This Blog is Systematic.

Also hydromatic, automatic: Rob Carvers blog. "It's Geeks Enlightening"

Tuesday, 10 November 2015

Random data: Evaluating "Trading the equity curve"

Everyone hates drawdowns (those periods when you're losing money whilst trading). If only there was a way to reduce their severity and length....

Quite a few people seem to think that "trading the equity curve" is the answer. The basic idea is that when you are doing badly, you reduce your exposure (or remove it completely) whilst still tracking your 'virtual' p&l (what you would have made without any interference). Once your virtual p&l has recovered you pile back into your system. The idea is that you'll make fewer losses whilst your system is turned off. It sounds too good to be true... so is it? The aim of this post is to try and answer that question.

This is something that I have looked at in the past, as have others, with mixed results. However all the analysis I've seen or done myself has involved looking at backtests of systems based on actual financial data. I believe that to properly evaluate this technique we need to use large amounts of *random* data, which won't be influenced by the fluke of how a few back tests come out. This will also allow us to find which conditions will help equity curve trading work, or not.

This is the second post in a series on using random data. The first post is here.

How do we trade the equity curve?

I'm going to assume you're reasonably familiar with the basic idea of equity curve trading. If not, then its probably worth perusing this excellent article from futures magazine.

An equity curve trading overlay will comprise of the following components:

- A way of identifying that the system is 'doing badly', and quantifying by how much.
- A rule for degearing the trading system given how badly it is doing
- A second rule for regearing the system once the 'virtual' account curve is 'doing better'

I've seen two main ways for identifying that the system is doing badly. The first is to use a straightforward drawdown figure. So, for example, if your drawdown exceeds 10% then you might take action.

(Sometimes rather than the 'absolute' drawdown, the drawdown since the high in some recent period is considered)

The second variation is to use a moving average (or some other similar filter) of the account curve. If your account curve falls below the moving average, then you take action.

(There are other variations out there, in particular I noticed that the excellent Jon Kinlay blog had a more complex variation)

As for degearing your system, broadly speaking you can eithier degear it all in one go, or gradually. Usually if we dip below the moving average of an equity curve then it is suggested that you cut your position entirely.

Whilst if you are using the current drawdown as your indicator, then you might degear gradually: drawdown; then by another 20% for a total of 40% when you hit a 20% drawdown and so on.

Note that this degearing will be in addition to the normal derisking you should always do when you lose money; if you lose 10% then you should derisk your system by 10% regardless of whether you are using an equity curve trading overlay.

Finally the 'regearing' rule is normally the reverse of the degearing rule and process.

Prior research

The idea of trading the equity curve is something that seems to have bypassed academic researchers (unless they are calling it something else - please write in if you know about any good research) so rather than any formal literature review I had a quick look at the first page of

- My website: systematicmoney.org
- twitter @investingidiocy
- linkedin/robert-stuart-carver
- facebook.com/Robert.Carver.Author
- github.com/robcarver17
- goodreads.com/Robert_Carver

- Favourite books
- Useful links
- Legal Disclaimer (please don't sue me)

Blog Archive

▶ 2019 (9)
▶ 2018 (10)
▶ 2017 (14)
▶ 2016 (11)
▼ 2015 (32)
▶ December (1)
▼ November (5)
Random data: Random wanderings in portfolio optimi...
David Versus Goliath
Random data: Evaluating "Trading the equity curve"...
Using random data
Typo in definition of carry rule for futures in "S...
▶ October (1)
▶ September (5)
▶ August (1)
▶ July (7)
▶ June (3)
▶ May (1)
▶ April (2)
▶ March (3)
▶ February (1)
▶ January (2)
▶ 2014 (21)
▶ 2013 (1)

Search This Blog

Search

Popular Posts

Interactive brokers native python API
Until quite recently interactive brokers didn't offer a python API for their automated trading software. Instead you had to put up wi...

Historic data from native IB python API
This is the second in a series of posts on how to use the native python API for interactive brokers . This post is an update of the post I w...

A simple breakout trading rule (pysystemtrade)

google.

Positive

adaptrade
crazy ivan (whether he's a reliable source I don't know...)
Jon Kinlay (though not the normal system)

Negative (or at least no clear benefit)

Futures magazine (and also here)
Anthony Garnett
futures.io
r-bloggers

Why random data?

I personally find the above research interesting, but not definitive one way or another. My main issue is that it was all done on different financial instruments and different kinds of trading systems, which unsuprisingly gave different results. This might be because there is something 'special' about those instruments where equity curve trading worked, but it's more likely to be just dumb luck. Note: It is slightly more plausible that different kinds of trading rules will give different results; and we'll explore this below.

I personally think that we can't properly evaluate this kind of overlay without using *random* data. By generating returns for different arbitrary trading strategies we can then judge whether *on average* equity curve trading will be better.

Another advantage of using random data to evaluate an equity curve overlay system is that we avoid potentially *overfitting*. If we run one version of the overlay on our system, and it doesn't work, then it is very tempting to try a different variation until it does work. Of course we could 'fit' the overlay 'parameters' on an out of sample basis. But this quite a bit of work; and we don't really know if we'd have the right parameters for that strategy going forward or if they just happened to be the best for the backtest we ran.

Finally using random data means we can discover what the key characteristic of a trading system is that will allow trading the equity curve to work, or not.

Designing the test

Which overlay method?

There are probably an infinite variety of methods of doing an equity curve trading overlay (of which I touched on just a few above). However to avoid making this already lengthy post the size of an encylopedia I am going to limit myself to testing just one method. In any case I don't believe that the results for other methods will be substantially different.

I'm going to focus on the most popular, moving average, method:

"When the equity curve falls below it's N day moving average, turn off the system. Keep calculating the 'virtual' curve, and it's moving average during this period. When the 'virtual' curve goes back above it's moving average then turn the system back on"

That just leaves us with the question of N. Futures magazine uses 10, 25 and 40 days and to make life simple I'll do the same. However to me at least these seem incredibly short periods of time. For these faster N trading costs may well overwhelm any advantage we get (and I'll explore this later).

Also wouldn't it be nice to avoid the 3 year drawdown in trend following that happened between 2011 and 2013? Because we're using random data (which we can make as long as we like) we can use longer moving averages which wouldn't give us meaningful results if we tested just a few account curves which were 'only' 20 years long.

So I'll use N=10, 25, 40, 64, 128, 256, 512

(In business days; 2 weeks, 5 weeks, 8 weeks, 3 months, 6 months, 1 year, 2 years)

```
def isbelow(cum_x, mav_x, idx):

    ## returns 1 if cum_x>mav_x at idx, 0 otherwise

    if cum_x[idx]>=mav_x[idx]:
        return 1.0
    return 0.0
```

B reakout . Not the classic home arcade game, seen here in Atari 2600 version, but what happens when a market price breaks out of a trading ...

Can you eat geometric returns?
This post is about a slightly obscure, but very important, issue. Should we use geometric or arithmetic means of returns to evaluate investm...

Placing orders in the native python IB API
This the fourth in a series of posts on using the native python API for interactive brokers . You should read the first , second , and thi...

Streaming market data from native python IB API
This the third in a series of posts on using the native python API for interactive brokers . You should read the first , and the second , be...

Using swigibpy so that Python will play nicely with Interactive Brokers API
The interactive brokers API is as far I know the only way that a non institutional client can access financial markets in a way which mak...

Things I wish interactive brokers would do with their API software
As regular readers know I use interactive brokers (IB) to run an automated futures trading system . Now in many ways IB are great. They hav...

Obligatory Bitcoin post
No, I'm not dead. You'd be forgiven for thinking I was given my lack of blog posts in the last few months, but I was busy; first p...

Correlations, Weights, Multipliers.... (pysystemtrade)
This post serves three main purposes: Firstly, I'm going to explain the main features I've just added to my python back-testing pa...

Tags

Systematic Trading (38) Python (29) Technology (28) Portfolio optimization (26) Interactive Brokers (22) pysystemtrade (19) Systematic_Trading_Book (17) Uncertainty (17) Statistics (14) Trend following (14) Investment idiocy (11) risk management (11) Systems building (9) Behavioural finance (8) Execution (6) Hedge funds (6) Random data (6) Volatility (6) Finance industry economics (5) High frequency trading (5) LOBO (5) Smart_Portfolios_Book (5) Conditional distributions (4) Equities (4) Kelly criterion (4) Novice investors (4) banks (4) carry (4) leverage (4) skew (4) sqllite (4) Git (3) Politics (3) housing associations (3) prediction (3) CAPM (2) Compound interest (2) ETF (2) Equity risk premium (2) Financial industry pay (2) asset allocation (2) bitcoin (2) handcrafting (2) ATR (1) Capital correction (1) Commodities (1) Docker (1) Factor models (1) Partial Correlations (1) QuantCon (1) Quantitative Easing (1) brexit (1) geometric returns (1) implicit fitting (1) kurtosis (1) leveraged trading book (1)

Quantocracy - featured blog

Top 60 UK investment blogs

```
def apply_overlay(x, N_length):
    """
    apply an equity curve filter overlay

    x is a pd time series of returns

    N_length is the mav to apply

    Returns a new x with 'flat spots'

    """
    if N_length==NO_OVERLAY:
        return x

    cum_x=x.cumsum()
    mav_x=pd.rolling_mean(cum_x, N_length)
    filter_x=pd.TimeSeries([isbelow(cum_x, mav_x,
idx) for idx in range(len(x))], x.index)

    ## can only apply with a lag (!)
    filtered_x=x*filter_x.shift(1)

    return filtered_x
```

Which criteria?

A nice way of thinking about equity curve trading is that it is a bit like buying insurance, or in financial terms a put option on your system's performance. If your system does 'badly' then the insurance policy prevents you from losing too much.

One of my favourite acronyms is TINSTAAFL. If we're buying insurance, or an option, then there ought to be a cost to it. Since we aren't paying any kind of explicit premium, the cost must come in the form of losing something in an implicit way. This could be a lower average return, or something else that is more subtle. This doesn't mean that equity curve trading is automatically a bad thing - it depends on whether you value the lower maximum drawdown* more than the implicit premium you are giving up.

** This assumes we're getting a lower maximum drawdown - as we'll see later this isn't always the case.*

I can think of a number of ways of evaluating performance which try and balance risk and reward. Sadly the most common, Sharpe Ratio, isn't appropriate here. The volatility of the equity curve with the overlay on will be, to use a technical term, weird - especially for large N. Long periods without returns will be combined with periods when return standard deviation is normal. So the volatility of the curve with an overlay will always be lower; but it won't be a well defined statistic. Higher statistical moments will also suffer.

Instead I'm going to use the metric *return / drawdown.* To be precise I'm going to see what effect adding the overlay has on the following account curve statistics:

- Average annual return
- Average drawdown
- Maximum drawdown
- Average annual return / average drawdown
- Average annual return / maximum drawdown

(*Note that return / drawdown is a good measure of performance as it is 'scale invarient'; if you double your leverage then this measure will be unchanged)*

My plan is to generate a random account curve, and then measure all the above. Then I'll pass it through the equity overlay, and remeasure the statistics.

Finally just to note that for most of this post I won't be considering costs. For small N, given we are closing our entire strategy down and then restarting it potentially every week, these could be enormous. Towards the end I will give you an idea of how sensitive the findings are to the costs of different trading instruments.

Which equity curve characteristics?

Broadly speaking the process for using random data is:

- Identify the important characteristics of the real data you need to model

- Calibrate against some real data
- Create a process which produces random data with the neccessary characteristics
- Produce the random data, and then do whatever it is you need to do

Notice that an obvious danger of this process is making random data that is 'too good'. In an extreme case with enough degrees of freedom you could end up producing 'random' data which looks exactly like the data you calibrated it against! There is a balance between having random data that is realistic enough for the tests you are running, and 'over calibrated'.

Identification

What characteristics of a trading system returns will affect how well an equity curve overlay will work? As in my previous post I'll be producing returns that have a particular volatility target - that won't affect the results.

They will also have a given expected Sharpe Ratio. With a negative Sharpe Ratio equity curve overlay should be fantastic - it will turn off the bad system. With a high positive Sharpe Ratio they will probably have no effect (at least for large enough N). It's in the middle that things will get more interesting. I'll test Sharpe Ratios from -2 to +2.

My intuition is that *skew* is important here. Negative skew strategies could see their short, sharp, losses reduced. Positive skew such as trend following strategies which tend to see slow 'bleeds' in capital might be improved by an overlay (and this seems to be a common opinion amongst those who like these kind of systems). I'll test skew from -2 (average for a short volatility or arbitrage system) to +1 (typical of a fast trend following system).

Finally I think that *autocorrelation* of returns could be key. If we tend to get losses one after the other then equity curve trading could help turn off your system before the losses get too bad.

Calibration

First then for the calibration stage we need some actual returns of real trading systems.

The systems I am interested in are the trading rules described in my book, and in this post: a set of trend following rule variations (exponentially weighted moving average crossover, or EWMAC for short) and a carry rule.

First skew. The stylised fact is that trend following, especially fast trend following, is positive skew. However we wouldn't expect this effect to occur at frequencies much faster than the typical holding period. Unsurprisingly daily returns show no significant skew even for the very fastest rules. At a weekly frequency the very fastest variations (2,8 and 4,16) of EWMAC have a skew of around 1.0. At a monthly frequency the variations (8,32 and 16,64) join the positive skew party; with the two slowest variations having perhaps half that.

Carry doesn't see any of the negative skew you might expect from say just fx carry; although it certainly isn't a positive skew strategy.

There is plenty of research showing that trend following rules produce returns that are typically negatively autocorrelated eg http://www.valuewalk.com/2015/09/autocorrelation-of-trend-following-returns-illusion-and-reality/ and http://www.trendfollowing.com/whitepaper/newedge.pdf. The latter paper suggests that equities have a monthly autocorrelation of around +0.2, whilst trend following autocorrelations come in around -0.3. Carry doesn't seem to have a significant autocorrelation.

My conclusion is that for realistic equity curves it isn't enough just to generate daily returns with some standard deviation and skew. We need to generate something that has certain properties at an appropriate time scale; and we also need to generate autocorrelated returns.

I'll show the effect of varying skew between -2 and +1, autocorrelation between -0.3 and +0.3, and Sharpe Ratio between -1 and +2.

How do we model?
In the previous post I showed how to generate skewed random data. Now we have to do something a bit fancier. This section is slightly technical and you might want to skip if you don't care where the random data comes from as long as it's got the right properties.

The classic way of modelling an autocorrelated process is to create an autoregressive AR1 model* (note I'm ignoring higher autoregression to avoid over calibrating the model).

** This assumes that the second, third, ... order autocorrelations follow the same pattern as they would in an AR1 model.*

So our model is:

r_t = Rho * r(t-1) + e_t

Where Rho is the desired autocorrelation and e_t is our error process: here it's skewed gaussian noise*.

** Introducing autocorrelation biases the other moments of the distribution. I've included*

corrections for this which works for reasonable levels of abs(rho)<0.8. You're unlikely to see anything like this level in a real life trading system.

This python code shows how the random data is produced, and checks that it has the right properties.

Now, how do we deal with different behaviour at different frequencies? There are very complicated ways of dealing with this (like using a Brownian bridge), but the simplest is to generate returns at a time scale appropriate to the speed of the indicator. This implies weekly returns for carry and fast EWMAC rules(2,4 and 4,8); and monthly for slower EWMAC rules. If you're trading a very fast trading rule then you should generate daily data, if you see a clear return pattern at that frequency.

I'll use daily returns for the rest of this post, but I've checked that they still hold true at weekly and monthly frequencies (using equity curve filter lookbacks at least 3 times longer than the frequency of returns we're generating).

Results

To recap I am going to be TESTING different lookbacks for the equity curve filter; and I am going to be GENERATING returns with skew between -2 and +1, autocorrelation between -0.4 and +0.4, and Sharpe Ratio between -1 and +2.

I'll keep standard deviation of returns constant, since that will just change the overall scale of each process and not affect the results. I'll look at the results with daily returns. The results won't be significantly different with other periods.

All the code you need is here.

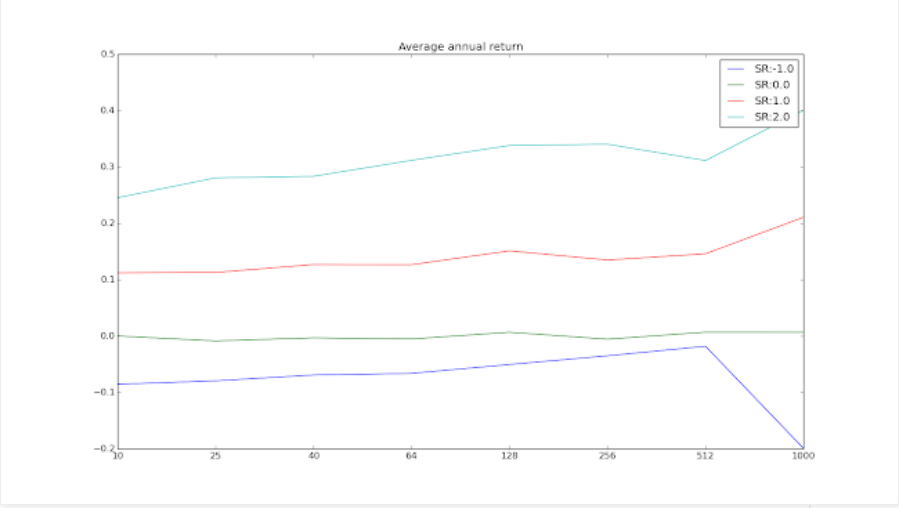
Sharpe Ratio

In this section I'll be varying the Sharpe Ratio whilst keeping the skew and autocorrelation fixed (at zero, and zero, respectively).

```
scenario_type="VaryingSharpeRatio"
period_length=1 ### daily returns
```

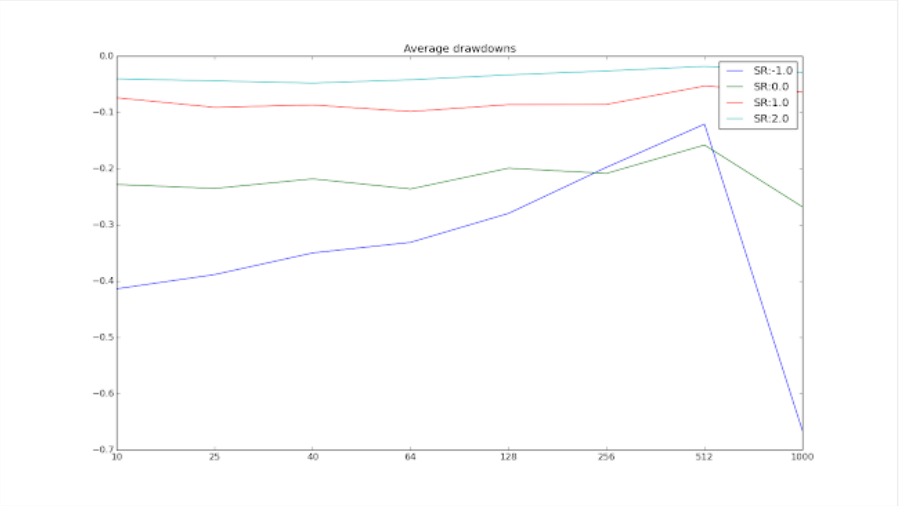
Average annual return

All of the plots that follow have the same format. Each line shows a different level of return characteristic (Sharpe Ratio in this case). The x axis shows the equity curve filter N day count that we're using for the moving average. Note that N=1000, which is always on the right hand side, means we aren't using a filter at all. The y axis shows the average value of the statistic of interest (in this case average annual return) across all the random equity curves that we generate and filter.



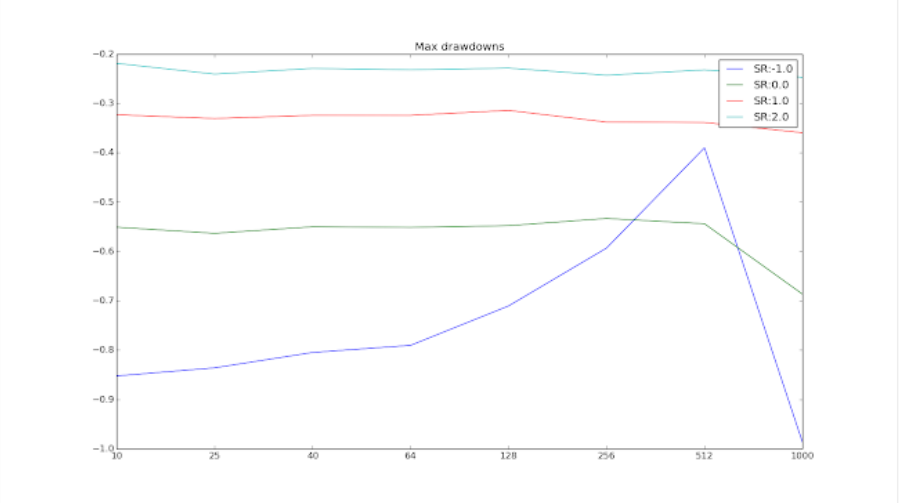
The good news is if you know you're trading system is rubbish, then applying an equity curve system, preferably with large N, improves the performance. If you knew your system was rubbish then of course rather than use a complicated filter to turn it off you wouldn't bother turning it on at all! However for all profitable equity curves equity curve trading reduces, rather than increases, your returns.

Average drawdown



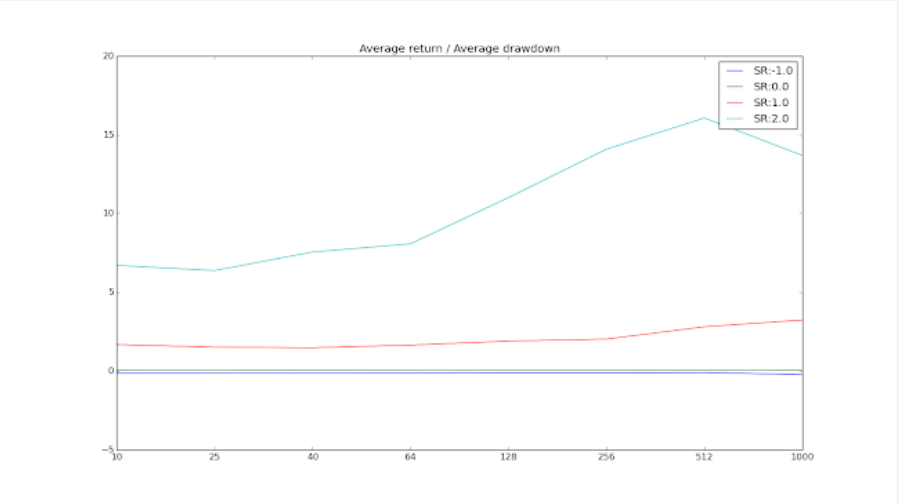
Again for systems which break even or lose money the average drawdown is lower with an equity curve trading system, as you might expect; again especially for large N. However for profitable systems there is no benefit, and average drawdowns may even be slightly worse.

Maximum drawdown



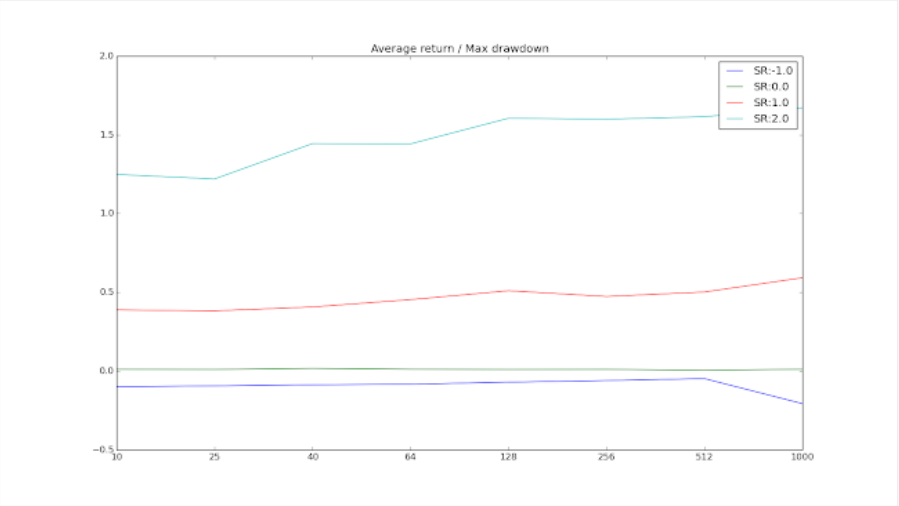
For profitable systems there might perhaps be a modest reduction in maximum drawdown for small values of N. For loss making systems the biggest reduction in drawdown is for large values of N; although all filters are better than none.

Average annual return / average drawdown



Let's now try and put together the return and drawdown into a simple statistic. For unprofitable systems the overlay makes no difference. For profitable systems it reduces the drawdown adjusted return (the 'hump' at the right hand side of the SR=2.0 line is an artifact caused by the fact we can't calculate this statistic when the average drawdown is zero).

Average annual return / maximum drawdown



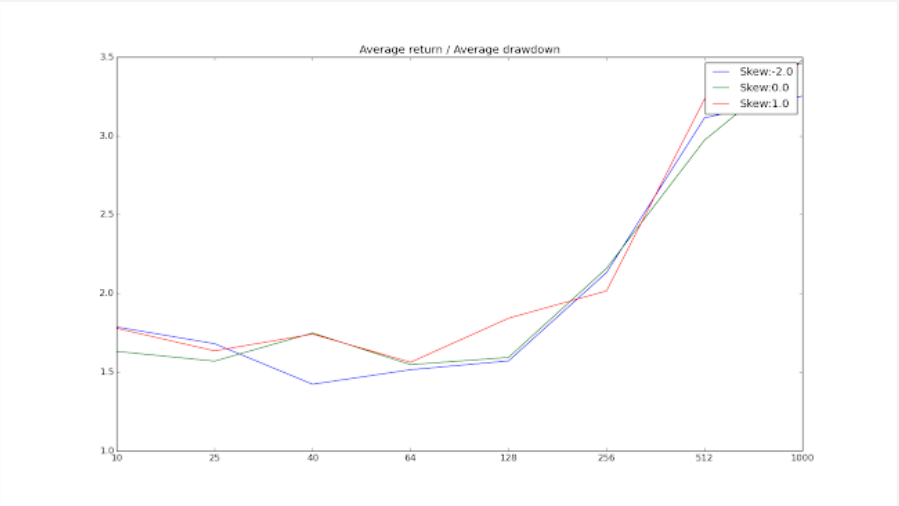
This statistic tells the same story. For a profitable system applying an equity curve overlay reduces the average return / max drawdown ratio; with faster overlays (small N) probably worse (and they would be much, much worse with trading costs applied). If you have a system that definitely loses money then applying an overlay, of any lookback, will mitigate your losses.

Skew

In this section I'll be varying the Skew whilst keeping the Sharpe Ratio and autocorrelation fixed (at one, and zero, respectively).

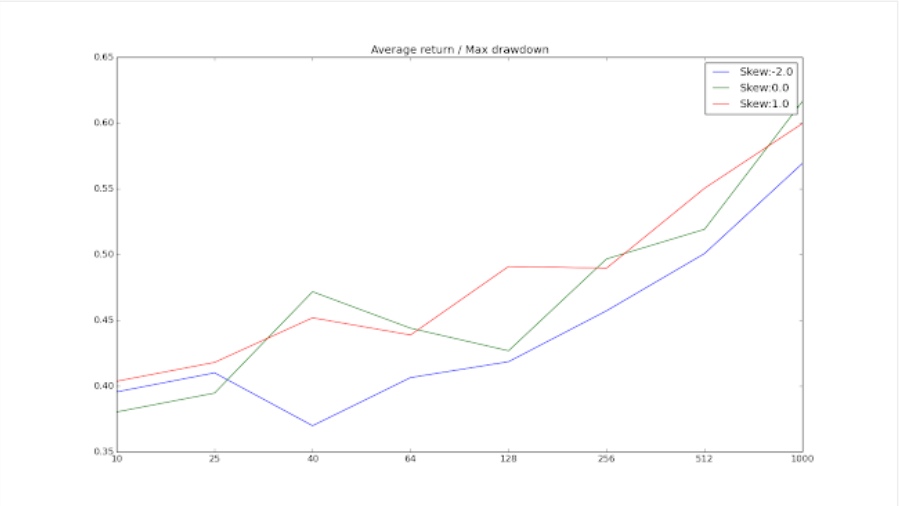
```
scenario_type="VaryingSkew"
period_length=1 ## daily returns
```

Average annual return / average drawdown



To save time let's jump ahead to the calculated statistics. Bluntly my intuition was wrong; skew makes no difference. The overlay harms returns for all skew values shown here.

Average annual return / maximum drawdown



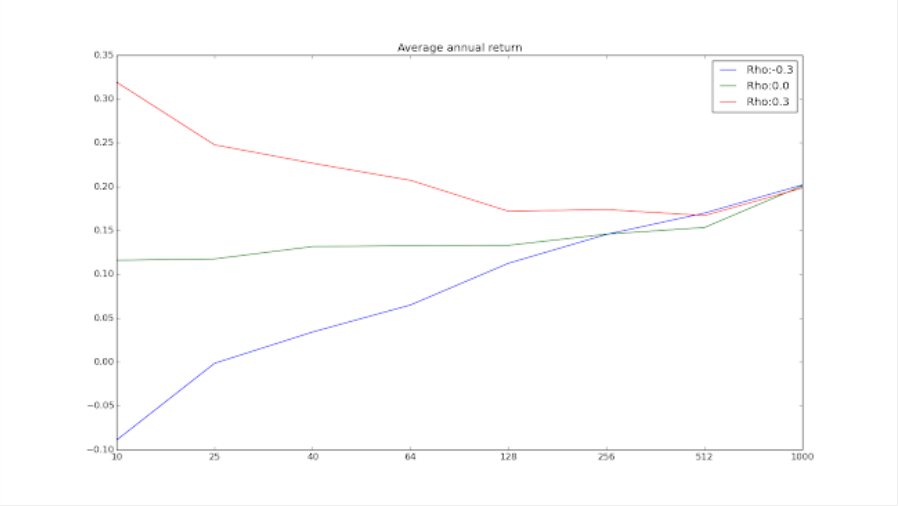
There is a similar story for return / max drawdown.

Autocorrelation

In this section I'll be varying the Autocorrelation whilst keeping the skew and Sharpe Ratio fixed (at zero, and one, respectively).

```
scenario_type="VaryingAuto"
period_length=1
```

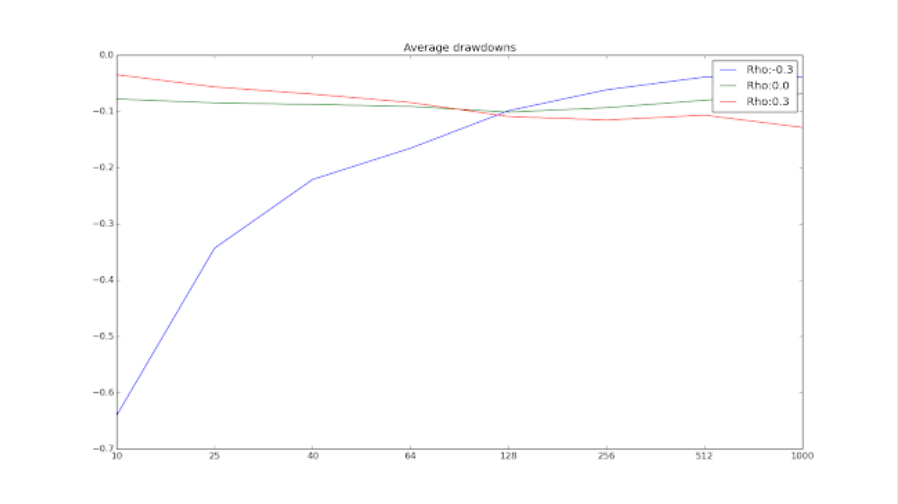
Average annual return



Well hang on... we have a result. If you have negative or zero autocorrelation then adding an equity curve overlay will make your returns much worse. But if you have positive autocorrelation it will improve them, with faster overlays doing best (remember we're ignoring costs here). This makes sense. After all if something has positive autocorrelation then we'd want to trend follow it.

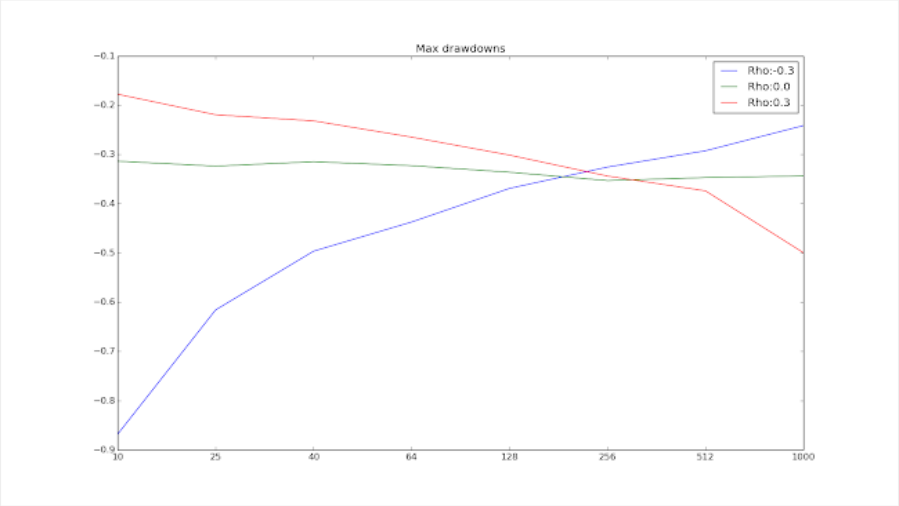
However as we've already discussed trend following systems seem to have negative autocorrelation. So it looks like equity curve overlays are a no-no for a trend following system.

Average drawdown



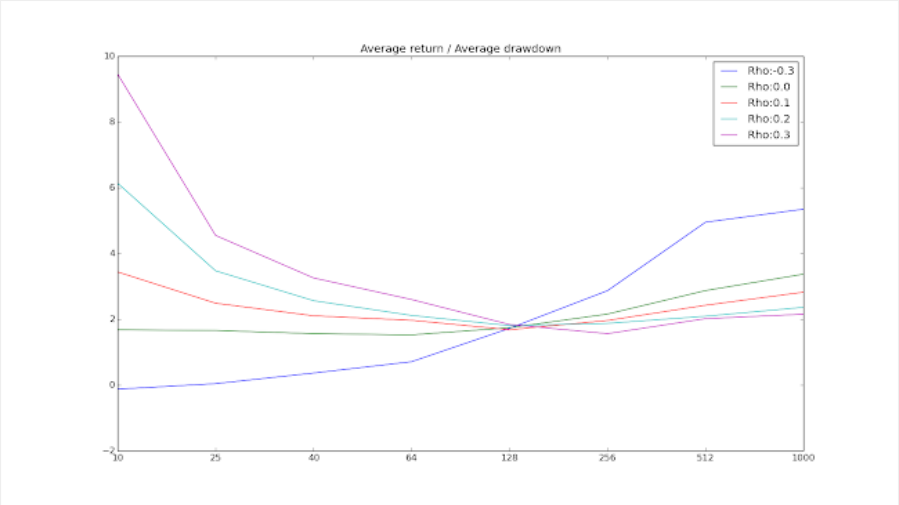
Again drawdowns are improved only if the autocorrelation is positive. They are much worse if it is negative (note that average drawdowns are smaller for negatively autocorrelated systems anyway).

Maximum drawdown



There is a similar picture for maximum drawdown.

Average annual return / average drawdown



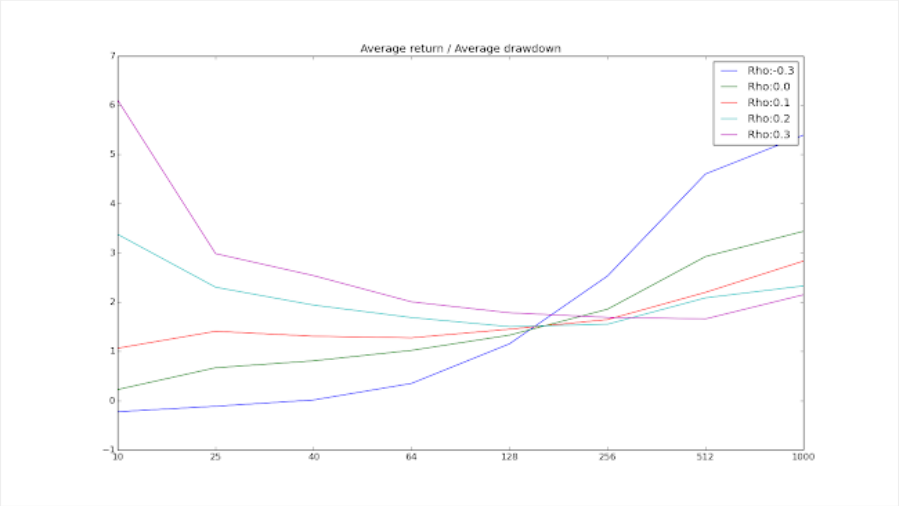
I've added more lines to this plot (`scenario_type="VaryingAutoMore"`) to see if we can find the 'break even' point at which you should use an equity curve overlay. It looks like for autocorrelation of 0.2 or greater, with an N length of 40 days or less.

Remember that I haven't included costs in any of these calculations. For information the annualised turnover added to each system by the equity curve filter ranges from around 23 for N_length 10 to 1.2 for N_length 512. With the cheapest futures I trade (standardised cost of 0.001 SR units per year, for something like NASDAQ) this is not a major problem, reducing average return with N length of 10 by around 1%.

** See chapter 12 of my book for details of how I calculate turnover and standardised costs*

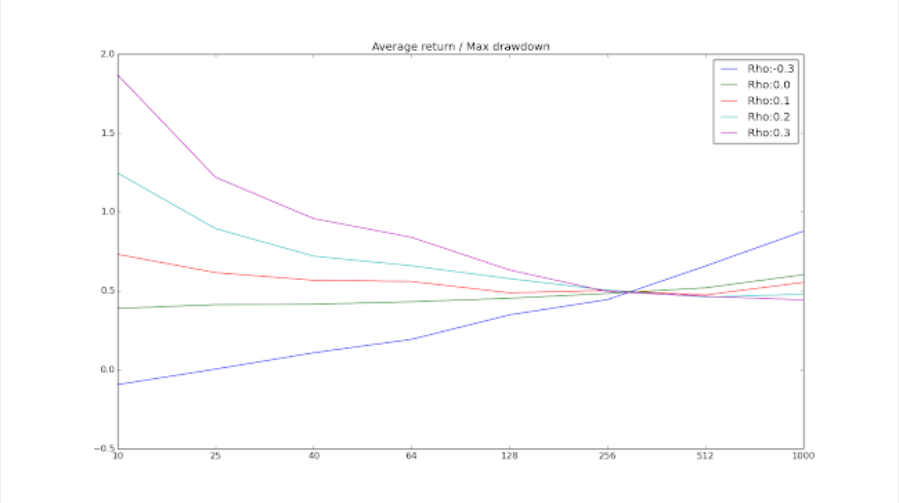
However let's see the results using a more expensive future, the Australian interest rate future, with a cost of around 0.03 SR units per year.

```
scenario_type="VaryingAutoMore"
period_length=1
annualised_costs_SR=0.03
```

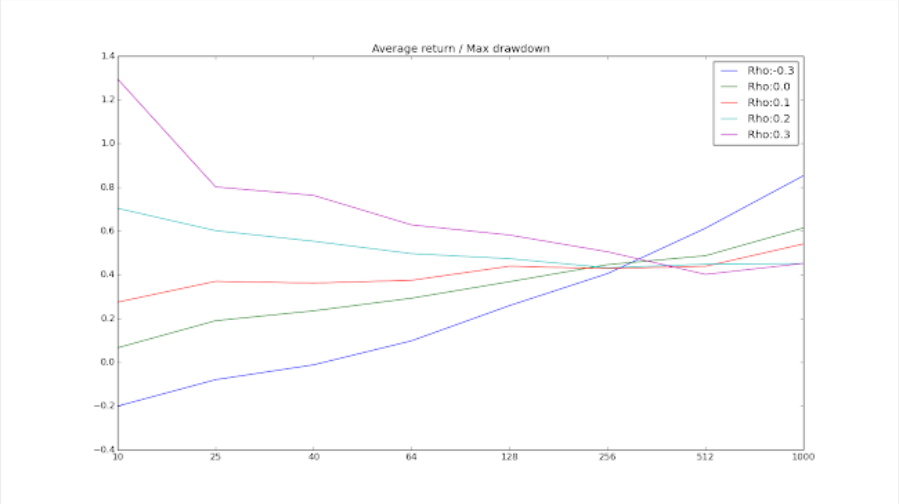


With costs smaller N look much worse. I'd suggest that at this cost level you need a positive autocorrelation of at least 0.2 before even considering trading the equity curve.

Average annual return / maximum drawdown



As before it looks like an autocorrelation of 0.1 or more will be enough to use equity curve trading; but if we apply costs as before we get this picture:



... and again only a higher autocorrelation will do.

Conclusion

The idea that you can easily improve a profitable equity curve by adding a simple moving average filter is, probably, wrong. This result is robust across different positive sharpe ratios and levels of skew. Using a shorter moving average for the filter is worse than using a slower one, even if we ignore costs.

There is one exception. If your trading strategy returns show positive autocorrelation then applying a filter with a relatively short moving average will probably improve your returns, ***but only if*** your trading costs are sufficiently low.

However if your strategy is a trend following strategy, then it probably has ***negative***

autocorrelation, and applying the filter will be an unmitigated disaster.

This is the second post in a series on using random data. The first post is here. The next post on portfolio optimisation is here.

Posted by Rob Carver at 09:59

Labels: Python, Random data, Statistics, Trend following, Uncertainty

7 comments:

Darrin 10 November 2015 at 14:41

Another solid post, Rob. Why do you prefer SR over the return/drawdown measure that was used in this post? Throughout the book, you use the SR to standardized risk but isn't the ret/dd ratio a more specific measure of that?

Reply

Rob Carver 10 November 2015 at 14:59

I can think of a few reasons:

Intepretability

I have been using Sharpe Ratios a long time, and I don't really have a feel for other performance statistics. If you told me something had a calmar ratio of 0.5 I wouldn't have a clue what you were talking about, and even if you explained that meant the average annual return to max drawdown was 0.5 I still wouldn't know if that was good or not.

Consistency

I create forecasts as proportional to expected returns scaled for standard deviation, which is basically a Sharpe Ratio. I also calculate risk for position scaling by using the standard deviation. Consistently using standard deviation as a measure of risk has its flaws, but at least I'm using the same measure throughout my system.

Symmettry:

Although the downfall of SR is that it assumes symmettry, I actually think it is a good thing to be as scared of an expected rise as a fall.

Statistical robustness:

When we calculate the standard deviation we use the whole distribution. With average drawdown we don't and with maximum we use only a single point. Return / max drawdown in particular is a terrible statistic to use with real data, since it is very much random what it comes out at.

Reply

Matt Haines Photography 18 November 2015 at 05:15

I've tried using a simple MA on win rates for several mean-reversion systems, and the results have always been dismal. Except for that one time I forgot to check for a future leak, and that one worked like a charm! 😀 The problem, as I saw it, is that these systems all had a very high win rate, 60-75%, and so losing trades were few and far between. There was no clustering to speak of, so the switching on and off merely removed good grades.

-Matt

Http://www.throwinggoodmoney.com

Reply

Replies

Rob Carver 18 November 2015 at 06:23

Interesting. I guess a system where you cut after the drawdown reached X% might protect you from a drawdown larger than you ever saw in simulation; i.e. the situation when the MR relationship breaks down.

Matt Haines Photography 18 November 2015 at 16:49

Thanks Rob, I'll have to look into that. Haven't tried anything using the drawdown as a filter. I suspect that it might not work, but my intuition is usually wrong about trading. Which is why I prefer robots and computers. These sort of of shortish MR trades tend to have many smaller wins, with the occasional invigoratingly large loss. A system that shuts off on drawdown thresholds might simply reduce the amount of little wins needed to scrape back into the black. But until I've tried it, I won't know.

Thanks for your comments and your blog. I've added it to my regular reading list.

Reply

Chad B 29 October 2018 at 17:35

Hi Rob,
Supposing I have a list of transactions in CSV format from running a strategy in a different programming language, i.e. strategy returns, which script in PST should I target for feeding these in? Forecasting.py?

Reply

Replies

Rob Carver 29 October 2018 at 18:00

Yes, assuming you are only interested in whether the other strategy produces longs or shorts, which you would then translate into forecasts of +10 and -10

Reply

Enter your comment...

Comment as: handsomehu1 ▾ Sign out

Publish Preview Notify me

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

Contact Me (Spam will be politely ignored)

Name

Email *

Message *

Send

Follow by Email

Email address... Submit

Subscribe To

Posts

Comments

Simple theme. Powered by Blogger.