Member-only story

A TEMA Crossover Strategy with Volume Confirmation in Python with Backtrader

PyQuantLab Following

11 min read · Jun 8, 2025

In the world of quantitative trading, choosing the right indicators and combining them effectively is paramount. While Simple Moving Averages (SMAs) and Exponential Moving Averages (EMAs) are foundational, they often suffer from lag, causing delayed entry and exit signals. The Triple Exponential Moving Average (TEMA) is designed to address this lag, offering a more responsive trend-following indicator.

However, even the most responsive indicators can generate false signals in choppy markets. This is where volume confirmation comes into play. By validating price movements with significant trading activity, we can filter out less reliable signals and potentially improve strategy performance.

This tutorial will guide you through creating a `backtrader` strategy that implements a TEMA crossover system, reinforced with volume confirmation, and integrated with essential risk management through a stop-loss mechanism. We will use cryptocurrency data (`ETH-USD`) to demonstrate its application in a highly dynamic market.

Why TEMA and Volume Confirmation?

Triple Exponential Moving Average (TEMA)

Developed by Patrick Mulloy, TEMA aims to reduce the inherent lag of traditional moving averages. Unlike a simple EMA, TEMA applies a complex calculation involving multiple EMAs to achieve faster responsiveness without sacrificing smoothness. This makes TEMA particularly valuable for traders who need earlier identification of trend changes, which can be critical in fast-moving markets like cryptocurrencies.

The formula for TEMA is:

$$TEMA = (3 \times EMA_1) - (3 \times EMA_2) + EMA_3$$

Where:

- $EMA_1 = EMA(Price, Period)$
- $EMA_2 = EMA(EMA_1, Period)$
- $EMA_3 = EMA(EMA_2, Period)$

Volume Confirmation

Price movements are more significant when backed by substantial trading volume.

- High Volume on Breakouts/Crossovers: Suggests strong conviction behind the price move.
- Low Volume on Breakouts/Crossovers: May indicate a false signal or lack of interest, often leading to reversals.

By requiring a TEMA crossover to be confirmed by above-average volume, we aim to:

- Reduce whipsaws and false signals.
- Increase the probability of successful trades by entering only when the market shows strong commitment.

Stop-Loss

No trading strategy is foolproof. A stop-loss is a critical risk management tool that automatically closes a position if the price moves against you by a predetermined amount. This limits potential losses and protects your trading capital from significant drawdowns.

Prerequisites

To follow this tutorial, ensure you have the following Python libraries installed:

```
pip install backtrader yfinance pandas matplotlib numpy
```

Step-by-Step Implementation

We'll structure our `backtrader` strategy into distinct components for clarity and modularity.

1. Initial Setup and Data Acquisition

First, we set up our environment and download the historical data. We'll use Ethereum (ETH-USD) data for this example.

```
import backtrader as bt
import yfinance as yf
import pandas as pd
import matplotlib.pyplot as plt

# Set matplotlib style for better visualization
%matplotlib inline
plt.rcParams['figure.figsize'] = (10, 6)

# Download historical data for Ethereum (ETH-USD)
# Remember the instruction: yfinance download with auto_adjust=Fals
print("Downloading ETH-USD data from 2021-01-01 to 2024-01-01...")
data = yf.download('ETH-USD', '2021-01-01', '2024-01-01', auto_adju
data.columns = data.columns.droplevel(1) # Drop the second level of
print("Data downloaded successfully.")
print(data.head()) # Display first few rows of the data

# Create a Backtrader data feed from the pandas DataFrame
data_feed = bt.feeds.PandasData(dataname=data)
```

Explanation:

- `yfinance.download`: Fetches historical cryptocurrency price data. `auto_adjust=False` is used as per our persistent instruction to ensure raw prices.
- `data.columns = data.columns.droplevel(1)`: `yfinance` can return a multi-level column index (e.g., `('Close', 'ETH-

`USD')`). `backtrader` expects a single-level index (`Close`). This line flattens the index.

- `bt.feeds.PandasData` : Converts our cleaned pandas DataFrame into a format `backtrader` can consume.

2. The TEMA Crossover Strategy with Volume Confirmation (`TEMAStrategy`)

This is the core of our trading system. We'll define a `bt.Strategy` class that incorporates TEMAs, volume filtering, and stop-loss management.

```
class TEMAStrategy(bt.Strategy):
    # Define strategy parameters
    params = (
        ('fast_period', 7),         # Period for the fast TEMA
        ('slow_period', 30),        # Period for the slow TEMA
        ('volume_period', 7),       # Period for the volume SMA to c
        ('stop_loss_pct', 0.01),    # Percentage for the stop-loss (
    )

    def __init__(self):
        # Initialize TEMA indicators
        # bt.indicators.TEMA automatically handles the triple expon
        self.fast_tema = bt.indicators.TEMA(self.data.close, period
        self.slow_tema = bt.indicators.TEMA(self.data.close, period

        # Create a CrossOver indicator to detect when fast_tema cro
        # crossover > 0 for fast_tema crossing above slow_tema (bul
        # crossover < 0 for fast_tema crossing below slow_tema (bea
        self.crossover = bt.indicators.CrossOver(self.fast_tema, se

        # Initialize Volume confirmation
        # Calculate Simple Moving Average of Volume
        self.volume_sma = bt.indicators.SMA(self.data.volume, perio
        # Create a boolean signal: True if current volume is greate
        self.volume_signal = self.data.volume > self.volume_sma

        # Variables to keep track of active orders to prevent multi
        self.order = None       # Holds a reference to any active b
        self.stop_order = None  # Holds a reference to any active s

    def notify_order(self, order):
        # This method is called by Cerebro whenever an order's stat

        # If the order has been completed (filled)
        if order.status in [order.Completed]:
            # If it was a buy order and we now have a long position
            if order.isbuy() and self.position.size > 0:
                # Calculate the stop-loss price (e.g., 1% below ent
                stop_price = order.executed.price * (1 - self.param
                # Place a sell stop order
                self.stop_order = self.sell(exectype=bt.Order.Stop,
                self.log(f'BUY EXECUTED, Price: {order.executed.pri
            # If it was a sell order (for shorting) and we now have
            elif order.issell() and self.position.size < 0:
                # Calculate the stop-loss price (e.g., 1% above ent
                stop_price = order.executed.price * (1 + self.param
                # Place a buy stop order to cover the short
                self.stop_order = self.buy(exectype=bt.Order.Stop,
                self.log(f'SELL EXECUTED (Short), Price: {order.exe

        # If the order is completed, canceled, or rejected, clear t
        if order.status in [order.Completed, order.Canceled, order.
            self.order = None # Clear main order reference
            if order == self.stop_order: # If the completed order w
                self.stop_order = None # Clear stop-loss order refe

    def log(self, txt, dt=None):
        ''' Logging function for the strategy '''
        dt = dt or self.datas[0].datetime.date(0) # Get current dat
        print(f'{dt.isoformat()}, {txt}')

    def next(self):
        # Prevent new orders if there's already an active order pen
        if self.order is not None:
            return
        # Trading logic: TEMA crossover with volume confirmation

        # Bullish signal: Fast TEMA crosses above Slow TEMA AND cur
        if self.crossover > 0 and self.volume_signal[0]:  # [0] ref
            if self.position.size < 0:  # If currently in a short p
                # Close the short position first
                self.log(f'CLOSING SHORT POSITION (Crossover Up), P
                if self.stop_order is not None:
                    self.cancel(self.stop_order) # Cancel any activ
                self.order = self.close() # Close the short positio
            elif not self.position:  # If not in any position
                # Open a long position
                self.log(f'OPENING LONG POSITION (Crossover Up with
                self.order = self.buy() # Execute a buy order

        # Bearish signal: Fast TEMA crosses below Slow TEMA AND cur
        elif self.crossover < 0 and self.volume_signal[0]: # [0] re
            if self.position.size > 0:  # If currently in a long po
                # Close the long position first
                self.log(f'CLOSING LONG POSITION (Crossover Down),
                if self.stop_order is not None:
                    self.cancel(self.stop_order) # Cancel any activ
                self.order = self.close() # Close the long position
            elif not self.position:  # If not in any position
                # Open a short position
                self.log(f'OPENING SHORT POSITION (Crossover Down w
                self.order = self.sell() # Execute a sell order
```

Explanation of `TEMAStrategy`:

- `params`: Defines the configurable parameters for our strategy, such as TEMA periods, volume period, and stop-loss percentage.
- `__init__(self)`:

- `self.fast_tema` and `self.slow_tema`: Instances of `bt.indicators.TEMA` are created for the fast and slow TEMA lines. `backtrader` automatically handles the complex TEMA calculation.
- `self.crossover`: `bt.indicators.CrossOver` is used to detect when the `fast_tema` crosses `slow_tema`. This indicator returns a positive value (+1) on an upward crossover and a negative value (-1) on a downward crossover.
- `self.volume_sma`: Calculates a Simple Moving Average of the `self.data.volume` (the volume line of our data feed).
- `self.volume_signal`: This is a boolean line. It's `True` when the current volume is greater than its SMA, indicating above-average volume.
- `self.order` and `self.stop_order`: These variables are crucial for managing order flow. We set them to `None` when no orders are pending or active.
- `notify_order(self, order)`: This is a callback method that `backtrader` invokes whenever an order's status changes.
- When an order is `Completed` (meaning it has been filled by the broker simulation), we proceed to place a corresponding stop-loss order.
- For a `buy` order (long position), a `sell` stop order is placed below the entry price.

- For a `sell` order (short position), a `buy` stop order is placed above the entry price.
- It also clears the `self.order` and `self.stop_order` references once orders are no longer active, allowing the strategy to place new orders.
- `log(self, txt, dt=None)` : A simple utility function to print informative messages to the console with the current date.
- `next(self)` : This method contains the core trading logic and is executed for each new bar of data.
- `if self.order is not None: return` : This is a safeguard to prevent multiple orders from being sent if a previous one is still pending.
- Entry/Exit Logic:
- Long Entry/Short Exit: If `self.crossover > 0` (fast TEMA crosses above slow TEMA, indicating an uptrend) AND `self.volume_signal[0]` (current volume is above its average, confirming the signal), the strategy checks if it's currently short. If so, it closes the short position. Otherwise, if not in any position, it opens a new long position.
- Short Entry/Long Exit: If `self.crossover < 0` (fast TEMA crosses below slow TEMA, indicating a downtrend) AND `self.volume_signal[0]` (current volume is above its average), the strategy checks if it's currently long. If so, it closes the

long position. Otherwise, if not in any position, it opens a new short position.

- `self.cancel(self.stop_order)` : When closing a position due to a TEMA crossover signal, any existing stop-loss order for that position must be canceled to avoid unintended trades.

3. Running the Backtest and Analyzing Results

Finally, we set up the `backtrader` Cerebro engine, add our strategy, data, and configure broker settings. We'll also add several `backtrader.analyzers` to get detailed performance statistics.

```
# Create a Cerebro entity
cerebro = bt.Cerebro()

# Add the strategy
cerebro.addstrategy(TEMAStrategy)

# Add the data feed
cerebro.adddata(data_feed)

# Set the sizer: invest 95% of available cash on each trade
cerebro.addsizer(bt.sizers.PercentSizer, percents=95)

# Set starting cash
cerebro.broker.setcash(100000.0) # Start with $100,000

# Set commission (e.g., 0.1% per transaction)
cerebro.broker.setcommission(commission=0.001)

# --- Add Analyzers for comprehensive performance evaluation ---
cerebro.addanalyzer(bt.analyzers.SharpeRatio, _name='sharpe')
cerebro.addanalyzer(bt.analyzers.DrawDown, _name='drawdown')
cerebro.addanalyzer(bt.analyzers.Returns, _name='returns')
cerebro.addanalyzer(bt.analyzers.TradeAnalyzer, _name='tradeanalyze
cerebro.addanalyzer(bt.analyzers.SQN, _name='sqn') # System Quality
cerebro.addanalyzer(bt.analyzers.Transactions, _name='transactions'

# Print starting portfolio value
print(f'Starting Portfolio Value: ${cerebro.broker.getvalue():,.2f}

# Run the backtest
print("Running backtest...")
results = cerebro.run()
print("Backtest finished.")

# Print final portfolio value
final_value = cerebro.broker.getvalue()
print(f'Final Portfolio Value: ${final_value:,.2f}')

# --- Get and print analysis results ---
strat = results[0] # Access the strategy instance from the results

print("\n--- Strategy Performance Metrics ---")

# 1. Returns Analysis
returns_analysis = strat.analyzers.returns.get_analysis()
total_return = returns_analysis.get('rtot', 'N/A') * 100
annual_return = returns_analysis.get('rnorm100', 'N/A')
print(f"Total Return: {total_return:.2f}%")
print(f"Annualized Return: {annual_return:.2f}%")

# 2. Sharpe Ratio (Risk-adjusted return)
sharpe_ratio = strat.analyzers.sharpe.get_analysis()
print(f"Sharpe Ratio: {sharpe_ratio.get('sharperatio', 'N/A'):.2f}"

# 3. Drawdown Analysis (Measure of risk)
drawdown_analysis = strat.analyzers.drawdown.get_analysis()
max_drawdown = drawdown_analysis.get('maxdrawdown', 'N/A')
print(f"Max Drawdown: {max_drawdown:.2f}%")
print(f"Longest Drawdown Duration: {drawdown_analysis.get('maxdrawd

# 4. Trade Analysis (Details about trades)
trade_analysis = strat.analyzers.tradeanalyzer.get_analysis()
total_trades = trade_analysis.get('total', {}).get('total', 0)
won_trades = trade_analysis.get('won', {}).get('total', 0)
```

```
    lost_trades = trade_analysis.get('lost', {}).get('total', 0)
    win_rate = (won_trades / total_trades) * 100 if total_trades > 0 el
    print(f"Total Trades: {total_trades}")
    print(f"Winning Trades: {won_trades} ({win_rate:.2f}%)")
    print(f"Losing Trades: {lost_trades} ({100-win_rate:.2f}%)")
    print(f"Average Win (PnL): {trade_analysis.get('won',{}).get('pnl',
    print(f"Average Loss (PnL): {trade_analysis.get('lost',{}).get('pnl
    print(f"Ratio Avg Win/Avg Loss: {abs(trade_analysis.get('won',{}).g

    # 5. System Quality Number (SQN) - Dr. Van Tharp's measure of syste
    sqn_analysis = strat.analyzers.sqn.get_analysis()
    print(f"System Quality Number (SQN): {sqn_analysis.get('sqn', 'N/A'


    # --- Plot the results ---
    print("\nPlotting results...")
    # iplot=False for static plot, style='candlestick' for candlestick
    # plotreturn=True to show the equity curve in a separate subplot
    cerebro.plot(iplot=False, style='candlestick',
                 barup=dict(fill=False, lw=1.0, ls='-', color='green'),
                 bardown=dict(fill=False, lw=1.0, ls='-', color='red'),
                 plotreturn=True, # Show equity curve
                 numfigs=1 # Ensure only one figure is generated
                )
    print("Plot generated.")
```

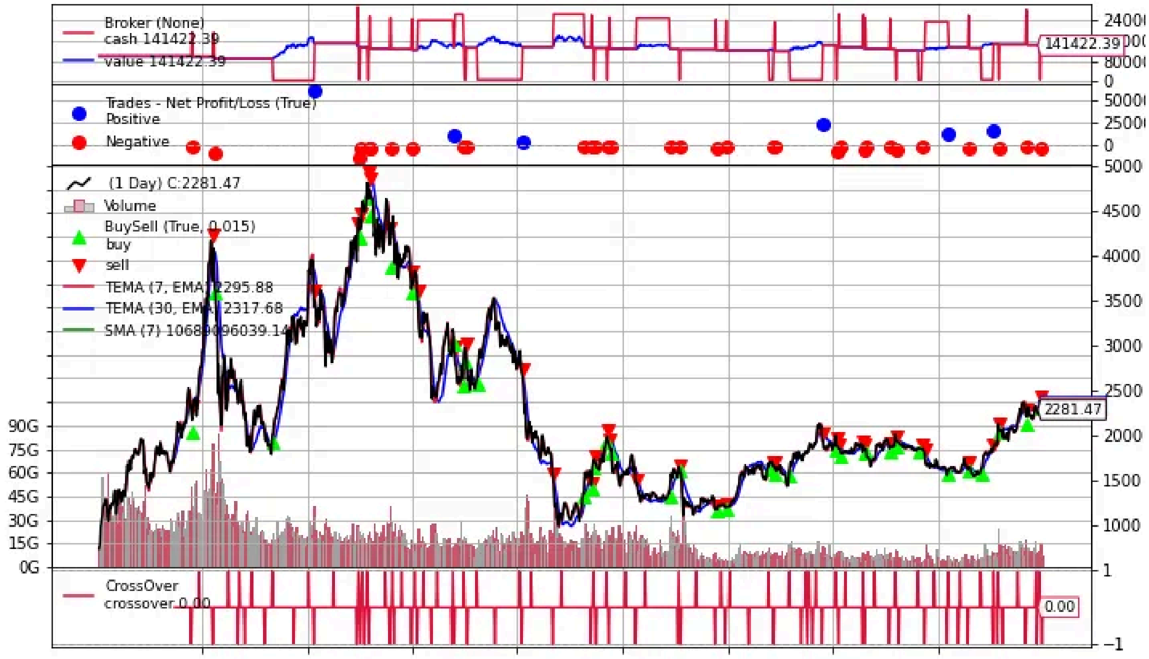
Explanation of Backtesting Setup:

- `bt.Cerebro()` : The central engine that orchestrates the backtest.
- `cerebro.addstrategy(TEMAStrategy)` : Registers our custom strategy with Cerebro.
- `cerebro.adddata(data_feed)` : Feeds the historical data into the backtesting engine.

- `cerebro.addsizer(bt.sizers.PercentSizer, percents=95)` : This position sizer ensures that 95% of the available cash is used for each trade, preventing over-allocation or leaving too much cash idle.
- `cerebro.broker.setcash(100000.0)` : Sets the initial trading capital.
- `cerebro.broker.setcommission(commission=0.001)` : Applies a commission of 0.1% on each trade, making the backtest more realistic.
- `cerebro.addanalyzer(...)` : These lines are crucial for evaluating the strategy's performance beyond just total return. They add various `backtrader` analysis modules:
- `SharpeRatio` : Measures risk-adjusted return.
- `DrawDown` : Calculates maximum drawdown and duration.
- `Returns` : Provides total and annualized returns.
- `TradeAnalyzer` : Offers detailed statistics on individual trades (wins, losses, average profit/loss).
- `SQN` : System Quality Number, a measure of strategy robustness.
- `Transactions` : Logs all trade transactions for review.
- `cerebro.run()` : Executes the backtest.

- Result Printing: The code then retrieves the analysis results from the `results` object (which holds the executed strategies) and prints them in a structured way, offering deep insights into the strategy's profitability, risk, and trade characteristics.

- `cerebro.plot(iplot=False)` : Generates a visual plot of the backtest. This plot is invaluable for visually inspecting trade entries, exits, and how the indicators behaved relative to price action. `iplot=False` ensures a static plot suitable for non-interactive environments.



Further Enhancements and Considerations

While this strategy is robust, here are some ideas for future improvements:

1. Refined Volume Confirmation: The current volume check is simple (`current_volume > SMA(volume)`). You could explore:

- Requiring `current_volume` to be a certain *multiple* of `volume_sma`.
- Differentiating volume by price direction (e.g., high volume on up-candles for buys, high volume on down-candles for sells).
- Using other volume indicators like On-Balance Volume (OBV) or Volume Price Trend (VPT).

1. Adaptive Periods for TEMAs: Just as in the previous example, you could make the `fast_period` and `slow_period` of the TEMAs adaptive to market volatility. This would require creating a custom `AdaptiveTEMA` indicator similar to the `AdaptiveSMA` you developed.
2. Dynamic Stop-Loss: Instead of a fixed percentage, implement a dynamic stop-loss based on volatility (e.g., a multiple of ATR) or trailing stops to lock in profits.
3. Take-Profit Targets: Add specific profit targets to exit positions once a certain gain is achieved, preventing profit erosion if the market reverses.
4. Market Regime Filtering: TEMA crossover strategies often perform best in trending markets. Consider adding an

additional filter (e.g., ADX indicator, or a higher timeframe trend filter) to avoid trading in choppy, non-trending markets.

5. Slippage Simulation: For higher realism, especially with crypto assets, enable slippage in `backtrader` to account for the difference between expected and executed trade prices.
6. Parameter Optimization: Use `backtrader`'s `optstrategy` feature to systematically test different combinations of `fast_period`, `slow_period`, `volume_period`, and `stop_loss_pct` to find the most robust and profitable settings for `ETH-USD` or other assets.
7. Timeframe Analysis: Test the strategy on different timeframes (e.g., hourly, weekly data) to see how its performance varies.

Conclusion

You have successfully built and backtested a TEMA Crossover strategy with Volume Confirmation and stop-loss using `backtrader`. This strategy represents a significant step beyond basic moving average systems by incorporating a more responsive indicator and a crucial market sentiment filter. By leveraging the power of `backtrader` and its robust analysis tools, you can thoroughly evaluate and refine your trading ideas, bringing you closer to developing profitable algorithmic strategies. Remember that backtesting is a continuous process of experimentation, learning, and adaptation.

Algorithmic Trading Python Quantitative Finance Backtesting Moving Average Strategy

Written by PyQuantLab

655 followers · 6 following

Following

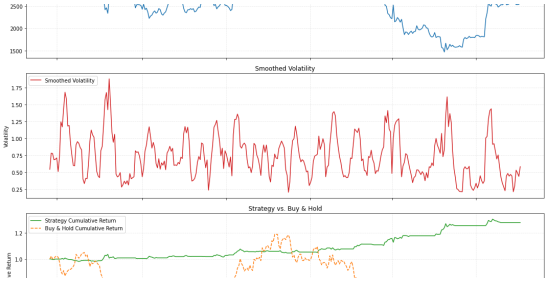
Your go-to place for Python-based quant tutorials, strategy deep-dives, and reproducible code. For more visit our website: www.pyquantlab.com

No responses yet

Steven Feng CAI

What are your thoughts?

More from PyQuantLab



PyQuantLab

Volatility Clustering Trading Strategy with Python

Ultimate Algorithmic Strategy Bundle has you covered with over 80 Python…

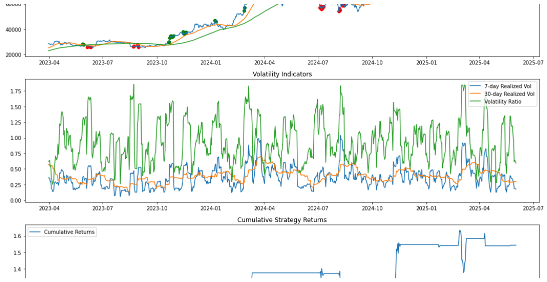
✦ Jun 3 👏 32 💬 3

PyQuantLab

An Algorithmic Exploration of Volume Spread Analysis…

📢 Note: You can read all articles for free on our website: pyquantlab.com

✦ Jun 9 👏 54 💬 1



PyQuantLab

Trend-Volatility Confluence Trading Strategy

Trades - Net Profit/Loss (True)
Positive
Negative
(1 Day) O:3349.59 H:3428.53 L:3298.80 C:3356.39
Volume
BuySell (True, 0.015)
buy
sell
ExponentialMovingAverage (7) 3387.10
ExponentialMovingAverage (30) 3500.92
ATR (30)

PyQuantLab

Building an Adaptive Trading Strategy with Backtrader: A…

The Ultimate Algorithmic Strategy Bundle has you covered with over 80…

✦ Jun 3 👏 60

📢 Note: You can read all articles for free on our website: pyquantlab.com

✦ Jun 4 👏 64

See all from PyQuantLab

Recommended from Medium

Swapnilphutane

How I Built a Multi-Market Trading Strategy That Pass…

When I first got into trading, I had no plans of building a full-blown system.…

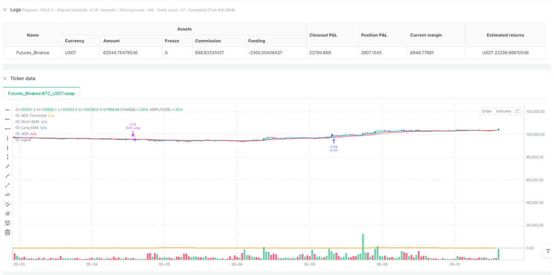
6d ago 👏 16 💬 1

Candence

Exposing Bernd Skorupinski Strategy: How I Profited ove…

I executed a single Nikkei Futures trade that banked $16,400 with a…

Jun 19 👏 2



Logs Progress : 100.0 % - Elapsed (second) : 6.141 Seconds - Total log count : 146 - Trade count : 57 - Completed (Tick 916.28KB)

Name	Assets					Closeout P&L	Position P&L	Current margin	Estimated returns
	Currency	Amount	Freeze	Commission	Funding				
Futures_Binance	USDT	62544.75479536	0	998.83331037	-2300.50408427	22790.869	2807.1545	6946.77681	USDT 22298.68610536

Ticker data

Futures_Binance.BTC_USDT.swap

O=103551.2 H=105000.1 L=103523.4 C=104788.8 V=21988.68 CHANGE=1.20% AMPLITUDE=1.43%

ADX Threshold n/a

Short EMA n/a

Long EMA n/a

ADX n/a

signal

FMZQuant

Multi-Timeframe Dynamic Trend Detection System: EM…

Strategy Overview

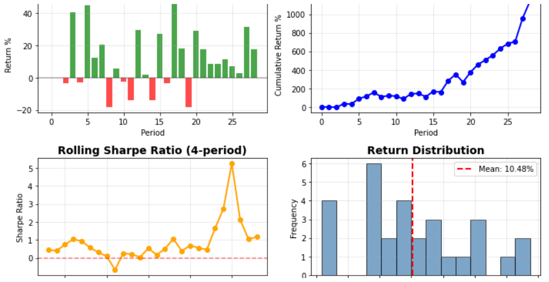
Jun 4 👏 1

MarketMuse

“I Let an AI Bot Trade for Me for 7 Days—It Made $8,000…

Subtitle: While you’re analyzing candlestick patterns, AI bots are fron…

✦ Jun 3 👏 75 💬 3

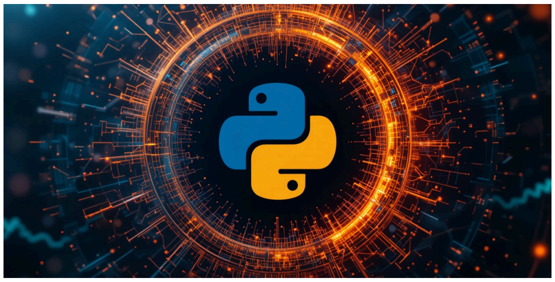


PyQuantLab

Rolling Backtest of a Multi-Timeframe Pivot Point…

📢 Note: You can read all articles for free on our website: pyquantlab.com

✦ Jun 22



Unicorn Day

The Quest for the Perfect Trading Score: Turning…

Navigating the financial markets… it feels like being hit by a tsunami of da…

✦ 3d ago 👏 43

See more recommendations