Long / Short Trading Strategy Design & Backtest (TS)

Blair Azzopardi

January 21, 2023

Contents

0. Introduction & Preliminaries

This following submission was prepared using Jupyter Lab as part of the June 2022 CQF cohort Final Project.

This section (0) and subsections cover some important preliminaries that form a basis for what follows.

Sections (1) and (2) correspond to Part I and Part II of the Final Project Brief. The subsections will correspond too.

Section (3) is a conclusion of sorts.

Section (4) includes an appendix. Here you will find a list of functions of numerical techniques implemented and information around packages installed.

References can be found at the very end.

0.1 Cointegration

Cointegration is a property of a collection of time series variables.

A time series p_t, is integrated of order d if $(1-L)^d p_t$ is a stationary process (i.e. mean, variance and autocorrelation remain constant over time) where L is the lag operator. The time series can be denoted $I(d)$.

If $d = 0$ then p_t is a stationary process denoted $I(0)$.

One can represent a lagged process using the following subscript notation $p_{t-d} = L^d p_t$.

A time series that can be expressed as an autoregressive process of order k as

$$p_t = a_1 p_{t-1} + a_2 p_{t-2} + \cdots + a_k p_{t-k} + \epsilon_t \tag{AR1}$$

also has a corresponding characteristic equation of degree k,

$$m^k - m^{k-1} a_1 - m^{k-2} a_2 - \cdots - a_k = 0. \tag{AR2}$$

If the above equation has a root when $m = 1$ of multiplicity r then p_t is integrated of order r, i.e. p_t is $I(r)$. If $r = 1$ then p_t is $I(1)$ and is said to have a unit root.

Several time series are co-integrated if a linear combination of them has a lower order of integration.

By locating a pair of tradable time series whose combination is $I(0)$, we can then trade the combination using a mean reverting strategy.

0.2 Ornstein–Uhlenbeck process (OU)

The Ornstein–Uhlenbeck process Y_t is a stationary Gauss–Markov process and is temporally homogeneous. The process is mean-reverting. It can be expressed as,

$$dY_t = \theta(\mu - Y_t)dt + \sigma dW_t \tag{OU1}$$

where θ is the speed of reversion, μ is the equilibrium level, σ the variance and W_t is a Wiener Process.

This can be solved by setting $X_t = Y_t - \mu$, and using an integrating factor $e^{\theta t}$,

$$\begin{aligned}
dX_t &= -\theta X_t dt + \sigma dW_t \\
dX_t + \theta X_t dt &= \sigma dW_t \\
e^{\theta t}(dX_t + \theta X_t dt) &= e^{\theta t}\sigma dW_t \quad \text{[integ. factor]} \\
d(e^{\theta t}X_t) &= e^{\theta t}\sigma dW_t \\
e^{\theta s}X_s\big|_{s=t}^{t+\tau} &= \sigma \int_{s=t}^{t+\tau} e^{\theta s} dW_s \\
e^{\theta(t+\tau)}X_{t+\tau} - e^{\theta t}X_t &= \sigma \int_{s=t}^{t+\tau} e^{\theta s} dW_s \\
X_{t+\tau} &= e^{-\theta\tau}X_t + \sigma \int_{s=t}^{t+\tau} e^{\theta(s-t-\tau)} dW_s \\
Y_{t+\tau} - \mu &= e^{-\theta\tau}(Y_t - \mu) + \sigma \int_{s=t}^{t+\tau} e^{\theta(s-t-\tau)} dW_s \\
Y_{t+\tau} &= e^{-\theta\tau}Y_t + \mu(1 - e^{-\theta\tau}) + \sigma \int_{s=t}^{t+\tau} e^{\theta(s-t-\tau)} dW_s
\end{aligned} \tag{OU2}$$

By setting $t = 0$ and $\tau = t$ in (OU2) we can determine the expectation of Y_t,

$$E(Y_t) = e^{-\theta t}Y_0 + \mu(1 - e^{-\theta t})$$

and also the variance,

$$
\begin{aligned}
Var(Y_t) &= E((Y_t - E(Y_t))^2)) \\
&= \sigma^2 E((\int_{s=0}^{t} e^{\theta(s-t)} dW_s)^2) \\
&= \sigma^2 E(\int_{s=0}^{t} e^{2\theta(s-t)} ds) \quad \text{[Ito isometry]} \\
&= \frac{\sigma^2}{2\theta} e^{2\theta(s-t)} \Big|_{s=0}^{t} \\
&= \frac{\sigma^2}{2\theta}(1 - e^{-2\theta t}) \qquad \text{(OU3)}
\end{aligned}
$$

0.3 Vector Autoregressive Process (VAR)

The Vector Autoregressive process of order p, denoted VAR(p), is a type of stochastic process that can capture the relationship between itself and its own past values.

$$
y_t = C + A_1 y_{t-1} + \cdots + A_p y_{t-p} + \epsilon_t \qquad \text{(VAR1)}
$$

where y_t here is a vector process, A_i are coefficient matrices for the lagged variables y_i and ϵ_t is an uncorrelated white noise vector process.

This is similar to our non-vector process in (AR1).

0.3.1 Solution

The above equation (VAR1) can be expressed in terms of block matrices by stacking vectors on top of one another. If we stack n vector processes together we can express (VAR1) as:

$$
Y = BZ + \epsilon \qquad \text{(VAR2)}
$$

with

$$
\begin{aligned}
Y &= \begin{bmatrix} y_p & y_{p+1} & \cdots & y_T \end{bmatrix}, \\
B &= \begin{bmatrix} C & A_1 & \cdots & A_p \end{bmatrix}, \text{ and} \\
Z &= \begin{bmatrix} 1 & 1 & \cdots & 1 \\ y_{p-1} & y_p & \cdots & y_{T-1} \\ y_{p-2} & y_{p-1} & \cdots & y_{T-2} \\ \vdots & \vdots & \vdots & \vdots \\ y_0 & y_1 & \cdots & y_{T-p} \end{bmatrix}
\end{aligned}
$$

where each y_k is a column vector $\begin{bmatrix} y_{1,k} & y_{2,k} & \cdots & y_{n,k} \end{bmatrix}^T$.

Equation (VAR2) can be solved using a multivariate least squares approach similar to OLS (see later section 1.1.1). The resulting estimate for B is:

$$\hat{B} = YZ^T(ZZ^T)^{-1} \tag{VAR3}$$

0.3.2 Stability

VAR(p) can be expressed using the lag operator L as:

$$(I - A_1L - .. - A_pL^p)Y_t = C + \epsilon_t$$

The process is stable if all the roots of the characteristic equation

$$|I - A_1x - .. - A_px^p| = 0$$

lie outside the unit circle.

In the special case where $p = 1$ we see that this is equivalent to all the eigenvalues of A_1 lying inside the interval bounded by -1 and 1:

$$\begin{aligned} 0 &= |I - A_1x| \\ &= |\lambda I - A_1|_{\lambda=\frac{1}{x}} \end{aligned} \tag{VAR4}$$

0.4 Data sets

We chose to study world indices. We collected Open, High, Low, Close and Volume data for each index and converted prices to USD values where necessary. The table below lists indices studied along with some useful statistics.

Table 1: World Indices data sets

ID	Name	Ccy	Start	N	μ	σ
INDU	Dow Jones Industrial Average	USD	2018-01-02	1259	29053.8	4035.3
SPX	S&P 500 INDEX	USD	2018-01-02	1259	3449.7	668.9
CCMP	NASDAQ Composite Index	USD	2018-01-02	1259	10434.9	2797.2

ID	Name	Ccy	Start	N	μ	σ
SPTSX	S&P/TSX Composite Index	CAD	2018-01-02	1242	13547.6	1988.8
MEXBOL	S&P/BMV IPC	MXN	2018-01-02	1259	45789.7	4950.7
IBOV	Ibovespa Brasil Sao Paulo Stoc	BRL	2018-01-02	1239	101550.3	14378.8
SX5E	EURO STOXX 50 Price EUR	EUR	2018-01-02	1271	4062.1	482.9
UKX	FTSE 100 Index	GBP	2018-01-02	1250	9185.8	876.7
CAC	CAC 40 Index	EUR	2018-01-02	1268	6505.9	815.0
DAX	Deutsche Boerse AG German Stoc	EUR	2018-01-02	1254	14967.2	2011.4
IBEX	IBEX 35 Index	EUR	2018-01-02	1266	9868.9	1276.0
FTSEMIB	FTSE MIB Index	EUR	2018-01-02	1257	25420.9	3340.7
OMX	OMX Stockholm 30 Index	SEK	2018-01-02	1245	201.2	36.8
SMI	Swiss Market Index	CHF	2018-01-03	1244	10886.8	1505.8
NKY	Nikkei 225	JPY	2018-01-04	1203	217.3	27.8
HSI	Hang Seng Index	HKD	2018-01-02	1233	25833.7	3534.3
SHSZ300	Shanghai Shenzhen CSI 300 Inde	CNY	2018-01-02	1215	4194.5	622.0
AS51	S&P/ASX 200	AUD	2018-01-02	1253	4693.7	525.7

1. Pairs Trade Design

We intend to choose pairs of tradeable assets that together are cointegrated. Each individual asset's process must be $I(1)$ and cointegration will generate a new process $I(0)$ which will be mean reverting.

In section (1.1) we will cover some preliminaries for regression and testing then in section (1.2) will introduce the Engle-Granger procedure.

1.1 Regression Estimation & Testing

1.1.1 Ordinary Least Squares (OLS)

We assume our model to be linear,

$$y = X\beta + e \tag{OLS1}$$

where X is a matrix of full rank so there is no perfect multicollinearity.

We wish to estimate β by minimizing the squares of the error term e^2 which can be expressed as

$$\begin{aligned} e^T e &= (y - X\hat{\beta})^T (y - X\hat{\beta}) \\ &= y^T y - y^T X\hat{\beta} - (X\hat{\beta})^T y + (X\hat{\beta})^T (X\hat{\beta}) \\ &= y^T y - y^T X\hat{\beta} - \hat{\beta}^T X^T y + (X\hat{\beta})^T (X\hat{\beta}) \\ &= y^T y - 2\hat{\beta}^T X^T y + (X\hat{\beta})^T (X\hat{\beta}). \end{aligned}$$

We look for a global minimum by examining the 2nd derivative of e^2 with respect to the estimated coefficient vector $\hat{\beta}$ and setting it to zero,

$$\begin{aligned}\frac{\partial e^T e}{\partial \hat{\beta}} &= -2X^T y + 2X^T X\hat{\beta} = 0 \\ \hat{\beta} &= (X^T X)^{-1} X^T y. \end{aligned} \tag{OLS2}$$

1.1.1.1 OLS Variance

To calculate the variance we note that

$$\begin{aligned}\hat{\beta} &= (X^T X)^{-1} X^T y \\ &= (X^T X)^{-1} X^T (X\beta + e) \\ &= \beta + (X^T X)^{-1} X^T e \\ \therefore \hat{\beta} - \beta &= (X^T X)^{-1} X^T e.\end{aligned}$$

Also since we assume homoskedasticity and no autocorrelation then:

$$E[ee^T] = \sigma^2 I$$

and so,

$$\begin{aligned}E((\hat{\beta} - \beta)(\hat{\beta} - \beta)^T] &= E[((X^T X)^{-1} X^T e)((X^T X)^{-1} X^T e)^T] \\ &= E[(X^T X)^{-1} X^T e e^T X (XX^T)^{-1}] \\ &= (X^T X)^{-1} X^T E[ee^T] X (XX^T)^{-1} \\ &= (X^T X)^{-1} X^T \sigma^2 X (XX^T)^{-1} \\ &= \sigma^2 (X^T X)^{-1} X^T X (XX^T)^{-1} \\ &= \sigma^2 (X^T X)^{-1}\end{aligned} \tag{OLS3}$$

Since X is not stochastic we can pull out from Expectation operator above.

1.1.2 Augmented Dickey-Fuller (ADF)

The Augmented Dickey-Fuller Test tests the null hypothesis that a unit root is present in an autoregressive (AR) time series model. The alternative hypothesis is the time series is stationary (or trend stationary).

ADF uses p lags of the dependent variable Y_t and tests for a unit root by testing the significance of ϕ in the following regression. See (Diamond, 2022).

$$\Delta Y_t = \phi Y_{t-1} + \sum_{k=1}^{p} \phi \Delta Y_{t-k} + \epsilon_t \tag{ADF1}$$

1.1.A Identifying optimal lag for ADF

To determine the optimal lag p for the ADF test one can run it with varying lags and choose the result with the lowest information criterion like AIC or BIC.

1.1.A.1 Akaike information criterion (AIC) The Akaike information criterion is an estimator of prediction error. It provides a means for model selection by estimating the relative quality of a statistical model given a set of data.

$$AIC = 2k - 2\ln\Big(\hat{L}\Big) \tag{AIC1}$$

where $\hat{L}$ is the maximized value of the likelihood function and k is the number of parameters.

1.1.A.2 Bayesian information criterion (BIC) Similarly, the Bayesian information criterion (also known as the Schwarz information criterion) is given by:

$$BIC = k\ln(n) - 2\ln\Big(\hat{L}\Big) \tag{BIC1}$$

where $\hat{L}$ and k are as above and n is the number of observations.

1.1.A.3 Log Liklihood Function (LLF) The Log Liklihood Function is the logarithm of the joint probability of the observed data,

$$\begin{aligned} l(\beta|X) &= \ln \prod_{i=1}^{N} p(y_i|X_i;\beta,\sigma^2) \\ &= \sum_{i=1}^{N} \ln p(y_i|X_i;\beta,\sigma^2) \\ &= \sum_{i=1}^{N} \ln \frac{\exp\Big\{\frac{-(y_i-\beta\cdot X_i)^2}{2\sigma^2}\Big\}}{\sqrt{2\pi\sigma^2}} = \sum_{i=1}^{N} \ln \frac{\exp\Big\{\frac{-e_i^2}{2\sigma^2}\Big\}}{\sqrt{2\pi\sigma^2}} \\ &= \sum_{i=1}^{N} \frac{-e_i^2}{2\sigma^2} - \frac{N}{2}\ln\big(2\pi\sigma^2\big) \end{aligned} \tag{LLF1}$$

To determine maximum value of equation (LLF1) we differentiate in terms of σ and solve for zero.

$$\begin{aligned} \frac{\partial l(\beta|X)}{\partial \sigma} &= 0 \\ \sum_{i=1}^{N} \frac{e_i^2}{\sigma^3} - \frac{N}{2\sigma} &= 0 \\ \hat{\sigma}^2 &= \frac{\sum_{i=1}^{N} e_i^2}{N} \end{aligned} \tag{LLF2}$$

Substituting (LLF2) into (LLF1) gives:

$$\begin{aligned} \ln\left(\hat{L}\right) &= l(\beta|X)\big|_{\sigma=\hat{\sigma}} \\ &= \sum_{i=1}^{N} \frac{-e_i^2}{2\hat{\sigma}^2} - \frac{N}{2}\ln\left(2\pi\hat{\sigma}^2\right) \\ &= \frac{-N}{2}(1 + \ln\left(2\pi\hat{\sigma}^2\right)) \end{aligned} \tag{LLF3}$$

1.1.B Stability check

To test the reliability of our results, a stability check will be made on any VAR(1) process fitting by testing that the eigenvalues of any coefficients lie within -1 and 1. See equation (VAR4).

1.2 Engle-Granger Procedure (EG)

The Engle Granger procedure is a method for determining and creating cointegrated processes. It constructs residuals (errors) based on a linear regression. It then tests the residuals to see if unit roots are present using an ADF test.

Step 1 - Check for stationarity

Fit one curve against another using OLS,

$$p_t^A = \begin{bmatrix} \mathbf{1} & p_t^B \end{bmatrix} \beta + e_t$$

then test the fitted residual, $\hat{e_t}$ for stationarity (using ADF)

$$\hat{e_t} = p_t^A - \hat{\beta}_1 p_t^B - k \tag{EG1}$$

where $\hat{\beta}_1$ is the estimate of the 2nd column of β and k is a constant.

To test for stationarity we use the Augmented Dickey Fuller test with lag 1. If the residual is non-stationary then no long-run relationship exists. The critical values used are from (MacKinnon, 2010).

In (Engle & Granger, 1987) and (Hendry & Juselius, 1999) no constant term is used for the ADF regression. This is also confirmed in (Zivot & Wang, 2003).

Step 2 - Equilibrium Correction Model (ECM)

Note that

$$\begin{aligned} p_t^A - p_{t-1}^A &= \begin{bmatrix} \mathbf{1} & p_t^B \end{bmatrix} \beta + e_t - \begin{bmatrix} \mathbf{1} & p_{t-1}^B \end{bmatrix} \beta - e_{t-1} \\ \Delta p_t^A &= \Delta p_t^B \beta + e_t - e_{t-1} \\ &= \Delta p_t^B \beta - (1-\alpha) e_{t-1} \end{aligned} \tag{EG2}$$

where $e_t = \alpha e_{t-1}$, and $\alpha - 1$ can be considered as the speed of correction towards equilibrium. It comes in very small moves with $1 - \alpha$ being very much smaller than one.

The 2nd term in (EG2) is a correction term. If positive then there's a downward correction in the current period and vice versa.

To determine the statistical significance of $1 - \alpha$ we use the lagged fitted residual $\hat{e}_{t-1}$ as the error correction in the above.

Steps 1 and 2 are repeated with the pair of assets in reverse. Only if both the normal and the reversed pair are statistically significant do we continue with the following step 3. In this case we choose the pair with the lowest t-value.

Step 3 - Calculate mean reversion parameters

Fit the stationary spread to the OU equation (OU1) where $Y_t = e_t$,

$$de_t = \theta(\mu_e - e_t)dt + \sigma_{eq} dW_t$$

which has the following solution (OU2) when starting from $t = \tau$ and noting residuals are 'mixed integrals' over Brownian Motion, i.e. $\epsilon_{t,\tau} = \sigma \int_{s=t}^{t+\tau} e^{\theta(s-t-\tau)} dW_s$,

$$e_{t+\tau} = e^{-\theta\tau} e_t + \mu_e(1 - e^{-\theta\tau}) + \epsilon_{t,\tau}$$

using VAR(1) to estimate parameters for lagged residuals and constant term would result in the following,

$$e_{t+\tau} = \beta_0 + \beta_1 e_t + \epsilon_{t,\tau}$$

then matching terms above we have

$$\begin{aligned} e^{-\theta\tau} &= \beta_1 \\ \theta &= -\ln(\beta_1)/\tau \end{aligned} \tag{EG3.1}$$

and

$$\begin{aligned} \mu_e(1 - e^{-\theta\tau}) &= \beta_0 \\ \mu_e &= \frac{\beta_0}{1 - \beta_1}. \end{aligned} \tag{EG3.2}$$

We calculate standard deviation using (OU3) as

$$\begin{aligned} \frac{\sigma_{OU}^2}{2\theta}(1 - e^{-2\theta\tau}) &= Var(\epsilon_{t,\tau}) \\ \sigma_{OU} &= \sqrt{\frac{2\theta\, Var(\epsilon_{t,\tau})}{1 - e^{-2\theta\tau}}} \end{aligned} \tag{EG3.3}$$

For trading bounds and to estimate P&L we use:

$$\sigma_{eq} \approx \frac{\sigma_{OU}}{\sqrt{2\theta}} \tag{EG3.4}$$

An indication of holding time is Half-life given by:

$$\tilde{\tau} = \frac{\ln 2}{\theta} \tag{EG3.5}$$

1.2.1 World indices denominated in USD

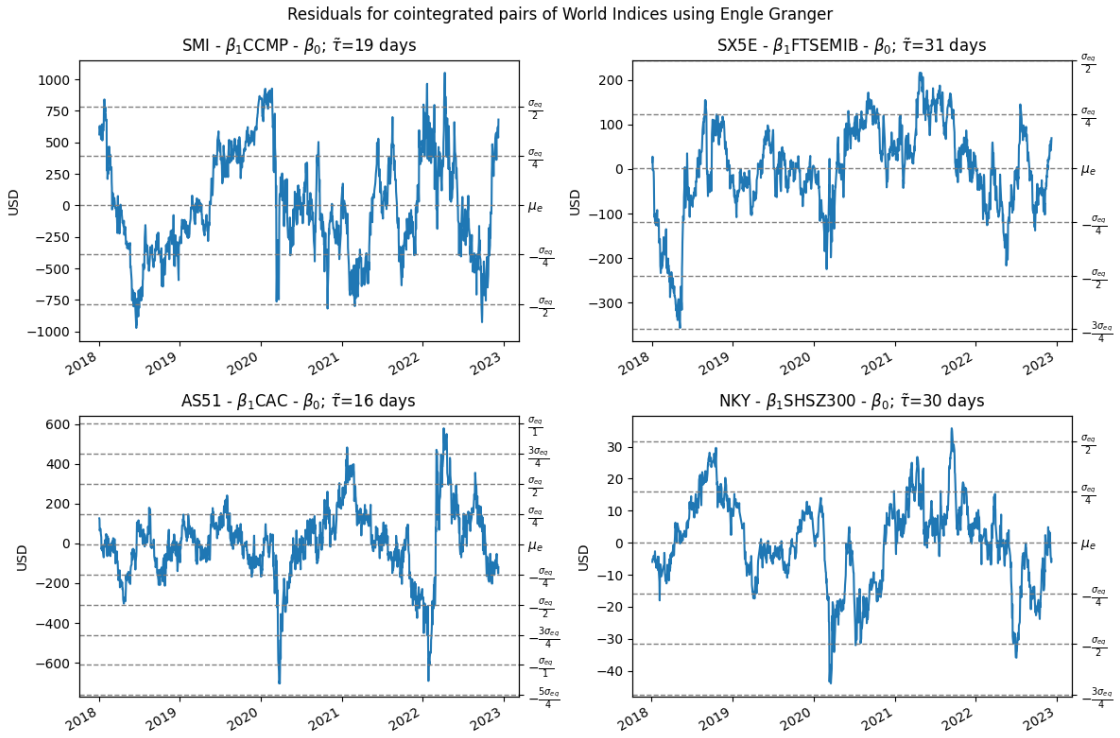
We run the EG test against pairs of World Indices each converted to USD.

The following pairs along with their calculated trade parameters all test as co-integrated. Also note that their OU VAR(1) fit tests as stable; as in section (1.1.B).

Table 2: Output from Engle Granger tests

pA	pB	beta_0	beta_1	theta	mu_e	sigma_eq	tau_tilde	pvalue	tvalue	stable
SMI	CCMP	5499.1629	0.5150	8.9648	2.2302	1565.5830	0.0773	0.0124	-3.8287	True
SX5E	FTSEMIB	469.1483	0.1414	5.6442	1.8634	481.4800	0.1228	0.0217	-3.6416	True
AS51	CAC	745.5020	0.6075	11.1406	-4.4586	606.6660	0.0622	0.0054	-4.0856	True
NKY	SHSZ300	53.8538	0.0389	5.8257	0.0397	63.4340	0.1190	0.0369	-3.4523	True

The following charts show the residuals over time for those cointegrated time series from above table.



1.3 Decide signals

We try suggested common approach; that is, enter on bounds $\mu_e \pm Z\sigma_{eq}$ and exit when e_t reverts to $\mu_e \pm r\sigma_{eq}$ (note that within the Python code we denote r as `radius`).

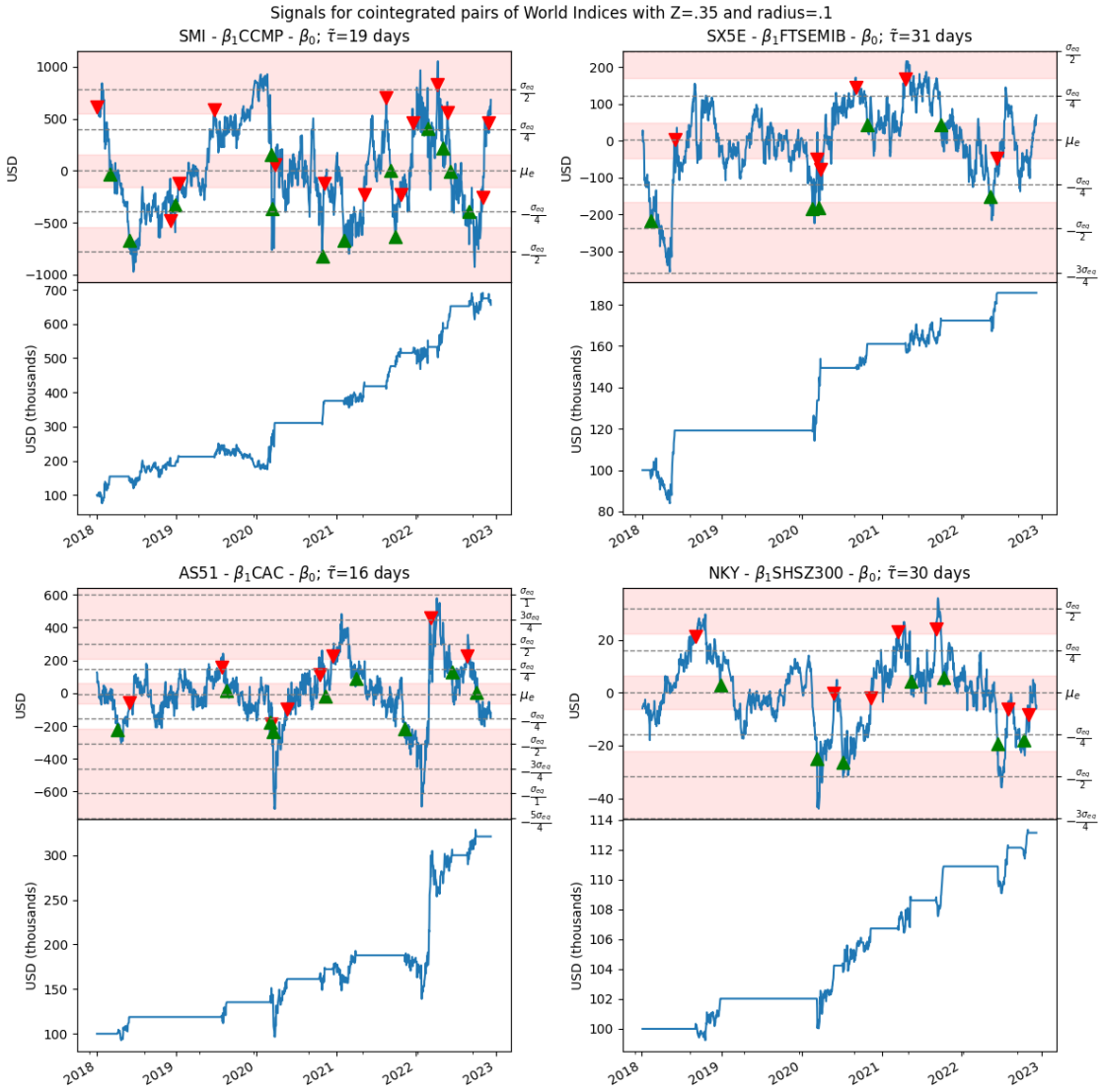
$$\text{strategy} = \begin{cases} \text{long} & \text{if } e_t < \mu_e - Z\sigma_{eq} \\ \text{short} & \text{if } e_t > \mu_e + Z\sigma_{eq} \\ \text{exit} & \text{if } |e_t - \mu_e| < r\sigma_{eq} \end{cases}$$

In our trade model, for each cointegrated pair, at previous market close, we calculate buy/sell signals by examining the residuals. This generates orders for execution on the following day. The executed order fixes the positive side of the spread at +100 units for a buy (-100 for a sell) and scales the negative side of the spread by β_1.

We considered using fixed notional for the combined spread but the implementation was too time consuming. Since we are using a fixed quantity we cannot easily compare pairs against one another using the absolute profit generated.

Also due to time constraints we chose not to introduce stop losses; this adds to some loss of realism.

The following charts show the trading strategy at play with green triangles representing buy trades and upside down red triangles showing sell trades. The light pink banding show the bounds where trades are triggered. The lower panels show accumulated value which includes cash and mark-to-market position value.



1.4 Optimize signals

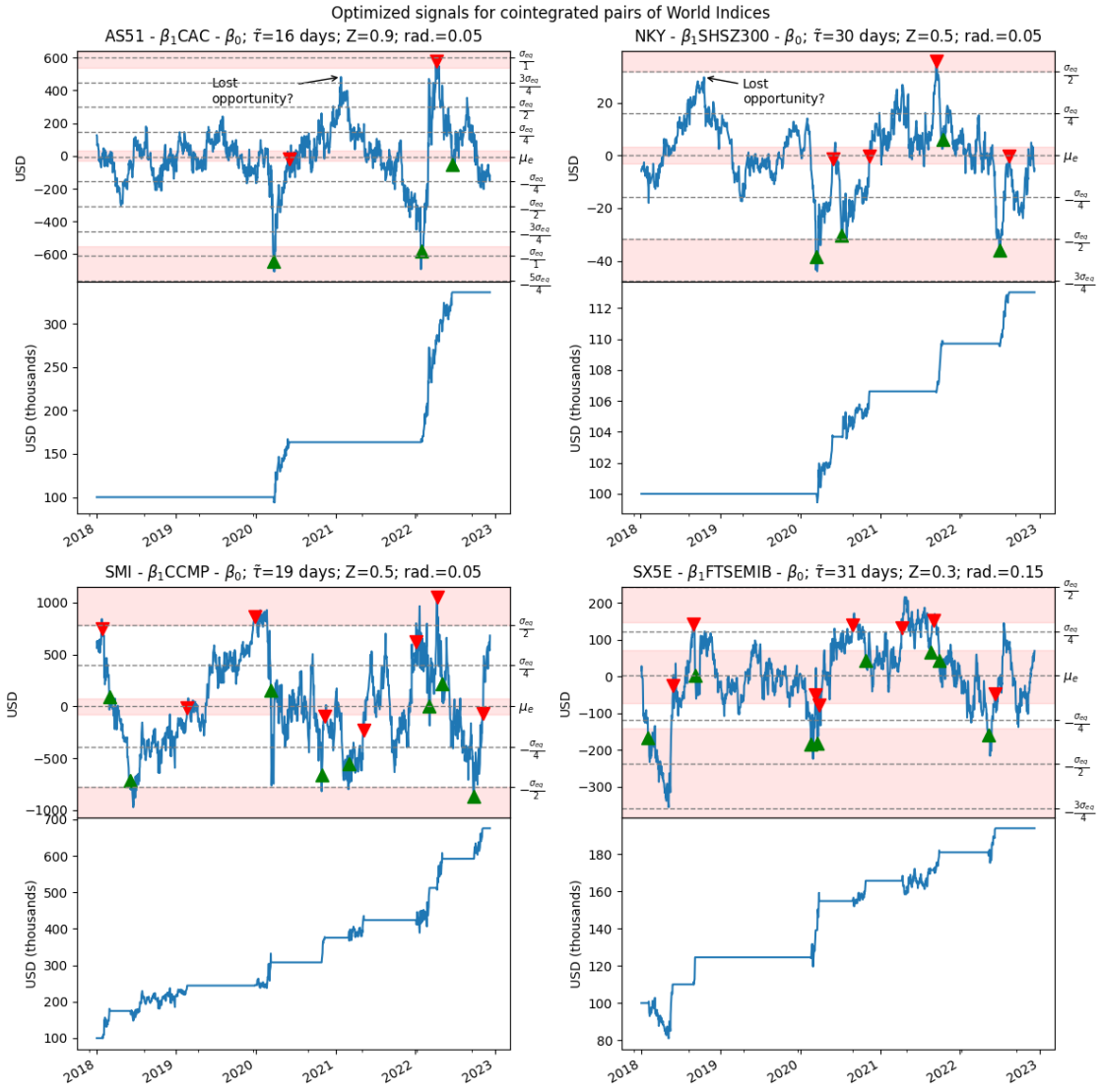
We test various combinations of Z and r and compute P&L, total number of trades and Sharpe Ratio.

It appears higher Sharpe Ratios occur for larger values of Z and higher profits typically for smaller values of Z. However, the higher profits come at the expense of a greater number of trades which might incur additional commissions and in some cases splippage.

Table 3: Output optimization

		Total PNL (1000s USD)				Total Trades				Sharpe Ratio (Ann)			
	pA	AS51	NKY	SMI	SX5E	AS51	NKY	SMI	SX5E	AS51	NKY	SMI	SX5E
	pB	CAC	SHSZ300	CCMP	FTSEMIB	CAC	SHSZ300	CCMP	FTSEMIB	CAC	SHSZ300	CCMP	FTSEMIB
Z	radius												
	0.05	236	14	674	101	25	14	29	14	0.76	1.08	0.79	0.92
0.30	0.10	232	13	545	93	34	14	33	14	0.78	1.01	0.70	0.92
	0.15	218	12	660	94	42	14	49	16	0.78	0.95	0.90	0.92
	0.05	210	13	576	22	10	8	16	2	0.56	1.37	1.02	0.49
0.50	0.10	184	13	450	25	10	8	16	2	0.51	1.34	0.94	0.56
	0.15	153	11	441	19	12	8	16	2	0.45	1.30	0.99	0.44
	0.05	239	0	0	28	8	0	0	2	0.89			0.72
0.70	0.10	212	0	0	31	8	0	0	2	0.84			0.80
	0.15	192	0	0	24	10	0	0	2	0.79			0.68
	0.05	236	0	0	0	6	0	0	0	1.07			
0.90	0.10	218	0	0	0	6	0	0	0	1.03			
	0.15	167	0	0	0	6	0	0	0	0.97			

We take those parameters with optimal Sharpe Ratio and plot them along with their book value as we did earlier. We note that for some strategies we appear to miss some opportunities as our Z was too high.



1.5 Vector Error Correction Model (VECM)

When the number of curves in our cointegration model is greater than two then our analysis must be based on VAR(p) and our ECM needs to be vectorized.

Using a similar derivation to (EG2) we have:

$$\Delta y_t = \Pi y_{t-1} + \sum_{k=1}^{p-1} \Gamma_k \Delta y_{t-k} + \epsilon_t \qquad \text{(VECM1)}$$

where each y_k is a column vector $\begin{bmatrix} y_{1,k} & y_{2,k} & \cdots & y_{n,k} \end{bmatrix}^T$, and

$$\Pi = \sum_{k=1}^{p} A_k - I$$
$$\Gamma_k = -\sum_{i=k+1}^{p} A_i.$$

1.5.1 Grangers Representation Theorem (GRT)

Grangers Representation Theorem (Engle & Granger, 1987) states that if Π has reduced rank $r \leq n-1$ then there exists $n \times r$ matrices α and β such that $\Pi = \alpha\beta^T$ and $\beta^T y$ is stationary $I(0)$. Each column of β is a cointegration vector and α is a parameter representing the speed of moving towards equilibrium.

1.5.2 Johansen's Maximum Likelihood Procedure

Π cannot be estimated using OLS and an alternative approach is Johansen's Maximum Likelihood Procedure. We follow the same steps as in (Jang & Ogaki, 2001) and (Diamond, 2022). Rather than implementing the procedure ourselves we use statsmodels' `coint_johansen` implementation.

Step 1 - Residual Product Moment Matrices Regress both Δy_t and y_{t-1} as follows:

$$\Delta y_t = \sum_{k=1}^{p-1} \Gamma_k \Delta y_{t-k} + R_{0t}$$
$$y_{t-1} = \sum_{k=1}^{p-1} \Gamma_k \Delta y_{t-k} + R_{kt}$$

and form residual product moment matrices using residuals R_{0t} and R_{kt} as follows:

$$S_{ij} = \frac{1}{T} \sum_{t=1}^{T} R_{it} R_{jt}, \quad i, j = 0, k$$

Step 2 - Eigenvalues & Eigenvectors The MLE of β in GRT is obtained as the eigenvectors of n rows corresponding to r largest eigenvalues from solving,

$$|\lambda S_{kk} - S_{k0} S_{00}^{-1} S_{0k}| = 0.$$

The eigenvectors are normalized such that $V^T S_{kk} V = I$.

Step 3 - Determine rank r To determine r in Step 2 we perform either a maximum eigenvalue statistic,

$$\lambda_{\max} = -T \ln(1 - \lambda_{r+1})$$

or trace statistics test

$$\lambda_{\text{trace}} = -T \sum_{i=r+1}^{n} \ln(1 - \lambda_i)$$

Step 4 - Estimate parameters Given β we calculate vector errors αe_t and estimate remaining parameters in (VECM1) by regression ΔY_t in

$$\Delta y_t = \alpha e_{t-1} + \sum_{k=1}^{p-1} \Gamma_k \Delta y_{t-k} + \epsilon_t.$$

1.5.3 Johansen's ML Procedure Results

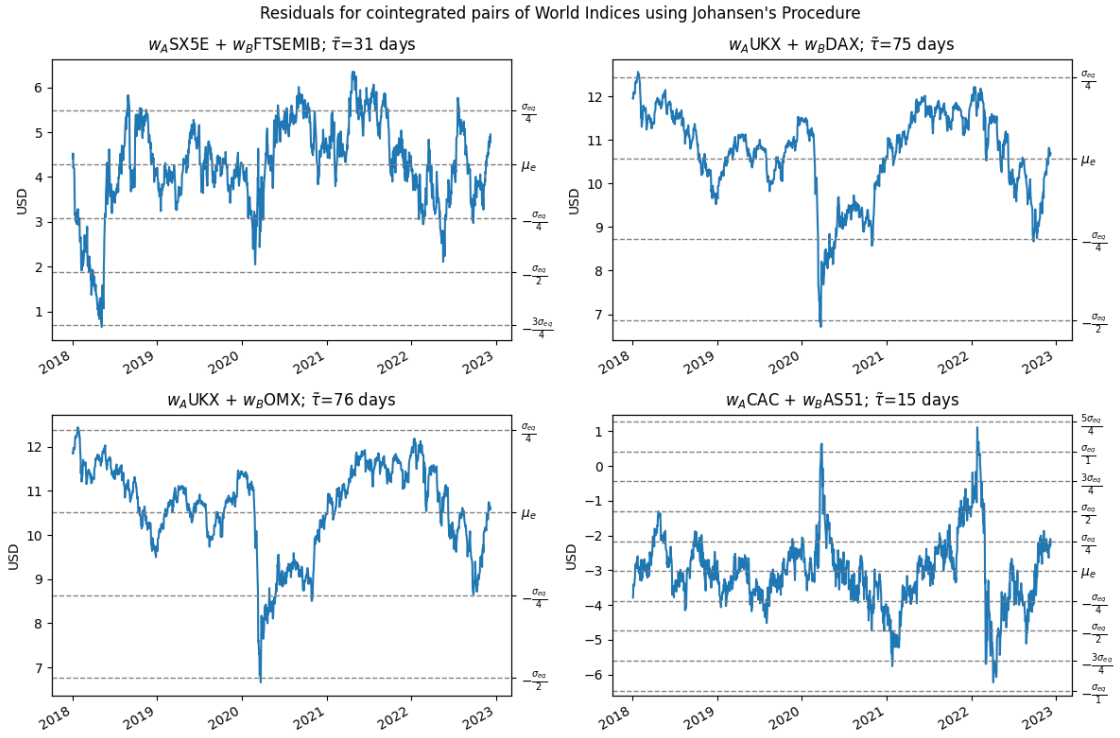
The following was calculated using statsmodels' `coint_johansen` routine. We filtered for those pairs whose test statistics exceeded the critical value of 90%. The residuals of the pairs was then fitted to an OU process as in (EG3) to calculate various trade parameters.

Table 4: Output from Johansen's Procedure

pA	pB	wA	wB	theta	mu_e	sigma_eq	tau_tilde	stable
SX5E	FTSEMIB	0.0100	-0.0014	5.6686	4.2849	4.7945	0.1223	True
UKX	DAX	0.0010	0.0001	2.3224	10.5780	7.4324	0.2985	True
UKX	OMX	0.0011	0.0046	2.3013	10.5077	7.4670	0.3012	True
CAC	AS51	0.0036	-0.0057	11.3849	-3.0210	3.4435	0.0609	True

The following charts show the residuals over time for those cointegrated time series from the above table.

Note that the residuals for the pair SX5E/FTSEMIB are similar to those determined using the EG method only differing by a scaling factor and intercept. Also the pair CAC/AS51 has similar but inverted residuals to those determined by EG method.



2. Backtesting

2.4 Drawdown & Sharpe Ratio

The pair NKY/SHSZ300 appeared to offer a higher Sharpe Ratio with consistent returns across a range of trading parameters. In particular the highest Sharpe Ratio was for $Z = 0.5$ and $r = 0.05$. We use 3rd party library pyfolio's `create_simple_tear_sheet` method to produce a plots of rolling Sharpe Ratio and drawdown.

This strategy didn't trigger an order until around 2020 Q1 where the Sharpe ratio started very poor and there was significant drawdown.

Table 5: Strategy parameters for NKY/SHSZ300 with Z=0.5, r=0.05.

	Parameters
Start date	2018-01-05
End date	2022-12-09
Total months	53

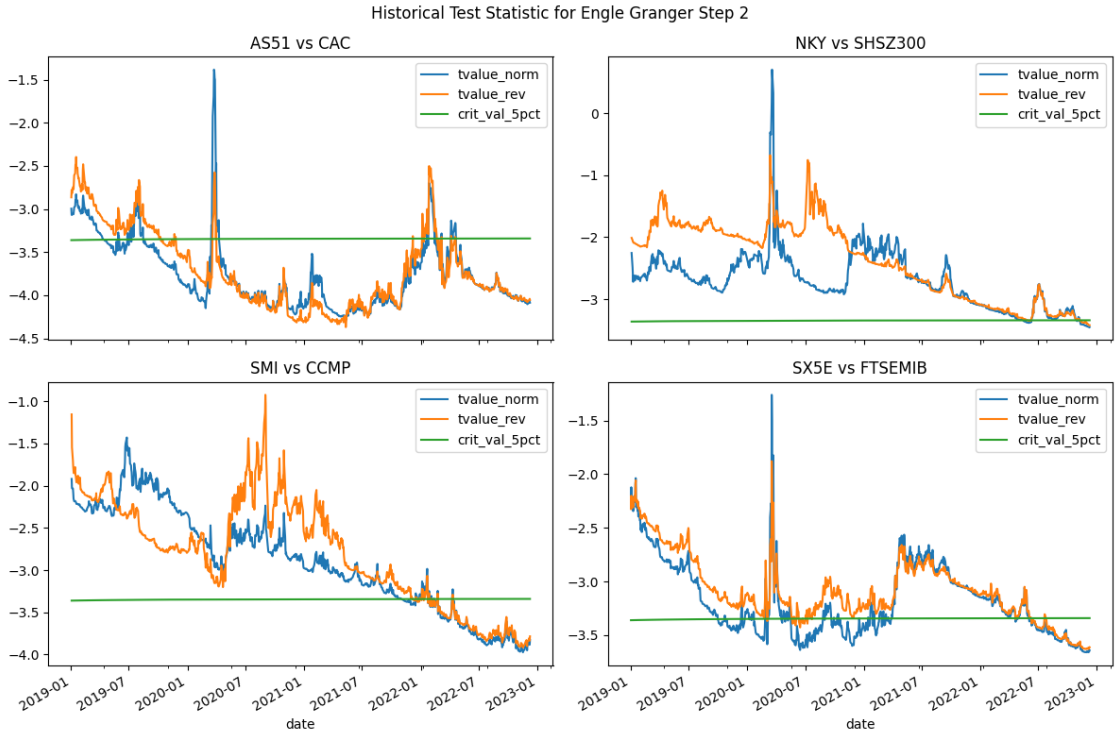
Table 6: Strategy performance statistics for NKY/SHSZ300 with Z=0.5, r=0.05.

	Backtest
Annual return	2.7%
Cumulative returns	12.9%
Annual volatility	2.0%
Sharpe ratio	1.37
Calmar ratio	1.68
Stability	0.89
Max drawdown	-1.6%
Omega ratio	1.87
Sortino ratio	2.53
Skew	2.89
Kurtosis	39.35
Tail ratio	4.98
Daily value at risk	-0.2%

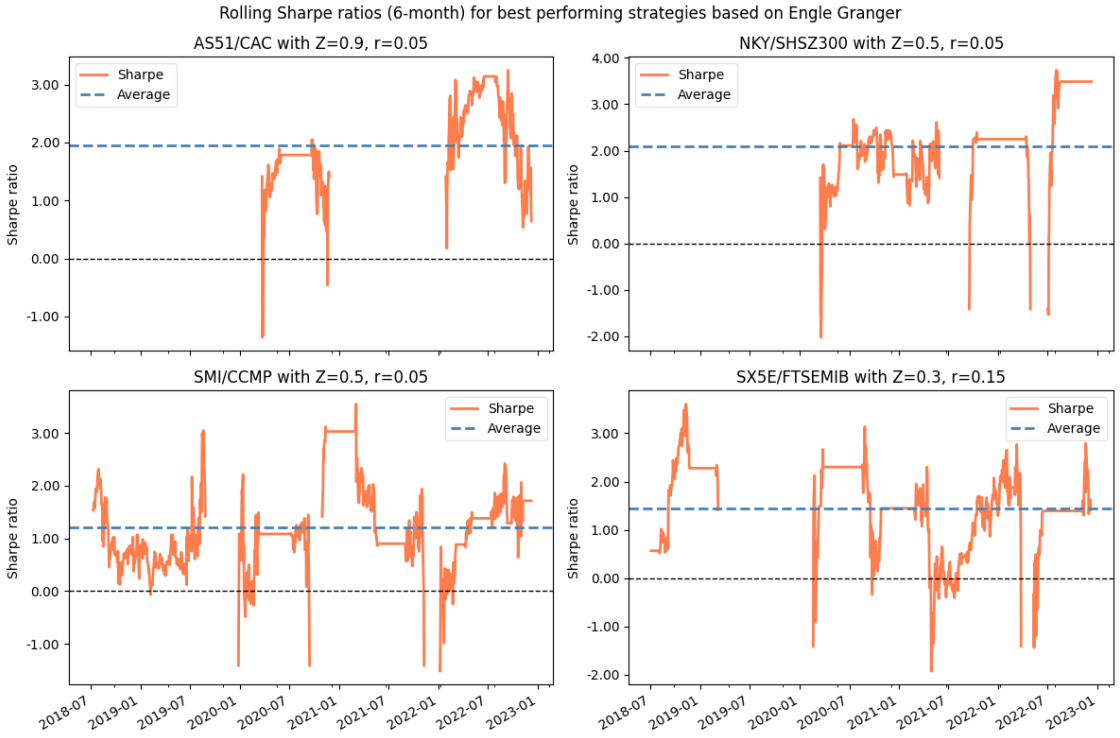

Backtesting plots for NKY/SHSZ300 with Z=0.5, r=0.05
Cumulative returns
Backtest
Cumulative returns
1.12
1.10
1.08
1.06
1.04
1.02
1.00
2018
2019
2020
2021
2022
2023
Rolling Sharpe ratio (6-month)
Sharpe
Average
Sharpe ratio
4.00
3.00
2.00
1.00
0.00
-1.00
-2.00
2018
2019
2020
2021
2022
2023
Underwater plot
Drawdown
0%
-0%
-0%
-1%
-1%
-1%
-1%
-1%
-1%
-2%
2018
2019
2020
2021
2022
2023

2.5 Re-estimation

Below we plot historical EG step 2 test statistics for the 4 pairs that were previously selected. We use an expanding time window approach to mimic the calculation one might make in a real world scenario. Each plot shows the statistic for normal and reverse configuration along with the 5% critical line.



We note that all pairs do not pass the cointegration test all the time. There also appears to be an inverse relationship between cointegration and rolling 6 month Sharpe ratio. When cointegration breaks down we see that the Sharpe ratio drops often to below 0.



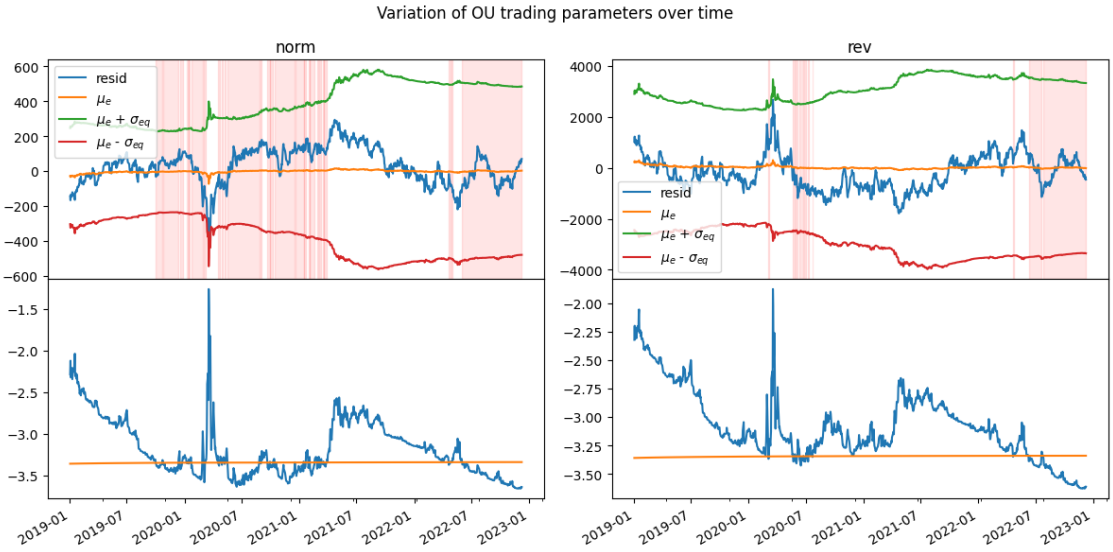
There are some models in the literature that suggest a connection between co-integration and Sharpe Ratio. In (Farago & Hjalmarsson, 2019) they find that if stock returns are fairly weakly correlated across time, cointegration implies very high Sharpe ratios. They do this by modelling the residual (or spread) as a Vector Moving Average (VMA) process.

2.5.1 Dynamic cointegration & regime change

By examining the pair SX5E/FTSEMIB's historic cointegration statistic we can see it goes through different regimes of having cointegration or not. Perhaps we could capture this and improve our trading strategy.

We again use an expanding time window and plot how μ_e and σ_{eq} evolve. We note that σ_{eq} can differ significantly over different cointegration periods. This suggests that it might be impractical to use for entry and exit signals especially if we choose a factor Z that is constant over time.

We will stick to using EG only in the normal direction now. It seems impractical and difficult to switch direction based on the most significant statistic (normal or reversed). This is either due to switching the polarity of the pair trade, or deciding which cutoffs to use if one opens a trade in the normal direction then the cointegration becomes more significant in the reverse direction.



We devise an enhanced trading strategy that uses a rolling standard deviation of the current residual (Bollinger Bands) to determine entry and exit signals.

The strategy calibrates with the market on a weekly basis and determines if cointegration exists and, if so, the relevant trading parameters. It then raises open/close orders based on whether our residual breaches the Bollinger Bands. The strategy exits a trade if it's within the r threshold like before.

We attempt to close a trade gracefully if cointegration breaks down. That is, once cointegration stops we only close the trade once it reaches the interval around μ_e. Clearly, this adds to a loss of realism as we had when we didn't introduce a stop loss mechanism previously.

We optimize for Z and r as before but also optimize for Bollinger Band window size (in Python code we denote as `std_win`).

The table below shows the results of this optimization.

Table 7: Output from dynamic cointegration optimization

Z	radius	std_win	Total PNL (1000s USD)	Total Trades	Sharpe Ratio (Ann)
0.5	0.05	30	39.9	27	0.24
0.5	0.05	40	43.3	29	0.25
0.5	0.05	50	31.5	23	0.18
0.5	0.10	30	27.8	33	0.16
0.5	0.10	40	34.7	31	0.20
0.5	0.10	50	30.7	27	0.18
0.5	0.15	30	30.4	33	0.18
0.5	0.15	40	37.7	31	0.22
0.5	0.15	50	33.2	27	0.19
1.0	0.05	30	51.2	17	0.28
1.0	0.05	40	39.8	17	0.22

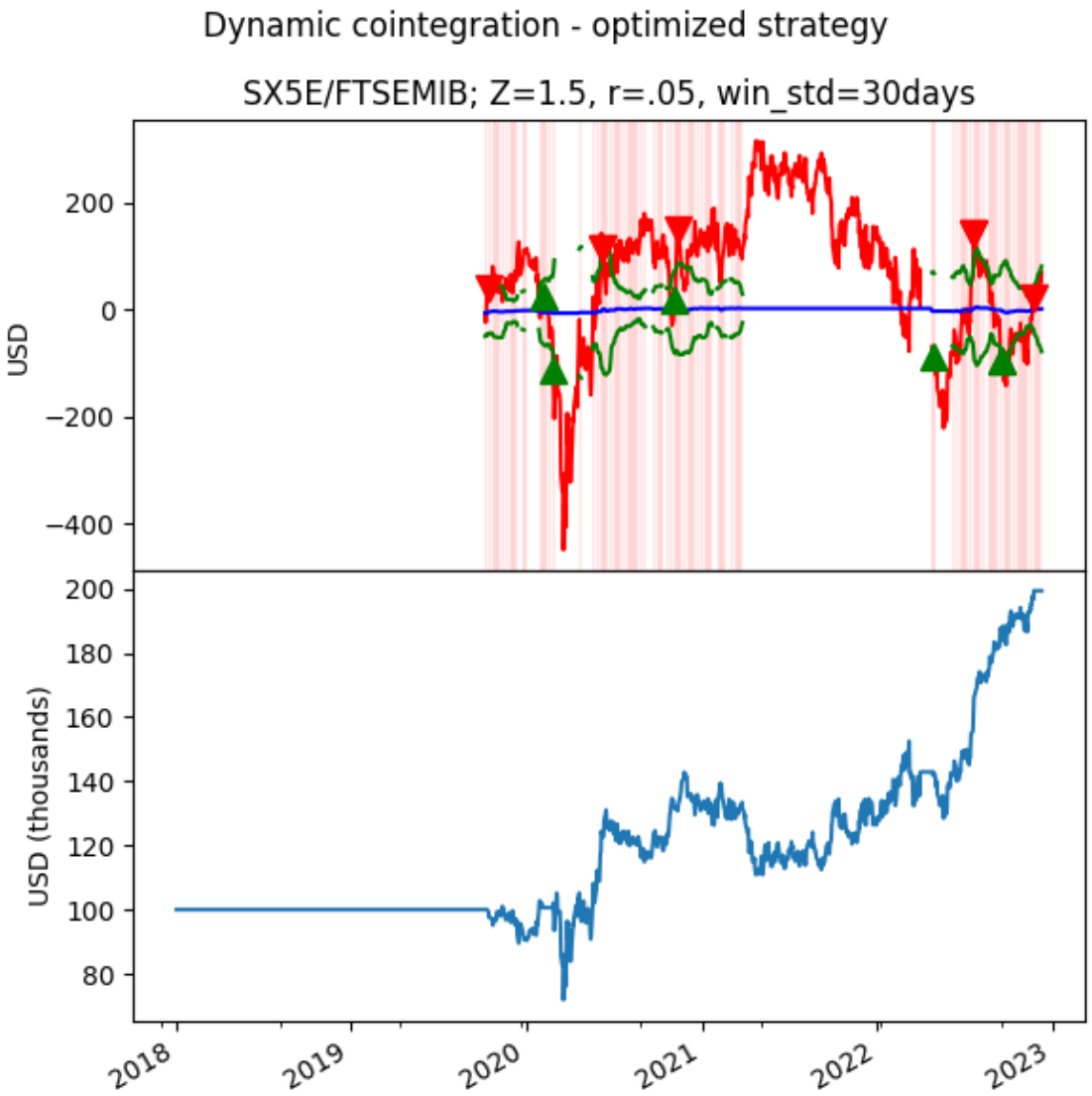
Z	radius	std_win	Total PNL (1000s USD)	Total Trades	Sharpe Ratio (Ann)
1.0	0.05	50	47.4	17	0.27
1.0	0.10	30	36.6	19	0.20
1.0	0.10	40	32.4	17	0.19
1.0	0.10	50	46.0	17	0.26
1.0	0.15	30	42.7	19	0.24
1.0	0.15	40	35.5	17	0.20
1.0	0.15	50	48.5	17	0.28
1.5	0.05	30	99.4	15	0.54
1.5	0.05	40	64.1	15	0.36
1.5	0.05	50	66.8	15	0.42
1.5	0.10	30	53.9	15	0.32
1.5	0.10	40	55.8	15	0.33
1.5	0.10	50	57.7	15	0.35
1.5	0.15	30	59.9	15	0.36
1.5	0.15	40	58.8	15	0.35
1.5	0.15	50	60.2	15	0.36

We note the optimal Sharpe Ratio of .54 is when $Z = 1.5$, radius $= .05$ and std_win $= 30$.

The chart below shows how the strategy plays out. The vertical light red bandings show when cointegration is active. As in similar charts, green triangle represent long trades and upside red triangles represent short trades. Although note that multiple trades can occur on a single day switching the entire position from long to short or vice-versa.

Cointegration switches between active and inactive quickly over the various calibration dates. This is why the light red cointegration bandings are very stripy.

The residual curve is only captured while cointegration is active or a position is open. It is clear that the residual curve is displayed while cointegration is inactive, this demonstrates that a position is not being closed in a timely fashion.



2.5.2 Additional condition on threshold

It is clear from the trade plots that when a buy or sell signal is generated by the algorithm, it doesn't necessarily occur at the local maximum/minimum of the mean reverting process.

One possible solution is to wait until the residual reaches a peak (or trough) and falls (or climbs) for a number of days before opening an order.

To do this we can look at the n day rolling maximum (or minimum) and compare it against the $n-1$ day rolling maximum (or minimum). If the value in our smaller window is below (or above) that of the larger window then we trigger a sell (or buy) order.

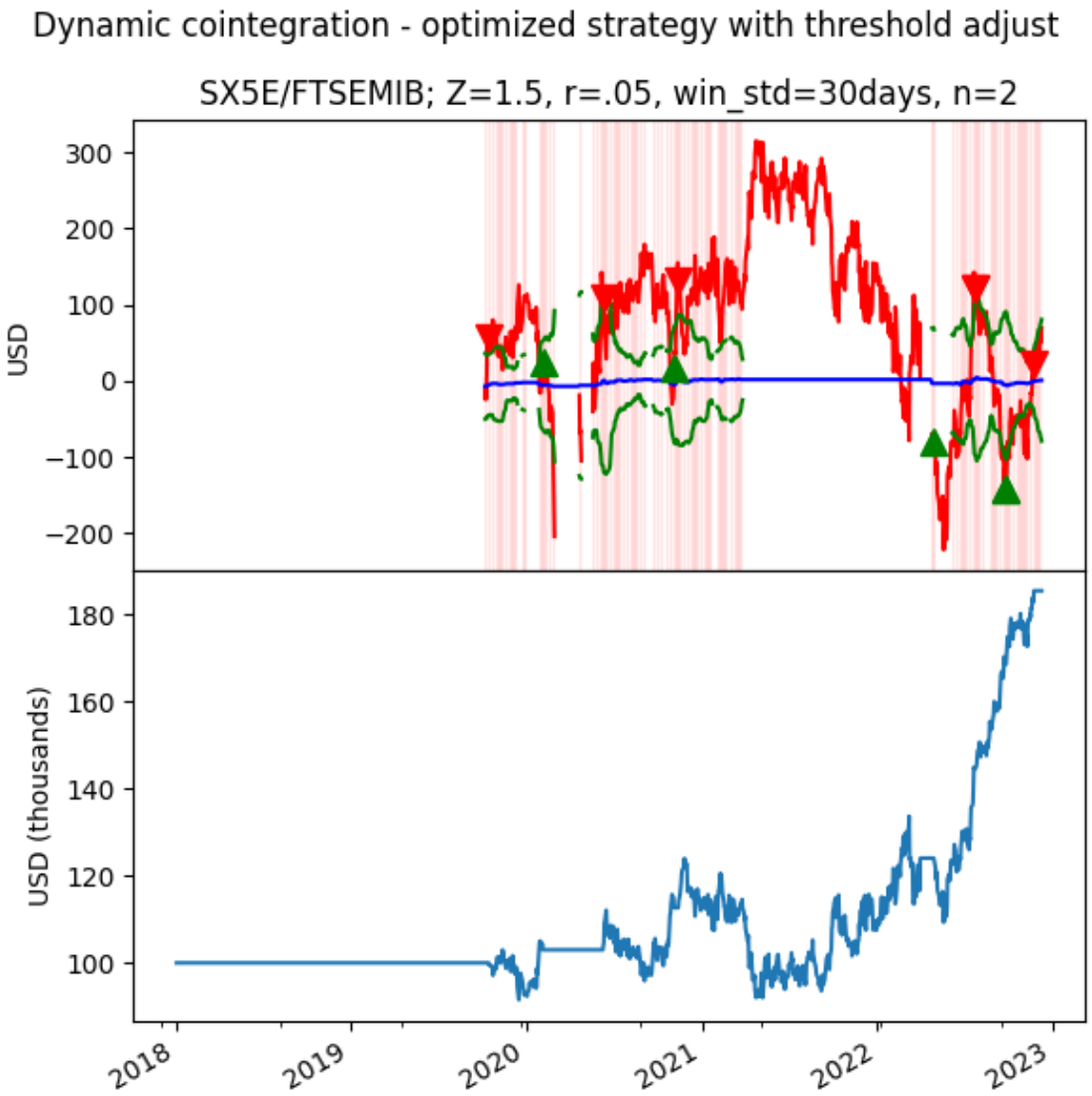
$$\text{strategy} = \begin{cases} \text{long} & \text{if } e_t < \mu_e - Z\sigma_{win} \text{ and } \min_{i=t-n+1}^{t} e_i > \min_{i=t-n}^{t} e_i \\ \text{short} & \text{if } e_t > \mu_e + Z\sigma_{win} \text{ and } \max_{i=t-n+1}^{t} e_i < \max_{i=t-n}^{t} e_i \\ \text{exit} & \text{if } |e_t - \mu_e| < r\sigma_{win} \text{ or cointegration fails} \end{cases}$$

Table 8: Output from dynamic cointegration rolling min/max threshold optimization

n	Total PNL (1000s USD)	Total Trades	Sharpe Ratio (Ann)
2	85.4	12	0.58
3	60.6	12	0.42
4	58.1	12	0.41

We note the optimal Sharpe Ratio of .58 is when $n = 2$ which is a small improvement over the strategy without additional threshold logic (.54).

The chart below shows how this enhanced strategy plays out.



2.6 Machine Learning

A machine learning approach might be useful in determining where the peaks and troughs appear in our residual curve.

With `scikit-learn` (Pedregosa et al., 2011) we test several classification techniques applied to the residual generated from our EG approach earlier.

$$e_t = \text{SX5E} - 0.1414 \times \text{FTSEMIB} - 469.1483$$

2.6.1 Features & Target

The following features were chosen. Due to symmetry it was decided that we focus on features generated by the absolute value of the residual. This simplifies the training to only search for maximums.

Label	Formula	Description
s_abs	$\lvert e_t \rvert$	absolute value of residual
sa_diff{d}	$\lvert e_{t-d} \rvert$	lagged residuals over d days; d = 1, 2, 3
sa_rmean{d}	$\frac{1}{d}\sum_{i=0}^{d-1} e_{t-i}$	simple moving average over d days; d = 5, 10, 15, 20
sa_rmax{d}	$\max_{i=0}^{d-1} e_{t-i}$	maximum over d days; d = 2, 3, 4, 5, 6, 7, 8
sa_rmin{d}	$\min_{i=0}^{d-1} e_{t-i}$	minimum over d days; d = 2, 3, 4, 5, 6, 7, 8
s_rstd{d}	$\sqrt{Var(e_{t-i}\lvert_{i=0}^{d-1})}$	rolling standard deviation over d days; d = 5, 10, 15, 20

The target was chosen as the maximum or minimum between intersections of the axis where $e_t = 0$. This is a simplification as ideally the axis should be were $e_t = \mu_e$. Note that in our absolute residual view these will all be mapped to maximums.

2.6.2 Pipeline

We created a pipeline with 3 steps:

Table 10: Pipeline steps for peak classification training

Step	Name	Description
1	scaler	`StandardScaler` to convert features to z scores.
2	feature_selector	`SelectKBest` with `mutial_info_classif` to removed unnecessary features.
3	classifier	One of `KNeighborsClassifier`, `SVC`, `DecisionTreeClassifier`, `RandomForestClassifier`, `MLPClassifier`, `AdaBoostClassifier` or `GaussianNB`

Many of the classifier algorithms are described in (Wilmott, 2019).

`KNeighborsClassifier` is a pattern recognition algorithm that stores and learns from training data points by calculating how they correspond to other data in n-dimensional space. It aims to find the k closest related data points in future, unseen data.

`SVC` assigns hyperplanes that best separate classes. The best hyperplanes are those with the largest distance between classes.

`DecisionTreeClassifier` is a supervised algorithm used to build models like the structure of a tree. It classifies data into finer and finer categories "from tree trunk, to branches to leaves".

`RandomForestClassifier` algorithm consists of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.

`MLPClassifier` is a multi-layer perceptron classifier. It is a feedforward artificial neural network. It uses backpropagation to learn a function relating the feature set to the target. It consists of at least 3 layers of neurons (input, at least one hidden and an output layer).

`AdaBoostClassifier` is short for adaptive boosting. Boosting combines weaker learning algorithms such as low depth decision trees for more powerful learning. In particular, during each learning iteration, Adaboost assigns weights on the data samples based on the error of each sample.

2.6.3 Cross Validation

We apply cross validation to prevent overfitting by splitting our timeseries into 5 growing pairs of sets, a training set and a test set (expanding window). We use `TimeSeriesSplit` class here. The training and test set are separated by a gap of 5 days.

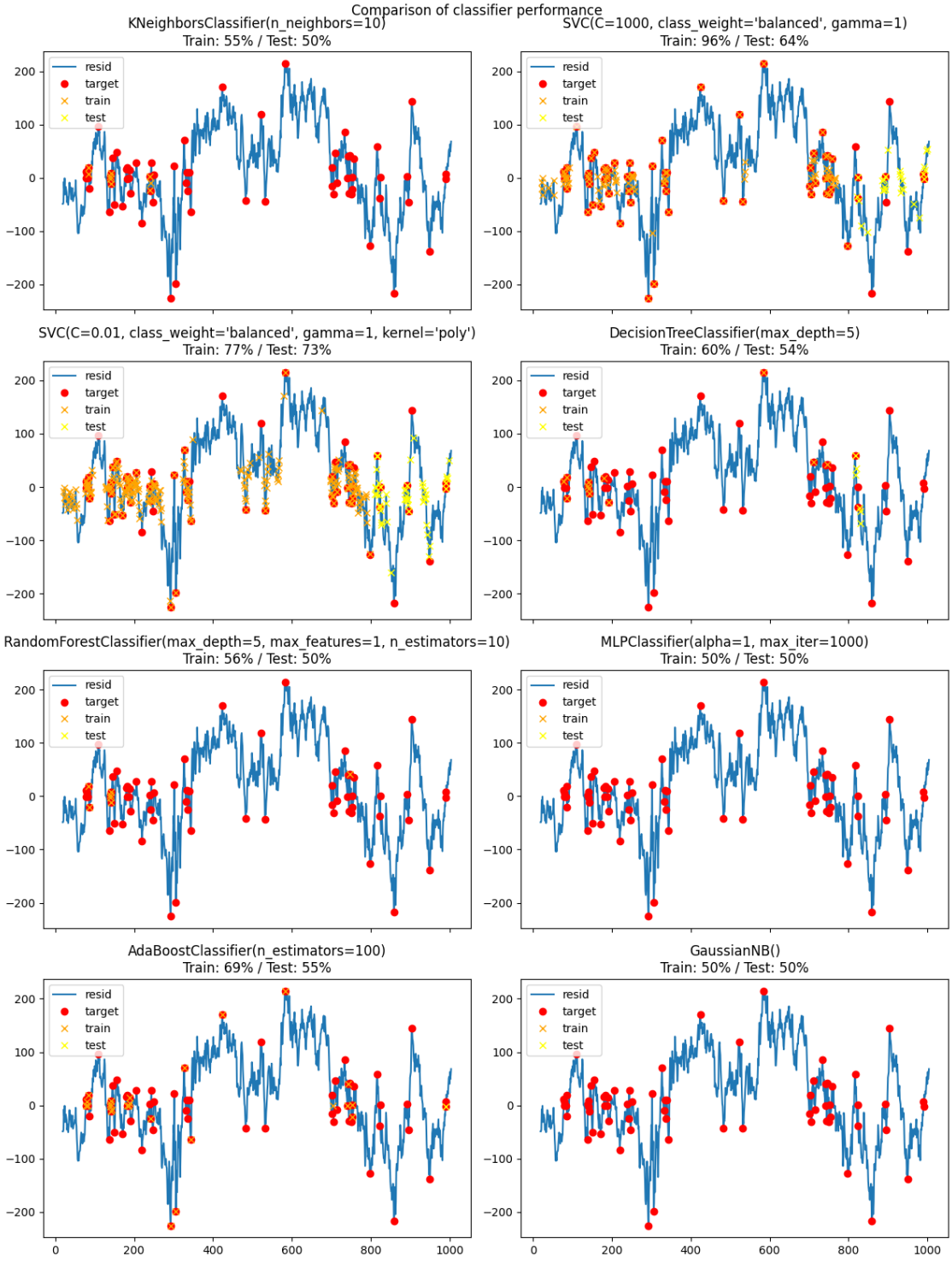
We then perform a grid search using `GridSearchCV` and ranging over a number of variation of parameters for depending on the classifier type.

2.6.4 Classification results

We chart the results of our classification study below. In each sub chart the residual is plotted with the target set marked with red circles. The training predictions are then marked with orange crosses and test predictions in yellow.

Since the target set of peaks and troughs is sparse, it is easy for a prediction to just return `False` for each time node and gain a high score so we use `balance_accuracy_score` which is designed to deal with imbalanced datasets.

From a visual inspection one can see that SVC classification performs better. In particular with rbf (default) and poly kernels. Decision Tree and AdaBoost appear to follow in performance.


Comparison of classifier performance
KNeighborsClassifier(n_neighbors=10)
Train: 55% / Test: 50%
SVC(C=1000, class_weight='balanced', gamma=1)
Train: 96% / Test: 64%
SVC(C=0.01, class_weight='balanced', gamma=1, kernel='poly')
Train: 77% / Test: 73%
DecisionTreeClassifier(max_depth=5)
Train: 60% / Test: 54%
RandomForestClassifier(max_depth=5, max_features=1, n_estimators=10)
Train: 56% / Test: 50%
MLPClassifier(alpha=1, max_iter=1000)
Train: 50% / Test: 50%
AdaBoostClassifier(n_estimators=100)
Train: 69% / Test: 55%
GaussianNB()
Train: 50% / Test: 50%
resid
target
train
test

2.6.5 Further development

Due to time constraints I didn't have time to implement a trading strategy to include a classifier such as SVC. Also using the fitted models as they are, would magically perform perfectly within the train data regions (for SVC in particular).

Also our feature set only used Close prices and didn't take advantage of other data points such as High, Low, Open and Volume.

Ideally, one might train with more real world data.

3.0 Conclusion

Cointegration appears to be a highly effective method for trading pairs or baskets of assets.

The Engle Granger procedure is a robust method for cointegration testing. The Johansen ML procedure appears to improve on EG's approach, in particular one does not need to choose a dependent variable and test in both directions (forward and reverse). Furthermore, Johansen's method is more suitable for detecting cointegrated baskets of assets. When testing for cointegration on pairs of assets perhaps both approaches should be used.

Care must be taken as statistical tests might find cointegration where there isn't any. In particular, one must have a supporting economic theory to why assets are cointegrated.

Care must also be taken as cointegration can be a transient process. One must regularly test if cointegration still exists and if not take remedial action such setting stop losses.

Having a cointegrated process which is mean reverting does not give information on where the peaks and troughs of the process will be. The best information in this regard is σ_{eq} and τ (half life). These parameters are themselves only estimates. There is scope for further exploration here perhaps with machine learning techniques such as classifiers or reinforcement learning, or perhaps with more traditional technical analysis.

4.0 Appendix

4.1 Code Summary

The following functions were coded in this project. All numerical implementations were tested against `statsmodels` equivalents (Seabold & Perktold, 2010). The tests are available inline within the accompanying Jupyter notebook.

Table 11: Summary of numerical techniques coded

Function	Description
`OLS_fit`	Implements Ordinary Least Squares method as detailed in report. Takes a vector y and matix X and returns a named tuple with parameters, residuals, stderr, tvalues, pvalues, eigen values, aic and bic information criteria. The method was testing against statsmodels' OLS.
`VAR_fit`	Implements Vector Autoregressive Process fitting as detailed in report. Takes a matrix data and lag p and returns a named tuple with beta, residuals, covariance matrices, stderr, tvalues, pvalues, aic and bic. The method was testing against statsmodels' VAR.

Function	Description
`ADF_test`	Implements Augmented Dickey Fuller Test as detailed in report. Takes a vector process y and returns a named tuple with underlying OLS result, ADF test value, pvalue and critical values. If scan_ic argument is "aic" or "bic" then will scan a range of lags and determine the one with the lowest information criteria. The method was testing against statsmodels' adfuller function.
`EG_test`	Implements Engle Granger Test as detailed in report. Takes two timeseries p_A, p_B and returns a named tuple with underlying OLS result, ADF result, error correction OLS result, ADF pvalue and ADF critical values. The method was testing against statsmodels' coint function which also uses EG test.
`OU_TP_fit`	Fits Ornstein–Uhlenbeck process as detailed in report. Takes I(0) timeseries resid and returns a named tuple with trade parameters tau, theta, sigma, mu_e, sigma_ou, sigma_eq, sigma_ou and VAR(1) stability flag.

4.2 Environment

This PDF was generated using a Jupyter Notebook running Python 3.10 on Ubuntu 22. The set up separates the Jupyter server enviroment from the individual kernels. The kernel environment was set up with the following pyproject.toml contents using `poetry`.

```
[tool.poetry]
name = "cqf-fp"
version = "0.1.0"
description = ""
authors = ["Blair Azzopardi <blairuk@gmail.com>"]
readme = "README.md"
packages = [{include = "cqf_fp"}]

[tool.poetry.dependencies]
python = ">=3.10,<3.11"
scipy = "^1.10.0"
statsmodels = "^0.13.5"
matplotlib = "^3.6.3"
pyfolio-reloaded = "^0.9.5"
scikit-learn = "^1.2.0"
yabte = "^0.1.2"
tabulate = "^0.9.0"
ipykernel = "^6.20.2"
more-itertools = "^9.0.0"
jinja2 = "^3.1.2"
ipylab = "^0.6.0"


[build-system]
requires = ["poetry-core"]
build-backend = "poetry.core.masonry.api"
```

4.2.1 New Python backtesting package - yabte

During this project and from past experience I felt existing Python backtesting modules had some issues such as lack of multiple asset support (backtesting.py) or being difficult to inline data (zipline). The new package is called `yabte`, an acronym for Yet Another BackTesting Engine and can be found on PyPI at https://pypi.org/project/yabte/.

References

[1] Diamond, D. R. (2022). *CQF June 2022 M6 L8 Cointegration.*

[2] Engle, R., & Granger, C. (1987). *Co-Integration and Error Correction: Representation, Estimation, and Testing.*

[3] Farago, A., & Hjalmarsson, E. (2019). Stock Price Co-Movement and the Foundations of Pairs Trading. *Journal of Financial and Quantitative Analysis*, *54*(2), 629–665. https://doi.org/10.1017/S0022109018000789

[4] Hendry, D. F., & Juselius, K. (1999). *Explaining Cointegration Analysis: Part I.*

[5] Jang, K., & Ogaki, M. (2001). *User Guide for Johansen's Method.*

[6] MacKinnon, J. G. (2010). *Critical Values for Cointegration Tests.* 19.

[7] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

[8] Seabold, S., & Perktold, J. (2010). *statsmodels: Econometric and statistical modeling with python.*

[9] Wilmott, P. (2019). *Machine learning: an applied mathematics introduction* (First edition). Panda Ohana Publishing.

[10] Zivot, E., & Wang, J. (2003). Cointegration. In E. Zivot & J. Wang, *Modeling Financial Time Series with S-Plus®* (pp. 415–460). Springer New York. https://doi.org/10.1007/978-0-387-21763-5_12