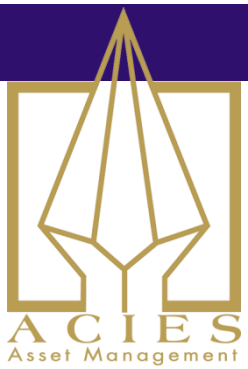


Rebalancing Act

Most literature on so called money management is about gambling, not professional trading. This document will explain a different way to look at how to change position sizes. Not to make risk larger or smaller, but rather to attempt to keep it under control.

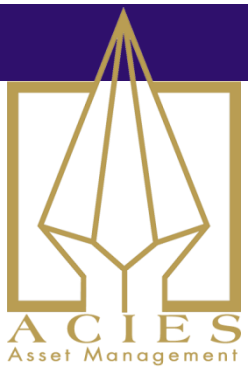
When you open a position, you likely apply some sort of volatility based formula for setting the size. If you don't, you probably should. There are several methods to do this, and they don't differ that much really. But the usual ways to calculate position sizes include looking at the volatility of the instrument in question and measuring how its fluctuations are likely to impact the overall investment portfolio.

In the demo I want to do here, I'll use a simple ATR based sizing method. It should be familiar to you all, but just in case, here it is again, in plain code: (C# for RightEdge as usual)

```
long numberOfContracts = (long) (
(Math.Max(1,SystemData.AccountValue * _riskFactor / (_atr.Current *
_point_value *
SystemData.AccountInfo.GetConversionRate(Symbol.CurrencyType,SystemD
ata.AccountCurrency, QuoteType.Last) ))));
```

And in text:

Multiplying the account value with the risk factor gives you a target daily fluctuation. Multiplying the ATR by the point value and the FX conversion rate gives you a per-contract fluctuation. Dividing the two gives you the number of contracts to trade.



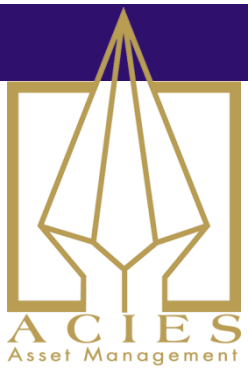
Back to the problem. This formula might not be perfect, but it's really not bad either and quite useful. But it doesn't account for how the input factors changes over time. If your holding period is a few days, that's just fine. Things normally don't change that much in a couple of days. But for longer term trading models, things can get really out of whack.

If you just leave it like that, two things will happen. Your risk will effectively become random and you will be living in an illusion of having controlled it.

Dynamic Markets

Think of the inputs to that position sizing formula. The key factors are instrument ATR and account value. Neither of those will be the same in a month. If your position is absolutely flat, but other positions make large gains for your account, your position will have lower risk than it should. Relative to the account, which is all that matters. If your position volatility increases, you'll find yourself with a higher risk than you had intended. As you can see, there are multiple scenarios possible where the risk will change over time. Often these changes are significant after a month or two.

Let's measure it. I'll use the Long 12 Months Momentum model for this demo. You already know this trading model and should have the full rules and code already. If you don't have it, send me a mail and I'll mail you that document. This model has a long holding period and has performed excellently despite its simplicity.



Quick rules recap (email me for detailed code if you don't already have it):

- If the price is higher than it was 250 trading days ago, be on the long side.
- That's it.

First, I'll run this simulation on the markets in the weekly futures report, using a position sizing formula like above with a risk factor of 0.001, or 10 basis points. In this first version, there's no rebalancing. Whatever the position size was when we opened the trade, that's what we'll stick with until the position is closed.

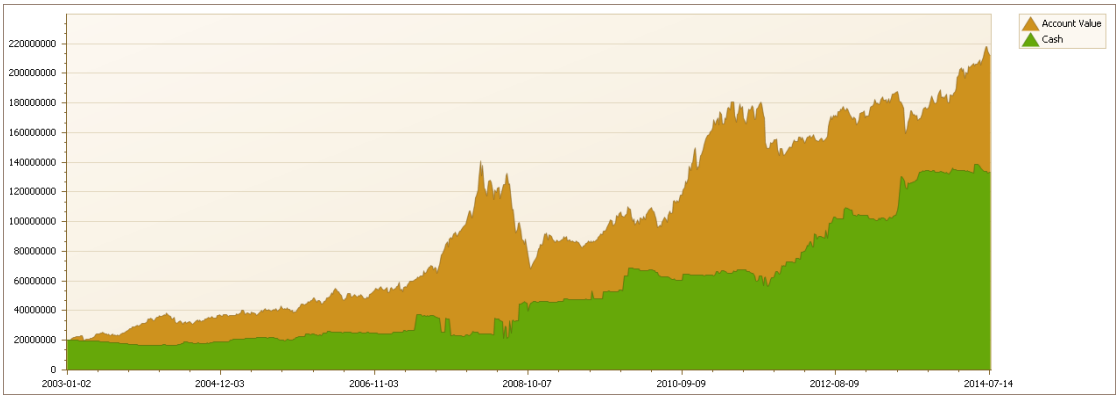
I've add a visual indicator to the charts, to show how the risk factor changes over time.

This risk indicator is calculated like this:

```
if(pos.Count == 0)
        _risk.Current =double.NaN;
else
        _risk.Current = (pos[0].CurrentSize * _atr.Current *
_point_value *
SystemData.AccountInfo.GetConversionRate(Symbol.CurrencyType,SystemDa
ta.AccountCurrency, QuoteType.Last)) / this.SystemData.AccountValue;
```

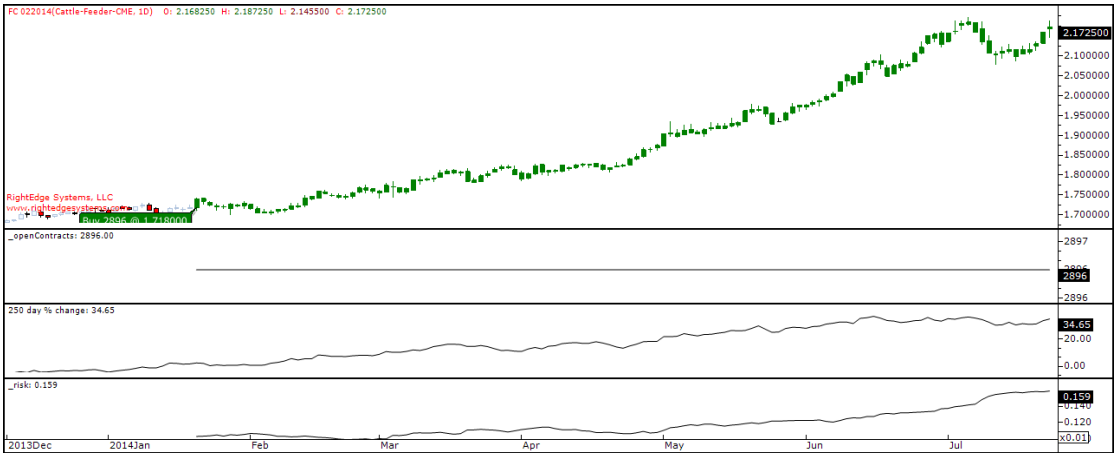
Same formula as before, but now we're looking to back the risk factor out, instead of inputting it. This indicator will plot the actual risk factor over time, showing how it changes.

Simulation Results – No Rebalance



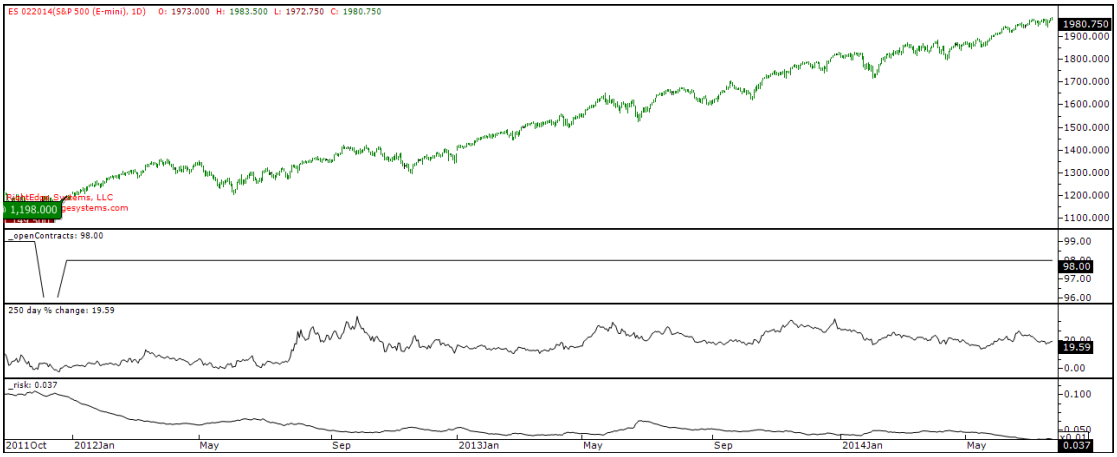
Annualized Return	22.8%
Max DD	-51.5%
Sharpe	0.83

Quite a wild strategy, but profitable over time. Let's see some trade charts for a clue about why it's such a wild ride.



Look at this trade chart for feeder cattle. Great trade, great profits. The top pane shows the price and buy/sell signals. Next pane shows number of open contracts in the simulation. Then you'll see the 12M momentum percent, which is the trade signal for this model. Last, and most important here, you'll see current risk factor, as it changes from the position inception.

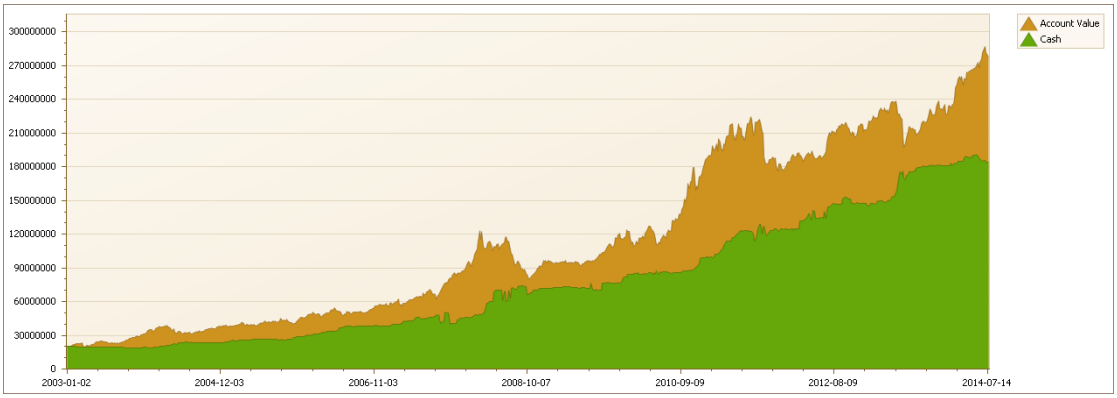
We start off with a risk factor of 0.001, so whenever a position is opened that's where this indicator will be. After that, it can go anywhere. In this case, the vola keeps going up and with it the actual risk of our position in FC. At the moment, this model holds a risk factor that's 60% higher than when it was opened. The timing of our entry signal was allowed to dictate the position size for a very long time. Doesn't really make sense.



Now look at the S&P 500 trade chart above. Same indicators. As you see, when the position was opened, the risk factor was at 10 basis points, as it should be. But the position was opened during a time of heightened vola, so as the market normalized, the actual risk got smaller and smaller. In the end, we're holding only 4.5 basis points risk factor, or less than half of what it should be. Any why again? Just because we happened to open the position at a volatile time.

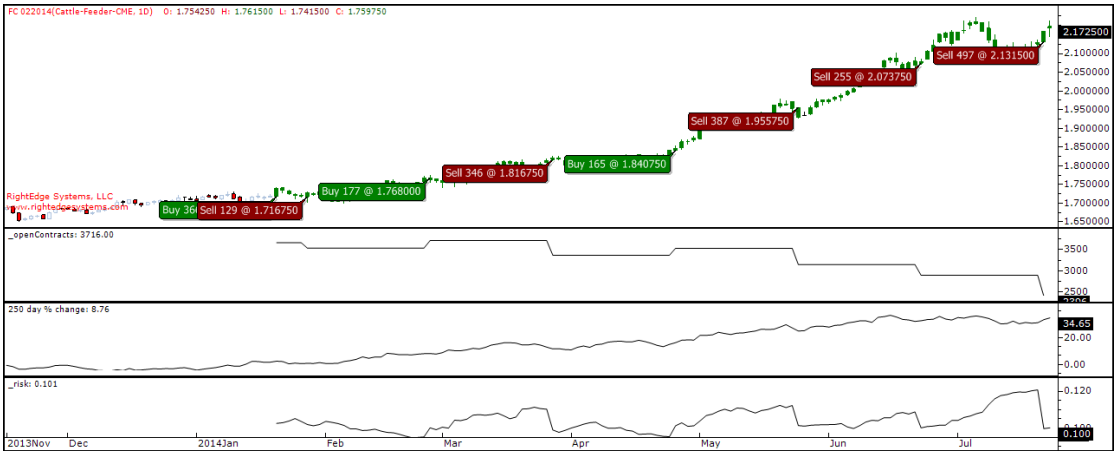
So let's try a simple rebalance. Once a month, we simply reset the trade size. That is, we recalculate the position size using the same formula as when the position was first opened. Then we compare the result with what we actually hold, and reduce or increase to match.

Simulation Results – With Rebalance



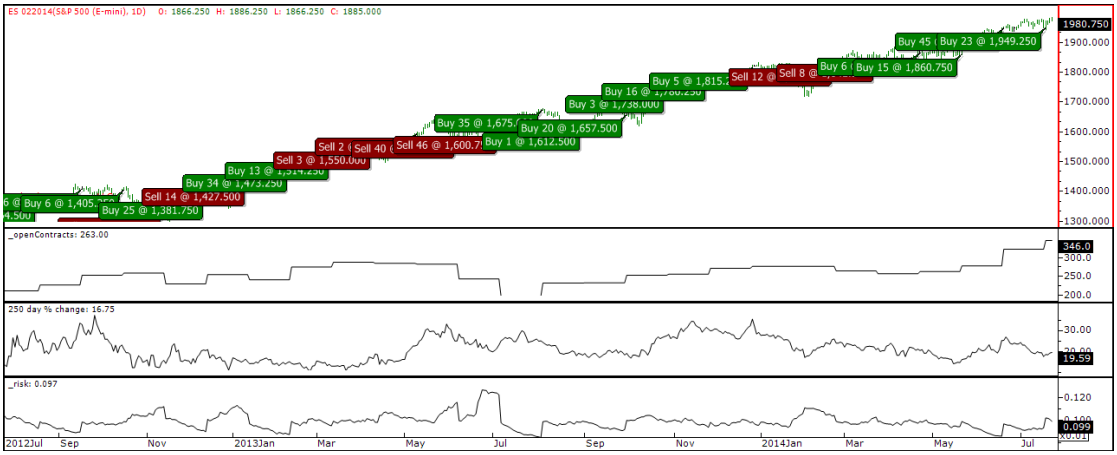
Annualized Return	25.7%
Max DD	35.1%
Sharpe	1.1

Yes, that's right. We got better performance at lower risk. Let's check the trade charts.



Same feeder cattle chart as before. Now it looks much more realistic. Note the bottom pane, how the actual risk factor at work stays the same. It's reset every 20 days. See how it moves back down to 10bp every time there's a trade in the top pane. Risk is kept under control instead of running around randomly.

But didn't this FC trade generate a lot less money in this version? Sure. But it took deliberate risk and made a good return. The first version rolled the dice and happened to win this time. Let's look at the S&P chart.



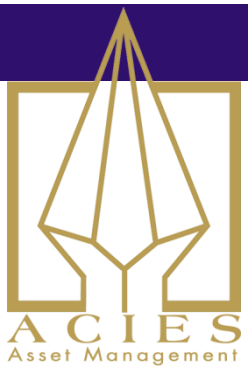
Yep, in this case we kept on increasing all the way up. See how the bottom indicator is periodically reset to 10bp. So for this trade, we made substantially more money over time. In the end though, it's not about more or less money. It's about controlling your risk.

Taking risk is fine. That's what we do for a living. But we take deliberate risk. A long term trading model that doesn't rebalance is essentially random in terms of risk.

Implementing Rebalancing in Simulations

As usual, I'll show you code examples in C#, written for RightEdge. It should be quite easy to translate this into your own simulation platform. Note that some platforms, like WealthLab for instance, doesn't support changing position size. That makes those platforms less suitable for serious simulations.

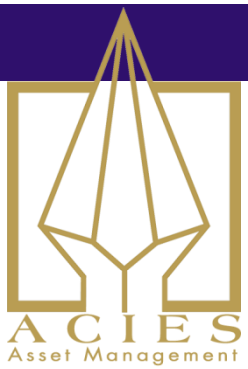
Here's a quick implementation of the rebalance use for the model above:




```
#region Rebalance
                // If position size rebalance is enabled, adjust position sizes every X days.
                bool rebalanceToday = false;
                if ( (currentBar % _rebalanceFrequency == 0) && (_rebalance == 1) ) rebalanceToday
= true;

                if ( rebalanceToday )
                        {
                                foreach(Position posx in OpenPositions)
                                        {
                                                targetContracts = positionSize(); //
Recalculate position size and adjust as needed.
                                                if ((targetContracts - posx.CurrentSize) > 0)

        SystemData.PositionManager.AddToPosition(posx.ID,(targetContracts -
posx.CurrentSize),OrderType.MarketOnOpen,0,"Rebalance");
                                                else

        SystemData.PositionManager.RemoveFromPosition(posx.ID, Math.Abs(targetContracts -
posx.CurrentSize),OrderType.MarketOnOpen,0,"Rebalance");
                                        }
                        }
#endregion
```

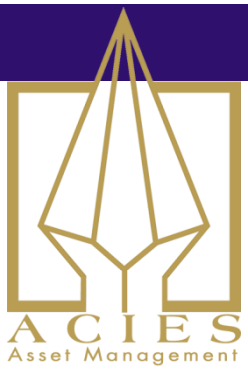
Explanation:

- _rebalance is a system parameter, toggling if you want to enable rebalancing or not. If not, this code is skipped.
- _rebalanceFrequency is also a system parameter, setting how many days apart we should do the rebalance.
- currentBar is a running count of how many trading bars (days) have been processed.
- If _rebalanceFrequency is 20, `(currentBar % _rebalanceFrequency == 0)` will be true every 20 days.
- So every 20 days, in that case, we run the actual rebalance code.
- This iterates all positions, calculates position size using the same formula as when the position was opened, and resets the trade size to that number.

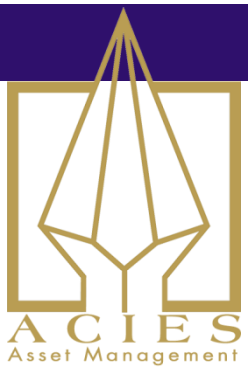


Perhaps you'd like the full code of the trading system used in the simulations above?

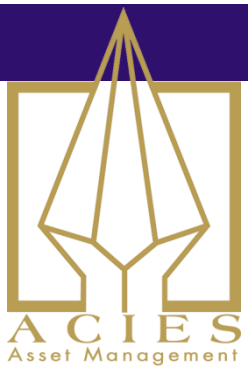
```
#region Using statements
using System;
using System.Drawing;
using System.Collections.Generic;
using System.Linq;
using RightEdge.Common;
using RightEdge.Common.ChartObjects;
using RightEdge.Indicators;

#endregion

#region System class
public class MySystem : MySystemBase
{
	public override void Startup()
	{
		// Perform initialization or set system wide options here

	}
}
#endregion

public class MySymbolScript : MySymbolScriptBase
{
	public override void Startup()
	{
		_atr = new AverageTrueRange(50);
		_atr.ChartSettings.ShowInChart = false;

		// Fetch system parameters
		SystemParameters prmtr = SystemData.SystemParameters;
		_riskFactor = prmtr["riskFactor"];
		_lookBack = (int) prmtr["lookBack"];
		_allowShorts = prmtr["allowShorts"];
		_rebalanceFrequency = (int)prmtr["rebalanceFreq"];
		_rebalance = (int)prmtr["rebalance"];

		// For keeping track of the price a year ago
		_yearAgo = new UserSeries();
		_yearAgo.ChartSettings.ShowInChart = false;

		_yearChange = new UserSeries();
		_yearChange.ChartSettings.ShowInChart = true;
		_yearChange.ChartSettings.ChartPaneName = "YearChangePane";
		_yearChange.ChartSettings.DisplayName = _lookBack + " day % change";


		_openContracts = new UserSeries();
```



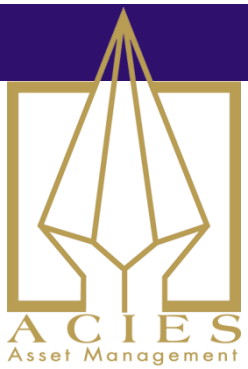

```
                _openContracts.ChartSettings.ShowInChart = true;
                _openContracts.ChartSettings.ChartPaneName = "Open Contracts";
                _openContracts.ChartSettings.LineType = SeriesLineType.Solid;

                _risk = new UserSeries();
                _risk.ChartSettings.ShowInChart = true;
                _risk.ChartSettings.ChartPaneName = "RiskPane";

        }
        AverageTrueRange _atr;
        double _riskFactor;
        int _lookBack;
        double _allowShorts;
        double _point_value;

        int currentBar = 0;
        int _rebalanceFrequency;
        int _rebalance;

        UserSeries _yearAgo;
        UserSeries _openContracts;
        UserSeries _yearChange;

        BarData _openBar;

        UserSeries _risk;

        public override void NewBar()
        {
                currentBar++;                                           // Keep a running
count of bars.
                if (SystemData.InLeadBars) return; // Don't run any logic until we have
enough history.
                if (currentBar < (_lookBack+10)) return;

                IList<Position> pos =
SystemData.PositionManager.GetOpenPositions(Symbol);
                long targetContracts;

#region Cosmetics
                if (pos.Count !=0)
                {
                        if (pos[0].Type == PositionType.Short)
                        {

        SystemData.GetPricePane(Symbol).SetBarColor(Bars.Current, Bars.Current,
Color.Red);
                                _openContracts.Current =  pos[0].CurrentSize;
                        }
                        else
                        {
```




```
        SystemData.GetPricePane(Symbol).SetBarColor(Bars.Current, Bars.Current,
Color.Green);
                                _openContracts.Current =  pos[0].CurrentSize;
                        }

                }
#endregion


#region Rebalance
                // If position size rebalance is enabled, adjust position sizes every X
days.
                bool rebalanceToday = false;
                if ( (currentBar % _rebalanceFrequency == 0) && (_rebalance == 1) )
rebalanceToday = true;

                if ( rebalanceToday )
                        {
                                foreach(Position posx in OpenPositions)
                                        {
                                                targetContracts = positionSize(); //
Recalculate position size and adjust as needed.
                                                if ((targetContracts - posx.CurrentSize)
> 0)

        SystemData.PositionManager.AddToPosition(posx.ID,(targetContracts -
posx.CurrentSize),OrderType.MarketOnOpen,0,"Rebalance");
                                                else

        SystemData.PositionManager.RemoveFromPosition(posx.ID, Math.Abs(targetContracts
- posx.CurrentSize),OrderType.MarketOnOpen,0,"Rebalance");
                                        }
                        }
#endregion

                if(pos.Count == 0)
                        _risk.Current = double.NaN;
                else
                        _risk.Current = (pos[0].CurrentSize * _atr.Current *
_point_value *
SystemData.AccountInfo.GetConversionRate(Symbol.CurrencyType,SystemData.AccountCurrenc
y, QuoteType.Last)) / this.SystemData.AccountValue;




                double yearAgo = Close.LookBack(_lookBack); // _lookBack defaults to
250.
                _yearAgo.Current = yearAgo;
                _yearChange.Current = ( (Close.Current / yearAgo) -1 ) *100;
```




```
                _point_value = Symbol.SymbolInformation.ContractSize;

                if (currentBar % 5 != 0) return; // Only trade once a week for this
model.

                if (_point_value==0.0) _point_value = 1; // Else model won't work for
other instruments than futures.

                        if(pos.Count==0) // No position open yet.
                        {
                                if (Close.Current > yearAgo) // go long
                                {
                                        long numberOfContracts = (long) (
(Math.Max(1,SystemData.AccountValue * _riskFactor /
                                                                (_atr.Current *
_point_value *
SystemData.AccountInfo.GetConversionRate(Symbol.CurrencyType,SystemData.AccountCurrenc
y, QuoteType.Last) )));

        TradingSystem.OpenPosition(Symbol,PositionType.Long,OrderType.MarketOnOpen,0,nu
mberOfContracts);
                                        _openBar = Bars.Current;
                                }
                                else
                                {
                                        if (_allowShorts == 1) // go short, if allowed
                                        {
                                                long numberOfContracts = (long) (
(Math.Max(1,SystemData.AccountValue * _riskFactor /
                                                                (_atr.Current *
_point_value *
SystemData.AccountInfo.GetConversionRate(Symbol.CurrencyType,SystemData.AccountCurrenc
y, QuoteType.Last) )); // <<== fx logic needed

        TradingSystem.OpenPosition(Symbol,PositionType.Short,OrderType.MarketOnOpen,0,n
umberOfContracts);
                                                _openBar = Bars.Current;
                                        }
                                }

                        }
                        else
                        {
                                if ((Close.Current > yearAgo) && pos[0].Type ==
PositionType.Short) // Flip short to long
                                {
                                        pos[0].CloseAtMarket("Market turned");
                                        long numberOfContracts = (long) (
(Math.Max(1,SystemData.AccountValue * _riskFactor /
                                                                (_atr.Current *
_point_value *
```




```
SystemData.AccountInfo.GetConversionRate(Symbol.CurrencyType,SystemData.AccountCurrenc
y, QuoteType.Last) ))); // <<== fx logic needed

        TradingSystem.OpenPosition(Symbol,PositionType.Long,OrderType.MarketOnOpen,0,nu
mberOfContracts);
                                }
                                else if((Close.Current < yearAgo) && pos[0].Type ==
PositionType.Long) // Flip long to short
                                {
                                        pos[0].CloseAtMarket("Market turned");
                                        if (_allowShorts == 1)
                                        {
                                                long numberOfContracts = (long)
((Math.Max(1,SystemData.AccountValue * _riskFactor /
                                                                        (_atr.Current *
_point_value *
SystemData.AccountInfo.GetConversionRate(Symbol.CurrencyType,SystemData.AccountCurrenc
y, QuoteType.Last) ))); // <<== fx logic needed

        TradingSystem.OpenPosition(Symbol,PositionType.Short,OrderType.MarketOnOpen,0,n
umberOfContracts);
                                        }
                                }
                        }
        }
        private long positionSize() // Simple position size calculation
        {
                long contracts =
                (long)(Math.Max(1,SystemData.AccountValue * _riskFactor / (_atr.Current
* _point_value *
SystemData.AccountInfo.GetConversionRate(Symbol.CurrencyType,SystemData.AccountCurrenc
y, QuoteType.Last) )));
                return contracts;
        }
}
```