EF4323 Trading Room Workshop

By YSL-B Capital

Group 7

YIN
LIU
SHEN
BIA

Table of Contents

Overview

The market consists of three securities, stock Ultimate Bedrock (UB), Gems Exploration (GEM), and the ETF that that is made up of half UB and half GEM (the value of the ETF is the sum of UB and GEM). The starting price of the securities are $50.00, $25.00, and $75.00, respectively. The final values will be randomly drawn from the interval: UB{$40, $60}, GEM{$20, $30}, and ETF{$60, $90}, and they are not revealed until the end of the case.

Three general strategies could be used to generate profit, price discovery, arbitrage, and market-making. Meanwhile, we also think beyond the basic ones and add some special functions. The details are provided in the following report. In the end, we will recommend a comprehensive trading strategy for our clients by taking into consideration the return and risk.

Strategies

Price Discovery

Thoughout the case, ten news with private information that estimates the final values of each security will be released randomly. The main idea of the price discovery is to get the first and most straightforward price estimation by recalculating the range for each stock every time a news item is released using the provided formula, $Ets = Ps + X(300-t)/50$. Subsequently, the upper and lower bound are $Ets + (300- t)/ 50$ and $Ets - (300-t)/ 50$. If we will long or short the securities at a price outside their estimated ranges, and this would give an almost surely risk-free profit.

The algorithm would firstly grab the news when it comes out. Next, it deduces the interval based on the above formula then update the upper and lower bound.

upper = min (news-based upper bound, prior upper bound)

lower= max (news-based lower bound, prior lower bound)

Even though the new range is unlikely to go beyond the precious boundary, we still set the max and min to prevent any ‘improbability.

Illustration 1

The market price (bid or ask) may fluctuate outside the range randomly throughout the case, showing in the red circle of Illustration1, and this is our profit opportunity. Whenever that happens, we submit a 5000-limit order, either selling at bid-RISK1 or buying at ask+RISK1. We set the RISK1 to be 0.01. It is used to prioritize our request to make it rank at the top of the bid or the ask column. Hence, **our order can be executed before the existing orders and increase the probability of getting the shares we want**.

Furthermore, we choose the **limit order** rather than the market order. The limit order will be executed only at or better than the specific price level, that is, the price outside the range. In contrast, the market order that trades at the best available price (buy at the asking price or sell at the bid price) in the current market can sometimes lead to unintended, and in some cases, significant costs due to the price fluctuation. Specifically, the price may change dramatically when there are suddenly large amounts of orders on either side. Then, using the market orders might get the shares at a cost inside the estimated range. Therefore, the limit order would be better since it guarantees the profit by eliminating the transactional risk.

What's more, to optimize the profit, in case there are more than one positive profit exists, we compute the payoff

Long = max (0, round(LOWER - M_SPREAD-ask,2))

Short = max (0, round(bid-(UPPER + M_SPREAD),2))

The M_SPREAD is 0.05, including the 2* transaction cost since we need to both buy and sell to generate profit and the RISK1 when we set the order price, of each stock and submit the order on the maximum spread. Illustration 2 shows that both long GEM and short UB have positive profit; however, UB's payoff of current bid and upper bound is 2.7 which is higher than GEM's spread. Hence, **our code will submit the order on the highest one**, shorting UB.

Long Opportunity		Short Opportunity	
UB	-6.28	UB	2.7
GEM	2.23	GEM	-3.78
ETF	-4.22	ETF	-0.53

Illustration 2

```
'''store the payoff'''
tempList=[]
ubShortD=max(0, round(UB_bid-(UB_UPPER + M_SPREAD),2))
tempList.append(ubShortD)
    ''' omit the rest'''
tempList.sort()
 '''sumbit the order on largest profit'''
 if tempList[5]==ubShortD and ubShortD>0:
     session.post('http://localhost:9999/v1/orders', params =
         {'ticker':'UB','type':'LIMIT','quantity':5000,'price':(UB_bid-
         RISK1),'action':'SELL'})
''' omit the rest'''
```

Code 1: compare payoff and submit order

Arbitrage between Stocks and ETF

Because there's no restriction on short-selling and real ETF price should be the sum of the two stocks' prices, an arbitrage opportunity exists when the price at which the ETF is traded is different from the value calculated. Specifically, the arbitrage between stocks and ETF occurs when ETF_bid > UB_ask + GEM_ask or when UB_bid + GEM_bid > ETF_ask. After taking the transaction cost into consideration, the arbitrage is profitable when ETF_bid > UB_ask + GEM_ask + 3 * Transaction Cost or when UB_bid + GEM_bid > ETF_ask + 3 * Transaction

Cost. Ideally, traders should short ETF while long UB and GEM simultaneously in the first scenario and inverse in the second case to gain the profit.

We decide to execute the arbitrage strategy for a limit amount of time. The arbitrage takes up the positions, and we have to hold it for a while until we find the chance to close it. If we combine both strategies throughout the case, we may need to sacrifice some space for the price discovery strategy. Since the gain of the price discovery strategy is guaranteed, we want to leave the position for it as much as possible. Hence, we will only execute the arbitrage strategy before the first news releases.

Furthermore, the algorithm is divided into two parts, arbitrage1 and arbitrage 2. The first one runs for the first 40 tick to capture the arbitrage opportunity while the second one runs after the 40 tick to close the previous arbitrage positions so that the arbitrage will not impact the price discovery. The number 40 is chosen because we have significant test results indicating that the first news is released after 50 ticks most of the time, and we want to leave some time to close the positions. The test result is displayed below.

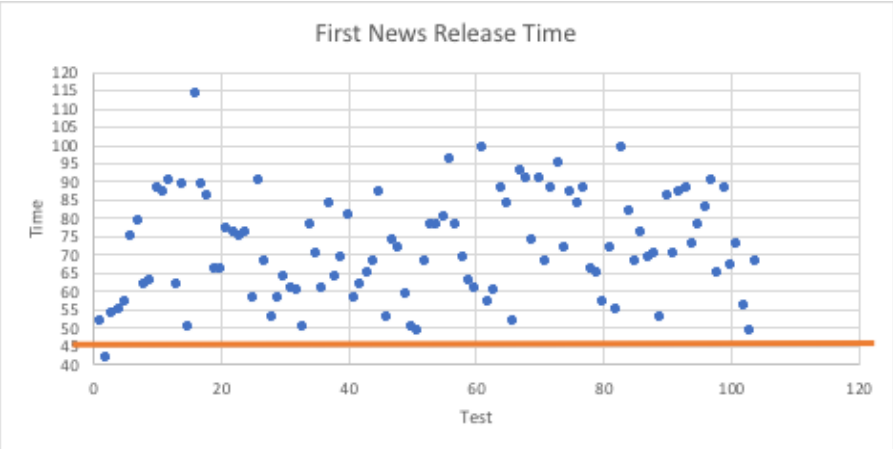

Figure 1: First News Release Time

The code for arbitrage1 and arbitrage2 is similar but slightly different. They both aim to capture the arbitrage opportunity and use the **market order** since we want the same position for all three stocks immediately. If we use the limit order, orders may be fulfilled different, thus loses the advantage of arbitrage. Moreover, as we mentioned before, the market order is costly, especially when the market is quite dynamic, and all the investors would implement the arbitrage whenever the chance exists. Therefore, a spread is needed to reduce the transactional risk. The bigger the spread, the safer on the arbitrage strategy; however, if the spread is too big, we may not have the profit opportunity. Thus, after numerous testing, we use **0.2** as the spread.

The difference for the two algorithms is size and condition. For arbitrage1, it will short ETF and long UB & GEM when *ETF_bid > (UB_ask + GEM_ask + SPREAD)* or long ETF and short UB & GEM when *(UB_bid + GEM_bid) > (ETF_ask + SPREAD).* The sizes are for the two scenarios are:

short_ETF_size = min(ETF_bid_size[0]/5, UB_ask_size[0]/5, GEM_ask_size[0]/5, 1000))

long_ETF_size=(ETF_ask_size[0]/5, UB_bid_size[0]/5, GEM_bid_size[0]/5, 1000)

We choose the size to be minimum of the top of each column and 1000 because we want to get the position at the best price as much as possible to reduce the risk. Moreover, our simulation shows that with large size such as 5000, the profit is small, around 10K to 50K, but the loss can be significant, more than 150K. Therefore, we further divide the bid/ask [0] by 5.

For arbitrage2, since its goal is to close the arbitrage 1's position to leave position for price discovery, we set it to be effected when *ETF_bid > (UB_ask + GEM_ask+ SPREAD) and ETF_pos>0 or (UB_bid + GEM_bid) > (ETF_ask+ SPREAD) and ETF_pos<0.* The buy and sell for each case are the same but the sizes are different, *min(short_ETF_size,abs(ETF_pos)) and min(long_ETF_size,abs(ETF_pos)),* respectively.

```
"""start arb"""
spread= 0.2
def Arbitrage1(session, UB_bid, UB_ask, GEM_bid, GEM_ask, ETF_bid, ETF_ask, short_ETF_size, long_ETF_size):
    """short etf and long two stocks"""
    if ETF_bid > UB_ask + GEM_ask + SPREAD:
       session.post('http://localhost:9999/v1/orders', params = {'ticker':'ETF','type':'MARKET','quantity':short_ETF_size,'action':'SELL'})
       session.post('http://localhost:9999/v1/orders', params = {'ticker':'UB','type':'MARKET','quantity':short_ETF_size,'action':'BUY'})
       session.post('http://localhost:9999/v1/orders', params = {'ticker':'GEM','type':'MARKET','quantity':short_ETF_size,'action':'BUY'})
    """LONG ETF AND SHORT TWO STOCKS"""
    if UB_bid + GEM_bid > ETF_ask + SPREAD:
        session.post('http://localhost:9999/v1/orders', params = {'ticker':'ETF','type':'MARKET','quantity':long_ETF_size,'action':'BUY'})
        session.post('http://localhost:9999/v1/orders', params = {'ticker':'UB','type':'MARKET','quantity':long_ETF_size,'action':'SELL'})
        session.post('http://localhost:9999/v1/orders', params = {'ticker':'GEM','type':'MARKET','quantity':long_ETF_size,'action':'SELL'})
    sleep(SPEEDBUMP)

"""end the arb and close out position"""
def Arbitrage2(session, UB_bid, UB_ask, GEM_bid, GEM_ask, ETF_bid, ETF_ask, short_ETF_size, long_ETF_size, UB_pos, GEM_pos, ETF_pos):
        """close out position --> short etf"""
        if ETF_bid > (UB_ask + GEM_ask+ SPREAD) and ETF_pos>0:
           session.post('http://localhost:9999/v1/orders', params = {'ticker':'UB','type':'MARKET','quantity':min(short_ETF_size,abs(ETF_pos)),'action':'BUY'})
           session.post('http://localhost:9999/v1/orders', params = {'ticker':'GEM','type':'MARKET','quantity':min(short_ETF_size,abs(ETF_pos)),'action':'BUY'})
           session.post('http://localhost:9999/v1/orders', params = {'ticker':'ETF','type':'MARKET','quantity':min(short_ETF_size,abs(ETF_pos)),'action':'SELL'})
        """long etf"""
        if (UB_bid + GEM_bid) > (ETF_ask+ SPREAD) and ETF_pos<0:
           session.post('http://localhost:9999/v1/orders', params = {'ticker':'UB','type':'MARKET','quantity':min(long_ETF_size,abs(ETF_pos)),'action':'SELL'})
           session.post('http://localhost:9999/v1/orders', params = {'ticker':'GEM','type':'MARKET','quantity':min(long_ETF_size,abs(ETF_pos)),'action':'SELL'})
           session.post('http://localhost:9999/v1/orders', params = {'ticker':'ETF','type':'MARKET','quantity':min(long_ETF_size,abs(ETF_pos)),'action':'BUY'})
        sleep(SPEEDBUMP)
```

Code 2: Arbitrage 1 and Arbitrage 2

Market Making

In this section, we will introduce the third strategy: combination of market making and price discovery. For the market making algorithm, we slightly modified the strategy that was used in case ALGO2.

As explained in the previous arbitrage section, we have generated statistical test result showing that the first news is released after 50 ticks most of the time. Hence, in order to prevent price discovery strategy being affected by market making, we decide to only execute market making strategy in the first 50 ticks.

```
if tick < 40 :
    #market making
    mktMaking(s, 'UB', UB_bid, UB_ask, UB_bid_size, UB_ask_size, UB_pos)
    mktMaking(s, 'GEM', GEM_bid, GEM_ask, GEM_bid_size, GEM_ask_size, GEM_pos)
    mktMaking(s, 'ETF', ETF_bid, ETF_ask, ETF_bid_size, ETF_ask_size, ETF_pos)

elif tick < 50:
    #cancel all of the current limit orders
    cancel_params = {'all': 1}
    s.post('http://localhost:9999/v1/commands/cancel', params = cancel_params)
    #close position
    closeP(s,'UB',UB_pos,UB_bid[0], UB_ask[0])
    closeP(s,'GEM',GEM_pos,GEM_bid[0], GEM_ask[0])
    closeP(s,'ETF',ETF_pos,ETF_bid[0], ETF_ask[0])
```

Code 3: Market Making

The whole market making strategy consists of three parts: market making, cancellation of all current limit orders and closing position.

For market making, we use OBS (Order Book Signals) as our main signal generator. OBS is built on order book information. For each tick, OBS is updated and calculated as follows:

```
for i in range(1,5):
    obsBid.append(sum(bid_size[0:i])/ sum(ask_size[0:i]))
    obsAsk.append(1/obsBid[i-1])
```

Code 4: OBS Formula

We calculate obsBid up to level 4 and obsAsk is just the reciprocal of the obsBid. OBS gives us a signal whenever OBS exceeds the predetermined threshold K. For example, if obsBid > K, then (bid + ask)/2 is expected to increase; if obsAsk > K, then (bid + ask)/2 is expected to decrease. If the middle price is expected to increase in a short time, we will place a more aggressive bid limit order and vice versa.

```
elif obsBid[0] > K:
    resp = s.post('http://localhost:9999/v1/orders', params = {'ticker':ticker,
'type':'LIMIT', 'quantity':defaultSize, 'price':bid[0]+0.01, 'action':'BUY'})


elif obsAsk[0] > K:
    resp = s.post('http://localhost:9999/v1/orders', params = {'ticker':ticker,
'type':'LIMIT', 'quantity':defaultSize, 'price':ask[0]-0.01, 'action':'SELL'})
```

Code 5: Place Aggressive Bid/Ask Order

Additionally, we have some predefined inventory thresholds to prevent holding too much position.

After 40 ticks, we stop the market making and cancel all of the limit orders and start to close current positions.

```
def closeP(s, ticker, pos, ask, bid):
    a = abs(pos) / 5000
    b = abs(pos) % 5000
    while a/2 > 0 :
        if pos > 0:
            s.post('http://localhost:9999/v1/orders', params = {'ticker':ticker, 'type':'LIMIT',
                'quantity':5000, 'price':ask,  'action':'SELL'})
        elif pos < 0:
            s.post('http://localhost:9999/v1/orders', params = {'ticker':ticker, 'type':'LIMIT',
                'quantity':5000, 'price':bid, 'action':'BUY'})
        sleep(SPEEDBUMP)
        a -= 1
    if pos > 0:
        s.post('http://localhost:9999/v1/orders', params = {'ticker':ticker, 'type':'LIMIT',
            'quantity':b, 'price':ask, 'action':'SELL'})
    elif pos < 0:
        s.post('http://localhost:9999/v1/orders', params = {'ticker':ticker, 'type':'LIMIT',
            'quantity':b, 'price':bid, 'action':'BUY'})
    sleep(SPEEDBUMP)
```

Code 6: Market Making Close Position

Special Features

Trading is a game where all participants should compete with each other and can be considered as a zero-sum game to some extent. The nature of the game decided that focusing only on the model itself is not enough, especially when all participants choose to use similar strategy. The only method to stand out of all others is to differentiate the strategy. Based on this, our trading algorithm has several special features that improve the execution speed as well as take other traders' trading behavior into account.

High Frequency Trading

The high-frequency algorithmic trading has recently come to dominate some financial markets as it results in more efficient financial markets. The faster the speed, the higher profit-earning opportunities, particularly on the arbitrage and market making. Thus, we use python to write out code. Comparing with VBA, python generally provide us a faster implementation of the strategies.

However, the speed cannot be too fast because it will cause order conflicts. Therefore, we add *sleep (SPEEDBUMP*) that would delay the execution for a little while in between the codes.

Free Lunch

When we conduct the evaluation about the market, we noticed that sometimes the price of the stock would increase dramatically within a very short period, indicating some market participants may receive the news about stock mispricing and then use market order to buy shares. As shown the Illustration 3 below, we believe the sudden drop of the price, circled in blue, comes from the market order submitted by other market participants.

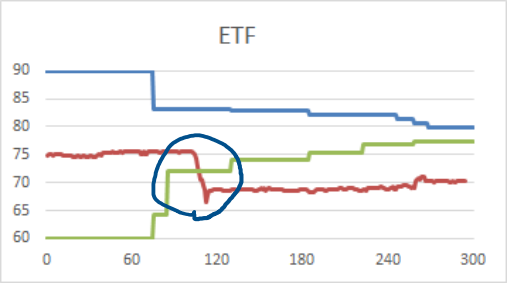

Illustration 3

This kind of irrational behavior would normally create a strong market impact, leading price to a new height that beyond our accurate estimation. **We would obtain a so called 'Free Lunch' if we set a limit sell order at a high price which got filled by market orders submitted by others.** In an extreme case, if the number of irrational traders is large enough, there is high possibility that one side of trader book would be cleared, which means we would be able to set the ask at whatever price we want and get enormous profit from other's mistakes. Even if market order does not reach the level of clearing trader book or pushing price over accurate estimation, once our limit order got filled, we would still profit from buying it back after the market price goes down, which normally is the case.

'Insider' Information

The other special feature of our strategy is designed to deal with information asymmetric. For most cases, our algorithm would profit mainly from mispricing of three different stocks. However, there is little chance that we are not able to receive any news showing the market is

over or undervalued, which would lead to low profit. For example, the Illustration 4, where the orange and blue straight lines represent our final price estimation and the rest one indicates the market price, show that for first 180 seconds, it would be difficult to get profit from price discovery from the news.

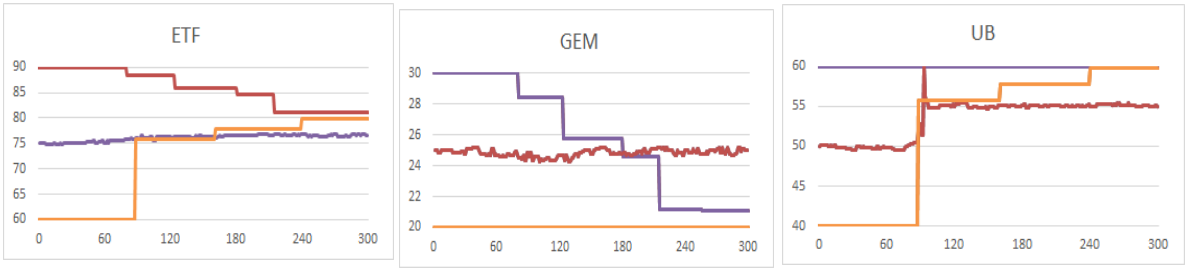

Illustration 4

Therefore, getting more information from others would be of benefits in such situation. Though we cannot get access to news from others directly, we are still able to guess the valuable news via the limit orders set by others. Considering market making is another possible strategy for the price discovery, which requires limit order as well, the difficulty of this special strategy would lie in identify whether other trader received news and started to buy huge amount or they just set some limit orders on both sides of the trader book. We decide to make some adjustments depending on the market situation. If we have enough fund to operate several accounts to participate in market together, **one of our account would read trader ID in the book trader and once our own account's trader ID appears in the list, it would automatically follow the bid or ask and submit the same limit buy or sell order**. Such order following would allow one of our account to get price information from other accounts whose strategy is equally rational and thus leading to higher return.

	Combined news	Single News
Test 1	694,770	627,000
Test 2	752.328	307,000
Test 3	466,536	347,000
Test 4	583,497	516,914
Test 5	377,872	360,000

Figure 2: Test result

We tested this several times with single news labeled as normal trader and combined news as trader with his own news as well as following the other one**. We can see from above results that with more information and following orders from others would create us more profit**. The liquidity of the market is almost the same while if we have one trader following the other, the limit orders to be filled are expected to be smaller, leading to less profit. This is one of the weaknesses of this strategy. However, this weakness can also be utilized to attack other traders from other companies. If we find someone from other companies apply the same rational strategy, price discovery, and submit limit orders only when the market is mispriced, we would try to locate his limit order and do the same, robbing his profit.

Performance Result

After designing the above algorithm trading, we then run numerous backtest on different combinations to achieve an optimal one. The results are detailed below.

Price Discovery

We have run the PD algorithm for fifty times and then compute the descriptive statistical summary. It has a sample mean of 514,275.25 and a standard deviation of 245,084.5453. Meanwhile, the 1st quantile and 3rd quantile are 324,403.25 and 750, 188.00, respectively. The

distribution is positively skewed, indicating a surprise on the right side (a more significant profit).

Price Discovery	
Mean	514,275.25
Standard Error	33,156.25
Median	441,642.50
1st quantile	324,403.25
3rd quantile	750,188.00
Standard Deviation	234,450.10
Skewness	0.45
Range	922,592.00
Minimum	135,253.00
Maximum	1,057,845.00
Count	50.00

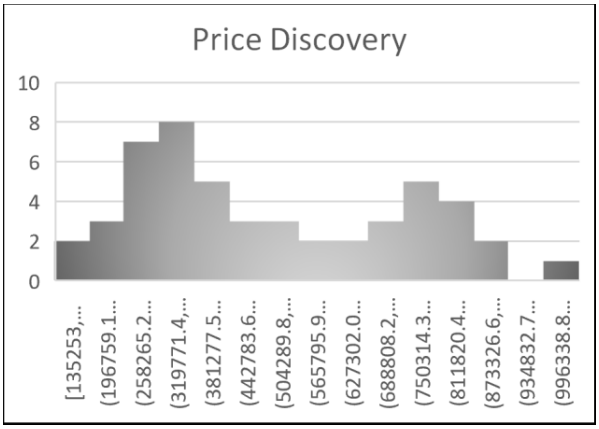

Figure 3: PD Performance

Price Discovery & Arbitrage between Stocks and ETF

Fifty results are collected. The sample mean for the price Discovery & Arbitrage is 468,623.584, with the standard deviation of 245,084.5453. Most of the profits concentrate on 269,541.50 and 650,365.00 (IQR). The distribution has a skewness of 0.72, indicating a slightly right skewness that is more preferred by the investors.

Price Discovery & Arbitrage	
Mean	468,623.58
Standard Error	34,660.19
Median	409,060.40
1st quantile	269,541.50
3rd quantile	650,365.00
Standard Deviation	245,084.55
Skewness	0.72
Range	906,641.40
Minimum	120,691.00
Maximum	1,027,332.40
Count	50.00

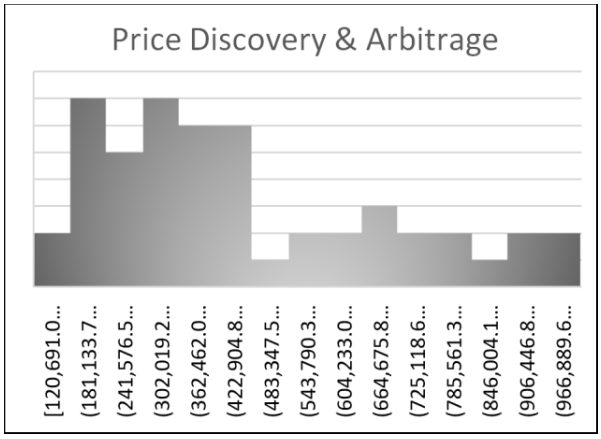

Figure 4: PD & Arb Performance

Price Discovery & Market Making

We ran total thirty-five simulations for price discovery and market making strategy. It is observed that the average profit/loss is $470,730.64 and the standard deviation is 248,063.05. Moreover, it has negative skewness, which is disliked by general investors.

Market Making & Price Discovery	
mean	470,730.64
standard error	41,930.31
median	491,158.43
1st quantile	262,969.50
3rd quantile	684,989.50
standard deviation	248,063.05
Skewness	(0.04)
range	985,496.00
minimum	16,502.00
maximum	1,001,998.00
count	35.00

Figure 5: PD & MM Performance

Since the Arbitrage and Market Making with Price Discovery separately does not generate a more profitable results, we believe that the combination of them could not have better yield thus forgo the algorithm.

Analysis

The spread of the strategy is quite large, ranging from 10K+ to 1 million. So does the standard deviation. Such volatile are mainly **due to the dependency on news quality and market dynamics**. If the news is quality, the estimation range shrink speedily thus making the algorithm more profitable. However, if the news has little or no impact on the boundary and the market price is always within or changed marinally around the interval (Illustration 5), our profit is mininmal as the profit opportunity disappears quickly. What's more, the number of traders also

influences performance since everyone is applying the same strategy at the same time, which will reduce our earning in some degree.

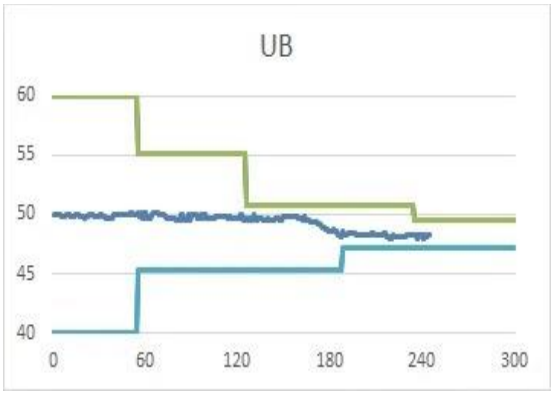

Illustration 5

We then compare the results of the three algoithms to make our deicison. Based on the Figure 6, we find that the **PD alone has the highest mean and is the least volatile** among the three strategies. Meanwhile, its minimum and maximum are higher than those of the others. The possible explanation to this result could be that arbitrage and market making could not ensure they will generate positive profits immediately in the first 50 ticks. For the arbitrage, its profits are based on the fact that when closing the postion, the final price of ETF is equal to the sum of the final price of UB and the final price of GEM. However, after 50 ticks, when running price discovery algorithm, these postions could be closed ahead of time and not making profits. For market making, it needs more time to generate considerable profits and it could merely generate positive profits in the first 50 ticks.

	PD	PD & Arb	PD & MM
Mean	514,275.25	468,623.58	470,730.64
STD	234,450.10	245,084.55	248,063.05
Skewness	0.45	0.72	-0.04
Min	135,253.00	120,691.00	16,502.00
Max	1,057,845.00	1,027,332.40	1,001,998.00

Figure 6

Conclusion

After carefully review the return and risk, we sincerely recommend the **Price Discovery** as the trading strategy. The performance results have demonstrated out outstanding quanlificaiton.

Just as our slogan states, ‘We Buld This Bank For You.’ We dedicate to guard your forturn and build your wealth.

Choose us, you will never regret.