In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import datetime as dt
import scipy.stats as scs
import statsmodels.api as sm
from arch.unitroot import PhillipsPerron
```

```
c:\program files\python35\lib\site-packages\statsmodels\compat\pandas.py:56: FutureWarning: The p
andas.core.datetools module is deprecated and will be removed in a future version. Please use the
 pandas.tseries module instead.
  from pandas.core import datetools
```

1 Get data

In [2]:
```
# Use American monthly CPI index from year 1978 to year 2017, seasonally adjusted, available
# https://www.bls.gov/cpi/research-series/allitems.xlsx
rawdata = pd.read_excel('AmeriCPI.xlsx', sheetname=1, header = 6)
rawdata.head()
```

Out[2]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUNE	JULY	AUG	SEP	OCT	NOV	DEC	AVG
0	1977	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	100.3	NaN
1	1978	100.8	101.3	101.8	102.7	103.6	104.2	104.7	105.3	106.1	106.7	107.4	108.2	104.4
2	1979	109.0	109.9	110.8	111.8	112.8	114.0	115.0	115.9	116.7	117.9	118.6	119.8	114.3
3	1980	121.2	122.6	123.8	124.6	125.6	126.4	127.4	128.5	129.8	130.6	131.7	132.5	127.0
4	1981	133.9	135.4	136.5	137.1	137.8	138.5	139.6	140.5	141.7	142.4	143.1	143.6	139.2

In [3]:
```
# The CPI is considered cointegrated with the commodity price indices, as researched by R.A.
rawdata2 = pd.read_excel('DJCI.xls', header = 6, skip_footer=129)
# TR: total return, ER: excess return
rawdata2.head()
```

Out[3]:

	Effective date	Dow Jones Commodity Index TR	Dow Jones Commodity Index ER	Dow Jones Commodity Index
0	2008-05-30	562.18	410.73	682.15
1	2008-06-02	566.74	414.00	687.58
2	2008-06-03	559.74	408.86	679.04
3	2008-06-04	556.23	406.28	674.75
4	2008-06-05	567.26	414.32	688.11

In [4]:
```
rawdata2 = rawdata2.iloc[1:,[0,3]]
rawdata2.index = range(len(rawdata2))
# remove an additional space, and remave the last column
rawdata2.columns = ['Effective date','DJCI']
```

In [5]:
```
resampled = rawdata2.resample('M', on='Effective date').mean()
data_array2 = np.ravel(resampled)
```

In [6]:
```
# The Dow Jones Commodity Index was recorded from 2018 the June, so we match the two
data_array1 = np.insert(rawdata.iloc[32:,1:13].values,0,rawdata.iloc[31,6:13])
```

2 Stationary test

In [7]:
```
# Phillips Perron test
# http://arch.readthedocs.io/en/latest/unitroot/tests.html#phillips-perron-testing
pp1 = PhillipsPerron(data_array1)
pp1
```

Out[7]:

Phillips-Perron Test
(Z-tau)

Test Statistic	0.491
P-value	0.985
Lags	13

In [8]:
```
# 'nc' indicates no trend component in the test
pp1.trend = 'nc'
pp1
```

Out[8]:

Phillips-Perron Test
(Z-tau)

Test Statistic	3.512
P-value	1.000
Lags	13

In [9]:
```
pp2 = PhillipsPerron(data_array2)
pp2
```

Out[9]:

Phillips-Perron Test (Z-
tau)

Test Statistic	-1.969
P-value	0.301
Lags	13

In [10]:
```
# 'ct' indicates a constant and linear time trend in the test
pp2.trend = 'ct'
pp2
```

Out[10]:

Phillips-Perron Test (Z-
tau)

Test Statistic	-1.950
P-value	0.628
Lags	13

The *null hypothesis* of the **Phillips-Perron test** is that there is a unit root, with the *alternative* that there is no unit root. If the p value is above a critical size, then the null cannot be rejected that there and the series appears to be a unit root. So here we can't reject the *null* and thus, it's *unstationary*.

3 Cointegration test and stationary test on the spread

In [11]:
```
regression_data2 = data_array2[5:]
regression_data1 = data_array1[5:]
delta_data_array1 = data_array1[:-1] - data_array1[1:]
delta_tp2 = delta_data_array1[4:]
delta_tp1 = delta_data_array1[3:-1]
delta_tm1 = delta_data_array1[1:-3]
delta_tm2 = delta_data_array1[:-4]
```

In [12]:
```
# Find the cointgration parameter
regression_matrix = sm.add_constant(np.array([regression_data1,delta_tm1,delta_tm2,delta_tp1
model = sm.OLS(regression_data2,regression_matrix)
results = model.fit()
```

In [13]:
```
# coefficients: alpha, beta, detla_1, delta_2, gamma_1, gamma_2
results.params
```

Out[13]:
```
array([978.27893045,  -1.1649586 , -27.56662999, -29.45654111,
       -31.77281647, -19.26055131])
```

In [15]:
```
beta = results.params[1]
```

In [16]:
```
# Phipplip Perron test on the spread
spread = data_array2 - beta*data_array1
pp3 = PhillipsPerron(spread)
pp3
```

Out[16]:

Phillips-Perron Test (Z-tau)

Test Statistic	-1.969
P-value	0.300
Lags	13

In [17]:
```
# 'ct' indicates a constant and linear time trend in the test
pp3.trend = 'ct'
pp3
```

Out[17]:

Phillips-Perron Test (Z-tau)

Test Statistic	-1.970
P-value	0.617
Lags	13

4 Include the spread, the ECM

In [18]:
```
demeaned_spread = spread - np.mean(spread)
demeaned_spread = demeaned_spread[5:]
```

In [19]:
```
ECM_regression_data2 = data_array2[5:] - data_array2[4:-1]
ECM_delta2_tm1 = data_array2[4:-1] - data_array2[3:-2]
ECM_regression_matrix = np.array([demeaned_spread,ECM_delta2_tm1,delta_tm1]).T
ECM_model = sm.OLS(ECM_regression_data2,ECM_regression_matrix)
ECM_results = ECM_model.fit()
```

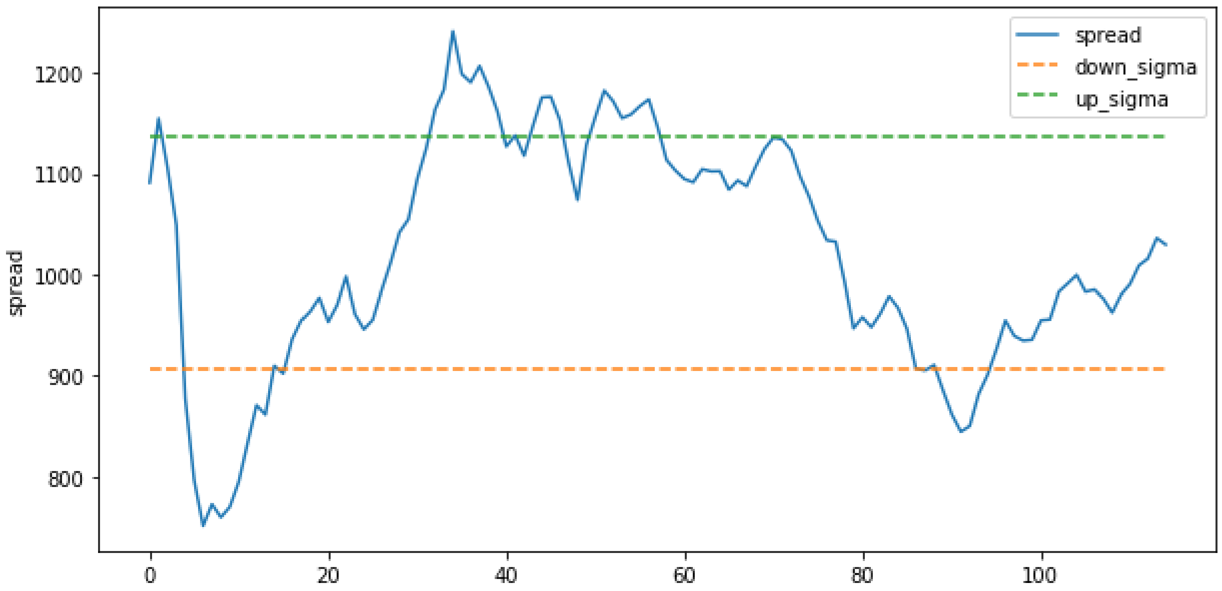
In [20]:
```
# coefficients: theta_0, theta_1, theta_2
ECM_results.params
```

Out[20]:
```
array([-0.00329482,  0.34211161,  0.54969131])
```

5 Plot

In [22]:
```
fig = plt.figure()
fig.set_size_inches(10,5)
ax = fig.add_subplot(111)
A = ax.plot(spread,label='spread')
unit = np.ones_like(spread)
up_sigma = np.mean(spread) + np.std(spread)
up_sigma = up_sigma*unit
down_sigma = np.mean(spread) - np.std(spread)
down_sigma = down_sigma*unit
B = ax.plot(down_sigma,'--',label='down_sigma')
C = ax.plot(up_sigma,'--',label='up_sigma')
ax.legend()
ax.set_ylabel('spread')
```

Out[22]:
```
<matplotlib.text.Text at 0x147d1745c0>
```



In [23]:
```
up_exceed = np.where(spread - up_sigma>0,True,False)
down_exceed = np.where(down_sigma - spread>0,True,False)
exceed = up_exceed|down_exceed
exceed_rate = len(exceed[exceed])/len(exceed)
exceed_rate
```

Out[23]:
```
0.34782608695652173
```