Downloading Stock Prices: Java

This document shows how to download stock prices from Yahoo Finance.

I created a java class StockDownloader that is designed to take stock symbol, start date and stop date and return arrays of dates, opens, highs, lows, closes, volumes, and adj-closes. StockDownloader dynamically builds URL based on stock symbol, start and stop dates for yahoo finance stock history URL. I used Scanner to read input stream line by line and parsed each entry (separated by comma) to construct arrays of data.

We need one package and two .java files for this to work.

Package: Stocks

Class: StockDownloader.java

Class: Stocks.java

Downloading Stock Prices: Java

```
package stocks;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.net.URL;
import java.net.URLConnection;
import java.util.Scanner;

/**
*
* @author Hochan
*/
public class StockDownloader {
	public static final int DATE = 0;
	public static final int OPEN = 1;
	public static final int HIGH = 2;
	public static final int LOW = 3;
	public static final int CLOSE = 4;
	public static final int VOLUME = 5;
	public static final int ADJCLOSE = 6;

	private ArrayList<GregorianCalendar> dates;
	private ArrayList<Double> opens;
	private ArrayList<Double> highs;
	private ArrayList<Double> lows;
	private ArrayList<Double> closes;
	private ArrayList<Integer> volumes;
	private ArrayList<Double> adjcloses;

	public StockDownloader(String symbol, GregorianCalendar start, GregorianCalendar
end) {
		dates = new ArrayList<GregorianCalendar>();
		opens = new ArrayList<Double>();
		highs = new ArrayList<Double>();
		lows = new ArrayList<Double>();
		closes = new ArrayList<Double>();
		volumes = new ArrayList<Integer>();
		adjcloses = new ArrayList<Double>();

		String url = "http://ichart.finance.yahoo.com/table.csv?s=" + symbol +
			"&a=" + start.get(Calendar.MONTH) +
			"&b=" + start.get(Calendar.DAY_OF_MONTH) +
			"&c=" + start.get(Calendar.YEAR) +
			"&d=" + end.get(Calendar.MONTH) +
			"&e=" + end.get(Calendar.DAY_OF_MONTH) +
			"&f=" + end.get(Calendar.YEAR) +
			"&g=d&ignore=.csv";

		try {
			URL yhoofin = new URL(url);
			URLConnection data = yhoofin.openConnection();
			Scanner input = new Scanner(data.getInputStream());
			String[] lineVector;
			String[] dateVector;
			//skip first line
			if (input.hasNext()) {
```

```
                        input.nextLine();
                }

                while (input.hasNext()) {
                        String line = input.nextLine();
                        lineVector = line.split(",");
                        dateVector = lineVector[DATE].split("-");
                        GregorianCalendar gc =
                                new GregorianCalendar(Integer.parseInt(dateVector[0]),
Integer.parseInt(dateVector[1]), Integer.parseInt(dateVector[2]));
                        dates.add(gc);
                        opens.add(Double.parseDouble(lineVector[OPEN]));
                        highs.add(Double.parseDouble(lineVector[HIGH]));
                        lows.add(Double.parseDouble(lineVector[LOW]));
                        closes.add(Double.parseDouble(lineVector[CLOSE]));
                        volumes.add(Integer.parseInt(lineVector[VOLUME]));
                        adjcloses.add(Double.parseDouble(lineVector[ADJCLOSE]));

                }
            }
            catch (Exception e) {
                System.out.Println(e.getMessage());
            }
        }
        public ArrayList<Double> getAdjusts() {
            return adjcloses;
        }
}
```

```
package stocks;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.GregorianCalendar;

/**
* @author hochan
*/
public class Stocks {

        void print(String msg, double[] x) {
            System.out.println(msg);
            for (double d : x) System.out.println(d);
        }

        /**
        * This is a "wrapped" signal processing-style autocorrelation.
        * For "true" autocorrelation, the data must be zero padded.
        */
        public void bruteForceAutoCorrelation(double[] x, double[] ac) {
            Arrays.fill(ac, 0);
            int n = x.length;
            for (int j = 0; j < n; j++) {
                for (int i = 0; i < n; i++) {
```

Downloading Stock Prices: Java

```
                        ac[j] += x[i] * x[(n + i - j) % n];
                }
        }
    }

    private double sqr(double x) {
        return x * x;
    }

    void test() {
        GregorianCalendar start = new GregorianCalendar(2012, 1, 1);
        GregorianCalendar end = new GregorianCalendar(2016, 1, 1);
        StockDownloader downloader = new StockDownloader("RY.TO", start, end);
        ArrayList<Double> adjcloses = downloader.getAdjusts();
        double[] data = new double[adjcloses.size()];
        for (int i = 0; i < adjcloses.size(); i++) {
            data[i] = adjcloses.get(i) / 100.0;
        }
        double[] ac1 = new double[data.length];
        bruteForceAutoCorrelation(data, ac1);
        print("Auto Correlation", ac1);
    }
    /**
    * @param args the command line arguments
    */
    public static void main(String[] args) {
        new Stocks().test();
    }
}
```