Member-only story

Linear Regression Forecast (LINEARREG) / Time Series Forecast (TSF): Riding the Trendline

PyQuantLab Following 5 min read · Jun 4, 2025

The Linear Regression Forecast (LINEARREG) and Time Series Forecast (TSF) are two functions in technical analysis that, within the TA-Lib library, are identical. They provide a dynamic, responsive measure of the underlying short-term trend in a price series.

Want to explore a wider range of indicators and build robust trading strategies? “The Complete Technical Analysis Guide” is

your comprehensive resource, covering:

- Essential Candlestick Patterns, a full suite of Moving Averages (SMA, EMA, TEMA, KAMA), and powerful Oscillators (RSI, MACD, Stochastics) (Chapters 2–4).
- In-depth analysis of Trend Strength (ADX, Aroon), Volume dynamics (OBV, MFI), and Market Volatility (ATR, Bollinger Bands) (Chapters 5–7).
- Practical guides to Strategy Development & Backtesting with Python, TA-Lib, and Backtrader, featuring detailed examples of Trend-Following, Mean-Reversion, and Breakout strategies (Chapters 9–12).
- An introduction to Advanced Statistical and Machine Learning approaches in trading (Chapter 13).

Master technical analysis from A to Z. Find out more and get your copy:

Theory:
At its core, the LINEARREG (or TSF) indicator calculates the endpoint of a linear regression line that has been statistically fitted to a set of input price data (typically closing prices) over a specified number of preceding periods (N).

Imagine taking the last N price points. A straight line (the linear regression line, or "line of best fit") is mathematically calculated to pass as closely as possible through these N points. The LINEARREG/TSF value is then the point on this calculated line that corresponds to the *end of the current period*. In essence, it answers the question: "If the linear trend observed over the past N periods were to continue perfectly, what would the price be at the close of this current period?"

Because it's re-calculated at each new period using a rolling window of N past periods, the LINEARREG/TSF line adapts to new price information quickly.

Usage:
The LINEARREG/TSF line is plotted directly on the price chart and serves several purposes for traders:

- Highly Responsive Trendline: It acts like a moving average but tends to be more responsive to recent price changes and can exhibit less lag than traditional SMAs or EMAs, especially

relative to the end of its calculation window. It effectively hugs the price action more closely.

- Short-Term Trend Identification:
- When the LINEARREG/TSF line is sloping upwards, it suggests a short-term uptrend.
- When it is sloping downwards, it indicates a short-term downtrend.
- A flattening line can suggest a weakening trend or consolidation.
- Signal Generation: Crossovers between the price and the LINEARREG/TSF line can be used to generate trading signals:
- Buy Signal: Price crossing above the LINEARREG/TSF line.
- Sell Signal: Price crossing below the LINEARREG/TSF line.
- Zero-Lag (Relative to Calculation End): Because it forecasts the value at the end of the *current* period based on data *up to and including* the current period (for its N-period window), it can be considered to have zero lag relative to the point it's forecasting for.
- Whipsaw Potential: Due to its high responsiveness, the LINEARREG/TSF line can be prone to generating frequent, and sometimes false, signals (whipsaws) in choppy, non-trending, or sideways markets. It generally performs better in clearly trending markets.

TA-Lib Functions:
TA-Lib provides two identical functions for this calculation:

- `talib.LINEARREG(close_prices, timeperiod=N)`
- `talib.TSF(close_prices, timeperiod=N)`

Both functions will yield the same output given the same input data and `timeperiod`.

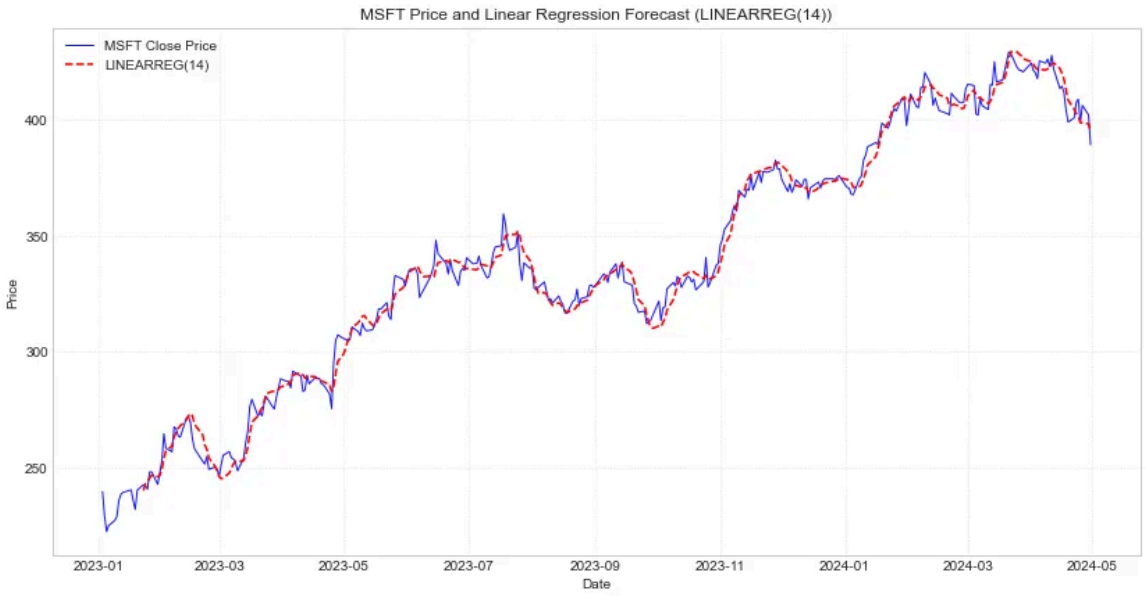
Code Example (Calculation & Plot using LINEARREG):

The following Python code demonstrates how to fetch stock data using `yfinance`, calculate the Linear Regression Forecast, and plot it overlapping the price chart.

```
import yfinance as yf
import talib
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# --- 1. Data Fetching using yfinance ---
ticker_symbol = "MSFT"  # Example: Microsoft Corp.
data_start_date = "2023-01-01"
data_end_date = "2024-05-01"

try:
    # Complying with user preference for yfinance download
    data = yf.download(ticker_symbol, start=data_start_date, end=da
    if data.empty:
        raise ValueError("No data downloaded")
```

```
    # Complying with user preference for droplevel
    if isinstance(data.columns, pd.MultiIndex) and data.columns.nle
        data.columns = data.columns.droplevel(level=1)
except Exception as e:
    print(f"Error downloading data for {ticker_symbol}: {e}")
    data = pd.DataFrame()

if data.empty or 'Close' not in data.columns or data['Close'].isnul
    print(f"Insufficient Close price data found for {ticker_symbol}
else:
    # For LINEARREG, we primarily need close prices.
    # If OHLC data is needed for other context or plotting style, i
    close_prices = data['Close'].dropna()
    date_index = close_prices.index

    # --- 2. Linear Regression Forecast (LINEARREG) Calculation ---
    time_period_linreg = 14    # Example period

    # LINEARREG needs a minimum number of data points equal to the
    # TA-Lib will output NaNs for the initial (timeperiod - 1) entr
    if len(close_prices) >= time_period_linreg:
        linreg_values = talib.LINEARREG(
            close_prices,
            timeperiod=time_period_linreg
        )

        indicator_name = f"LINEARREG({time_period_linreg})"
        print(f"\n--- Linear Regression Forecast ({indicator_name})

        valid_linreg = linreg_values[~np.isnan(linreg_values)]
        if len(valid_linreg) >= 5:
            print(f"Output {indicator_name} (last 5 valid): {valid_
        elif len(valid_linreg) > 0:
            print(f"Output {indicator_name} (all valid): {valid_lin
        else:
            print(f"Output {indicator_name}: No valid values calcul

        # --- 3. Plotting ---
        plt.figure(figsize=(14, 7))
        plt.plot(date_index, close_prices, label=f'{ticker_symbol}
        plt.plot(date_index, linreg_values, label=indicator_name, c

        plt.title(f'{ticker_symbol} Price and Linear Regression For
        plt.xlabel('Date')
```

```
            plt.ylabel('Price')
            plt.legend()
            plt.grid(True, linestyle=':', alpha=0.5)
            plt.show()

        else:
            print(f"\nSkipping {indicator_name} plot for {ticker_symbol
            if not close_prices.empty:
                 print(f"Available data points after cleaning: {len(clo
```



Explanation of the Code:

1. Import Libraries: Standard libraries are imported.
2. Data Fetching:

- `yf.download()` retrieves historical data. Your preferences for `auto_adjust=False` and `droplevel` are applied. Error handling

for the download is included.

- `close_prices` are extracted and `NaN` values dropped.

3. Linear Regression Forecast Calculation:

- `time_period_linreg` (e.g., 14) is set.
- A check `if len(close_prices) >= time_period_linreg:` ensures enough data. TA-Lib's `LINEARREG` will produce `NaN`s for the first `timeperiod - 1` values.
- `talib.LINEARREG()` calculates the forecast values.
- The last few valid forecast values are printed.

4. Plotting:

- The closing price is plotted.
- The `LINEARREG` values are plotted on the same axes, as it's an overlap study, typically shown as a line following the price.
- Standard plot elements are added.
- `plt.show()` displays the chart.

5. Insufficient Data Handling: A message is shown if data is insufficient.

The Linear Regression Forecast / Time Series Forecast offers a statistically grounded, responsive way to track short-term trends. While it can be susceptible to whipsaws in sideways markets, its ability to quickly adapt makes it a useful component in a trader's toolkit, especially when combined with other forms of analysis.

Algorithmic Trading | Quantitative Finance | Linear Regression | Python | Technical Analysis

Written by PyQuantLab

655 followers · 6 following

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

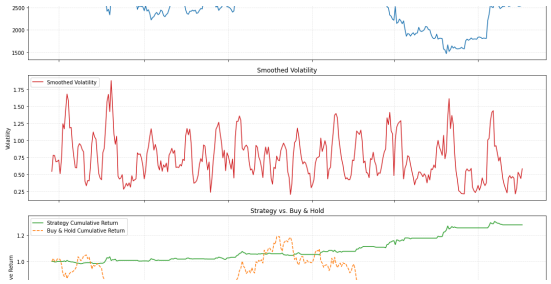
Following

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...

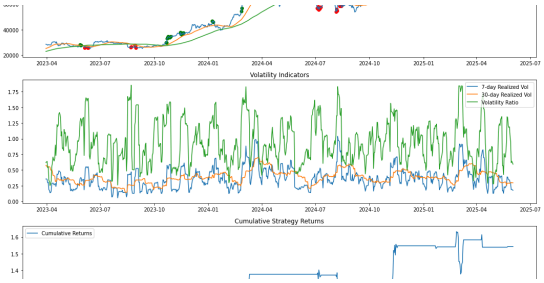
Jun 3 32 3

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

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

Jun 3 60

PyQuantLab

Building an Adaptive Trading Strategy with Backtrader: A...

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....

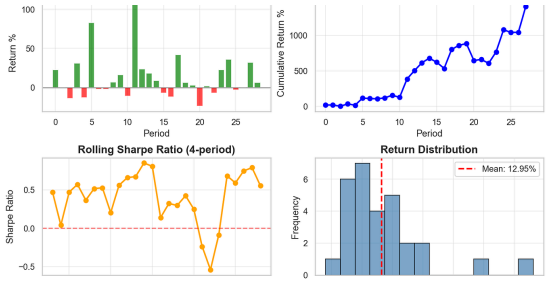
6d ago 16 1

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

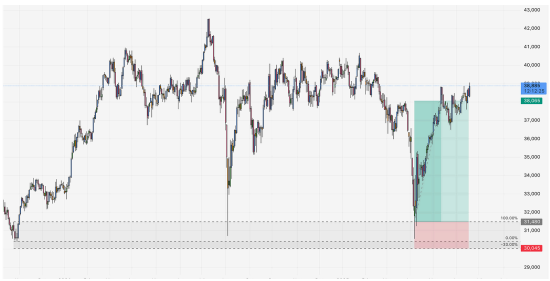


PyQuantLab

Keltner Channels and ADX Trend-Following Trading...

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

Jun 20 4



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

Unicorn Day

Why Buy-and-Hold Is Your Portfolio's Silent Killer Durin...

You're sitting in your favorite coffee shop , scrolling through financial...

Jun 23 6

Navnoor Bawa

QOADRS Mathematical Implementation: A Step-by-...

How we achieved 90.7% accuracy processing 9,462 real options...

Jun 13 2

See more recommendations