This document is for informational purposes only and does not constitute financial advice. Trading involves risk, including potential loss of capital. Past performance is not a guarantee of future results. Consult a licensed financial advisor before making investment decisions.

This document serves as a brain and information dump and most of the methodology/ideas outlined here were not originally developed by myself.

IDE to use: VSCode

Choosing a Trading Strategy

- Here are 5 potential "strategy bases" for algorithmic trading:
    1. Arbitrage: Exploiting price differences between markets for the same asset (buying an asset for a lower price in one market and selling it in another market for higher) (for example, garage sale flippers that buy something for cheap at a garage sale and then sell it on online for more)
    2. Relative pricing: This strategy is great for two products with casual relationships (e.g., the price of apple and the price of a carton of apple juice)
        - Common pairs that are traded against each other include ETFs and their inverse counterpart, a particular stock against an ETF that contains that stock, or synthetic options structures
    3. Correlations: Trading based on mutual connections between two assets. There is often no intuitive reason for the correlation, but it still works.
    4. Mean Reversions: Trading based on values that fluctuate around a mean because you expect the current value to revert to the mean (can be used for pairs trading, statistical arbitrage, etc.) (For example, trading based on the mean of BTC vs ETH dominance after normalizing the data set that measures the dominance)
    - When thinking of an idea you're essentially developing an economic hypothesis that the world works in a certain way
        - Rather than studying trading strategies, study financial phenomena. Pretend you are a professor looking for phenomena and trying to understand why they happen.
            - If you don't understand the market, you're likely just data-mining and overfitting.

- Decide the level of liquidity to trade in.
    - If you're not an institutional investor/whale, you may be able to find more of an edge in lower liquidity markets that they don't trade
    - Keep in mind that order execution is easier in high liquidity markets and slippage is reduced in these markets

- Mold idea into a trading strategy.
    - Keep the strategy simple.
        - Less parameters means less chance of messing up code, and also reduces the chance of overfitting.
            - Better to explain 60% of the data with 2-3 parameters than 90% with 10.
        - There are lots of bells and whistles you can add to a simple strategy later on to get more of a competitive advantage.
        - A simple strategy also helps avoid analysis paralysis.

- Pricing data alone may not be enough to base a strategy on anymore, since much of what can be found there has been found.

Elements of a Trading Strategy (explained further below)
- Entry and exit rules
- Risk management
- Position sizing
- Time frame

- If you instantly jump to optimizing these with real data, you greatly increase the risk of overfitting.

Entry and exit rules:
- If you are trading in low-liquidity markets, define a minimum order-book depth for position entry so that slippage will be manageable.
- Determine what will systematically serve as the buy and sell signal.
    - Keeping this simple reduces the chances of overfitting.
- Also consider the type of exit order you will use to "protect profits".

- Furthermore, consider if profit targets are relevant to your strategy.

Position Sizing:
- Sizing rules have a big impact on overall portfolio performance, but efficient sizing is difficult.
    - One of the challenges to position sizing are fat-tailed distributions that cause unpredictable anomalies (known as black swans).
    - Many statistical measures are also ineffective for position sizing.

- That being said, it's important to consider and can make or break a trading system.

Algorithm-Specific Risk Management

- Determine your risk management strategies
    - A good strategy can be formed solely by trying to remove the bad trades in a strategy (but make sure to not overfit)

- Consider implementing the use of short-term performance to adjust your risk management parameters as your algorithm moves along.

- Make sure to compare your model's performance to the performance of risk factors such as the five Fama-French factors (and the overall market risk) to see what risk it's most exposed to, if any. If it's over-exposed to certain risk factors, consider hedging against that risk.
    - Keep in mind that you can't avoid all risk factors and the ultimate goal is for your portfolio to be as risk independent as possible, rather than each individual model.

Cost Considerations

- After designing the strategy, you should be able to get a good idea of what the trading costs associated with that strategy are. If you determine that the strategy is relatively very expensive, consider throwing it away.
    - Costs consist of exchange fees + slippage

Slippage

- In backtesting, databases often record the opening price (e.g. first trade of the day) or the midpoint price of the opening range (the opening time period of trading). Similarly for the closing price, the database assumes the exact recorded price at market close.
    - In reality if you place a market order at the open, it will likely be filled somewhere within the opening range of prices, not necessarily the exact opening price. The same thing happens with market orders at close. This is known as slippage.
        - Cumulative slippage can lead to reduced profits/increased losses in live trading compared to backtesting results.

Historical Data for Backtesting

- Find the data that you need to execute the strategy.

- Check data integrity
    - Make sure to avoid survivorship bias (for example, some assets go to 0 or are delisted, but these aren't reflected in every data set)
    - If you're trading indexes, make sure backtesting data includes the index's historical constituents, not just the current ones.

    - Avoid look-ahead bias by only testing datasets with assets that were successful in the future.
    - The historic data should contain a variety of volatility levels and market regimes
    - Be explicit and honest with yourself about data assumptions you use

- Choose as large of a time window as possible while maintaining relevance to the timeframe and execution of your strategy.
    - If your strategy is short-term, use recent data for backtesting.
    - Time window suggestions:
        - Short term strategy window: 1-2 years
        - Intermediate: 2-4 years
        - Long term: 4-8 years

- Split up your time window into samples dedicated for backtesting/optimization and samples dedicated to the walk forward analysis.
    - The walk forward windows should be ¼ or ⅓ of the length of the time windows used for optimization.
    - My method for splitting windows in the case that data tests need to be run before testing, for example for pairs trading. This is to avoid look ahead bias as much as possible:
        - First third of data is for running data tests (e.g., verifying cointegration of a pair)
        - Second third of data is for backtesting
        - First and second third together are then to be used for optimization
        - The last third is to be used for the walk forward test
    - Make sure there are gaps between the end of your optimization windows and the start of your walk forward analysis windows
    - Don't conduct this preliminary backtest or the optimization process on the walk forward samples. Only touch those samples when you start the walk forward analysis.

Conducting the Backtest

- Implement a feature that accounts for trading fees during the backtest

- If you're running a short-term strategy, implement a feature that excludes trades on days of major events (unless you're running an event-driven system).
    - Massive price slippage occurs around these events
        - Overall though, your decision regarding major economic events should depend on strategy type and trading frequency.
    - Examples of such events are Fed interest rate announcements. Research major events that are relevant to the asset you're trading and account for them in your trade execution.

- For a strategy that isn't for very high liquidity markets, set up a few different backtests that assume different liquidity conditions.
    - Also for such markets, have your algorithm make sure there is enough order book depth before buying to accommodate your buy order without slippage.
        - These will show if your strategy will fall apart the moment liquidity is reduced.

- IF your strategy is meant to work across markets, test it in a basket of fundamentally diverse markets.

Evaluating Backtest Results

- Plot the data of your back test. First you must evaluate the performance by asking these questions:
    - Does the behavior make sense?
    - Does the trading frequency make sense?
    - Is anything weird going on?
- If you find that there's an issue, go back and re-design the algorithm and/or the code.

- Estimate the profitability of your strategy by calculating the profit/loss for the length of price history that you used.
    - High unusual loss should be viewed as a preliminary warning sign.
        - If this high loss occurred during "normal" market conditions, go back to the drawing board.
    - If the backtest doesn't show at least mild profitability, go back to the drawing board.

- Now compare the preliminary backtest results with the theoretical expectation that you had before backtesting. The two should be in-line.
    - Make sure signals were correctly shown and that orders were correctly executed.
    - If the results deviate dramatically from the theoretical expectation, you must find the cause. If the strategy seems to be flawed, you must redesign, keeping the initial theory in mind.
        - At this stage, it's important to rule out unusual market activity like limit move circuit breakers or economic events as being the cause.

- Check the beta when compared to market returns. If it's 0.3 or less, move on. Above 0.6, abandon the strategy or go back to the drawing board. If it's somewhere in between, implement beta hedging before you optimize.

- Make sure that trades were evenly distributed across the backtesting window.
    - This avoids biases caused by market conditions (e.g. only trading during a bull market will overestimate performance of the algorithm).

        - Develop your own market and volatility regime filters so you can segment time windows more effectively and determine which regimes your algorithm works best and worst in.

- Make sure that the standard deviation for the size of wins/losses and the length for periods of wins/losses is low. If not, go back to the drawing board.

- You should also segment the price history as it is a statistically sound decision; divide the backtesting period into an equal number of small intervals.
    - Small profits and losses that are randomly distributed across the time frames indicate acceptable performance. Otherwise, go back to the drawing board.

- If you tested the strategy across different markets:
    - Good performance across different markets is a good sign of a robust strategy
    - Bad performance across markets is a sign to consider abandoning the strategy (if it was designed with multiple markets in mind)

- At this stage, you're only determining if the strategy is worthy of further development. If it passes all the above tests, move forward.

Optimization

Optimization is the process of adjusting certain parameters in a strategy to achieve the best results with the strategy.

- This is done with a search function/objective function that searches for the best values of parameters to meet your objective (e.g, maximizing profit, minimizing risk, etc.) by repeatedly conducting backtests with various combinations of parameters.

Four key decisions must be made to set up your optimization framework (explained further below):

1. Variables (parameters that can be tuned) and scan ranges (the range of possible parameter values and the increments between those values)
2. An adequate data sample must be selected
3. The search function used to identify the best parameters
4. Guidelines by which you'll evaluate the optimization

<u>1: Determine the parameters to optimize (and their scan ranges)</u>

- Focus on optimizing parameters that affect risk management and capital allocation

- Identify key variables that impact the strategy the most, and exclude others from the optimization process.
    - If you don't know the significance of each variable, systematically test each one across a certain scan range while keeping the other variables constant. Do this one by one to measure their impact on results and determine which ones are most significant.

- Choose a small number of parameters (relative to the total amount of parameters for your algorithm) to optimize. The optimized parameters should be the ones that you determined are most significant.
    - This is to avoid overfitting.

- Now determine the range of values to test and the increment size between values for each parameter.
    - Consider processing power when determining this. A wider range and smaller increments requires more processing power and time.
    - The step size-to-range ratio should be the same (or very similar) across all variables being optimized. An example is in your green notebook.
        - It's not always easy to find the best step sizes, but it's important to attempt to do so and keep the step sizes proportional to each other.

2: Select an adequate data sample:

- Use the same sample for optimization as you used for the initial backtesting.

3: Determine the search method to use:

- The search method used determines the amount of time the optimization process takes.
    - An "intelligent" search method can be used to reduce the sources needed.

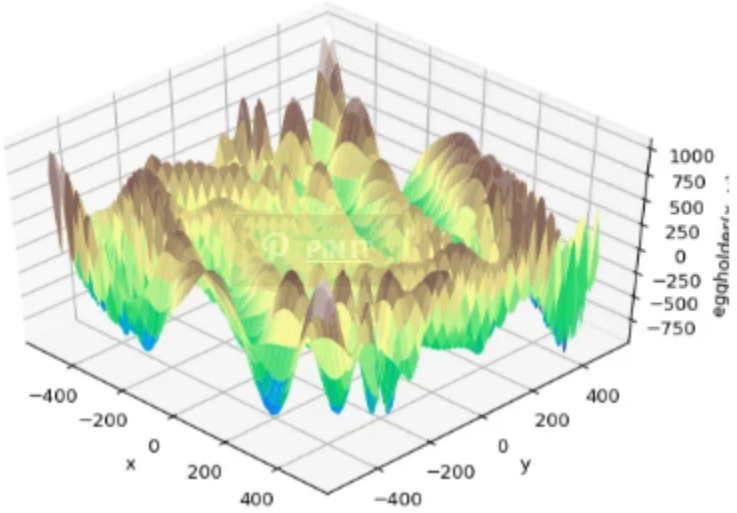
- The types of search functions you can use for optimization can be found here.

4: Guidelines for evaluating the optimization:

- Successful optimization would yield a net profit above 0 and a maximum drawdown less than 20% (in testing).
    - If the maximum drawdown is much larger than the average drawdown, you should investigate what caused it (could be market conditions, unexpected events, etc.). Keep in mind that market distributions of returns are fat tailed, and risk is one of the major costs in trading. You must take into account anomalous events because they will happen during live trading.
        - If your optimized strategy doesn't pass these tests and you aren't able to reduce the risk exposure, go back to the drawing board.

- A robust and consistent strategy is defined by the following attributes. Make sure your optimized algorithm aligns with them. If not, back to the drawing board:
    - It generates alpha and is profitable on a significant proportion of parameter sets
        - The distribution of performance throughout the parameter combination sets should be even with small variations. If you have a "spiky" optimization set, where little changes in parameters create sharp/unpredictable changes in the performance of a model or system, seriously consider reworking your idea, as this means the search

methods are inherently invalid. A “spiky” optimization set is displayed in the graph below.



- A greater number of profitable variable sets (over 50%) increases the likelihood that results are statistically significant (examples of good optimization profiles can be found here)
- Additional good signs to check for which indicate robustness:
    - The optimal parameter set’s performance is close (within 1 standard deviation) to the average performance of all parameter sets
    - The average simulation result minus one standard deviation still generates alpha

- It also generates alpha across different market trends…
- … in a majority of different historical time periods…
- … and in different volatility regimes and unexpected events (think of fat-tailed distributions and black swan events).

- If the above guidelines have been fulfilled, make sure the strategy meets these characteristics:
    - An even distribution of when the trades are executed.
    - An even distribution of trading profits.
    - A balance between long and short profits.
    - A large sample size of trades were executed. You should have at least 30 trades in your data, but ideally you’d have a **much** larger amount.
    - A balance between winning and losing runs (in terms of length and frequency)
    - The optimized strategy operates within acceptable risk limits (good Sharpe ratio for the duration of the strategy’s use) and performs well under adverse conditions.

- Another key indicator of a good strategy is that during periods of low volatility, the strategy performance decreases but there is no large dip in the equity curve.

Walk Forward Analysis

- A walk forward analysis is a method of testing a strategy by testing the optimized strategy on historical data that wasn't included in the sample of data used for optimization. (You should've set aside that historical data when performing the initial preliminary backtest)
    - The walk forward analysis answers 4 questions:
        - Is the system robust and can it generate alpha in real-time trading?
        - What rate of return is to be expected?
        - How will changes in market behavior affect performance?
        - What is the best variable set for real time trading?

    - The primary end goal is to evaluate if the strategy is robust and can be repeated, or rather the result of overfitting. Walk forward analysis is the only statistically reliable way to gauge a strategy's performance.

- Perform a walk forward test on a few of the top strategies from the optimization phase. Their performance should not stray drastically from each other during the Walk Forward Analysis. If they do, it is reasonable to conclude that your algorithm is overfit and you should go back to the drawing board.

- Strategies that have at least 50-60% Walk Forward Efficiency (WFE) can be considered robust. Anything lower indicates an overfit strategy and it shouldn't be traded.
    - WFE = ((Out-of-sample performance)/(In-sample performance)) * 100%

- If a strategy loses money during the walk-forward test, you do not have a tradable strategy.

- The bigger the risk-to-reward ratio (RRR), the better.
    - RRR = Net Profit / Maximum Drawdown.
        - In general, it should be at least 3.

Determining Starting Account Size

The effective formula to analyze a strategy's risk is the maximum drawdown in backtesting multiplied by a margin of safety multiplier (~1.5).

- Extend this concept by adding margin requirements as a factor for the risk analysis (margin requirement is the minimum amount of money your broker requires you to have in your account for you to trade).
    - This means that your required starting account size for live trading will be equal to the (maximum drawdown times the safety multiplier) + margin required by broker
        - Ex: $15,000 max drawdown x 1.5 + required margin of $13,100 = $35,600

        - To be even more conservative, multiply the margin requirement by two (in this example, $13,100 x 2 = $26,200, and the total starting account size will equal $48,700)
        - This process allows you to weather to extreme drawdowns (worst-case scenario) and continue trading

Live Trading

- Start live trading with a very small amount of capital just to test the proceedings and make sure everything works.
    - Make sure live behavior is the same as behavior during testing
    - Make sure that the algorithm executes efficiently
    - If a trading strategy produces drawdowns in real time that exceed drawdowns during testing for optimization/walk forward analyses, pause the algorithm and consider why this may be happening.
    - Make sure trading costs or market impact don't destroy your strategy.
    - Don't abandon a strategy solely based on underwhelming short-term performance.

- Before live trading, define criteria under which you'll shut down the live system and implement this as a hard limit/stop loss, but make sure it isn't within any of the ranges of drawdowns that you saw during testing

- Don't wait too long between the points that you finish testing and begin real live trading.

- Re-evaluate the system and its performance after lengths of time corresponding to the walk-forward and optimization windows, but don't meddle with the system when it's live. Remember that there was no meddling during the backtest. Don't look at your computer a lot when the algorithm is trading.
    - If short term performance of a strategy that you're live trading is poor, look into using Constant Proportion Portfolio Insurance to decrease leverage and position size.
        - Don't decide to shut down a strategy solely based on short-term performance. Just decrease the weighting until you have enough data to prove that the strategy has become invalid.
    - When it comes to live trading, a model that is optimized on 2 years data will likely remain stable for up to 6 months. For a 4 year optimization window, usability will likely be up to 1 year, with 2 years being the best-case scenario. But it is important to understand that if your strategy is based on historically-valid and fundamental principles, such as human psychology, it may prevail for longer.

- Don't pocket all your profits. Reinvest profits into data, equipment, and even personnel
    - Run your trading like you'd run a business

- Live trading experience is very important. Try to learn as much as possible from your time spent live trading.

- While your algorithm is live trading, keep researching, developing algorithms, and improving. You want to keep improving and you also want to be prepared for when your current running algorithms experience alpha decay.

Portfolio Risk Management

- Once you develop more and more strategies and accumulate capital, diversifying the algorithms that you are trading, the assets you're trading, and the asset classes that you're trading.

- Portfolios based on mean-variance efficiency and the CAPM aren't good for trading.

- Use risk parity for portfolio optimization (instead of allocating capital evenly, allocate capital based on the risk of that portion of the portfolio)

- Make sure to compare your portfolio's performance to the performance of risk factors such as the five Fama-French factors (and the overall market risk) to see what risk it's most exposed to, if any. Ideally you don't want to be over-exposed to any risk factors.
    - You generally don't want your models to be super risk constrained on their own, but you hope that you can combine models so that the portfolio isn't overexposed to a certain risk. Every single model shouldn't be risk constrained.

- Returns of different aspects/strategies of your portfolio shouldn't be very correlated.

- You can try creating multiple variations of strategies based on the same trading rule to have many models that are uncorrelated

Leverage

- Don't leverage too high when times are good
- Kelly formula is doctor Ernet Chan's favorite risk management tool
    - The Kelly formula only works if returns are Gaussian (they follow a normal distribution)
    - The Kelly formula should only be used as a maximum. It's generally recommended to only use half of the Kelly.
        - A fundamental problem with the Kelly formula is that it takes returns as an input, which you can't truly predict.
        - Don't use short-term performance for computing the Kelly leverage
        - On the other hand, volatility can be easily predicted
- You can utilize volatility predictions to determine leverage (risk parity)
- One should always look at the downside potential when determining leverage, not the upside potential