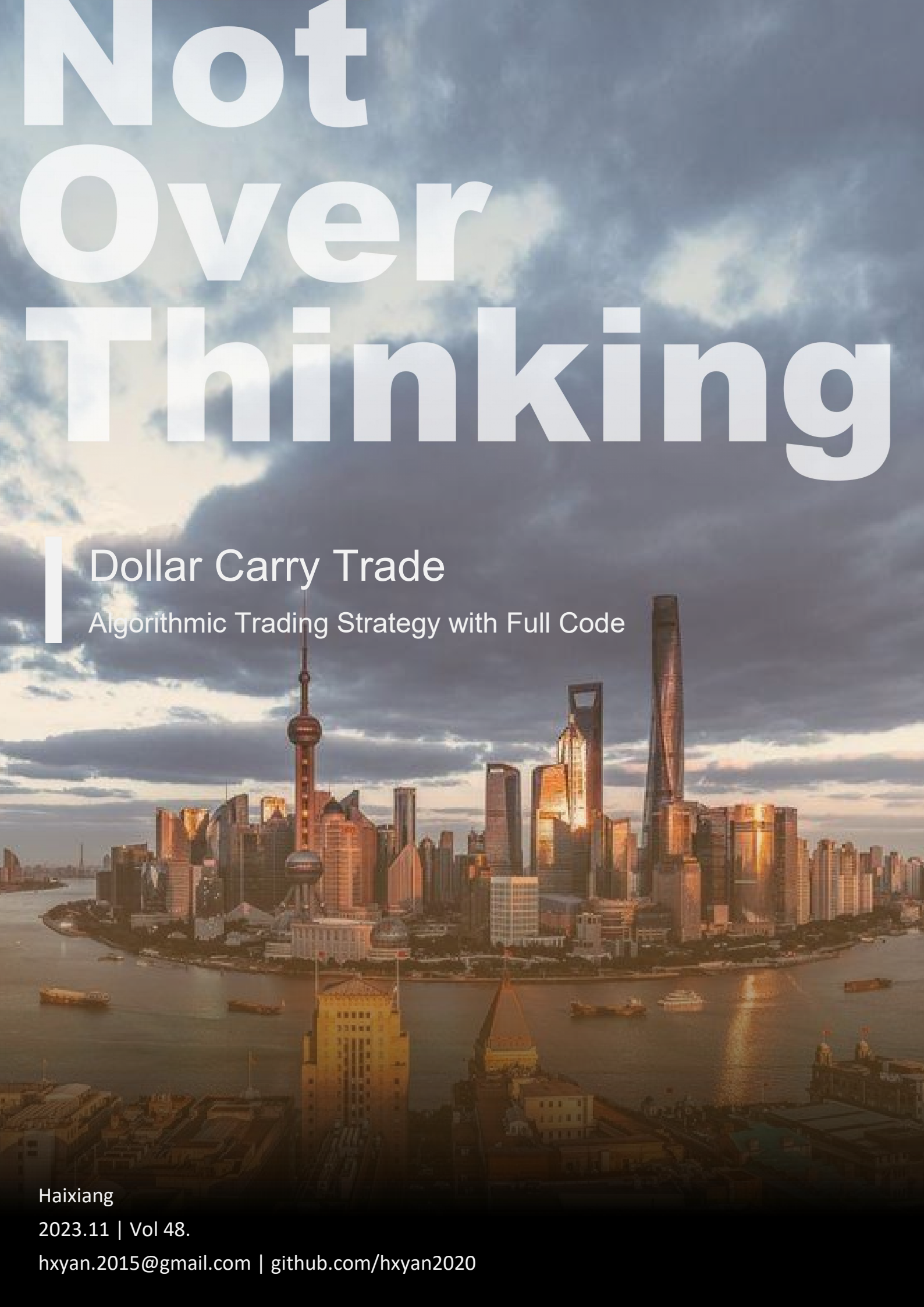
Not Over Thinking

Dollar Carry Trade

Algorithmic Trading Strategy with Full Code



Haixiang

2023.11 | Vol 48.

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

STRATEGY & ECONOMIC RATIONALE

The investment universe consists of currencies from developed countries (the Euro area, Australia, Canada, Denmark, Japan, New Zealand, Norway, Sweden, Switzerland, and the United Kingdom).

The average forward discount (AFD) is calculated for this basket of currencies (each currency has an equal weight). The average 3-month rate could be used instead of the AFD in the calculation. The AFD is then compared to the 3-month US Treasury rate.

The investor goes long on the US dollar and goes short on the basket of currencies if the 3-month US Treasury rate is higher than the AFD. The investor goes short on the US dollar and long on the basket of currencies if the 3-month US Treasury rate is lower than the AFD. The portfolio is rebalanced monthly.

BUY	SELL
goes long on the US dollar	goes short on the basket of currencies if the 3-month US Treasury rate is higher than the AFD

PARAMETER & VARIABLES

PARAMETER	VALUE
MARKETS TRADED	Currency
FINANCIAL INSTRUMENTS	CFDs, forwards, futures, swaps
REGION	Global
PERIOD OF REBALANCING	Monthly
NO. OF TRADED INSTRUMENTS	10
WEIGHTING	Equal weighting
LOOKBACK PERIODS	3 months
LONG/SHORT	Long & Short

ALGORITHM

```
import numpy as np
from AlgorithmImports import *

class DollarCarryTrade(QCAlgorithm):

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

        self.symbols = {
                        "CME_AD1" : "OECD/KEI_IR3TIB01_AUS_ST_M", # Australian Dollar Futures, Continuous Contract #1
                        "CME_BP1" : "OECD/KEI_IR3TIB01_GBR_ST_M", # British Pound Futures, Continuous Contract #1
                        "CME_CD1" : "OECD/KEI_IR3TIB01_CAN_ST_M", # Canadian Dollar Futures, Continuous Contract #1
```

```
                        "CME_EC1" : "OECD/KEI_IR3TIB01_EA19_ST_M",# Euro FX Futures,
Continuous Contract #1
                        "CME_JY1" : "OECD/KEI_IR3TIB01_JPN_ST_M", # Japanese Yen Futures,
Continuous Contract #1
                        "CME_MP1" : "OECD/KEI_IR3TIB01_MEX_ST_M", # Mexican Peso Futures,
Continuous Contract #1
                        "CME_NE1" : "OECD/KEI_IR3TIB01_NZL_ST_M", # New Zealand Dollar
Futures, Continuous Contract #1
                        "CME_SF1" : "SNB/ZIMOMA"                  # Swiss Franc Futures,
Continuous Contract #1
                        }

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

            # Interbank rate data.
            cash_rate_symbol = self.symbols[symbol]
            self.AddData(QuandlValue, cash_rate_symbol, Resolution.Daily)

        self.treasury_rate = self.AddData(QuandlValue, 'FRED/DGS3MO',
Resolution.Daily).Symbol

    def OnData(self, data):
        fd = {}
        for future_symbol, cash_rate_symbol in self.symbols.items():
            if cash_rate_symbol in data and data[cash_rate_symbol]:
                if self.Securities[future_symbol].GetLastData() and (self.Time.date() -
self.Securities[future_symbol].GetLastData().Time.date()).days < 5:
                    cash_rate = data[cash_rate_symbol].Value
                    # Update cash rate only once a month.
                    fd[future_symbol] = cash_rate

        if len(fd) == 0: return

        afd = np.mean([x[1] for x in fd.items()])

        if self.Securities[self.treasury_rate].GetLastData() and (self.Time.date() -
self.Securities[self.treasury_rate].GetLastData().Time.date()).days < 5:
            treasuries_3m_rate = self.Securities[self.treasury_rate].Price

            count = len(self.symbols)
            if treasuries_3m_rate > afd:
                # Long on the US dollar and goes short on the basket of currencies.
                for symbol in self.symbols:
                    self.SetHoldings(symbol, -1 / count)
            else:
                # Short on the US dollar and long on the basket of currencies.
                for symbol in self.symbols:
                    self.SetHoldings(symbol, 1 / count)

# 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

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

# 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

PSR	0.000%	Sharpe Ratio	0.129
Total Trades	2081	Average Win	0.07%
Average Loss	-0.06%	Compounding Annual Return	0.846%
Drawdown	26.000%	Expectancy	0.365
Net Profit	21.653%	Loss Rate	39%
Win Rate	61%	Profit-Loss Ratio	1.23
Alpha	0.002	Beta	0.091
Annual Standard Deviation	0.058	Annual Variance	0.003
Information Ratio	-0.319	Tracking Error	0.158
Treynor Ratio	0.083	Total Fees	$74.74
Estimated Strategy Capacity	$0	Lowest Capacity Asset	CME_AD1.QuantpediaFutures 2S
Portfolio Turnover	0.16%		

Fig 2. Performance Metrics

Fig 3. Drawdown

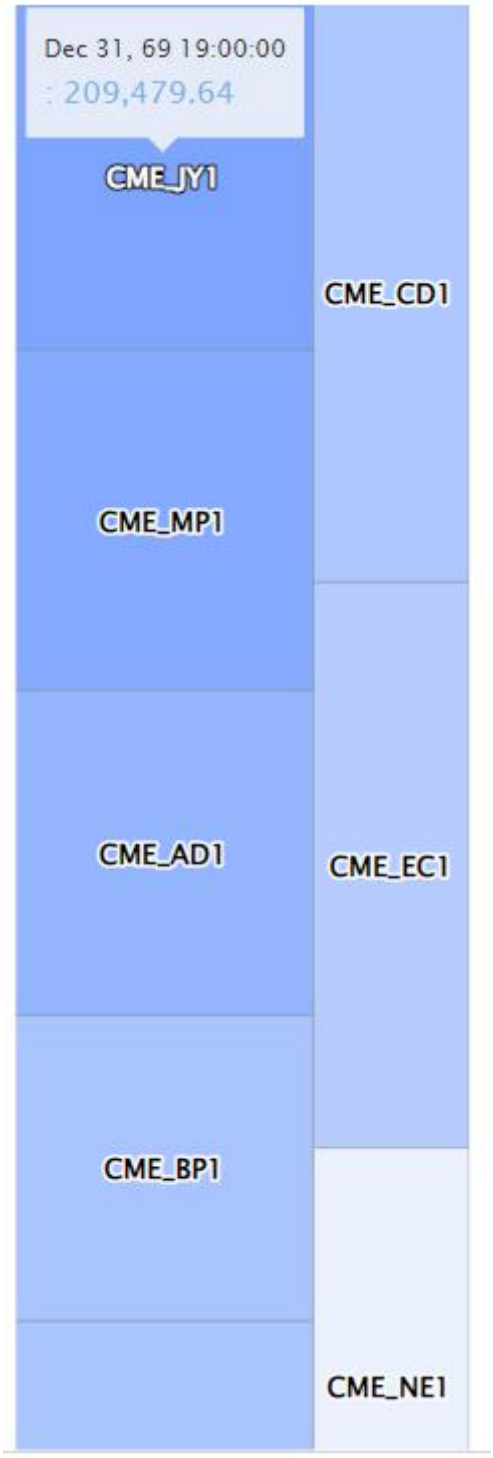


Fig 4. Assets Sales Volume