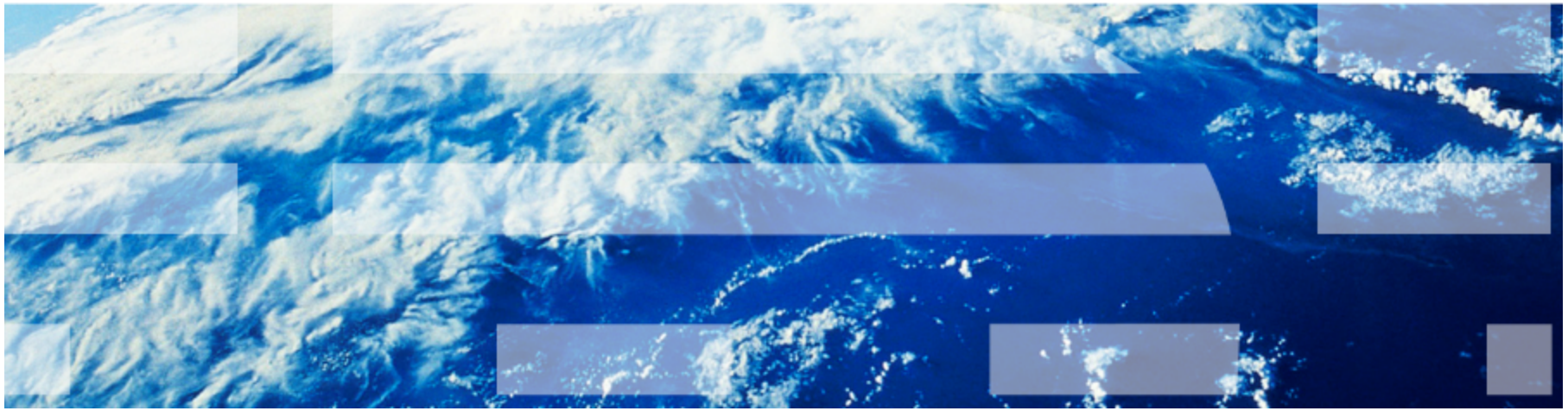
E6893 Big Data Analytics:

Currency Trend Analyzer

Tim Paine – tkp2108
Mark Aligbe – ma2799



December 22, 2014

Presentation Overview

- Project Background
- Overview
- Trend Analysis and Visualization
- Price Prediction with Mahout
- Future Work

Presentation Overview

- Project Background
- Overview
- Trend Analysis and Visualization
- Price Prediction with Mahout
- Future Work

Project Background - Bloomberg

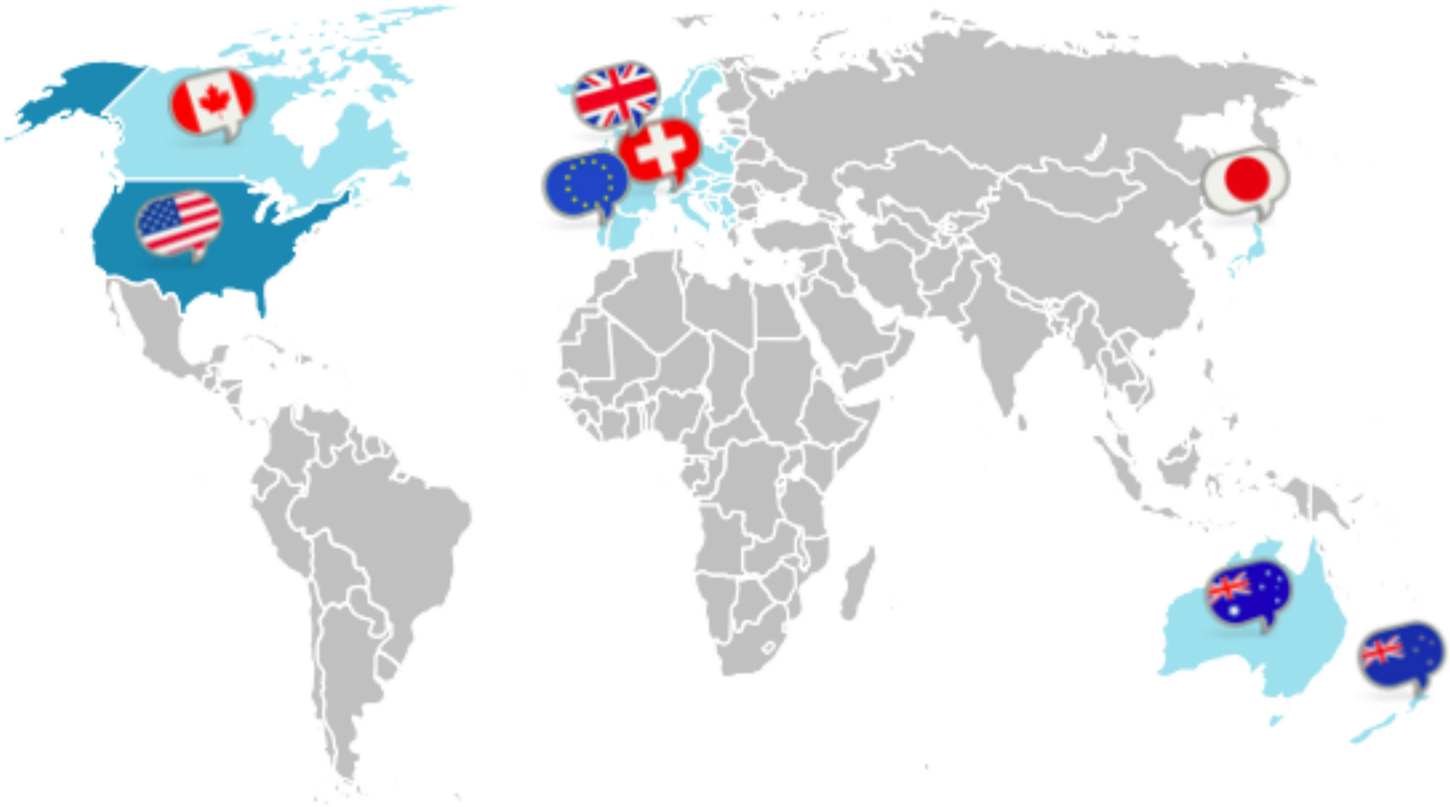
- Stock trends are well analyzed by tools like Bloomberg

Project Background – Currency Markets

- Every day, price data is generated at milliseconds granularity (30,000-60,000 new data points per day)
- Currency Markets can be volatile
- Decentralized (no exchange, all data comes from brokers)
- Not well suited for Bloomberg (no “fundamentals”, analysts do not rate currencies, only useful for technical indicators)
- Trading can occur around the clock (no market hours)

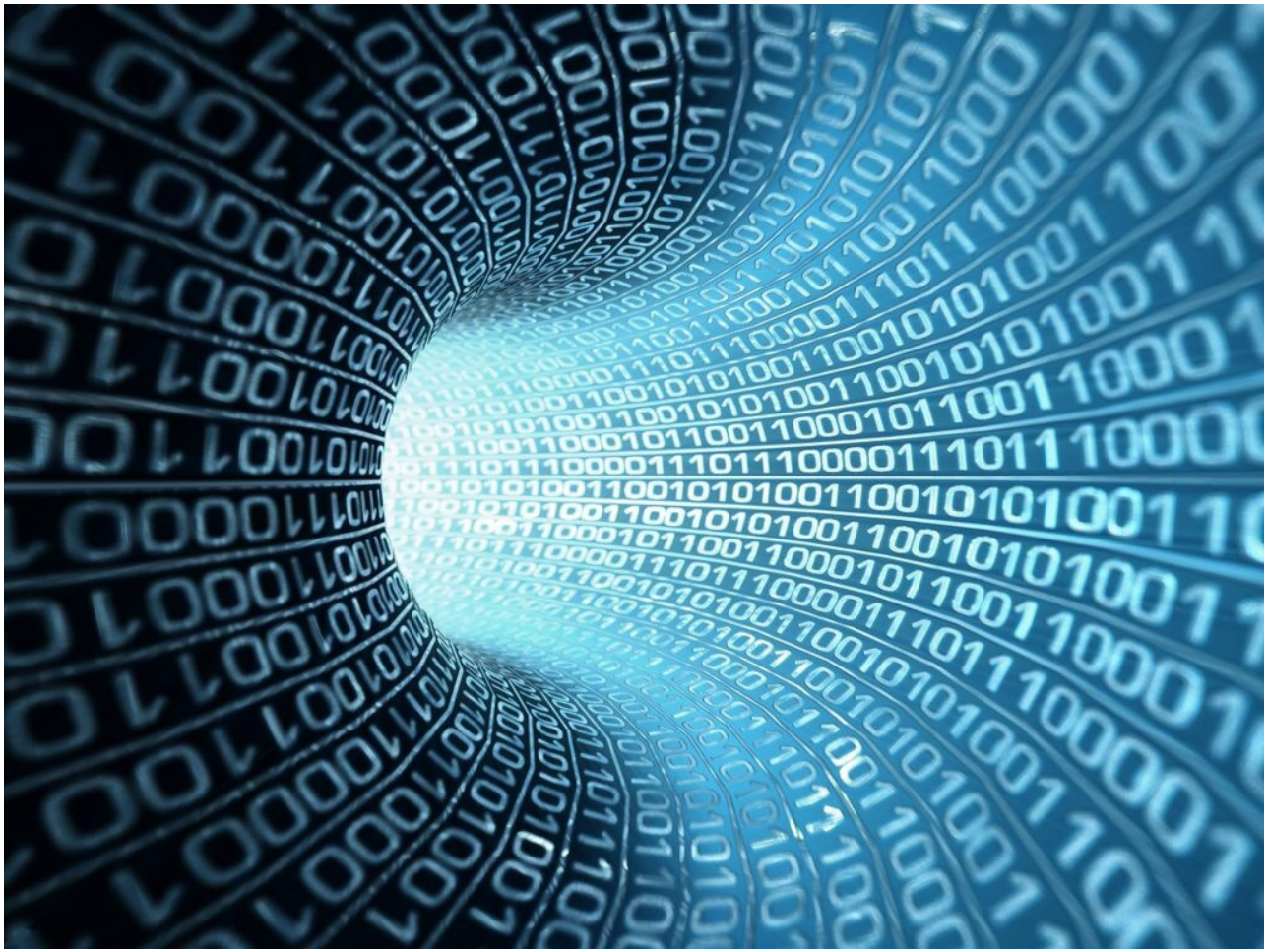
Project Background – Currency Markets

- Focusing on 7 major USD based currency markets
- EURUSD, USDJPY, USDCHF, AUDUSD, USDCAD, NZDUSD, GBPUSD,



Project Background

- Need a tool to manage data streams, process data, visualize data, and analyze data
- On the order of 1,000,000 data points per day



Project Background - Summary

- Every day, price data is generated at milliseconds granularity (30,000-60,000 new data points per day)
- Tools like Bloomberg do not target currency markets, and so are optimal for analyzing them
- Traders interested in currency markets need a tool to provide them trend information in order to exploit technical indicators when executing trades
- Big Data Techniques can help process and identify trends, and provide insights into the volatile currency markets
- Need to stream, process, visualize, and analyze large quantities of data

Presentation Overview

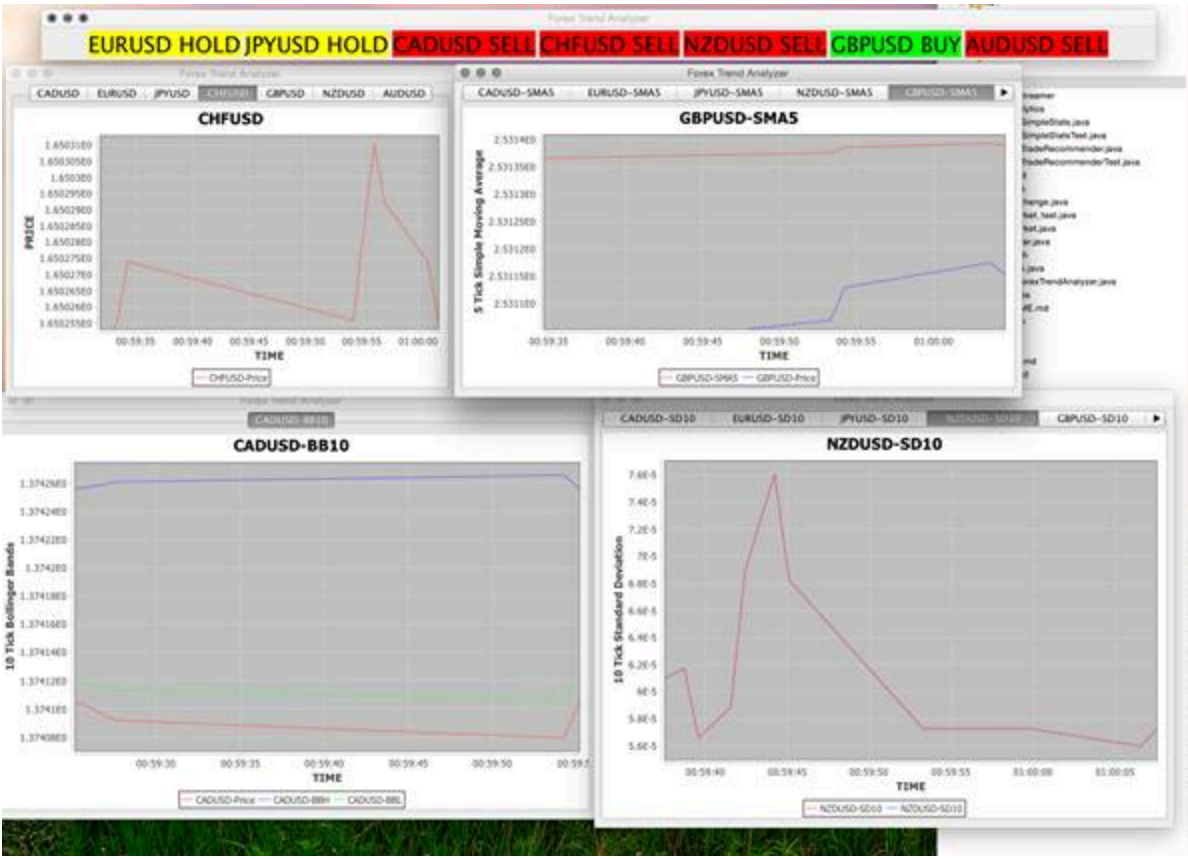
- Project Background
- Overview
- Trend Analysis and Visualization
- Price Prediction with Mahout
- Future Work

Project Overview

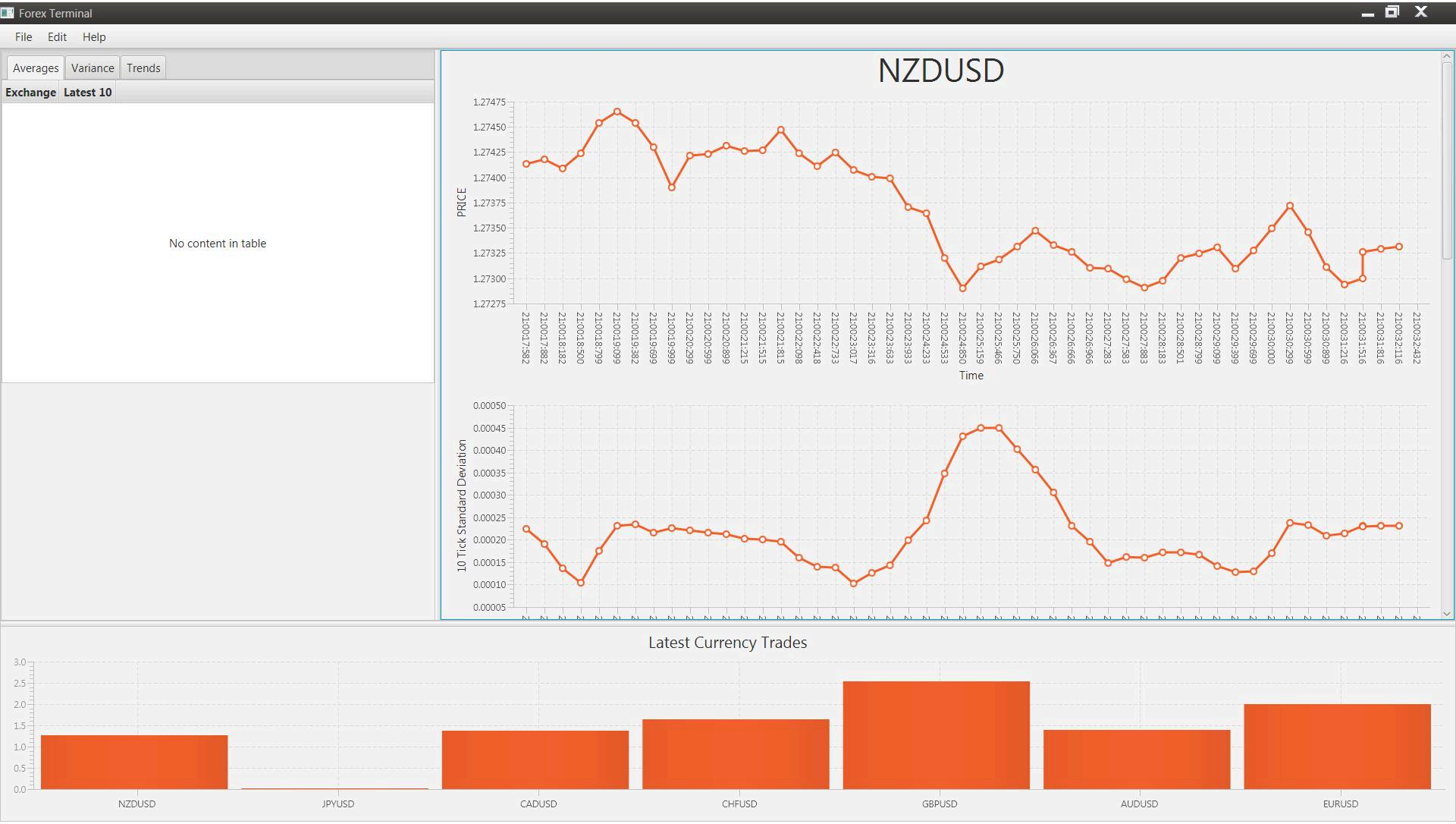
- The Currency Trend Analyzer performs these 4 tasks
- Aggregate Data streams and feed them to a processing engine
- Separate the data into currencies, insert into time series data structures
- Visualize the data in time series charts
- Collect and display metrics about the data
- Try to make predictions from the data (price movement)
- Provide this all in a bundled application with GUI

An Open-source Bloomberg terminal for currency markets

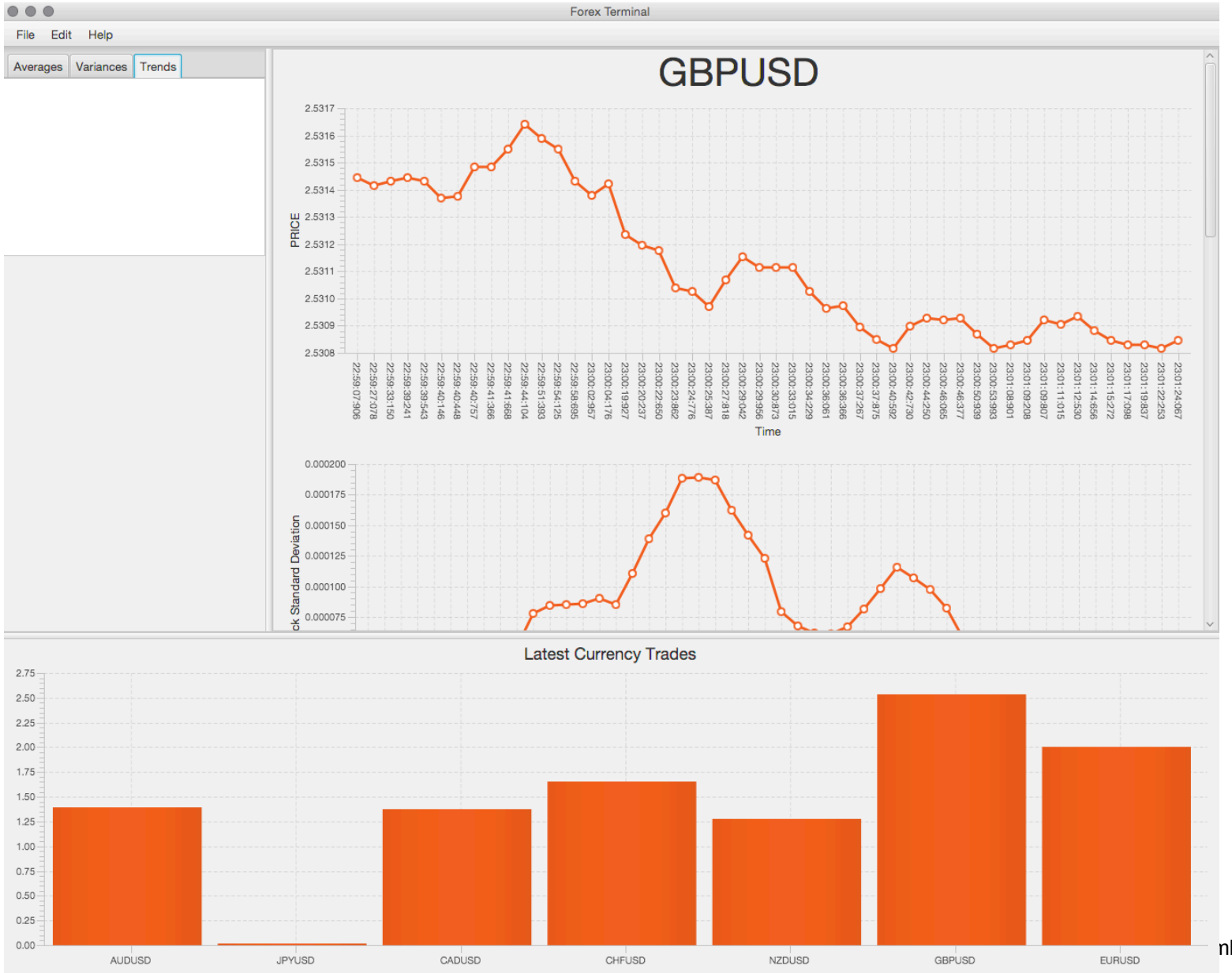
V1: Java Swing UI



V2: JavaFX UI



V2: JavaFX UI


Forex Terminal
File Edit Help
Averages Variances Trends
GBPUSD
PRICE
Time
Standard Deviation
Latest Currency Trades
AUDUSD
JPYUSD
CADUSD
CHFUSD
NZDUSD
GBPUSD
EURUSD

Presentation Overview

- Project Background
- Overview
- Trend Analysis and Visualization
- Price Prediction with Mahout
- Future Work

Data Aggregation

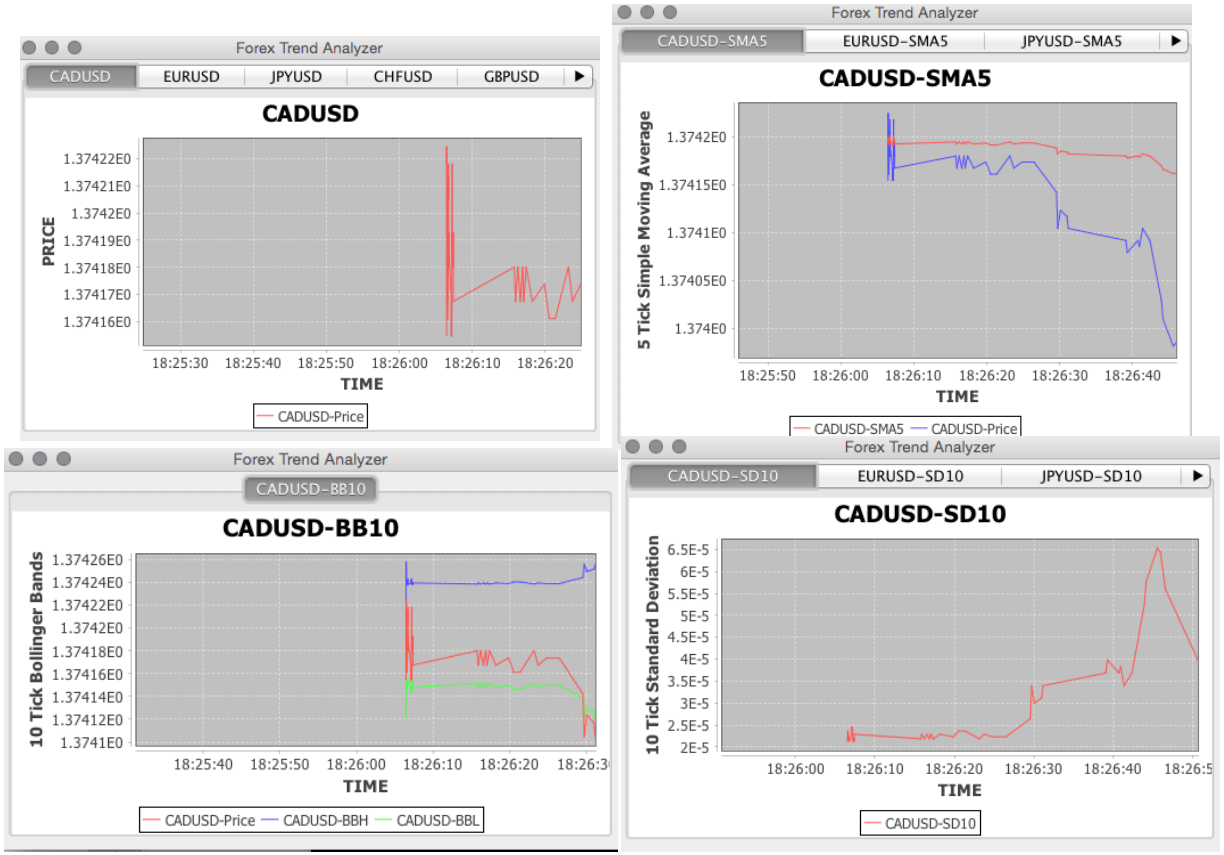
- Data feeds are connected to Feed Objects
- Currently fed from CSV files, but could be connected to broker APIs
- Feeds are aggregated by a Timer module which aggregates feed data into time-steps
- i.e. feeds a block of data every 5 seconds
- Milliseconds granularity (can feed data as fast as every millisecond/whatever the processing power of the server can provide)
- Timer module sends data to Market Module, which holds the most up to date data for all of the currency pairs
- Market feeds ForexTrendAnalyzer main method

```java
/*
package data_streamer.feed;

import java.io.IOException;

/**
 * The Class Feed.
 */
@SuppressWarnings("unchecked")
public class Feed extends TimeSeries<Dictionary<String, String>>
{

    /**
     * Instantiates a new feed.
     *
     * @param filename the filename
     */
    public Feed(String filename, String p)
    {
        super(filename, p);
    }

    /**
     * Load feed from filename.
     *
     * @return true, if successful (file exists), false otherwise
     * @throws IOException
     */
    public void loadFeed() throws IOException
    {
        // System.out.println(path);
        q = File.import_csv(path);
    }

    /* pop and return top element */
    /* (non-Javadoc)
     * @see market.feed.TimeSeries#pop() */
    @Override
    public Dictionary<String, String> pop()
    {
        return q.remove(0);
    }

    @Override
    public ArrayList<Dictionary<String, String>> pop(long t) throws ParseException
    {
        ArrayList<Dictionary<String, String>> ret = new ArrayList<Dictionary<String, String>>();
        if (q.size() > 0)
            // System.out.println("sim time:\t" + t);
```

```java
/*
package data_streamer.feed;

import java.text.ParseException;

/**
 * The Class TimeSeries.
 *
 * @param <T> the generic type
 */
public abstract class TimeSeries<T> implements Queue<Object>
{

    /** feed name */
    protected String                            name;
    /** path to file */
    protected String                            path;

    /** convert yyyyMMdd hhmmssfff to long */
    protected SimpleDateFormat                  f;

    /** time series */
    protected ArrayList<Dictionary<String, String>> q;

    /**
     * Instantiates a new time series.
     *
     * @param filename the filename
     */
    public TimeSeries(String filename, String p)
    {
        name = filename;
        path = p;
        q = new ArrayList<Dictionary<String, String>>();
        f = new SimpleDateFormat("yyyyMMdd hhmmssSSS");
    }

    public String getPath()
    {
        return path;
    }

    /**
     * pop and return top element
     *
     * @return the map
     */
    public abstract Dictionary<String, String> pop();
```

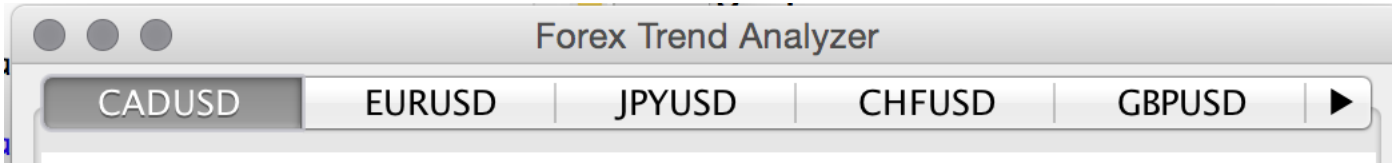
Data Visualization

- Time Series are visualized with updating graphs
- As new data becomes available, graphs dynamically adjust to accommodate a certain window (i.e. 30 seconds) of data
- Aim to provide Bloomberg-like windowed graphs for the major metrics



Data Metrics

- Provide several major technical indicators in an extensible framework
- Simple Moving Averages, Standard Deviations, Bollinger Bands, etc
- Display for all 7 currency pairs in tabbed windows
- Extensible, can add more indicators (RSI, Average crossing, etc) easily

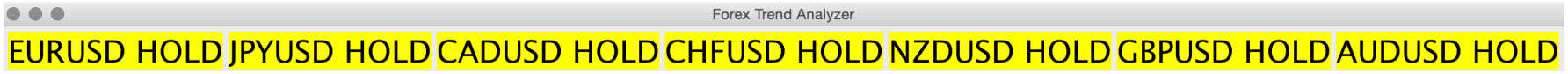


Presentation Overview

- Project Background
- Overview
- Trend Analysis and Visualization
- Price Prediction with Mahout
- Future Work

Price Prediction

- Leverage Mahout's Machine learning libraries to analyze the data
- Given some information about recent prices, generate some insight into whether the price will move up (buy), down (sell), or stay about the same (hold). \
- Detailed algorithm description given in our report (and at the end of the presentation)
- Simple status bar tells user whether a given currency pair is a BUY, a SELL, or a HOLD



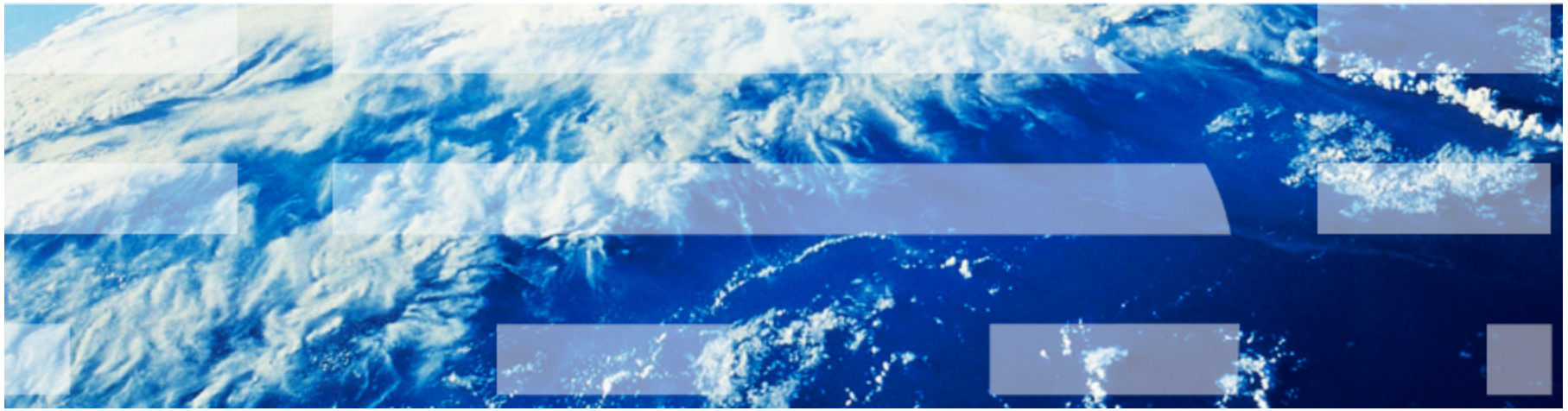
Presentation Overview

- Project Background
- Overview
- Trend Analysis and Visualization
- Price Prediction with Mahout
- Future Work

Future Work

- This project is very extensible and modular, able to expand beyond currency markets to any high granularity price based market
- Prime target for open sourcing
- More metrics to add, more currency pairs to add, more trends to identify
- User interface improvements and enhancements, porting to a web UI
- Better and more accurate price prediction (Support Vector Machine based prediction, regression based prediction)
- Scale-up speed by leveraging Hadoop – because there is so much data streaming in real time, at a certain point, we will need Hadoop to accelerate our computations

Thank you for your attention



December 22, 2014

Price Prediction

- The algorithm takes in two lists of trades in a market; the recent history of the market (e.g. the test set), that you want it to decide on, and the entire history of the market, sans the test set (e.g. the train set)
- It computes scores of ranges of trades. The score is the sum of the difference between the ith item and the first item (i does not include the first item), multiplied by the r2 of the least linear regression. of the range
- The second task is to normalize all the scores, and feed them to Mahout for processing
- For every two scores: we compute the percent difference between the scores. A larger percent difference therefore means that two ranges are more dissimilar. We then subtract this score from -1 (1) for negative (positive) results, which gives us the exact opposite semantic.
- This gives us n^2 scores, as each range has been correlated with every other range
- We give this to a Mahout ItemSimilarity, which then considers the similarity matrix of all ranges.

Price Prediction

- We then ask Mahout to score the similarity of all ranges to the test range. We sort by most similar, and only filter results that are trending in the same direction
- Once we have the most similar ranges, we score a small range of the market ahead of that range, and see which direction the market went in.
- If, for a majority of those ranges, the market went up, we say SELL (you're going to lose money otherwise)!
- If, for a majority of those ranges, the market went down, we say BUY (you're going to be rich soon)!
- Otherwise, just hold (the market is too unpredictable in similar scenarios).
- To decide how long to maintain a decision, we continue evaluating the future market, until we run out of history (we reach the present), or the trend is broken.
- We maintain how long each similar range lasted for, average their durations, and return that as our decision.