FINAL PROJECT TUTORIAL

Pairs Trading

Multivariate Cointegration

Dr Richard Diamond
CQF ARPM

July 2021

A. Cointegration in paris and initial trading signals

We will review **Python Lab** with attention to:

(a) matrix form of regression, (b) Engle-Granger correction equations, and

(c) OU process fitting on mean-reverting spread.

Full OU fitting includes **sigma_eq** (separate notebook, screen only).

B. Scanning markets/terms structures with multivariate cointegration

We will review **R code** that utilizes package (library) *urca* – the qualified implementation of Johansen Procedure for multivariate cointegration testing, and estimation with trends.

ca.jo(data, ecdet = "const", type="eigen", K=*Nlags*, spec="longrun")

Statistical Arbitrage

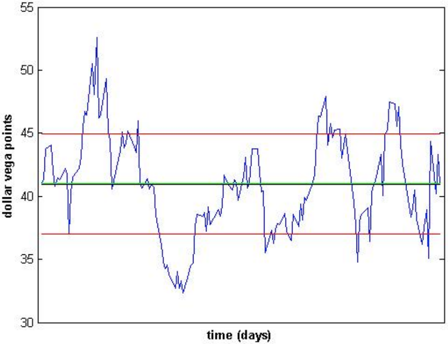
- Signal Generation from Cointegrated Spread
- Fitting to OU Process (how good the mean-reversion is)
- Trade Bounds Optimisation and Backtesting

Statistical Arbitrage

Cointegrated prices have a mean-reverting spread $e_t = \boldsymbol{\beta}'_{\boldsymbol{Coint}} \boldsymbol{P_t}$, when it goes *significantly* above/below μ_e, it gives a signal.

1. **How to generate P&L?** Trade design and algorithmic considerations.
2. **How to evaluate P&L?** Drawdown control and backtesting.

Signals from Mean-Reverting Spread



Signal generation and positions for assets A and B.

$$e_t \gg \mu_e \quad \text{enter with} \quad [-100\%\, P^A, +\beta_C\%\, P^B]$$

$$e_t \ll \mu_e \quad \text{enter with} \quad [100\%\, P^A, -\beta_C\%\, P^B]$$

To make the trading systematic and controlled, you will need:

- **Loadings** β_{Coint} give positions, the spread is coint residual

$$e_t = P_t^A + \beta_B P_t^B + \cdots + \beta_G P_t^G$$

- **Bounds** $\mu_e \pm Z\,\sigma_{eq}$ give **entry** signal, while **exit** at $e_t \approx \mu_e$

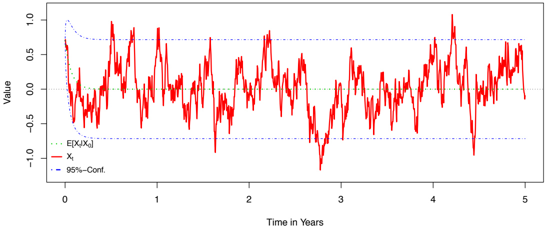
  Instead of assuming $Z = 1$ you can vary in the range $[0.7, 1.3]$ or as fitting to your spread.

- **Half-life** between the crossings $e_t = \mu_e$.

$$\widetilde{\tau} \propto \ln 2/\theta$$

  Average time between exists that fix positive P&L.

OU Process simulated



We consider the process because it generates **mean-reversion**.
Your empirical spread e_t might/might not be as good as this.

$$de_t = -\theta(e_t - \mu_e)\,dt + \sigma_{OU}\,dX_t \tag{1}$$

- $\theta \ll 0$ is the speed of reversion to the equilibrium μ_e
- σ is the scatter of BM diffusion (not of reversion σ_{eq}).

Fitting to OU Process

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

Two terms of SDE solution: reversion and autoregression

$$e_t = C + B e_{t-1} + \epsilon_{t,\tau} \qquad \text{run a regression}$$

$$e^{-\theta\tau} = B \quad \Rightarrow \quad \boxed{\theta = -\frac{\ln B}{\tau}} \tag{2}$$

$$\left(1 - e^{-\theta\tau}\right)\mu_e = C \quad \Rightarrow \quad \boxed{\mu_e = \frac{C}{1 - B}} \tag{3}$$

Signal-generating Bounds

$$\sigma_{eq} = \sqrt{\frac{\text{SSE} \times \tau}{1 - e^{-2\theta\tau}}} \qquad (4)$$

SSE is sum of squared residuals of your regression for e_t, for this AR(1). It represents covariance, SSE $\times\, \tau = \Sigma_\tau$.

σ_{OU} is parameter of the SDE, Brownian Motion diffusion *over each small dt*. Not needed for trading *per se*.

$$\begin{aligned} \sigma_{OU} &= \sigma_{eq}\sqrt{2\theta} \\ &= \sqrt{\frac{2\theta\,\text{SSE}}{1 - e^{-2\theta\tau}}}. \end{aligned}$$

OU Fit – Model Risk

IN PRACTICE we want to trade with tight bounds$Z < 1$ of the higher frequency spread.

$$\mu_e \pm Z\,\sigma_{eq}$$

For the largest profit per trade, typically $Z > 1.5$, the strategy is prone to the breakouts (partitioning of the coint relationship).

Ex ante testing for regime-change is of little help. Adaptive estimation with Kalman or other filtration means unwanted rebalancing, however.

You are constructing the model (cointegration) as much as you are testing for it. There are a number of ways where model not suitable, typically, (a) spread too tight, below bid/ask spread, and (b) OU process might not fit well.

Implementation Notes

From Vector Autoregression to Multivariate Cointegration

Returns are modelled with Vector Autoregression but forecasting is poor.

Prices can be tied up with special error correction equations:

$$\boldsymbol{\Delta P_t} = \boldsymbol{\Pi}\, \boldsymbol{P_{t-1}} + \boldsymbol{\Gamma_1 \Delta P_{t-1}} + \epsilon_t$$

$\Longrightarrow$ $\boldsymbol{\Pi}$ must have reduced rank, otherwise *rhs* will not balance *lhs*.

Now to make this look alike Engle-Granger, we decompose coefficients $\boldsymbol{\Pi} = \alpha\, \beta'_{Coint}$

$$\boldsymbol{\Delta P_t} = \alpha\, (\underbrace{\beta'_C\, \boldsymbol{P_{t-1}} + \mu_e}) + \boldsymbol{\Gamma_1 \Delta P_{t-1}} + \epsilon_t$$

Cointegrating Vector Estimators β'_{Coint}

	1	2	3	4	5	6	7
Canada	6.78395	-1.96320	-9.07554	7.03629	2.56142	6.25519	-2.08045
France	4.86921	4.86043	-2.08623	-7.28739	2.28808	-1.59825	-1.60875
Germany	-15.76001	-5.94947	0.12170	3.34469	-0.01972	-4.04040	4.24522
Japan	-1.22250	5.52024	-0.70856	1.03285	-0.17938	-0.08242	1.76463
UK	27.19903	-13.06796	-0.55980	-0.36245	-1.03954	-1.76308	0.23821
US	-10.25644	13.17254	7.00734	-0.56186	-5.15207	2.16214	-2.37646
Const	-117.01015	-5.47002	59.45116	-32.77753	5.05186	-8.11528	-7.19582

- $n - 1$ columns are linearly dependent on the 1st column.
- $r = 1$ columns of β are cointegrating vectors, take the first column and standardise it.

$$\begin{bmatrix} 1 & 0.7178 & -2.3231 & -0.1802 & 4.0093 & -1.5119 & -17.2481 \end{bmatrix}$$

The allocations $\hat{\beta}'_{Coint}$ provide a mean-reverting spread.

Sequential Testing for Cointegration Rank

Trace Statistic and Maximum Eigenvalue tests rely on eigenvalues of $\mathbf{\Pi}$.

r	lambda	1-lambda	ln(1-lambda)	Trace	CV trace	MaxEig	CV MaxEig
0	0.0167	0.9833	-0.0168	105.7518	103.8473	44.8038	40.9568
1	0.0094	0.9906	-0.0094	60.9479	76.9728	25.1283	34.8059
2	0.0046	0.9954	-0.0046	35.8197	54.0790	12.3440	28.5881
3	0.0038	0.9962	-0.0038	23.4757	35.1928	10.2469	22.2996
4	0.0031	0.9969	-0.0031	13.2287	20.2618	8.3510	15.8921
5	0.0018	0.9982	-0.0018	4.8777	9.1645	4.8777	9.1645

- Trace statistic $H_0 : r = r^*$, and $H_1 : r > r^*$. **Table above $r^* = 1$**

$$LR_{r^*} = -T \sum_{i=r^*+1}^{n} \ln(1 - \lambda_i)$$

- Maximum eigenvalue statistic $H_0 : r = r^*$, and $H_1 : r = r^* + 1$

$$LR_{r^*} = -T \ln(1 - \lambda_{r^*+1})$$

Implementation Notes - R

Cointegration Analysis – Johansen Procedure is a useful **screening tool**.

The workhorse is *ca.jo()* function from the **R package** *urca*.

```
          test 10pct  5pct  1pct
r <= 6 |   4.67  7.52  9.24 12.97
r <= 5 |   5.87 13.75 15.67 20.20
r <= 4 |   9.78 19.77 22.00 26.81
r <= 3 |  24.98 25.56 28.14 33.24
r <= 2 |  44.91 31.66 34.40 39.79
r <= 1 |  46.88 37.45 40.30 46.82
r = 0  | 101.10 43.25 46.45 51.91
```

cajorls() presents the output as a set of familiar OLS equations with EC term, separate line for each price.

Implementation Notes - Python

from statsmodels.tsa.stattools import coint

coint(PriceA, PriceB)

Critical values were fixed to MacKinnon(2010) after been wrong for about 2011-2017!

import statsmodels.tsa.stattools as ts

ts.adfuller() gives a rudimentary output for DF Test for stationarity.

Requirement (TS Topic): implement Engle-Granger procedure from the first principles. Enclosed R code gives a complete example.

Implementation Notes - Python (Cont)

import statsmodels.tsa.vector_ar.vecm as cajo

johansen_test = cajo.coint_johansen(PricesData, 0, 2)

Python routines output will be very similar to VECM output from R routines (package *urca*).

To install dev version, use *git()* instead of *pip*. Refer to the source code comments to understand inputs and outputs.

`https://www.statsmodels.org/dev/generated/statsmodels.tsa.vector_ar.vecm.VECM.html`

Relevant Econometric Advances

1. Estimation of regression adaptively, via a state-space model known as Kalman filter, removes the need for rolling parameters
   `www.thealgoengineer.com/2014/online_linear_regression_kalman_filter/`

   (a) Recursive re-estimation of coint residual $\widehat{e}_t = P_t^A - \widehat{\beta}_C P_t^B - \widehat{\mu_e}$ 'contradicts' the idea of long-term error correction: $\widehat{\beta}_C$ stable.
   (b) But you can apply Kalman filter for the fine-tuning of OU process.

2. `cran.r-project.org/web/views/Robust.html`

3. `cran.r-project.org/web/views/Econometrics.html`

END OF WORKSHOP