Problem2

Pej

November 29, 2018

1 Introduction and Outlines

Implement one Smart Beta strategy and discuss pros and cons compared to a chosen benchmark.

Disclaimer: I do not have background in portfolio management/construction. The following is the result of reading the Wiki pages on CAPM and Markowitz's Modern Portfolio Theory. Also, it is under base assumption of no mistake in the codes !

So with this being said, the following is an attempt to construct a dynamic portfolio from some random holdings of SPY. Dynamics means the diversification is change at some fixed periods. For example, every month ('M') a certain calculation is done and based on those the diversification (how the capital is distributed on the assets)is modified.

Main Assumption: In the diversification it is assumed that there is no cap on the capitals. That is any amount of shares/stocks can be bought and also liquidity is possible at the observation times (Monthly for example).

Strategy Outline :

- calculate a β_i associate to the i-th asset rate r_i, using a market index. In our case we used SPY as the benchmark. β's are calculated at the end of an observation period ($alloc_period$). The calculation is based on the covariances and the volatility of SPY over a predetermined period (lookBackDays).
- based on the $\{\beta_i\}$ decide to sell and/or buy new stocks (from the pre-considered set of tickers). In this step the position (weight) w_i is digital, that is you either buy the i-th asset or sell completely.

$$\beta_i = \frac{cov(r_i, r_{SPY})}{vol(SPY)} \tag{1}$$

- apply the weights to the portfolio for the next period.

In the above the decision of the portfolio manager on how to define the $w_i = W_i(\{\beta_j\})$ is left as a quantile parameter that defines a cuttoff on what *beta* is accepted and what is not.

2 Packages

```
In [109]: import pandas as pd
          import datetime
          import matplotlib.pyplot as plt
          import numpy as np
          import numpy.linalg as lg
```

3 Data and Source

The data are gathered from yahoo finance using their API tools to fetch the database through python.

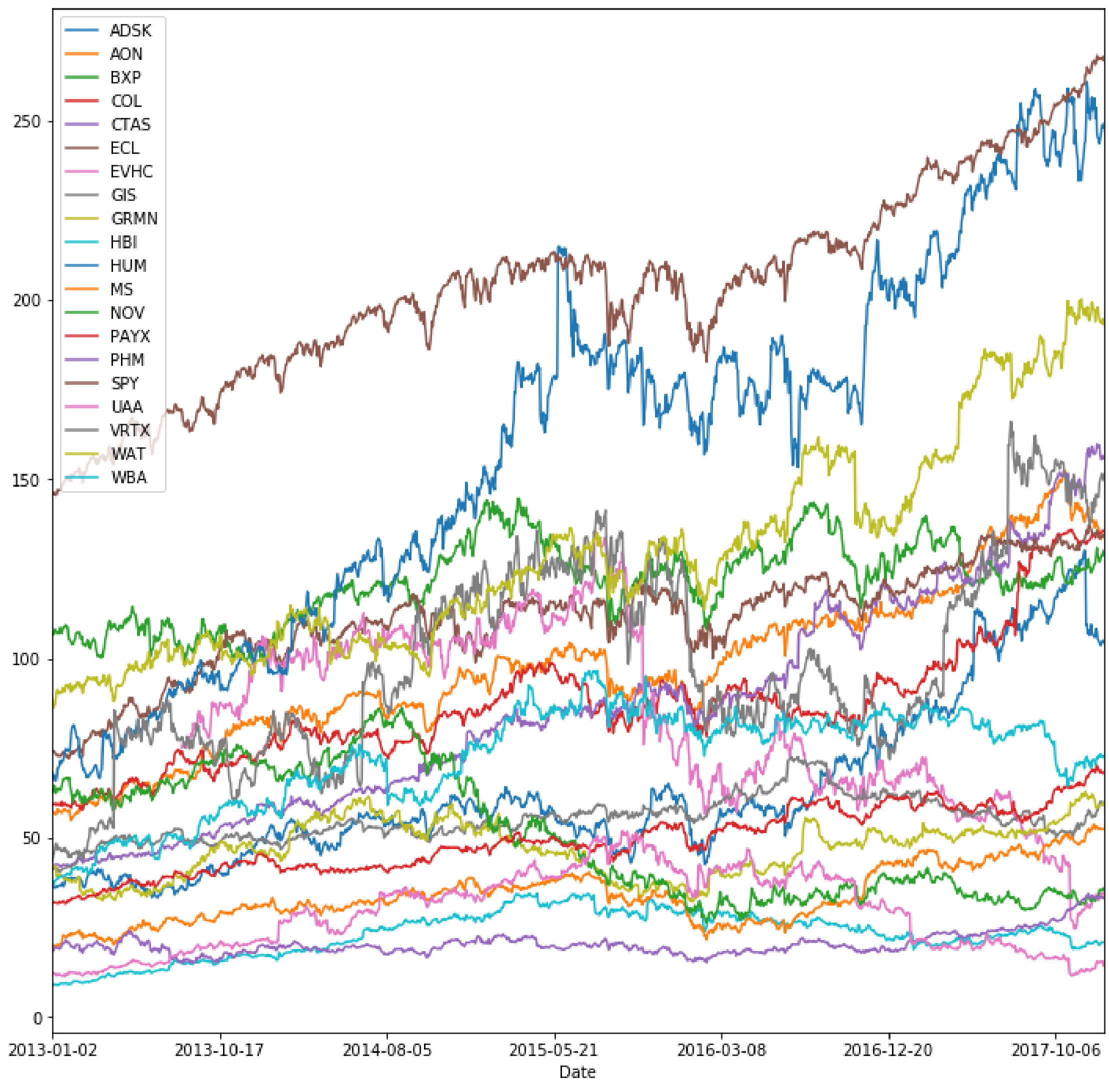
```
In [110]: ### loading data from file
          closes  = pd.read_csv('./SnP_holdings_random_pick_20.csv', index_col=0 )
          ### We are going to use the overnight rate (unit of time scale = 1 day)
          closes_return = closes.pct_change().dropna()
          closes_return = closes_return.reset_index()
          closes_return['Date'] = pd.to_datetime(closes_return['Date'])
          closes_return = closes_return.set_index('Date')
```

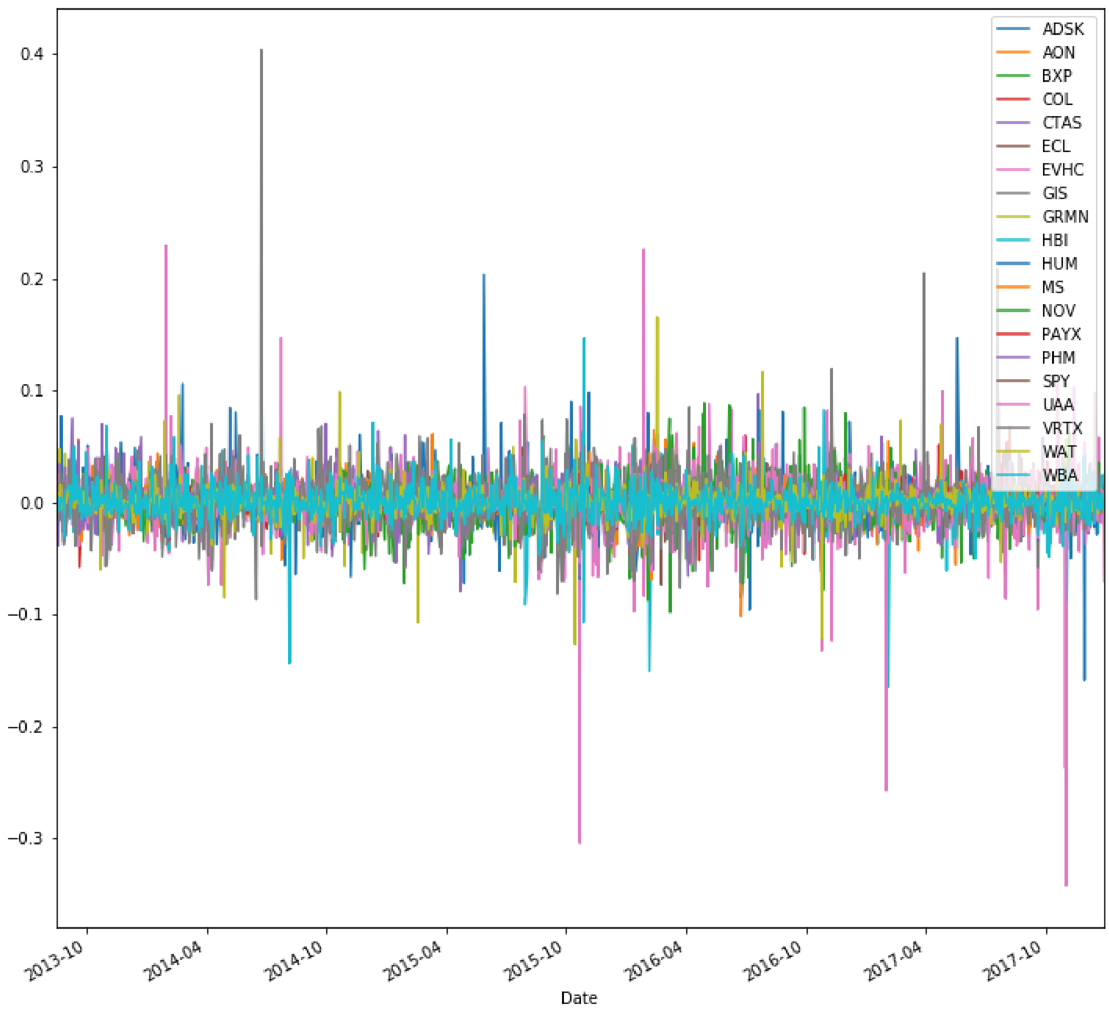
4 Parameters

```
In [111]: ##### the strategy:
          ##### is based on reallocating capital over a range of
          ##### assets on  'alloc_period' based.
          ##### the period 'alloc_period' can be modified bellow.
          alloc_period='M'

          ##### the vol and covariance of the assets are obtain
          ##### from historical data by looking back from the
          ##### day of allocation back till 'lookBackDays' number of days
          ##### 'lookBackDays' is defined and can be modified
          lookBackDays=70

          ##### the allocation is based on comparing the beta of each asset
          ##### in the basket and comparing it to
          ##### the a cutoff  set at the beginning.
          ##### The cutoff is defined by a 'quantile_'.
          quantile_ =0.5  # != 1.0 or there is no allocation

          ##### number of asset in the basket = number of columns in the
          ##### data - 1, since one column is the index itself
          num_assets = closes.shape[1]-1 ## -1 because one col is the spy (index)
```

5 Illustration

```
In [112]: ### Illustration of the closes
          closes.plot(figsize=[12,12])
          closes_return.plot(figsize=[12,12])
          plt.show()
```





6 The market index is SPY

```
In [113]: ########### variance of of spy ( the index) this will be used as the numeraire to obta
          spy_vol_3M1M = closes_return['SPY'].rolling(lookBackDays).std().resample(alloc_period)
```

7 Covariances

```
In [114]: ### cov matrixes we create a set of historical covariance matrices from
          ### historical data of the period [today, today-lookBackDays]
          cov_mtx_time_series = closes_return.rolling(lookBackDays).cov().dropna()
          idx_start_date = np.where(closes_return.index == cov_mtx_time_series.index[0][0])[0][0
          ### total number of days with a proper cov matrix
          total_dates_in_series=(closes_return.shape[0] -idx_start_date)
          ### the row of the first date in the spy (to be used later)
          spy_index= np.where(cov_mtx_time_series[0:1].columns=='SPY')[0][0]
```

8 $\{\beta_i\}$

In [115]:
```
#### constructing the beta series
betas = np.zeros([total_dates_in_series, num_assets])

for i in range(total_dates_in_series):
    cov = cov_mtx_time_series[(i*(num_assets+1)):((i+1)*(num_assets+1))]
    betas[i, :] = list(cov.values[spy_index, :spy_index]) + list(cov.values[spy_index,

cols=list(cov.columns[:spy_index])+list(cov.columns[(spy_index+1):])

betas = pd.DataFrame(betas,
                     index = closes_return.index[idx_start_date:],
                     columns=cols)

betas = betas.resample(alloc_period).mean()
### adjust for variance of market index (SPY in this case)
for r in range(betas.shape[0]):
    betas.iloc[r]= betas.iloc[r].apply(lambda x: x/spy_vol_3M1M.iloc[r])
```

9 Weights and Reallocation of Caps Based on β's

In [116]:
```
###  Digital Strategy
### we assign the weghts in the following way (a bit silly though !)
### in each allocation period the assets with a beta higher than cutoff
### of that period receives a w=1 and otherwise w=0
### (notice this in practice means you need to sell all the position on one asset==> h
### ==> liquidity issue


weights = np.zeros(betas.shape, dtype=int)
for r in range(betas.shape[0]):
    cutoff  = betas.iloc[r].quantile(quantile_)
    weights[r, :]  = betas.iloc[r].apply(lambda x: (0,1)[x>cutoff])

weights_df = pd.DataFrame(weights, index= betas.index, columns=cols)
```

10 Returns

In [117]:
```
########## application to the returns
### lets exclude the SPY from returns
portfolio_return = np.zeros([betas.shape[0]-1, 1])
closes_return_assets = closes_return.drop(['SPY'], axis=1)
for alloc_idx in range(len(betas.index)-1):
    start_date = betas.index[alloc_idx]
    end_date   = betas.index[alloc_idx+1]
    period_returns =closes_return_assets.loc[start_date:end_date]
```

```
        ### This is the aggregate return daily
        total_per_date_in_period_return = period_returns.mul(weights[alloc_idx, :], ).sum(
        # full return in alloc_period -  aggregate return Monthly
        portfolio_return[alloc_idx,0]=(total_per_date_in_period_return + 1.0).prod() - 1

    col_name = 'portfolio %s '%(alloc_period) + 'Return'
    portfolio_return=pd.DataFrame(portfolio_return, index=betas.index[1:], columns=[col_na
```

11 Illustration of returns

```
In [118]: #####  The index return over same period:

          market_index = closes_return['SPY']
          return_per_period_idx=np.zeros([betas.shape[0]-1, 1])
          for alloc_idx in range(len(betas.index)-1):
              start_date = betas.index[alloc_idx]
              end_date   = betas.index[alloc_idx+1]
              period_returns_idx = market_index.loc[start_date:end_date]
              return_per_period_idx[alloc_idx, 0] = (period_returns_idx+1).prod()-1


          col_name = 'SPY %s '%(alloc_period) + 'Return'
          return_per_period_idx=  pd.DataFrame(return_per_period_idx, index=betas.index[1:], col
```

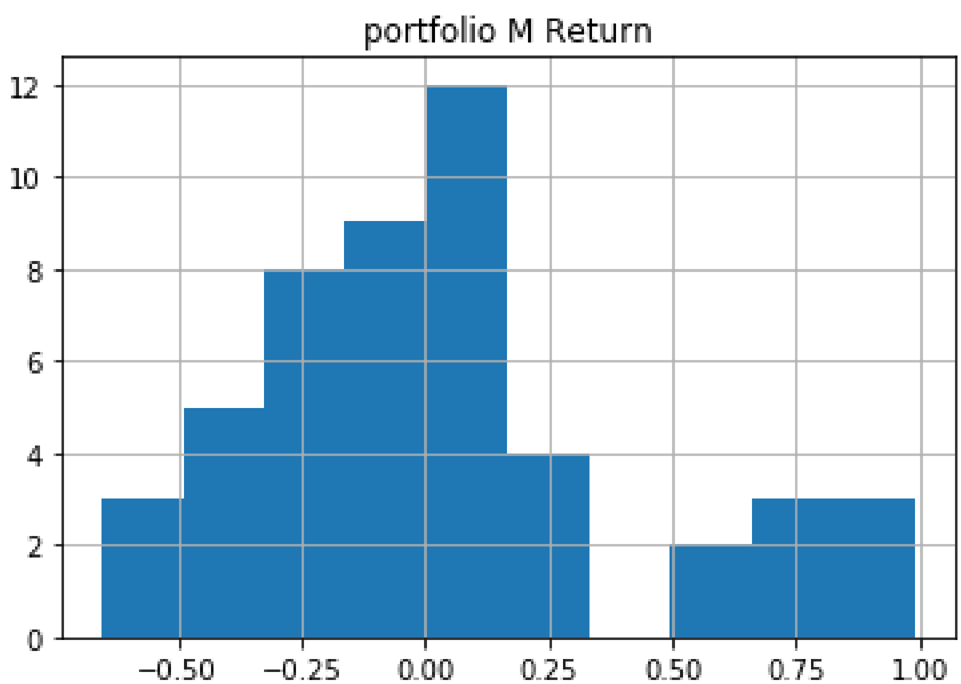
12 Overview and Final Results

```
In [123]: ###### Final result
          print '==========================================================='
          print 'alloc_period : ', alloc_period
          print 'hist data used from %d days prior : '%lookBackDays
          print 'allocation to quantiles > %f : '%quantile_
          print '=========================================================\n\n'


          #print 'Total return over the horizon of the strategy %:', ((portfolio_return+1.).prod
          print 'Total vol of the portfolio %' ,  portfolio_return.std()[0] *100
          if alloc_period=='M':
              print 'Annual mean return over the horizon of the strategy %:', portfolio_return.m

          print '========================================================= \n\n'

          print '========================================================='
          #print 'Total return of SPY (over the full years) %:', (((return_per_period_idx+1).pro
          print 'Total vol of the Index %' ,  return_per_period_idx.std()[0] *100
          if alloc_period=='M':
              print 'Annual mean return over the horizon of the Index %:', return_per_period_idx
```

```
print '============================================================'
```

```
============================================================
alloc_period :  M
hist data used from 70 days prior :
allocation to quantiles > 0.500000 :
============================================================


Total vol of the portfolio % 40.1092223792
Annual mean return over the horizon of the strategy %: 46.9854895287
============================================================


============================================================
Total vol of the Index % 3.00572856573
Annual mean return over the horizon of the Index %: 9.67134343784
============================================================
```

In [182]:
```
portfolio_return.hist()
plt.show()
```

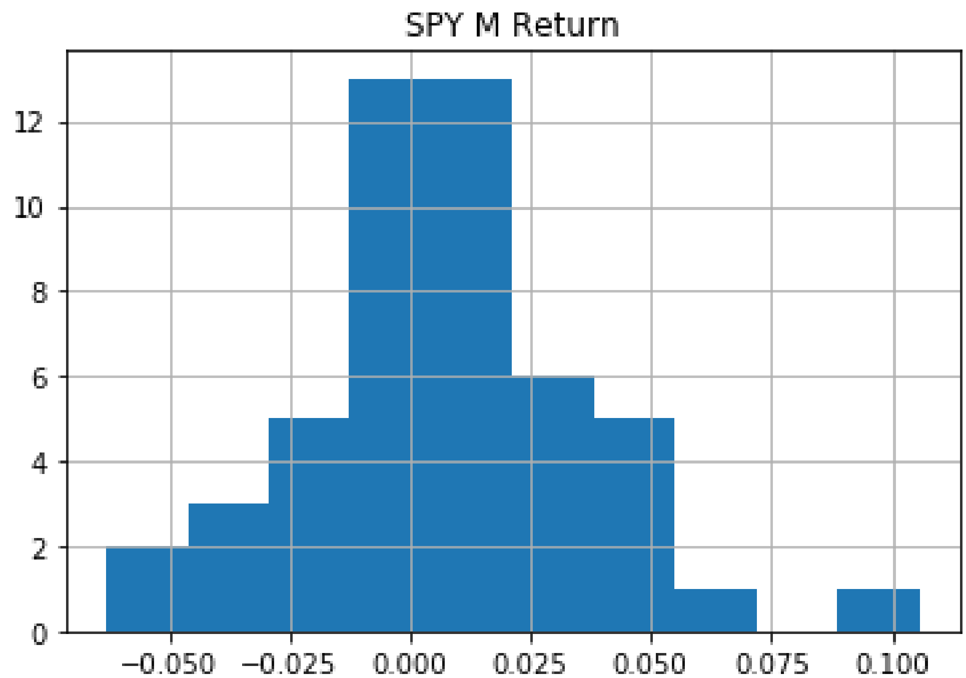


In [181]:
```
portfolio_return *100
```

```
Out[181]:             portfolio M Return
          Date
          2013-12-31           30.850111
          2014-01-31           10.407848
          2014-02-28           29.051271
          2014-03-31          -35.761914
          2014-04-30          -28.586576
          2014-05-31           53.548087
          2014-06-30           91.111906
          2014-07-31          -36.694971
          2014-08-31           -7.686768
          2014-09-30           -1.340038
          2014-10-31           14.610343
          2014-11-30           69.339104
          2014-12-31          -12.881044
          2015-01-31          -46.641394
          2015-02-28           98.763274
          2015-03-31           -9.712619
          2015-04-30          -27.257293
          2015-05-31            6.203842
          2015-06-30          -28.002931
          2015-07-31           10.540944
          2015-08-31          -64.762432
          2015-09-30          -54.477730
          2015-10-31           86.692670
          2015-11-30           -1.786692
          2015-12-31          -42.552455
          2016-01-31          -65.365010
          2016-02-29          -16.820930
          2016-03-31           10.975783
          2016-04-30           50.546005
          2016-05-31          -17.291260
          2016-06-30          -32.369408
          2016-07-31           80.944140
          2016-08-31           15.752938
          2016-09-30          -20.652282
          2016-10-31          -38.888551
          2016-11-30           79.972333
          2016-12-31          -25.665009
          2017-01-31           17.197721
          2017-02-28            9.452241
          2017-03-31            8.440916
          2017-04-30           -4.622728
          2017-05-31           11.889035
          2017-06-30           11.786402
          2017-07-31           22.401877
          2017-08-31           -6.678591
          2017-09-30            1.223639
```

```
2017-10-31            0.013816
2017-11-30          -14.582586
2017-12-31           11.222381
```

In [183]:
```
return_per_period_idx.hist()
plt.show()
```



In [177]:
```
return_per_period_idx *100
```

Out[177]:

Date	SPY M Return
2013-12-31	2.038675
2014-01-31	-3.068226
2014-02-28	3.939071
2014-03-31	0.640398
2014-04-30	1.520294
2014-05-31	2.625828
2014-06-30	1.577750
2014-07-31	-1.394143
2014-08-31	1.893599
2014-09-30	-1.838475
2014-10-31	2.085659
2014-11-30	3.922155
2014-12-31	-0.801160
2015-01-31	-3.925823

2015-02-28	5.620460
2015-03-31	-2.007980
2015-04-30	0.100844
2015-05-31	0.270690
2015-06-30	-2.486502
2015-07-31	2.423710
2015-08-31	-6.299219
2015-09-30	-3.839595
2015-10-31	10.554015
2015-11-30	0.365512
2015-12-31	-2.715214
2016-01-31	-5.929196
2016-02-29	-0.082595
2016-03-31	5.346255
2016-04-30	0.150470
2016-05-31	1.701155
2016-06-30	-0.361496
2016-07-31	5.061449
2016-08-31	0.119754
2016-09-30	-0.779815
2016-10-31	-0.992170
2016-11-30	3.688723
2016-12-31	1.186001
2017-01-31	1.789469
2017-02-28	3.920017
2017-03-31	-0.577789
2017-04-30	0.757548
2017-05-31	1.411290
2017-06-30	0.124225
2017-07-31	2.245700
2017-08-31	0.234904
2017-09-30	2.121865
2017-10-31	2.356406
2017-11-30	3.217141
2017-12-31	1.579686