Fiona's blood and tears-v2

June 28, 2020

Author: Fiona Jiaqi Wu
Date: 06-28-2020
Paper: Is Momentum the Fatal Seductress in the Cryptocurrency Market?
Class: Corporate Finance - Professor Narayanan Jayaraman & Professor Andras Danis

Note: The following is the code illustration for formation period and holding period both equal to 3 months.

0.0.1 1. Data Cleaning

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[2]:
```
path = '/Users/Fiona/Desktop/Term paper/crypto-markets.xlsx'
df = pd.read_excel(path)
```

[3]:
```
df.head()
```

[3]:

	symbol	name	date	ranknow	open	high	low	close	volume	\
0	BTC	Bitcoin	2013-04-28	1	135.30	135.98	132.10	134.21	0	
1	BTC	Bitcoin	2013-04-29	1	134.44	147.49	134.00	144.54	0	
2	BTC	Bitcoin	2013-04-30	1	144.00	146.93	134.05	139.00	0	
3	BTC	Bitcoin	2013-05-01	1	139.00	139.89	107.72	116.99	0	
4	BTC	Bitcoin	2013-05-02	1	116.38	125.60	92.28	105.21	0	

	market
0	1488566728
1	1603768865
2	1542813125
3	1298954594
4	1168517495

[4]:
```
# there are 2071 cryptocurrencies
df.name.value_counts()
```

[4]:
```
Litecoin                        2042
Namecoin                        2042
```

```
Bitcoin                             2042
Peercoin                            2041
Novacoin                            2041
                                    …
STACS                                  3
Atlas Protocol                         2
BEAT                                   2
Blockchain Certified Data Token        2
BitNautic Token                        2
Name: name, Length: 2071, dtype: int64
```

[6]:
```
# symbols and names have different value counts. This shows some cryptos have the same symbol
df.symbol.value_counts()
```

[6]:
```
BITS     3189
PXC      2123
BTB      2049
BTM      2043
NMC      2042
         …
STACS       3
ATP         2
BCDT        2
BTNT        2
BEAT        2
Name: symbol, Length: 2005, dtype: int64
```

0.0.2 Get rid of rows which do not have volume records

[7]:
```
df1 = df.loc[(df['volume'] != 0)]
df1.shape
```

[7]:
```
(914477, 10)
```

0.0.3 choose period between 2015-09-01 and 2018-11-29 - 39 months of data

[8]:
```
df2 = df1.loc[(df['date'] >= '2015-09-01') & (df['date'] <= '2018-11-29')]
df2.symbol.value_counts()
```

[8]:
```
BITS    2307
BTM     1662
GCC     1656
FAIR    1494
CPC     1484
        …
```

```
ILC         3
BTNT        2
BCDT        2
BEAT        2
ATP         2
Name: symbol, Length: 1999, dtype: int64
```

0.0.4 Pick the ones with at least $1 million and daily volume at least 100

use 'name' as value_counts() because there are different coins use the same symbol

[10]:
```
df3 = df2.loc[(df['date'] == '2018-11-29') & (df['market'] >= 1000000) &␣
↪(df['volume'] >= 100)]
# print(df3['name'].value_counts())
```

[11]:
```
df4 = df2.loc[df2['name'].isin(df3['name'])]
# print(df4.shape)
# print(df4['name'].value_counts())
```

0.0.5 Keep only currencies with more than 90 days records

we narrowed it down to 750 cryptocurrencies left

[28]:
```
df5 = pd.DataFrame(df4['name'].value_counts() >= 90)
df5 = df5[df5['name'] == True]
df5 = df4.loc[df4['name'].isin(df5.index)]
# print('the shape of df5 is: ', df5.shape)
# print(df5.name.value_counts())
```

0.0.6 Exclude USDT (Thether) - a stablecoin tied to US dollars

[29]:
```
df5 = df5[df5['symbol'] != 'USDT']
print(df5.shape)
print(df5.name.value_counts())
```

```
(337078, 10)
Dotcoin            1186
ReddCoin           1186
FLO                1186
OKCash             1186
Namecoin           1186
                   ...
Cryptosolartech      94
Qubitica             93
Timicoin             92
Moneytoken           92
```

```
Ethersocial          90
Name: name, Length: 749, dtype: int64
```

0.0.7 2. Calculate Monthly Returns

[85]:
```
df5.tail()
```

[85]:

	symbol	name	date	ranknow	open	high	low
564278	XSD	SounDAC	2018-11-25	1290	0.069628	0.100426	0.060031
564279	XSD	SounDAC	2018-11-26	1290	0.086572	0.102605	0.063123
564280	XSD	SounDAC	2018-11-27	1290	0.065282	0.098405	0.064442
564281	XSD	SounDAC	2018-11-28	1290	0.081352	0.106668	0.081352
564282	XSD	SounDAC	2018-11-29	1290	0.091048	0.364850	0.074758

	close	volume	market
564278	0.086558	508	1262178
564279	0.065963	1499	961864
564280	0.081527	1088	1188817
564281	0.091134	835	1328897
564282	0.075003	535	1093675

[86]:
```
df6 = df5.drop(['symbol','ranknow','open','high','low','volume','market'], axis=1)
df6 = df6.set_index(['date'])
```

[87]:
```
df6 = df6.pivot_table(index='date', columns='name', values = 'close')
# df6.tail()
```

[33]:
```
df7 = df6.resample('M').ffill().pct_change().fillna(0)
```

[34]:
```
# save df7 on my desktop as monthly return and import it back to python
df8 = pd.read_csv('/Users/Fiona/Desktop/monthly_return.csv')
```

[35]:
```
df_returns = df8.drop(['date'], axis =1)
df_returns.tail()
```

[35]:

	0chain	0x	0xBitcoin	1World	2GIVE	4NEW	AI Doctor
34	0.000000	0.364329	-0.149514	0.130952	-0.224000	0.000000	0.054916
35	-0.534670	-0.284973	-0.295870	-0.366528	-0.135939	0.000000	-0.377192
36	-0.097753	-0.155488	0.004668	-0.104963	0.008047	-0.513911	-0.203914
37	0.765346	0.174734	-0.268176	0.220365	-0.256813	-0.320219	0.099099
38	-0.564908	-0.429983	-0.040125	-0.318219	-0.288519	-0.327870	-0.438575

	ALIS	ALQO	AMLT	…	bitUSD	carVertical	district0x
34	-0.244072	-0.503321	-0.033540	…	-0.077586	0.077096	0.097868
35	-0.092612	0.066267	0.060795	…	-0.009346	-0.478805	-0.301347
36	-0.215623	0.560011	-0.060925	…	-0.047170	-0.025333	-0.026910

```
37 -0.135306  0.392085 -0.377918  … -0.009901     0.120383    0.365508
38 -0.260614 -0.579813  1.061385  … -0.010994    -0.575092   -0.593121

    doc.com Token    eosDAC  iExec RLC  indaHash      nUSD  savedroid  \
34      -0.566039 -0.607357  -0.172928 -0.081040  0.000000   0.000000
35      -0.257709 -0.190806  -0.279431 -0.377053 -0.002169   0.000000
36      -0.426027 -0.223038  -0.065649 -0.217861  0.004035  -0.193868
37      -0.245350  0.304548   0.107927 -0.056570 -0.004615  -0.241611
38      -0.375513 -0.653097  -0.434102 -0.384325 -0.010762  -0.333333

     ugChain
34  0.189014
35 -0.547719
36 -0.335361
37 -0.234875
38 -0.596240

[5 rows x 749 columns]
```

[36]:
```
flattened_returns = df_returns.values.flatten('F')
flattened_returns
```

[36]:
```
array([ 0.        ,  0.        ,  0.        , …, -0.33536104,
       -0.2348746 , -0.59624049])
```

[37]:
```
Returns = pd.DataFrame(flattened_returns, columns = ['ret'])
Returns
```

[37]:
```
            ret
0      0.000000
1      0.000000
2      0.000000
3      0.000000
4      0.000000
…         …
29206  0.189014
29207 -0.547719
29208 -0.335361
29209 -0.234875
29210 -0.596240

[29211 rows x 1 columns]
```

[38]:
```
df8.set_index(['date'], inplace = True)
names = list(df8.columns)
```

[39]:
```
import itertools
```

[42]:
```
X = 39
```

[43]:
```
res = list(itertools.chain.from_iterable(itertools.repeat(i, X)
                                         for i in names))
```

[44]:
```
len(res)
```

[44]:
```
29211
```

[45]:
```
Names = pd.DataFrame(res, columns = ['Names'])
```

[46]:
```
df8 = df8.reset_index()
df9 = df8['date']
```

[47]:
```
Date = pd.concat([df9]*749, ignore_index=True)
```

[48]:
```
Date = pd.DataFrame(Date)
```

[49]:
```
data = pd.concat([Date,Names,Returns], axis = 1)
data.set_index(['date'], inplace = True)
data.columns = ['name','ret']
data
```

[49]:

date	name	ret
9/30/15	0chain	0.000000
10/31/15	0chain	0.000000
11/30/15	0chain	0.000000
12/31/15	0chain	0.000000
1/31/16	0chain	0.000000
…	…	…
7/31/18	ugChain	0.189014
8/31/18	ugChain	-0.547719
9/30/18	ugChain	-0.335361
10/31/18	ugChain	-0.234875
11/30/18	ugChain	-0.596240

[29211 rows x 2 columns]

[50]:
```
# data.loc[data['name'] == 'Bitcoin']
```

0.0.8 3. Calculate the rolling culmulative returns

[51]:
```
J = 3 # formation period length
K = 3 # holding period length
```

[52]:
```
data['logret'] = np.log(1 + data['ret'])
umd = data.groupby(['name'])['logret'].rolling(J, min_periods=J).sum()
umd = umd.reset_index()
umd['cumret']=np.exp(umd['logret'])-1
```

0.0.9 4. Formation of 10 Momentum Porfolios

[54]:
```
# For each date: assign ranking 1-10 based on cumret
# 1=lowest 10=highest cumret
umd.replace(0, np.nan, inplace = True )
umd = umd.dropna(axis=0, subset=['cumret'])
umd = umd.drop_duplicates()
```

[55]:
```
umd['momr']=umd.groupby('date')['cumret'].transform(lambda x: pd.qcut(x, 10,␣
↪labels=False))
```

[56]:
```
umd.momr=umd.momr.astype(int)
umd['momr'] = umd['momr']+1
```

[57]:
```
# change the date format using pd.to_datetime()
umd['date'] = pd.to_datetime(umd.date)
```

[58]:
```
umd.loc[umd['momr'] == 10]
```

[58]:

	name	date	logret	cumret	momr
67	0x	2018-01-31	2.367888	9.674822	10
71	0x	2018-05-31	0.333319	0.395593	10
72	0x	2018-06-30	0.369068	0.446385	10
73	0x	2018-07-31	-0.088553	-0.084746	10
112	0xBitcoin	2018-07-31	-0.161947	-0.149514	10
…	…	…	…	…	…
28851	bitUSD	2018-03-31	0.129068	0.137767	10
28855	bitUSD	2018-07-31	0.057708	0.059406	10
28856	bitUSD	2018-08-31	-0.009390	-0.009346	10
29009	eosDAC	2018-05-31	0.356011	0.427623	10
29129	nUSD	2018-08-31	-0.002171	-0.002169	10

[1077 rows x 5 columns]

[59]:
```
from pandas.tseries.offsets import MonthEnd
from pandas.tseries.offsets import MonthBegin
```

[60]:
```
umd['form_date'] = umd['date']
umd['medate'] = umd['date']+MonthEnd(0)
```

[61]:
```
umd['hdate1']=umd['medate']+MonthBegin(1)
umd['hdate2']=umd['medate']+MonthEnd(K)
umd = umd[['name', 'form_date','momr','hdate1','hdate2']]
```

[62]:
```
umd
```

[62]:

	name	form_date	momr	hdate1	hdate2
35	0chain	2018-08-31	7	2018-09-01	2018-11-30
36	0chain	2018-09-30	3	2018-10-01	2018-12-31
37	0chain	2018-10-31	7	2018-11-01	2019-01-31
38	0chain	2018-11-30	8	2018-12-01	2019-02-28
63	0x	2017-09-30	3	2017-10-01	2017-12-31
…	…	…	…	…	…
29206	ugChain	2018-07-31	8	2018-08-01	2018-10-31
29207	ugChain	2018-08-31	5	2018-09-01	2018-11-30
29208	ugChain	2018-09-30	2	2018-10-01	2018-12-31
29209	ugChain	2018-10-31	1	2018-11-01	2019-01-31
29210	ugChain	2018-11-30	1	2018-12-01	2019-02-28

[10644 rows x 5 columns]

[63]:
```
# join rank and return together
data.reset_index(inplace = True)
data = data[['name','date','ret']]
```

[64]:
```
port = pd.merge(data, umd, on =['name'], how = 'inner')
port.replace(0, np.nan, inplace = True)
```

[65]:
```
port['date'] = pd.to_datetime(port.date)
```

[66]:
```
# the date has to be between hdate1 and hdate2
port = port[(port['hdate1']<=port['date']) & (port['date']<=port['hdate2'])]
```

[67]:
```
umd2 = port.sort_values(by=['date','momr','form_date','name']).drop_duplicates()
umd2.shape
```

[67]: (27438, 7)

[68]:
```
umd3 = umd2.groupby(['date','momr','form_date'])['ret'].mean().reset_index()
umd3.head(20)
```

[68]:

	date	momr	form_date	ret
0	2015-12-31	1	2015-11-30	0.363597
1	2015-12-31	2	2015-11-30	-0.002330
2	2015-12-31	3	2015-11-30	0.098868
3	2015-12-31	4	2015-11-30	0.190355
4	2015-12-31	5	2015-11-30	0.543714

```
5  2015-12-31     6 2015-11-30  0.080233
6  2015-12-31     7 2015-11-30  0.063129
7  2015-12-31     8 2015-11-30  0.000391
8  2015-12-31     9 2015-11-30 -0.134231
9  2015-12-31    10 2015-11-30  0.026343
10 2016-01-31     1 2015-11-30  0.051819
11 2016-01-31     1 2015-12-31  0.645427
12 2016-01-31     2 2015-11-30  0.564772
13 2016-01-31     2 2015-12-31  0.237809
14 2016-01-31     3 2015-11-30  0.475278
15 2016-01-31     3 2015-12-31  0.196431
16 2016-01-31     4 2015-11-30 -0.011661
17 2016-01-31     4 2015-12-31  1.810724
18 2016-01-31     5 2015-11-30  1.671957
19 2016-01-31     5 2015-12-31  0.229588
```

[69]:
```
umd3 = umd3.sort_values(by=['date','momr'])
umd3.shape
```

[69]:
```
(1050, 4)
```

[73]:
```
# Create one return group per momentum group
fionaret = umd3.groupby(['date','momr'])['ret'].mean().reset_index()
fionastd = umd3.groupby(['date','momr'])['ret'].std().reset_index()
fionaret = fionaret.rename(columns = {'ret': 'fionaret'})
fionastd = fionastd.rename(columns = {'ret': 'fionastd'})
fionaretdata = pd.merge(fionaret, fionastd, on=['date','momr'], how='inner')
fionaretdata = fionaretdata.sort_values(by=['momr'])
# fionaretdata = fionaretdata.dropna(axis = 0)
# fionaretdata.head(20)
```

[76]:
```
fionaretdata.groupby(['momr'])['fionaret'].describe()[['count','mean', 'std']]
```

[76]:
```
       count      mean       std
momr
1       36.0  0.812538  1.688072
2       36.0  0.360346  0.685876
3       36.0  0.287231  0.653371
4       36.0  0.304946  0.688136
5       36.0  0.492138  1.079526
6       36.0  0.429907  0.971345
7       36.0  0.373626  0.854265
8       36.0  0.419659  0.889817
9       36.0  0.604897  1.631351
10      36.0  0.708243  1.668174
```

[77]:
```
fretdata2 = fionaretdata.pivot(index='date', columns='momr', values='fionaret')

# Add prefix port in front of each column
fretdata2 = fretdata2.add_prefix('port')
fretdata2 = fretdata2.rename(columns={'port1':'losers', 'port10':'winners'})
fretdata2['long_short'] = fretdata2['winners'] - fretdata2['losers']

# Compute Long-Short Portfolio Cumulative Returns
fretdata3 = fretdata2
fretdata3['1+losers']=1+fretdata3['losers']
fretdata3['1+winners']=1+fretdata3['winners']
fretdata3['1+ls'] = 1+fretdata3['long_short']

fretdata3['cumret_winners']=fretdata3['1+winners'].cumprod()-1
fretdata3['cumret_losers']=fretdata3['1+losers'].cumprod()-1
fretdata3['cumret_long_short']=fretdata3['1+ls'].cumprod()-1
```

[78]:
```
from scipy import stats
```

0.0.10 5. Portfolio Summary Statistics

[79]:
```
# Mean
mom_mean = fretdata3[['winners', 'losers', 'long_short']].mean().to_frame()
mom_mean = mom_mean.rename(columns={0:'mean'}).reset_index()

# T-Value and P-Value
t_losers = pd.Series(stats.ttest_1samp(fretdata3['losers'],0.0)).to_frame().T
t_winners = pd.Series(stats.ttest_1samp(fretdata3['winners'],0.0)).to_frame().T
t_long_short = pd.Series(stats.ttest_1samp(fretdata3['long_short'],0.0)).
 ↪to_frame().T

t_losers['momr']='losers'
t_winners['momr']='winners'
t_long_short['momr']='long_short'

t_output =pd.concat([t_winners, t_losers, t_long_short])\
    .rename(columns={0:'t-stat', 1:'p-value'})

# Combine mean, t and p
mom_output = pd.merge(mom_mean, t_output, on=['momr'], how='inner')
```

[80]:
```
mom_output
```

[80]:

	momr	mean	t-stat	p-value
0	winners	0.708243	2.547372	0.015404
1	losers	0.812538	2.888047	0.006607
2	long_short	-0.104295	-0.654167	0.517280

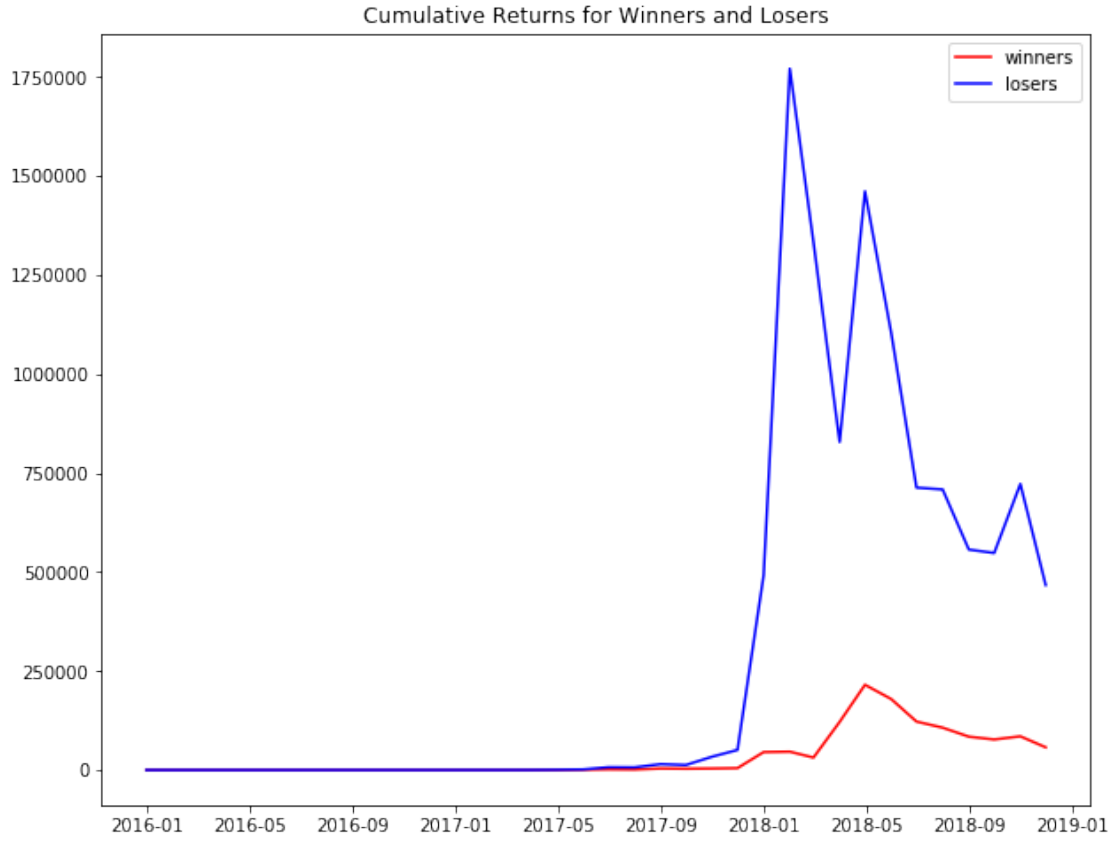
0.0.11 6. Visualization

[81]:
```
%matplotlib inline
```

[82]:
```
plt.figure(figsize = (10, 8))
plt.plot(fretdata3.index, fretdata3['cumret_winners'], 'r-', label = 'winners')
plt.plot(fretdata3.index, fretdata3['cumret_losers'], 'b-',label = 'losers')
plt.title("Cumulative Returns for Winners and Losers")
plt.legend()
plt.show()
```

```
/Users/Fiona/opt/anaconda3/lib/python3.7/site-
packages/pandas/plotting/_matplotlib/converter.py:103: FutureWarning: Using an
implicitly registered datetime converter for a matplotlib plotting method. The
converter was registered by pandas on import. Future versions of pandas will
require you to explicitly register matplotlib converters.

To register the converters:
        >>> from pandas.plotting import register_matplotlib_converters
        >>> register_matplotlib_converters()
  warnings.warn(msg, FutureWarning)
```



[83]:
```
plt.figure(figsize = (10, 8))
plt.ylim(-5,2)
plt.plot(fretdata3.index, fretdata3['cumret_long_short'], 'b-', label␣
↪='long-short')
plt.title("Long/Short Strategy Cumulative Returns")
plt.legend()
plt.show()
```