03_pyfolio_demo

September 29, 2021

1 From zipline to pyfolio

Pyfolio facilitates the analysis of portfolio performance and risk in-sample and out-of-sample using many standard metrics. It produces tear sheets covering the analysis of returns, positions, and transactions, as well as event risk during periods of market stress using several built-in scenarios, and also includes Bayesian out-of-sample performance analysis.

- Open-source backtester by Quantopian Inc.
- Powers Quantopian.com
- State-of-the-art portfolio and risk analytics
- Various models for transaction costs and slippage.
- Open source and free: Apache v2 license
- Can be used:
  - stand alone
  - with Zipline
  - on Quantopian

1.1 Imports & Settings

[1]:
```
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
%matplotlib inline
from pathlib import Path

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from pyfolio.utils import extract_rets_pos_txn_from_zipline
from pyfolio.plotting import (plot_perf_stats,
                              show_perf_stats,
                              plot_rolling_beta,
                              plot_rolling_returns,
                              plot_rolling_sharpe,
                              plot_drawdown_periods,
                              plot_drawdown_underwater)
```

```
from pyfolio.timeseries import perf_stats, extract_interesting_date_ranges
```

[3]:
```
sns.set_style('whitegrid')
```

1.2 Converting data from zipline to pyfolio

[4]:
```
with pd.HDFStore('backtests.h5') as store:
    backtest = store['backtest/equal_weight']
backtest.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1008 entries, 2013-01-02 00:00:00+00:00 to 2016-12-30
00:00:00+00:00
Data columns (total 39 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   period_open              1008 non-null   datetime64[ns, UTC]
 1   period_close             1008 non-null   datetime64[ns, UTC]
 2   starting_cash            1008 non-null   float64
 3   ending_cash              1008 non-null   float64
 4   portfolio_value          1008 non-null   float64
 5   returns                  1008 non-null   float64
 6   longs_count              1008 non-null   int64
 7   shorts_count             1008 non-null   int64
 8   long_value               1008 non-null   float64
 9   short_value              1008 non-null   float64
 10  long_exposure            1008 non-null   float64
 11  pnl                      1008 non-null   float64
 12  short_exposure           1008 non-null   float64
 13  capital_used             1008 non-null   float64
 14  orders                   1008 non-null   object
 15  transactions             1008 non-null   object
 16  gross_leverage           1008 non-null   float64
 17  positions                1008 non-null   object
 18  net_leverage             1008 non-null   float64
 19  starting_exposure        1008 non-null   float64
 20  ending_exposure          1008 non-null   float64
 21  starting_value           1008 non-null   float64
 22  ending_value             1008 non-null   float64
 23  factor_data              1008 non-null   object
 24  prices                   1008 non-null   object
 25  treasury_period_return   1008 non-null   float64
 26  trading_days             1008 non-null   int64
 27  period_label             1008 non-null   object
 28  algorithm_period_return  1008 non-null   float64
 29  algo_volatility          1007 non-null   float64
 30  benchmark_period_return  1008 non-null   float64
```

```
 31   benchmark_volatility     1007 non-null   float64
 32   alpha                    0 non-null      object
 33   beta                     0 non-null      object
 34   sharpe                   1004 non-null   float64
 35   sortino                  1004 non-null   float64
 36   max_drawdown             1008 non-null   float64
 37   max_leverage             1008 non-null   float64
 38   excess_return            1008 non-null   float64
dtypes: datetime64[ns, UTC](2), float64(26), int64(3), object(8)
memory usage: 315.0+ KB
```

pyfolio relies on portfolio returns and position data, and can also take into account the transaction costs and slippage losses of trading activity. The metrics are computed using the empyrical library that can also be used on a standalone basis. The performance DataFrame produced by the zipline backtesting engine can be translated into the requisite pyfolio input.

[5]:
```
returns, positions, transactions = extract_rets_pos_txn_from_zipline(backtest)
```

[6]:
```
returns.head().append(returns.tail())
```

[6]:
```
2013-01-02 00:00:00+00:00    0.000000
2013-01-03 00:00:00+00:00    0.000000
2013-01-04 00:00:00+00:00    0.000000
2013-01-07 00:00:00+00:00    0.000000
2013-01-08 00:00:00+00:00   -0.000005
2016-12-23 00:00:00+00:00   -0.000233
2016-12-27 00:00:00+00:00    0.000160
2016-12-28 00:00:00+00:00   -0.000847
2016-12-29 00:00:00+00:00    0.000735
2016-12-30 00:00:00+00:00   -0.000606
Name: returns, dtype: float64
```

[7]:
```
positions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1004 entries, 2013-01-08 00:00:00+00:00 to 2016-12-30
00:00:00+00:00
Columns: 750 entries, Equity(0 [A]) to cash
dtypes: float64(750)
memory usage: 5.8 MB
```

[8]:
```
positions.columns = [c for c in positions.columns[:-1]] + ['cash']
positions.index = positions.index.normalize()
positions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1004 entries, 2013-01-08 00:00:00+00:00 to 2016-12-30
00:00:00+00:00
Columns: 750 entries, Equity(0 [A]) to cash
```

```
dtypes: float64(750)
memory usage: 5.8 MB
```

[9]:
```python
transactions.symbol = transactions.symbol.apply(lambda x: x.symbol)
```

[10]:
```python
transactions.head().append(transactions.tail())
```

[10]:
```
                                         sid symbol       price  \
2013-01-08 21:00:00+00:00     Equity(85 [AGN])    AGN   86.680005
2013-01-08 21:00:00+00:00   Equity(213 [ARIA])   ARIA   19.651001
2013-01-08 21:00:00+00:00   Equity(367 [BIIB])   BIIB  144.390001
2013-01-08 21:00:00+00:00    Equity(811 [DFS])    DFS   40.090001
2013-01-08 21:00:00+00:00   Equity(1059 [FDO])    FDO   57.320002
2016-12-29 21:00:00+00:00     Equity(66 [AEO])    AEO   15.270000
2016-12-29 21:00:00+00:00    Equity(833 [DKS])    DKS   52.380000
2016-12-29 21:00:00+00:00   Equity(1757 [MAT])    MAT   27.630000
2016-12-30 21:00:00+00:00  Equity(2181 [PDCO])   PDCO   41.030000
2016-12-30 21:00:00+00:00  Equity(2953 [URBN])   URBN   28.480000

                                                   order_id  amount  \
2013-01-08 21:00:00+00:00  eb7dbd283656403ca8c9d6c1188dc727    2334
2013-01-08 21:00:00+00:00  682c967689d540748db436fd51d5163a    7590
2013-01-08 21:00:00+00:00  a428261b64d4412db94a4573faa1cc04    1365
2013-01-08 21:00:00+00:00  956824d7a6064717b7aef1594daa65dc    5073
2013-01-08 21:00:00+00:00  863bd8a8917b476390c04d8924fa923d    3496
2016-12-29 21:00:00+00:00  ba96247cd5ee435da2c104312b81c7f7    3461
2016-12-29 21:00:00+00:00  027a5f84d39549e5ad05470eb14ed268    1063
2016-12-29 21:00:00+00:00  638e5a7fee514e4091b637db8da6903c    1183
2016-12-30 21:00:00+00:00  5fdc0d539a0f4ccd8554dc6db26f36b2    -119
2016-12-30 21:00:00+00:00  d49b7ce165dd4daaa515da54b6f2c497   -1006

                          commission                        dt    txn_dollars
2013-01-08 21:00:00+00:00       None 2013-01-08 21:00:00+00:00 -202311.131306
2013-01-08 21:00:00+00:00       None 2013-01-08 21:00:00+00:00 -149151.099321
2013-01-08 21:00:00+00:00       None 2013-01-08 21:00:00+00:00 -197092.351185
2013-01-08 21:00:00+00:00       None 2013-01-08 21:00:00+00:00 -203376.572741
2013-01-08 21:00:00+00:00       None 2013-01-08 21:00:00+00:00 -200390.728571
2016-12-29 21:00:00+00:00       None 2016-12-29 21:00:00+00:00  -52849.470534
2016-12-29 21:00:00+00:00       None 2016-12-29 21:00:00+00:00  -55679.940343
2016-12-29 21:00:00+00:00       None 2016-12-29 21:00:00+00:00  -32686.290035
2016-12-30 21:00:00+00:00       None 2016-12-30 21:00:00+00:00    4882.569999
2016-12-30 21:00:00+00:00       None 2016-12-30 21:00:00+00:00   28650.879685
```

[11]:
```python
HDF_PATH = Path('..', 'data', 'assets.h5')
```

1.2.1 Sector Map

[12]:
```
assets = positions.columns[:-1]
with pd.HDFStore(HDF_PATH) as store:
    df = store.get('us_equities/stocks')['sector'].dropna()
    df = df[~df.index.duplicated()]
sector_map = df.reindex(assets).fillna('Unknown').to_dict()
```

1.2.2 Benchmark

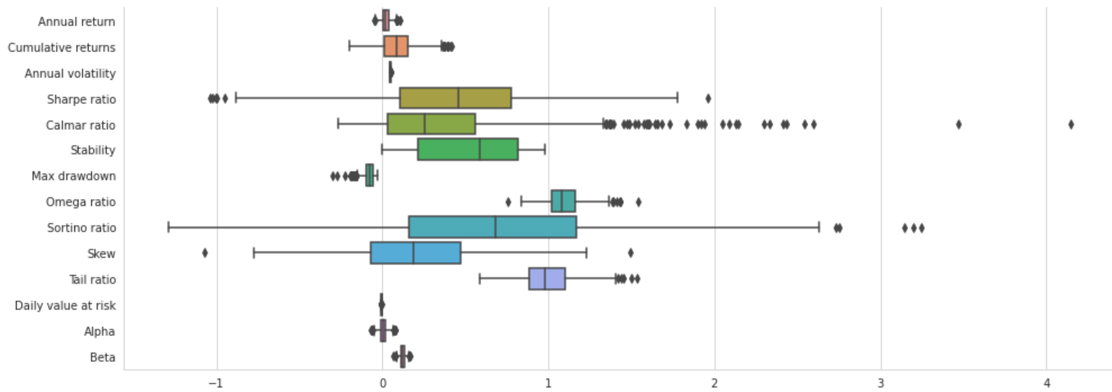
[13]:
```
with pd.HDFStore(HDF_PATH) as store:
    benchmark_rets = store['sp500/fred'].close.pct_change()
benchmark_rets.name = 'S&P500'
benchmark_rets = benchmark_rets.tz_localize('UTC').filter(returns.index)
benchmark_rets.tail()
```

[13]:
```
DATE
2016-12-23 00:00:00+00:00    0.001252
2016-12-27 00:00:00+00:00    0.002248
2016-12-28 00:00:00+00:00   -0.008357
2016-12-29 00:00:00+00:00   -0.000293
2016-12-30 00:00:00+00:00   -0.004637
Name: S&P500, dtype: float64
```

[14]:
```
perf_stats(returns=returns,
           factor_returns=benchmark_rets)
#            positions=positions,
#            transactions=transactions)
```

[14]:
```
Annual return          0.019619
Cumulative returns     0.080817
Annual volatility      0.047487
Sharpe ratio           0.432879
Calmar ratio           0.336024
Stability              0.555919
Max drawdown          -0.058387
Omega ratio            1.085094
Sortino ratio          0.630497
Skew                   0.223701
Kurtosis               6.125539
Tail ratio             0.988875
Daily value at risk   -0.005901
Alpha                  0.005922
Beta                   0.121033
dtype: float64
```

[15]:
```
fig, ax = plt.subplots(figsize=(14, 5))
plot_perf_stats(returns=returns,
                factor_returns=benchmark_rets,
                ax=ax)
sns.despine()
fig.tight_layout();
```



1.3 Returns Analysis

Testing a trading strategy involves backtesting against historical data to fine-tune alpha factor parameters, as well as forward-testing against new market data to validate that the strategy performs well out of sample or if the parameters are too closely tailored to specific historical circumstances.

Pyfolio allows for the designation of an out-of-sample period to simulate walk-forward testing. There are numerous aspects to take into account when testing a strategy to obtain statistically reliable results, which we will address here.

[16]:
```
oos_date = '2016-01-01'
```

[17]:
```
show_perf_stats(returns=returns,
                factor_returns=benchmark_rets,
                positions=positions,
                transactions=transactions,
                live_start_date=oos_date)
```

```
<IPython.core.display.HTML object>
```

1.3.1 Rolling Returns OOS

The `plot_rolling_returns` function displays cumulative in and out-of-sample returns against a user-defined benchmark (we are using the S&P 500):

[18]:
```
plot_rolling_returns(returns=returns,
                     factor_returns=benchmark_rets,
```

```
                          live_start_date=oos_date,
                          cone_std=(1.0, 1.5, 2.0))
plt.gcf().set_size_inches(14, 8)
sns.despine()
plt.tight_layout();
```

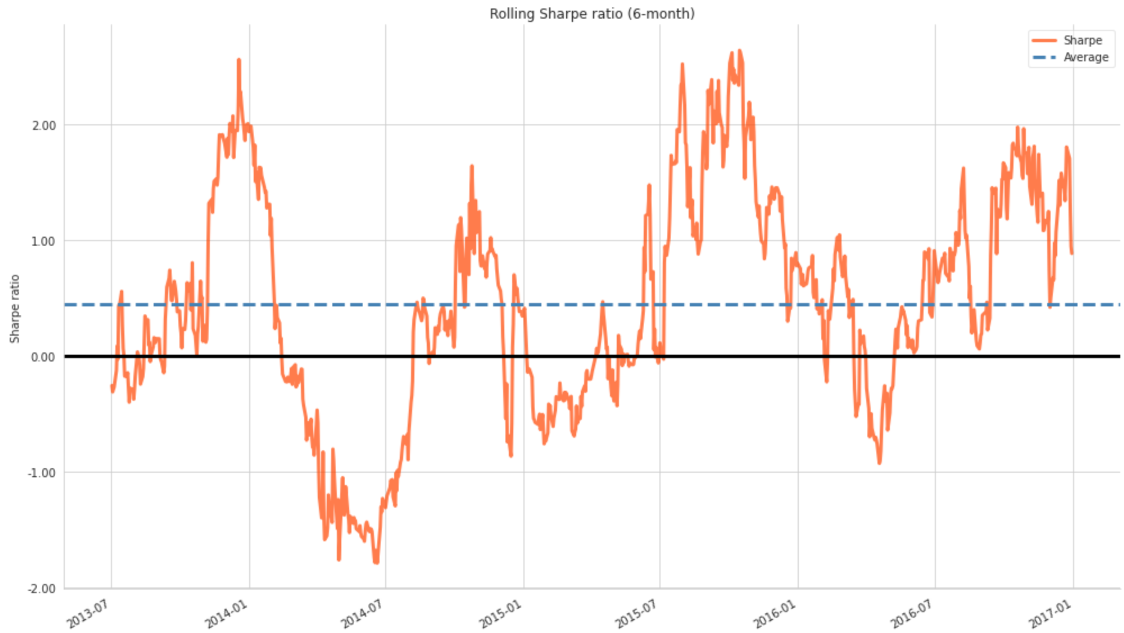
The plot includes a cone that shows expanding confidence intervals to indicate when out-of-sample returns appear unlikely given random-walk assumptions. Here, our strategy did not perform well against the benchmark during the simulated 2017 out-of-sample period

1.4 Summary Performance Statistics

pyfolio offers several analytic functions and plots. The perf_stats summary displays the annual and cumulative returns, volatility, skew, and kurtosis of returns and the SR. The following additional metrics (which can also be calculated individually) are most important: - Max drawdown: Highest percentage loss from the previous peak - Calmar ratio: Annual portfolio return relative to maximal drawdown - Omega ratio: The probability-weighted ratio of gains versus losses for a return target, zero per default - Sortino ratio: Excess return relative to downside standard deviation - Tail ratio: Size of the right tail (gains, the absolute value of the 95th percentile) relative to the size of the left tail (losses, abs. value of the 5th percentile) - Daily value at risk (VaR): Loss corresponding to a return two standard deviations below the daily mean - Alpha: Portfolio return unexplained by the benchmark return - Beta: Exposure to the benchmark

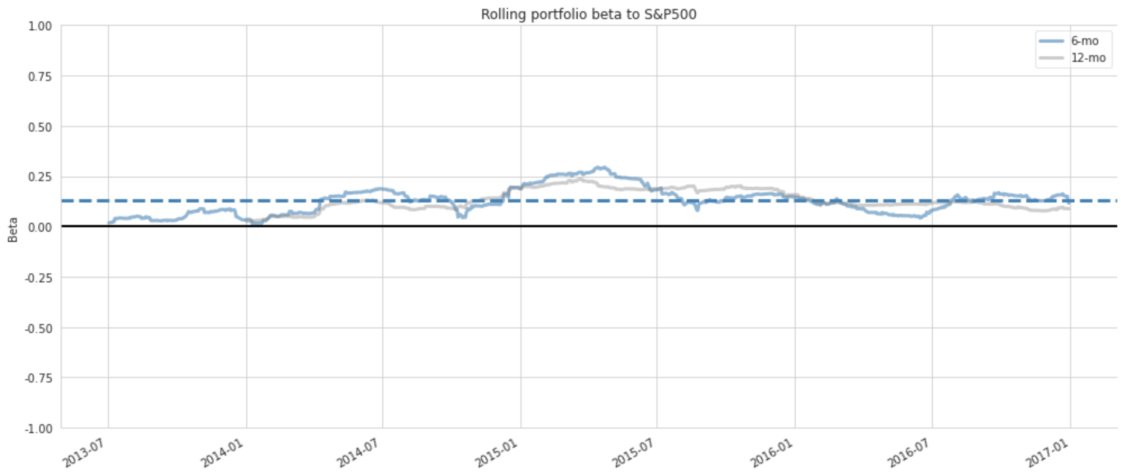
1.4.1 Rolling Sharpe

[19]:
```
plot_rolling_sharpe(returns=returns)
plt.gcf().set_size_inches(14, 8)
sns.despine()
plt.tight_layout();
```



1.4.2 Rolling Beta

[20]:
```
plot_rolling_beta(returns=returns, factor_returns=benchmark_rets)
plt.gcf().set_size_inches(14, 6)
sns.despine()
plt.tight_layout();
```

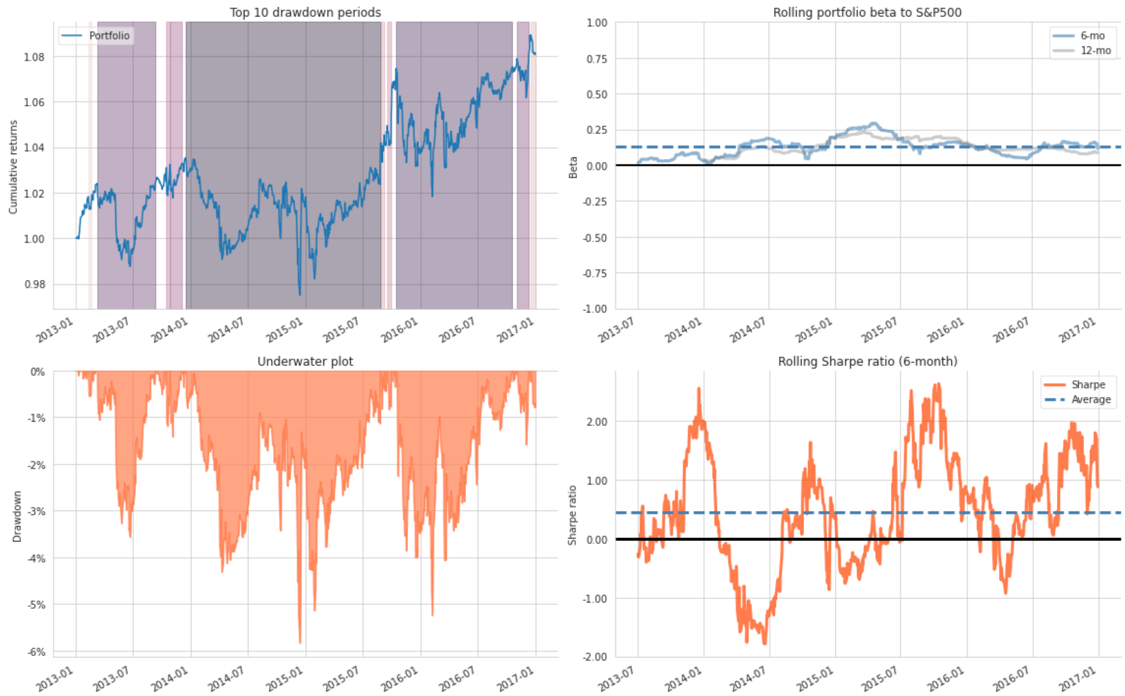


1.5 Drawdown Periods

The plot_drawdown_periods(returns) function plots the principal drawdown periods for the portfolio, and several other plotting functions show the rolling SR and rolling factor exposures to the market beta or the Fama French size, growth, and momentum factors:

[21]:
```
fig, ax = plt.subplots(nrows=2, ncols=2, figsize=(16, 10))
axes = ax.flatten()

plot_drawdown_periods(returns=returns, ax=axes[0])
plot_rolling_beta(returns=returns, factor_returns=benchmark_rets, ax=axes[1])
plot_drawdown_underwater(returns=returns, ax=axes[2])
plot_rolling_sharpe(returns=returns)
sns.despine()
plt.tight_layout();
```

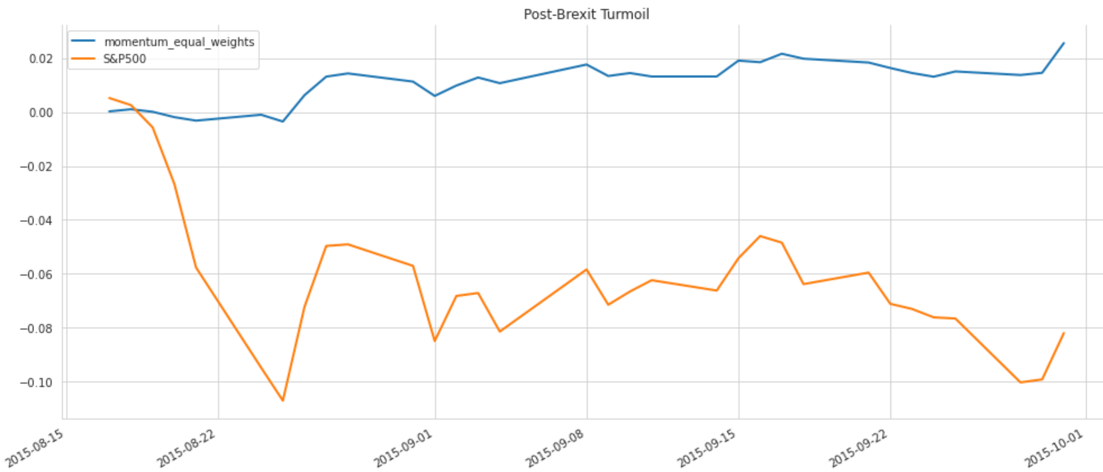


This plot, which highlights a subset of the visualization contained in the various tear sheets, illustrates how pyfolio allows us to drill down into the performance characteristics and exposure to fundamental drivers of risk and returns.

1.6 Modeling Event Risk

Pyfolio also includes timelines for various events that you can use to compare the performance of a portfolio to a benchmark during this period, for example, during the fall 2015 selloff following the Brexit vote.

[22]:
```
interesting_times = extract_interesting_date_ranges(returns=returns)
(interesting_times['Fall2015']
 .to_frame('momentum_equal_weights').join(benchmark_rets)
 .add(1).cumprod().sub(1)
 .plot(lw=2, figsize=(14, 6), title='Post-Brexit Turmoil'))
sns.despine()
plt.tight_layout();
```


Post-Brexit Turmoil
momentum_equal_weights
S&P500
0.02
0.00
−0.02
−0.04
−0.06
−0.08
−0.10
2015-08-15
2015-08-22
2015-09-01
2015-09-08
2015-09-15
2015-09-22
2015-10-01