Pairs trading in China's futures market using Statistical Arbitrage and Kalman Filter techniques

Project Abstract

Contrary to a more developed market, arbitrage opportunities are not readily realized which suggests there might be opportunities for those looking and able to take advantage of them. My project focuses on China's futures market using Statistical Arbitrage and Pair trading techniques. The project run Augmented Dickey Fuller test on the spread to confirm statistically whether the series is mean reverting or not, calculate Kalman Filter regression on the spread series and a lagged version of the spread series in order to then use the coefficient to calculate the half-life of mean reversion. The results show that though has a relative lower daily Sharpe ratio (2.87 vs. 3.67), the out-sample portfolio has a higher expected daily return, and that the out-sample portfolio has a relative higher CAGR (0.0858 vs. 0.07882) but also has a relative longer average drawdown days.

1. Introduction/Project Motivation

1.1 The project topic: Statistical Arbitrage: Pair trading in China's Futures Markets.

Stocks cannot be shorted according to current China's trading rules. Contrary to a more developed market, arbitrage opportunities are not readily realized which suggests there might be opportunities for those looking and able to take advantage of them. Therefore, I decided to focus on China's futures market using Statistical Arbitrage and Pair trading techniques.

1.2 The strategy idea

The trading strategy implemented in this project is called "Statistical Arbitrage Trading", also known as "Pairs Trading" which is a contrarian strategy designed to profit from the mean-reverting behavior of a certain pair ratio. The assumption behind this strategy is that the spread from pairs that show properties of co-integration is mean reverting in nature and therefore will provide arbitrage opportunities if the spread deviates significantly from the mean.

1.3 The data set

Data set will come from China Financial Futures Exchange (CFFEX)、Shanghai Futures Exchange (SHFE)、Dalian Commodity Exchange (DCE) and Zhengzhou Commodity Exchange (ZCE). All the daily data from the above four Exchanges will be accessed through UQER's API (https://uqer.io/).Due to the availability of data. The trading strategy will be back-tested for 678 days (the period from 3/30/2015 to 31/12/2017). The first 542 day (the period from 3/30/2015 to 14/11/2016 accounts for 80% of total period) is the in sample back-testing period, and the rest 136day (the period from 15/11/2016 to 31/12/2017 accounts for 20 % of total period) is the out sample back-testing period.

China Financial Futures Exchange (CFFEX) is a demutualized exchange dedicated to the trading, clearing and settlement of financial futures, options and other derivatives. On September 8, 2006, with the

approval of the State Council and China Securities Regulatory Commission (CSRC), CFFEX was established in Shanghai by Shanghai Futures Exchange, Zhengzhou Commodity Exchange, Dalian Commodity Exchange, Shanghai Stock Exchange and Shenzhen Stock Exchange.

Shanghai Futures Exchange (SHFE) is organized under relevant rules and regulations. A self-regulated entity, it performs functions that are specified in its bylaws and state laws and regulations. The China Securities Regulatory Commission (CSRC) regulates it. At present, futures contracts' underlying commodities, i.e., gold, silver, copper, aluminum, lead, steel rebar, steel wire rod, natural rubber, fuel oil and zinc, are listed for trading.

Dalian Commodity Exchange (DCE) is a futures exchange approved by the State Council and regulated by China Securities Regulatory Commission (CSRC). Over the years, through orderly operation and stable development, DCE has already become world's largest agricultural futures market as well as the largest futures market for oils, plastics, coal, metallurgical coke, and iron ore. It is also an important futures trading center in China. By the end of 2017, a total of 16 futures contracts and 1 option contract have been listed for trading on DCE, which include No.1 soybean, soybean meal, corn, No.2 Soybean, soybean oil, linear low density polyethylene (LLDPE), RBD palm olein, polyvinyl chloride (PVC), metallurgical coke, coking coal, iron ore, egg, fiberboard, blockboard, polypropylene (PP), corn starch futures and soybean meal option.

Zhengzhou Commodity Exchange (ZCE) is the first pilot futures market approved by the State Council. At present, the listed products on ZCE include : wheat (Strong Gluten Wheat and Common Wheat), Early Long Grain Non-glutinous Rice, Japonica Rice, Cotton, Rapeseed, Rapeseed Oil, Rapeseed Meal, White Sugar, Steam Coal, Methanol, Pure Terephthalic Acid (PTA) and Flat Glass, form a comprehensive range of products covering several crucial areas of the national economy include agriculture, energy, chemical industry and construction materials.

1.4 Motivation of choosing this particular strategy domain

My focuses on China's future market is out of the following main reasons:

To begin with, due to the not-shorting limitation of China's stock markets, we only can long stocks, which makes it is impossible to do pair trading with stocks in China. Because when we do pair trading, we always long few stocks and short ones with high correlation.

What is more, there are very few algo trading firms/strategies that are operating in the China's future exchange. I believe this should provide great opportunities, as there is little competition. Contrary to a more developed market, arbitrage opportunities aren't readily realized which suggests there might be opportunities for those looking and able to take advantage of them.

Last but not the least, UQER (https://uqer.io/) provides excellent APIs, through which I can access all daily main contract data from four future exchange of China. As we all know, high quality data plays a crucial role in algo trading. The accessibility of data is one of important factors We should consider when we are choosing markets and strategies.

A brief outline of what we will do in the following chapters:

1) Define our symbol pair, download the relevant price data from UQER and make sure the data downloaded for each symbol is of the same length.

2) Every possible contract pair will be tested for co-integration. An ADF test will be performed such that, the alternative hypothesis is that the pair to be tested is stationary.

4) Run an Augmented Dickey Fuller test on the spread to confirm statistically whether the series is mean reverting or not. We will also calculate the Hurst exponent of the spread series.

5) Run a Kalman Filter regression on the spread series and a lagged version of the spread series in order to then use the coefficient to calculate the half-life of mean reversion.

6) Calculate Z-scores for trading signal, define enter and out Z-score level for back-testing.

2. Data Mining

2.1 Access the daily main contract data from the four future exchanges.

The daily trading prices of main contract are accessed through UQER's API. The first 542 day (the period from 3/30/2015 to 14/11/2016 accounts for 80% of total period) is the in sample back-testing period, and the rest 136day (the period from 15/11/2016 to 31/12/2017 accounts for 20 % of total period) is the out sample back-testing period.

Using in sample data, we find there are 5 contracts from CFFEX, 14 contracts from SHFE, 16 contracts from DCE, 18 contracts from ZCE. Delete the repeated Contracts in CFFEX, there are 48 contracts remaining.

```
#####  import the necessary libraries ####
import numpy as np
import pandas as pd
import seaborn as sns
from CAL.PyCAL import *
import matplotlib as mpl
mpl.style.use('bmh')
#sns.set_style('white')# bmhggplot
import matplotlib.pylab as plt
from datetime import  datetime
from pandas import DataFrame, Series
import statsmodels.api as sm
from statsmodels.tsa.stattools import adfuller as ADF

##### access the daily trading price of main contract use UQER's API  #####
field = ['tradeDate','exchangeCD','contractObject','settlePrice','closePrice']
#settlePrice——settle price
# using the API of UQER access data
data =
DataAPI.MktMFutdGet(tradeDate=u"",mainCon=u"1",contractMark=u"",contractObject=u"",startDate=u"2
0080101",endDate=u"20171231",field=field,pandas="1")
code = list(set(data['exchangeCD']))  #delete the repeated objects

df = DataFrame()
for i in code:
        df1 = DataFrame(data[data['exchangeCD']==i]['contractObject'])
        df1.columns = [i]
```

```
        df = pd.concat([df,df1],axis=1)

a1 = list(df['CCFX'])
a2 = list(df['XSGE'])
a3 = list(df['XDCE'])
a4 = list(df['XZCE'])
# access the contracts in CFFEX but not in SHFE
CFFEX = DataFrame(list(set(a1).difference(set(a2))),columns=['CCFX'])
# access the contracts in SHFE but not in CFFEX
SHFE = DataFrame(list(set(a2).difference(set(a1))),columns=['XSGE'])
# access the contracts in DCE but not in ZCE
DCE = DataFrame(list(set(a3).difference(set(a4))),columns=['XDCE'])
# access the contracts in ZCE but not in ZCE
ZCE = DataFrame(list(set(a4).difference(set(a3))),columns=['XZCE'])
s = pd.concat([CFFEX,SHFE,DCE,ZCE],axis=0)
s.dropna()

print 'The # of Contracts in CFFEX：',len(CFFEX),'There are：',list(CFFEX['CCFX'])
print 'The # of Contracts in SHFE：',len(SHFE),'There are：',list(SHFE['XSGE'])
print 'The # of Contracts in DCE：',len(DCE),'There are：',list(DCE['XDCE'])
print 'The # of Contracts in ZCE：',len(ZCE),'There are：',list(ZCE['XZCE'])
print 'Delete the repeated Contracts in CFFEX, the remaining：' ,len(SHFE)+len(DCE)+len(ZCE)
```

OUT:

```
The # of Contracts in CFFEX： 5 There are： ['TF', 'IH', 'IC', 'T', 'IF']
The # of Contracts in SHFE： 14 There are： ['NI', 'ZN', 'FU', 'AG', 'RU', 'AL', 'PB', 'BU', 'AU', 'SN', 'RB', 'HC',
'CU', 'WR']
The # of Contracts in DCE： 16 There are： ['A', 'C', 'B', 'CS', 'BB', 'PP', 'I', 'J', 'M', 'L', 'JM', 'FB', 'JD', 'V', 'Y', 'P']
The # of Contracts in ZCE： 18 There are： ['MA', 'OI', 'RS', 'SR', 'CF', 'JR', 'WH', 'AP', 'CY', 'LR', 'PM', 'SM', 'FG',
'RM', 'TC', 'RI', 'SF', 'TA']
Delete the repeated Contracts in CFFEX, the remaining： 48
```

2. 2 delete the contract with turnover volume less than 10000

Using in sample data, we delete the contract with turnover volume less than 10000. There are 36 contracts with turnover volume more than 10000.

```
data =
DataAPI.MktMFutdGet(tradeDate='20171229',mainCon=1,contractMark=u"",contractObject=u"",startDate=u
"",endDate=u"",field=[u"contractObject",u"exchangeCD",u"tradeDate",u"closePrice",u"turnoverVol"],pandas
="1")
data = data[data.turnoverVol > 10000 ][data.exchangeCD != u'CCFX'] # not include Contracts from CCFEX
print 'Main Contracts with Turnover Volumn more than 10000：' ,len(data),'there
are:',list(data['contractObject'])
data
```

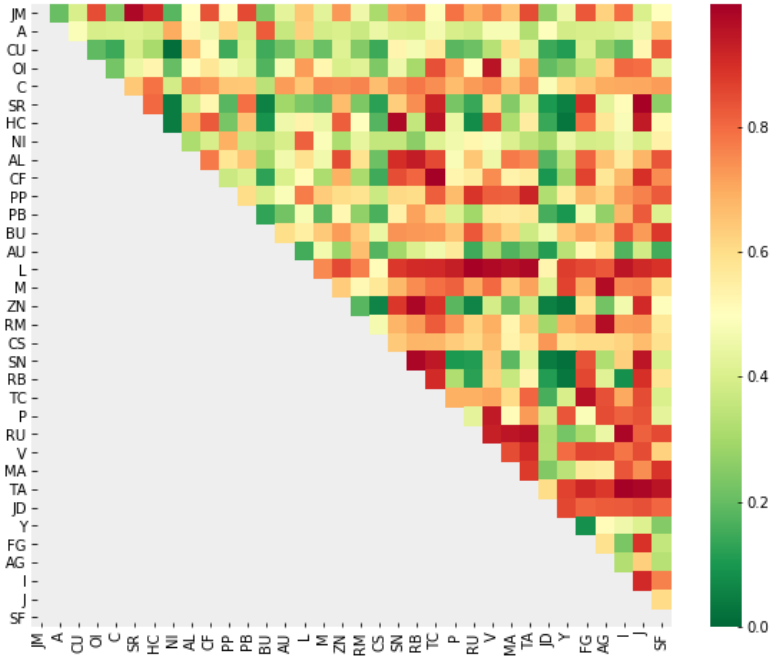
OUT：

Main Contracts with Turnover Volume more than 10000：
36 there are: ['A', 'AG', 'AL', 'AP', 'AU', 'BU', 'C', 'CF', 'CS', 'CU', 'FG', 'HC', 'I', 'J', 'JD', 'JM', 'L', 'M', 'MA', 'NI', 'OI', 'P', 'PB', 'PP', 'RB', 'RM', 'RU', 'SF', 'SM', 'SN', 'SR', 'TA', 'V', 'Y', 'TC', 'ZN']

2.3 find potential trading pairs

Now that stocks have been filtered for their data and daily liquidity, every possible stock pair for each industry will be tested for co-integration.

Plot the heatmap of pvalue_matrix：



Using in sample data, an ADF test will be performed such that, the alternative hypothesis is that the pair to be tested is stationary. The null hypothesis will be rejected for p-values < 0.05. There are 23 pairs with p-values less than 0.05.

```
def find_cointegrated_pairs(dataframe, critial_level = 0.05):
    n = dataframe.shape[1] # the length of dateframe
    pvalue_matrix = np.ones((n, n)) # initialize the matrix of p
    keys = dataframe.keys() # get the column names
    pairs = [] # initilize the list for cointegration
    for i in range(n):
        for j in range(i+1, n): # for j bigger than i
            stock1 = dataframe[keys[i]] # obtain the price of two contract
            stock2 = dataframe[keys[j]]
            result = sm.tsa.stattools.coint(stock1, stock2) # get conintegration
            pvalue = result[1] # get the pvalue
            pvalue_matrix[i, j] = pvalue
            if pvalue < critial_level: # if p-value less than the critical level
                pairs.append((keys[i], keys[j], pvalue)) # record the contract with that p-value
    return pvalue_matrix, pairs
pvalue_matrix, pairs  = find_cointegrated_pairs(data); print(pairs)
```

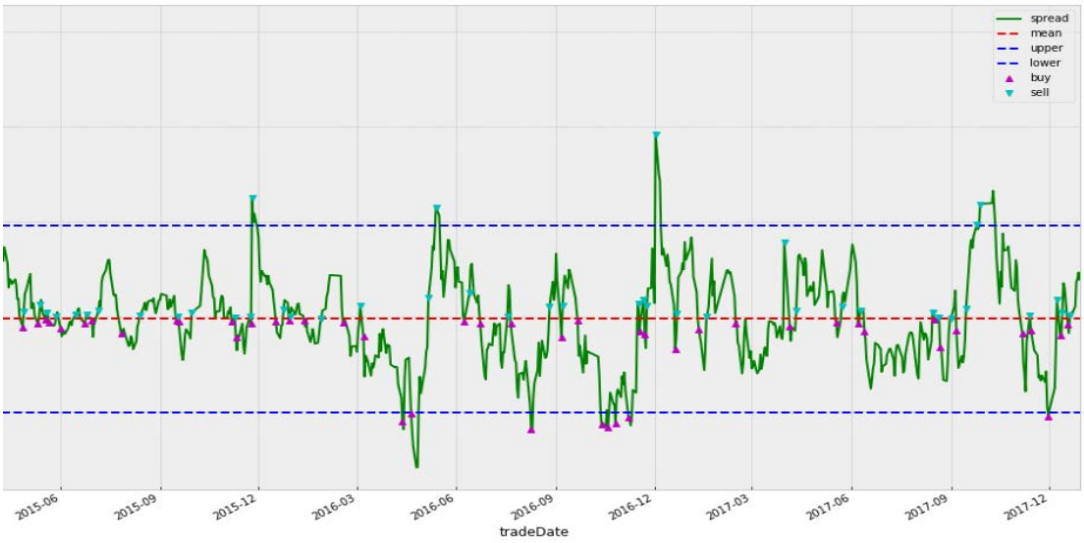
OUT:

	S1	S2	Pvalue
20	TA	I	0.003710
5	CF	TC	0.007014
2	SR	J	0.008478
6	L	RU	0.010882
21	TA	J	0.015553
12	ZN	RB	0.018324
14	SN	RB	0.018869
19	RU	I	0.019091
0	JM	SR	0.020848
9	L	TA	0.021215
3	HC	SN	0.022591
7	L	V	0.026507
11	M	AG	0.027911
13	RM	AG	0.033350
18	RU	TA	0.036407
8	L	MA	0.042057
16	TC	FG	0.042588
1	OI	V	0.043445
22	TA	SF	0.044489
4	HC	TC	0.046238
10	L	I	0.046778
17	RU	MA	0.048415
15	SN	J	0.049904

3 Data Analysis

3.1 Trading logic

(1) Calculate the spread of each pair (Spread = Y – *hedge ratio * X*).
(2) Using Kalman Filter Regression Function to calculate *hedge ratio.*
(3) Calculate z-score of 's', using rolling mean and standard deviation for time period of 'half-life' intervals. Save this as z-score.
(4) Using half life Function to calculate half life.
(5) Define upper entry Z-score = 2.0, lower entry Z-score = 2.0, exit Z-score = 0.0.
(6) When Z-score crosses upper entry Z-score, go SHORT; close the position with Z-score return exit Z-score.
(7) When Z-score crosses lower entry Z-score, go LONG; close the position with Z-score return exit Z-score.
(8) Back-test each pair, and calculate the performance statistics, each as max drown down, Sharpe ratio.
(9) Build up portfolios with equal market value distribution, each pair has the same market value.



3.2 Kalman Filter

From Wikipedia, the free encyclopedia : Kalman filtering, also known as linear quadratic estimation (LQE), is an algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone, by estimating a joint probability distribution over the variables for each time-frame. The filter is named after Rudolf E. Kálmán, one of the primary developers of its theory.

Because the Kalman filter updates its estimates at every time step and tends to weigh recent observations more than older ones, a particularly useful application is estimation of rolling parameters of the data. When using a Kalman filter, there's no window length that we need to specify. This is useful for computing the moving average if that's what we are interested in, or for smoothing out estimates of other quantities. Thanks to Quantopian (https://www.quantopian.com/lectures/kalman-filters), they already provide the source code for calculating the moving average and Regression with Kalman Filter.

```
def KalmanFilterAverage(x):
    # Construct a Kalman filter
    from pykalman import KalmanFilter
    kf = KalmanFilter(transition_matrices = [1],
                      observation_matrices = [1],
                      initial_state_mean = 0,
                      initial_state_covariance = 1,
                      observation_covariance=1,
                      transition_covariance=.01)

    # Use the observed values of the price to get a rolling mean
    state_means, _ = kf.filter(x.values)
    state_means = pd.Series(state_means.flatten(), index=x.index)
    return state_means

#  Kalman filter regression
def KalmanFilterRegression(x,y):
    delta = 1e-3
    trans_cov = delta / (1 - delta) * np.eye(2) # How much random walk wiggles
```

```
obs_mat = np.expand_dims(np.vstack([[x], [np.ones(len(x))]]).T, axis=1)

kf = KalmanFilter(n_dim_obs=1, n_dim_state=2, # y is 1-dimensional, (alpha, beta) is 2-dimensional
        initial_state_mean=[0,0],
        initial_state_covariance=np.ones((2, 2)),
        transition_matrices=np.eye(2),
        observation_matrices=obs_mat,
        observation_covariance=2,
        transition_covariance=trans_cov)

# Use the observations y to get running estimates and errors for the state parameters
state_means, state_covs = kf.filter(y.values)
return state_means
```

3.3 Hurst exponent and Half life

The Hurst exponent is used as a measure of long-term memory of time series. It relates to the auto-correlations of the time series, and the rate at which these decrease as the lag between pairs of values increases. Studies involving the Hurst exponent were originally developed in hydrology for the practical matter of determining optimum dam sizing for the Nile river's volatile rain and drought conditions that had been observed over a long period of time. The name "Hurst exponent", or "Hurst coefficient", derives from Harold Edwin Hurst (1880–1978), who was the lead researcher in these studies; the use of the standard notation H for the coefficient relates to his name also.

To simplify things, the important info to remember here is that a time series can be characterized in the following manner with regard to the Hurst exponent (H):

H < 0.5 - The time series is mean reverting

H = 0.5 - The time series is a Geometric Brownian Motion

H > 0.5 – The time series is trending

However just because a time series displays mean reverting properties, it doesn't necessarily mean that we can trade it profitably – there's a difference between a series that deviates and mean reverts every week and one that takes 10 years to mean revert. I'm not sure too many traders would be willing to sit and wait around for 10 years to close out a trade profitably.

To get an idea of how long each mean reversion is going to take, we can look into the "half-life" of the time series.

```
def half_life(spread):
    spread_lag = spread.shift(1)
    spread_lag.iloc[0] = spread_lag.iloc[1]
    spread_ret = spread - spread_lag
    spread_ret.iloc[0] = spread_ret.iloc[1]
    spread_lag2 = sm.add_constant(spread_lag)
    model = sm.OLS(spread_ret,spread_lag2)
    res = model.fit()
    halflife = int(round(-np.log(2) / res.params[1],0))

    if halflife <= 0:
        halflife = 1
    return halflife
```

3.4 Back-test Engine

The back-test engine follows the steps:

(1) Calculate Spread = Y – hedge ratio * X .

(2) Using Kalman Filter Regression Function to calculate hedge ratio.

(3)Calculate z-score of ‘s’, using rolling mean and standard deviation for time period of ‘half-life’ intervals. Save this as z-score.

(4) Using half life Function to calculate half life.

(5) Define upper entry Z-score = 2.0, lower entry Z-score = 2.0, exit Z-score = 0.0.

(6) When Z-score crosses upper entry Z-score, go SHORT; close the position with Z-score return exit Z-score.

(7) When Z-score crosses lower entry Z-score, go LONG; close the position with Z-score return exit Z-score.

```
def backtest(s1, s2, x, y ):
    ####################################################################
    # INPUT:
    # s1: the symbol of contract one
    # s2: the symbol of contract two
    # x: the price  series of contract one
    # y: the price series of contract two
    # OUTPUT:
    # df1['cum rets']: cumulative returns in pandas data frame
    # sharpe: sharpe ratio
    # CAGR: CAGR

    # run regression to find hedge ratio and then create spread series
    df1 = pd.DataFrame({'y':y,'x':x})
    state_means = KalmanFilterRegression(KalmanFilterAverage(x),KalmanFilterAverage(y))

    df1['hr'] = - state_means[:,0]
    df1['spread'] = df1.y + (df1.x * df1.hr)

    # calculate half life
    halflife = half_life(df1['spread'])

   # calculate z-score with window = half life period

    meanSpread = df1.spread.rolling(window=halflife).mean()
    stdSpread = df1.spread.rolling(window=halflife).std()
    df1['zScore'] = (df1.spread-meanSpread)/stdSpread

    ####################################################################
    # trading logic
    entryZscore = 2
```

```
exitZscore = 0

#set up num units long
df1['long entry'] = ((df1.zScore < - entryZscore) & ( df1.zScore.shift(1) > - entryZscore))
df1['long exit'] = ((df1.zScore > - exitZscore) & (df1.zScore.shift(1) < - exitZscore))
df1['num units long'] = np.nan
df1.loc[df1['long entry'],'num units long'] = 1
df1.loc[df1['long exit'],'num units long'] = 0
df1['num units long'][0] = 0
df1['num units long'] = df1['num units long'].fillna(method='pad')

#set up num units short
df1['short entry'] = ((df1.zScore >  entryZscore) & ( df1.zScore.shift(1) < entryZscore))
df1['short exit'] = ((df1.zScore < exitZscore) & (df1.zScore.shift(1) > exitZscore))
df1.loc[df1['short entry'],'num units short'] = -1
df1.loc[df1['short exit'],'num units short'] = 0
df1['num units short'][0] = 0
df1['num units short'] = df1['num units short'].fillna(method='pad')

df1['numUnits'] = df1['num units long'] + df1['num units short']
df1['spread pct ch'] = (df1['spread'] - df1['spread'].shift(1)) / ((df1['x'] * abs(df1['hr'])) + df1['y'])
df1['port rets'] = df1['spread pct ch'] * df1['numUnits'].shift(1)

df1['cum rets'] = df1['port rets'].cumsum()
df1['cum rets'] = df1['cum rets'] + 1

name = "bt"+ s1 + "-" + s2  + ".csv"
df1.to_csv(name)
#####################################################################

try:
   sharpe = ((df1['port rets'].mean() / df1['port rets'].std()) * sqrt(252))
except ZeroDivisionError:
   sharpe = 0.0

#####################################################################
start_val = 1
end_val = df1['cum rets'].iat[-1]

start_date = df1.iloc[0].name
end_date = df1.iloc[-1].name
days = (end_date - start_date).days

CAGR = round(((float(end_val) / float(start_val)) ** (252.0/days)) - 1,4)

return df1['cum rets'], sharpe, CAGR
```

3.5 In sample back testing results

The in sample back testing period is from 2015/2/27 to 2017/6/15.

(1) In sample back testing of each pair

- **Performance statistics**

There are 14 pairs passed further ADF test, the performance statistics is showed in the following table.

Stat	TC-CF	RU-L	RB-SN	SR-JM	AG-M	AG-RM	TA-RU
Start	2015/3/27	2015/3/27	2015/3/27	2015/3/27	2015/3/27	2015/3/27	2015/3/27
End	2017/6/15	2017/6/15	2017/6/15	2017/6/15	2017/6/15	2017/6/15	2017/6/15
Risk-free rate	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Total Return	13.65%	30.53%	13.27%	29.03%	12.08%	15.83%	13.77%
Daily Sharpe	0.9	1.62	1.13	1.51	1.22	1.31	1.04
Daily Sortino	0.74	1.79	0.84	0.75	1.05	1.31	0.95
CAGR	5.93%	12.75%	5.77%	12.16%	5.27%	6.84%	5.98%
Max Drawdown	-6.47%	-5.72%	-5.13%	-9.12%	-2.70%	-4.97%	-4.04%
Calmar Ratio	0.92	2.23	1.13	1.33	1.96	1.38	1.48
MTD	-1.76%	0.00%	0.00%	1.69%	0.79%	1.03%	-0.87%
3m	3.47%	2.86%	3.91%	5.92%	0.34%	0.92%	-1.54%
6m	4.17%	5.71%	4.09%	4.90%	0.69%	1.41%	-1.54%
YTD	4.17%	3.26%	4.09%	4.90%	0.69%	1.41%	-1.54%
1Y	11.07%	9.99%	3.99%	12.06%	4.48%	7.33%	3.33%
3Y (ann.)	5.93%	12.75%	5.77%	12.16%	5.27%	6.84%	5.98%
5Y (ann.)	-	-	-	-	-	-	-
10Y (ann.)	-	-	-	-	-	-	-
Since Incep. (ann.)	5.93%	12.75%	5.77%	12.16%	5.27%	6.84%	5.98%
Daily Sharpe	0.9	1.62	1.13	1.51	1.22	1.31	1.04
Daily Sortino	0.74	1.79	0.84	0.75	1.05	1.31	0.95
Daily Mean (ann.)	6.20%	12.72%	5.94%	12.20%	5.41%	6.99%	6.19%
Daily Vol (ann.)	6.89%	7.85%	5.27%	8.07%	4.44%	5.33%	5.95%
Daily Skew	-0.02	6	-0.41	-3.15	0.71	1.15	0.7
Daily Kurt	9.89	69.38	14.99	46.47	12.05	9.63	7.72
Best Day	2.31%	6.40%	1.70%	2.96%	1.74%	2.24%	1.94%
Worst Day	-2.45%	-1.71%	-2.71%	-5.21%	-1.67%	-1.27%	-1.52%
Monthly Sharpe	0.82	1.97	1.32	1.57	2.03	2.24	1.18
Monthly Sortino	1.27	3.4	2.28	1.1	4.31	5.85	1.75
Monthly Mean (ann.)	5.95%	12.08%	5.64%	11.65%	5.11%	6.59%	5.86%
Monthly Vol (ann.)	7.25%	6.12%	4.26%	7.40%	2.52%	2.95%	4.96%
Monthly Skew	0.12	0.65	0.19	-2.08	0.29	0.05	0.24
Monthly Kurt	0.96	1.08	0.12	6.51	-0.68	-0.58	1.47
Best Month	5.14%	5.21%	3.13%	3.63%	1.75%	2.17%	4.06%
Worst Month	-4.69%	-2.78%	-1.97%	-6.95%	-0.95%	-0.91%	-2.83%
Yearly Sharpe	1.46	1.19	4.54	1.38	0.87	0.98	0.5
Yearly Sortino	-	-	-	-	-	-	-
Yearly Mean	8.08%	7.99%	3.54%	10.09%	3.65%	5.10%	3.64%
Yearly Vol	5.53%	6.69%	0.78%	7.33%	4.19%	5.22%	7.33%
Yearly Skew	-	-	-	-	-	-	-
Yearly Kurt	-	-	-	-	-	-	-
Best Year	11.99%	12.72%	4.09%	15.27%	6.61%	8.80%	8.82%
Worst Year	4.17%	3.26%	2.99%	4.90%	0.69%	1.41%	-1.54%
Avg. Drawdown	-1.33%	-1.18%	-1.11%	-1.28%	-0.88%	-1.20%	-1.31%
Avg. Drawdown Days	37.77	28.23	23	22.61	24.56	20.89	13.1

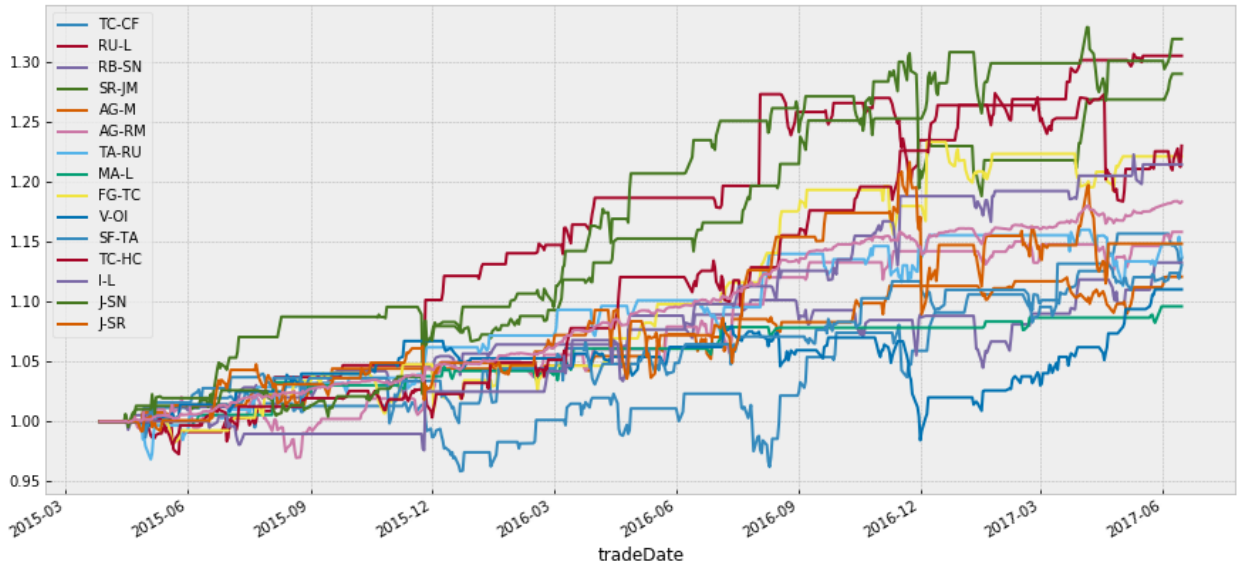
Stat	TC-CF	RU-L	RB-SN	SR-JM	AG-M	AG-RM	TA-RU
Avg. Up Month	2.27%	1.81%	1.12%	2.04%	0.93%	1.08%	1.21%
Avg. Down Month	-0.92%	-0.60%	-0.64%	-1.17%	-0.20%	-0.35%	-0.74%
Win Year %	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	50.00%
Win 12m %	82.35%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%

Stat	MA-L	FG-TC	V-OI	SF-TA	TC-HC	I-L	J-SN	J-SR
Start	2015/3/27	2015/3/27	2015/3/27	2015/3/27	2015/3/27	2015/3/27	2015/3/27	2015/3/27
End	2017/6/15	2017/6/15	2017/6/15	2017/6/15	2017/6/15	2017/6/15	2017/6/15	2017/6/15
Risk-free rate	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Total Return	9.61%	22.13%	11.03%	13.05%	23.01%	21.46%	31.93%	14.84%
Daily Sharpe	1.37	1.35	0.95	0.83	1.41	1.3	1.72	0.69
Daily Sortino	1.34	1.11	0.85	0.71	0.92	1.05	2.05	0.49
CAGR	4.22%	9.42%	4.82%	5.68%	9.78%	9.15%	13.29%	6.43%
Max Drawdown	-1.09%	-3.71%	-9.03%	-3.99%	-7.04%	-3.21%	-4.61%	-10.44%
Calmar Ratio	3.88	2.54	0.53	1.42	1.39	2.85	2.89	0.62
MTD	0.00%	0.00%	0.00%	1.00%	0.37%	0.00%	1.40%	0.00%
3m	0.87%	-0.17%	5.35%	-0.11%	-1.84%	1.86%	1.56%	0.10%
6m	1.65%	-0.98%	8.85%	3.38%	-2.68%	2.23%	2.87%	0.10%
YTD	1.65%	0.30%	8.85%	1.86%	-2.68%	2.23%	0.84%	0.10%
1Y	3.33%	11.23%	4.53%	6.20%	10.12%	10.62%	6.84%	7.13%
3Y (ann.)	4.22%	9.42%	4.82%	5.68%	9.78%	9.15%	13.29%	6.43%
5Y (ann.)	-	-	-	-	-	-	-	-
10Y (ann.)	-	-	-	-	-	-	-	-
Since Incep. (ann.)	4.22%	9.42%	4.82%	5.68%	9.78%	9.15%	13.29%	6.43%
Daily Sharpe	1.37	1.35	0.95	0.83	1.41	1.3	1.72	0.69
Daily Sortino	1.34	1.11	0.85	0.71	0.92	1.05	2.05	0.49
Daily Mean (ann.)	4.32%	9.56%	5.01%	5.97%	9.90%	9.31%	13.20%	6.97%
Daily Vol (ann.)	3.17%	7.08%	5.29%	7.24%	7.05%	7.13%	7.69%	10.16%
Daily Skew	5.77	3.24	-0.28	0.74	-3.37	4.09	3.73	-2.34
Daily Kurt	59.61	59.69	8.58	13.45	49.94	44.86	41.21	31.76
Best Day	2.47%	5.65%	1.61%	3.18%	1.64%	5.02%	5.86%	2.77%
Worst Day	-0.75%	-3.71%	-2.08%	-2.81%	-5.59%	-2.76%	-1.59%	-6.93%
Monthly Sharpe	1.73	1.52	0.86	0.95	1.36	2.32	2.1	1.27
Monthly Sortino	13.28	3.87	0.85	1.67	1.06	-	7.3	3.16
Monthly Mean (ann.)	4.11%	9.09%	4.81%	5.63%	9.48%	8.74%	12.55%	6.28%
Monthly Vol (ann.)	2.37%	5.99%	5.59%	5.95%	6.94%	3.77%	5.98%	4.93%
Monthly Skew	2.25	0.56	-1.28	-0.06	-1.85	0.78	1.27	0.55
Monthly Kurt	4.91	0.64	3.75	0.31	6.95	0.71	1.25	0.29
Best Month	2.65%	5.32%	3.43%	3.83%	3.95%	3.56%	5.75%	4.23%
Worst Month	-0.21%	-2.25%	-4.97%	-3.65%	-6.74%	-1.48%	-1.45%	-1.93%
Yearly Sharpe	2.01	0.73	0.34	1.06	0.56	0.94	0.77	0.72
Yearly Sortino	-	-	-	-	-	-	-	-
Yearly Mean	2.55%	9.01%	2.88%	5.60%	9.89%	9.08%	10.12%	4.75%
Yearly Vol	1.27%	12.32%	8.45%	5.29%	17.78%	9.69%	13.13%	6.58%
Yearly Skew	-	-	-	-	-	-	-	-
Yearly Kurt	-	-	-	-	-	-	-	-
Best Year	3.45%	17.72%	8.85%	9.34%	22.46%	15.93%	19.40%	9.40%
Worst Year	1.65%	0.30%	-3.10%	1.86%	-2.68%	2.23%	0.84%	0.10%
Avg. Drawdown	-0.62%	-1.44%	-1.46%	-1.50%	-1.42%	-1.20%	-1.49%	-1.70%
Avg. Drawdown Days	39	30.73	55.1	32.4	17.21	16.76	21.9	28.1

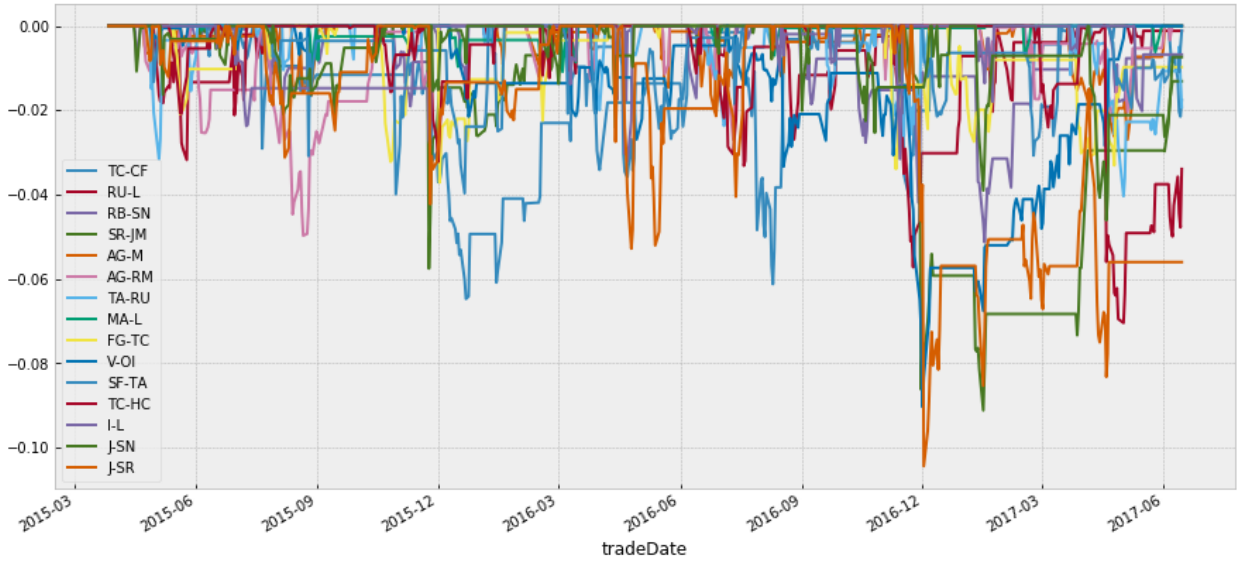
Stat	MA-L	FG-TC	V-OI	SF-TA	TC-HC	I-L	J-SN	J-SR
Avg. Up Month	0.97%	1.87%	1.43%	1.51%	1.69%	1.51%	1.87%	1.50%
Avg. Down Month	-0.03%	-0.64%	-0.89%	-1.05%	-1.01%	-0.11%	-0.36%	-0.69%
Win Year %	100.00%	100.00%	50.00%	100.00%	50.00%	100.00%	100.00%	100.00%
Win 12m %	100.00%	100.00%	70.59%	100.00%	100.00%	100.00%	100.00%	100.00%

As one can see, results vary considerably between pairs. Maximum drawdown ranges from a low of 1.09% to a high of 10.45%. CAGR ranges from 4.22% to 12.75%. Total return ranges from 9.61% to 31.93%.

- **Accumulated returns for each trading pair**



- **The drawn down plot of each pair**



(2) In sample back testing of portfolio

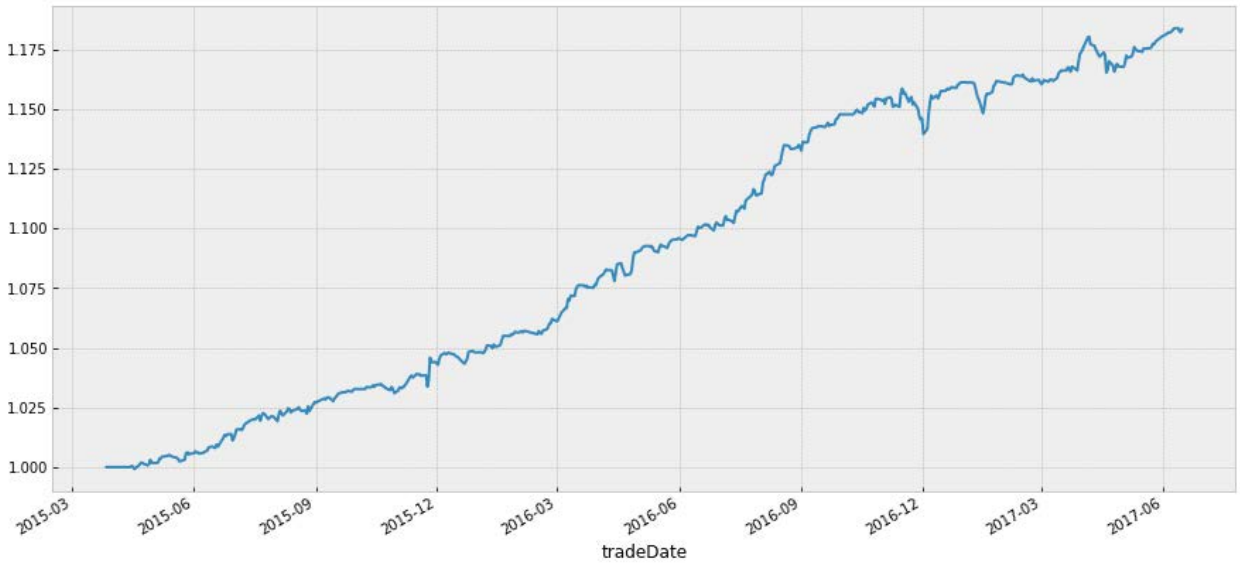
Portfolio: the fund is equally distributed among the above 14 contracts. The market value for each contract is 1/14of total amount of cash.

- **Performance statistics**

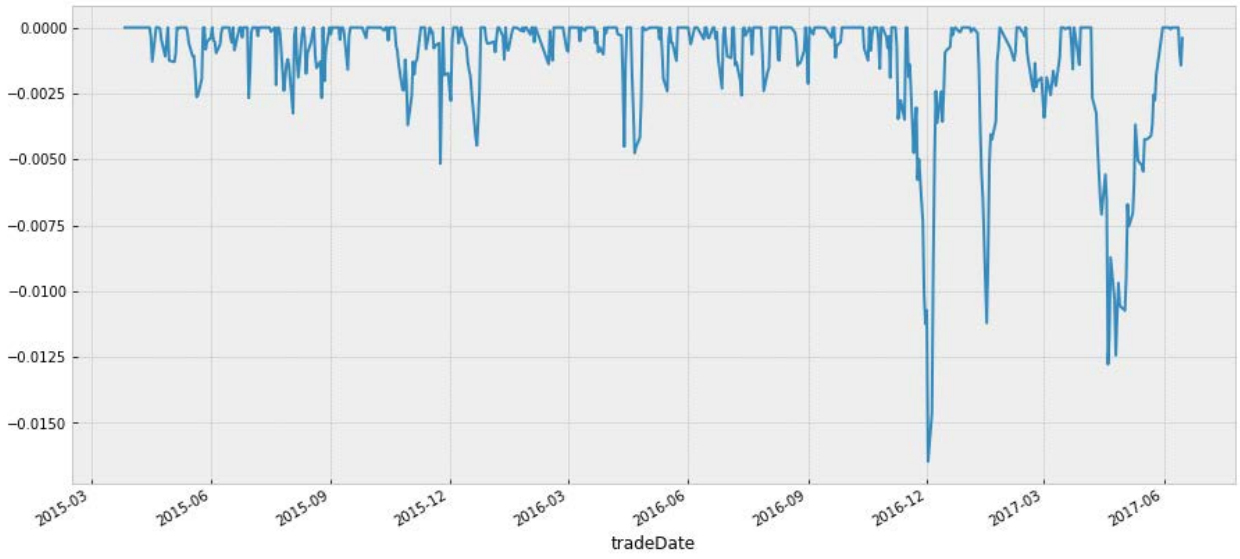
start	**2015/3/27 0:00**
end	2017/6/15 0:00
rf	0
total_return	**0.183477**
daily_sharpe	**3.67899**
daily_sortino	**5.47123**
cagr	0.07882
max_drawdown	-0.016463
calmar	4.78771
mtd	0.00258525
three_month	0.0153632
six_month	0.0224795
ytd	0.0192067
one_year	0.0751948
daily_mean	0.0787084
daily_vol	0.021394
daily_skew	0.396993
daily_kurt	5.58188
best_day	0.00870807
worst_day	-0.00623146
monthly_sharpe	3.56163
monthly_sortino	6.97083
monthly_mean	0.0753179
monthly_vol	0.021147
monthly_skew	-0.226228
monthly_kurt	-0.0897517
best_month	0.0182876
worst_month	-0.00727017
yearly_sharpe	1.01327
yearly_mean	0.0635658
yearly_vol	0.0627332
best_year	0.107925
avg_drawdown	-0.00176623
avg_drawdown_days	5.54217
avg_up_month	0.00799471
avg_down_month	-0.00360324
win_year_perc	1

As we can see from the above table, the total return of portfolio is 18%, the daily Sharpe ratio is 3.67. The maximum drown down is 1.6%, the average drawn down days is 5.5.

- **Accumulated returns for portfolio**



- **The drawn down plot of portfolio**

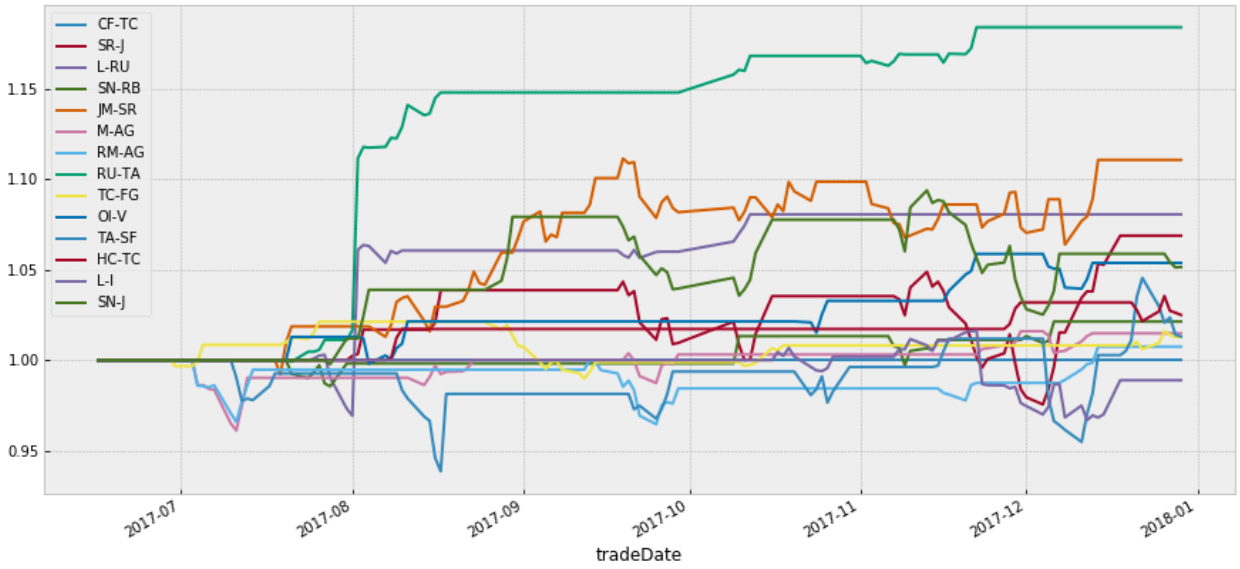


3.5 Out Sample back testing results

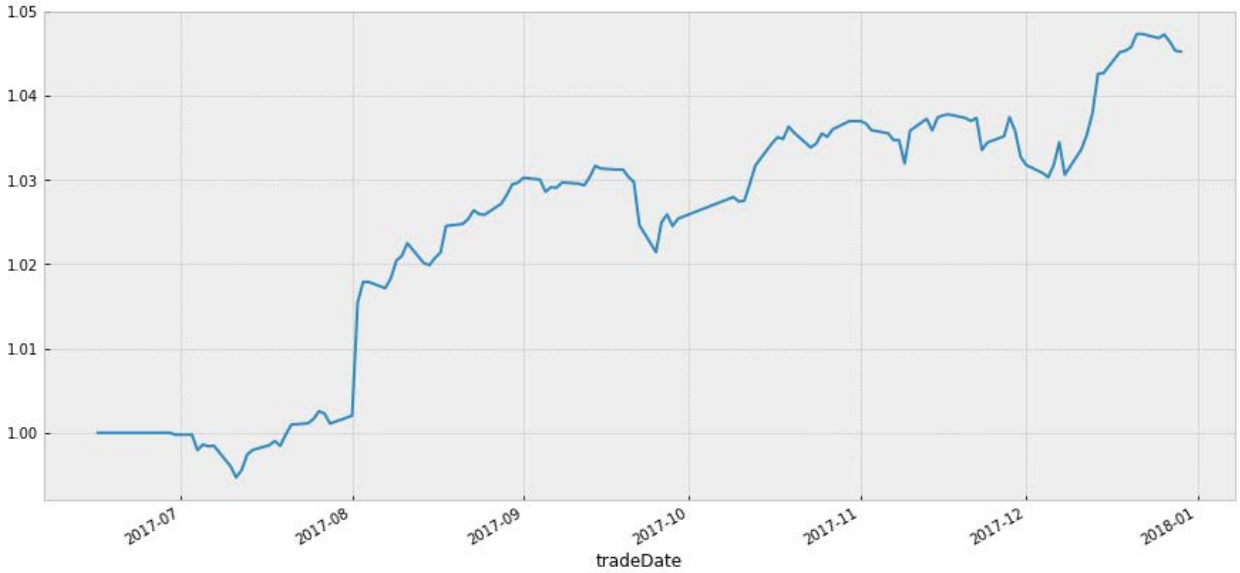
The out sample back testing period is from 2017/6/16 to 2017/12/31.

(1) Out sample back testing of each pair

- **Accumulated returns for each trading pair**



- **Accumulated returns for portfolio**



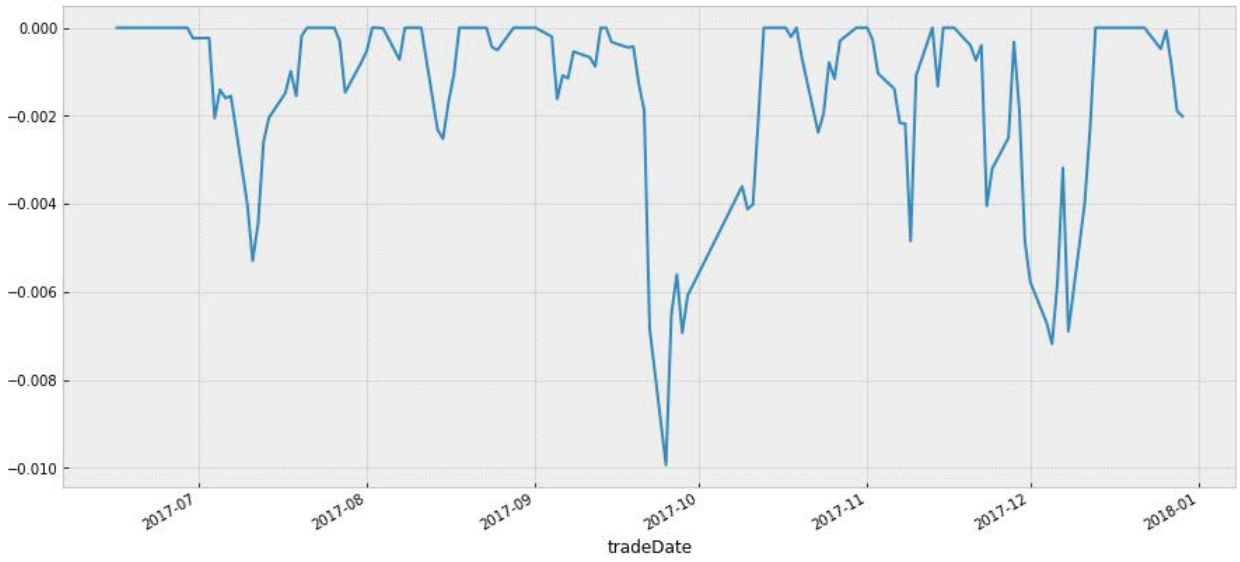
- **Performance statistics of portfolio**

start	**2017/6/16 0:00**
end	2017/12/29 0:00

rf	0
total_return	0.0451815
daily_sharpe	2.87283
daily_sortino	4.7023
cagr	0.0858358
max_drawdown	-0.00992652
calmar	8.64711
mtd	0.0120648
three_month	0.0193065
daily_mean	0.0829146
daily_vol	0.0288616
daily_skew	2.55029
daily_kurt	19.6438
best_day	0.013389
worst_day	-0.00496555
monthly_sharpe	2.12459
monthly_sortino	703.731
monthly_mean	0.0899254
monthly_vol	0.042326
monthly_skew	0.863054
monthly_kurt	0.277174
best_month	0.0278041
worst_month	-0.00412476
avg_drawdown	-0.00307792
avg_drawdown_days	9.69231
avg_up_month	0.01329
avg_down_month	-0.00409867

As we can see from the above table, the total return of portfolio is 4.5%, the daily Sharpe ratio is 2.87. The maximum drown down is 0.9%, the average drawn down days is 9.69.

- **The drawn down plot of portfolio**



4 Key Findings

(1) Although the out-sample portfolio has a relative lower daily Sharpe ratio (2.87 vs. 3.67), the out-sample portfolio has a higher expected daily return (0.0829 vs. 0.0787) .

(2) The out-sample portfolio has a relative longer average drawdown days (9.69 vs. 5.54)

(3) The out-sample portfolio has a relative higher CAGR (0.0858 vs. 0.07882)

Challenges/Limitations

(1) Further research can test the in sample performance with different entry and exit z-score pairs, through numbers of simulation with different entry and exit z-score pairs to find the optimize z-score pairs.

(2) This research report is based on daily trading data; the same back-testing engine can be used to analyze the minute data, hour data and half data.

(3) The back-testing algorithm is not take slippage and trading fees into consideration.

(4) Further research can explore other filters instead of just Kalman filter.

(5) Another window to optimize is the length of the training period and how frequently the Kalman filter has to be recalibrated.

(6) The back-testing is based on main contracts data, in real trading the main contracts should be project to the special contracts in each month.

Conclusion:

Contrary to a more developed market, arbitrage opportunities are not readily realized which suggests there might be opportunities for those looking and able to take advantage of them. My project focuses on China's futures market using Statistical Arbitrage and Pair trading techniques. The project run Augmented Dickey Fuller test on the spread to confirm statistically whether the series is mean reverting or not, calculate Kalman Filter regression on the spread series and a lagged version of the spread series in order to then use the coefficient to calculate the half-life of mean reversion. The results show that though has a relative lower daily Sharpe ratio (2.87 vs. 3.67), the out-sample portfolio has a higher expected daily return, and that the out-sample portfolio has a relative higher CAGR (0.0858 vs. 0.07882) but also has a relative longer average drawdown days. The back-testing algorithm can be used to analyze the minute data、 hour data. The main limitation is that the back test is not take slippage and trading fees into consideration.

Bibliography

Xing Tao, Bachelor in Computer Science (LZU), master in Information System and Management Science (PKU), passed CFA level 1-3 exams. My current job is an investment manager of real estates, lands and

infrastructures. Trading is one of my hobbies since 5 years ago. I am trying to be quant, and trying to apply a PhD programing in computing finance.