Haohan Wang Follow
Dec 30, 2015 · 8 min read

Backtest of Time Series Momentum strategy

In the *last article*, I discussed the cross-sectional momentum strategy and showed you the outcome of the backtest using the strategy. In order to take advantage of the research that I have done in the previous article which is related to measuring time-series momentum, I would like to perform a backtest on this strategy and also to compare these two momentum strategies.

Here is some information that I collected from the research about the comparison between these two strategies. It worths to mention that momentum strategies, in general, have proved to be robust across time, countries and asset classes. Even though the existence of the momentum abnormal returns continues to challenge the market efficiency theory(since 'the existence of momentum is a market anomaly, which finance theory struggles to explain. The difficulty is that an increase in asset prices, in and of itself, should not warrant a further increase. Such increase, according to the efficient-market hypothesis, is warranted only by changes in demand and supply or new information (cf. fundamental analysis).')

Also, the research mentions the test conducted by them shows that both time-series and cross-sectional momentum strategies produce significant positive outcomes under numerous implementations in the majority of developed stock markets. The time-series momentum strategy outperforms the cross-sectional momentum strategy under optimal implementations conditions in all markets and is statistically significant in half of these markets.

One possible explanation for the superiority of the time-series momentum strategy is that it forms portfolios from slightly smaller capitalization stocks with a greater spread in past performance between the winner and loser stocks. However, the transaction costs and risk from time-series momentum strategy are higher than the costs from cross-sectional momentum strategy which is largely a consequence of time-series momentum strategy tends to select smaller and growth stocks and generating a higher turnover a market cycle. We could take these findings into our consideration when we are trying to construct a strategy.

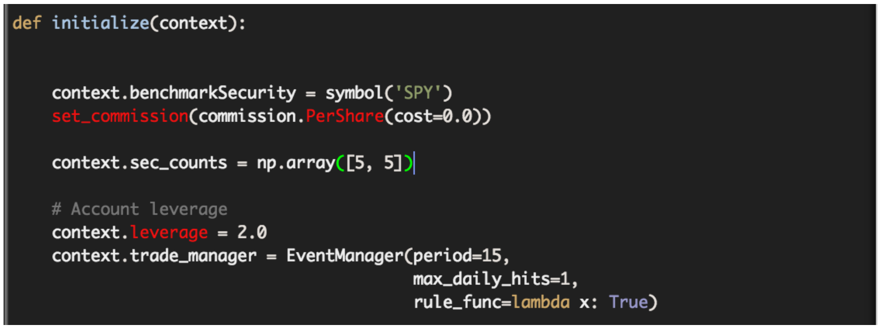
Different from cross-sectional momentum which assign into winner and loser portfolios based on their ranking (I allocated top 1% quantile stocks in my long basket and bottom 1% in my short basket according to the strategy I used in my last article), stocks are assigned to portfolio on the basis of their absolute returns over some pre-defined periods for time-series momentum strategy. Therefore, the number of stocks assigned to the winner and loser baskets will typically differ if we use time-series momentum strategy, that is to say, if the market is doing well possibly more securities will be in the winner portfolio but equally more securities will be in the loser basket if the market has been doing poorly. (Only if the returns are normally distributed, the cut-offs will result in the required number o stocks being allocated to either winner or loser baskets equally.)

In sum, the analysis above implies that the time-series momentum strategy may perform better as it will load up on winning stocks when markets are strong and on losing stocks when markets are weak in turn.

The key to the successful momentum strategies is to identify winners and losers early in their up or down cycle and reversing these positions in close to an optimum fashion. Instead of setting up a single or double cut-off such as if returns of stocks are above x% and the loser portfolio is those are below x%, I choose to use the Moving Average Crossover strategy to set up criteria to filter the winner and loser stocks as I introduced before in my 'Measuring Momentum' *article* in the backtest I will show you below. In order to capture true trends in markets, I choose to use the popular moving average pair which include 50-day simple moving average(SMA) and 200-day simple moving average(SMA)(this pair is often looked at by big financial institutions as a long range indicator of market direction).

I modified the original Algo based on my understanding and research I did about moving average crossover and momentum strategy which I will explain them in detail for you later.

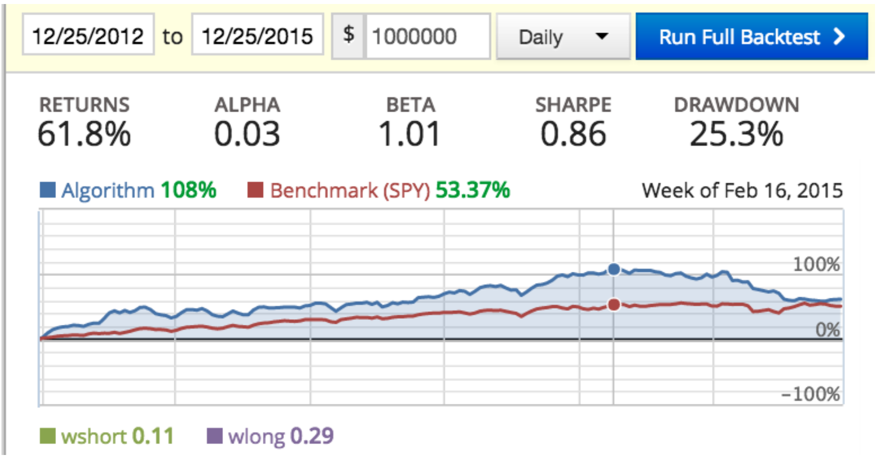
Let's look at the first piece of code here, I set up some steps before trading. In this strategy, we only want to long and short certain number of stocks (long top 5 qualified stocks and short bottom 5 underperformed stocks) so I set it up here by 'np.array' function. Also, we intend to create a strategy which has long and short positions equal to each other(I will show you how to calculate the weight for each stock in the last picture). The last step here is to set up the portfolio rebalancing frequency which is every 15 days(around 2 weeks) and each stock can only be traded once per day.(Because long holding periods may result in lower Sharpe Ratios and easily to present the backtest statistical significance problem.)



```
def initialize(context):


    context.benchmarkSecurity = symbol('SPY')
    set_commission(commission.PerShare(cost=0.0))

    context.sec_counts = np.array([5, 5])

    # Account leverage
    context.leverage = 2.0
    context.trade_manager = EventManager(period=15,
                                         max_daily_hits=1,
                                         rule_func=lambda x: True)
```

The picture below is the backtesting outcome of this strategy over 3 years from 2012 to 2015. The total returns are 61.8%. The beta is 1.01 which is kind of high and may imply that this strategy could easily be influenced by the movements of the market. As for the Sharpe Ratio, it is relatively low compared to the total returns which means for every unit of risk we are taking the returns may not be as good as we expected.

The possible reason behind those problems I could think of is that the total number of stocks we are trading on each side is relatively too few. The long/short strategy is supposed to betting on the average performance of a large number of stocks included in the portfolio and the larger the number of stocks we are trading the higher the possibility that out portfolio will perform well.



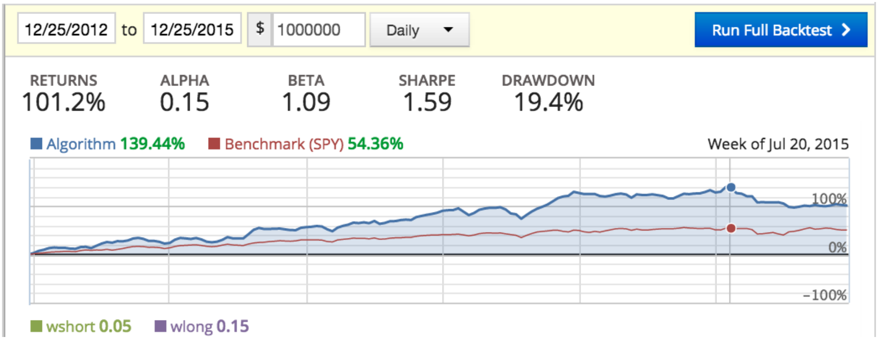
12/25/2012 to 12/25/2015 $ 1000000 Daily

RETURNS	ALPHA	BETA	SHARPE	DRAWDOWN
61.8%	0.03	1.01	0.86	25.3%

🟦 Algorithm 108% 🟥 Benchmark (SPY) 53.37% Week of Feb 16, 2015

🟩 wshort 0.11 🟪 wlong 0.29

So I make a little change here, I change the number of stocks we are going to trade from 5 into 10 in both long and short basket. The graph below shows the result after changing. Although the beta is still high, the Sharpe Ratio improves a lot, which gives us more confidence about the statistical significance about this strategy. The Max Drawdown is also smaller after I enlarging the trading volume.



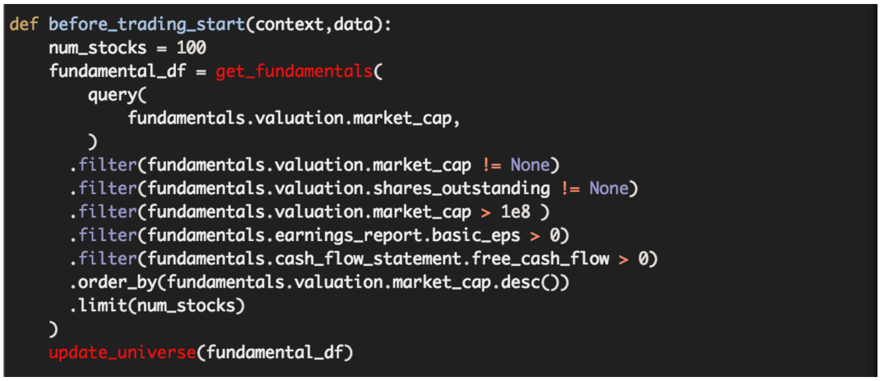
12/25/2012 to 12/25/2015 $ 1000000 Daily

RETURNS	ALPHA	BETA	SHARPE	DRAWDOWN
101.2%	0.15	1.09	1.59	19.4%

🟦 Algorithm 139.44% 🟥 Benchmark (SPY) 54.36% Week of Jul 20, 2015

🟩 wshort 0.05 🟪 wlong 0.15

The picture below is the step I used to filter down the universe that I will choose stocks from. Based on the criteria I pass in which include the market cap should be larger than $100,000,000, the EPS of the company should be positive as well as the free cash flow of the firm, I can keep all of those qualified stocks into my daily tradable universe. Then I rank the stocks following descending order based on their Market Cap. After filtering down our 'daily universe of stocks', the total number of stock we can get access to after this step is 100.

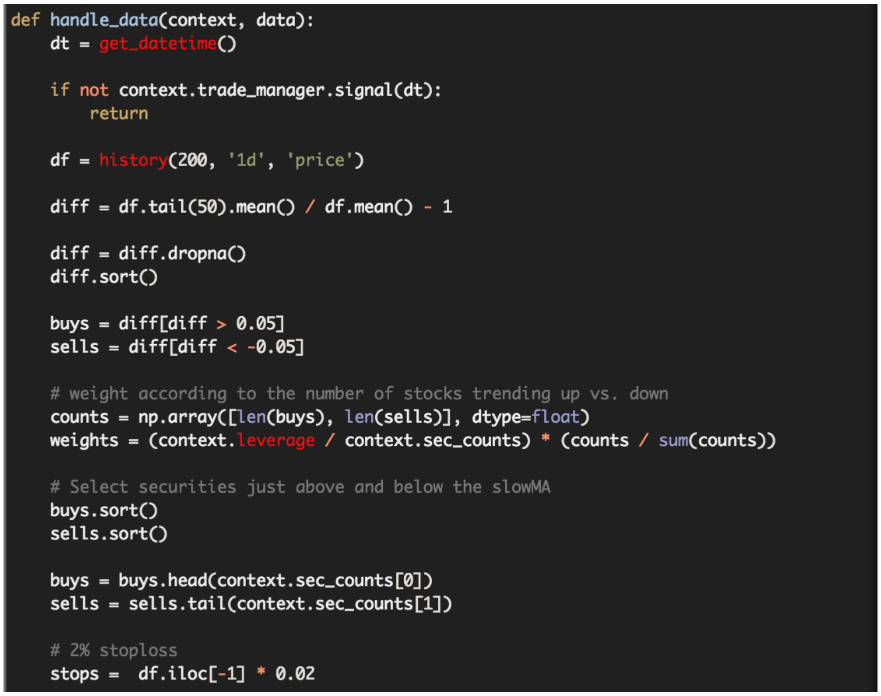


```
def before_trading_start(context,data):
    num_stocks = 100
    fundamental_df = get_fundamentals(
        query(
            fundamentals.valuation.market_cap,
        )
      .filter(fundamentals.valuation.market_cap != None)
      .filter(fundamentals.valuation.shares_outstanding != None)
      .filter(fundamentals.valuation.market_cap > 1e8 )
      .filter(fundamentals.earnings_report.basic_eps > 0)
      .filter(fundamentals.cash_flow_statement.free_cash_flow > 0)
      .order_by(fundamentals.valuation.market_cap.desc())
      .limit(num_stocks)
    )
    update_universe(fundamental_df)
```

As I mentioned before about the two moving averages I used in this strategy, this step will show the mechanism behind this strategy according to the relationship between two averages.

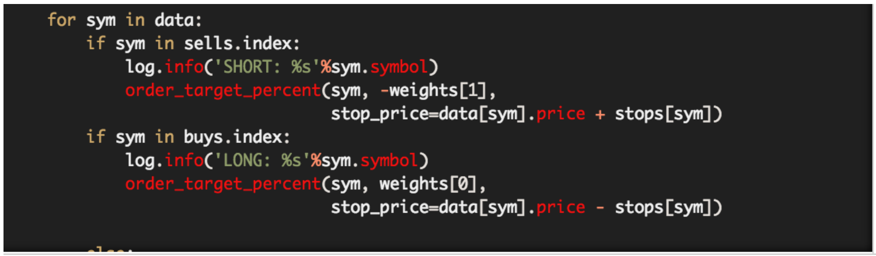
If the 50-day SMA crosses above the 200-day SMA there is a 'BUY' signal generated while a 'SELL' signal is triggered by a 50-day SMA crossing below the 200-day SMA. But still there will be some false signals showing up, the strategy that I came up with regarding this problem is to add a filter. Instead of measuring if 50-day SMA is higher than 200-day SMA(you can see from the formula I use here to check (50-day SMA/200-day SMA)—1), I choose to wait until the 50-day SMA is 5% higher than 200-day SMA then buy the stocks meeting this criterion; and I do the same to the short portfolio: if 50-day SMA is 5% lower than 200-day SMA I will add those qualified stocks into my short portfolio.

Also, I rank both portfolios in ascending order(according to the gap between two moving averages). Then I choose only the top 5(or 10) stocks whose 50-day SMA are just slightly higher than the 200-day SMA plus 5% from the long portfolio and only the bottom 5 stocks(or 10 stocks) whose 50-day SMA are just slightly lower than 200-day SMA minus 5% from short portfolio.



```
def handle_data(context, data):
    dt = get_datetime()

    if not context.trade_manager.signal(dt):
        return

    df = history(200, '1d', 'price')

    diff = df.tail(50).mean() / df.mean() - 1

    diff = diff.dropna()
    diff.sort()

    buys = diff[diff > 0.05]
    sells = diff[diff < -0.05]

    # weight according to the number of stocks trending up vs. down
    counts = np.array([len(buys), len(sells)], dtype=float)
    weights = (context.leverage / context.sec_counts) * (counts / sum(counts))

    # Select securities just above and below the slowMA
    buys.sort()
    sells.sort()

    buys = buys.head(context.sec_counts[0])
    sells = sells.tail(context.sec_counts[1])

    # 2% stoploss
    stops =  df.iloc[-1] * 0.02
```

Finally, we iterate through each stock in our daily universe to check if they belong to the ‘long’ or ‘short’ basket. We also set up the weight for each stock in both baskets in this step. Then we can purchase or sell the corresponding stocks based on the steps that we setup.



```
    for sym in data:
        if sym in sells.index:
            log.info('SHORT: %s'%sym.symbol)
            order_target_percent(sym, -weights[1],
                                 stop_price=data[sym].price + stops[sym])
        if sym in buys.index:
            log.info('LONG: %s'%sym.symbol)
            order_target_percent(sym, weights[0],
                                 stop_price=data[sym].price - stops[sym])
        else:
```

This is just a simple application about time-series momentum strategy. I am still learning and practicing on how to construct a more statistical significant strategy with higher Sharpe Ratio, higher alpha but lower beta and Max Drawdown. I will keep updating my learning progress here.

At the end of last article I introduced some benefits about Algo trading, I would like to supplement some limits about Algo trading/backtesting here. Actually there are a lot of pitfalls that we have to try to avoid when we are conducting backtest on our strategies such as the overfitting problem that I mentioned before, or some changes within the certain stocks like stock splits or you may accidently include delisted stocks or sometimes your Algo tries to sell stocks can no be shorted, etc. Besides these pitfalls, the predictive power of any backtests rests on the central assumption that the statistical properties of the price series are unchanging(means we assume that if the trading rules that were profitable in the past will be profitable in the future). Obviously, these assumptions are invalidated in varying degrees such as a country's economic prospect changes, or a financial market's structure changes. Whenever there is a *'regime shifts'(*I explained this concept in my previous *article), a regime change causes future samples to follow a different distribution,* the strategies that performed well before may stop performing. Backtests done using data prior to such regime shifts may be quite worthless.

Hope you enjoy the content and please feel free to leave your comment on anything inefficient or worth optimizing here. I would love to hear from you. Thank you!

Reference:

- Quantopian: Momentum Strategy with a Dynamic Universe
- https://en.wikipedia.org/wiki/Momentum_(finance)
- http://researchcommons.waikato.ac.nz/bitstream/handle/10289/9343/thesis.pdf?sequence=3&isAllowed=y
- http://www.investopedia.com/university/movingaverage/movingaverages4.asp
- http://arno.uvt.nl/show.cgi?fid=115650
- http://www.benzinga.com/trading-ideas/technicals/15/08/5764478/study-determines-the-best-moving-average-crossover-trading-st