FIN521 Investments

Statistical Arbitrage Trading Strategy

Group 2

1801212871 李艾洁 1801212853何欣蔚 1801212837邓欣禾 1801212857 黄跞览
1801212836 邓浩田 1801214715 Schedl, Christian 1801214699 Collantes Huacon, Jackson Alfredo

April 4, 2019

Contents

Part 1 Introduction of Statistical Arbitrage

1 What is Statistical Arbitrage?

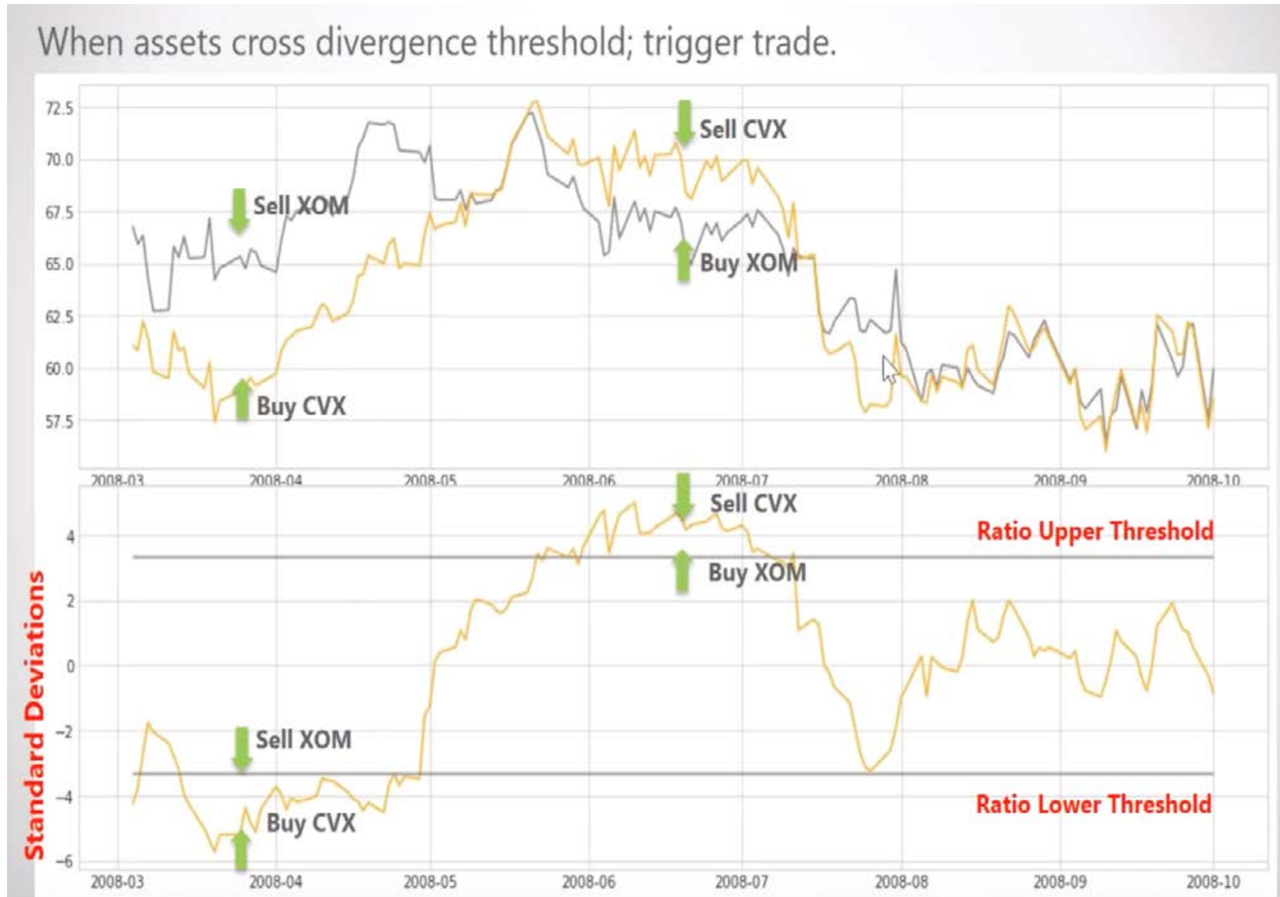
- **Statistical Arbitrage**
  - An attempt to profit from pricing inefficiencies that are identified through the use of mathematical models.
  - Gain benefit from the likelihood that prices will trend toward a historical norm. Unlike pure arbitrage, statistical arbitrage is not riskless.

- **Common features** of Statistical Arbitrage（Avellaneda and Lee, 2008）
  - Rules-based trading signals not driven by fundamentals
  - Market-neutral or zero beta with the market
  - Statistical mechanism for generating excess returns

1 Example: Pairs Trading

- Simplest form of statistical arbitrage
- How does it work?
  - Select 2 stocks that move together
  - Look out for any divergence in prices
  - When prices diverge, buy low-priced stock and sell high-priced stock
- Standardize prices to make stocks comparable
- Formulate upper and lower threshold
- Measure divergence as standard deviations from the mean

ExxonMobil (XOM) → Chevron (CVX)



Part 2 Trading Target

PHBS 北京大学汇丰商学院

Trading target: SSE 50ETF (上证50ETF）

- Advantages of SSE 50ETF:
  - **Better tracking results**
    - ✓A portfolio of large-cap blue chips: diversify firm-specific risks
  - **Lower transaction cost**
    - ✓Passive management strategy: low management fee
    - ✓High liquidity: low market impact cost
  - **Easier to form tracking portfolio**
    - ✓Higher transparency

Part 3 Construct a Tracking Portfolio

3 Construct a Tracking Portfolio

- To track the trading target, SSE 50ETF, we form a tracking portfolio with the same factor beta configuration, which implies:

$$R_{\text{ETF}} = E(R_{\text{ETF}}) + \beta_1 F_1 + \beta_2 F_2 + \cdots + \beta_{\text{k}} F_{\text{k}} + \varepsilon_{\text{ETF}}$$

$$R_{\text{TP}} = E(R_{\text{TP}}) + \beta_1 F_1 + \beta_2 F_2 + \cdots + \beta_{\text{k}} F_{\text{k}} + \varepsilon_{\text{TP}}$$

- We can use any K + 1 independent stocks to track the 50ETF under the K-factor model.

3.1 Step 1: Determine factors and PCA

Choice of factors

- **Bilson, Brailsford, and Hooper (2001)**
  - 27 emerging market returns, 1985-1997
  - Local macroeconomic factors (interest rates, goods prices, monetary supply changes, etc.)
  - Also used principal component analysis (PCA) for common sensitivity

Review of classics

- **Chen, Roll, and Ross (1986)**
  - 1953–1983, standard period = 1 month
  - Unexpected change in inflation, GNP, investor confidence
- **Burmeister and Wall (1986)**
  - Unexpected change in aggregate risk premium, term structure, inflation, and real final sales

3.1 Step 1: Determine factors and PCA

- Data source: WIND database/Marco part
- Frequency: monthly
- Estimation Period: January, 2010-December, 2017

Type	Number of factors	Factor indices
A: aggregate	5	Real GDP growth rates, fixed asset investment, aggregate financing
C: consumption	6	CPI, consumer satisfaction index, total retail sales
M: monetary and finance	8	M2, RMB loans of FIs, deposit rate, loan rate
P: production	6	PPI, value added to the industrial enterprises
T: trade and international	10	Exchange rate, FDI, trade balance
Total	35	

3.1 Step 1: Determine factors and PCA

Principal component analysis

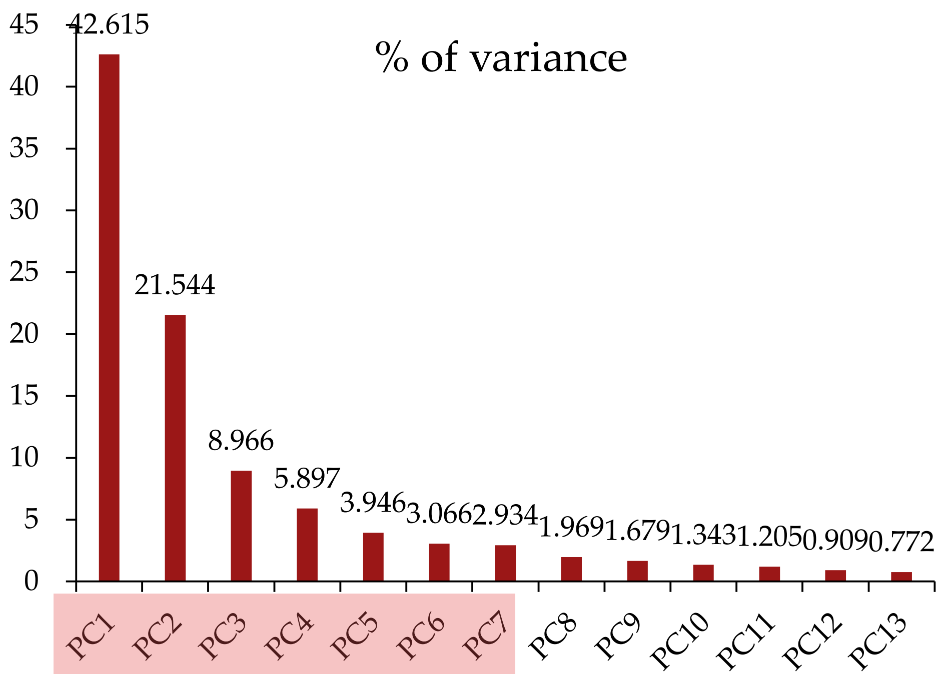
- Orthogonal transformation
  - Invented by Karl Pearson in 1910; developed and named by Harold Hotelling in the 1930s
  - Correlated variables (**raw factors**) → linear uncorrelated **principal components**
  - The first PC has the largest variance
  - Dimension reduction

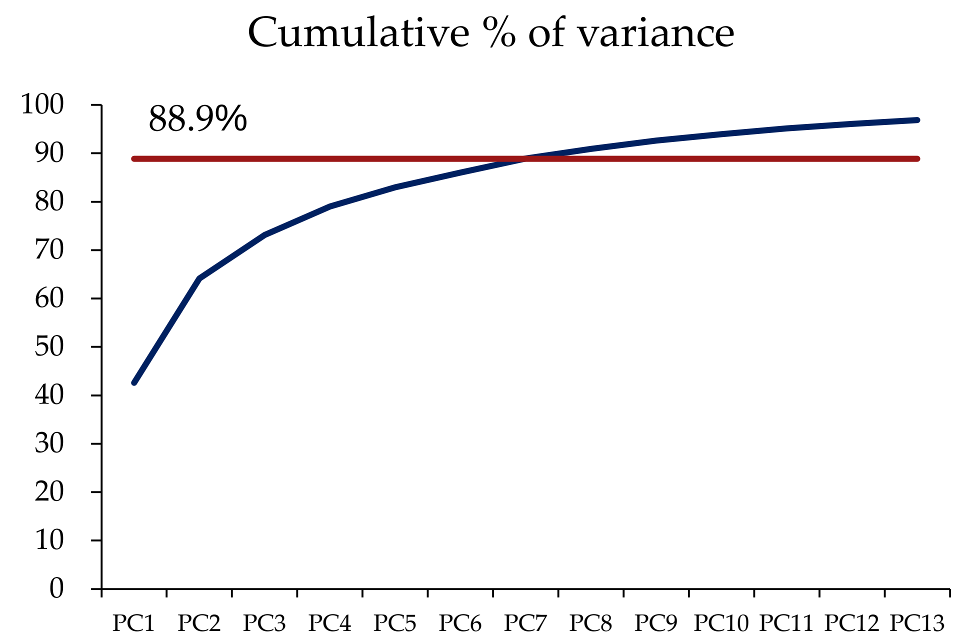
```python
'''Using PCA to reduce the dimension of 35 macro factors'''
def pca(dataMat,percentage=0.9):
    '''The pca function has two parameters where dataMat is a data set  and columns represent features;
       percentage indicates the variances that should be explained by PCs, the default is 0.9.'''
    meanVals=np.mean(dataMat,axis=0)  #Data standardization
    meanRemoved=dataMat-meanVals
    covMat=np.cov(meanRemoved,rowvar=0)  #cov()
    eigVals,eigVects=np.linalg.eig(np.mat(covMat))  #Find eigenvalues and eigenvectors
    k=eigValPct(eigVals,percentage) #The number of PCs required to interpret 90% of the variances
    print(k)

    eigValInd=np.argsort(eigVals)  #Sort eigenvalues from small to large
    eigValInd=eigValInd[:-(k+1):-1] #Take the first k eigenvalues
    redEigVects=eigVects[:,eigValInd]   #Returns eigenvectors corresponding to k eigenvalues
    lowDDataMat=meanRemoved*redEigVects #Dot product the original data and eigenvectors to obtain reduced-dimension data.
    return lowDDataMat

pca_35 = np.loadtxt("PCA_35factor.csv",delimiter=',')
lowDDataMat = pca(pca_35,0.88)
np.savetxt('pca_7pc.csv',lowDDataMat,delimiter=',')   #Return k PCs
```

3.1 Step 1: Determine factors and PCA

Principal components

- 35 factors → 7 principal components → 89% of total variance explained
- PCs are linear combinations of macroeconomic factors
- KMO measure = 0.833 > 0.7

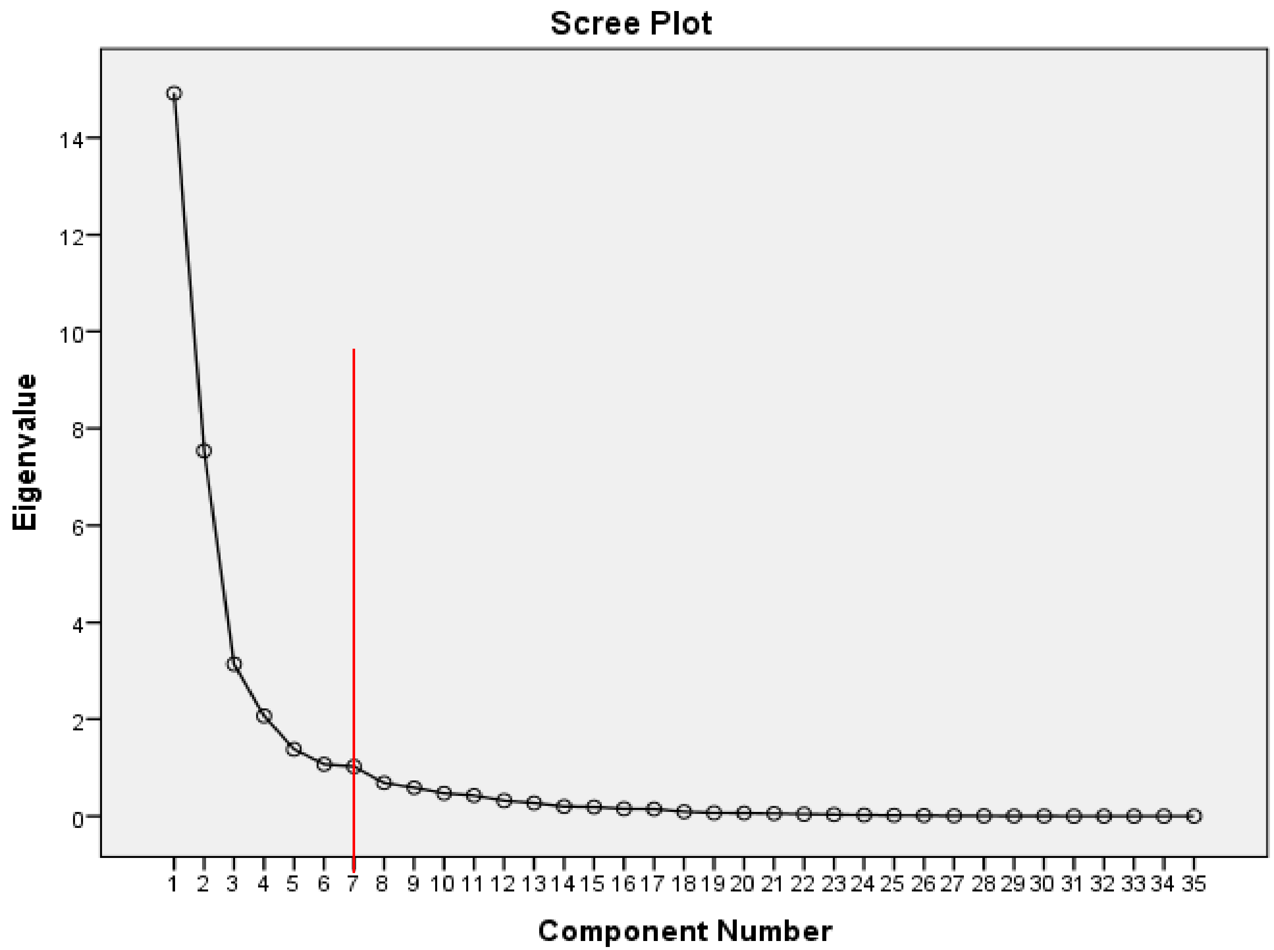




3.1 Step 1: Determine factors and PCA

Principal components

- 35 factors → 7 principal components → explain 89% of total variance
- PCs are linear combinations of macroeconomic factors
- KMO measure = 0.833 > 0.7



3.1 Step 1: Determine factors and PCA

	Component matrix						
	Principal Components						
	1	2	3	4	5	6	7
Fixed Asset Investments	0.923	-0.322	-0.111	0.002	0.083	0.038	0.035
Aggregate Financing to the Real Economy: YTD	-0.300	0.216	0.247	-0.472	0.187	0.498	0.384
Real GDP Growth Rate	0.919	0.125	-0.283	-0.022	0.003	0.060	-0.039
Consumer Price Index: Year-Over Year	0.784	-0.097	0.359	-0.011	-0.319	0.166	0.001
Total Retail Sales of Consumer Goods: YoY	0.916	-0.037	-0.109	-0.049	-0.027	0.200	0.032
Total Retail Sales of Consumer Goods	-0.868	0.337	0.191	0.024	-0.001	-0.055	0.074
M2: Monetary Aggregate	0.721	-0.041	-0.647	-0.043	0.019	0.101	-0.012
RMB Loans of Financial Institutions (FIs): M	0.725	0.131	-0.566	-0.016	-0.018	0.053	-0.126
FIs: New RMB Loans	-0.430	0.224	-0.168	0.745	-0.125	-0.128	0.061
Demand Deposit Interest Rate: M	0.559	-0.208	0.567	0.135	-0.267	-0.025	-0.213

…

pc principal components

f raw factors

M component matrix

$$\mathbf{pc} = \mathbf{M} \cdot \mathbf{f}$$

$$\begin{bmatrix} pc_1 \\ pc_2 \\ \cdots \\ pc_7 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{17} \\ m_{21} & m_{22} & \cdots & m_{27} \\ \vdots & \vdots & \ddots & \vdots \\ m_{35,1} & m_{35,2} & \cdots & m_{35,7} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \cdots \\ f_{35} \end{bmatrix}$$

3.2 Step 2: Compute factor betas—by regression

Data

- 2010/01–2017/12 monthly data of 7 factors, rate of return of SSE 50 ETF and constituent stock prices from Wind

Regression on Trading Target

$$R_{\text{ETF}} = E(R_{\text{ETF}}) + \beta_1 F_1 + \beta_2 F_2 + \cdots + \beta_7 F_7 + \varepsilon_{\text{ETF}}$$

- Conduct regression on the rate of return of SSE 50ETF to get the coefficient associated with 7 factors: $(\beta_1 \quad \beta_2 \quad \ldots \quad \beta_7)$

Regression on Constituent Stocks of SSE 50ETF

$$R_i = E(R_i) + \beta_{i1} F_1 + \beta_{i2} F_2 + \cdots + \beta_{i7} F_7 + \varepsilon_i, \qquad i = 1 \ldots 42$$

- There are 42 stocks with no missing values in all 7 dimensions across the entire period
- Conduct regression on the rate of return of 42 stocks, to get their betas associated with 7 factors respectively

$$B = \begin{pmatrix} \beta_{11} & \beta_{12} & \cdots & \beta_{17} \\ \vdots & \vdots & \vdots & \vdots \\ \beta_{42,1} & \beta_{42,2} & \cdots & \beta_{42,7} \end{pmatrix}_{42\times 7}$$

3.3 Step 3: Select stocks and compute weights

- In theory, we can use any 7 + 1 independent stocks to track 50ETF under the 7-factor model.
- According to the following equation, the weights of the 8 selected stocks can be calculated.

$$\begin{pmatrix} \beta_{11} & \beta_{21} & \cdots & \beta_{71} & \beta_{81} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \beta_{17} & \beta_{27} & \cdots & \beta_{77} & \beta_{87} \\ 1 & 1 & \cdots & 1 & 1 \end{pmatrix} \times \boxed{\begin{pmatrix} w_1 \\ \vdots \\ w_7 \\ w_8 \end{pmatrix}} = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_7 \\ 1 \end{pmatrix}$$

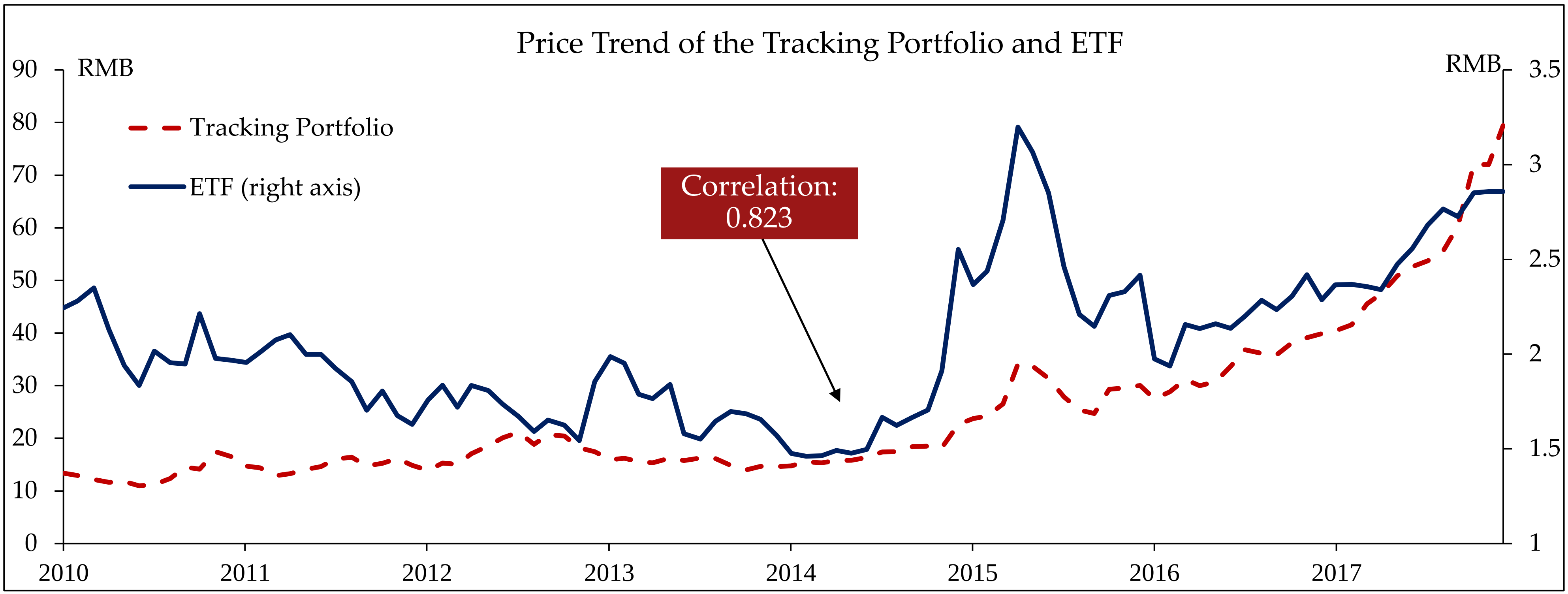
```
'''Matching risk parameters and choosing 8 stock to tarck ETF50'''
data=pd.read_csv("dataset2.csv",encoding='gbk')
data=data.drop(columns='日期')
coefs=np.zeros(shape=(43,7))#Create a coefficient matrix
for i in range(43):#Multiple Regression respectively
    newdata=data.iloc[:,[i,43,44,45,46,47,48,49]].dropna()#Take out the independent and dependent variables and remove the null value
    y=newdata.iloc[:,0]
    X=newdata.iloc[:,1:]
    linreg = LinearRegression()
    model=linreg.fit(X, y)#solve the regression
    coefs[i]=linreg.coef_


ran_index=['浦发银行','民生银行','宝钢股份','中国石化','南方航空','中信证券','招商银行','保利地产']#by random
#ran_index=np.random.choice(index,8)
ran_coefs=coefs.loc[ran_index,:].values.T#Take out the coefficient matrix and transpose
ones=np.ones(8)
ran_coefs=np.insert(ran_coefs,0,values=ones).reshape(8,8)#Construct coefficient matrix
coef=coefs.iloc[0,:].values
one=np.ones(1)
coef=np.insert(coef,0,values=one)
x=np.linalg.solve(ran_coefs,coef)#Solve weights
print("选取的股票为:"+ran_index1)
print("权重为: ")
print(x)
```

3.3 Step 3: Selection method 1—by industry

- By industry: choose 8 stocks from different industries among 42 stocks

Tracking Portfolio's Constituents and Weights

Stock name	招商银行	保利地产	恒瑞医药	贵州茅台	海螺水泥	中国平安	中国建筑	中国石油
Stock name in English	China Merchants Bank	Poly Developments	Hengrui Medicine	Kweichow Moutai	Conch Cement	Ping An Insurance	CSCEC	China National Petroleum
weight	-0.8189	-0.5210	0.6419	0.0751	-0.1485	0.2354	0.2003	0.2937

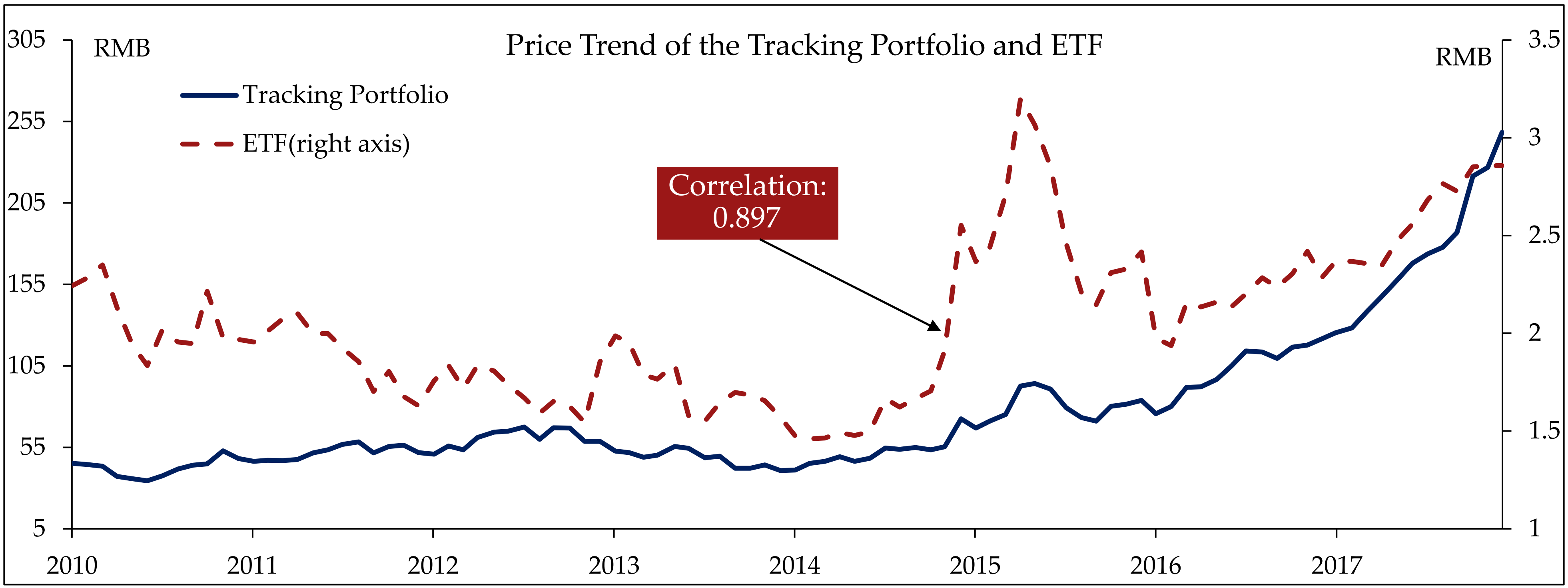


3.3 Step 3: Selection method 2—by size

- By size: choose 8 stocks with the highest free float market capitalization from 42 stocks

Tracking Portfolio's Constituents and Weights

Stock name	中国平安	招商银行	贵州茅台	兴业银行	民生银行	中信证券	浦发银行	伊利股份
Stock name in English	Ping An Insurance	China Merchants Bank	Kweichow Moutai	Industrial Bank	Minsheng Bank	CITIC Securities	Pudong Development Bank	Yili
weight	0.250	-0.391	0.331	0.242	-0.075	0.316	0.248	0.078

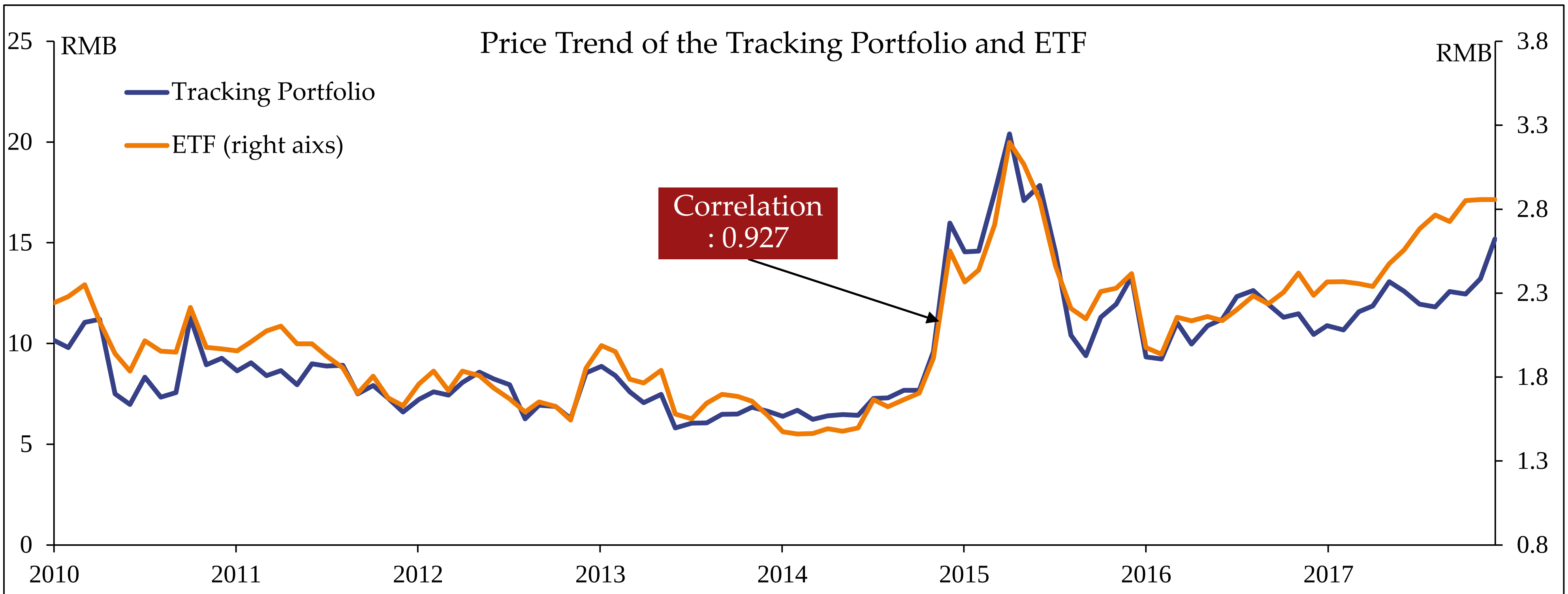


3.3 Step 3: Selection method 3—by random

- By random: randomly choose 8 stocks from 42 stocks

Tracking Portfolio's Constituents and Weights

Stock name	浦发银行	民生银行	宝钢股份	中国石化	南方航空	中信证券	招商银行	保利地产
Stock name in English	Pudong Development Bank	China Minsheng Bank	Baoshan Iron & Steel	China Petroleum & Chemical	China Southern Airlines	CITIC Securities	China Merchants Bank	Poly Developments
Weight	-0.025	-0.239	-1.376	1.092	0.628	0.407	0.046	0.466



Part 4 Arbitrage Strategy

4 Arbitrage strategy

Mean reversion

- The assumption that stock prices will go back to its mean once it deviates
- Identifying the trading range and computing the average price

Spread

- The spread, difference in prices, $S = P_{\text{tracking portfolio}} - P_{\text{ETF}}$
- Think of it as a new "stock"

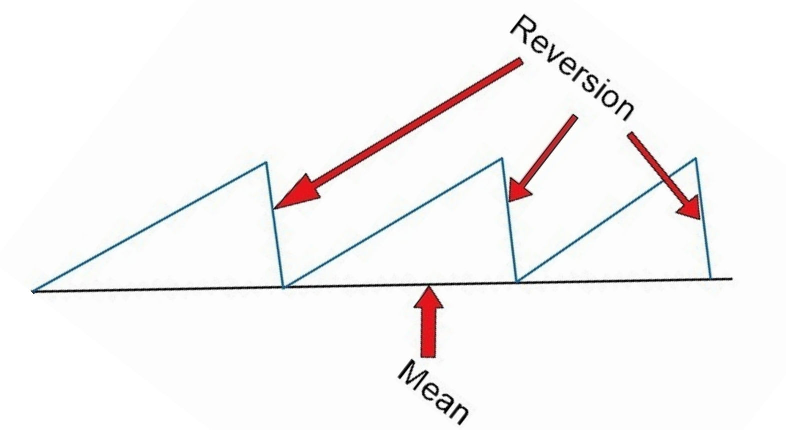
Normalization

- Dimensionless variable $x(t) = \dfrac{S(t) - \bar{S}}{\sigma_S}$

$\bar{S}$: In-sample mean of spread
σ_s : In-sample standard deviation of spread

Arbitrage opportunities

- Should be around zero
- If not, chances for arbitrage



PHBS 北京大学汇丰商学院 商界军校

Principles

■When $x(t)$ goes up ↑
- **Short sell** the "spread" stock
- If still up ↑ , buy back to **stop loss**
- If down ↓ , also buy back to **take profit**

Similarly

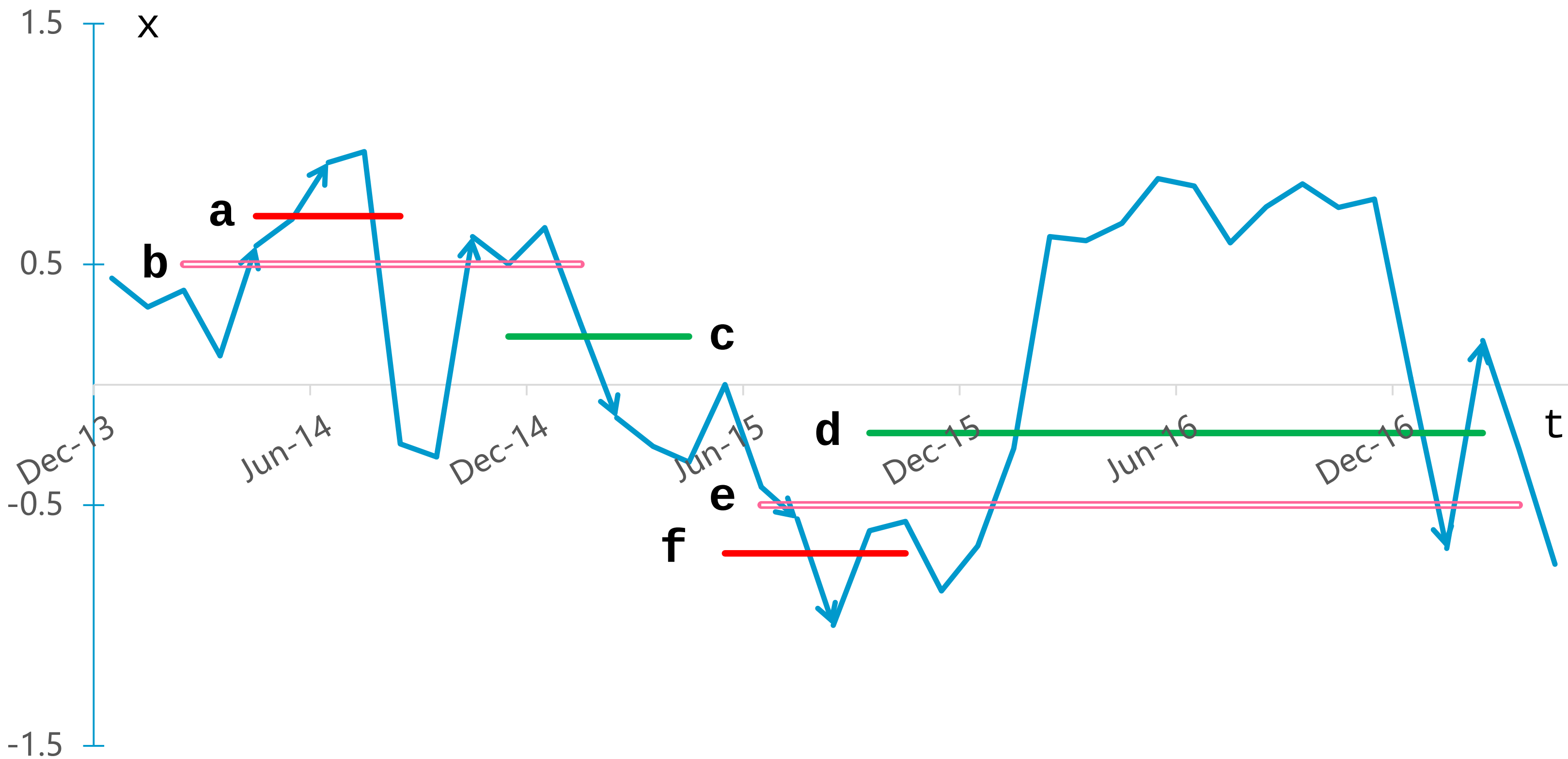
■When $x(t)$ goes down ↓
- We believe it will go up back to its mean
- Thus **long the stock**
- Also have to stop loss and take profit

Note that

■ $S = P_{\text{tracking portfolio}} - P_{\text{ETF}}$
- Long "S": buy TP, sell ETF
- Short "S": sell TP, buy ETF

4.1 Methodology

Spreads and trading signals



b: *upper threshold,* short "S" to open position

- **a**: *close position to* stop loss
- **c**: *close position to* take profit

e: *lower threshold,* long "S" to open position

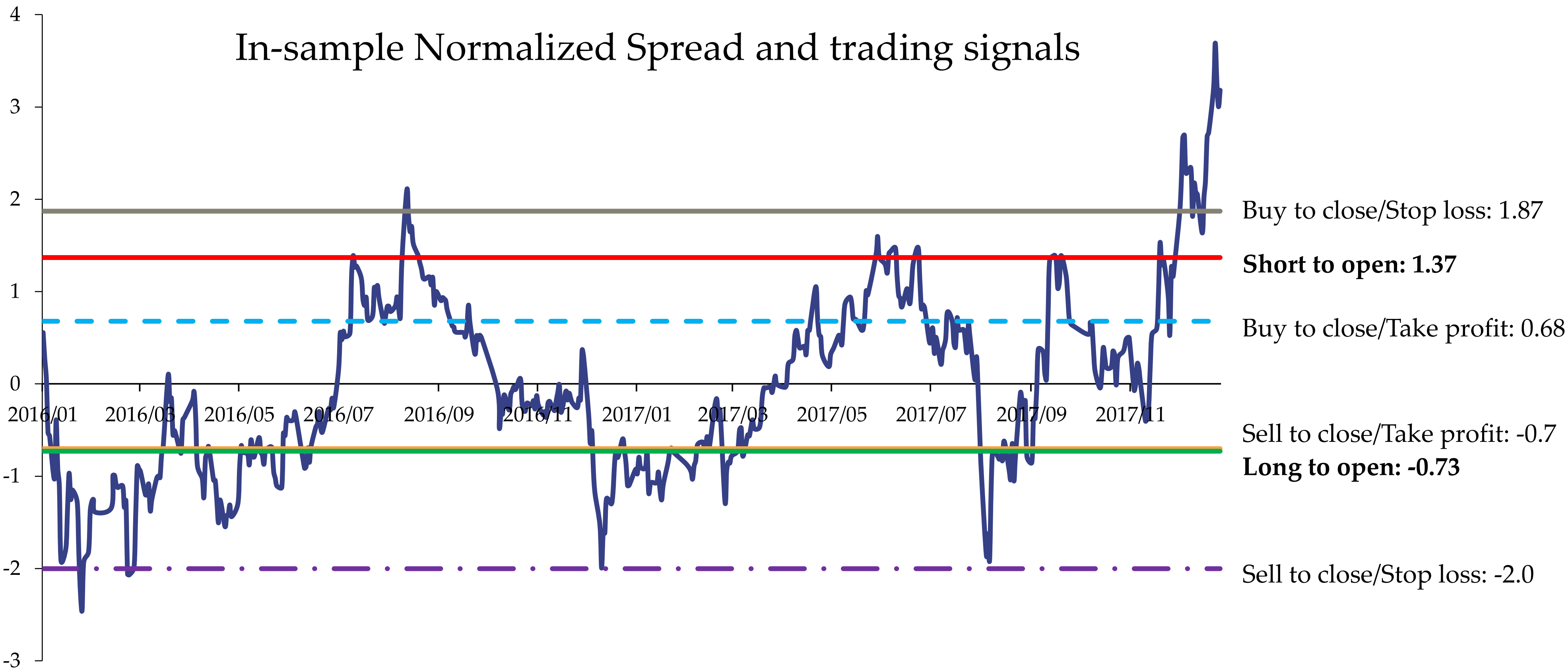
- **f**: *close position to* stop loss
- **d**: *close position to* take profit

4.2 In-sample estimation

- Time window: 2016-01-2017-12, daily closing prices of tracking portfolio and ETF
- Mean of spread from 2016 to 2017 : $\bar{S} = 9.23$
- Standard deviation of spread from 2016 to 2017 : $\sigma_s = 0.97$
- But how to determine trading signals?
  - ✓We choose the **optimal 6 trading signals** that maximize the profit between 2016-01 to 2017-12

4.2 In-sample estimation

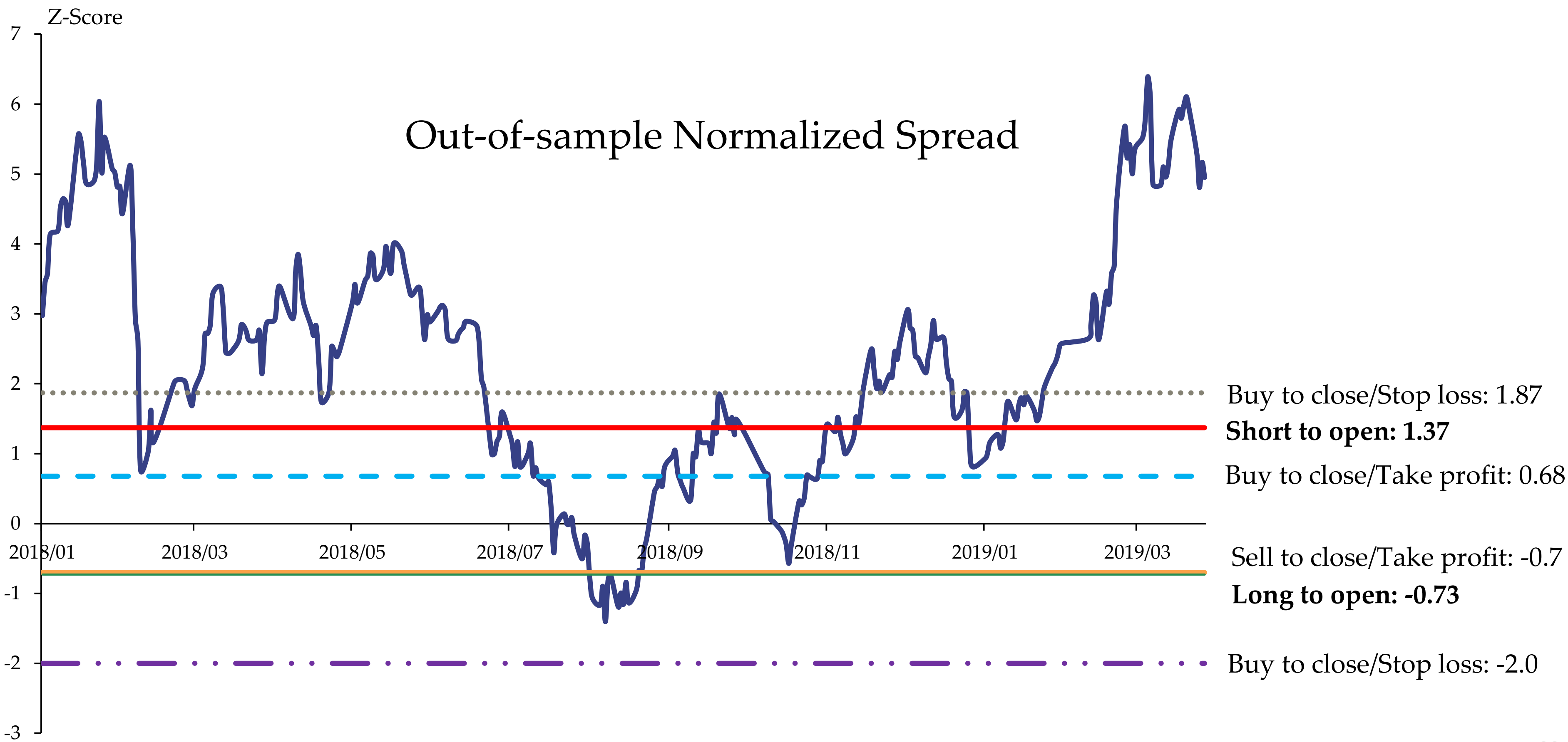
- Maximization results:



Part 5 Back-testing Results

5.1 Testing Results

- Out-of-sample: daily data from 2018-01-01 to 2019-03-28
- Return: **10.36%**
- Opening and closing position for 6 times over one year and three months

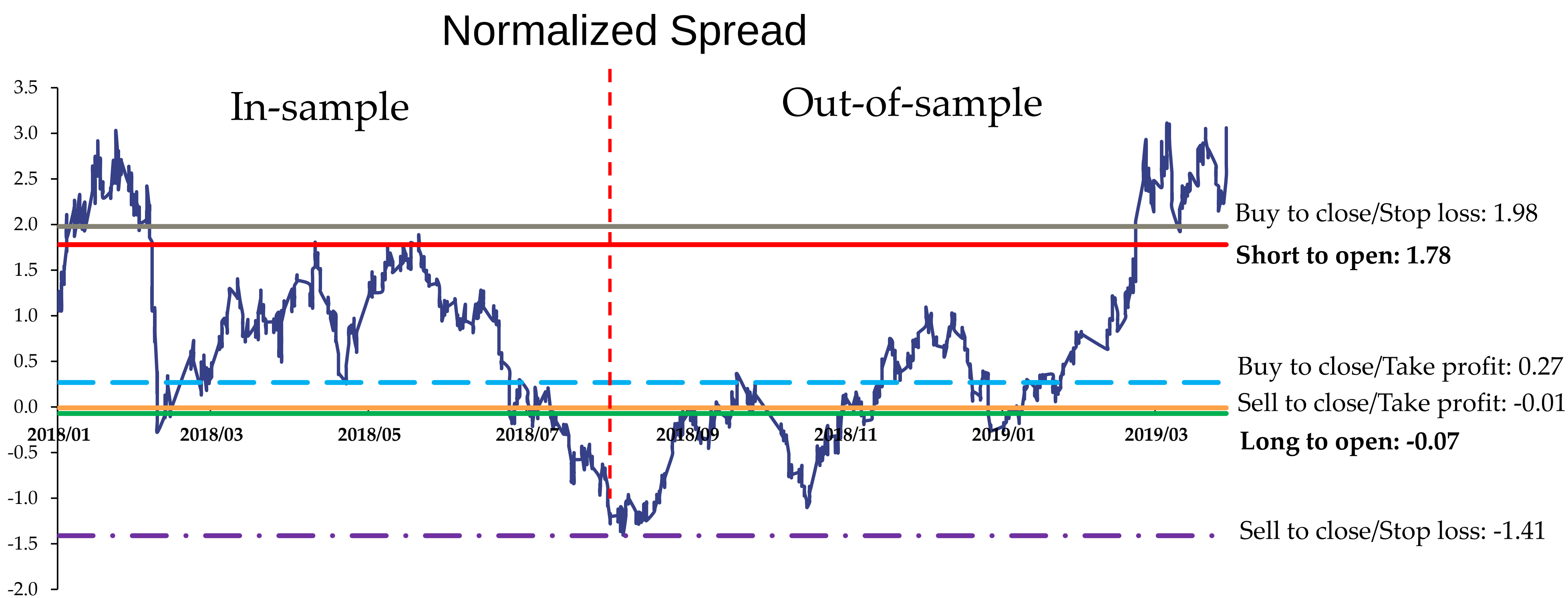


5.2 Trading Frequency

Data frequency: 30-minutely data

- In-sample estimation
  - ✓ Time window: 2018-01-02–2018-08-13
  - ✓ $\bar{S} = 10.45$, $\sigma_S = 1.61$
  - ✓ Trading signals as illustrated below

- Out-of-sample back-testing
  - ✓ Time window: 2018.08.14~2019.03.28
  - ✓ Return: **15.16%**
  - ✓ Trading frequency: 7 times over 7 months



5.3 Transaction Cost

- Opening or closing a position in statistical arbitrage involves two transactions at the same time: shorting tracking portfolio and longing ETF or shorting ETF and longing tracking portfolio
- Transaction cost when opening or closing a position: $\tau \times (P_{\mathrm{ETF}} + P_{\mathrm{TP}})$

  τ :transaction cost rate

Tax and Commissions

Stamp Tax	0.10000%
Certificate Management Fee	0.00200%
Securities Transaction Fee	0.00487%
Transfer Fee	0.00200%
Commission	<0.3%
Total Transaction Cost	<0.41%

Market Impact Cost

How much additionally a trader must pay over the initial price due to market slippage, i.e. the cost incurred because the transaction itself changed the price of the asset

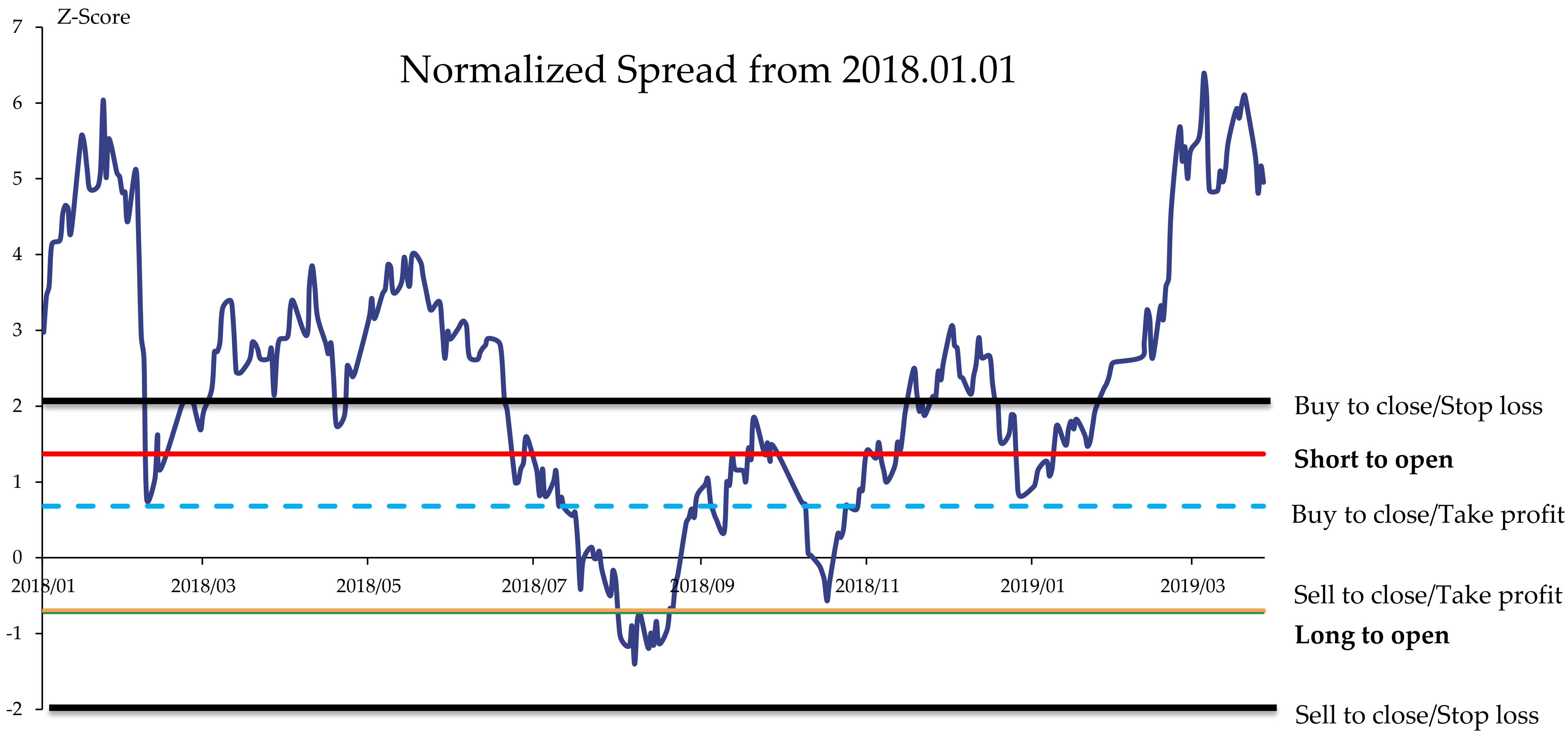
5.4 Strategy results

- Our strategy is still profitable with transaction cost. Transaction cost only leads to lower rate of return.
- Trading more frequently increases rate of return.

	Return without transaction cost	Return with transaction cost (0.4%)	Testing period
Daily frequency	10.36%	3.55%	2018/01/02–2019/03/28 Over 15 months
30-minute frequency	15.16%	8.8%	2018/08/14–2019/03/28 Over 7 months

5.5 Risk Control

- *N* times leverage will lead to *N* times more profit and also *N* times more loss. Higher leverage reduces risk tolerance
- Due to marginal call, stop-loss orders should be given earlier than without leverage



Review

- Statistical arbitrage
  - Introduction
  - Select the trading target
  - Construct a tracking portfolio
    - Macroeconomic factors
    - Principal component analysis and regression for β
    - By size, by industry, or at random
  - Arbitrage strategy
    - Mean reversion
    - In-sample data
  - Back-testing results
    - Trading frenquency
      - Daily
      - 30-minutely
    - Transaction cost
    - Risk control

Appendix: Back-testing and transaction cost code

PHBS 北京大学汇丰商学院

```python
'''Backtesting the return rate in 2018 without transaction cost'''
import numpy as np
import pandas as pd
from random import randint
import xlrd

#xs_show = [0.68, 1.37, 1.87] #In order:Take profit, Upper threshold, Stop loss, daily data
#xm_show = [-0.7, -0.73, -2.0]#In order:Take profit, Lower threshold, Stop loss
xs_show = [0.27, 1.78, 1.98]   #30min data
xm_show = [-0.01, -0.07, -1.41]
#ETF_data = pd.read_excel('2018至今Daily计算return.xlsx', dtype = np.float64)
ETF_data = pd.read_excel('2018下半年至今30min计算return.xlsx', dtype = np.float64)
df_all = ETF_data.iloc[::].values

df = df_all[...,:1]          #normalized_spread
df_s = df_all[...,1:2]       #Spread
df_k = df_all[...,2:3]       #Sum of price

def xm_f(df,xs):
    # Set a flag to indicate whether opened a position
    flag = False
    ware = 0
    sign = []  # Record the date of transaction
    i = 0    #Record the number of loops
    sum = 0  # Recorded return rate
    counter = 0  # Record the total number of transactions
    for df_x in df:
        if i == 0:
            i += 1
            continue
        if df[i - 1:i] > df[i:i + 1]:
            EQ = False
            RQ = True
        else:
            EQ = True
            RQ = False
```

Appendix: Back-testing and transaction cost code

```
        if flag == False:
            # if RQ == True and df[i - 1:i] < xs[1] and df[i:i + 1] > xs[1] and df[i:i+1] < xs[0]:
            #     flag = True
            #     sign.append(i)
            if RQ == True and df[i - 1:i] > xs[1] and df[i:i + 1] <= xs[1] and df[i:i+1] > xs[2]:
                flag = True
                ware = df[i:i + 1]
                sign.append(i)
        else:
            if EQ == True and df[i - 1:i] < xs[0] and df[i:i + 1] >= xs[0]:
                sign.append(i)
                sum += (df_x - ware)
                counter += 1
                flag = False
            elif EQ == False and df[i - 1:i] > xs[2] and df[i:i + 1] <= xs[2]:
                sign.append(i)
                sum += (df_x - ware)
                counter += 1
                flag = False
        i += 1
    return sum, counter, sign

sum_xs, counter_xs, sign_xs = xs_f(df, xs_show)
sum_xm, counter_xm, sign_xm = xm_f(df, xm_show)

print("总收益率：")
print(sum_xm+sum_xs)
print("交易路径：")
print(sign_xs)
print(sign_xm)

'''Backtesting in 2018 and considering transaction costs'''
cost_xs = 0
cost_xm = 0

for i in range(counter_xs):
    cost_xs += 4/1000 * (df_k[sign_xs[2*i]] + df_k[sign_xs[2*i+1]])

for i in range(counter_xm):
    cost_xm += 4/1000 * (df_k[sign_xm[2*i]] + df_k[sign_xm[2*i+1]])

print("总cost:")
print(cost_xm+cost_xs)
```

References

- Andersen, H., & Tronvoll, H. (2018). Statistical arbitrage trading with implementation of machine learning: an empirical analysis of pairs trading on the Norwegian stock market. Retrieved from https://brage.bibsys.no/xmlui/handle/11250/2561266
- Avellaneda, M., & Lee, J.-H. (2010). Statistical arbitrage in the US equities market. *Quantitative Finance*. https://doi.org/10.1080/14697680903124632
- Bilson, C. M., Brailsford, T. J., & Hooper, V. J. (2001). Selecting macroeconomic variables as explanatory factors of emerging stock market returns. *Pacific-Basin Finance Journal*, *9*(4), 401–426. https://doi.org/10.1016/S0927-538X(01)00020-8
- Chen, N.-F., & Ingersoll, J. E. (1983). Exact Pricing in Linear Factor Models with Finitely Many Assets: A Note. *The Journal of Finance*, 38(3), 985–988. https://doi.org/10.2307/2328092
- Chen, N.-F., Roll, R., & Ross, S. A. (1986). Economic Forces and the Stock Market. *The Journal of Business*, 59(3), 383-403. Retrieved from https://www.jstor.org/stable/2352710
- Interactive Brokers. (n.d.). *QuantConnect - Pairs Trading with Python*. Retrieved from https://www.youtube.com/watch?v=cZFqYJmQoTM&t=315s
- Roll, R., & Ross, S. A. (1980). An Empirical Investigation of the Arbitrage Pricing Theory. *The Journal of Finance*, *35*(5), 1073–1103. https://doi.org/10.1111/j.1540-6261.1980.tb02197.x
- Shanghai Stock Exchange. (n.d.). Shanghai Stock Exchange. Retrieved April 3, 2019, from http://english.sse.com.cn/products/equities/overview/
- Wind Economic Database. (n.d.). Wind-Economic Database. Retrieved April 3, 2019, from https://www.wind.com.cn/en/edb.html