

# 02\_vectorized\_backtest

September 29, 2021

## 1 Vectorized Backtest

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

```
[2]: from pathlib import Path
from time import time
import datetime

import numpy as np
import pandas as pd
import pandas_datareader.data as web

from scipy.stats import spearmanr

import matplotlib.pyplot as plt
from matplotlib.ticker import FuncFormatter
import seaborn as sns
```

```
[3]: sns.set_style('whitegrid')
np.random.seed(42)
```

### 1.1 Load Data

#### 1.1.1 Return Predictions

```
[4]: DATA_DIR = Path('.', 'data')
```

```
[5]: data = pd.read_hdf('00_data/backtest.h5', 'data')
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 190451 entries, ('AAL', Timestamp('2014-12-09 00:00:00')) to ('YUM',
Timestamp('2017-11-30 00:00:00'))
Data columns (total 6 columns):
#   Column      Non-Null Count  Dtype
---  -
0   predicted    74054 non-null  float64
```

```

1  open      190451 non-null float64
2  high      190451 non-null float64
3  low       190451 non-null float64
4  close     190451 non-null float64
5  volume    190451 non-null float64
dtypes: float64(6)
memory usage: 10.2+ MB

```

### 1.1.2 SP500 Benchmark

```
[6]: sp500 = web.DataReader('SP500', 'fred', '2014', '2018').pct_change()
```

```
[7]: sp500.info()
```

```

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1044 entries, 2014-01-01 to 2018-01-01
Data columns (total 1 columns):
#   Column  Non-Null Count  Dtype
---  -
0    SP500   1042 non-null      float64
dtypes: float64(1)
memory usage: 16.3 KB

```

## 1.2 Compute Forward Returns

```
[8]: daily_returns = data.open.unstack('ticker').sort_index().pct_change()
daily_returns.info()
```

```

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 751 entries, 2014-12-09 to 2017-11-30
Columns: 257 entries, AAL to YUM
dtypes: float64(257)
memory usage: 1.5 MB

```

```
[9]: fwd_returns = daily_returns.shift(-1)
```

## 1.3 Generate Signals

```
[10]: predictions = data.predicted.unstack('ticker')
predictions.info()
```

```

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 751 entries, 2014-12-09 to 2017-04-14
Columns: 257 entries, AAL to YUM
dtypes: float64(257)
memory usage: 1.5 MB

```

```
[11]: N_LONG = N_SHORT = 15
```

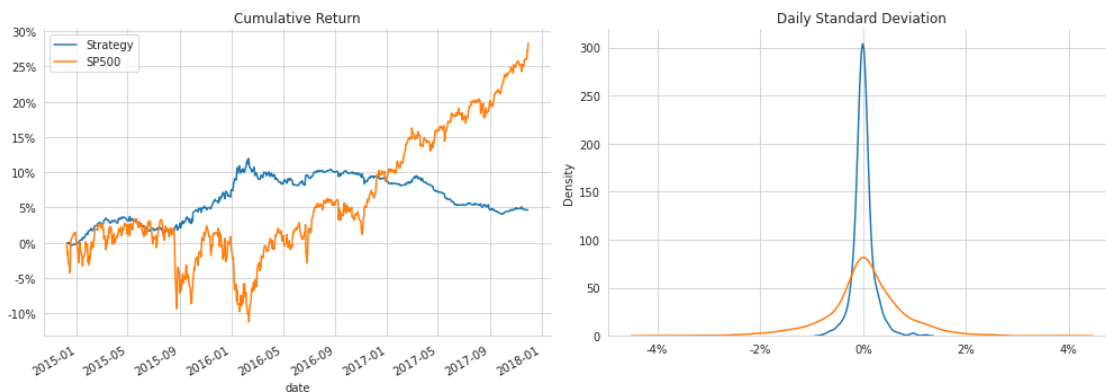
```
[12]: long_signals = ((predictions
                        .where(predictions > 0)
                        .rank(axis=1, ascending=False) > N_LONG)
                        .astype(int))
short_signals = ((predictions
                  .where(predictions < 0)
                  .rank(axis=1) > N_SHORT)
                  .astype(int))
```

## 1.4 Compute Portfolio Returns

```
[13]: long_returns = long_signals.mul(fwd_returns).mean(axis=1)
short_returns = short_signals.mul(-fwd_returns).mean(axis=1)
strategy = long_returns.add(short_returns).to_frame('Strategy')
```

## 1.5 Plot results

```
[14]: fig, axes = plt.subplots(ncols=2, figsize=(14,5))
strategy.join(sp500).add(1).cumprod().sub(1).plot(ax=axes[0], title='Cumulative_
↪Return')
sns.distplot(strategy.dropna(), ax=axes[1], hist=False, label='Strategy')
sns.distplot(sp500, ax=axes[1], hist=False, label='SP500')
axes[1].set_title('Daily Standard Deviation')
axes[0].yaxis.set_major_formatter(FuncFormatter(lambda y, _: '{:.0%}'.
↪format(y)))
axes[1].xaxis.set_major_formatter(FuncFormatter(lambda y, _: '{:.0%}'.
↪format(y)))
sns.despine()
fig.tight_layout();
```



```
[15]: res = strategy.join(sp500).dropna()
```

```
[16]: res.std()
```

```
[16]: Strategy    0.001981  
      SP500      0.007923  
      dtype: float64
```

```
[17]: res.corr()
```

```
[17]:          Strategy    SP500  
      Strategy    1.0000 -0.1184  
      SP500      -0.1184  1.0000
```