Backtesting is arguably the most critical part of the Systematic Trading Strategy (STS) production process, sitting between strategy development and deployment (live trading). If a strategy is flawed, rigorous backtesting will hopefully expose this, preventing a loss-making strategy from being deployed. **Backtesting** uses historic data to quantify STS performance. **Simulated/live trading** deploys a tested STS in real time: signaling trades, generating orders, routing orders to brokers, then maintaining positions as orders are executed.

Most frameworks go beyond backtesting to include some live trading capabilities.

The Components of a Backtesting Framework

**Data and STS acquisition:** The acquisition components consume the STS script/definition file and provide the requisite data for testing. If the framework requires any STS to be recoded before backtesting, then the framework should support canned functions for the most popular technical indicators to speed STS testing. Users determine how long of a historical period to backtest based on what the framework provides, or what they are capable of importing.

**Performance testing** applies the STS logic to the requested historic data window and calculates a broad range of risk & performance metrics, including max drawdown, Sharpe & Sortino ratios. Most all of the frameworks support a decent number of visualization capabilities, including equity curves and deciled-statistics.

**Optimization** tends to require the lion’s share of computing resources in the STS process. If your STS require optimization, then focus on a framework that supports scalable distributed/parallel processing.

In the context of strategies developed using **technical indicators**, system developers attempt to find an optimal set of parameters for each indicator. Most simply, optimization might find that a 6 and 10 day moving average crossover STS accumulated more profit over the historic test data than any other combination of time periods between 1 and 20. Already with this trivial example, 20 \* 20 = 400 parameter combinations must be calculated & ranked.

In a **portfolio context**, optimization seeks to find the optimal weighting of every asset in the portfolio, including shorted and leveraged instruments. On a periodic basis, the portfolio is rebalanced, resulting in the purchase and sale of portfolio holdings as required to align with the optimized weights.

**Position sizing** is an additional use of optimization, helping system developers simulate and analyze the impact of leverage and dynamic position sizing on STS and portfolio performance.

Standard capabilities of open source Python backtesting platforms seem to include:

* Event driven
* Very flexible, unrestrictive licensing
* Decent collection of pre-defined technical indicators
* Standard performance metric calculation/visualization/reporting capabilities

### PyAlgoTrade, Zipline, Backtrader, QSTrader, Backtesting.py, pysystemtrade

### Backtrader

This platform is exceptionally well documented, with an accompanying blog and an active on-line community for posting questions and feature requests. Backtrader supports a number of data formats, including CSV files, Pandas DataFrames, blaze iterators and real time data feeds from three brokers. These data feeds can be accessed simultaneously, and can even represent different timeframes. Supported brokers include Oanda for FX trading and multi-asset class trading via Interactive Brokers and Visual Chart.

* Project Page: [www.backtrader.com](https://www.backtrader.com/)
* Github: [github.com/mementum/backtrader](https://github.com/mementum/backtrader)
* License: [GPL v3.0](https://github.com/mementum/backtrader/blob/master/LICENSE)

### Data Feeds (Live Too)

* Built-in support for several sources: CSV, Database-Sources, YahooFinance, Interactive Brokers, Oanda v1, ...
* Any number of simultaneous data feeds (memory constrained, obviously) can be run simultaneously
* Multiple timeframes can be mixed and run
* Integrated Resampling and Replaying

### Event and Vectorized

* The trading logic and the broker are always run on an event by event basis
* The calculation for indicators is vectorized if possible (source data can be preloaded)

<https://ntguardian.wordpress.com/2017/06/12/getting-started-with-backtrader/>