17) Cointegrated Pairs Trading

Vitor Kamada

June 2018

Reference

Tables, Graphics, and Figures from

https://www.quantopian.com/lectures

Lecture 44 Introduction to Pairs Trading

Correlation Without Cointegration

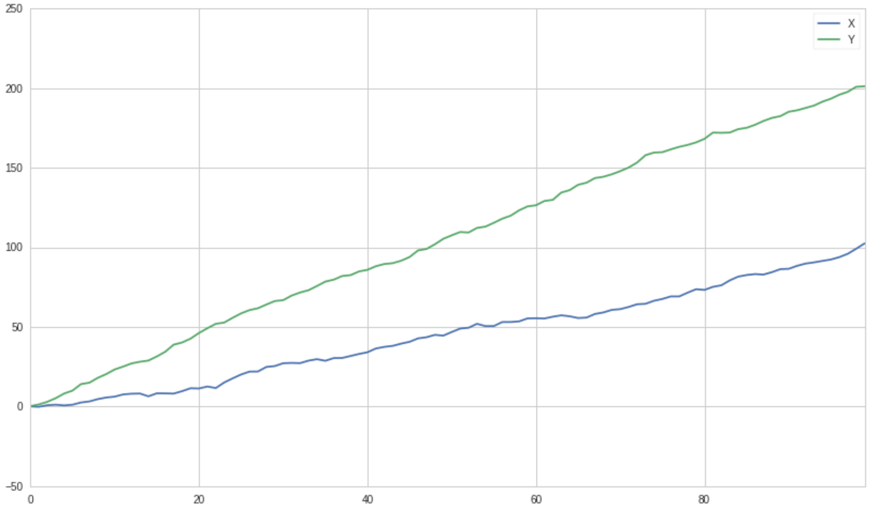
```
X_returns = np.random.normal(1, 1, 100)
Y_returns = np.random.normal(2, 1, 100)
X_diverging = pd.Series(np.cumsum(X_returns), name='X')
Y_diverging = pd.Series(np.cumsum(Y_returns), name='Y')
print 'Correlation: ' + str(X_diverging.corr(Y_diverging))
```

```
0.993134380128
```

```
score, pvalue, _ = coint(X_diverging,Y_diverging)
print 'Cointegration test p-value: ' + str(pvalue)
```

```
0.884633444839
```

pd.concat([X_diverging, Y_diverging], axis=1).plot()



Cointegration Without Correlation

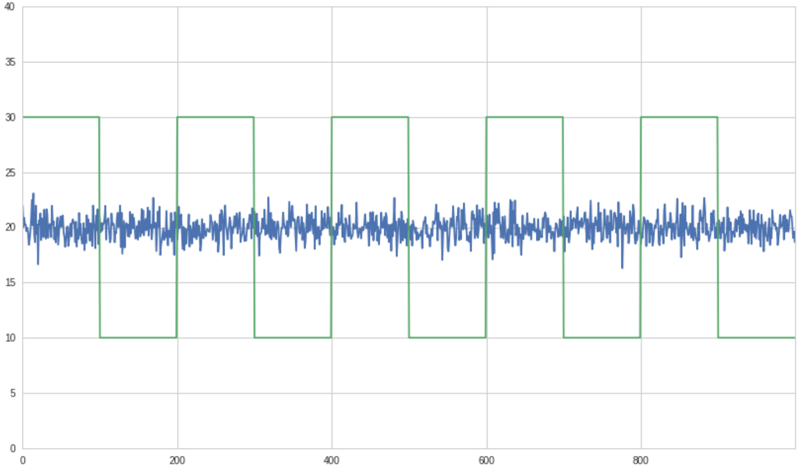
```
Y2 = pd.Series(np.random.normal(0, 1, 1000), name='Y2') + 20
Y3 = Y2.copy()
Y3[0:100] = 30
Y3[100:200] = 10
Y3[200:300] = 30
Y3[300:400] = 10
Y3[400:500] = 30
Y3[500:600] = 10
```

Correlation: -0.0413040695809

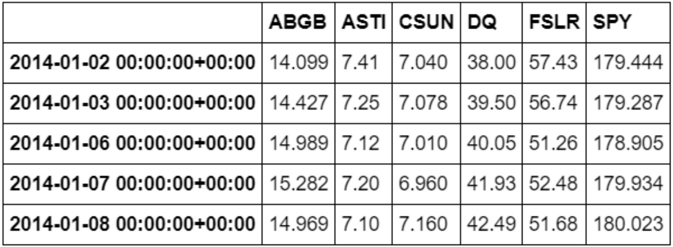
Cointegration test p-value: 0.0

Y2.plot()

```
Y3.plot()
plt.ylim([0, 40]);
```



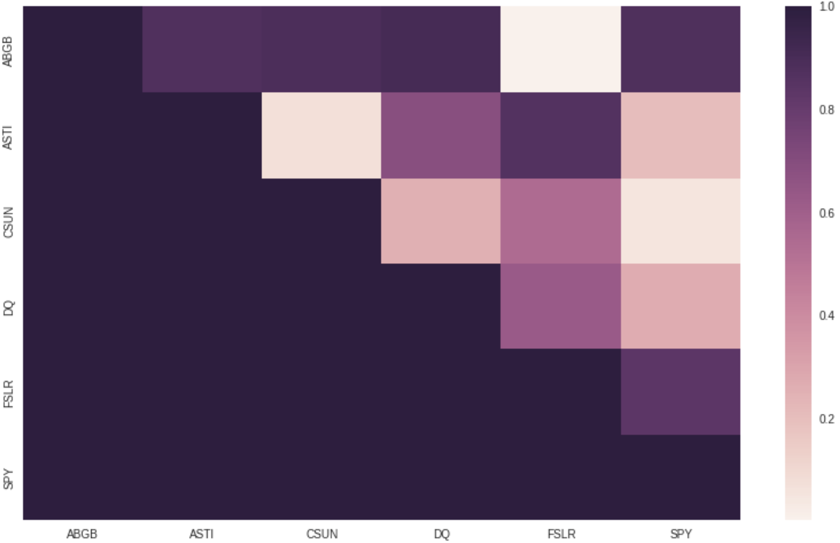
Alternative Energy Securities



	ABGB	ASTI	CSUN	DQ	FSLR	SPY
2014-01-02 00:00:00+00:00	14.099	7.41	7.040	38.00	57.43	179.444
2014-01-03 00:00:00+00:00	14.427	7.25	7.078	39.50	56.74	179.287
2014-01-06 00:00:00+00:00	14.989	7.12	7.010	40.05	51.26	178.905
2014-01-07 00:00:00+00:00	15.282	7.20	6.960	41.93	52.48	179.934
2014-01-08 00:00:00+00:00	14.969	7.10	7.160	42.49	51.68	180.023

```
scores, pvalues, pairs = find_cointegrated_pairs(prices_df)

import seaborn

seaborn.heatmap(pvalues, xticklabels=symbol_list
        , yticklabels=symbol_list )
```

print pairs


ABGB
ASTI
CSUN
DQ
FSLR
SPY
ABGB
ASTI
CSUN
DQ
FSLR
SPY
1.0
0.8
0.6
0.4
0.2

Testing for Cointegration

```
S1 = prices_df[’ABGB’]
S2 = prices_df[’FSLR’]
score, pvalue, _ = coint(S1, S2)
pvalue
```

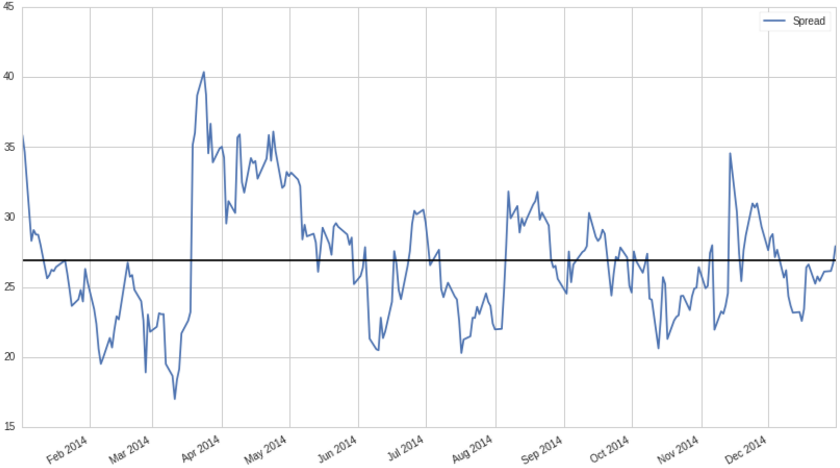
```
0.00495111083
```

Calculating the Spread

```
S1 = sm.add_constant(S1)
results = sm.OLS(S2, S1).fit()
S1 = S1['ABGB']
b = results.params['ABGB']
spread = S2 - b * S1
```

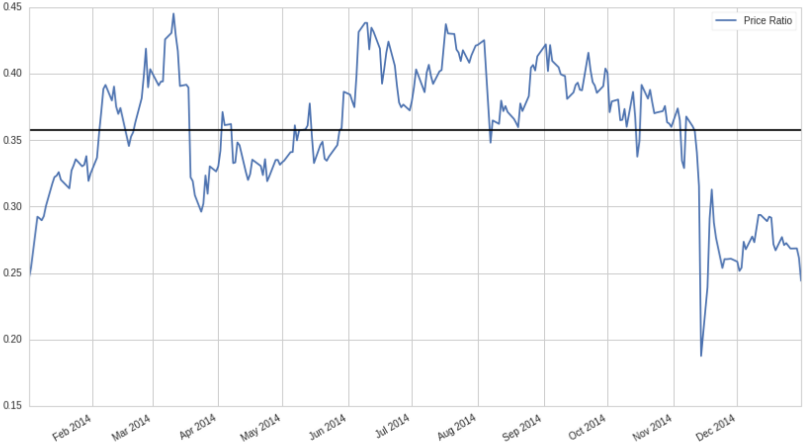
spread.plot()

plt.axhline(spread.mean(), color='black')



ratio = S1/S2

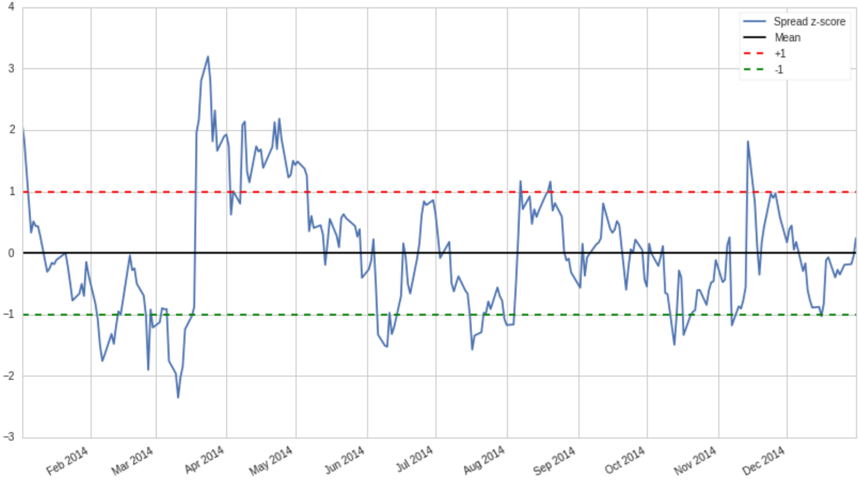
ratio.plot()



Normalize the Spread

```
def zscore(series):

        return (series - series.mean()) / np.std(series)

zscore(spread).plot()

plt.axhline(zscore(spread).mean(), color=’black’)

plt.axhline(1.0, color=’red’, linestyle=’--’)

plt.axhline(-1.0, color=’green’, linestyle=’--’)

plt.legend([’Spread z-score’, ’Mean’, ’+1’, ’-1’]);
```

spread = S2 - b * S1

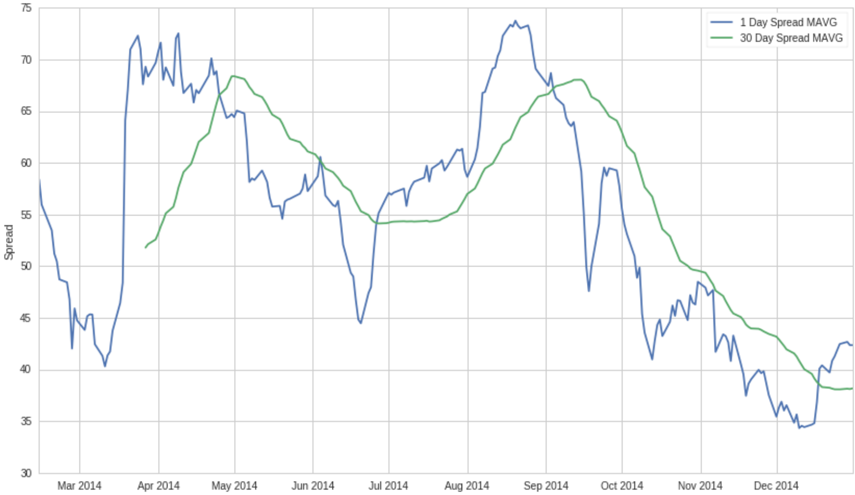


Moving Averages

```
rolling_beta = pd.ols(y=S1, x=S2, window_type=’rolling’
        , window=30)
spread = S2 - rolling_beta.beta[’x’] * S1
spread.name = ’spread’

spread_mavg1 = pd.rolling_mean(spread, window=1)
spread_mavg1.name = ’spread 1d mavg’

spread_mavg30 = pd.rolling_mean(spread, window=30)
spread_mavg30.name = ’spread 30d mavg’
```

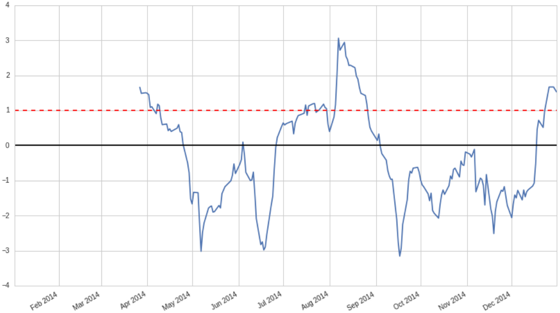
plt.plot(spread_mavg1.index, spread_mavg1.values)

plt.plot(spread_mavg30.index, spread_mavg30.values)



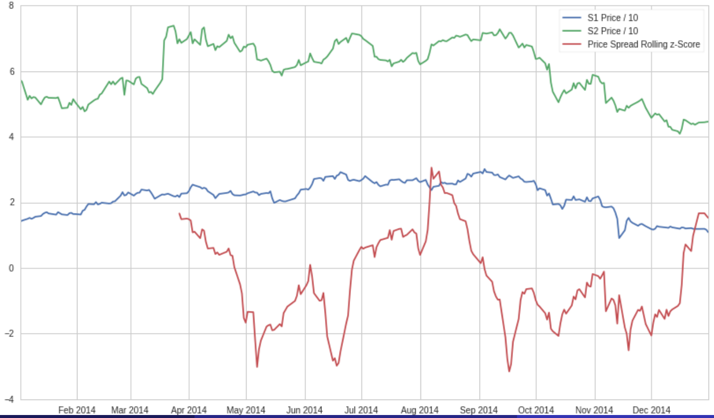
std_30 = pd.rolling_std(spread, window=30)

```
std_30.name = 'std 30d'
zscore_30_1 = (spread_mavg1 - spread_mavg30)/std_30
zscore_30_1.name = 'z-score'
zscore_30_1.plot()
plt.axhline(0, color='black')
plt.axhline(1.0, color='red', linestyle='--');
```



plt.plot(S1.index, S1.values/10)

```
plt.plot(S2.index, S2.values/10)
plt.plot(zscore_30_1.index, zscore_30_1.values)
plt.legend(['S1 Price / 10', 'S2 Price / 10', 'Price Spread Rolling
z-Score']);
```



symbol_list = ['ABGB', 'FSLR']

```
prices_df = get_pricing(symbol_list, fields=['price']
        , start_date='2015-01-01'
        , end_date='2016-01-01')['price']
prices_df.columns = map(lambda x: x.symbol, prices_df.columns)
S1 = prices_df['ABGB']
S2 = prices_df['FSLR']
score, pvalue, _ = coint(S1, S2)
print 'p-value: ', pvalue
```

```
p-value: 0.991161185763
```