


Roman Paolucci [Follow](#)

Aug 18, 2020 · 10 min read · Member-only



A Quick and Easy way to Build and Test Stock Trading Strategies

Backtesting, Data, Metrics, Live Implementation, and Pitfalls



Photo by [Pixabay](#) from [Pexels](#)

Note from Towards Data Science's editors: While we allow independent authors to publish articles in accordance with our [rules and guidelines](#), we do not endorse each author's contribution. You should not rely on an author's works without seeking professional advice. See our [Reader Terms](#) for details.

...

Algorithmic Trading Strategy Development

Backtesting is the hallmark of quantitative trading. Backtesting takes historical or synthetic market data and tests the profitability of algorithmic trading strategies. This topic demands expertise in statistics, computer science, mathematics, finance, and economics. This is exactly why in large quantitative trading firms there are specific roles for individuals with immense knowledge (usually at the doctorate level) of the respective subjects. The necessity for expertise cannot be understated as it separates winning (or seemingly winning) trading strategies from losers. My goal for this article is to break up what I consider an elementary backtesting process into a few different sections...

- I. The Backtesting Engine
- II. Historical and Synthetic Data Management
- III. Backtesting Metrics
- IV. Live Implementation



The main backtesting engine will be built in Python using a library called [backtrader](#). Backtrader makes it incredibly easy to build trading strategies and implement them on historical data immediately. The entire library centers around the Cerebro class. As the name implies, you can think of this as the brain or engine of the backtest.

I decided to build a function so I can parameterize different aspects of the backtest on the fly (and eventually build a pipeline). The next backtrader class we need to discuss is the Strategy class. If you look at the function I built to house the instance of Cerebro you'll see that it takes a strategy as an input — this is expecting a backtrader Strategy class or subclass. Subclasses of the Strategy class are exactly what we will be using to build our own strategies. If you need a polymorphism & inheritance refresher or don't know what that means see my article [3 Python Concepts in under 3 minutes](#). Let's build a strategy for our backtest...



In the MainStrategy, Strategy subclass above I create fields that we are interested in knowing about before making a trading decision. Assuming a daily data frequency the fields we have access to every step in time during the backstep are...

- Day price open
- Day price close
- Day price low
- Day price high
- Day volume

It doesn't take a genius to understand that you cannot use the high, low, and close to make an initial or intraday trading decision as we don't have access to that information in realtime. However, it is useful if you want to store it and access that information from previous days.

The big question in your head right now should be *where is the data coming from?*

The answer to that question is in the structure of the backtrader library. Before we run the function housing Cerebro we will add all the strategies we want to backtest, and a data feed — the rest is all taken care of as the Strategy superclass holds the datas series housing all the market data.

This makes our lives extremely easy.

Let's finish the MainStrategy class by making a simple long-only mean reversion style trading strategy. To access a tick of data we override the next function and add trading logic...



Every day gets a z-score calculated for it (number of standard deviations from the mean) based on its magnitude we will buy or sell equity. Please note, yes I am using the day's close to make the decision, but I'm also using the day's close as an entry price to my trade — it would be wiser to use the next days open.

The next step is to add the strategy to our function housing the Cerebro instance...

In the code above I add the strategy provided to the function to the Cerebro instance. This is beyond the scope of the article but I felt pressed to include it. If the strategy that we are backtesting requires additional data (some AI model) we can provide it in the args parameter and add it to the Strategy subclass.

Next, we need to find a source of historical or synthetic data.

Historical and Synthetic Data Management

I use historical and synthetic market data synonymously as some researchers have found synthetic data indistinguishable (and therefore is more abundant as its generative) from other market data. For the purposes of this example we will be using the YahooFinance data feed provided by backtrader, again making implementation incredibly easy...



It's almost ridiculous how easy it is to add a data feed to our backtest. The data object uses backtraders YahooFinanceData class to retrieve data based on the symbol, fromdate, and todate. Afterwards, we simply add that data to the Cerebro instance. The true beauty in backtrader's architecture comes in realizing that the data is automatically placed within, and iterated through every strategy added to the Cerebro instance.

Backtesting Metrics

There are a variety of metrics used to evaluate the performance of a trading strategy, we'll cover a few in this section. Again, backtrader makes it extremely easy to add *analyzers* or performance metrics. First, let's set an initial cash value for our Cerebro instance...

[Sign In](#)[Get started](#)

Sharpe Ratio

As referenced in the show Billions, the amount of return per unit of risk, the Sharpe Ratio. Defined as follows...

Several downfalls, and assumptions. It's based on historical performance, assumes normal distributions, and can often be used to incorrectly compare portfolios. However, it's still a staple to any strategy or portfolio.

System Quality Number

An interesting metric I've always enjoyed using as it includes volume of transactions in a given period. Computed by multiplying the SQN by the number of trades and dividing the product by 100. Below is a chart of values across different segments, regions, and sectors.

Photo from [VanTharp](#)

Let's now add these metrics to Cerebro and run the backtest...





Now, all we have to do is run the code above and we get the performance of our strategy based on the metrics we specified...

Here is the output...

```
Starting Portfolio Value: 100000.00

2018-01-30, 160.95
2018-01-30, BUY CREATE, 160.95

2018-01-31, 161.4
2018-02-01, 161.74
2018-02-02, 154.72
2018-02-02, BUY CREATE, 154.72
2018-02-05, 150.85
2018-02-05, BUY CREATE, 150.85
2018-02-06, 157.16
2018-02-07, 153.79

2018-02-08, 149.56
2018-02-09, 151.39
2018-02-12, 157.49
2018-02-13, 159.07
2018-02-14, 162.0
2018-02-15, 167.44
2018-02-16, 166.9
2018-02-20, 166.33

2018-02-21, 165.58
2018-02-22, 166.96
2018-02-23, 169.87
```



2018-03-01, 169.38
2018-03-02, 170.55
2018-03-05, 171.15
2018-03-06, 171.0
2018-03-07, 169.41
2018-03-08, 171.26
2018-03-09, 174.2
2018-03-12, 175.89
2018-03-13, 174.19
2018-03-14, 172.71
2018-03-15, 172.92
2018-03-16, 172.31
2018-03-19, 169.67
2018-03-20, 169.62
2018-03-21, 165.77
2018-03-21, BUY CREATE, 165.77
2018-03-22, 163.43
2018-03-22, BUY CREATE, 163.43
2018-03-23, 159.65
2018-03-23, BUY CREATE, 159.65

2018-03-26, 167.22
2018-03-27, 162.94
2018-03-28, 161.14
2018-03-29, 162.4
2018-04-02, 161.33
2018-04-03, 162.99
2018-04-04, 166.1
2018-04-05, 167.25

2018-04-06, 162.98
2018-04-09, 164.59
2018-04-10, 167.69
2018-04-11, 166.91
2018-04-12, 168.55
2018-04-13, 169.12
2018-04-16, 170.18
2018-04-17, 172.52

2018-04-18, 172.13
2018-04-19, 167.25
2018-04-20, 160.4
2018-04-23, 159.94
2018-04-24, 157.71
2018-04-25, 158.4
2018-04-26, 158.95

2018-04-27, 157.11
2018-04-30, 159.96
2018-05-01, 163.67
2018-05-02, 170.9
2018-05-03, 171.21

2018-05-04, 177.93
2018-05-07, 179.22
2018-05-08, 180.08
2018-05-09, 181.35

2018-05-10, 183.94
2018-05-11, 183.24
2018-05-14, 182.81
2018-05-15, 181.15
2018-05-16, 182.84

2018-05-17, 181.69
2018-05-18, 181.03
2018-05-21, 182.31
2018-05-22, 181.85
2018-05-23, 183.02
2018-05-24, 182.81

2018-05-25, 183.23
2018-05-29, 182.57
2018-05-30, 182.18
2018-05-31, 181.57
2018-06-01, 184.84
2018-06-04, 186.39

2018-06-05, 187.83
2018-06-06, 188.48

2018-06-07, 187.97
2018-06-08, 186.26
2018-06-11, 185.81



[Sign In](#)[Get started](#)

2018-06-15, 183.48
2018-06-18, 183.39

2018-06-19, 180.42
2018-06-20, 181.21
2018-06-21, 180.2
2018-06-22, 179.68
2018-06-25, 177.0
2018-06-25, BUY CREATE, 177.00

2018-06-26, 179.2
2018-06-27, 178.94
2018-06-28, 180.24
2018-06-29, 179.86

2018-07-02, 181.87
2018-07-03, 178.7
2018-07-05, 180.14
2018-07-06, 182.64
2018-07-09, 185.17

2018-07-10, 184.95
2018-07-11, 182.55
2018-07-12, 185.61
2018-07-13, 185.9
2018-07-16, 185.5
2018-07-17, 186.02

2018-07-18, 185.0
2018-07-19, 186.44
2018-07-20, 186.01
2018-07-23, 186.18
2018-07-24, 187.53
2018-07-25, 189.29

2018-07-26, 188.7
2018-07-27, 185.56
2018-07-30, 184.52
2018-07-31, 184.89
2018-08-01, 195.78
2018-08-02, 201.51
2018-08-03, 202.09
2018-08-06, 203.14
2018-08-07, 201.24
2018-08-08, 201.37
2018-08-09, 202.96
2018-08-10, 202.35
2018-08-13, 203.66
2018-08-14, 204.52
2018-08-15, 204.99
2018-08-16, 208.0
2018-08-17, 212.15
2018-08-20, 210.08
2018-08-21, 209.67
2018-08-22, 209.68

2018-08-23, 210.11
2018-08-24, 210.77
2018-08-27, 212.5
2018-08-28, 214.22
2018-08-29, 217.42
2018-08-30, 219.41
2018-08-31, 221.95

2018-09-04, 222.66
2018-09-05, 221.21
2018-09-06, 217.53
2018-09-07, 215.78
2018-09-10, 212.88

2018-09-11, 218.26
2018-09-12, 215.55
2018-09-13, 220.76
2018-09-14, 218.25
2018-09-17, 212.44
2018-09-18, 212.79
2018-09-19, 212.92
2018-09-20, 214.54

2018-09-21, 212.23
2018-09-24, 215.28
2018-09-25, 216.65
2018-09-26, 214.92
2018-09-27, 219.34
2018-09-28, 220.11





```
2018-10-04, 222.3
2018-10-05, 218.69
2018-10-08, 218.19
2018-10-09, 221.21

2018-10-10, 210.96
2018-10-11, 209.1
2018-10-12, 216.57
2018-10-15, 211.94
2018-10-16, 216.61

2018-10-17, 215.67
2018-10-18, 210.63
2018-10-19, 213.84
2018-10-22, 215.14
2018-10-23, 217.17

2018-10-24, 209.72
2018-10-25, 214.31
2018-10-26, 210.9
2018-10-29, 206.94
2018-10-30, 207.98
2018-10-31, 213.4

2018-11-01, 216.67
2018-11-02, 202.3
2018-11-02, BUY CREATE, 202.30
2018-11-05, 196.56
2018-11-05, BUY CREATE, 196.56

2018-11-06, 198.68
2018-11-06, BUY CREATE, 198.68
2018-11-07, 204.71
2018-11-08, 204.0

2018-11-09, 200.06
2018-11-12, 189.99
2018-11-12, BUY CREATE, 189.99
2018-11-13, 188.09
2018-11-13, BUY CREATE, 188.09

2018-11-14, 182.77
2018-11-14, BUY CREATE, 182.77
2018-11-15, 187.28
2018-11-16, 189.36

2018-11-19, 181.85
2018-11-20, 173.17
2018-11-20, BUY CREATE, 173.17
2018-11-21, 172.97

2018-11-23, 168.58
2018-11-26, 170.86
2018-11-27, 170.48
2018-11-28, 177.04

2018-11-29, 175.68
2018-11-30, 174.73
2018-12-03, 180.84
2018-12-04, 172.88
2018-12-06, 170.95

2018-12-07, 164.86
2018-12-10, 165.94
2018-12-11, 165.0
2018-12-12, 165.46
2018-12-13, 167.27
2018-12-14, 161.91
2018-12-17, 160.41
2018-12-18, 162.49
2018-12-19, 157.42
2018-12-20, 153.45
2018-12-20, BUY CREATE, 153.45
2018-12-21, 147.48
2018-12-21, BUY CREATE, 147.48
2018-12-24, 143.67
2018-12-24, BUY CREATE, 143.67
2018-12-26, 153.78
2018-12-27, 152.78
2018-12-28, 152.86
2018-12-31, 154.34
Final Portfolio Value: 100381.48
Return: OrderedDict([('rtot', 0.0038075421029081335), ('ravg',
1.5169490449833201e-05), ('rnorm', 0.003830027474518453), ('rnorm100',
```



Live Implementation

This is less of a section more of a collection of resources that I have for implementing trading strategies. If you want to move forward and implement your strategy in a live market check out these articles...

- [Algorithmic Trading with Python](#)
- [Build an AI Stock Trading Bot for Free](#)
- [Algorithmic Trading System Development](#)
- [Multithreading in Python for Finance](#)
- [How to Build a Profitable Stock Trading Bot](#)
- [Build a Neural Network to Manage a Stock Portfolio](#)

Algorithmic Trading Strategy Pitfalls

- **Not understanding the drivers of profitability in a strategy:** There are some that believe, regardless of the financial or economic drivers, if you can get enough data and stir it up in such a way that you can develop a profitable trading strategy. I did too when I was younger, newer to the algorithmic trading, and more naive. The truth is if you can't understand the nature of your own trading algorithmic it will never be profitable.
- **Not understanding how to split historical or synthetic data to both search for and optimize strategies:** This one is big and the consequences are dire. I had a colleague back in college tell me he developed a system with a 300% return trading energy. Naturally, and immediately, I knew he completely overfitted a model to historical data and never actually implemented it in live data. Asking a simple question about his optimization process confirmed my suspicion.
- **Not willing to accept down days:** This one is interesting, and seems to have a greater effect on newer traders more than experienced ones. The veterans know there will be down days, losses, but the overarching goal is to have (significantly) more wins than losses. This issue generally arises in deployment — assuming that the other processes
- **Not willing to retire a formerly profitable system:** Loss control, please understand risk-management and loss control. Your system was successful, and that's fantastic — that means you understand the design process and can do it again. Don't get caught up in *how well it did* look at *how well it's doing*. This has overlap with understanding the drivers of profitability in your strategy — you will know when and why it's time to retire a system if you understand why it's no longer profitable.

Thanks to TDS Editors



More from Towards Data Science

Follow

Your home for data science. A Medium publication sharing concepts, ideas and codes.

Read more from Towards Data Science




[Sign In](#)[Get started](#)

 Andreas Cha... in Data Folks Indo...


What Data Scientist Should Know About Statistical Experiments and Significant...



 Kork Ling Hui


Why you should take UQF2101J: Pursuit Of Happiness



 Andreas Cha... in Data Folks Indo...

Data Science Learning Path with Python



 Christopher Gray

Data Analysis on Billboard's Music Hot 100 Chart of 2000



 Sindhukorrapti


Handling Outliers, is it difficult?!



 Asel Men... in Towards Data Scie...


Data Retention: Handling Data with Many Missing Values and Less Than 1000 Observations



 Roman Paol... in Towards Data Sci...

Singular Value Decomposition



 Ayushi Ja... in Towards Data Scien...

Understanding Regression using COVID-19 Dataset—Detailed Analysis



[About](#) [Help](#) [Terms](#) [Privacy](#)

