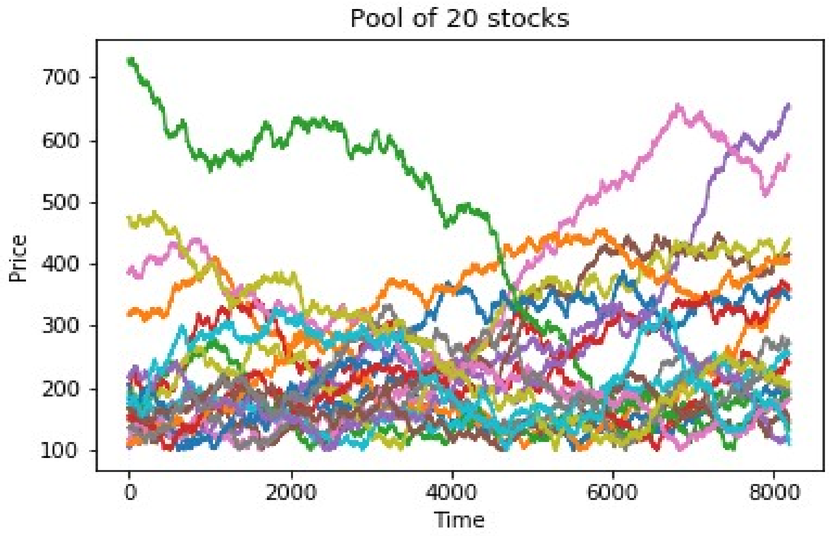
Simulation First Results

Braedyn Au
June 9, 2020

Stocks

- 20 stocks simulated with fBM
- https://www.mathworks.com/matlabcentral/fileexchange/38935-fractional-brownian-motion-generator
- The algorithm I have starts at 0 and moves up and down which can create negative numbers.
- I shifted each stock price up by the minimum price and then added a flat 100 for each stock—how to improve?
- The algorithm can only generate powers of 2 number of datapoints, so to match the 7200 minutes if a 5 day trading week, I generate 2^{13} = 8192 points and set point 992 as time 0 for the simulation. The 900 points before this time are used to find an optimal sharpe ratio for the initial portfolio allocation spread.
- Sharpe ratios at future time steps are then calculated based on the 900 points before that time, similar to a simple moving average.



Sharpe Ratio

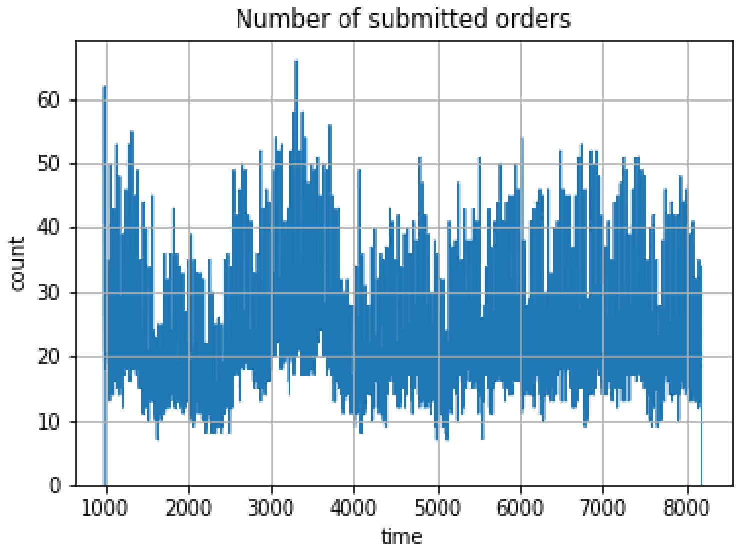
- I find the percentage change between each time step to find the expected return percentage and standard devation of the returns.
- Return negative value for minimization via sequanetial least squareds quadratic programming (SLSQP)
- I’ve seen some calculations use log returns instead of arithmetic, and also normalizing to an annual sharpe ratio. Are these applicable here?

```python
def sharpe(alloc, stocks, vol, ti, tf):
    """
    remake of sharpe calculation following
    https://www.mlq.ai/python-for-finance-portfolio-optimization/#h1sjvcte25p1r8or1e1ngd82h2r8ha1
    uses allocation percentage instead of weights
    """
    Rp = 0
    var = 0
    Rf = 0.010
    for i,j in enumerate(stocks):
        stepReturn = 100*np.diff(stockPool[j][ti:tf])/stockPool[j][ti:tf-1]
        Rp += alloc[i]*vol*np.mean(stepReturn)
        var += alloc[i]*alloc[i]*vol*vol*np.var(stepReturn)
    stdp = np.sqrt(var)

    #print(Rp,stdp)
    if stdp == 0:
        print("dividebyzero")
    return -(Rp-Rf)/stdp
```

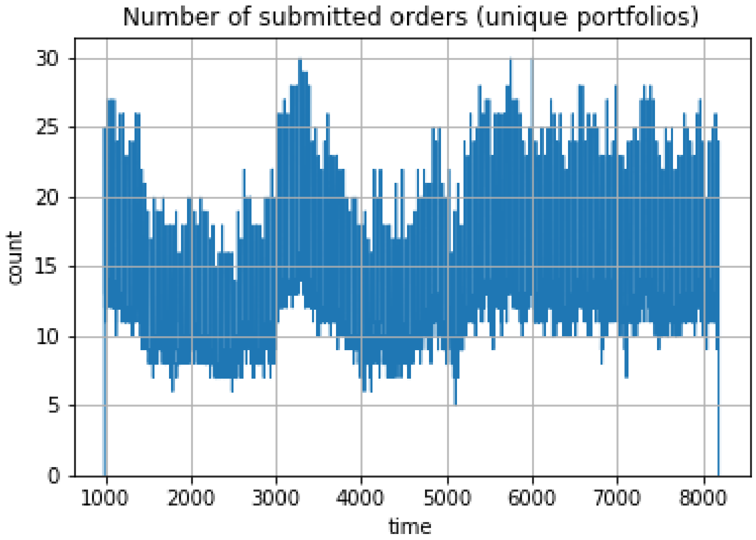
With overlap

- 5 portfolios with a uniform random number of stocks between 8-12, chosen from a uniformly distributed pool of 20 stocks, and uniform random volume chosen from 10^3, 10^4, 10^5, 10^6 dollars.
- Sharpe ratio optimization done through portfolio percentage volume allocation—when the change in a stock's percentage allocation corresponds to at least one whole stock at the price at that given time, an order for that stock is sent through.
- The portfolio's allocation spread is updated to the optimal point at the end of each step.
- Working on function to generate portfolios at specific overlap levels and with non uniform distributions.



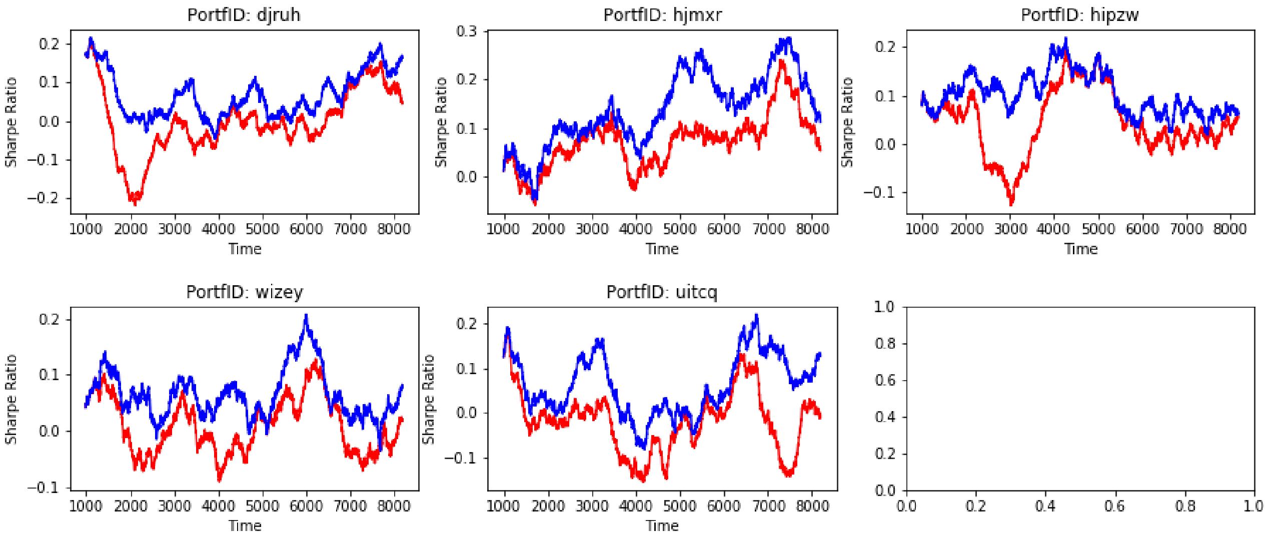
Without overlap

- 5 portfolios each with 4 unique stocks and each with a volume of 10^6 dollars .
- Same dynamics as before.
- In terms of number of stocks and volume, how best to compare to portfolios with overlap?
- How to threshold?



Sharpe comparison

- Compared Sharpe ratio’s of the 5 non-overlapping portfolios when follwing a dynamic portfolio spread (blue) vs a static portfolio spread (red).



Artificial Broker

- I also wrote a broker which would collect orders from traders and match buyers and sellers following a first in first out algorithm.
- Would result in a final list containing every transaction with time of sale, buyer ID, seller ID, stock, volume, and a unique transaction ID.
- Was originally planning on optimzing portfolios by their weights on each stock, not the portfolio percentage allocation. In the dynamics, when the order is generated, stocks with a sell order immediately have their weights changed, but stocks with a buy order don’t have their weights changed until the transaction is complete. The trouble was finding a limit on the weights when optimizing since I did not code an ‘available cash’ variable.
- Realized this wasn’t applicable for non-overlapping stocks, but could be interesting later on.