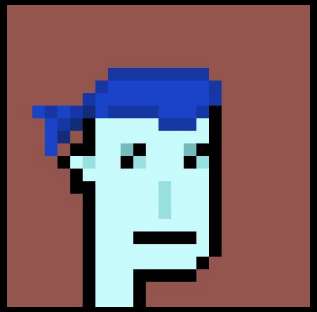
Crypto Trading Desk

@advaykoranne.eth: Desk Head



@Jack65604942: QT

Josh Park: QR

@joshuaahuang: SWE

Elizabeth Tang: QT

Prateek Gautam: QR

Lucas Maley: QR

Agenda

1. Abstract
2. Pairs Trading Strategy
3. Back Test
4. Results
5. Lagged Linear Regression Strategy
6. Next Steps
7. Conclusion

Abstract

Our model is based of a paper titled “Cryptocurrency Alpha Models via Intraday Technical Trading” by two Stanford professors. Many times in the crypto market when individuals want to exit their position they exchange for another crypto position rather than using a off ramp to fiat currency. Often time during times of great market volatility people exit into stable coins making the price sometimes exceed $1. In this presentation we will present our results of a pairs trading strategy as well a lagged linear regression trading strategy.

Correlation of Assets Demonstration (ETH-USD scaled up)

Strategy

1. Watch for deviations in the price ratio from the rolling average ratio
2. If the ratio deviates from the rolling average by ± z**σ**, open a position
    a. We swept the value of z over a large range to determine the most profitable benchmark
    b. **σ** is the standard deviation in the difference between the instantaneous price ratio and rolling average ratio
3. Close the position upon the return of the price ratio deviation to the specified window ± 0.75**σ**
4. Repeat over specified time interval
5. Evaluate alpha

Strategy

When the Z-score	...we buy	...and we sell
> threshold	‘Ratio’ units of asset 2	1 unit of asset 1
< -threshold	1 unit of asset 1	‘Ratio’ units of asset 2

‘Ratio’ = (price of asset 1) / (price of asset 2)

Checking Correlation of Assets

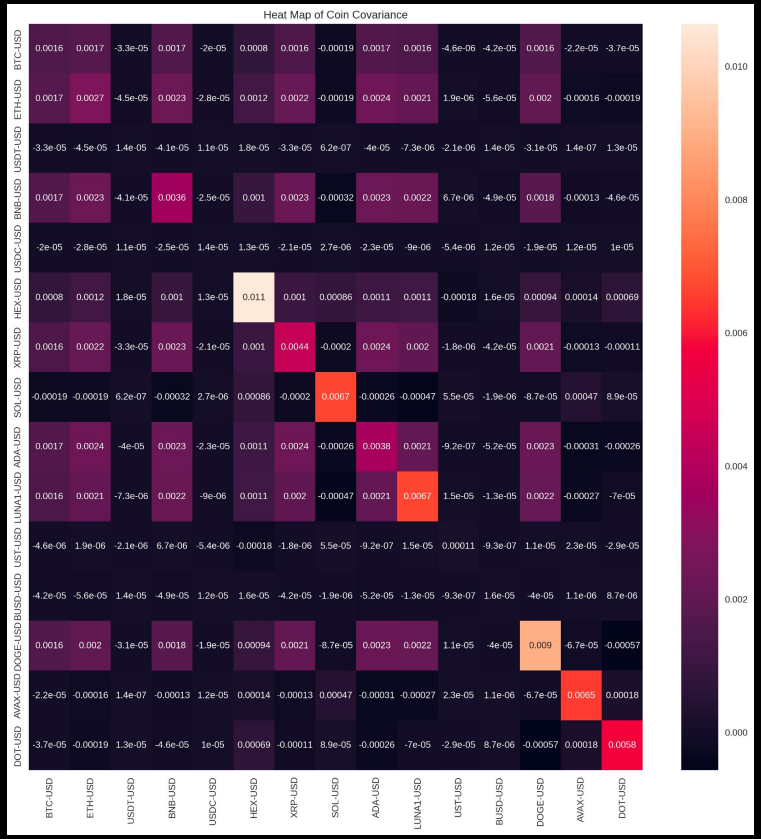
```
def correlated_coins(value):
  COINS = ['BTC-USD', 'ETH-USD','USDT-USD', 'BNB-USD','USDC-USD', 'HEX-USD', 'XRP-USD','SOL-USD','ADA-
USD','LUNA1-USD','UST-USD', 'BUSD-USD','DOGE-USD','AVAX-USD','DOT-USD' ]
  a = COINS
  b = COINS
  c = []
  d=[]
  for i in range(len(a)):
    for j in range(len(b)):
      d.append(a[i])
      d.append(b[j])
      cov = covariance_No_map(d, dt.date(year = 2020, month = 1, day = 1), dt.date.today())
      correlations = correlation_coeffs(cov)
      d=[]
      if (correlations[0][1]>value and correlations[0][1]<1):
          c.append((a[i],b[j]))
          d=[]
  print(c)
correlated_coins(0.75)
```

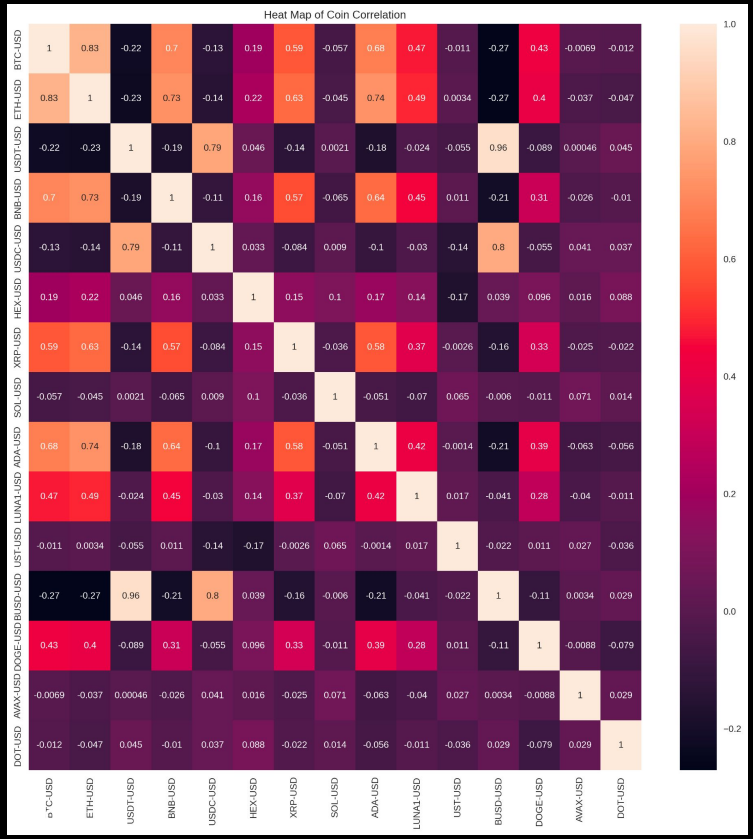
```
[('BTC-USD', 'ETH-USD'),
('ETH-USD', 'BTC-USD'),
('USDT-USD', 'USDC-USD'),
('USDT-USD', 'BUSD-USD'),
('USDC-USD', 'USDT-USD'),
('USDC-USD', 'BUSD-USD'),
('BUSD-USD', 'USDT-USD'),
('BUSD-USD', 'USDC-USD']
```

A list of currency pairs that have correlation values above 0.75

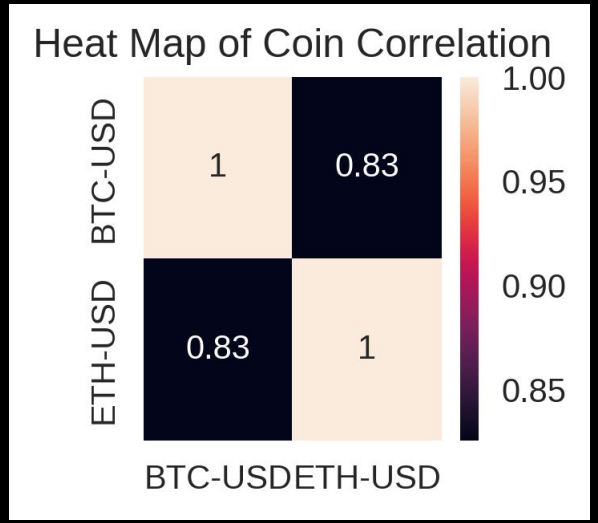
For a given input of cryptocurrencies, checks which two currencies have correlations over a certain threshold

Checking Correlation of Assets

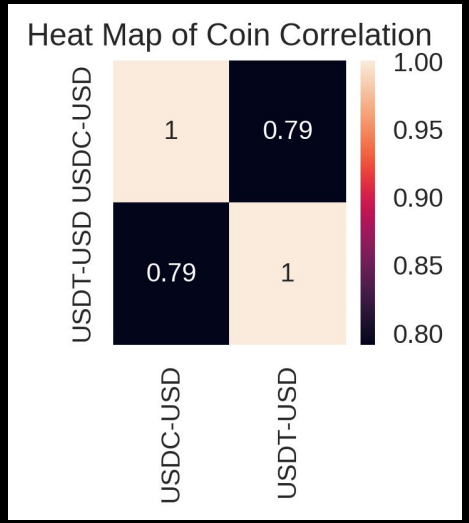




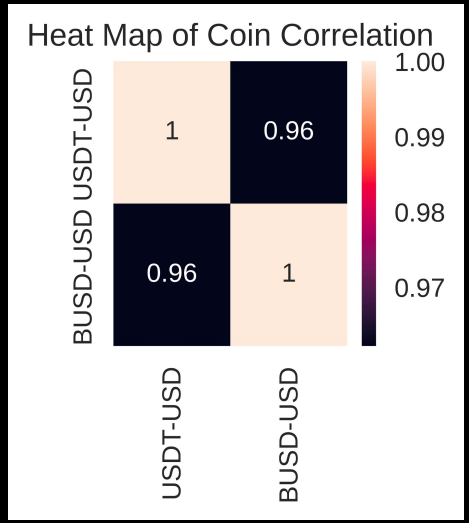
Checking Correlation of Assets



BTC-USD and ETH-USD correlation



USDC-USD and USDT-USD correlation



USDT-USD and BUSD-USD correlation

Output

- - - zscore > threshold - - -

:: INDEX = 1222 :: ZSCORE = 1.108

- - - opening position - - -

Buy 1.3588025129140167 ETH-USD

Sell 0.1 BTC-USD

current pnl: 139.46107600844198

- - - zscore re-enters [-0.75, 0.75] - - -

:: INDEX = 1224 :: ZSCORE = 0.535

- - - closing position - - -

Sell 1.3588025129140167 ETH-USD

Buy 0.1 of BTC-USD

current pnl: 169.48715501737024

Output

----------Final Output----------

token1: BTC-USD

token2: ETH-USD

strategy unit: 0.1

p/l: 153.68393967764723

start: -52261.16357421875

end: 39492.35498046875

Back Test

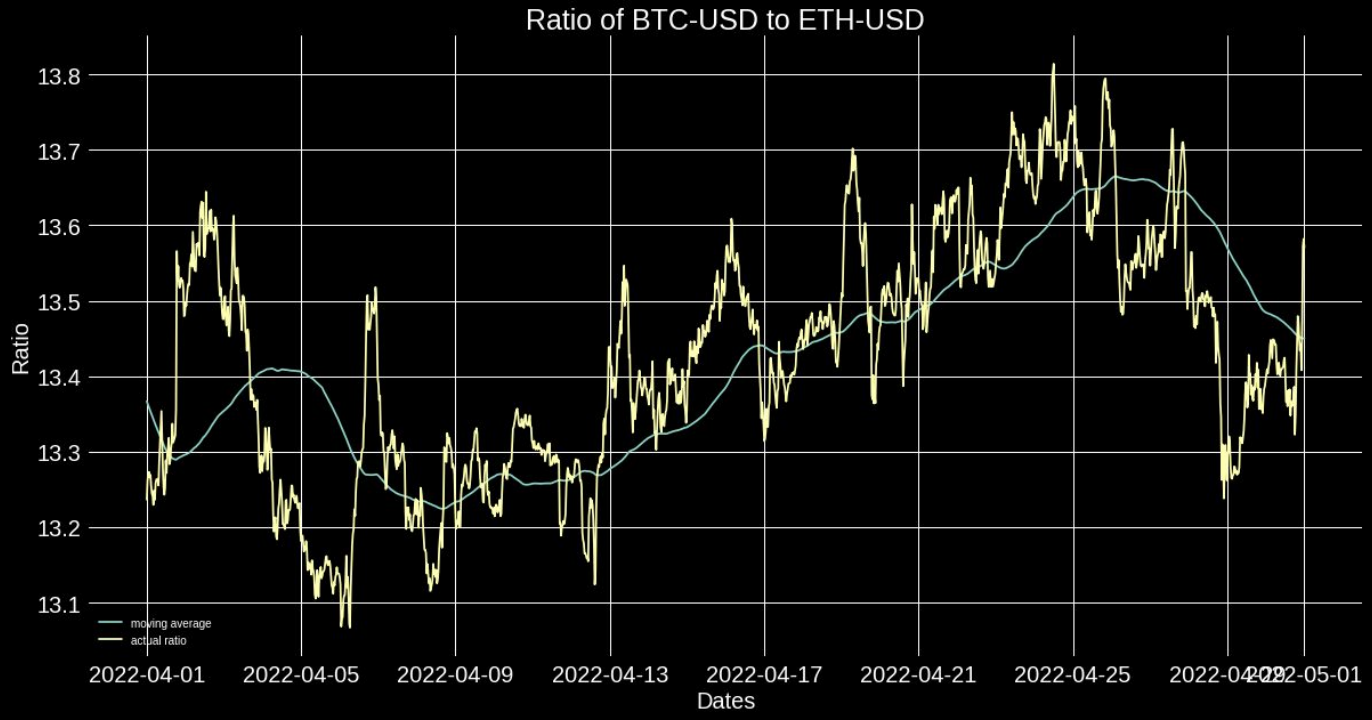
1. We tested our strategy using yfinance data spaced at 30m intervals.
2. The buy_sell( ) function contains a ‘for loop’ that iterates over our specified time period, making decisions as the strategy specifies.
3. At the end of its operation, it produces the profit / loss that would have been incurred had we employed this strategy over the interval.
4. This form of back testing only sees a single price value for each 30m interval.

Implementation

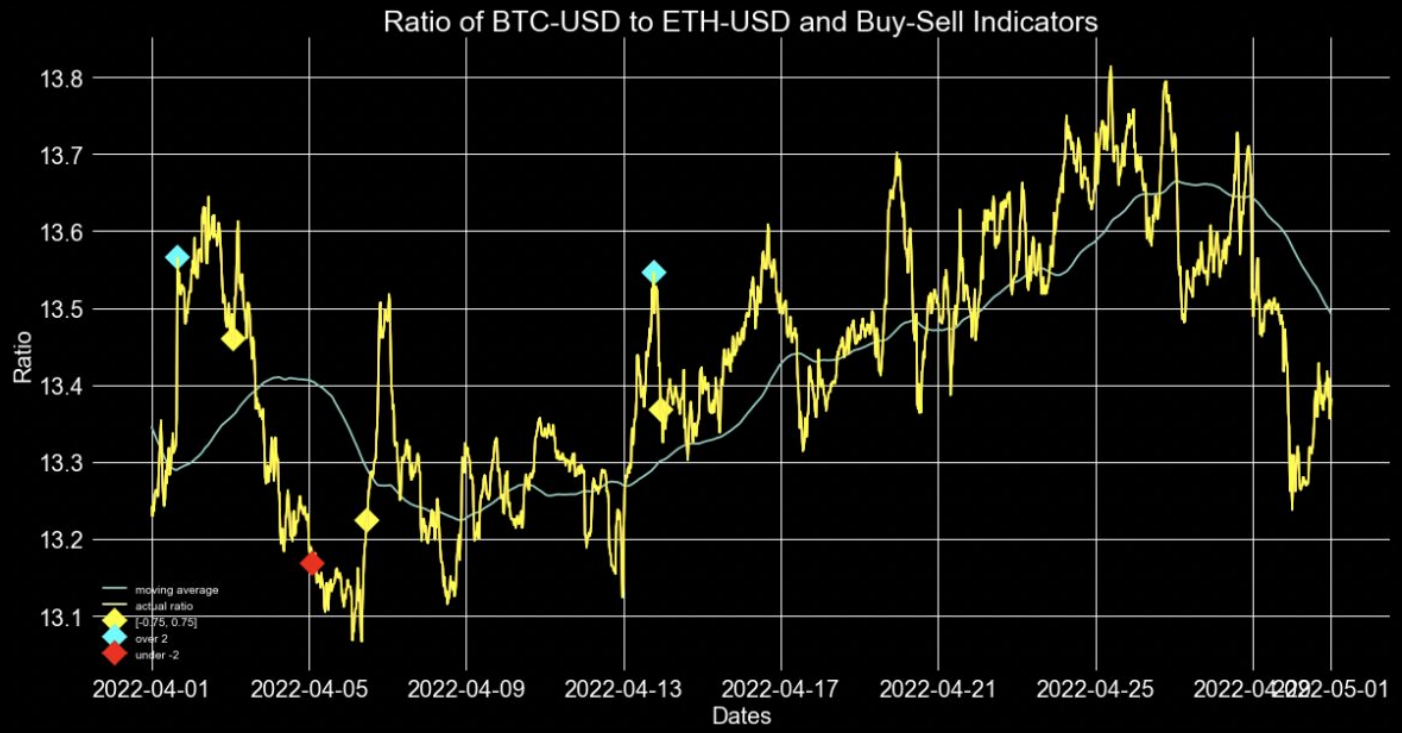
```
def buy_sell(unit, zscore, portfolio, threshold):
    pnl = 0
    open_pos = False
    pos = (0,0)
    for i in range(len(zscore)-(M-1)):
        Z = zscore[i]
        if (Z <= 0.75 and Z >= -0.75 and open_pos == True):
          print("- - - zscore re-enters [-0.75, 0.75] - - - \n :: INDEX = " + str(i) +
                " :: ZSCORE = " + str(Z) + "\n - - - closing position - - -")
          print("Sell " + str(abs(pos[1]))+ " "+COINS[1])
          print("Buy " + str(unit) + " of "+COINS[0])
          pnl = pnl + prices[i, 1] * pos[1] + prices[i, 0] * pos[0]
          open_pos = False
          print("current pnl: " + str(pnl) + "\n")

        elif (Z > threshold and open_pos == False):
          print("- - - zscore > 2 - - - \n :: INDEX = " + str(i) + " :: ZSCORE = " + str(Z) + "\n - - - opening position - - - ")
          print("Buy " +str(avg_price_ratio[i]*unit)+" "+COINS[1])
          print("Sell " + str(unit) + " "+COINS[0])
          pnl = pnl - prices[i, 1] * avg_price_ratio[i] * unit + prices[i, 0] * unit
          open_pos = True
          pos = (-unit, avg_price_ratio[i]*unit)
          print("current pnl: " + str(pnl) + "\n")

        elif (Z < -threshold and open_pos == False):
          print("- - - zscore < -2 - - - \n :: INDEX = " + str(i) + " :: ZSCORE = " + str(Z) + "\n - - - opening position- - -")
          print("Buy " + str(unit) + " "+COINS[0])
          print("Sell "+str(avg_price_ratio[i]*unit)+" "+COINS[1])
          pnl = pnl - prices[i, 0] * unit + prices[i, 1] * avg_price_ratio[i] * unit
          open_pos = True
          pos = (unit, -avg_price_ratio[i]*unit)
          print("current pnl: " + str(pnl) + "\n")

    if open_pos:
        print("Time is up, close remaining open position. \n pnl : " + str(pnl) + "\n")
        pnl = pnl + prices[i, 1] * pos[1] + prices[i, 0] * pos[0]

    end_portfolio = (prices[len(prices)-1, 0]*portfolio[0] + prices[len(prices)-1, 1]*portfolio[1])

    return (pnl, start_portfolio, end_portfolio)
```

```
t# LOAD DATA
start = dt.datetime(2022,4,1)
end = dt.datetime(2022,5,1)
# COIN 0 IS TRADED AT UNIT PER POSITION
# COIN 1 IS TRADED AT RATIO UNITS PER POSITION
COINS = ['BTC-USD', 'ETH-USD']
# this horrible function (below) sorts the coins into
alphabetical order and
# idk how to prevent this, coin 0 must always be alphabetically
before coin 1
prices = yf.download(COINS, start, interval = "30m")
returns = np.log(1+ prices['Adj Close'].pct_change()).dropna()
prices, returns = prices['Adj Close'].to_numpy(),
returns.to_numpy()
print(prices.shape)
print(prices)

price_ratio = np.divide(prices[:,0],prices[:,1])
print(price_ratio.shape)
M = 200 # Discrete time M-fold averager:
avg_price_ratio = np.convolve(price_ratio, np.ones(M), 'valid')
/ M
#print(avg_price_ratio)
```

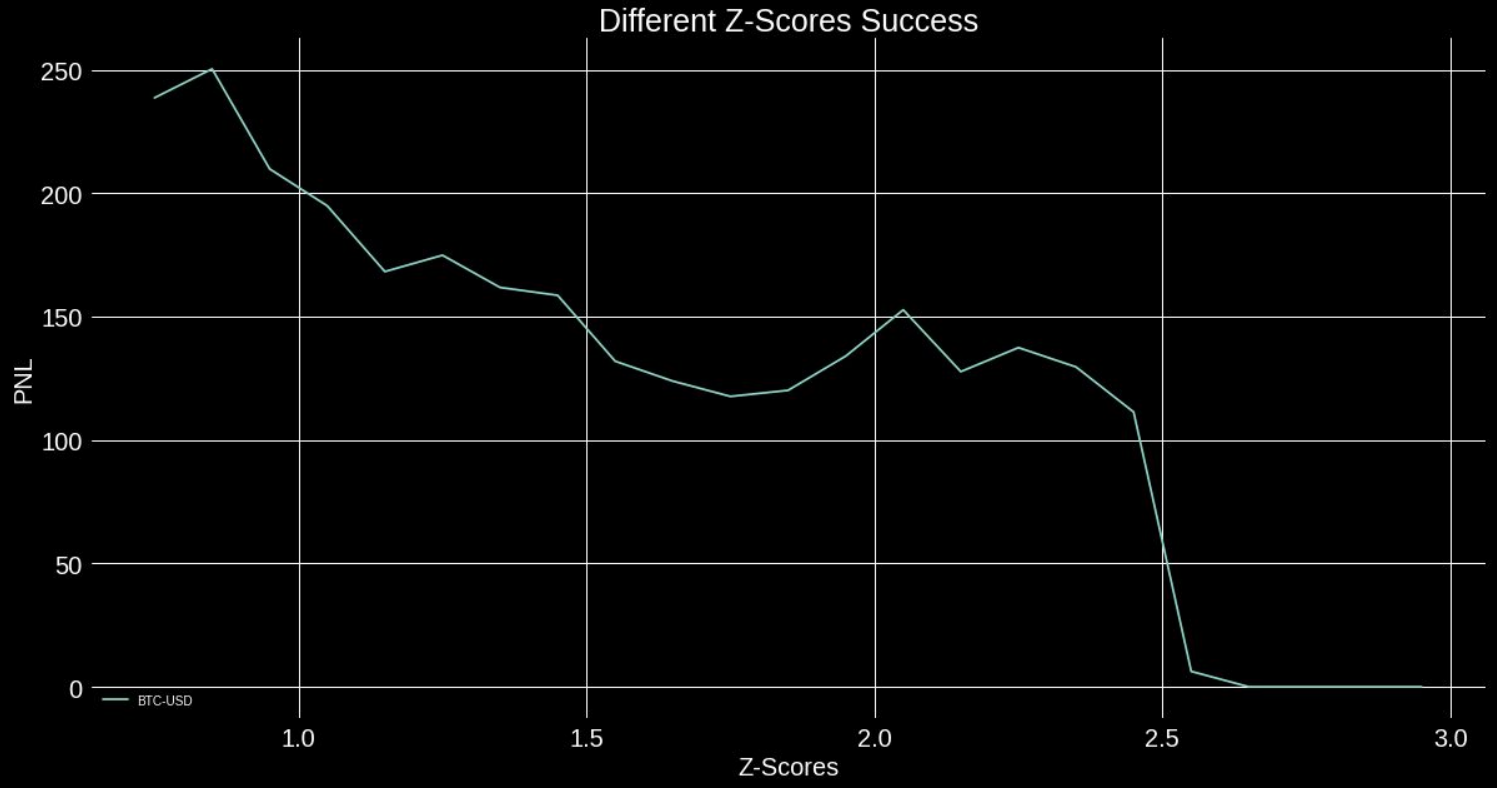
Results : Price Ratios



Results : Acting Upon Changes in Price Ratio

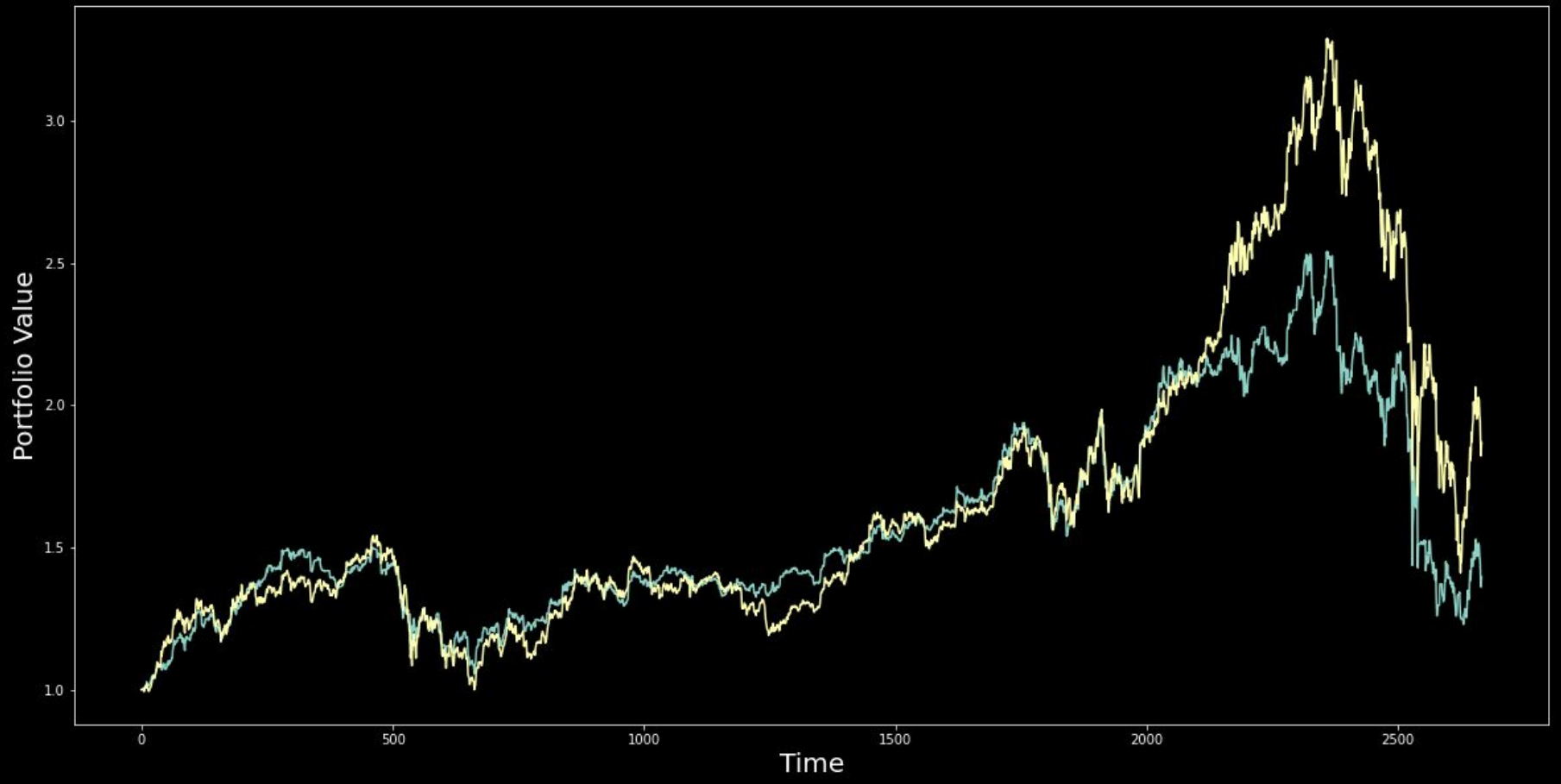


Results: PnL for trading (Apr 1 – May 1 2022)



Lagged Linear Regression Strategy

1. Find coin that is autocorrelated
2. Calculate log returns
3. Construct a linear regression model using lagged log returns as features
4. Create long/short positions based on the model's prediction of future log returns


Portfolio Value
3.0
2.5
2.0
1.5
1.0
0
500
1000
1500
2000
2500
Time

Conclusion

1. Lagged linear regression had been known to have higher returns.
2. We did not account for trading fees.
3. It seems that the Stanford paper was oddly unique with their returns.
4. Pairs trading may be over-utilized by the crypto trading community.
5. Next semester: look into other DeFi project specifically decentralized options market and find alpha there.

Resources

1. http://stanford.edu/class/msande448/2021/Final_reports/gr3.pdf