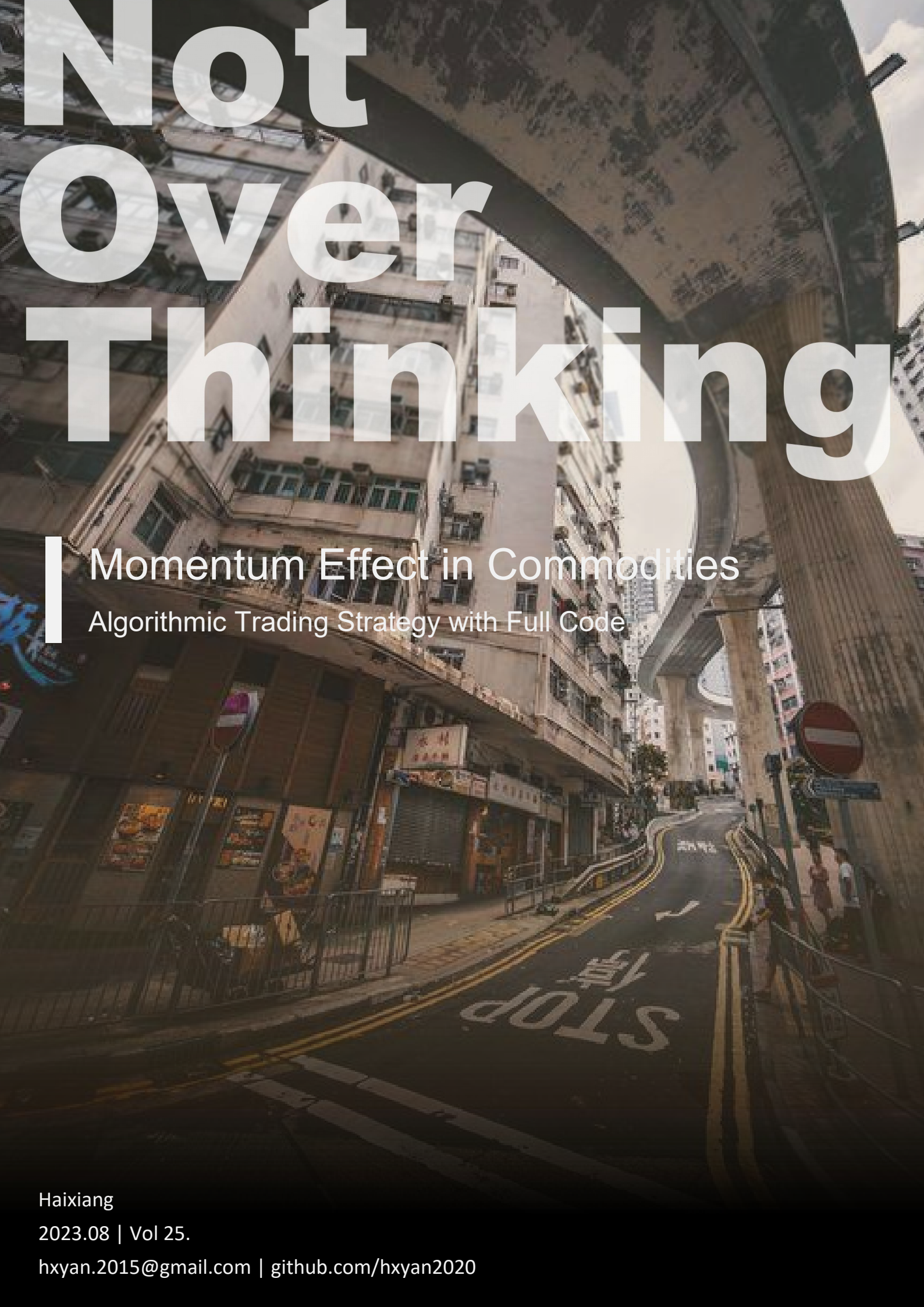
Not Over Thinking



Momentum Effect in Commodities

Algorithmic Trading Strategy with Full Code

Haixiang

2023.08 | Vol 25.

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

STRATEGY & ECONOMIC RATIONALE

Create a universe of tradable commodity futures. Rank futures performance for each commodity for the last 12 months and divide them into quintiles. Go long on the quintile with the highest momentum and go short on the quintile with the lowest momentum. Rebalance each month.

BUY	SELL
the quintile with the highest momentum	the quintile with the lowest momentum

PARAMETER & VARIABLES

PARAMETER	VALUE
MARKETS TRADED	Commodities
FINANCIAL INSTRUMENTS	CFDs, futures
REGION	Global
PERIOD OF REBALANCING	Monthly
NO. OF TRADED INSTRUMENTS	10
WEIGHTING	Equal Weighting
LOOKBACK PERIODS	12 months
LONG/SHORT	Long and short

ALGORITHM

```
from AlgorithmImports import *

class MomentumEffectCommodities(QCAlgorithm):

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

        self.symbols = [
                        "CME_S1",   # Soybean Futures, Continuous Contract
                        "CME_W1",   # Wheat Futures, Continuous Contract
                        "CME_SM1",  # Soybean Meal Futures, Continuous Contract
                        "CME_BO1",  # Soybean Oil Futures, Continuous Contract
                        "CME_C1",   # Corn Futures, Continuous Contract
                        "CME_O1",   # Oats Futures, Continuous Contract
                        "CME_LC1",  # Live Cattle Futures, Continuous Contract
                        "CME_FC1",  # Feeder Cattle Futures, Continuous Contract
                        "CME_LN1",  # Lean Hog Futures, Continuous Contract
                        "CME_GC1",  # Gold Futures, Continuous Contract
                        "CME_SI1",  # Silver Futures, Continuous Contract
                        "CME_PL1",  # Platinum Futures, Continuous Contract
                        "CME_CL1",  # Crude Oil Futures, Continuous Contract
                        "CME_HG1",  # Copper Futures, Continuous Contract
                        "CME_LB1",  # Random Length Lumber Futures, Continuous Contract
```

```
                        "CME_NG1",  # Natural Gas (Henry Hub) Physical Futures, Continuous
Contract
                        "CME_PA1",  # Palladium Futures, Continuous Contract
                        "CME_RR1",  # Rough Rice Futures, Continuous Contract
                        "CME_DA1",  # Class III Milk Futures
                        "ICE_RS1",  # Canola Futures, Continuous Contract
                        "ICE_GO1",  # Gas Oil Futures, Continuous Contract
                        "CME_RB2",  # Gasoline Futures, Continuous Contract
                        "CME_KW2",  # Wheat Kansas, Continuous Contract
                        "ICE_WT1",  # WTI Crude Futures, Continuous Contract

                        "ICE_CC1",  # Cocoa Futures, Continuous Contract
                        "ICE_CT1",  # Cotton No. 2 Futures, Continuous Contract
                        "ICE_KC1",  # Coffee C Futures, Continuous Contract
                        "ICE_O1",   # Heating Oil Futures, Continuous Contract
                        "ICE_OJ1",  # Orange Juice Futures, Continuous Contract
                        "ICE_SB1",  # Sugar No. 11 Futures, Continuous Contract
                        ]

        self.period = 12 * 21
        self.SetWarmUp(self.period, Resolution.Daily)
        self.data = {}

        for symbol in self.symbols:
            data = self.AddData(QuantpediaFutures, symbol, Resolution.Daily)
            data.SetFeeModel(CustomFeeModel())
            data.SetLeverage(5)
            self.data[symbol] = self.ROC(symbol, self.period, Resolution.Daily)

        self.recent_month = -1

    def OnData(self, data):
        if self.IsWarmingUp:
            return

        # rebalance once a month
        if self.recent_month == self.Time.month:
            return
        self.recent_month = self.Time.month

        perf = { x[0] : x[1].Current.Value for x in self.data.items() if
self.data[x[0]].IsReady and x[0] in data and data[x[0]] }

        long = []
        short = []
        if len(perf) >= 5:
            sorted_by_performance = sorted(perf.items(), key = lambda x:x[1], reverse=True)
            quintile = int(len(sorted_by_performance) / 5)
            long = [x[0] for x in sorted_by_performance[:quintile]]
            short = [x[0] for x in sorted_by_performance[-quintile:]]

        # trade execution
        invested = [x.Key.Value for x in self.Portfolio if x.Value.Invested]
```

```
        for symbol in invested:
            if symbol not in long + short:
                self.Liquidate(symbol)

        for symbol in long:
            self.SetHoldings(symbol, 1 / len(long))
        for symbol in short:
            self.SetHoldings(symbol, -1 / len(short))

# Quantpedia data.
# NOTE: IMPORTANT: Data order must be ascending (datewise)
class QuantpediaFutures(PythonData):
    def GetSource(self, config, date, isLiveMode):
        return
SubscriptionDataSource("data.quantpedia.com/backtesting_data/futures/{0}.csv".format(confi
g.Symbol.Value), SubscriptionTransportMedium.RemoteFile, FileFormat.Csv)

    def Reader(self, config, line, date, isLiveMode):
        data = QuantpediaFutures()
        data.Symbol = config.Symbol

        if not line[0].isdigit(): return None
        split = line.split(';')

        data.Time = datetime.strptime(split[0], "%d.%m.%Y") + timedelta(days=1)
        data['back_adjusted'] = float(split[1])
        data['spliced'] = float(split[2])
        data.Value = float(split[1])

        return data

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

BACKTESTING PERFORMANCE

Fig 1. Overall Performance

PSR	0.000%	Sharpe Ratio	0.106
Total Trades	3063	Average Win	1.11%
Average Loss	-1.55%	Compounding Annual Return	0.088%
Drawdown	76.700%	Expectancy	0.003
Net Profit	2.060%	Loss Rate	41%
Win Rate	59%	Profit-Loss Ratio	0.71
Alpha	0.028	Beta	-0.106
Annual Standard Deviation	0.204	Annual Variance	0.042
Information Ratio	-0.127	Tracking Error	0.271
Treynor Ratio	-0.206	Total Fees	$2170.53
Estimated Strategy Capacity	$0	Lowest Capacity Asset	CME_W1.QuantpediaFutures 2S

Fig 2. Performance Metrics

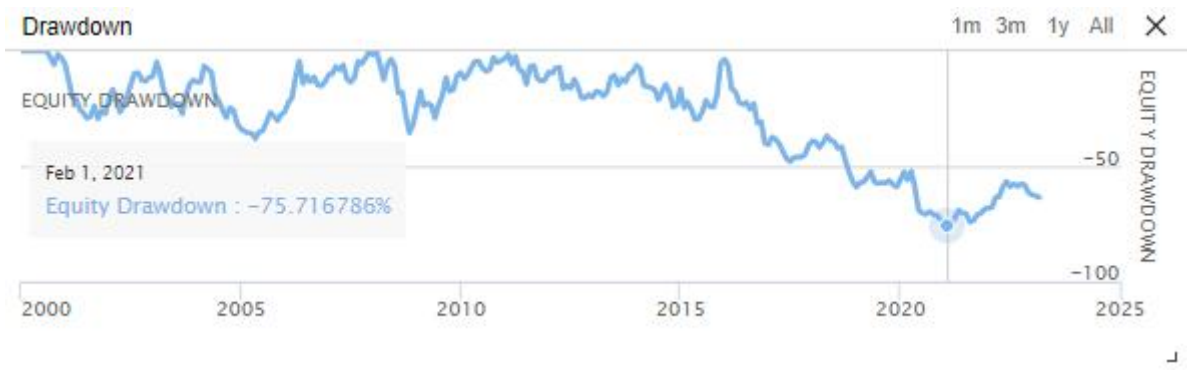


Fig 3. Drawdown

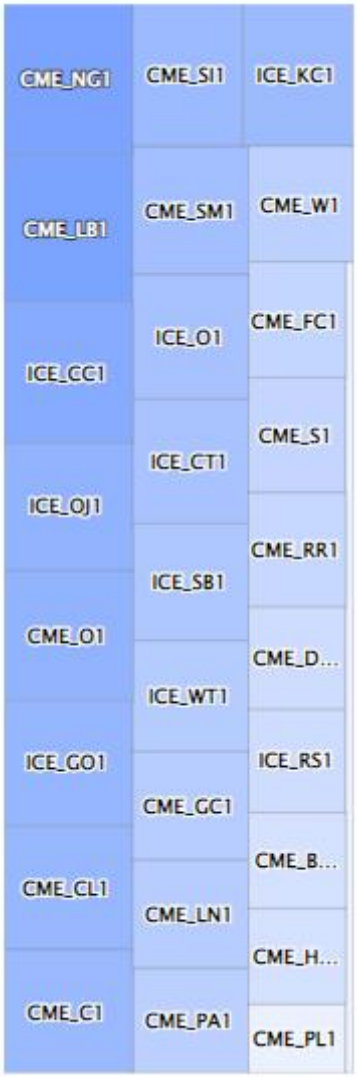


Fig 4. Assets Sales Volume