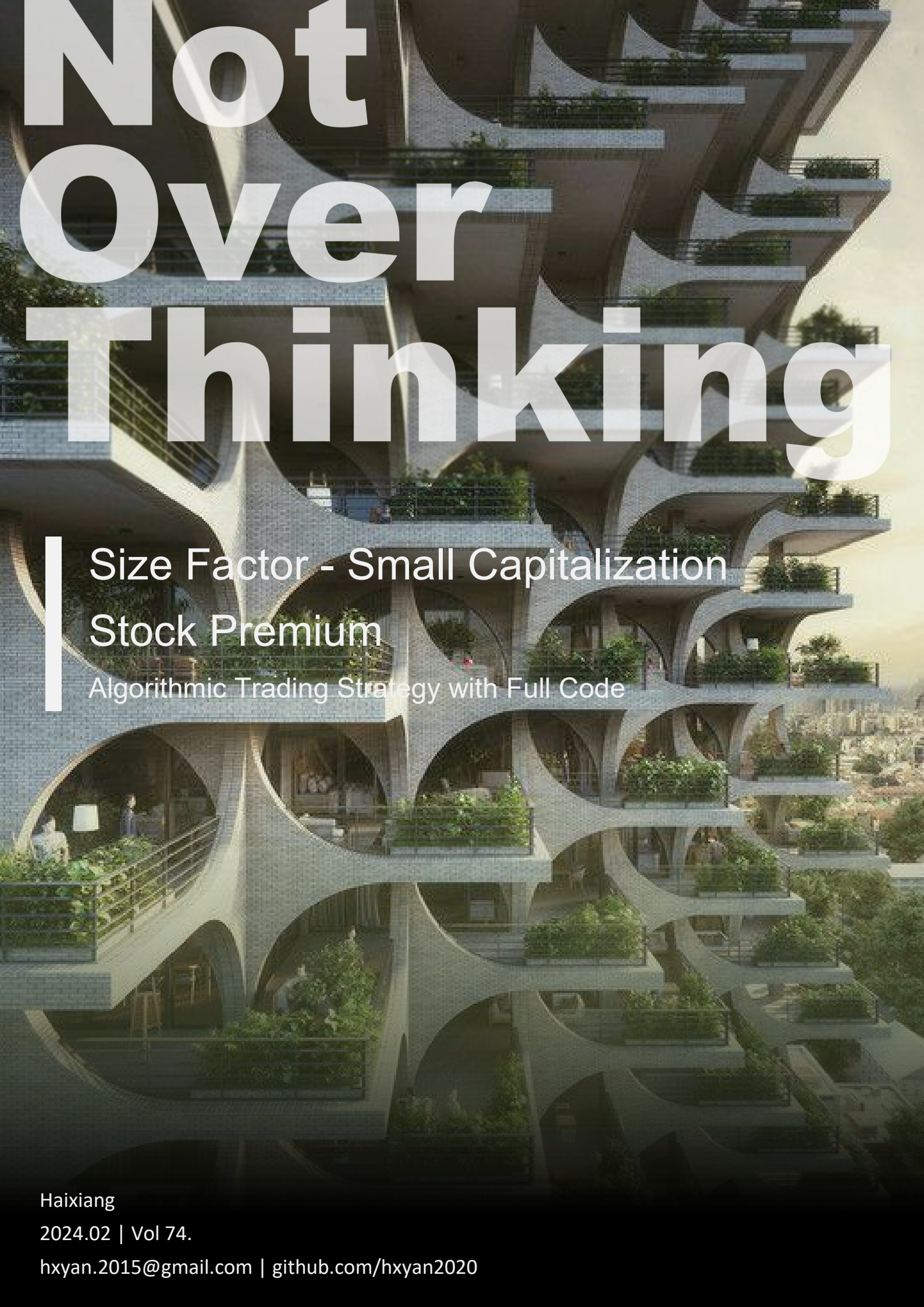

Not Over Thinking

Size Factor - Small Capitalization Stock Premium

Algorithmic Trading Strategy with Full Code

Haixiang

2024.02 | Vol 74.

hxyan.2015@gmail.com | github.com/hxyan2020

STRATEGY & ECONOMIC RATIONALE

The investment universe contains all NYSE, AMEX, and NASDAQ stocks. Decile portfolios are forme d based on the market capitalization of stocks. To capture “size” effect, SMB portfolio goes lo ng small stocks (lowest decile) and short big stocks (highest decile).

BUY	SELL
goes long small stocks (lowe st decile)	short big stocks (highest decile)

PARAMETER & VARIABLES

PARAMETER	VALUE
MARKETS TRADED	Equity
FINANCIAL INSTRUMENTS	Stocks
REGION	United States
PERIOD OF REBALANCING	Tearly
NO. OF TRADED INSTRUMENTS	1000
WEIGHTING	Equal weighting
LOOKBACK PERIODS	N/A
LONG/SHORT	Long & short

ALGORITHM

```
from AlgorithmImports import *
class ValueBooktoMarketFactor(QCAlgorithm):

    def Initialize(self):
        self.SetStartDate(2000, 1, 1)
        self.SetCash(100000)

        self.symbol = self.AddEquity('SPY', Resolution.Daily).Symbol

        self.coarse_count = 3000
        self.quantile:int = 5

        self.long = []
        self.short = []

        self.month = 12
        self.selection_flag = False
        self.UniverseSettings.Resolution = Resolution.Daily
        self.AddUniverse(self.CoarseSelectionFunction, self.FineSelectionFunction)
        self.Schedule.On(self.DateRules.MonthEnd(self.symbol), self.TimeRules.AfterMarketOpen(s
elf.symbol), self.Selection)

    def OnSecuritiesChanged(self, changes):
        for security in changes.AddedSecurities:
            security.SetFeeModel(CustomFeeModel())
            security.SetLeverage(10)

    def CoarseSelectionFunction(self, coarse):
        if not self.selection_flag:
            return Universe.Unchanged
```

```
        selected = [x.Symbol for x in coarse if x.HasFundamentalData and x.Market == 'usa']
        return selected

    def FineSelectionFunction(self, fine):
        sorted_by_market_cap = sorted([x for x in fine if x.MarketCap != 0 and  \
                                ((x.SecurityReference.ExchangeId == "NYS") or (x.SecurityRefere
nce.ExchangeId == "NAS") or (x.SecurityReference.ExchangeId == "ASE"))],    \
                                key = lambda x:x.MarketCap, reverse=True)

        top_by_market_cap = [x for x in sorted_by_market_cap[:self.coarse_count]]

        if len(top_by_market_cap) >= self.quantile:
            quintile = int(len(top_by_market_cap) / self.quantile)

            self.long = [i.Symbol for i in top_by_market_cap[-quintile:]]
            self.short = [i.Symbol for i in top_by_market_cap[:quintile]]

        return self.long + self.short

    def OnData(self, data):
        if not self.selection_flag:
            return
        self.selection_flag = False

        # Trade execution.
        long_count = len(self.long)
        short_count = len(self.short)

        stocks_invested = [x.Key for x in self.Portfolio if x.Value.Invested]
        for symbol in stocks_invested:
            if symbol not in self.long + self.short:
                self.Liquidate(symbol)

        # Leveraged portfolio - 100% long, 100% short.
        for symbol in self.long:
            if symbol in data and data[symbol]:
                self.SetHoldings(symbol, 1 / long_count)

        for symbol in self.short:
            if symbol in data and data[symbol]:
                self.SetHoldings(symbol, -1 / short_count)

        self.long.clear()
        self.short.clear()

    def Selection(self):
        if self.month == 12:
            self.selection_flag = True

        self.month += 1
        if self.month > 12:
            self.month = 1
# Custom fee model.class CustomFeeModel(FeeModel):
    def GetOrderFee(self, parameters):
        fee = parameters.Security.Price * parameters.Order.AbsoluteQuantity * 0.00005
        return OrderFee(CashAmount(fee, "USD"))
```

BACKTESTING PERFORMANCE

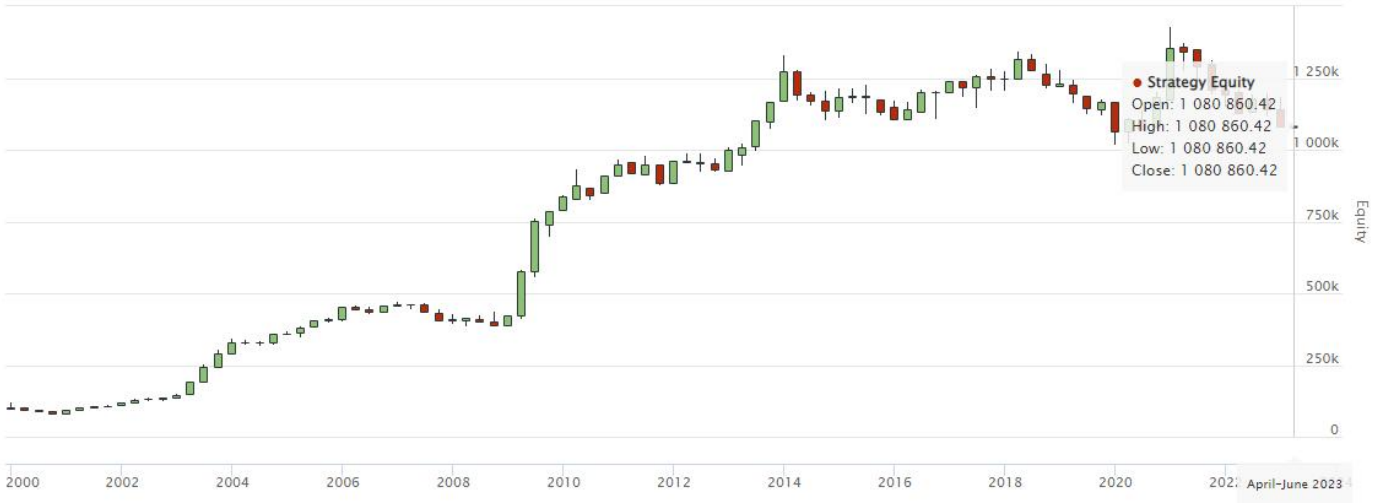

Fig 1. Overall Performance

Total Trades	25866	Average Win	0.09%
Average Loss	-0.03%	Compounding Annual Return	10.770%
Drawdown	34.800%	Expectancy	0.689
Net Profit	980.860%	Sharpe Ratio	0.723
Probabilistic Sharpe Ratio	3.815%	Loss Rate	56%
Win Rate	44%	Profit-Loss Ratio	2.82
Alpha	0.098	Beta	-0.32
Annual Standard Deviation	0.11	Annual Variance	0.012
Information Ratio	0.093	Tracking Error	0.235
Treynor Ratio	-0.249	Total Fees	$1235.36
Estimated Strategy Capacity	$10000.00	Lowest Capacity Asset	YNDX UWU1S0AN2N39
Portfolio Turnover	0.37%		

Fig 2. Performance Metrics