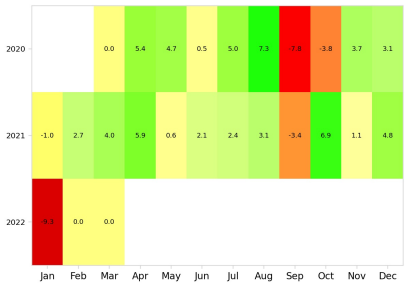
Strategy Description

Simple Breakout Trading Algorithm for Python, 2020-2022 starting at $10,000

Key Statistics

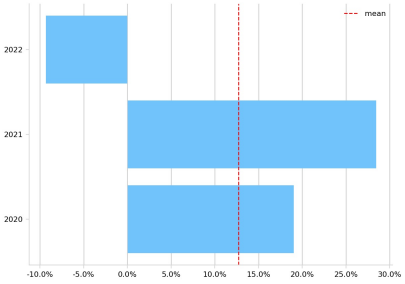
Days Live	-	Drawdown	13.4%
Turnover	1%	Probabilistic SR	48%
CAGR	17.9%	Sharpe Ratio	1.0
Markets	Equity	Information Ratio	-0.6
Trades per Day	0.0	Strategy Capacity (USD)	910M

Monthly Returns

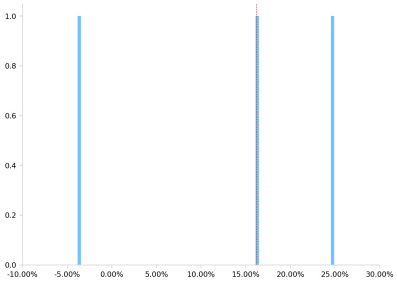


Cumulative Returns

Annual Returns



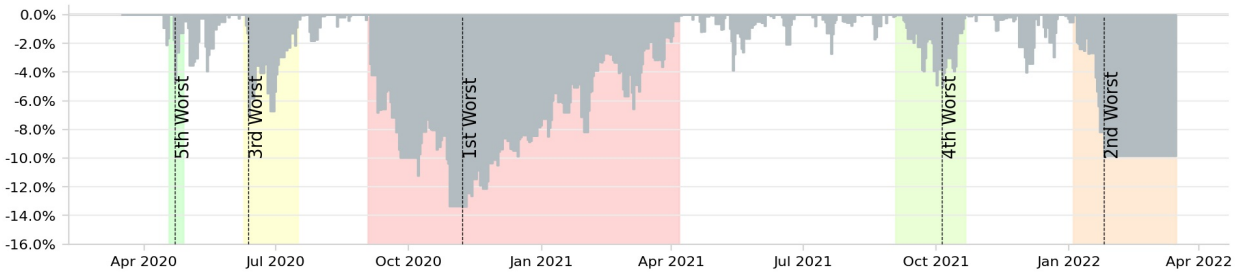
Returns Per Trade



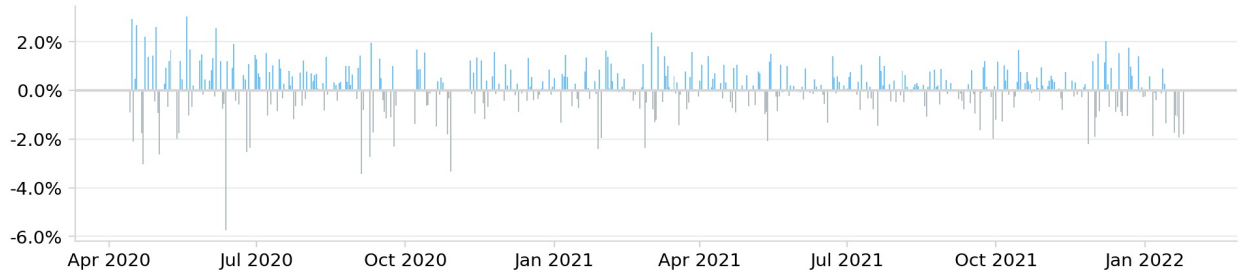
Asset Allocation

Insufficient Data

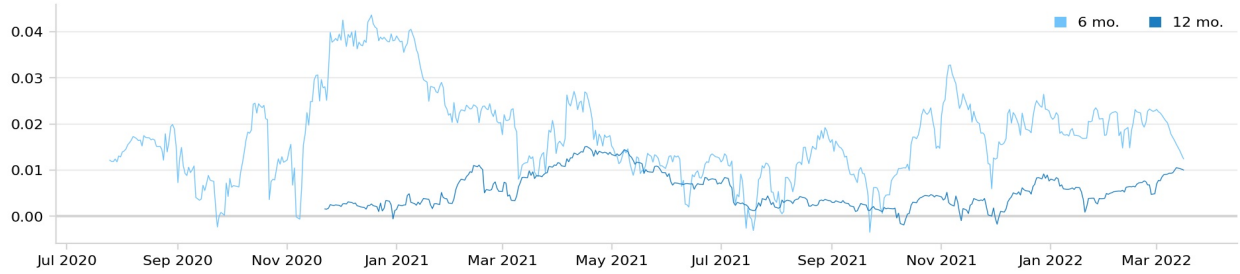
Drawdown



Daily Returns

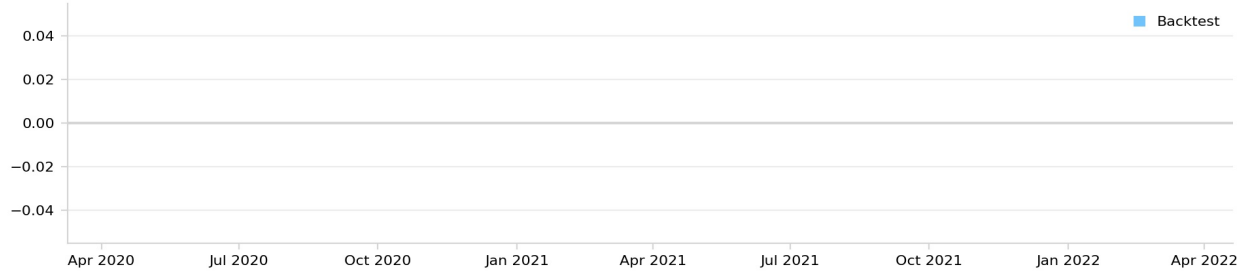


Rolling Portfolio Beta (6 Months)

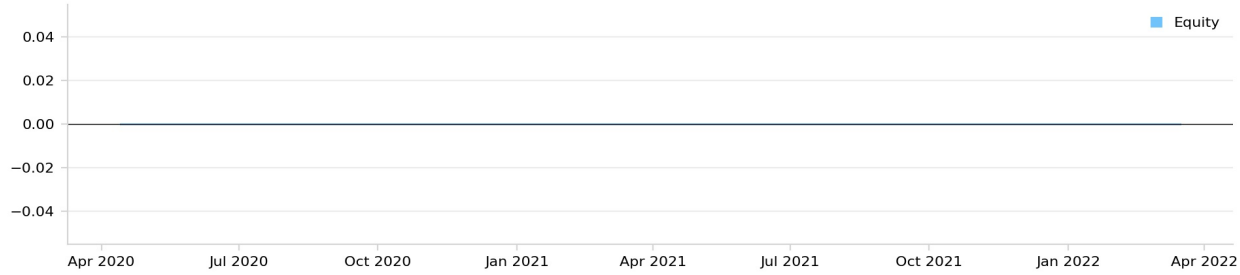


Rolling Sharpe Ratio (6 Months)

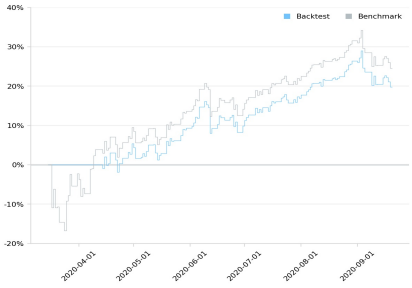
Leverage



Long-Short Exposure



COVID-19 Pandemic 2020



```
#region imports
from AlgorithmImports import *
#endregion
import numpy as np

class StockTradingBot(QCAlgorithm):

    def Initialize(self):
        # Starting cash amount
        self.SetCash(10000)

        # Start/end dates
        self.SetStartDate(2020,3,15)
        self.SetEndDate(2022,3,15)

        # Add asset
        self.symbol = self.AddEquity("SPY", Resolution.Daily).Symbol

        # Lookback length for b/o (in days)
        self.lookback = 20

        # Upper/lower limit for lookback length
        self.ceiling, self.floor = 30, 10

        # Price offset for stop order
        self.initialStopRisk = 0.98
        self.trailingStopRisk = 0.9

        # Schedule function 20 minutes after every market open
        self.Schedule.On(self.DateRules.EveryDay(self.symbol), \
                         self.TimeRules.AfterMarketOpen(self.symbol, 20), \
                         Action(self.EveryMarketOpen))


    def OnData(self, data):
        # Plot security's price
        self.Plot("Data Chart", self.symbol, self.Securities[self.symbol].Close)


    def EveryMarketOpen(self):
        # Dynamically determine lookback length based on 30 day volatility change rate
        close = self.History(self.symbol, 31, Resolution.Daily)["close"]
        todayvol = np.std(close[1:31])
        yesterdayvol = np.std(close[0:30])
        deltavol = (todayvol - yesterdayvol) / todayvol
        self.lookback = round(self.lookback * (1 + deltavol))

        # Account for upper/lower limit of lockback length
        if self.lookback > self.ceiling:
            self.lookback = self.ceiling
        elif self.lookback < self.floor:
            self.lookback = self.floor

        # List of daily highs
        self.high = self.History(self.symbol, self.lookback, Resolution.Daily)["high"]

        # Buy in case of breakout
        if not self.Securities[self.symbol].Invested and \
                self.Securities[self.symbol].Close >= max(self.high[:-1]):
            self.SetHoldings(self.symbol, 1)
            self.breakoutlvl = max(self.high[:-1])
            self.highestPrice = self.breakoutlvl


        # Create trailing stop loss if invested
        if self.Securities[self.symbol].Invested:
```

```
# If no order exists, send stop-loss
if not self.Transactions.GetOpenOrders(self.symbol):
    self.stopMarketTicket = self.StopMarketOrder(self.symbol, \
                        -self.Portfolio[self.symbol].Quantity, \
                        self.initialStopRisk * self.breakoutlvl)

# Check if the asset's price is higher than highestPrice & trailing stop price not below initial stop price
if self.Securities[self.symbol].Close > self.highestPrice and \
        self.initialStopRisk * self.breakoutlvl < self.Securities[self.symbol].Close * self.trailingStopRisk:
    # Save the new high to highestPrice
    self.highestPrice = self.Securities[self.symbol].Close
    # Update the stop price
    updateFields = UpdateOrderFields()
    updateFields.StopPrice = self.Securities[self.symbol].Close * self.trailingStopRisk
    self.stopMarketTicket.Update(updateFields)

    # Print the new stop price with Debug()
    self.Debug(updateFields.StopPrice)

# Plot trailing stop's price
self.Plot("Data Chart", "Stop Price", self.stopMarketTicket.Get(OrderField.StopPrice))
```