Not Over Thinking

Volatility Risk Premium Effect

Algorithmic Trading Strategy with Full Code

Haixiang

2023.08 | Vol 24.

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

STRATEGY & ECONOMIC RATIONALE

Each month, at-the-money straddle, with one month until maturity, is sold at the bid price with a 5% option premium, and an offsetting 15% out-of-the-money puts are bought (at the ask price) as insurance against a market crash. The remaining cash and received option premium are invested in the index. The strategy is rebalanced monthly.

BUY	SELL
(see above)	(see above)

PARAMETER & VARIABLES

PARAMETER	VALUE
MARKETS TRADED	Equity
FINANCIAL INSTRUMENTS	Futures, Options, Swaps
REGION	Global
PERIOD OF REBALANCING	Monthly
NO. OF TRADED INSTRUMENTS	4
WEIGHTING	Equal weighting
LOOKBACK PERIODS	Depends
LONG/SHORT	Long Only

ALGORITHM

```
from AlgorithmImports import *

class VolatilityRiskPremiumEffect(QCAlgorithm):

    def Initialize(self):
        self.SetStartDate(2010, 1, 1)
        self.SetCash(100000)

        data = self.AddEquity("SPY", Resolution.Minute)
        data.SetLeverage(5)
        self.symbol = data.Symbol

        option = self.AddOption("SPY", Resolution.Minute)
        option.SetFilter(-20, 20, 25, 35)

        self.last_day = -1

    def OnData(self,slice):
        # Check once a day.
        if self.Time.day == self.last_day:
            return
        self.last_day = self.Time.day

        for i in slice.OptionChains:
            chains = i.Value
```

```
        if not self.Portfolio.Invested:
            # divide option chains into call and put options
            calls = list(filter(lambda x: x.Right == OptionRight.Call, chains))
            puts = list(filter(lambda x: x.Right == OptionRight.Put, chains))

            # if lists are empty return
            if not calls or not puts: return

            underlying_price = self.Securities[self.symbol].Price
            expiries = [i.Expiry for i in puts]

            # determine expiration date nearly one month
            expiry = min(expiries, key=lambda x: abs((x.date()-self.Time.date()).days-
30))
            strikes = [i.Strike for i in puts]

            # determine at-the-money strike
            strike = min(strikes, key=lambda x: abs(x-underlying_price))

            # determine 15% out-of-the-money strike
            otm_strike = min(strikes, key = lambda x:abs(x - float(0.85) *
underlying_price))

            atm_call = [i for i in calls if i.Expiry == expiry and i.Strike == strike]
            atm_put = [i for i in puts if i.Expiry == expiry and i.Strike == strike]
            otm_put = [i for i in puts if i.Expiry == expiry and i.Strike ==
otm_strike]

            if atm_call and atm_put and otm_put:
                options_q = int(self.Portfolio.MarginRemaining / (underlying_price *
100))

                # sell at-the-money straddle
                self.Sell(atm_call[0].Symbol, options_q)
                self.Sell(atm_put[0].Symbol, options_q)

                # buy 15% out-of-the-money put
                self.Buy(otm_put[0].Symbol, options_q)

                # buy index.
                self.SetHoldings(self.symbol, 1)

        invested = [x.Key for x in self.Portfolio if x.Value.Invested]
        if len(invested) == 1:
            self.Liquidate(self.symbol)
```

BACKTESTING PERFORMANCE

Fig 1. Overall Performance

PSR	3.966%	Sharpe Ratio	0.613
Total Trades	924	Average Win	1.30%
Average Loss	-1.52%	Compounding Annual Return	10.561%
Drawdown	30.100%	Expectancy	0.173
Net Profit	276.705%	Loss Rate	37%
Win Rate	63%	Profit-Loss Ratio	0.85
Alpha	0.016	Beta	0.695
Annual Standard Deviation	0.133	Annual Variance	0.018
Information Ratio	-0.127	Tracking Error	0.096
Treynor Ratio	0.117	Total Fees	$1545.32
Estimated Strategy Capacity	$4600000.00	Lowest Capacity Asset	SPY 325YRWXKHWQQU|SPY R735QTJ8XC9X

Fig 2. Performance Metrics

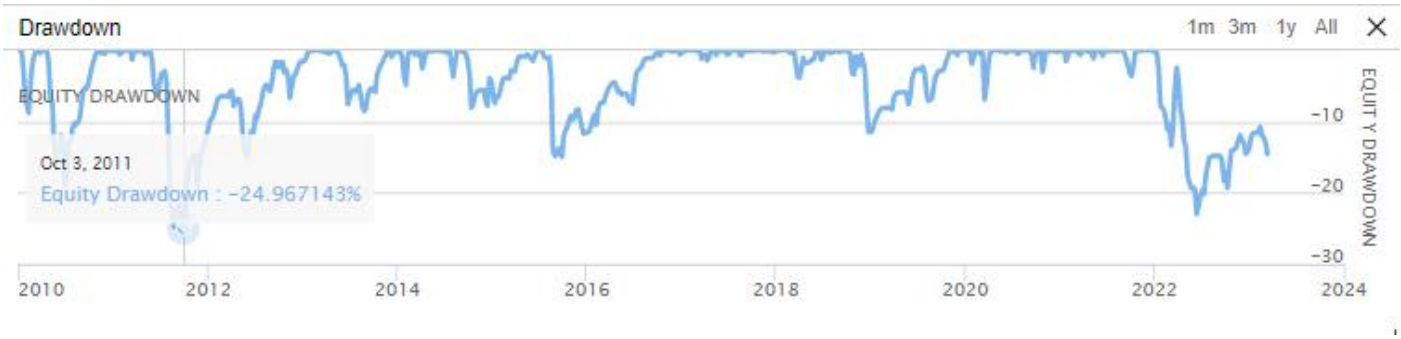

Fig 3. Drawdown