Backtesting.py

Backtest trading strategies in Python

Does it seem like you had missed getting rich during the recent crypto craze? Despair not—the international financial markets continue their move rightwards every day. You still have your chance. But successful traders all agree emotions have no place in trading — if you are ever to enjoy a fortune attained by trading, better first make sure your strategy or system is well-tested and working reliably toward consistent profits. Mechanical or algorithmic trading, they call it. They'll usually recommend signing up with a broker and trading on a demo account for a few months … But you know better. You know some programming.

> It is far better to foresee even without certainty than not to foresee at all.
>
> — *Henri Poincare*

Backtesting.py is a Python framework for inferring viability of trading strategies on historical (past) data. Of course, past performance is not indicative of future results, but a strategy that proves itself resilient in a multitude of market conditions can, with a little luck, remain just as reliable in the future. Improved upon the vision of Backtrader, and by all means surpassingly comparable to other accessible alternatives, Backtesting.py is lightweight, fast, user-friendly, intuitive, interactive, intelligent and, hopefully, future-proof. It is also documented well, including a handful of tutorials.

Compatible with forex, crypto, stocks, futures ...

Backtest any financial instrument for which you have access to historical candlestick data.

Blazing fast, convenient

Built on top of cutting-edge ecosystem libraries (i.e. Pandas, NumPy, Bokeh) for maximum speed and ergonomics.

Small, clean API

The API reference is easy to wrap your head around and fits on a single page.

Technical indicator library agnostic

Compatible with any sensible technical analysis library, such as TA-Lib, Tulip, pandas-ta ...

Built-in SAMBO optimizer

Test hundreds of strategy variants in mere seconds, resulting in heatmaps you can interpret at a glance.

High-level API

Think market timing, swing trading, money management, stop-loss and take-profit prices, leverage, machine learning ...

Interactive visualization

Simulated trading results in telling interactive charts you can zoom into. See *Example*.

Vectorized or event-based backtesting

Signal-driven or streaming, model your strategy enjoying the flexibility of both approaches.

Composable strategies

Contains a library of predefined utilities and general-purpose strategies that are made to stack.

Download

PyPI GitHub Docs

■ Backtesting.py works with Python 3. You need to know some Python to effectively use this software.

Example

The example shows a simple, unoptimized **moving average cross-over** strategy. It's a common introductory strategy and a pretty decent strategy overall, provided the market isn't whipsawing sideways.

We begin with **10,000 units of currency** in cash, realistic **0.2% broker commission**, and we trade through **9 years** worth of **Alphabet Inc.** stock.

Whenever the fast, 10-period simple moving average of closing prices crosses **above** the slower, 20-period moving average, we go **long**, buying as many stocks as we can afford. When it crosses **below**, we close our long position and go **short** (assuming the underlying instrument is actually a CFD and can be shorted).

We record most significant statistics this simple system produces on our data, and we show a plot for further manual inspection.

```
from backtesting import Backtest, Strategy from backtesting.lib import crossover from backtesting.test
import SMA, GOOG class SmaCross(Strategy): n1 = 10 n2 = 20 def init(self): close = self.data.Close
self.sma1 = self.I(SMA, close, self.n1) self.sma2 = self.I(SMA, close, self.n2) def next(self): if
crossover(self.sma1, self.sma2): self.position.close() self.buy() elif crossover(self.sma2, self.sma1):
self.position.close() self.sell() bt = Backtest(GOOG, SmaCross, cash=10000, commission=.002,
exclusive_orders=True) output = bt.run() bt.plot()
```

```
Start 2004-08-19 00:00:00 End 2013-03-01 00:00:00 Duration 3116 days 00:00:00 Exposure Time [%] 94.27374
Equity Final [$] 81812.37 Equity Peak [$] 81879.03 Return [%] 718.1237 Buy & Hold Return [%] 607.37036
Return (Ann.) [%] 27.96479 Volatility (Ann.) [%] 39.08925 CAGR [%] 18.52838 Sharpe Ratio 0.71541 Sortino
Ratio 1.43503 Calmar Ratio 0.85275 Max. Drawdown [%] -32.79366 Avg. Drawdown [%] -5.24919 Max. Drawdown
Duration 680 days 00:00:00 Avg. Drawdown Duration 38 days 00:00:00 # Trades 93 Win Rate [%] 54.83871
Best Trade [%] 57.43355 Worst Trade [%] -16.39664 Avg. Trade [%] 2.16381 Max. Trade Duration 121 days
00:00:00 Avg. Trade Duration 32 days 00:00:00 Profit Factor 2.27119 Expectancy [%] 2.69468 SQN 1.94546
Kelly Criterion 0.25935 _strategy SmaCross(n1=10, n2=20)
```

Find **better examples**, including executable Jupyter notebooks, in the project documentation.

Documentation

What Users are Saying

The proof of [this] program's value is its existence.

Alan Perlis

Some things are so unexpected that no one is prepared for them.

Leo Rosten

[...] When all else fails, read the instructions.

Cahn

The financial markets generally are unpredictable. So that one has to have different scenarios … The idea that you can actually predict what's going to happen contradicts my way of looking at the market.

George Soros

If you don’t find a way to make money while you sleep, you will work until you die.

Warren Buffet

A good forecaster is not smarter than everyone else, he merely has his ignorance better organised.