Not Over Thinking

Momentum Factor and Style Rotation Effect

Algorithmic Trading Strategy with Full Code

Haixiang

2023.10 | Vol 40.

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

STRATEGY & ECONOMIC RATIONALE

Russell's ETFs for six equity styles are used (small-cap value, mid-cap value, large-cap value, small-cap growth, mid-cap growth, large-cap growth). Each month, the investor calculates 12-month momentum for each style and goes long on the winner and short on the loser. The portfolio is rebalanced each month.

BUY	SELL
The momentum winner	The momentum loser

PARAMETER & VARIABLES

PARAMETER	VALUE
MARKETS TRADED	Equity
FINANCIAL INSTRUMENTS	ETFs
REGION	United States
PERIOD OF REBALANCING	Monthly
NO. OF TRADED INSTRUMENTS	6
WEIGHTING	Equal weighting
LOOKBACK PERIODS	12-months
LONG/SHORT	Long only

ALGORITHM

```
class MomentumFactorAndStyleRotationEffect(QCAlgorithm):

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

        self.tickers = [
            'IWS', # iShares Russell Midcap Value ETF
            'IWP', # iShares Russell Midcap Growth ETF
            'IWN', # iShares Russell 2000 Value ETF
            'IWO', # iShares Russell 2000 Growth ETF
            'IVE', # iShares S&P 500 Value ETF
            'IVW'  # iShares S&P 500 Growth ETF
        ]

        self.mom = {}

        self.period = 12 * 21
        self.SetWarmUp(self.period)

        for ticker in self.tickers:
            security = self.AddEquity(ticker, Resolution.Daily)
            security.SetFeeModel(CustomFeeModel())
            security.SetLeverage(10)
```

```
            self.mom[security.Symbol] = self.MOM(security.Symbol, self.period)

        self.recent_month = -1

    def OnData(self, data):
        if self.recent_month == self.Time.month:
            return
        self.recent_month = self.Time.month

        mom_ready = [ s for s in self.mom if self.mom[s].IsReady and s in data]
        if mom_ready:
            sorted_mom = sorted(mom_ready, key = lambda x: self.mom[x].Current.Value,
reverse=True)

            for symbol in sorted_mom[1:-1]:
                if self.Portfolio[symbol].Invested:
                    self.Liquidate(symbol)

            winner = sorted_mom[0]
            loser = sorted_mom[-1]

            if self.Securities[winner].Price != 0 and self.Securities[winner].IsTradable:
                if (self.Time.month == 10 and self.Time.year == 2020) and winner.Value ==
'IVW':    # prevent data error
                    self.Liquidate(winner)
                else:
                    self.SetHoldings(winner, 1)

            if self.Securities[loser].Price != 0 and self.Securities[loser].IsTradable:
                if (self.Time.month == 10 and self.Time.year == 2020) and loser.Value ==
'IVW':     # prevent data error
                    self.Liquidate(loser)
                else:
                    self.SetHoldings(loser, -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

PSR	0.000%	Sharpe Ratio	-0.032
Total Trades	630	Average Win	2.75%
Average Loss	-2.78%	Compounding Annual Return	-1.174%
Drawdown	53.600%	Expectancy	0.033
Net Profit	-23.984%	Loss Rate	48%
Win Rate	52%	Profit-Loss Ratio	0.99
Alpha	-0.001	Beta	-0.041
Annual Standard Deviation	0.099	Annual Variance	0.01
Information Ratio	-0.304	Tracking Error	0.196
Treynor Ratio	0.079	Total Fees	$1190.73
Estimated Strategy Capacity	$3100000.00	Lowest Capacity Asset	IVE RV0PWMLXVHPH

Fig 2. Performance Metrics

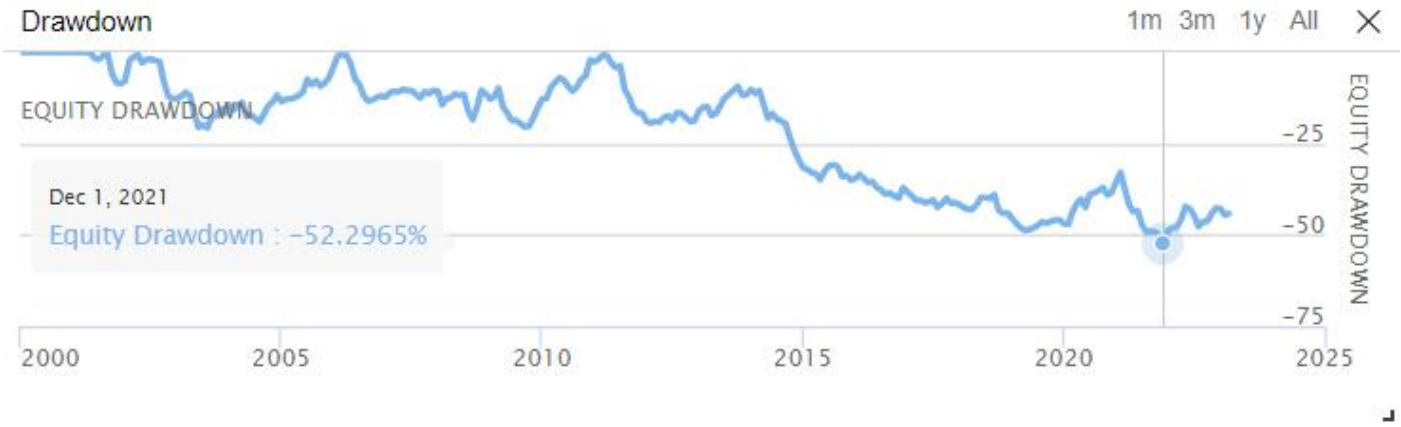

Fig 3. Drawdown

Fig 4. Assets Sales Volume