Graham Giller Follow

Mar 7 · 13 min read · · Listen

Optimizing a Trading Strategy without Backtesting the Alpha

Using a Synthetic Alpha to Remove the Composite Hypothesis Problem

Backtesting, Alphas and Trading Strategies

For quants, trading is the process of repetitively taking positions in the markets as a function of information known about the future distribution of the returns of an asset. In the process of analyzing this, I generally draw a distinction between:

1. alphas, which is formally the expected mean of future returns conditioned on information I have access to; and,
2. trading strategy, which is the process used to map an alpha to a desired position.

A common tool to investigate both of these things is a backtest, or trading simulation, that takes historical price data, and the alphas we would have calculated from it, and models the performance of trading done in response to the alpha on that same price data. As an analytical technique this has some flaws but it is also, nevertheless, an essential and important part of strategy development.

returns of assets. That is

$$\alpha_t = \mathbb{E}[r_t|\mathcal{I}_s] \text{ for times } s < t$$

Here r is the return of the traded asset over the interval $(s,t]$ and the alpha, α, is the conditional expectation of those returns given the information set available to the trader. That is

$$\alpha_t = f(\mathcal{I}_s)$$

for some function, f, designed by the trader to process existing information and predict future returns. This function can be anything you want, but "useful" functions will be ones for which the first equation applies. Typically, statistical methods will be used to determine that the proposed function is, indeed, a useful one.

Definitions: Trading Strategies

A trading strategy is a recipe, or algorithm, also designed by the trader, that takes as an input an alpha, together with information on the current position, risk limits, etc., and outputs a recommended new position. We can write this as

$$h(\alpha_t, h_{t-1}, \dots)$$

where the h are the holdings, at a given time, in the assets with expected return α. The "…" is a place-holder for other parameters, often introduced to control the trading, that are usually fixed by examining the performance of a given algorithm in simulation. I usually call this function "the holding function."

The Composite Hypothesis Problem

One major problem with backtesting is doing both the alpha development and parameter


It is well known that a proper test of the validity of a system developed via some form of data-driven-inference can only be done on data that it was not part of the model development. All tests done in-sample are going to be biased to deliver results *that we already know describe that data*. However, the test for our algorithm is not how it performs on historic data but how well it performs on the new data, created daily for us by the markets, that we actually will be trading on. Prior to live trading, good practice is then to divide the historic data into two, or more, periods and to optimize only on the *in-sample* data and to test the system on untouched *out-of-sample* data.

Even with a good in-sample/out-of-sample discipline, which means we do not run our tests many-many times out-of-sample to see if we can do better with different parameter settings, there is still a problem with this analysis. It is the composite hypothesis problem:

> When evaluating the performance of trading on any data set, whether in-sample or out-of-sample, we are always jointly testing both that the alpha model is good and that the trading strategy is suitable.

This is a problem, particularly in "close to efficient" markets where alphas are weak (low R^2) and returns have fat tails (so statistics like the Sharpe Ratio are unreliable measures of performance).

Creating Synthetic Alphas

We can potentially separate the two composite hypotheses by using a rigorous statistical discipline to develop alphas, out-of-sample testing to validate that the models constructed are generally correct (that they *generalize well* in the language of machine learning), and by not using historic alphas at all to design the trading strategy.


numerically.

2. Monte Carlo simulation, which involves creating a large number of trials of potential trading algorithms on synthetic, randomly generated, data and maximizing performance of statistics computed from aggregated results.

3. Creating synthetic alphas by constructing information sets that are known to be have the right statistical properties but do not rely on a particular functional recipe, f, *that we don't really know whether it is true or not*, and then using those synthetic alphas in trading simulations on real data.

The first two methods rely on the construction of mathematical models for markets that are sufficiently accurate that they are statistically indistinct from real data. Experience teaches me that this is challenging, and methods based on the flawed assumptions of Normally distributed returns will not succeed.

The third method provides an environment in which to optimize trading strategy *on real data,* but it doesn't tell you how to build those alphas. It can be used to optimize trading strategy, but it absolutely cannot be used for the development of predictive models, known as alpha building.

Linear Additive Noise

Looking again at the definition of an alpha, we can always write it as

$$r_t = \alpha_t + \varepsilon_t$$
$$\text{where } \mathbb{E}[\varepsilon_t | \mathcal{I}_s] = 0$$
$$\text{and } \mathbb{E}[\alpha_t \varepsilon_t | \mathcal{I}_s] = 0$$

That is the return is decomposed into an alpha and an additive noise process which has mean zero and is independent of the alpha.


$$\alpha_t = \varphi r_t + \eta_t$$

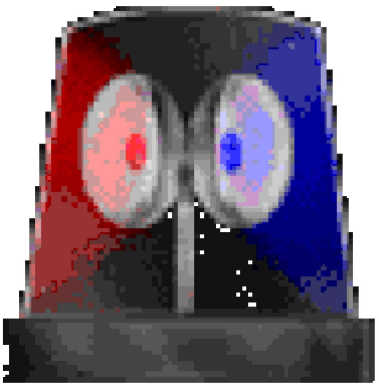
where φ reduces the scale of the synthetic alpha to that of any actual alpha we might have constructed and η is contaminating noise generated artificially. We would just have to arrange that the value of φ and the variance of η be of the the right magnitudes to deliver an information coefficient (I.C.) similar to that of a real alpha and not to disrupt the variance process of the actual observed returns, r. That turns out to be just as tricky as creating a realistic Monte Carlo, for it is essentially the same thing.

Synthetic Alphas from Natural Covariance

Suppose we want to trade a real stock, with ticker $YY, from an alpha with an I.C. of around 10%, so R^2 in the linear model is roughly 1%. In the real world of single stock trading this would be a pretty good system — see my prior article on the accuracy of the Martingale model for stock prices for reference.

If we could find a partner stock, ticker $XX, with returns that are correlated with those of $YY by about 10% then we can use these returns as our synthetic alpha without having to introduce any contaminating noise and we could model the trading of $YY in response to this alpha on actual historic returns, thereby avoiding the issue of whether the simulation created represents a realistic returns process — because it is the actual real returns.

To be clear, if x represents the returns of $XX and y represents the returns of $YY, we are going to model these as



Causality Alert!

but this does not mean that I'm suggesting we can use advanced information from $XX to build an *actual* trading strategy for $YY. At decision time *s* we are as ignorant of the future returns of $XX as we are those of $YY. This is just an analytical strategy to eliminate the composite hypothesis problem in our analysis of the trading strategy *h*.

Finding Synthetic Alphas in the Real World

If you want to trade stock $YY, and want to use this method to optimize your trading strategy, you're going to need a method to find suitable candidates for the *alpha proxy* stock $XX. Fortunately, this is very easy to do in the tools we have access to.

Getting a Universe of Daily Returns

First I'll pull the universe of NASDAQ–100 stocks discussed my previous article Is the Trend Your Friend? A Survey of the Persistence of Returns, by Year

Is the Trend Your Friend? A Survey of the Persistence of Returns, by Year

The Autocorrelation of NASDAQ–100 Member Returns

medium.com

Adventures-in-Financial-Data-Science/Find_Correlated_Stocks_for_the_NASDAQ_100.ipynb at...

Here I am collecting the scripts I have used to prepare my book "Adventures in Financial Data Science." These are...

github.com

But if you can't access that, or want to use another Python3 platform, here's the code

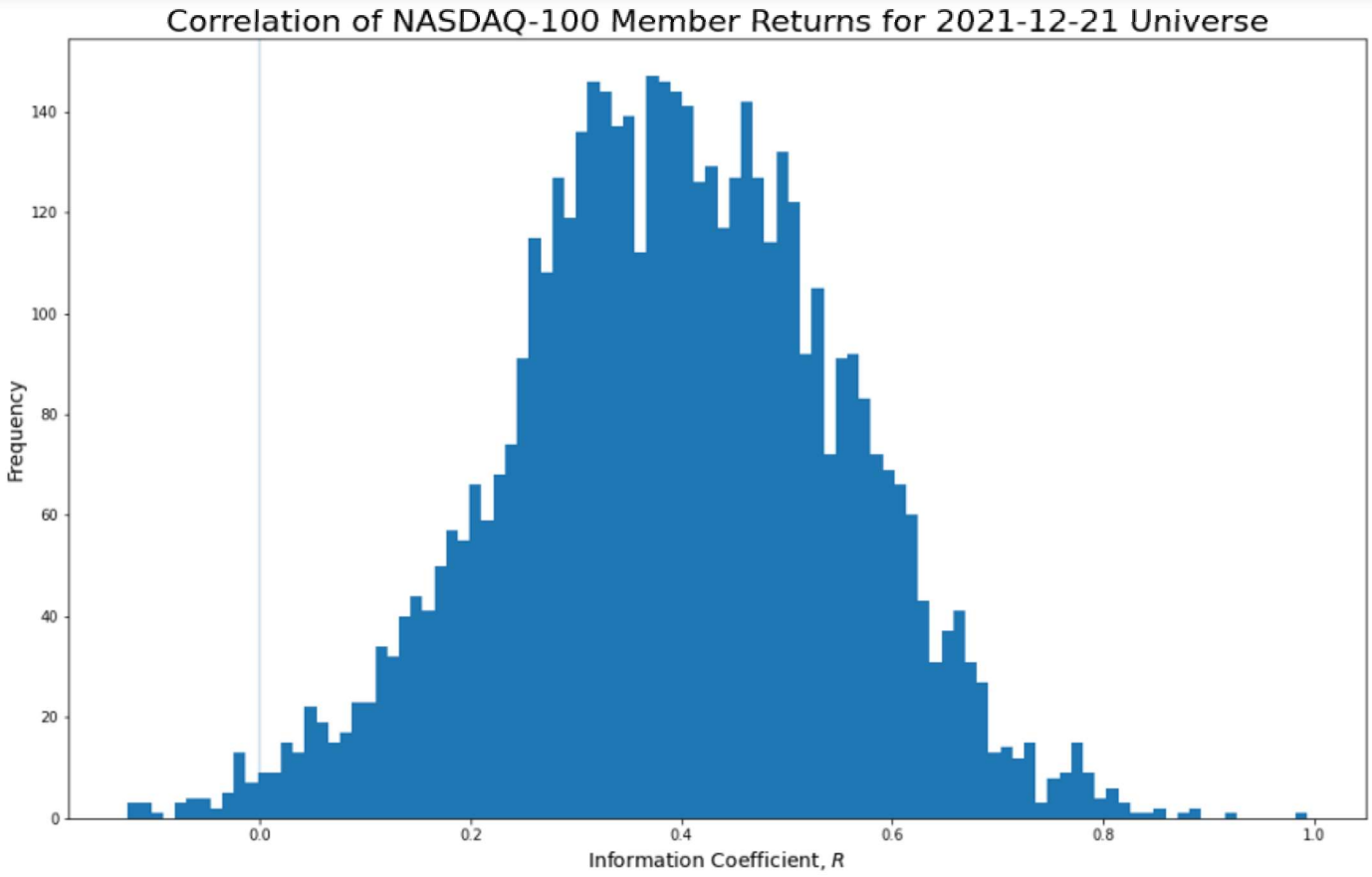
```
print("Installing yfinance and getting data...")
!pip install yfinance 1>/dev/null
import pandas as pd
import numpy as np
from yfinance import download
from datetime import datetime

# set time frame of analysis
today=datetime.now()
first_year=today.year-3

# this list as of 2020-12-21 https://www.nasdaq.com/press-release/annual-changes-to-the-nasdaq-100-index-2020-12-11
# you must use Adjusted Close to account for distributions such as dividends and splits, not simple Close

prices=download(['AAPL','ADBE','ADI','ADP','ADSK','AEP','ALGN','AMAT','AMD','AMGN','AMZN','ANSS','ASML','ATVI','AVGO','BIDU','BIIB','BKNG','CDNS','CDW','CERN','CHKP','CHTR','CMCSA','COST','CPRT','CRWD','CSCO','CSX','CTAS','CTSH','DLTR','DOCU','DXCM','EA','EBAY','EXC','FAST','FB','FISV','FOXA','GILD','GOOG','GOOGL','HON','IDXX','ILMN','INCY','INTC','INTU','ISRG','JD','KDP','KHC','KLAC','LRCX','LULU','MAR','MCHP','MDLZ','MELI','MNST','MRNA','MRVL','MSFT','MTCH','MU','NFLX','NTES','NVDA','NXPI','OKTA','ORLY','PAYX','PCAR','PDD','PEP','PTON','PYPL','QCOM','REGN','ROST','SBUX','SGEN','SIRI','SNPS','SPLK','SWKS','TCOM','TEAM','TMUS','TSLA','TXN','VRSK','VRSN','VRTX','WBA','WDAY','XEL','XLNX','ZM'],"%d-01-01" % first_year,today.strftime("%Y-%m-%d"))["Adj Close"]
prices.index=pd.DatetimeIndex(prices.index).to_period('D') # this is to make sure the index is properly time aware
prices
```

```
returns=pd.DataFrame({
    ticker:prices[ticker]/prices[ticker].shift()*1e2-1e2 \
        for ticker in list(prices)
})
```

And finally I'll compute the covariance matrix, which I'll vectorize the lower triangle of via the method `.melt()`.

```
# compute correlation matrix
rho=returns.corr(method='pearson',min_periods=10)

for row in range(rho.shape[0]):
    for column in range(row,rho.shape[0]):
        rho.iloc[row,column]=np.nan # to keep lower left triangle

rho["LeftTicker"]=rho.index
rhov=rho.melt(id_vars='LeftTicker',
    var_name='RightTicker',
    value_name='Correlation').dropna()

# get rid of entries labeled "Ticker" created by melt
rhov=rhov[(rhov["LeftTicker"]!="Ticker")& \
    (rhov["RightTicker"]!="Ticker")]
```

This creates a data-frame with 5,050 entries each of which is the correlation of daily returns between pairs of stocks selected from the NASDAQ–100. The aggregate distribution of these correlations looks like the histogram below.



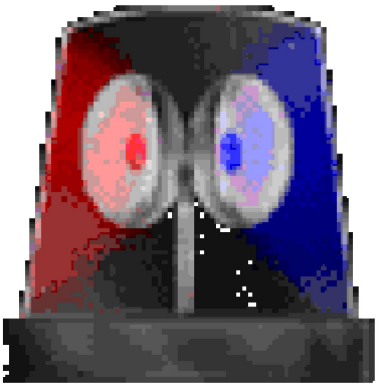
Note that most stocks have a correlation *much bigger* than the 10% we're targeting in this analysis. In fact, a pair of NASDAQ stocks picked at random are likely to have a correlation of returns of around 45%. However, an I.C. of that level is simply too large to be realistic, stock I.C.s for daily trading are typically much smaller — because markets are, in fact, reasonably efficient, if not perfectly.

Selecting Candidate Stocks

At this stage, selecting candidate stocks is pretty easy. For your $YY you might want to choose a market favourite, like $TLSA, or a social media company like $SNAP, for example. I'm just going to pick a stock and it's partner entirely at random for this study.

```
np.random.seed(123456) # set the seed to guarantee reproducability
pair=np.random.choice(rhov.index[ \
    (rhov["Correlation"]>=0.09e0)&(rhov["Correlation"]<=0.11e0)])
x_name,y_name=pair.split(",")
```

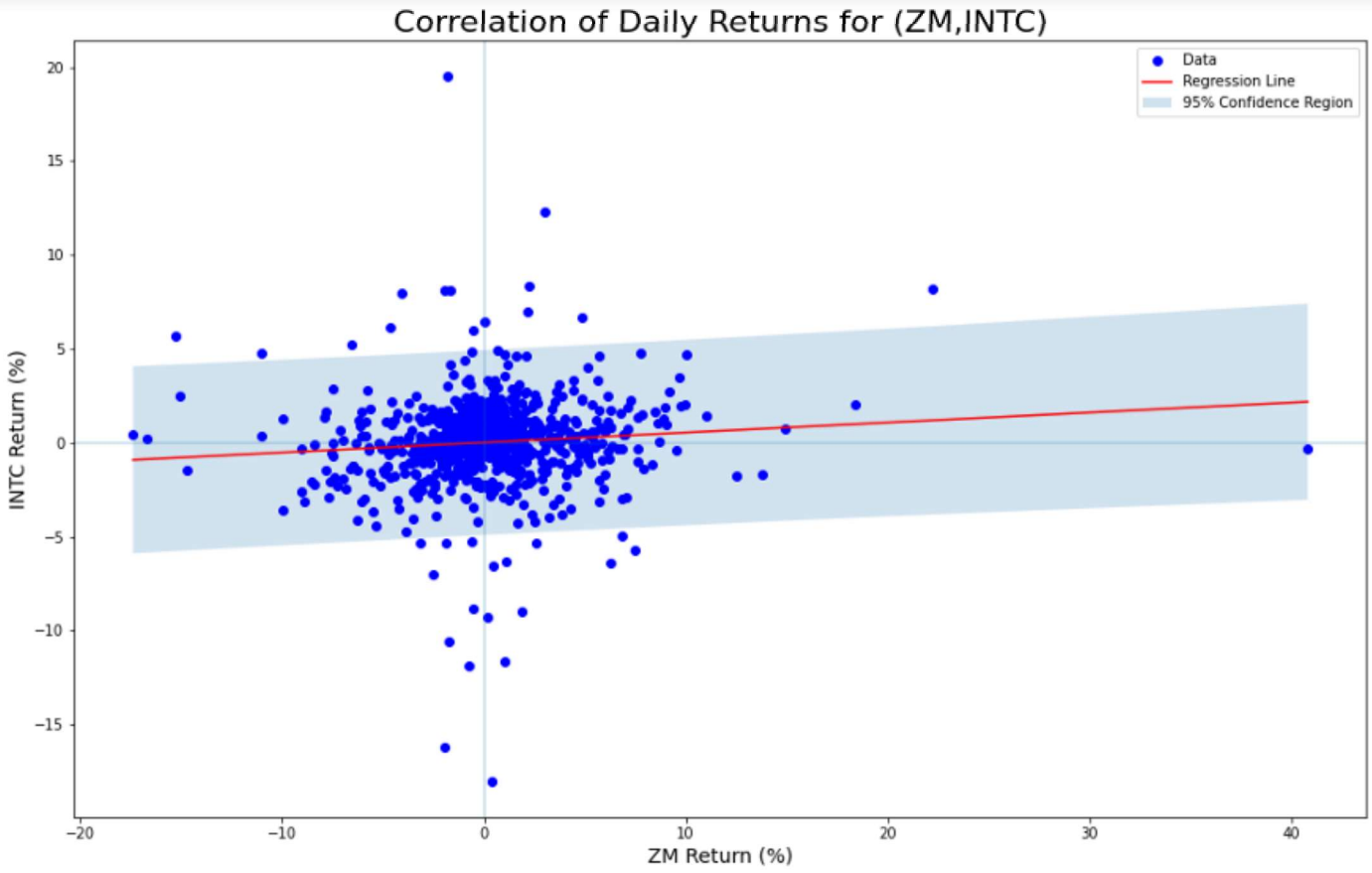
the traded stock and COVID driven internet success Zoom, $ZM, for the alpha generating signal.



Correlation Alert!

Take care: do not interpret this to m use the returns of Zoom as a trading signal for the future returns of Intel. For a start, the historic correlation is pretty low, at 9.1%, but most importantly we are just using the returns of $ZM as a proxy for "some alpha we might invent for Intel with an I.C. of around 10%" because its correlation is at that level and this way we an use the real world returns of Intel as an environment where we can optimize a trading strategy.

Longitudinally, the returns are shown in the figure below. The linked notebook shows how to compute this plot.



This is a significant regression with an t statistic of 2.45 and a p value of 0.015, but the scatter plot illustrates the low level of information really contained here. If you have an alpha with an IC of 10%, you will see similar plots.

Defining a Trading Strategy

I'm going to define a really elementary trading strategy to test here, with one free parameter.

1. If the alpha is larger than B times it's standard deviation, then take a position of 1 unit in the direction of the alpha;
2. otherwise, hold the current position.

This is really simple, but is driven by the heuristic that we want to maximize the signal-to-noise ratio of our trading, which we will do by selectively vetoing trades that do not have a

$$h_t = h(\alpha_t, h_{t-1}) = \begin{cases} \operatorname{sign} \alpha_t & \text{if } |\alpha_t| \geq B\sigma_\alpha \\ h_{t-1} & \text{otherwise.} \end{cases}$$

Here, h refers to the position we adopt at the *beginning* of the interval $(t-1,t]$, to experience the returns *over* that interval. This generates a sequence of profits

$$\{h_t r_t : t \in [1, T]\}$$

from which we can compute performance statistics, such as the Sharpe Ratio. Note that if we *were* paying transaction costs, the P/L sequence would be something like

$$\{h_t r_t - \kappa |h_t - h_{t-1}| : t \in [1, T]\}$$

and the second term would affect the choice of holding function.

Optimizing the Trading Strategy

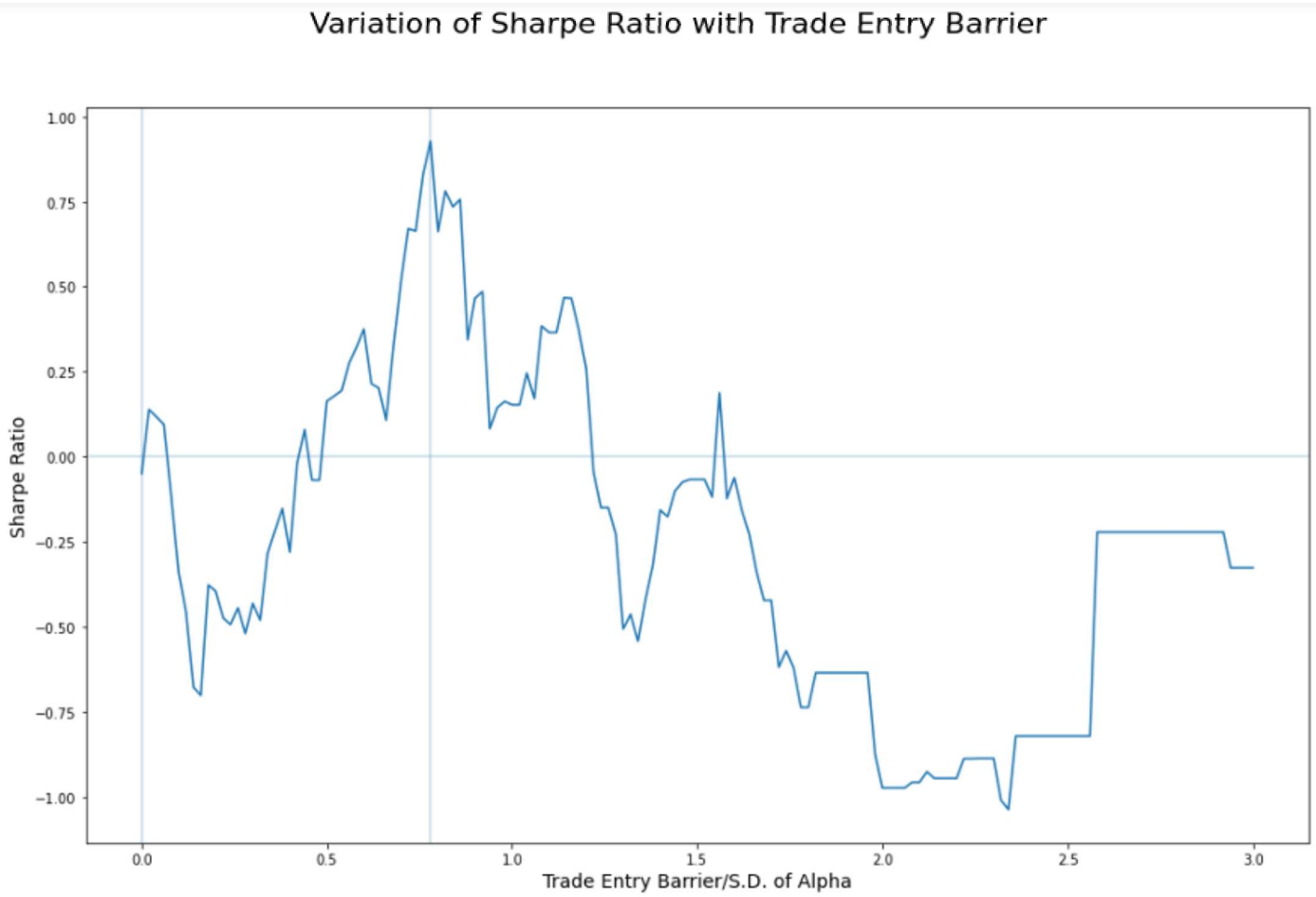
What value should we use for B? We can look to theory for that, and some work I've done suggest $B \approx 1$ (the precise value depends on the distribution of the alpha), but we can also just simply scan through different values and compute their performance and, because we're not using a real alpha here, this analysis is not actually testing the performance of that alpha. It is purely optimizing the performance of a trading strategy built around a signal with a similar information content — and we have no ambiguity that it does have that information content because we've chosen it to do so in this particular data set.

```
from tqdm.notebook import tqdm # makes a progress bar

# set up the experiment
alpha_sd=data[x_name].std()
```


```
        scan=data[[x_name,y_name]].copy() # take a copy of the returns
        scan["alpha_std"]=scan[x_name]/alpha_sd
        scan.loc[scan.index[0],"holding"]=0e0

        # simulate trading strategy
        for i,date in enumerate(scan.index[1:]):
            scan.loc[date,"holding"]= \
                np.sign(scan.loc[date,"alpha_std"]) \
                    if abs(scan.loc[date,"alpha_std"])>=barrier \
                    else scan.loc[scan.index[i],"holding"]

        # P/L accounting, make sure causally correct
        scan["prior_holding"]=scan["holding"].shift()
        scan["pnl"]=scan["prior_holding"]*scan[y_name]
        sharpe.loc[barrier,"Sharpe"]=scan["pnl"].mean() \
            /scan["pnl"].std()*np.sqrt(252e0)

    barrier=sharpe.index[sharpe["Sharpe"].argmax()]
```

This loops over values of B between 0 and 3, executes the trading strategy for that particular value, and does P/L accounting to measure performance. I find people tend to make a big deal over “backtesting environments,” but the truth is they are not that mysterious or complicated to build. You just need to make sure the math is done in a causally proper manner.

The results are:

We see the Sharpe Ratio peaks for $B=0.78$, and has a value of 0.93 at the peak. Grinold & Kahn's *Fundamental Law of Active Management* suggests we should use

$$\text{S.R.} \approx R\sqrt{N}$$

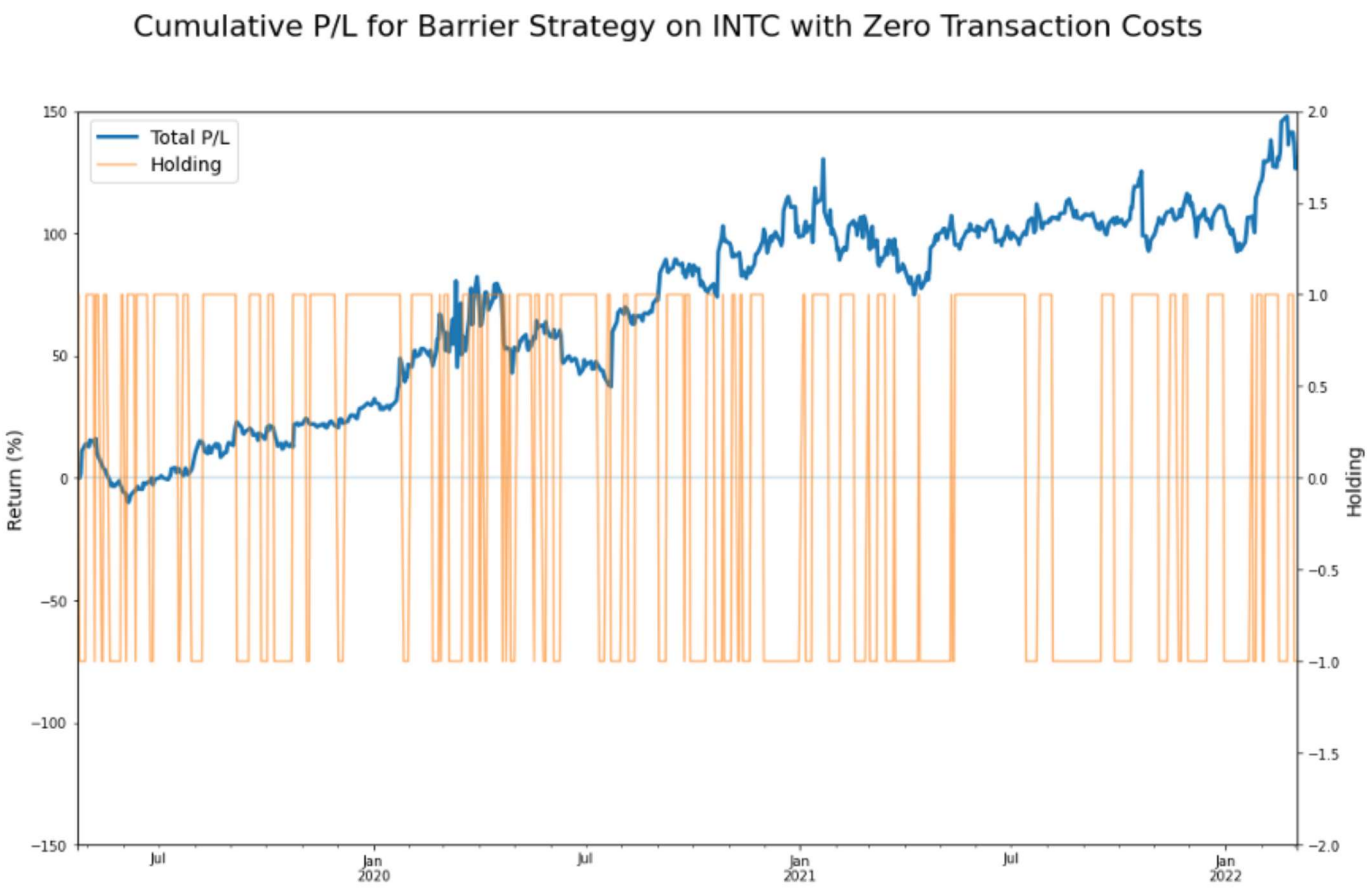
where R is the I.C. and N is the number of trades per annum. This would give a Sharpe Ratio of 1.4, suggesting we've not extracted *all* of the value out of the alpha with this algorithm — but we're in the right ballpark.

My work suggests this rule is true for the trading algorithms I'm able to analyze analytically, giving a formula such as

performance management, and this result likely contains a large sampling error.

Performance of the Optimized Trading Algorithm

Finally, let's take a look at the performance of the trading algorithm, and compare that side-by-side with the positions held. The point of looking at this plot is to find out if the performance is delivered steadily throughout the testing period or can be attributed to a small number of particularly lucky trades.



Conclusions

What I've tried to demonstrate here that it is possible to perform numerical optimization of trading strategies without having to backtest the alpha as part of the process. This is critically important as alphas with realistic Information Coefficients will often show periods of underperformance. Obviously, it would make more sense to aggregate performance over several proxy securities, $XX, for a given $YY, to truly understand what particular setting of B, or other parameters for more complex trading strategies, delivers the best performance.

If you like this article and would like to read more of my work, consider my book *Adventures in Financial Data Science* which is available as an eBook for Kindle, and also from Apple Books and Google Books. A revised second edition will be published by *World Scientific*.

Adventures in Financial Data Science: The empirical properties of financial data and some other...

Adventures in Financial Data Science: The empirical properties of financial data and some other things that interested...

www.amazon.com

Alternatively, you can order the paperback directly from me via our website.

Store — Giller Investments

And if you want to chat with me about my work, consider joining our slack channel.

Slack

Edit description

join.slack.com