

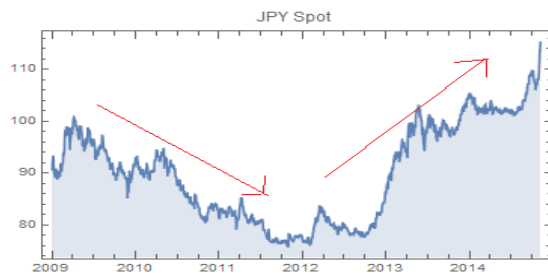
Detecting patterns in the Time Series

The ability to detect patterns and signals in the time series plays an important role the data analysis, data models and forecasting. Patterns help transforming raw data into information, which is much more valuable to explain (i) when (ii) what and (iii) how the changes occurred to some data series. Patterns also point out at relationships that may exist between certain observations and their causations.

Detecting patterns in time series with Mathematica is straightforward and reasonably simple task. This is due to much improved version of **Mathematica 10** that comes with the dedicated TimeSeries pack of routines and functions that are particularly built for the efficient time series analysis. These are complemented with a rich repository of resources available in related disciplines – such as probability and statistics, physics or image processing.

Consider the historical time series for the **JPY/USD exchange rate** that we want to analyse and detect patterns in the data. We select the 5Y window for this demonstration case:

```
DateListPlot[jpyhdata, PlotLabel -> "JPY Spot", Filling -> Axis]
```

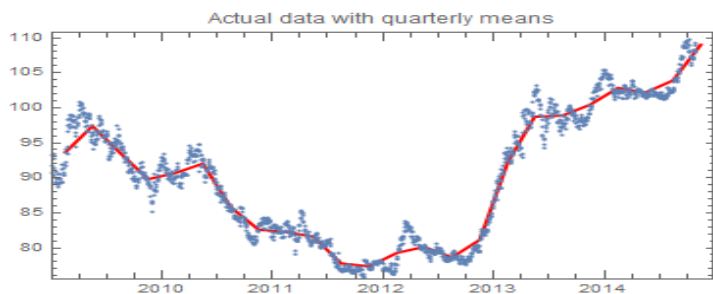


The JPY spot fluctuated in the past 5 years between [75, 115] with two fundamental trends: (i) *appreciation* – till the end of 2012 followed by (ii) *depreciation* - from 2013 till present days.

Our objective is to break the series into non-overlapping windows that we may want to analyse in detail. Our main toll will be the *TimeSeriesAggregate* function.

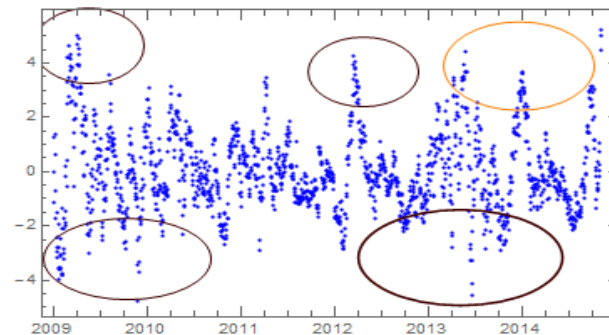
Let's look at the quarter-by-quarter mean:

```
hmean = TimeSeriesAggregate[jpyhdata, win];
Show[DateListPlot[%, PlotStyle -> {Red, Thick},
PlotLabel -> "Actual data with quarterly means"], ListPlot[jpyhdata]]
```



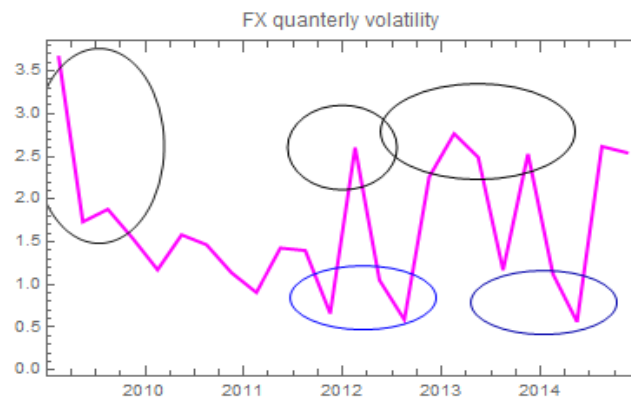
The quarterly mean value evolution indicates where the actual trend stays relative to the noise around that mean.

The residuals are showing patterns with different magnitude of dispersion.



Let's have a look at volatility:

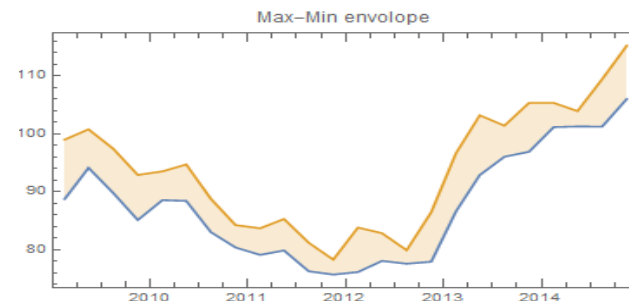
```
hvol = TimeSeriesAggregate[jpyhdata, win, StandardDeviation];
DateListPlot[%, PlotStyle -> {Thick, Magenta},
PlotLabel -> "FX quarterly volatility"]
```



As the trend changed its slope and directions, the volatility was moving quite significantly. We can identify the periods of high volatility (early 2009, early 2012, twice in 2013 and late 2014). On the other hand, the yen appreciation period was marked by subdued volatility. This behaviour is quite common – stiff resistance around historical lows leads to the narrowing of dispersion range and hence lower volatility.

To investigate this dispersion further, we can have a look at the 'envelope bands' formed by max-min set:

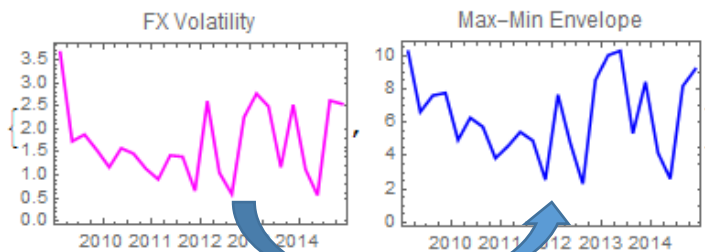
```
mins = TimeSeriesAggregate[jpyhdata, win, Min];
maxs = TimeSeriesAggregate[jpyhdata, win, Max];
DateListPlot[{mins, maxs}, PlotLabel -> "Max-Min envelope", Filling -> {1 -> {2}}]
```



The envelope shows the tightening and widening trends with particular peaks around the change of volatility curvature and inflection points.

There is also a striking similarity between the volatility and envelope shapes:

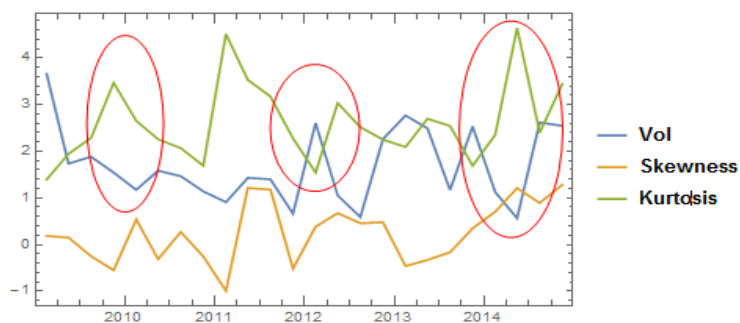
```
{DateListPlot[hvol, PlotStyle -> Magenta, PlotLabel -> "FX Volatility"],
DateListPlot[TimeSeriesThread[First[#] - Last[#] &, {maxs, mins}],
PlotStyle -> Blue, PlotLabel -> "Max-Min Envelope"]}
```



Max-Min envelope is a good volatility predictor and any time the envelope gradient changes, we may expect the move in the underlying volatility. This is a useful pattern to follow when studying dispersion in the time series.

Additional useful piece of analysis can be obtained from the higher-order moments such as skewness and kurtosis:

```
DateListPlot[{hvol, TimeSeriesAggregate[jpyhdata, win, Skewness],
TimeSeriesAggregate[jpyhdata, win, Kurtosis]},
PlotLegends -> {"Vol", "Skewness", "Kurtosis"}]
```

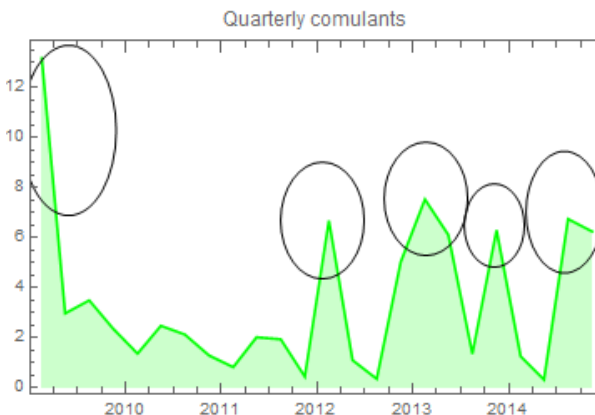


We can observe:

- 1) Positive skewness every time vol increases
- 2) Lack of symmetry between volatility and kurtosis – this means other factors than volatility making contribution into the tails of the distribution

Alternative way to explore the data patterns is to analyse cumulants – the log transforms of moments. The second quarterly cumulant serves as a good variance proxy and displays the following behaviour:

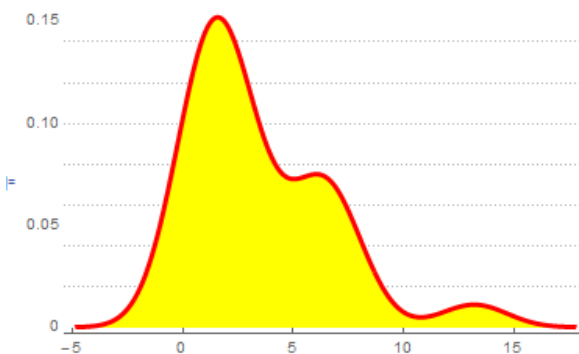
```
DateListPlot[TimeSeriesAggregate[jpyhdata, win, Cumulant[#, 2] &],
PlotStyle -> Green, Filling -> Axis, PlotLabel -> "Quarterly cumulants"]
```



Cumulants are quite similar in shape to the volatility patterns and exhibit corresponding peaks / valleys.

Cumulants preserve certain distributional characteristics:

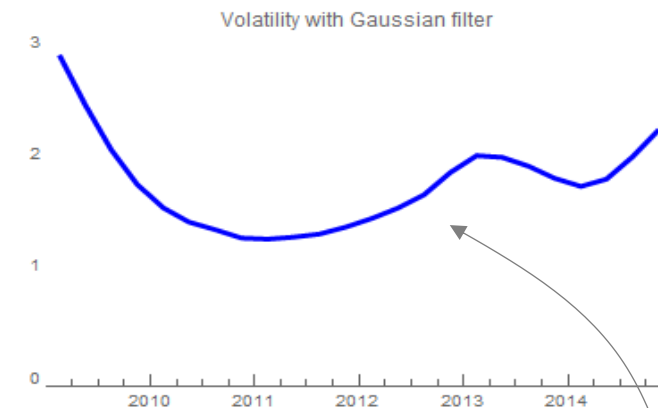
```
cs = TimeSeriesAggregate[jpyhdata, win, Cumulant[#, 2] &];
SmoothHistogram[cs, PlotTheme -> "Business", PlotStyle -> Red, Filling -> Axis,
FillingStyle -> Yellow]
```



This is distributional patterns of quadratic cumulants over the entire observation period. Long right-hand tail indicates bi-modal behaviour and most likely non-parametric representation of the underlying data.

We can borrow filtering technique from signal & image processing for data adaptation. This is the adjusted volatility with the Gaussian linear quartic filter:


```
DateListPlot[GaussianFilter[hvol, 4], PlotTheme -> "Web", PlotStyle -> Blue,
PlotLabel -> "Volatility with Gaussian filter"]
```



The filter eliminates the 'noise' and turn the volatility into a smoother function.

We can fit the Linear Model (cubic polynomial) to the filtered volatility data:

```
gf = GaussianFilter[hvol, 4]
```

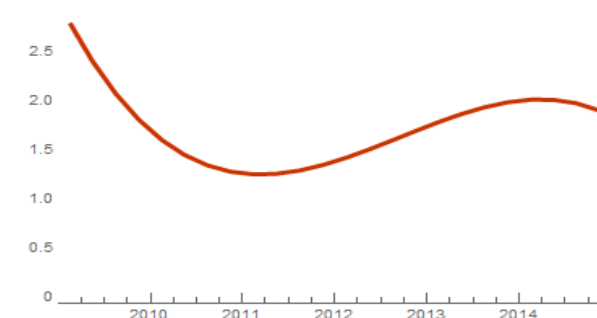
TimeSeries [ Time: 15 Feb 2009 to 15 Nov 2014
Data points: 24]

```
lmf = LinearModelFit[gf, {1, x, x^2, x^3}, x]
```

```
FittedModel [ 77952.6 - 0.0000657698x + 1.8494 × 10-14 x2 - 1.73315 × 10-24 x3 ]
```

and plot it:

```
Table[{x, lmf[AbsoluteTime[x]]}, {x, gf["Times"]};
DateListPlot[%, PlotTheme -> "Web"]
```



The cubic model provides reasonable representation of volatility patterns as function of time.