

Not Over Thinking

Re-balancing Premium in
Cryptocurrencies

Algorithmic Trading Strategy with Full Code

Haixiang

2024.01 | Vol 68.

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

STRATEGY & ECONOMIC RATIONALE

The investment universe consists of 27 cryptocurrencies: BAT (Basic Attention Token), BTC (Bitcoin), BTG (Bitcoin Gold), DAI (Dai), DATA (Data Coin), DGB (DigiByte), EOS (EIS.io), ETH (Ethereum), FUN (FUN Token), IOTA (Iota), LRC (Loopring token), LTC (Litecoin), MANA (Mana coin), NEO (Neo), OMG (OMG, Formally known as OmiseGo), REQ (Request), SAN (Sentiment Network Token), SNT (Status), TRX (Tron), WAX (Wax), XLM (Stellar), XMR (Monero), XRP (Ripple), XVG (Verge), ZEC (Zcash), ZIL (Zilliqa) and ZRX (0x). Two portfolios are created. The first portfolio is the daily rebalanced portfolio of all 27 cryptos to ensure that the assets have equal weights. The second portfolio is not rebalanced at all: an investor buys the equally-weighted crypto portfolio and lets the weights drift. Then the investor goes long the first portfolio and shorts the second portfolio with 70% weight. The ratio between first and second portfolio is daily rebalanced.

BUY	SELL
goes long the first portfolio	shorts the second portfolio with 70% weight

PARAMETER & VARIABLES

PARAMETER	VALUE
MARKETS TRADED	Crypto
FINANCIAL INSTRUMENTS	cryptos
REGION	Global
PERIOD OF REBALANCING	Daily
NO. OF TRADED INSTRUMENTS	27
WEIGHTING	Equal weighting
LOOKBACK PERIODS	N/A
LONG/SHORT	Long & short

ALGORITHM

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

    def Initialize(self):
        self.SetStartDate(2015, 1, 1)
        self.SetCash(100000000)

        self.cryptos = [
            "BTCUSD",
            "BATUSD",
            # "BTGUSD",
            "DAIUSD",
            "DGBUSD", "EOSUSD",
            "ETHUSD", "FUNUSD",
            "LTCUSD", "NEOUSD",
            "OMGUSD", "SNTUSD",
            "TRXUSD", "XLMUSD",
            "XMRUSD", "XRPUSD",
            "XVGUSD", "ZECUSD",
            "ZRXUSD", "LRCUSD",
            "REQUSD", "SANUSD",
            "WAXUSD", "ZILUSD",
            "IOTAUSD",
```

```

        "MANAUSD",
        "DATAUSD"
    ]

    self.short_side_percentage = 0.7
    self.data = {}
    self.SetBrokerageModel(BrokerageName.Bitfinex)

    for crypto in self.cryptos:
        # GDAX is coinmarket, but it doesn't support this many cryptos, so we choose Bitfin
        data = self.AddCrypto(crypto, Resolution.Minute, Market.Bitfinex)
        data.SetFeeModel(CustomFeeModel())
        data.SetLeverage(10)

        self.data[crypto] = SymbolData()

    self.was_traded_already = False          # wait for the price data to come only once
    self.prev_short_portfolio_equity = 0      # short leg equity tracking

def OnData(self, data):
    if not (self.Time.hour == 9 and self.Time.minute == 30):
        return

    all_cryptos_are_ready = True             # data warmup flag

    # check if all cryptos has ready data
    for crypto in self.cryptos:
        if crypto in data and data[crypto]:
            # update crypto price for weight calculation
            self.data[crypto].last_price = data[crypto].Value
            # if there is at least one crypto, which doesn't have data, then don't trade and br
        else:
            all_cryptos_are_ready = False
            break

    if all_cryptos_are_ready or self.was_traded_already:
        self.was_traded_already = True

        # long strategy equity calculation
        long_portfolio_equity = self.Portfolio.TotalPortfolioValue
        long_equity_to_trade = long_portfolio_equity / len(self.cryptos)

        # short strategy equity calculation
        short_portfolio_equity = self.Portfolio.TotalPortfolioValue * self.short_side_perce
        short_equity_to_trade = short_portfolio_equity / len(self.cryptos)

        # trading/rebalance
        for crypto, symbol_obj in self.data.items():
            if crypto in data and data[crypto]:
                # short strategy
                if not self.Portfolio[crypto].Invested:
                    short_q = np.floor(short_equity_to_trade / symbol_obj.last_price)
                    if abs(short_q) >= self.Securities[crypto].SymbolProperties.MinimumOrderSize:
                        self.MarketOrder(crypto, -short_q)

                # long strategy
                long_q = np.floor(long_equity_to_trade / symbol_obj.last_price)
                # currency was traded before
                if symbol_obj.quantity is not None:
                    # calculate quantity difference

```

Not Over Thinking – where I share my journey to algorithmic trading and investments in shortest words possible

```
diff_q = long_q - symbol_obj.quantity
```

```
# rebalance position
```

```
if abs(diff_q) >= self.Securities[crypto].SymbolProperties.MinimumOrder
```

Size:

```
self.MarketOrder(crypto, diff_q)
```

```
# change new quantity
```

```
symbol_obj.quantity += diff_q
```

```
else:
```

```
# rebalance position
```

```
if abs(long_q) >= self.Securities[crypto].SymbolProperties.MinimumOrder
```

Size:

```
self.MarketOrder(crypto, long_q)
```

```
# change new quantity
```

```
symbol_obj.quantity = long_q
```

```
class SymbolData():
```

```
def __init__(self):
```

```
    self.last_price = None
```

```
    self.quantity = None
```

```
# 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

Not Over Thinking – where I share my journey to algorithmic trading and investments in shortest words possible

Total Trades	41751	Average Win	0.05%
Average Loss	-0.02%	Compounding Annual Return	17.645%
Drawdown	60.600%	Expectancy	0.348
Net Profit	282.780%	Sharpe Ratio	0.585
Probabilistic Sharpe Ratio	5.798%	Loss Rate	59%
Win Rate	41%	Profit-Loss Ratio	2.31
Alpha	-0	Beta	0.211
Annual Standard Deviation	0.28	Annual Variance	0.078
Information Ratio	-1.107	Tracking Error	0.554
Treynor Ratio	0.775	Total Fees	\$818494.51
Estimated Strategy Capacity	\$0	Lowest Capacity Asset	FUNUSD E3
Portfolio Turnover	1.72%		

Fig 2. Performance Metrics