19) Mean Reversion on Futures

Vitor Kamada

December 2018

Reference

Tables, Graphics, and Figures from

https://www.quantopian.com/lectures

Lecture 53 Mean Reversion on Futures

Random Walk

$$P_t = \mu + P_{t-1} + \epsilon_t$$

$$r_t = P_t - P_{t-1} = \mu + \epsilon_t$$

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
from statsmodels.tsa.stattools import coint, adfuller

import matplotlib.pyplot as plt
from quantopian.research.experimental import continuous_future, history
```

Soybean Crush

```
soy_meal_mult = symbols('SMF17').multiplier
soy_oil_mult = symbols('BOF17').multiplier
soybean_mult = symbols('SYF17').multiplier
```

```
sm_future = continuous_future('SM', offset=0,
            roll='calendar', adjustment='mul')
sm_price = history(sm_future,
          'price','2014-01-01', '2017-01-01','daily')

bo_future = continuous_future('BO',
          offset=0, roll='calendar', adjustment='mul')
bo_price = history(bo_future,
      'price', '2014-01-01', '2017-01-01', 'daily')

sm_price.plot()
bo_price.multiply(soy_oil_mult//soy_meal_mult).plot()
plt.ylabel('Price')
plt.legend(['Soybean Meal', 'Soybean Oil']);
```

Soybean Meal vs Soybean Oil

Augmented Dickey-Fuller Test

```
print 'p-value: ', coint(sm_price, bo_price)[1]
```

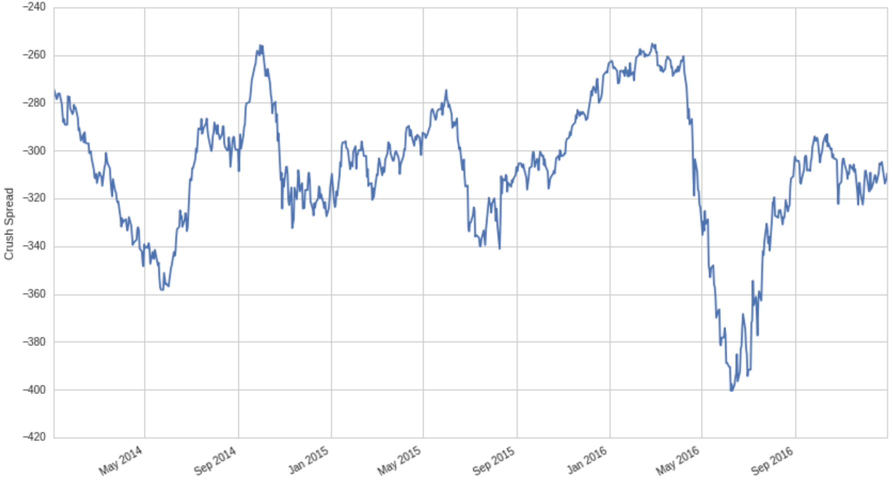
```
p-value:  0.228842012164
```

```
sm_future = continuous_future('SM', offset=1, roll='calendar', adjustment='mul')
sm_price = history(sm_future, 'price', '2014-01-01', '2017-01-01', 'daily')

bo_future = continuous_future('BO', offset=1, roll='calendar', adjustment='mul')
bo_price = history(bo_future, 'price', '2014-01-01', '2017-01-01', 'daily')

sy_future = continuous_future('SY', offset=0, roll='calendar', adjustment='mul')
sy_price = history(sy_future, 'price', '2014-01-01', '2017-01-01', 'daily')

crush = sy_price - (sm_price + bo_price)
crush.plot()
plt.ylabel('Crush Spread');
```

```
print 'p-value for stationarity: ', adfuller(crush)[1]
```

```
p-value for stationarity:  0.0253387237046
```

Crush Spread



Profitability of Oil Refining

3:2:1 Crack Spread: Buy three crude oil, sell two gasoline, Sell one heating oil

---

Fattening Feeder Cattle

8:4:3 Cattle Crush: Buy 8 October live-cattle, Sell 4 May feeder cattle, Sell 3 July corn

Potential relationships between Futures and Stocks

- Crude oil futures and oil stocks
- Gold futures and gold mining stocks
- Crude oil futures and airline stocks
- Currency futures and exporters
- Interest rate futures and utilities
- Interest rate futures and Real Estate Investment Trusts (REITs)
- Corn futures and agricultural processing companies

Futures and Stocks

```
ty_future = continuous_future('TY', offset=0, roll='calendar',
                              adjustment='mul')
ty_prices = history(ty_future, 'price', '2009-01-01', '2017-01-01','daily')
ty_prices.name = ty_future.root_symbol

equities = symbols(['EQR', 'SPY'])
equity_prices = get_pricing(equities, fields='price',
                            start_date='2009-01-01', end_date='2017-01-01')
equity_prices.columns = map(lambda x: x.symbol, equity_prices.columns)

data = pd.concat([ty_prices, equity_prices], axis=1)
data = data.dropna()
```

```
data.plot()
plt.legend();
```

```
print 'Cointegration test p-value: ', coint(data['TY'], data['EQR'])[1]
```

```
Cointegration test p-value:  0.0299261276671
```

EQR (REIT) and Ten-Year Interest Rate Futures

Crude Oil Futures and Oil Company Stocks

```
cl_future = continuous_future('CL', offset=0, roll='calendar', adjustment='mul')
cl_prices = history(cl_future, 'price', '2007-01-01', '2017-04-06','daily')
cl_prices.name = cl_future.root_symbol

equities = symbols(['XOM', 'SPY'])
equity_prices = get_pricing(equities, fields='price', start_date='2007-01-01',
                            end_date='2017-04-06')
equity_prices.columns = map(lambda x: x.symbol, equity_prices.columns)

data = pd.concat([cl_prices, equity_prices],axis=1)
data = data.dropna()
```

```
data['stock_ret'] = np.log(data['XOM']).diff()
data['spy_ret'] = np.log(data['SPY']).diff()
data['futures_ret'] = np.log(data['CL']).diff()

# Compute excess returns in excess of SPY
data['stock_excess'] = data['stock_ret'] - data['spy_ret']

#Compute lagged futures returns
data['futures_lag_diff'] = data['futures_ret'].shift(1)
data = data[2:].dropna()
data.tail(5)
```

Data Table

	CL	XOM	SPY	stock_ret	futures_ret	spy_ret	stock_excess	futures_lag	futures_lag_diff
2017-03-31 00:00:00+00:00	50.85	82.00	235.72	-0.020281	0.010279	-0.002331	-0.017950	0.014610	0.014610
2017-04-03 00:00:00+00:00	50.25	82.08	235.37	0.000975	-0.011870	-0.001486	0.002461	0.010279	0.010279
2017-04-04 00:00:00+00:00	51.13	82.35	235.50	0.003284	0.017361	0.000552	0.002732	-0.011870	-0.011870
2017-04-05 00:00:00+00:00	50.82	82.53	234.77	0.002183	-0.006081	-0.003105	0.005288	0.017361	0.017361
2017-04-06 00:00:00+00:00	51.74	83.02	235.39	0.005920	0.017941	0.002637	0.003282	-0.006081	-0.006081

Contemporaneous and Lagged Correlation

```
contemp_corr = data['stock_excess'].shift(1).corr(data['futures_lag_diff'])
#Compute correlation of excess stock returns with lagged futures returns
lagged_corr = data['stock_excess'].corr(data['futures_lag_diff'])
print 'Contemporaneous correlation: ', contemp_corr
print 'Lagged correlation          : ', lagged_corr
```

```
Contemporaneous correlation:  0.257312975324
Lagged correlation         :  -0.0519203748947
```

OLS Result

```
result = sm.OLS(data['stock_excess'],
                sm.add_constant(data['futures_lag_diff'])).fit()
result.summary2()
```

Model:	OLS	Adj. R-squared:	0.002
Dependent Variable:	stock_excess	AIC:	-16587.6204
Date:	2017-04-25 18:21	BIC:	-16575.9124
No. Observations:	2576	Log-Likelihood:	8295.8
Df Model:	1	F-statistic:	6.958
Df Residuals:	2574	Prob (F-statistic):	0.00840
R-squared:	0.003	Scale:	9.3463e-05

	Coef.	**Std.Err.**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-0.0001	0.0002	-0.6881	0.4914	-0.0005	0.0002
futures_lag_diff	-0.0216	0.0082	-2.6377	0.0084	-0.0377	-0.0055

Futures Lag Diff and Stock Excess

```
data['futures_lag_diff'].plot(alpha=0.50, legend=True)
data['stock_excess'].plot(alpha=0.50, legend=True);
```