Not Over Thinking

Currency Momentum Factor

Algorithmic Trading Strategy with Full Code

Haixiang

2023.03 | Vol 19.

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

STRATEGY & ECONOMIC RATIONALE

Create an investment universe consisting of several currencies (10-20). Go long three currencies with the highest 12-month momentum against USD and go short three currencies with the lowest 12-month momentum against USD. Cash not used as margin invest on overnight rates. Rebalance monthly.

BUY	SELL
Currencies with highest 12-month momentum against USD	Currencies with lowest 12-month momentum against USD

PARAMETER & VARIABLES

PARAMETER	VALUE
MARKETS TRADED	Currency
FINANCIAL INSTRUMENTS	CFD, forward, future, swap
REGION	Global
PERIOD OF REBALANCING	Monthly
NO. OF TRADED INSTRUMENTS	10
WEIGHTING	Equal weighting
LOOKBACK PERIODS	12-month
LONG/SHORT	Long & Short

ALGORITHM

<data_tools.py>

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

# Quandl "value" data
class QuandlValue(PythonQuandl):
    def __init__(self):
        self.ValueColumnName = 'Value'

# 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(config.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
```

<main.py>

```
import data_tools
from AlgorithmImports import *

class CurrencyMomentumFactor(QCAlgorithm):

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

        self.data = {}
        self.period = 12 * 21  ##21 years till 2020
        self.SetWarmUp(self.period, Resolution.Daily)

        self.symbols = [
                        "CME_AD1", # Australian Dollar Futures, Continuous Contract #1
                        "CME_BP1", # British Pound Futures, Continuous Contract #1
                        "CME_CD1", # Canadian Dollar Futures, Continuous Contract #1
                        "CME_EC1", # Euro FX Futures, Continuous Contract #1
                        "CME_JY1", # Japanese Yen Futures, Continuous Contract #1
                        "CME_MP1", # Mexican Peso Futures, Continuous Contract #1
                        "CME_NE1", # New Zealand Dollar Futures, Continuous Contract #1
                        "CME_SF1"  # Swiss Franc Futures, Continuous Contract #1
                        ]

        for symbol in self.symbols:
            data = self.AddData(data_tools.QuantpediaFutures, symbol, Resolution.Daily)
##load data from data_tools.QuantpediaFutures
            data.SetFeeModel(data_tools.CustomFeeModel()) ##calling function SetFeeModel()
defined earlier
            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 monthly
```

```
        if self.Time.month == self.recent_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]] } ## .items() is system default
function

        long = []
        short = []
        if len(perf) >= 6: ## if there are more than 6 currencies
            sorted_by_performance = sorted(perf.items(), key = lambda x:x[1],
reverse=True)
            long = [x[0] for x in sorted_by_performance[:3]] ## long the top 3 currencies
with highest momentum
            short = [x[0] for x in sorted_by_performance[-3:]] ## short the bottom 3
currencies with lowest momentum

        # 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))
```

BACKTESTING PERFORMANCE

Fig 1. Overall Performance

PSR	0.000%	Sharpe Ratio	-0.013
Total Trades	1823	Average Win	0.35%
Average Loss	-0.56%	Compounding Annual Return	-0.444%
Drawdown	45.800%	Expectancy	-0.035
Net Profit	-9.813%	Loss Rate	41%
Win Rate	59%	Profit-Loss Ratio	0.63
Alpha	0.002	Beta	-0.057
Annual Standard Deviation	0.067	Annual Variance	0.004
Information Ratio	-0.318	Tracking Error	0.184
Treynor Ratio	0.015	Total Fees	$966.93
Estimated Strategy Capacity	$0	Lowest Capacity Asset	CME_BP1.QuantpediaFutures 2S

Fig 2. Performance Metrics

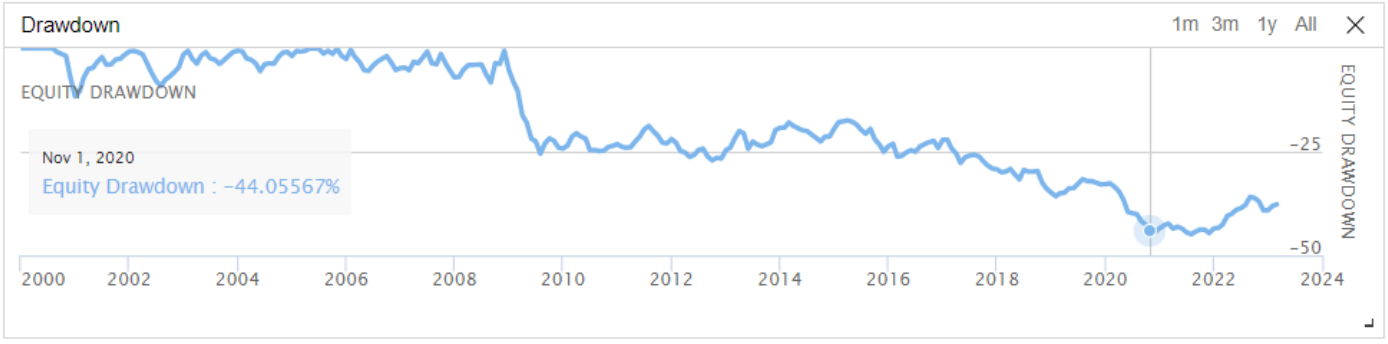

Fig 3. Drawdown

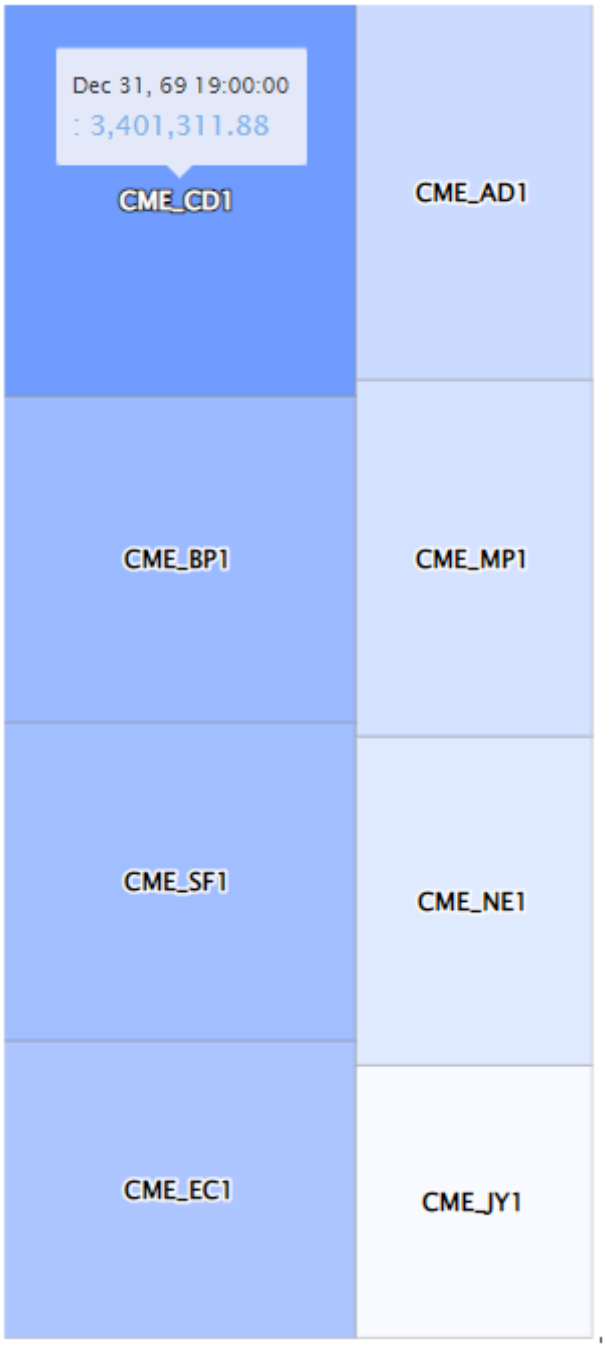

Fig 4. Assets Sales Volume