ICM142 – Programming for Finance Assignment

Yue Gong
(25807255)

1) What are the basics of interaction of cryptography and economics?

The interaction of cryptography and economics is the economic principle and theory focusing on the support block chain and its derivative application. It focuses on game theory and incentive design, which is mainly used in the design of block chain mechanism.
On the contrary, system encryption economics focuses on institutional economics in block chain and encryption economics. The economy, like the institutional economics of its branches, is a system of coordinated exchange.

What is your fundamental understanding of Bitcoin and blockchain technology?

The blockchain is a digital ledger that has a record of all transactions across a peer-to-peer network. From this technology, people can confirm trade such as fund transfer, and voting, by the internet without the central authority. Meanwhile, Bitcoin is well known as cryptocurrency, which is a medium of exchange such as US dollar, when the blockchain was invented. In addition, cryptocurrency is used to control the establishment of monetary units and verify the transfer of funds by encryption techniques.

2) key events for Cryptocurrencies from February 15, 2017 till February 15, 2018

1. 2017,09,29

Korea bans ICOs, but ETH holds strong.
BTC price goes down by 20.1 from 4190 to 4169.9 USD.

2. 2017,10,20

Bitcoin passes $100B in market capitalization.
BTC price goes up by 294.5 from 5,698.6 to 5,993.1 USD.

3. 2017,10,24

Bitcoin again forks into two digital currencies, BTC and BTG
BTC price goes down by 390.5 from 5,903.6 to 5,513.1 USD.

4. 2017.11.08

Bitcoin SegWit2x Hard Fork to be cancelled
BTC price goes up by 341.6 from 7102.8 to 7444.4 USD.

5. 2017,11,27.

Liquidity network makes off-chain protocol for Ethereum
BTC price goes up by 414.8 from 9318.4 to 9733.2 USD.

6. 2017,12,11

Bitcoin price recovers from a low of $13K as CBOE's future contracts go live.
BTC price goes up by 1673.5 from 15,059.6 to 16,732.5 USD.

7. 2017,12,20

A Bitcoin exchange in South Korea (YouBit) goes bust after nearly 2/5ths of customers BTC are stolen.
BTC price goes down by 920 from 17345 to 16425 USD.

8. 2018,01,02

Peter Thiel's Founders Fund makes monster bet on bitcoin
BTC price goes up by 445.18 from 13,444.9 to 14,754.1 USD

9. 2018,01,03

Ripple sets all-time price high.
BTC price goes up by 446 from 14709.82 to 15155 USD.
10. 2018,01,19
Staff letter: engaging on fund innovation and cryptocurrency-related holdings.
BTC price goes up by 334.8 from 11,245.4 to 11,580.2 USD.
11. 2018,01,24
South Korea announce that they will not ban cryptocurrency trading, but ban on anonymous trading and cracks down on money laundering.
BTC price goes up by 595 from 10819 to 11414 USD.
12. 2018,01,26
Starbucks chairman states they are keen on crypto, but not on Bitcoin.
BTC price goes down by 76 from 11146 to 11070 USD.
13. 2018,01,30
U.S. regulators subpoena crypto exchange bitfinex
BTC price goes down by 1078.8 from 11,244.8 to 10,166.0 USD.
14. 2018,02,05
Major banks in UK ban purchase of cryptocurrencies on credit cards.
BTC price goes down by 1250.1 from 8200 to 6949.9 USD.
15. 2018,02,13
Dubai trader gets first Mideast license in cryptocurrencies
BTC price goes down by 364.3 from 8,903.5 to 8,539.2 USD.
16. 2018,02,14
Bitcoin rises back above $9,000 while Saudi Banks announce tests for Ripple payment technology.
BTC price goes up by 939.5 from 8515.9 to 9455.4 USD.

Source: Adapted from Kornilov, D. el al. (2018). *Cryptocurrency and ICO Market Overview for 2017*. Retrieved from https://www.coinspeaker.com/2018/01/04/cryptocurrency-ico-market-overview-2017. Kornilov, D. el al. (2018).

Figure 1: Bitcoin Price
Source: Adapted from Investing website. Retrieved from https://uk.investing.com/crypto/bitcoin/chart

Figure 1 states a lot of great change of price in Bitcoin from end of September, 2017 to February, 2018. In addition, there is a smooth curve from February, 2017 to September, 2017, which means Bitcoin price had little change in this time interval.

These result can be found from Python Code:

The mean return on key events dates is: 0.005769226800776739

The median return on key events dates is 0.028126328700856185

The standard deviation on key events dates is: 0.07503831960559568

The correlation between key events variable and BTC is: -0.002125515675878762

The correlation between key events variable and ETH: 0.02817646825460795

The correlation between key events variable and XRP: -0.01590000200532828

The correlation between key events variable and Nasdaq: 0.02360323334877748

Figure 2: OLS Regression Results

```
==============================================================================
Dep. Variable:                      y   R-squared:                       0.711
Model:                            OLS   Adj. R-squared:                  0.639
Method:                 Least Squares   F-statistic:                     9.848
Date:                Tue, 07 Aug 2018   Prob (F-statistic):            0.00148
Time:                        23:35:29   Log-Likelihood:                 27.605
No. Observations:                  16   AIC:                            -47.21
Df Residuals:                      12   BIC:                            -44.12
Df Model:                           3
Covariance Type:            nonrobust
================================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
key_events      0.0167      0.013      1.333      0.207      -0.011       0.044
BTC_keys       -0.0996      0.168     -0.593      0.564      -0.465       0.266
XRP_keys        0.5391      0.169      3.186      0.008       0.170       0.908
Nasdaq_keys     1.7443      1.331      1.310      0.215      -1.157       4.645
==============================================================================
Omnibus:                        2.279   Durbin-Watson:                   2.163
Prob(Omnibus):                  0.320   Jarque-Bera (JB):                1.329
Skew:                           0.704   Prob(JB):                        0.514
Kurtosis:                       2.889   Cond. No.                         107.
```

Figure 2 states the OLS regression results. The input variables(independent variables) include the days that have key events, and the returns of BTC, XRP and Nasdaq in key events days, while output variable(dependent variable) is the return of ETH in key events days. In this figure, R-squared is 0.711, which means that the model approximates the real data points well. In addition, because of low p-value of XRP(< 0.05), the regressor XRP returns. on key events days is significant at the significant level at 0.05 for ETH returns. Meanwhile, Durbin-Watson value is 2.163, which states there is no autocorrelation, and high Cond. No. states that there is a multicollinearity in this model.

3) Function called assess_portfolio()

To get important statistics from assess_portfolio function, these inputs need to be given:
(1) A date range: [‘2017/15/02’, ‘2018/15/02’]
(2) Symbols: [‘BTC’, ‘ETH’, ‘XRP’, ‘LTC’]
(3) Portfolio allocations for each asset: [0.2, 0.3, 0.4, 0.1]
(4) Total starting value of the portfolio: 1000000
(5) Daily risk-free rate(usually LIBOR)

The results (outputs) would be given from the execution and are all rounded to five decimal places:
(1) Cumulative return by adding all daily return in the given time interval:
The cumulative return is 4.22574.
(2) Average period return:
The average period return is 0.01158.
(3) Standard deviation of daily returns:

The standard deviation of return is 0.06353.

(4) Sharp ratio of the overall portfolio, given daily risk-free rate based on expected return and average return and the standard deviation of return:

The sharp ratio is: 66.33871.

(5) Moving historical volatility. 15 days rolling window is assumed and to calculate the volatility of previous 15 days return. For the first 14 days, there is no moving historical volatility, so ‘NON’ is used to represent no data. Because there is no return in 2017/15/02, so the starting date is 2017/16/02.

Figure 3. Moving Historical Volatility

Date	Moving Historical Volatility
2017/16/02	NON
2017/17/02	NON
2017/18/02	NON
2017/19/02	NON
2017/20/02	NON
2017/21/02	NON
2017/22/02	NON
2017/23/02	NON
2017/24/02	NON
2017/25/02	NON
2017/26/02	NON
2017/27/02	NON
2017/28/02	NON
2017/01/03	NON
2017/02/03	NON
2017/03/03	0.0236875
2017/04/03	0.0240986
2017/05/03	0.0267774

(6) Ending value of the portfolio:

The ending value of the portfolio is 10649410.79515.

Therefore, these output returns are positive while the daily are not all positive. From the result, the cumulative return is positive (4.22574), which means the price increases totally, while the average return is positive as well (0.01158). Meanwhile, standard deviation of return （0.06353) is used to measure the investment's volatility, which presents most of the returns are close to the average return. In addition, sharp ratio means the average return in excess of the risk-free rate per unit of volatility over risk (Sharp, 1966). The sharp ratio of the portfolio is 66.33871, which means the portfolio has the better excess return.

4) Event Study of a Specific Market event in the Cryptocurrency Market

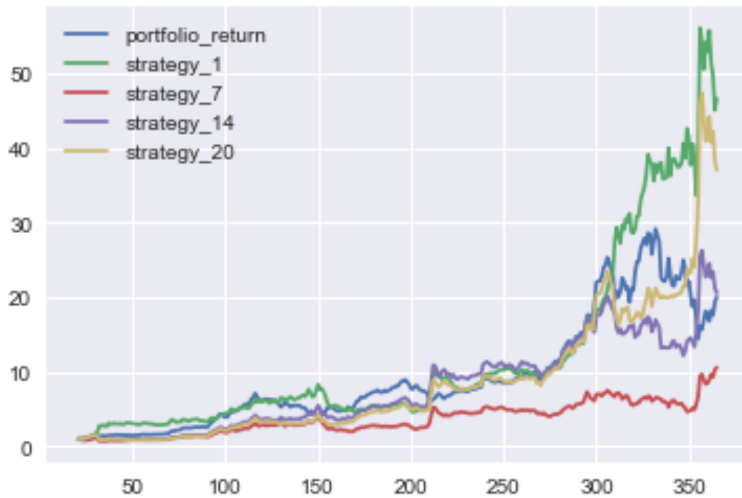
The event is when the daily median price/daily close price of the cryptocurrency is 15% lower than the previous day. The time period is from February 15, 2017 to February 15, 2018. After choosing BTC and ETH relatively liquid cryptocurrencies, their weights are equal to form a portfolio.

As market events happen (daily return is less than -0.15), triple their next-day position.

Based on and Hedging and Momentum strategy, to derive the absolute performance of the momentum strategy for the different momentum intervals (in 1,7,14,30 days). And we need to multiply the positioning's derived above(shifted by one day) by the BTC and ETH daily return in the portfolio.

The trading strategy plot is below.

Figure 4. Trading Strategy Plot



According to the figure, with 1, 7, 14, 30 days, strategy_1 is better than strategy_7, strategy_14, and strategy_30. Meanwhile, strategy_7 has the worst situation. Portfolio return line is the portfolio return without strategy, which is worse than strategy_1 and strategy_20, but better than others.

5) Backtesting

Price volatility can be used to reflect the situation in cryptocurrency market. Then volatility higher than 1.6 and the rolling window is 8 days.
In some days, strategy_14 and strategy_8 are better than portfolio return and

Figure 5. Trading Strategy Plot

The entry date is 2017-04-26, when
price volatility is larger than 1.6
The every trade between
2017-04-26 00:00:00 and
2017-04-27 00:00:00 is:55.58
Opportunities are expected 1 times per year.

Reference:
Sharpe, W. F. (1966). Mutual Fund Performance. *Journal of Business*. 39 (S1): 119–138

Appendix A. Python Code for Question (2)
Appendix B. Python Code for Question (3)
Appendix C. Python Code for Question (4)
Appendix D. Python Code for Question (5)
Appendix E. Excel 

Appendix A. Python Code for Question (2)

```
from xlrd import open_workbook
import numpy as np
import statsmodels.api as sm
import pandas as pd
from pandas import Series, DataFrame
from xlrd import xldate_as_tuple
import datetime

#importing the data from xlsx file
def read_from_excel():
    book = open_workbook('/Users/joyce/Desktop/Python/Python/Currencies.xlsx')
    sheet = book.sheet_by_index(0)
    dates = []
    BTC_prices = []
    ETH_prices = []
    XRP_prices = []
    Nasdaq_prices = []
    key_events = []
    LTC_prices = []
    LIBOR_rate = []
    for i in range(1,sheet.nrows):
        dates.append(datetime.datetime(*xldate_as_tuple(sheet.cell_value(i, 0), 0)).strftime('%Y/%d/%m'))
        BTC_prices.append(sheet.cell_value(i,1))
        key_events.append(sheet.cell_value(i,2))
        ETH_prices.append(sheet.cell_value(i,3))
        XRP_prices.append(sheet.cell_value(i,4))
        Nasdaq_prices.append(sheet.cell_value(i,5))
        LTC_prices.append(sheet.cell_value(i,6))
        LIBOR_rate.append(sheet.cell_value(i,7))
    return dates,BTC_prices,key_events,ETH_prices,XRP_prices,Nasdaq_prices,LTC_prices,LIBOR_rate


#2
def calculate_return_from_prices(input_prices):
    returns = []
```

```
    for i in range(1,len(input_prices)):
        log_price_in_position_i = np.log(input_prices[i])
        log_price_in_position_i1 = np.log(input_prices[i-1])
        return_in_position_i = log_price_in_position_i -log_price_in_position_i1
        returns.append(return_in_position_i)
    return returns

# delete all 0 prices from Nasdaq price, because there are only 252 working days on Nasdaq
def delete_0_nasdaq(key_events,nasdaq_prices):
    nasdaq = []
    nasdaq_keys = []
    for i in range(len(key_events)):
        if nasdaq_prices[i] != 0:
            nasdaq.append(nasdaq_prices[i])
            nasdaq_keys.append(key_events[i])
    return nasdaq_keys,nasdaq

def returns_on_key_events_dates(all_returns,key_events):
    #keep returns only for key events dates
    returns_on_key_events_dates = []
    for i in range(len(all_returns)):
        if key_events[i] == 1:
            returns_on_key_events_dates.append(all_returns[i])
    return returns_on_key_events_dates

def median_return_key(returns_on_key_events):
    N = len(returns_on_key_events)
    returns_on_key_events.sort()

    #find the length is even or odd
    if (N%2==0): #if even
        m1 = N/2
        m2 = (N/2)+1

    #Convert to integer
        m1 = int(m1)-1
        m2 = int(m2)-1
        median = (returns_on_key_events[m1]+returns_on_key_events[m2])/2

    else: #if odd
        m = (N+1)/2
        m = int(m)-1
        median = returns_on_key_events[m]

    return median

#according to the formula of standard deviation
#calculate its mean return and sum of list

def average_list(L1):
    average_list = sum(L1) / len(L1)
    return average_list

def variance_list(L1):
```

```
    average = average_list(L1)
    variance = 0
    for x in L1:
        diff_sq = (average-x)**2
        variance += diff_sq
    variance /= len(L1)
    return variance

def std_dev(L1):
    variance = variance_list(L1)
    return np.power(variance, 0.5)

def corr_x_y(x,y):
    n = len(x)
    #find the sum of products
    products = []
    for i, j in zip(x,y):
        products.append(i*j)
    sum_pro_x_y = sum(products)
    sum_x = sum(x)
    sum_y = sum(y)
    squared_sum_x = sum_x ** 2
    squared_sum_y = sum_y ** 2
    x_square = []
    for ii in x:
        x_square.append(ii**2)
    x_square_sum = sum(x_square)
    y_square = []
    for jj in y:
        y_square.append(jj**2)
    y_square_sum = sum(y_square)
    #use the formula to calculate
    numerator = n*sum_pro_x_y - sum_x*sum_y
    denominar_1 = n*x_square_sum-squared_sum_x
    denominar_2 = n*y_square_sum-squared_sum_y
    denominar = (denominar_1*denominar_2)**0.5
    correlation = numerator / denominar
    return correlation

def get_key_events(key_events):
    have_key_events = []
    for i in key_events:
        if i == 1:
            have_key_events.append(i)
    return have_key_events

def     calculate_regression(y,X):
    X = sm.add_constant(X)
    model = sm.OLS(y,X).fit()
#predictions = model.predict(X)
    result = model.summary()
    return result
```

```
def delete_zero_rate(rates):
    rate = []
    for i in rates:
        if i != 0:
            rate.append(i)
    return rate

[dates,BTC_prices,key_events,ETH_prices,XRP_prices,Nasdaq_prices,LTC_prices, LIBOR_rate] =
read_from_excel() #export data
have_key_events = get_key_events(key_events) #get the days that have key events
[Nasdaq_keys,Nasdaq] = delete_0_nasdaq(key_events,Nasdaq_prices)#delete all 0 prices from Nasdaq
prices
return_on_all_BTC = calculate_return_from_prices(BTC_prices)
return_on_all_ETH = calculate_return_from_prices(ETH_prices)
return_on_all_XRP = calculate_return_from_prices(XRP_prices)

return_on_all_Nasdaq = calculate_return_from_prices(Nasdaq) #get all returns
return_on_key_events_BTC = returns_on_key_events_dates(return_on_all_BTC, key_events[1:])
return_on_key_events_ETH = returns_on_key_events_dates(return_on_all_ETH, key_events[1:])
return_on_key_events_XRP = returns_on_key_events_dates(return_on_all_XRP, key_events[1:])
return_on_key_events_Nasdaq = returns_on_key_events_dates(return_on_all_Nasdaq, Nasdaq_keys[1:])

#
#print average BTC return on the key events
print("The mean return on key events dates is:", average_list(return_on_key_events_BTC))
#print median return on key events
print("The median return on key events dates is", median_return_key(return_on_key_events_BTC))
#print standard deviation on key events
print("The standard deviation on key events dates is:", std_dev(return_on_key_events_BTC))

#correlation between key events variable and BTC
corr_btc_key = corr_x_y(key_events[1:],return_on_all_BTC)
print("The correlation between key events variable and BTC is:", corr_btc_key)
#correlation between key events variable and ETH
corr_eth_key = corr_x_y(key_events[1:],return_on_all_ETH)
print("The correlation between key events variable and ETH:", corr_eth_key)
#correlation between key events variable and XRP
corr_xrp_key = corr_x_y(key_events[1:],return_on_all_XRP)
print("The correlation between key events variable and XRP:", corr_xrp_key)
#correlation between key events variable and Nasdaq
corr_nas_key = corr_x_y(Nasdaq_keys[1:], return_on_all_Nasdaq)
print("The correlation between key events variable and Nasdaq:", corr_nas_key)
#
#x usually means our input variable in dataframe type
#x includes the the days that have key events and the returns of BTC, XRP, Nasdaq in key events days
X = DataFrame({"key_events": have_key_events, "BTC_events": return_on_key_events_BTC,
           "XRP_keys":return_on_key_events_XRP,"Nasdaq_keys":return_on_key_events_Nasdaq})

#y means dependent variable and the output
#y includes the return of ETH in key events days
y = return_on_key_events_ETH
print("The regression result is: \n", calculate_regression(y,X))
```

Appendix B. Python Code for Question (3)

```
# Question 3
#rolling window is 15 days
def calculate_mhist_volatility(portfolio_return):
    mhist_volatility = []
    for i in range(len(portfolio_return)):
        if i < 15:
            mhist_volatility.append('NON')# the first 15 data have no historical volatility
        else:
            mhist_volatility.append(std_dev(portfolio_return[i-15:i]))
    return mhist_volatility

def portfolio_daily_value(portfolio_price, start_price):
    portfolio_value = [start_price]
    for i in range(1, len(portfolio_price)):
        simple_rate = (portfolio_price[i] - portfolio_price[i-1])/portfolio_price[i-1]
        portfolio_value.append(portfolio_value[i-1]*(1+simple_rate))
    return portfolio_value

def assess_portfolio(date_range, symbols, portfolio_weights, start_value, risk_free_rate):
#dates list has two elements including start and end date
    df = pd.read_excel('/Users/joyce/Desktop/Python/Python/Currencies.xlsx')
    price_dates = {}
    Date = df['Date'].dt.strftime('%Y/%d/%m')
    #calculate daily return in the given time interval of each symbol
    for i in range(len(Date)):
        if Date[i] == date_range[0]: #Start_time and report its index
            j = i
            days = 0# calculate how many days in the time interval
            while Date[j] != date_range[1]:#until end date
                for k in range(len(symbols)):
                    if symbols[k] not in price_dates:
                        price_dates[symbols[k]] = [df[symbols[k]][j]]
                    else:
                        price_dates[symbols[k]].append(df[symbols[k]][j])
                j += 1
                days += 1
            for l in range(len(symbols)):
                price_dates[symbols[l]].append(df[symbols[l]][j]) #add end date data to each symbol
    return_daily = {}
    for k in range(len(symbols)):
        return_daily[symbols[k]] = calculate_return_from_prices(price_dates[symbols[k]])

    #calculate portfolio daily return in the given time interval
    portfolio_return = []
    portfolio_price = []
    for i in range(days):
        returns = 0
        prices = 0
        for j in range(len(symbols)):
            returns += return_daily[symbols[j]][i] * portfolio_weights[j]
            prices += price_dates[symbols[j]][i] * portfolio_weights[j]
        portfolio_return.append(returns)
```

```
        portfolio_price.append(prices)
    #calculate cumulative portfolio return in the time interval
    cumulative_return = np.sum(portfolio_return)
    #calculate the average period return
    average_return = np.mean(portfolio_return)
    #calculate Standard deviation of daily returns
    std_dev_return = std_dev(portfolio_return)
    #sharp_ratio = (expected portfolio return - risk_free rate) / standard deviation
    sharp_ratio = (cumulative_return - risk_free_rate)/std_dev_return
    # moving rolling window = 15
    mhist_volatility = calculate_mhist_volatility(portfolio_return)
    #calculate portfolio daily value
    end_value = portfolio_daily_value(portfolio_price, start_value)[-1]
    return portfolio_return,cumulative_return, average_return, std_dev_return,sharp_ratio,
mhist_volatility,end_value


#
return_on_all_LTC = calculate_return_from_prices(LTC_prices)
#assess portfolio inputs
date_range = ['2017/15/02','2018/15/02']
symbols = ['BTC', 'ETH', 'XRP', 'LTC']
portfolio_weights = [0.2, 0.3, 0.4, 0.1]
start_value = 1000000
risk_free_rate = np.sum(LIBOR_rate)/252
[portfolio_return,cumulative_return, average_return, std_dev_return,sharp_ratio,
mhist_volatility,end_value] =
assess_portfolio(date_range,symbols,portfolio_weights,start_value,risk_free_rate)
print("The cumulative return is : %.5f" % cumulative_return)
print("The average period return is : %.5f" % average_return)
print("The standard deviation of return is: %.5f" % std_dev_return)
print("The sharp ratio is: %.5f" % sharp_ratio)
volatility_frame = {'Date':dates[1:],'Moving Historical Volatility':mhist_volatility}
mhist_frame = pd.DataFrame(volatility_frame)
print("The mhist_volatility is \n", mhist_frame)
print("The ending value of the portfolio is %.5f" %end_value )
```

Appendix C. Python Code for Question (4)

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

#reading data from excel
df = pd.read_excel('/Users/joyce/Desktop/Python/Python/Currencies.xlsx')
#choose BTC and ETH cryptocurrencies.
#calculate their log returns
df['BTC_log_return'] = np.log(df['BTC'] / df['BTC'].shift(1))
df['ETH_log_return'] = np.log(df['ETH'] / df['ETH'].shift(1))
#calculate their daily returns
df['BTC_daily_return'] = (df['BTC'] - df['BTC'].shift(1)) / df['BTC'].shift(1)
df['ETH_daily_return'] = (df['ETH'] - df['ETH'].shift(1)) / df['ETH'].shift(1)
```

```
#get the portfolio return, assume equal weights
df['portfolio_return'] = df['BTC_log_return'] * 0.5 + df['ETH_log_return'] * 0.5

#specific market event
#daily return is lower than -15%, then triple their positions
def spf_market_event(currencies):
    currency = currencies.split('_')[0]
    returns = '%s_daily_return' % currency
    for i in range(len(df[currencies])):
        if df[returns][i] <= -0.15:
            df[currencies][i] *= (-3)

BTC_cols = []
ETH_cols = []
#momentuam strategy by using previous 1,7,14,30 days
for i in [1,7,14,30]:
    BTC_col = 'BTC_position_%s' % i
    ETH_col = 'ETH_position_%s' % i
    df[BTC_col] = np.sign(df['BTC_daily_return'].rolling(i).mean())
    df[ETH_col] = np.sign(df['ETH_daily_return'].rolling(i).mean())
    spf_market_event(BTC_col)
    spf_market_event(ETH_col) #results from market events
    BTC_cols.append(BTC_col)
    ETH_cols.append(ETH_col)

#to derive the absolute performance of the momentum strategy for the different momentum intervals(in
days)
sns.set()
strats = ['portfolio_return']
# calculate daily portfolio return based on postion
for col in BTC_cols:
    strat = 'strategy_%s' % col.split('_')[2]
    BTC_c = 'BTC_position_%s' % col.split('_')[2]
    ETH_c = 'ETH_position_%s' % col.split('_')[2]
    df[strat] = df[BTC_c].shift(1) * df['BTC_log_return'] * 0.5 + df[ETH_c].shift(1) *
df['ETH_log_return'] * 0.5
    strats.append(strat)
df[strats].dropna().cumsum().apply(np.exp).plot()
plt.show() # get the plot
```

Appendix D. Python Code for Question (5)

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import datetime

#reading data from excel
df = pd.read_excel('/Users/joyce/Desktop/Python/Python/Currencies.xlsx')
#choose BTC and ETH cryptocurrencies.
#calculate their log returns
```

```
df['BTC_log_return'] = np.log(df['BTC'] / df['BTC'].shift(1))
df['ETH_log_return'] = np.log(df['ETH'] / df['ETH'].shift(1))
#calculate their daily returns
df['BTC_daily_return'] = (df['BTC'] - df['BTC'].shift(1)) / df['BTC'].shift(1)
df['ETH_daily_return'] = (df['ETH'] - df['ETH'].shift(1)) / df['ETH'].shift(1)

#get the portfolio return, assume equal weights
df['portfolio_return'] = df['BTC_log_return'] * 0.5 + df['ETH_log_return'] * 0.5

# calculate daily standard deviation by 30 days.
df['BTC_std'] = df['BTC'].rolling(8).std()
df['ETH_std'] = df['ETH'].rolling(8).std()
df['BTC_std_change'] = (df['BTC_std'] - df['BTC_std'].shift(1)) / df['BTC_std'].shift(1)
df['ETH_std_change'] = (df['ETH_std'] - df['ETH_std'].shift(1)) / df['ETH_std'].shift(1)
print(df['BTC_std_change'])
#specific market event
#if standard deviation is larger than 1.25 then triple their positions
def spf_market_event(currencies):
    currency = currencies.split('_')[0]
    volatility_change = '%s_std_change' % currency
    events_time = []
    for momentum in range(len(df[currencies])):
        if df[volatility_change][momentum] >= 1.6:
            df[currencies][momentum] *= (-3)
            events_time.append(df['Date'][momentum])

    return events_time


BTC_cols = []
ETH_cols = []
events_time = []
#momentuam strategy by using previous 1,7,14,30 days
for i in [4,8,10,14]:
    BTC_col = 'BTC_position_%s' % i
    ETH_col = 'ETH_position_%s' % i
    df[BTC_col] = np.sign(df['BTC_daily_return'].rolling(i).mean())
    df[ETH_col] = np.sign(df['ETH_daily_return'].rolling(i).mean())
    events_time.append(spf_market_event(BTC_col))
    events_time.append(spf_market_event(ETH_col)) #results from market events
    BTC_cols.append(BTC_col)
    ETH_cols.append(ETH_col)
print(events_time)
#to derive the absolute performance of the momentum strategy for the different momentum intervals(in
days)
sns.set()
strats = ['portfolio_return']
# calculate daily portfolio return based on postion
for col in BTC_cols:
    strat = 'strategy_%s' % col.split('_')[2]
    BTC_c = 'BTC_position_%s' % col.split('_')[2]
    ETH_c = 'ETH_position_%s' % col.split('_')[2]
    df[strat] = df[BTC_c].shift(1) * df['BTC_log_return'] * 0.5 + df[ETH_c].shift(1) *
df['ETH_log_return'] * 0.5
```

```
    strats.append(strat)
df[strats].cumsum().plot()
plt.show() # get the plot

entry = '2017-04-26' # start date of enter trade
entry = datetime.datetime.strptime(entry, '%Y-%m-%d')
exits = entry + datetime.timedelta(days=1) # exit date, the next day of enter date

pd = df[np.logical_and(df['Date'] >= entry, df['Date'] <= exits)]

strategy = pd['BTC'] - pd['BTC'].shift(1) + pd['ETH'] - pd['ETH'].shift(1)
print("The every trade between %s and %s is:%.2f" % (entry,exits,(strategy[-1:])))
# get the frequency of market event
print("Opportunities are expected %i times per year." %
np.average([len(events_time[0]),len(events_time[1])]))
```