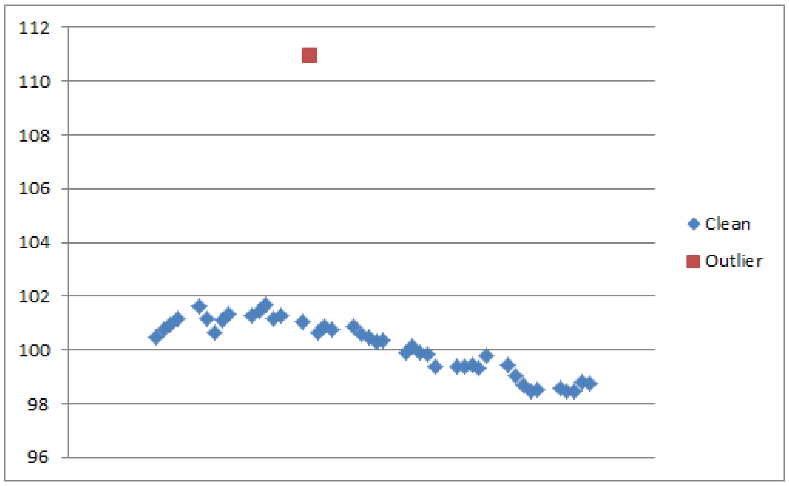
Outliers Problem

Write an application to detect outlier data points in the attached CSV file. Please implement the solution in C#.

For this problem, an outlier is defined as a data point with a value that is significantly different from the points immediately around it.



The purpose of this exercise is for you to demonstrate your coding style – the way you structure your code and the approach you take. We are interested in your ability to write flexible / maintainable code e.g. interfaces and informative comments.

Requirements:

- The application should read the CSV file in, inform the user of the outliers found, and then write a clean version of the data with the outlier points removed.

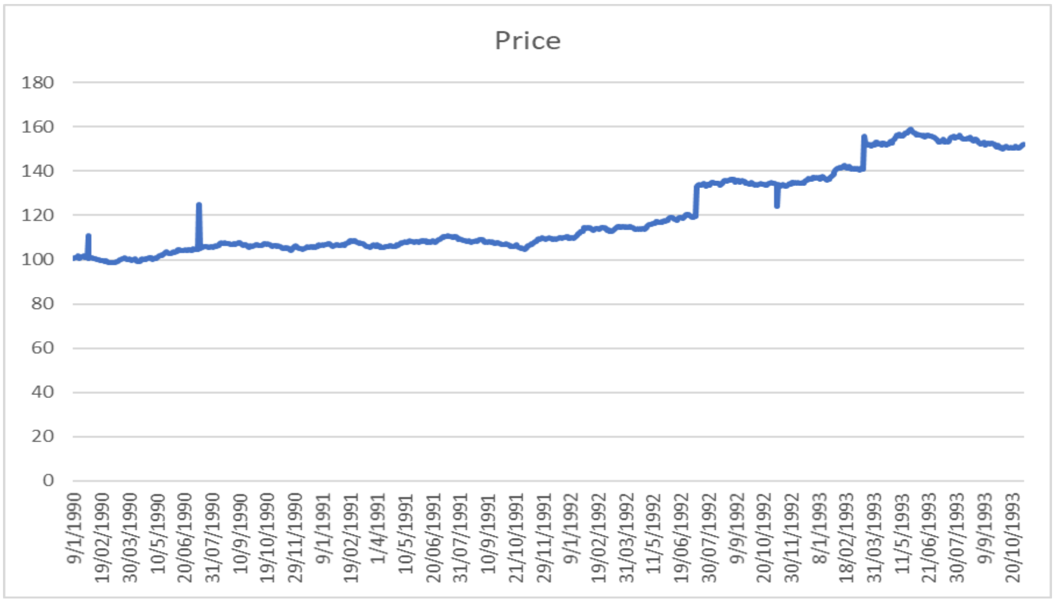
- When analyzing each point the outlier algorithm should only use previous (historical) data points, **as if the data were being received chronologically**.

- **The outlier detection algorithm should not be complex**. It is fine for you to use a basic detection algo of your own design – you don't have to apply a "standard" outlier test as found online (and such tests may not work well for this dataset).

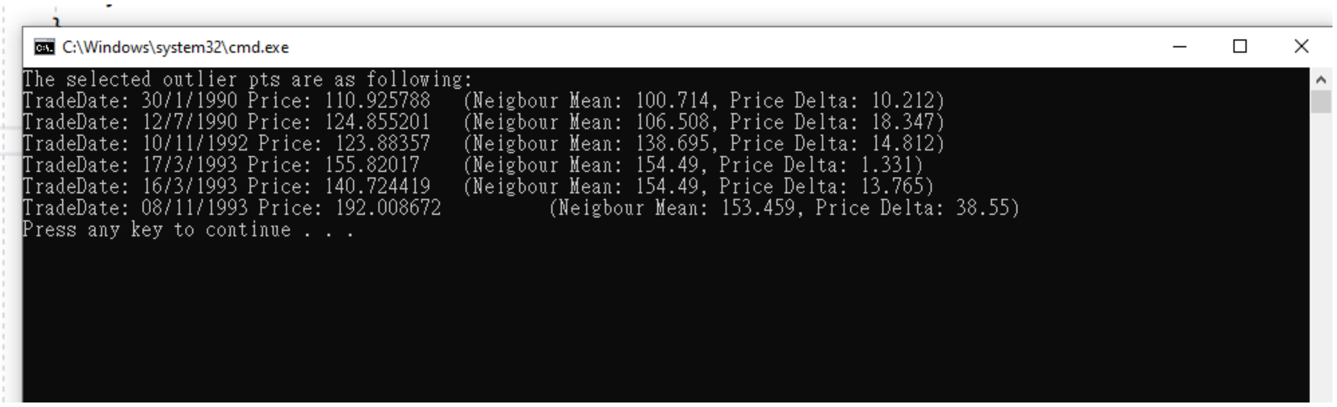
- **You only need to implement a single algorithm against the CSV file data source**. However, you should consider how your code could be extended to support additional functionality. In this case that might mean allowing for alternative outlier detection algorithms or other data sources / outputs such as a database or remote process.

Answers

According to given **Outliers.csv** data file, we can locate several outlier points in the graph as shown in the following:

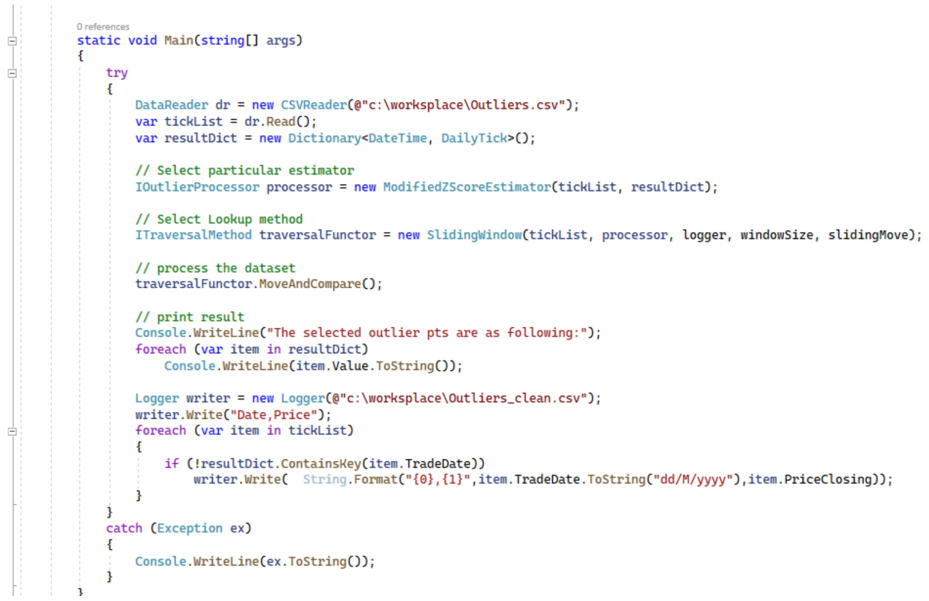


The program will preform the outlier detector to spot out the estimated outlier points and bookmarks them and finally showing on the console screen:



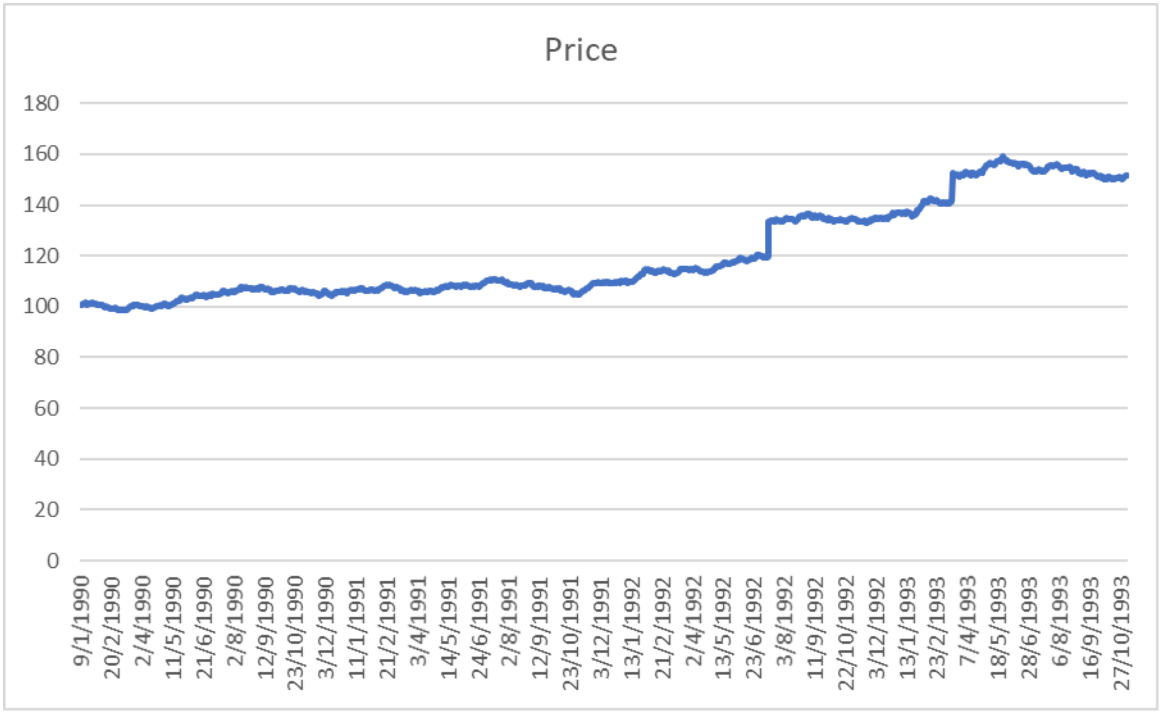
```
The selected outlier pts are as following:
TradeDate: 30/1/1990 Price: 110.925788   (Neigbour Mean: 100.714, Price Delta: 10.212)
TradeDate: 12/7/1990 Price: 124.855201   (Neigbour Mean: 106.508, Price Delta: 18.347)
TradeDate: 10/11/1992 Price: 123.88357   (Neigbour Mean: 138.695, Price Delta: 14.812)
TradeDate: 17/3/1993 Price: 155.82017    (Neigbour Mean: 154.49, Price Delta: 1.331)
TradeDate: 16/3/1993 Price: 140.724419   (Neigbour Mean: 154.49, Price Delta: 13.765)
TradeDate: 08/11/1993 Price: 192.008672          (Neigbour Mean: 153.459, Price Delta: 38.55)
Press any key to continue . . .
```

Hence, it will generate a next **Outliers_clean.csv** for a normalized data., the smoothing result can be shown in the graph:



```
0 references
static void Main(string[] args)
{
    try
    {
        DataReader dr = new CSVReader(@"c:\worksplace\Outliers.csv");
        var tickList = dr.Read();
        var resultDict = new Dictionary<DateTime, DailyTick>();

        // Select particular estimator
        IOutlierProcessor processor = new ModifiedZScoreEstimator(tickList, resultDict);

        // Select Lookup method
        ITraversalMethod traversalFunctor = new SlidingWindow(tickList, processor, logger, windowSize, slidingMove);

        // process the dataset
        traversalFunctor.MoveAndCompare();

        // print result
        Console.WriteLine("The selected outlier pts are as following:");
        foreach (var item in resultDict)
            Console.WriteLine(item.Value.ToString());

        Logger writer = new Logger(@"c:\worksplace\Outliers_clean.csv");
        writer.Write("Date,Price");
        foreach (var item in tickList)
        {
            if (!resultDict.ContainsKey(item.TradeDate))
                writer.Write(  String.Format("{0},{1}",item.TradeDate.ToString("dd/M/yyyy"),item.PriceClosing));
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.ToString());
    }
}
```

The default file path can be changed directly in main() or alter the contents to test the outliers checking result with Outlier csv.



<u>How to do the outlier detection</u>

Problem analysis

In general, the outlier occurrence means the point of daily closing price is normally far from the mean and the other data points in a large variance. Hence, several outlier points will possible be occurred in mathematical expression they are representing the local maxima and local minima, which is the sudden jump and sudden drop with a certain period of time.

Practical solutions

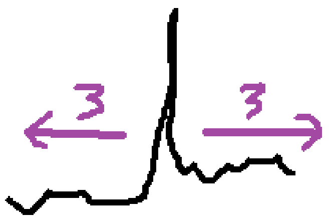
As I may know that the outlier can be detected by measuring the price distance with the sample mean in a particular threshold, to locate each of the outliers I use sliding windows to walk through each window prices dataset to check any max or min points in using a simple modified Z-Score evaluation. It is parametric approach to compute the z-scope for each sample (each sliding window) on the data set with a threshold must be specified. According to a reading article "Detection of Outliers", I use the modified Z-score by Iglewicz and Hoaglin assumption:

$$M_i = \frac{0.6745(x_i - \tilde{x})}{\mathrm{MAD}}$$

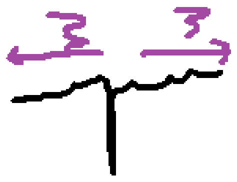
with MAD denoting the median absolute deviation and $\tilde{x}$ denoting the median.

In the article, the authors recommend the modified Z-score with an absolute value > 3.5 can be labeled as potential outliers. But to deal with this example, I make a minor changing of scale down the MAD incrementally but keep 67.45% price distance between the observe price value and its neighbor sample means. It will treat as potential outlier in a price distance assumption if price slope >= 10.

As I grab the potential outlier points in first step, I do the next step to check each delta changes with its neighbouring price. To make the idea simple, I've used the price check function *IsHighDelta()* to verified.



To find the maxima



To find the minma

Class design

To make the program flexible and higher replacement with alternative methods, I use adaptive design pattern to relax the class design. By using the C# interface *IOutlierProcessor* it can dynamically replace another algorithm classes by implementing the expose interface. As the aggregator interface, I use it as parameter in SlidingWindow class to invoke the computation on each request of sliding movement. As the same, the SlidingWindow can be said as method class by implement *ITraveralMethod* in similar design. Certain number of unit testing have been included.

Conclusion

In doing the unit testing, I found that the program could do better in price comparison. I think I can make a price sorting from high to low price ascending in each window to reduce the time complexity. In meanwhile, those assumption units can almost be parametric changed by config or curve seeking. I may not intend to make changes here due to time constraint. I'd welcome to submit my work in this code test interview. Thank you.