CM Implementation: Production Spread

Group 1

Yating Liu | Yichen Song | Yu Sheng | Weiqi Tong

Introduction:

The commodity production spread is defined as the difference between the price of a raw material commodity and the price of a finished product created from that commodity. To trade commodity production spread, investors usually hold a long position in raw materials and a short position in finished products.

Trading production spread has many benefits including lower risk and attractive margin rates. Commodity production spread trading belongs to hedging strategies which are used for minimising trading risks. For corporation that produce this kind of commodities, trading the contract based on the production spread is a way to hedge their risk on the different stages of the manufacturing cycle. For example, when the raw material price rises, companies which produce finished products will make money on the production spread to cover the loss on the actual product sales and keep the price fixed. However, for the speculative investors like us, we do not produce any commodities involved or carry any actual commodities. The principle of our trading is a speculation on the mispricing between the commodities contracts. We are trading for a pure profit directly from the widening or the narrowing of the production spread.

I. Specification

A. Universe

Several types of spread are frequently seen in the oil and agriculture industries:

- Crack spread is the difference between a barrel of crude oil and petroleum products refined from it
- Crush spread is the margin between soybean futures, soybean oil and meal futures.
- Spark spread uses natural gas as the raw material component and electricity as the finished product. For coal, the difference is called dark spread.

For this implementation, we trade crush spread. Besides, since trading spot involves storage costs and other factors, we trade the continuous contract futures instead for simplicity.

B. Date Range

We work with daily price data for continuous contract futures ranging from 2010 to 2018. We decide to use historical data starting from 2010 because based on our research, the spread that represents the economics of crushing soybeans into soybean meal and oil has been rising since 2013. Thus, we want to fully incorporate this upward trend in our implementation by going back three years and starting with 2010. We further divide our data into two sets: training set (or in-

sample set) which ranges from 2010-01-01 to 2017-01-01, and testing set (or out-of-sample set) which ranges from 2017-01-01 to present.

C. Data Sources

We downloaded data for continuous soybean futures and soybean oil and meal futures traded in CBOT and CME from quandl where we can access all the historical data with zero cost.

D. Signal Generation

The crush spread is quoted as the difference between the combined sales value of soybean meal and oil and the price of soybeans. The Board Crush spread includes Soybean futures, which are traded in cents per bushel, Soybean Meal futures priced in dollars per short ton, and Soybean Oil futures traded in cents per pound. To determine the relationship of the three commodities and potential trading opportunities, it is necessary to convert soybean meal and soybean oil prices to cents per bushel because of their different pricing units.

Convert prices into cents per bushel:

Soybeans: No conversion required. Soybean futures are quoted in cents per bushel
Soybean Meal: 0.022 x price of soybean meal (44 lbs/2000 lbs = 0.022)
Soybean Oil: 11 x price of soybean oil (11 pounds of oil per 60 lb. bushel)
Calculate the crush spread:
[Price of Soybeans ($/bu.) - (Price of Soybean Meal ($/short ton) x .022) - Price of Soybean Oil (¢/lb) x 11]

E. Portfolio Construction

The ratio of soybean, soybean oil and soybean meals continuous contract is 1:1:1.
We short the spread when the spread goes down and crosses over **mean+z*std**. By shorting the spread, we short 1/3 unit of soybean, and long 1/3 unit of soybean oil and soybean meals respectively. We long the spread when the spread goes up and crosses over mean-z*std. More specifically, we long 1/3 unit of soybean, and short 1/3 unit of soybean oil and soybean meals respectively.
Therefore, rebalancing period fully depends on the signals.
For z value, we choose 0.5 throughout our implementation. This value is consistent with what's commonly used in many academic researches. We didn't go through many z values to pick the best one due to the risk of overfitting.

F. Execution

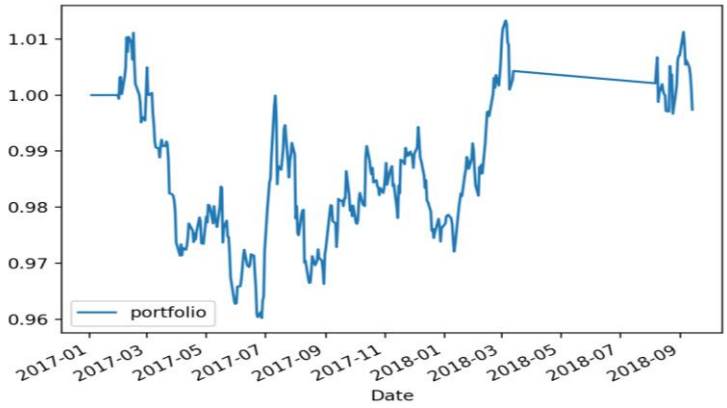
We start with initial investment $10000.
Every time when we find a signal for long position, we divide current total asset (including initial investment and cumulative PnL so far) into three equal parts A1, A2 and A3. We will long A1/Psb of soybean, and short A2/Pso unit of soybean oil and A3/Psm soybean meals respectively. We do the opposite for a signal of short position. Due to the unique property of the

futures, we might find there are some amount of asset left unused since we only need to pay the initial margin for each purchase. We assume this residual amount will be deposited in a margin account and can be used for any possible future marginal call.

II. Implementation

A. PnL Graph

The plot below shows the cumulative return of our portfolio:



There is a significant loss from 2017-02 to 2017-07, and a fluctuation period is following afterwards, which may show that using mean and standard deviation from a long period produce large deviations. From the graph, we can see there is a straight-line downward trend from 2018-03 to 2018-08. There might be a continuous subtle divergence in the spread between 2018-03 and 2018-08, so we may suffer the loss from not closing the position properly.

B. Stats

Annualized return: -0.199% | Sharpe ratio: -3.70 | Maximum drawdown: 4.96% | Volatility: 1.2%

From the data shown above, we find that the annualized return and sharpe ratio are negative, which indicates the current execution is far from satisfactory. Thus refinements are very necessary to further improve our portfolio.

C. Difficulties

All of our group members are not familiar with futures trading; we have no prior experience with futures. From the very beginning, we are quite puzzled with margin, contract unit, contract size and contract with different maturities. To implement more easily, we use continuous contract, which could represent the most active and most traded contract on the market. We further did some research on CME, to fully understand contract unit and contract size.

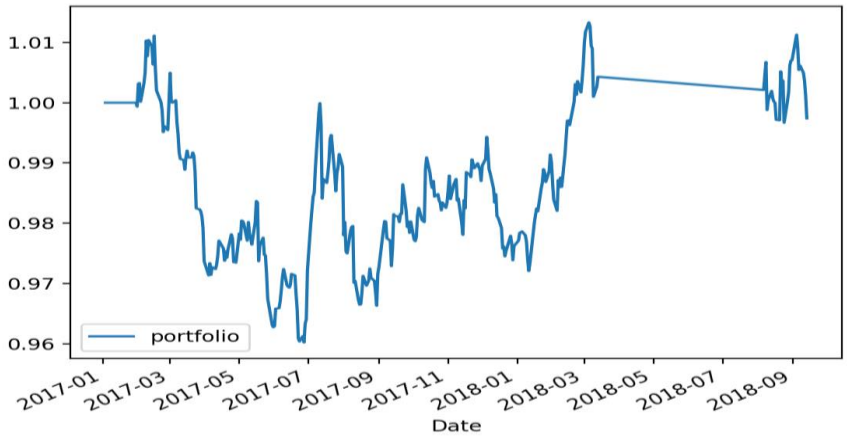
Due to different versions of Python our group members have, we had totally different results from the exact same Python code. Thus, it was challenging for us to exchange our ideas. After detailed analysis of our code, the main reason is due to the difference between Python2 and Python3. For instance, when evaluating performance, we got different results with integer division. We solved the problem after group members using Python2 included a line

'from __future__ import division' at the beginning of their program.
Besides, we have trouble determining transaction cost. We did some research online and found that in one paper regarding trading crush spread, they use 1.5 cents per bushel to estimate transaction cost. They also mentioned transaction cost is mainly ask bid spread, and it is stable. Therefore, for our following refinements, we use transaction cost of 1.5 cents per bushel.

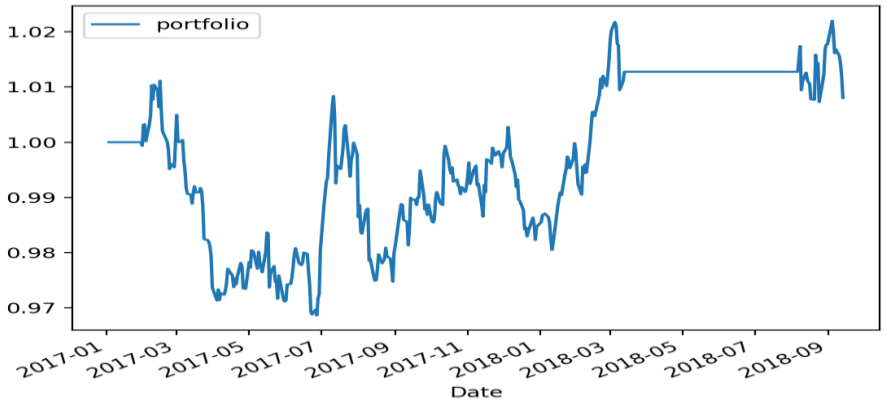
III. Refinements

A. Implemented

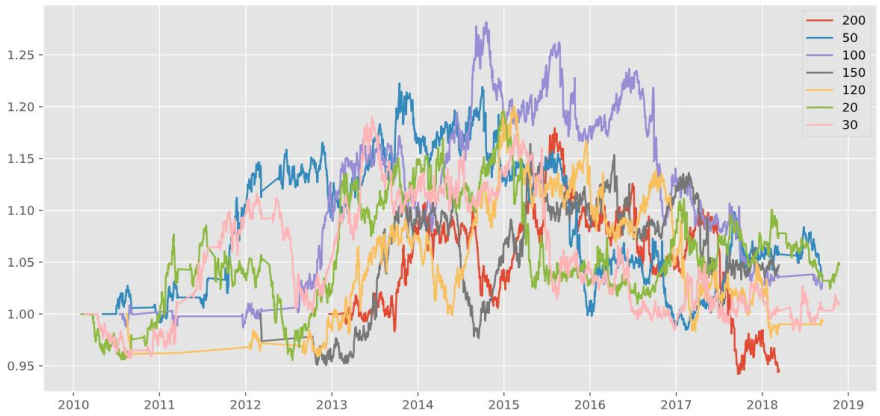
(1) We include **transaction costs**. Transaction cost is mainly composed of bid-ask spread which is very stable in the production spread. We assume it to be 1.5 cents per bushel(including agency cost), which is consistent with many academic research results. Opportunity cost is very low so it can be neglected.



(2) Furthermore, we included stop loss in this refinement. When the spread crosses over the threshold we set, we will clear our positions.



(3) In the previous implementation, we calculated the mean and standard deviation of crush spread based on data from 2010-2017. Here to better refine our strategy, we would like to use a rolling window for calculating mean and standard deviation. The plot below shows the result that we got after comparing the effects of different rolling window. In this way, we could effectively incorporate the most updated information, which is more precise.



(4) We also adopted a so-called two-step open method: when the spread crosses over a lower threshold (depends on z1), we would enter the trade with half of our wealth; when the spread crosses over a higher threshold (depends on z2), we invest the other half.

We didn't tune the hyperparameter z1 and z2. Under certain circumstances, the spread crosses over z1 but doesn't cross over z2, so we could only gain half of the profit as we did before. Thus, this result may seems unsatisfactory compared with former refinement. Since the cumulative return is rather volatile, we decide not to include two-step open method in our comprehensive out-of-sample test.

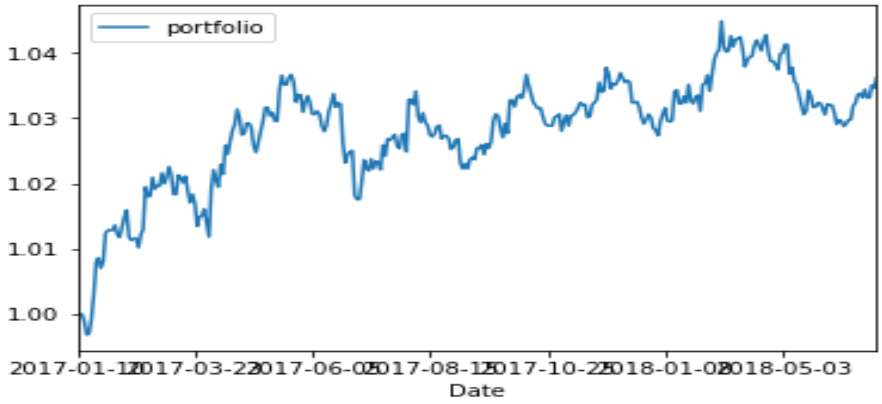
B. Proposals

In order to further diversify our portfolio and hedge the risk, we may consider the options on the soybean crush spread in the CBOT. Options on the soybean crush spread allow investor to establish a crush spread position using a single contract. In addition, buyers of crush spread options do not face the margin requirements necessary in trading an outright futures spread.

IV. Conclusion

A. Out of Sample Test(s)

The plot below shows the cumulative return of our portfolio when we set z=0.5, stop loss=1, transaction cost=0.015, and use 30 days as rolling period for signal calculation.



Stats:
Annual return: 2.66% | Sharpe ratio: -0.4775 | Maximum drawdown: 1.84% | Volatility: 0.86%

Although sharpe ratio is negative, we could employ leverage to improve the performance. Also, the performance of our portfolio is rather stable with low maximum drawdown and volatility.

B. Trading Recommendation

Based on our research and implementation, we would recommend trading the crush spread in the near future. As we can see from the out of sample test above, the trading strategy is profitable after we include transaction cost and stop loss, and choose an appropriate period for rolling mean and standard deviation. Moreover, we expect the profit to continue into the near future for the following reasons: First of all, changes in demand for high protein feed and depletion of South American soybean stocks during the winter months tend to boost up the crush spread. Moreover, the demand for soybeans around the world is a function of increasing population and wealth levels. In the first quarter of 2018, the world's population rose by over 19 million people. More people with more money, are competing for finite raw materials each day, and food is no exception. It is highly possible that soybean price will increase and thus push up the crush spread.

V. Appendix

All code:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# @Date    : 2018-11-27 15:26:02
# @Author  : EricYichen (ericyichen@outlook.com)
# @Link    : ${link}
# @Version : $Id$

import os
import gc
import quandl
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


# https://www.cmegroup.com/trading/agricultural/grain-and-oilseed/soybean-crush-spreads.html
# soybean futures contract:
#       Cents per bushel,
#       5,000 bushels (~136 metric tons)
# soybean meal futures contract:
#       Dollars and Cents per short ton,
#       100 Short Tons (~ 91 metric tons)
# soybean oil futures contract:
#       Cents per pound,
#       60,000 pounds (lbs) (~ 27 metric tons)

def get_data():
    soybean = quandl.get("CHRIS/CME_S1")
    soymeal = quandl.get("CHRIS/CME_SM1")
    soyoil = quandl.get("CHRIS/CME_BO1")

    soybean.to_csv('/Users/songyichen/Desktop/MAFN/Hedge Fund/soybean.csv')
    soymeal.to_csv('/Users/songyichen/Desktop/MAFN/Hedge Fund/soymeal.csv')
    soyoil.to_csv('/Users/songyichen/Desktop/MAFN/Hedge Fund/soyoil.csv')

    return soybean, soymeal, soyoil


def get_signal(train_period=30, ratio={"soybean": 1, "soymeal": 1, "soyoil": 1}):
    # train_period: integer, eg: 100 days, time used to calculate mean and std
    # ratio: the ratio used to calculate spread

    soybean = pd.read_csv('soybean.csv', index_col=0)
    soymeal = pd.read_csv('soymeal.csv', index_col=0)
    soyoil = pd.read_csv('soyoil.csv', index_col=0)

    signal = pd.DataFrame(index=soybean.index)
```

```
    signal['spread'] = soybean['Last'] - 0.022 * soymeal['Last'] - 11 * soyoil['Last']

    signal['MA' + str(train_period)] = signal['spread'].rolling(window=train_period).mean()
    signal['std' + str(train_period)] = signal['spread'].rolling(window=train_period).std()
    signal.dropna(inplace=True)

    signal['prev_spread'] = signal['spread'].shift(1)
    signal['prev_ma'] = signal['MA' + str(train_period)].shift(1)
    signal['prev_std'] = signal['std' + str(train_period)].shift(1)

    signal.to_csv('signal.csv')

    del soybean, soymeal, soyoil
    gc.collect()

    return signal


def trade(signal, z=0.5, stop_loss=float('inf')):
    flag = 0  # indicating if we hold positions, 0: no position, 1: short spread, -1: long spread
    positions = pd.DataFrame(index=signal.index, columns=['soybean', 'soymeal', 'soyoil'])
    positions.iloc[0] = [0, 0, 0]
    cum_position = {'soybean': 0, 'soymeal': 0, 'soyoil': 0}
    pnl = pd.DataFrame(index=signal.index, columns=['portfolio'])
    pnl.iloc[0]['portfolio'] = 10000
    for i, (spread, mean, std, prev_spread, prev_mean, prev_std) in enumerate(signal.values):
        if i == 0:
            continue

        # short spread
        if flag == 0 and spread > mean + z * std and prev_spread < prev_mean + z * prev_std:
            positions.iloc[i]['soybean'] = -1 / 3 * pnl.iloc[i - 1]['portfolio'] /
soybean.iloc[i]['Last']
            positions.iloc[i]['soymeal'] = 1 / 3 * pnl.iloc[i - 1]['portfolio'] /
soymeal.iloc[i]['Last']
            positions.iloc[i]['soyoil'] = 1 / 3 * pnl.iloc[i - 1]['portfolio'] /
soyoil.iloc[i]['Last']

            cum_position['soybean'] -= 1 / 3 * pnl.iloc[i - 1]['portfolio'] /
soybean.iloc[i]['Last']
            cum_position['soymeal'] += 1 / 3 * pnl.iloc[i - 1]['portfolio'] /
soymeal.iloc[i]['Last']
            cum_position['soyoil'] += 1 / 3 * pnl.iloc[i - 1]['portfolio'] / soyoil.iloc[i]['Last']

            flag = 1

            pnl.iloc[i]['portfolio'] = pnl.iloc[i - 1]['portfolio']

        # long spread
        elif flag == 0 and spread < mean - z * std and prev_spread > prev_mean - z * prev_std:
            positions.iloc[i]['soybean'] = 1 / 3 * pnl.iloc[i - 1]['portfolio'] /
soybean.iloc[i]['Last']
```

```
            positions.iloc[i]['soymeal'] = -1 / 3 * pnl.iloc[i - 1]['portfolio'] /
soymeal.iloc[i]['Last']
            positions.iloc[i]['soyoil'] = -1 / 3 * pnl.iloc[i - 1]['portfolio'] /
soyoil.iloc[i]['Last']

            cum_position['soybean'] += 1 / 3 * pnl.iloc[i - 1]['portfolio'] /
soybean.iloc[i]['Last']
            cum_position['soymeal'] -= 1 / 3 * pnl.iloc[i - 1]['portfolio'] /
soymeal.iloc[i]['Last']
            cum_position['soyoil'] -= 1 / 3 * pnl.iloc[i - 1]['portfolio'] / soyoil.iloc[i]['Last']

            flag = -1

            pnl.iloc[i]['portfolio'] = pnl.iloc[i - 1]['portfolio']

        # clear position or stop loss
        elif flag == 1 and ((spread < mean and prev_spread > prev_mean) or (
                spread > mean + stop_loss * std and prev_spread < prev_mean + stop_loss *
prev_std)):
            positions.iloc[i]['soybean'] = -cum_position['soybean']
            positions.iloc[i]['soymeal'] = -cum_position['soymeal']
            positions.iloc[i]['soyoil'] = -cum_position['soyoil']

            cum_position['soybean'] = 0
            cum_position['soymeal'] = 0
            cum_position['soyoil'] = 0

            pnl.iloc[i]['portfolio'] = pnl.iloc[i - 1]['portfolio'] - positions.iloc[i]['soybean']
* (
            soybean.iloc[i]['Last'] - soybean.iloc[i - 1]['Last'])
            + positions.iloc[i]['soymeal'] * (soymeal.iloc[i]['Last'] - soymeal.iloc[i -
1]['Last'])
            + positions.iloc[i]['soyoil'] * (soyoil.iloc[i]['Last'] - soyoil.iloc[i - 1]['Last'])
            - transaction_cost * positions.iloc[i]['soybean'] * contract_size['soybean']

            flag = 0

        # clear position or stop loss
        elif flag == -1 and ((spread > mean and prev_spread > prev_mean) or (
                spread < mean - stop_loss * std and prev_spread > prev_mean - stop_loss *
prev_std)):
            positions.iloc[i]['soybean'] = -cum_position['soybean']
            positions.iloc[i]['soymeal'] = -cum_position['soymeal']
            positions.iloc[i]['soyoil'] = -cum_position['soyoil']

            cum_position['soybean'] = 0
            cum_position['soymeal'] = 0
            cum_position['soyoil'] = 0

            pnl.iloc[i]['portfolio'] = pnl.iloc[i - 1]['portfolio'] + positions.iloc[i]['soybean']
* (
            soybean.iloc[i]['Last'] - soybean.iloc[i - 1]['Last'])
```

```
            - positions.iloc[i]['soymeal'] * (soymeal.iloc[i]['Last'] - soymeal.iloc[i -
1]['Last'])
            - positions.iloc[i]['soyoil'] * (soyoil.iloc[i]['Last'] - soyoil.iloc[i - 1]['Last'])
            - transaction_cost * positions.iloc[i]['soybean'] * contract_size['soybean']
            flag = 0

        else:
            positions.iloc[i] = positions.iloc[i - 1]
            if positions.iloc[i]['soybean'] > 0:
                pnl.iloc[i]['portfolio'] = pnl.iloc[i - 1]['portfolio'] +
positions.iloc[i]['soybean'] * (
                soybean.iloc[i]['Last'] - soybean.iloc[i - 1]['Last'])
                - positions.iloc[i]['soymeal'] * (soymeal.iloc[i]['Last'] - soymeal.iloc[i -
1]['Last'])
                - positions.iloc[i]['soyoil'] * (soyoil.iloc[i]['Last'] - soymeal.iloc[i -
1]['Last'])
                - transaction_cost * positions.iloc[i]['soybean'] * contract_size['soybean']
            elif positions.iloc[i]['soybean'] < 0:
                pnl.iloc[i]['portfolio'] = pnl.iloc[i - 1]['portfolio'] -
positions.iloc[i]['soybean'] * (
                soybean.iloc[i]['Last'] - soybean.iloc[i - 1]['Last'])
                + positions.iloc[i]['soymeal'] * (soymeal.iloc[i]['Last'] - soymeal.iloc[i -
1]['Last'])
                + positions.iloc[i]['soyoil'] * (soyoil.iloc[i]['Last'] - soymeal.iloc[i -
1]['Last'])
                - transaction_cost * positions.iloc[i]['soybean'] * contract_size['soybean']
            else:
                pnl.iloc[i] = pnl.iloc[i - 1]

    ret = pnl / pnl.iloc[0]
    ret.index = pd.to_datetime(ret.index)
    ret.plot()
    plt.show()
    return ret


def two_step_trade(signal, z1=0.3, z2=0.8, stop_loss=1):
    flag = 0  # indicating if we hold positions, 0: no position, 1: short spread, -1: long spread
    positions = pd.DataFrame(index=signal.index, columns=['soybean', 'soymeal', 'soyoil'])
    positions.iloc[0] = [0, 0, 0]
    cum_position = {'soybean': 0, 'soymeal': 0, 'soyoil': 0}
    pnl = pd.DataFrame(index=signal.index, columns=['portfolio'])
    pnl.iloc[0]['portfolio'] = 10000
    for i, (spread, mean, std, prev_spread, prev_mean, prev_std) in enumerate(signal.values):
        if i == 0:
            continue

        # short spread
        if flag == 0 and spread > mean + z1 * std and prev_spread < prev_mean + z1 * prev_std:
            positions.iloc[i]['soybean'] = -1 / 6 * pnl.iloc[i - 1]['portfolio'] /
soybean.iloc[i]['Last']
            positions.iloc[i]['soymeal'] = 1 / 6 * pnl.iloc[i - 1]['portfolio'] /
```

```
soymeal.iloc[i]['Last']
            positions.iloc[i]['soyoil'] = 1 / 6 * pnl.iloc[i - 1]['portfolio'] /
soyoil.iloc[i]['Last']

            cum_position['soybean'] -= 1 / 6 * pnl.iloc[i - 1]['portfolio'] /
soybean.iloc[i]['Last']
            cum_position['soymeal'] += 1 / 6 * pnl.iloc[i - 1]['portfolio'] /
soymeal.iloc[i]['Last']
            cum_position['soyoil'] += 1 / 6 * pnl.iloc[i - 1]['portfolio'] / soyoil.iloc[i]['Last']

            flag = 1 / 2

            pnl.iloc[i]['portfolio'] = pnl.iloc[i - 1]['portfolio']

        elif flag == 1 / 2 and spread > mean + z2 * std and prev_spread < prev_mean + z2 *
prev_std:
            positions.iloc[i]['soybean'] = -1 / 6 * pnl.iloc[i - 1]['portfolio'] /
soybean.iloc[i]['Last']
            positions.iloc[i]['soymeal'] = 1 / 6 * pnl.iloc[i - 1]['portfolio'] /
soymeal.iloc[i]['Last']
            positions.iloc[i]['soyoil'] = 1 / 6 * pnl.iloc[i - 1]['portfolio'] /
soyoil.iloc[i]['Last']

            cum_position['soybean'] -= 1 / 6 * pnl.iloc[i - 1]['portfolio'] /
soybean.iloc[i]['Last']
            cum_position['soymeal'] += 1 / 6 * pnl.iloc[i - 1]['portfolio'] /
soymeal.iloc[i]['Last']
            cum_position['soyoil'] += 1 / 6 * pnl.iloc[i - 1]['portfolio'] / soyoil.iloc[i]['Last']

            flag = 1

            pnl.iloc[i]['portfolio'] = pnl.iloc[i - 1]['portfolio']


        # long spread
        elif flag == 0 and spread < mean - z1 * std and prev_spread > prev_mean - z1 * prev_std:
            positions.iloc[i]['soybean'] = 1 / 6 * pnl.iloc[i - 1]['portfolio'] /
soybean.iloc[i]['Last']
            positions.iloc[i]['soymeal'] = -1 / 6 * pnl.iloc[i - 1]['portfolio'] /
soymeal.iloc[i]['Last']
            positions.iloc[i]['soyoil'] = -1 / 6 * pnl.iloc[i - 1]['portfolio'] /
soyoil.iloc[i]['Last']

            cum_position['soybean'] += 1 / 6 * pnl.iloc[i - 1]['portfolio'] /
soybean.iloc[i]['Last']
            cum_position['soymeal'] -= 1 / 6 * pnl.iloc[i - 1]['portfolio'] /
soymeal.iloc[i]['Last']
            cum_position['soyoil'] -= 1 / 6 * pnl.iloc[i - 1]['portfolio'] / soyoil.iloc[i]['Last']

            flag = -1 / 2

            pnl.iloc[i]['portfolio'] = pnl.iloc[i - 1]['portfolio']
```

```
        elif flag == -1 / 2 and spread < mean - z2 * std and prev_spread > prev_mean - z2 *
prev_std:
            positions.iloc[i]['soybean'] = 1 / 6 * pnl.iloc[i - 1]['portfolio'] /
soybean.iloc[i]['Last']
            positions.iloc[i]['soymeal'] = -1 / 6 * pnl.iloc[i - 1]['portfolio'] /
soymeal.iloc[i]['Last']
            positions.iloc[i]['soyoil'] = -1 / 6 * pnl.iloc[i - 1]['portfolio'] /
soyoil.iloc[i]['Last']

            cum_position['soybean'] += 1 / 6 * pnl.iloc[i - 1]['portfolio'] /
soybean.iloc[i]['Last']
            cum_position['soymeal'] -= 1 / 6 * pnl.iloc[i - 1]['portfolio'] /
soymeal.iloc[i]['Last']
            cum_position['soyoil'] -= 1 / 6 * pnl.iloc[i - 1]['portfolio'] / soyoil.iloc[i]['Last']

            flag = -1

            pnl.iloc[i]['portfolio'] = pnl.iloc[i - 1]['portfolio']

        # clear position or stop loss
        elif flag == 1 and ((spread < mean and prev_spread > prev_mean) or (
                spread > mean + stop_loss * std and prev_spread < prev_mean + stop_loss *
prev_std)):
            positions.iloc[i]['soybean'] = -cum_position['soybean']
            positions.iloc[i]['soymeal'] = -cum_position['soymeal']
            positions.iloc[i]['soyoil'] = -cum_position['soyoil']

            cum_position['soybean'] = 0
            cum_position['soymeal'] = 0
            cum_position['soyoil'] = 0

            pnl.iloc[i]['portfolio'] = pnl.iloc[i - 1]['portfolio'] - positions.iloc[i]['soybean']
* (
            soybean.iloc[i]['Last'] - soybean.iloc[i - 1]['Last'])
            + positions.iloc[i]['soymeal'] * (soymeal.iloc[i]['Last'] - soymeal.iloc[i -
1]['Last'])
            + positions.iloc[i]['soyoil'] * (soyoil.iloc[i]['Last'] - soyoil.iloc[i - 1]['Last'])
            - transaction_cost * positions.iloc[i]['soybean'] * contract_size['soybean']

            flag = 0

        # clear position or stop loss
        elif flag == -1 and ((spread > mean and prev_spread > prev_mean) or (
                spread < mean - stop_loss * std and prev_spread > prev_mean - stop_loss *
prev_std)):
            positions.iloc[i]['soybean'] = -cum_position['soybean']
            positions.iloc[i]['soymeal'] = -cum_position['soymeal']
            positions.iloc[i]['soyoil'] = -cum_position['soyoil']

            cum_position['soybean'] = 0
            cum_position['soymeal'] = 0
```

```
            cum_position['soyoil'] = 0

            pnl.iloc[i]['portfolio'] = pnl.iloc[i - 1]['portfolio'] + positions.iloc[i]['soybean']
* (
            soybean.iloc[i]['Last'] - soybean.iloc[i - 1]['Last'])
            - positions.iloc[i]['soymeal'] * (soymeal.iloc[i]['Last'] - soymeal.iloc[i -
1]['Last'])
            - positions.iloc[i]['soyoil'] * (soyoil.iloc[i]['Last'] - soyoil.iloc[i - 1]['Last'])
            - transaction_cost * positions.iloc[i]['soybean'] * contract_size['soybean']
            flag = 0


        else:
            positions.iloc[i] = positions.iloc[i - 1]
            if positions.iloc[i]['soybean'] > 0:
                pnl.iloc[i]['portfolio'] = pnl.iloc[i - 1]['portfolio'] +
positions.iloc[i]['soybean'] * (
                soybean.iloc[i]['Last'] - soybean.iloc[i - 1]['Last'])
                - positions.iloc[i]['soymeal'] * (soymeal.iloc[i]['Last'] - soymeal.iloc[i -
1]['Last'])
                - positions.iloc[i]['soyoil'] * (soyoil.iloc[i]['Last'] - soymeal.iloc[i -
1]['Last'])
                - transaction_cost * positions.iloc[i]['soybean'] * contract_size['soybean']
            elif positions.iloc[i]['soybean'] < 0:
                pnl.iloc[i]['portfolio'] = pnl.iloc[i - 1]['portfolio'] -
positions.iloc[i]['soybean'] * (
                soybean.iloc[i]['Last'] - soybean.iloc[i - 1]['Last'])
                + positions.iloc[i]['soymeal'] * (soymeal.iloc[i]['Last'] - soymeal.iloc[i -
1]['Last'])
                + positions.iloc[i]['soyoil'] * (soyoil.iloc[i]['Last'] - soymeal.iloc[i -
1]['Last'])
                - transaction_cost * positions.iloc[i]['soybean'] * contract_size['soybean']
            else:
                pnl.iloc[i] = pnl.iloc[i - 1]

    ret = pnl / pnl.iloc[0]
    ret.index = pd.to_datetime(ret.index)
    ret.plot()
    plt.show()
    return ret


# a naive way, no rolling mean, std, split data into train and test data, calculate mean and std
based on train
def train_test_split(df):
    return df['2010-01-01':'2017-01-01'], df['2017-01-01':]


def naive_signal():
    signal = pd.read_csv('~/Desktop/MAFN/Hedge Fund/signal.csv', index_col=0)
    spread = signal[['spread']]
    del signal
```

```
    train_spread, test_spread = train_test_split(spread)
    df = pd.DataFrame(index=test_spread.index)

    mean = np.mean(np.array(train_spread))
    std = np.std(np.array(train_spread))
    sig = (test_spread - mean) / std

    soybean = pd.read_csv('soybean.csv', index_col=0)
    soymeal = pd.read_csv('soymeal.csv', index_col=0)
    soyoil = pd.read_csv('soyoil.csv', index_col=0)

    df['spread'] = test_spread

    df['ma'] = mean
    df['std'] = std
    df['prev_spread'] = df['spread'].shift(1)
    df['prev_ma'] = mean
    df['prev_std'] = std

    d = '~/Desktop/MAFN/Hedge fund/naive.csv'
    df.to_csv(d)

    del soybean, soymeal, soyoil, spread, train_spread, test_spread
    gc.collect()

    return df


if __name__ == '__main__':
    soybean, soymeal, soyoil = get_data()
    signal = get_signal()
    naive_signal = naive_signal()
    # backtest from 2017-01-01
    soybean = soybean['2017-01-01':]
    soymeal = soymeal['2017-01-01':]
    soyoil = soyoil['2017-01-01':]
    signal = signal['2017-01-01':]
    # without transaction cost, without stop loss
    transaction_cost = 0
    trade(naive_signal, z=0.5, stop_loss=float('inf'))
    # without transaction cost, with stop loss
    trade(naive_signal, z=0.5, stop_loss=1)
    # with transaction cost, without stop loss
    transaction_cost = 0.015
    trade(naive_signal, z=0.5, stop_loss=float('inf'))
    # rolling without transaction cost, without stop loss
    trade(signal, z=0.5, stop_loss=float('inf'))
```