**PCP改良策略**

1. 主要想法

同樣延續過去實習生執行的PCP套利設計，主要是

* 想承擔一點風險，增加PCP(Put-Call Parity無風險套利)的獲利空間
* 將PCP分成三隻腳 (**腳1:較貴的選擇權+腳2:期貨+腳3:較便宜的選擇權**)，如果買比原價位多100點的，最多損失100點，不過最好的情況是可多獲得100點
* 腳3設計在原本PCP履約價的 +-200區間內，與原本價格相比，買較便宜的，賣較貴的，如果在短時間內沒有找到符合的選擇權，就做原本的PCP套利

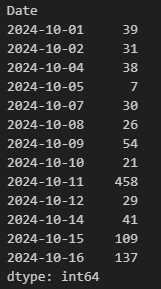
1. 策略架構
2. 成交的tick資料去mapping，組合成PCP型式，以及將市場上成交數據抓近來，設定條件如下：

* 先**觀察到期之月選合約**
* 以call當作基準，去mapping put跟future資料，如果在**call附近5分鐘內找不到put，則移除資料**
* 把call, put, future時間列出，同時切分成Date(年月日)、Time(時分秒)

df\_pcp：PCP合併資料 (1020筆資料、23個欄位)

df\_tick：市場選擇權成交數據 (10428660筆資料、15個欄位) – **已排除IV = 9999的數據**

觀察PCP操作頻率，10/11、10/15、10/16成交筆數較多



1. 觀察逐筆成交的選擇權費用以及套利機會如何，設計出適合的策略

觀察PCP公式

理論大：Buy Future + Short Call + Long put -> 直接用future買賣來判斷(B)

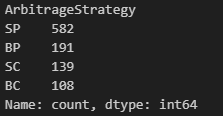
* 如果call > put，short call當腳1，long put當腳3 (BP)
* Call履約價 > Put履約價，類似多頭價差
* Call履約價 < Put履約價，類似空頭價差
* 如果call < put，short call當腳3，long put當腳1 (SC)
* Call履約價 > Put履約價，類似多頭價差 (空頭)
* Call履約價 < Put履約價，類似空頭價差

實際大：Sell Future + Long Call + Short put-> 直接用future買賣來判斷(S)

* 如果call > put，long call當腳1，short put當腳3 (SP)
* Call履約價 > Put履約價，類似空頭價差
* Call履約價 < Put履約價，類似多頭價差
* 如果call < put，long call當腳3，short put當腳1 (BC)
* Call履約價 > Put履約價，類似空頭價差
* Call履約價 < Put履約價，類似多頭價差

將腳3類別存到ArbitrageStrategy欄位中，產出 df\_origin\_pcp

計算BP, SC, SP, BC四種類別之數量，發現SP佔據大多數



1. 鎖住腳1及期貨，再根據其成交時間(TradeTime)，找出對應之可能數據(腳3)

操作想法：透過迴圈的方式去執行，設計雙層迴圈。

第一層-日期級別：遍歷每天的PCP策略資料，選出與該日對應的tick資料

第二層-PCP每筆策略級別：在每一天的篩選基礎上，依次處理當天每筆PCP策略資料

* 設定df\_tick資料範圍**介於PCP腳1、期貨組好後，往前扣30秒，開始觀察1.5分鐘內之數據** (PCP數據最小單位是分鐘，發現會出現四捨五入的現象，時間做更動才能也剛好抓到原本PCP的數據)
* 觀察其ArbitrageStrategy，如果是BC、SC，僅需抓取Call，反之如果是BP、SP，抓取Put
* 設定抓取之履約價不得超過**原strike\_price的 +-200區間內**
* 將每一筆資料可能之腳3存成result1，只剩1004筆資料

觀察履約價分布，以及每一筆PCP數據有多少可能之腳3

1. **篩選最划算之腳3**

操作想法：

第一層迴圈分是第幾筆PCP

第二層迴圈專注於該筆PCP，去抓取屬於其之FilteredTicks

* 針對每一筆PCP去跑回歸，去抓取各自的FilteredTicks
* 抓出其履約價、IV、成交價格、標的名稱、買賣權別、時間等去觀察
* 透過ArbitrageStrategy去分類Call, Put
* 按照履約價、成交時間做排序，且根據履約價分組，**只蒐集各個履約價在五分鐘內的第一筆資料**，這樣一組PCP策略最多可能也只會有5~6個選擇
* **挑選出IV最大的數據(暫定)**，將其數據導回至PCP數據中，取代原有的腳3數據
* 匯出成 df\_pcp\_copy
* 將call, put strike提出來變成欄位，重新計算費用(使用不同於原本的腳3會導致費用不同，如何估算)，匯出成df\_final\_pcp

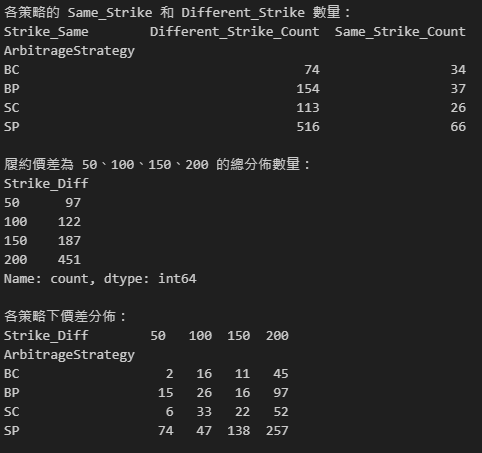
(目前因為篩選機制是以IV最大為考量，所以不小心推進到極端值abs(200)好像也是很正常的事情，不過可以觀察為甚麼有些不會推展到極端值(真正的outliner)，然後每個履約價第一筆數據會不會時間上有落差)

目前的問題是，各個履約價的第一筆可能仍有時間上的差異，不過因為是用成交價當資料，所以也沒辦法只能多考慮滑價成本

1. 假設原本PCP時間：004300

2. 價平 - 004237

3. 價平+200 - 004359 (中間可能還會有其他價平資料)

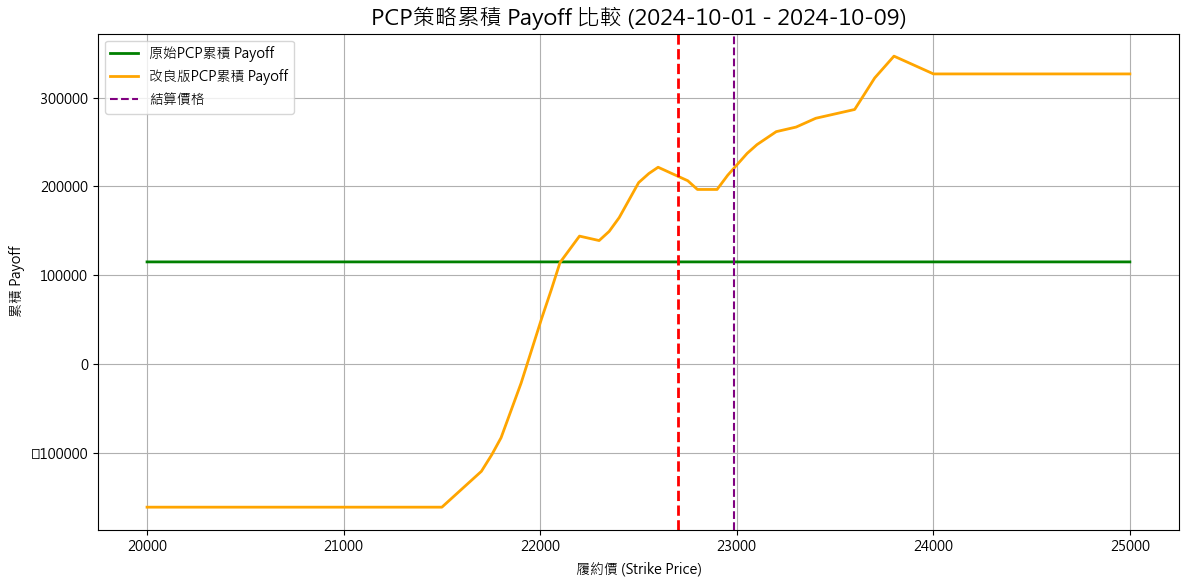


1. 比較PCP與修正後PCP策略之獲利差異

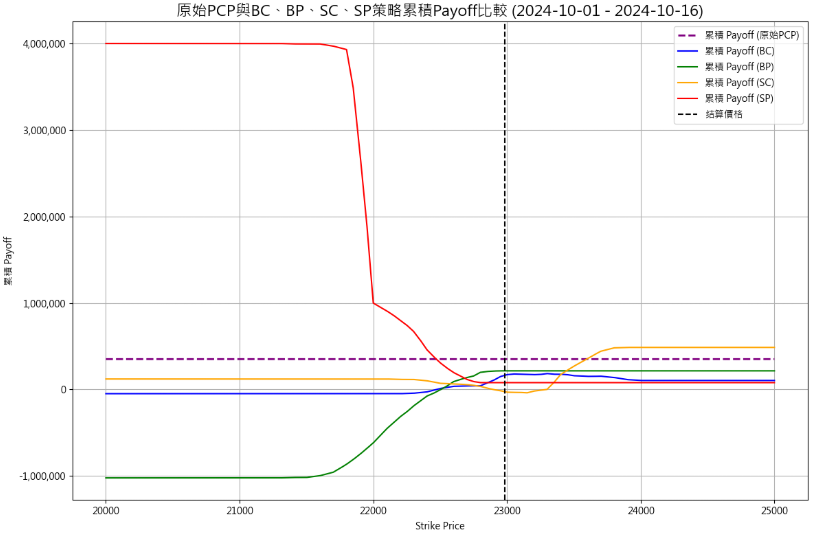
原始PCP：抓取StrikePirce、Fee (Total Fee)、ArbitrageStrategy

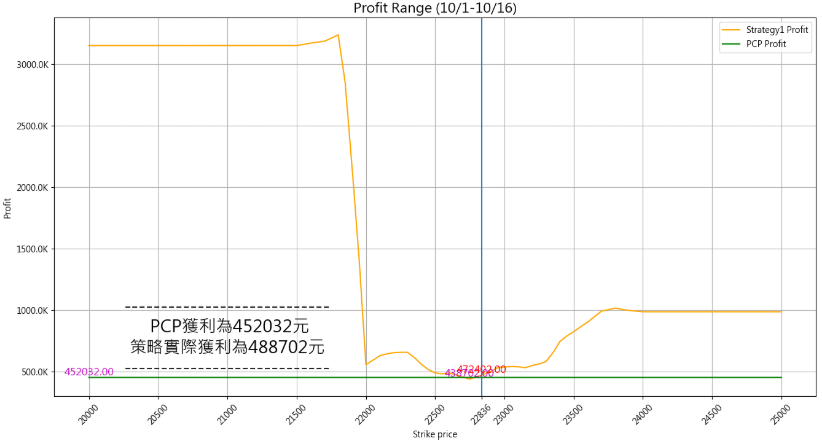
修正後PCP，call跟put履約價不同，可以直接抓CallID, PutID裡面的履約價數字：抓取Call\_Strike、Put\_Strike、Fee、ArbitrageStrategy

* 每一筆數據去觀察績效 -> 發現改良PCP Spread兩個履約價可能只差一點點，可是績效差非常多
* 多筆數據一同觀察
* 不同時間範圍下 -> 普遍日期之下都和下方圖表呈現類似，只有10/11, 10/14, 10/15是反方向

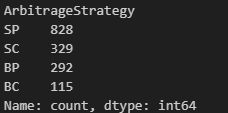
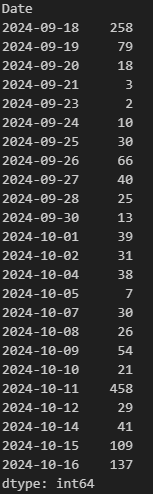


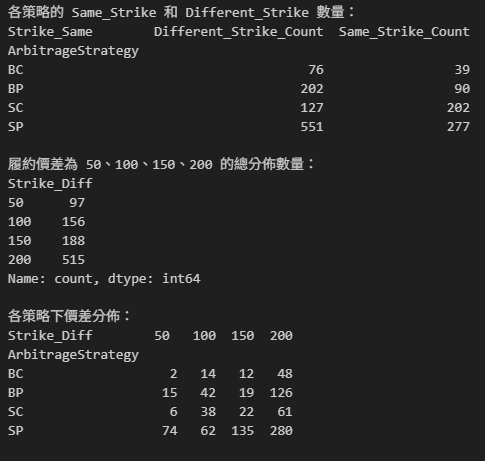
* BC, BP, SC, SP策略觀察，SP可能因數量大，所以Spread的差異顯著

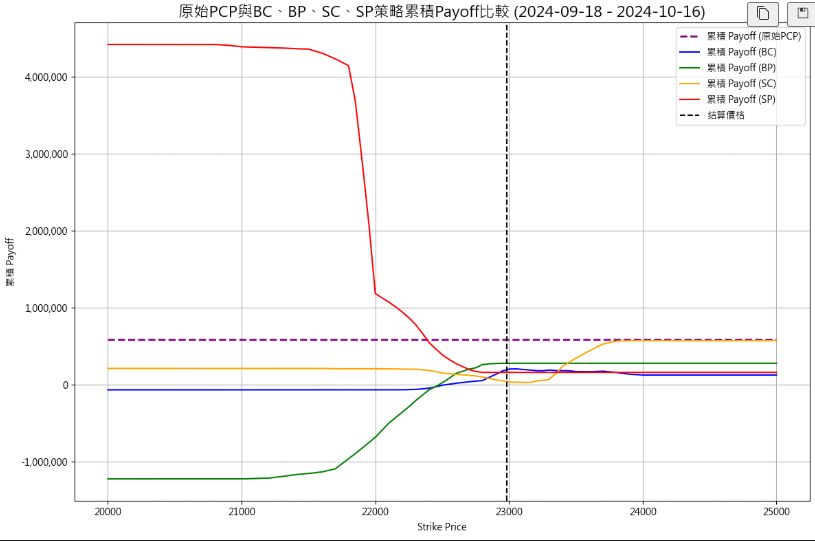


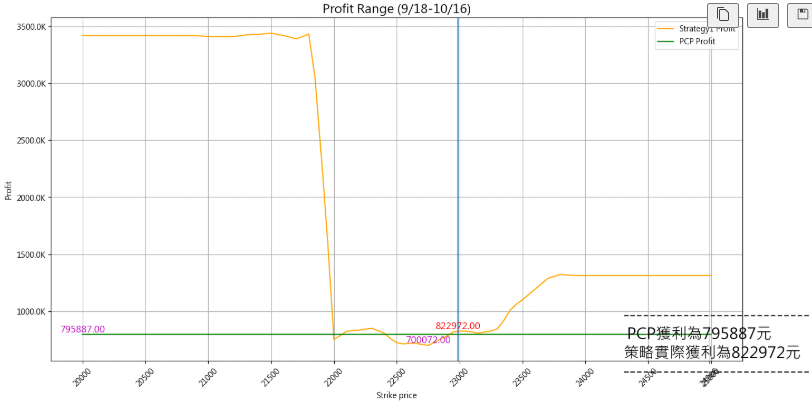


10月到期之數據：

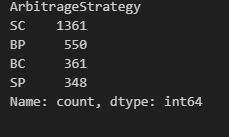
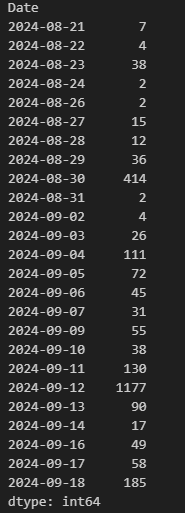


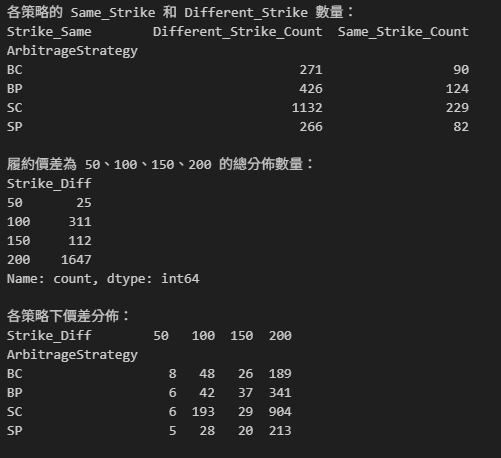


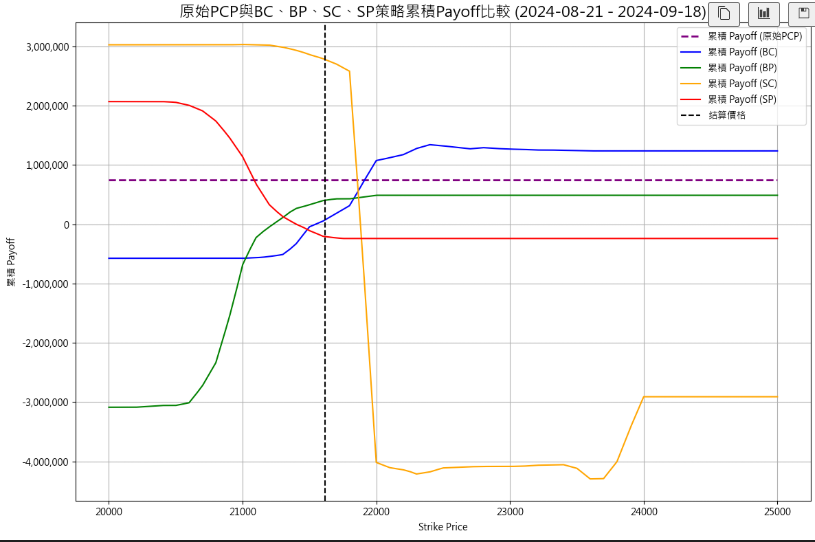


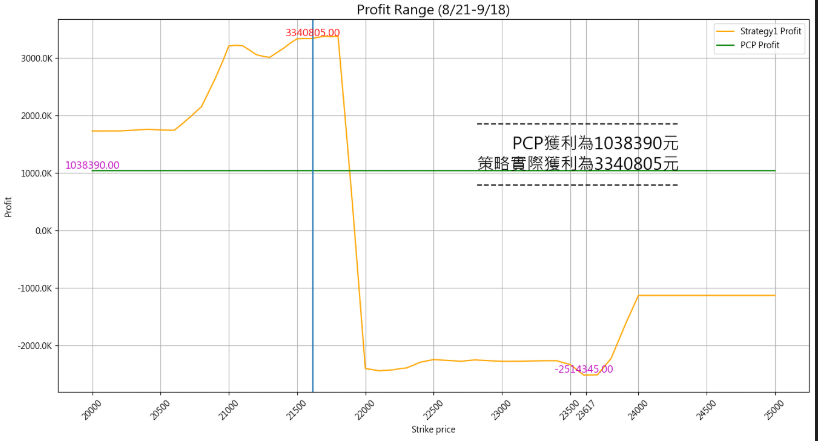


9月到期之數據：









其實改良版PCP沒有完全在初始版PCP之上，有一小部分在其下方 (不構成無風險套利)，不過接著要把稅、手續費、滑價成本加入做觀察 (原本其實有考慮fee，但不確定換第三隻腳後，fee要怎麼算)，這樣可能會更直觀

代辦：

1. **加入稅、手續費、滑價成本做觀察**
2. 針對原始PCP，新增新函數 Fee計算，最後分別以CallFee, PutFee, FutureFee來呈現，最後**可以比較與原先PCP數據上的Fee是否有差距**

目前來說，僅只有考慮買入時之價格，賣出的部分也暫時用買入的價錢來推波助瀾

* 買入費用

稅：

小台 0.00002\*50

選擇權 0.001\*50

手續費：

小台一口13.3元

選擇權一口10.8元

買賣都要稅&費

由於數據陳列時就是將每筆數據都獨立開來，因此可能會小幅低估/高估總花費

* 賣出/平倉費月

進入結算，選擇權有履約價值的，會持接收期貨稅 (期貨結算價多少，按照此價格收一點稅)，手續費會比平倉時便宜一點點

期貨也類似

1. 針對改良的PCP，腳1與期貨延續原始pcp中的數據，僅需要更新腳3數據)
2. **觀察各月份狀況 (目前只做過9、10月，8月的數據呈現有點怪怪的)，期望可以看過去5~10月的數據，抑或是調整成只看單月合約**
3. 另外，記得要和前期實習生策略做對比

設定腳3選擇權履約價在原本的K+-200區間內，找5分鐘內符合條件的選擇權，與原本的價格對比，買較便宜之選擇權、賣較貴之選擇權

1. 與選鄰近的腳3績效對比
2. **想觀察以時間為橫軸，PCP策略績效變化，同時觀察max drawdown跟delta 之變化**

期貨數據先用每天收盤價，每天觀察一次策略的績效

**動態累加、非等頻率建倉+每日資產價格波動 = 真實損益浮動必須逐日重估**

* 數據：

累積報酬、平均日報酬、日報酬標準差、最大回撤、夏普比率、勝率(贏過原始PCP的機會)、每筆交易平均報酬

* 累積報酬曲線、每日報酬變化圖、每日淨Delta圖、策略類型逐日報酬累積、最大回撤走勢圖

腳3類型 (BP、BC、SP、SC)，履約價範圍(K+-200區間內)、空頭or多頭價差

1. **篩選划算腳3其他方法**

操作想法：

透過第一層迴圈分是第幾筆PCP

第二層迴圈專注於該筆PCP，去抓取屬於其之FilteredTicks (df\_origin\_pcp也同樣專注在該筆數據)

針對每一筆PCP去跑回歸，去抓取各自的FilteredTicks

1. 抓出其履約價、IV、成交價格、成交數量去觀察

2. 使用履約價、IV去繪製IV CURVE (成交數量可當加權)

3. 去資料數據去擬和出一條IV CURVE

4. 比較是否有Outlier出現 (如果資料太少可能要回前面去修改抓取的時間範圍)

5. 透過以下條件做觀察

BP (Short Call + Long Put) - Put 的 IV SKEW：

1. 高於IV SKEW：使用原本的PCP策略

2. 低於IV SKEW：Long put

SC (Short Call + Long Put) - Call 的 IV SKEW：

1. 高於IV SKEW：Short Call

2. 低於IV SKEW：使用原本的PCP策略

SP (Long Call + Short Put) - Put 的 IV SKEW：

1. 高於IV SKEW：Short put

2. 低於IV SKEW：使用原本的PCP策略

BC (Long Call + Short Put) - Call 的 IV SKEW：

1. 高於IV SKEW：使用原本的PCP策略

2. 低於IV SKEW：Long call

3. 將新的PCP腳3及成交價格加入到PCP中

**5. 從bid ask spread切入 要自己算IV**

**6. 觀察PCP單以及成交的數量，觀察是否存在市場異常點**

**7. 觀察是否有超級便宜之腳3，可嘗試選擇用0作替代(不要腳3看風報比如何)**

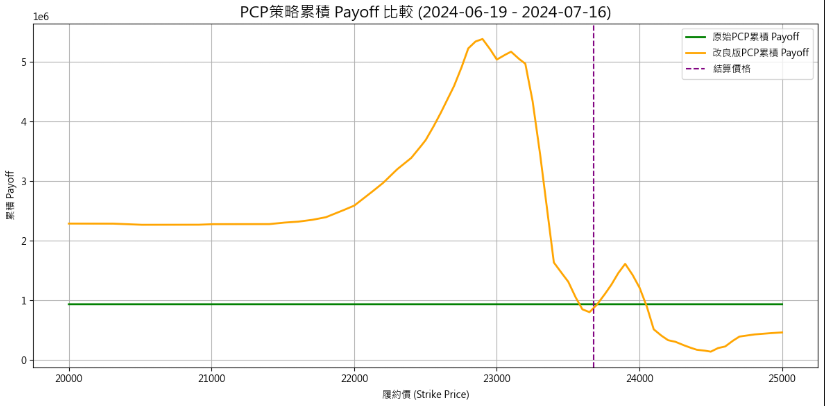
**8. 以秒資料去看PCP數據**

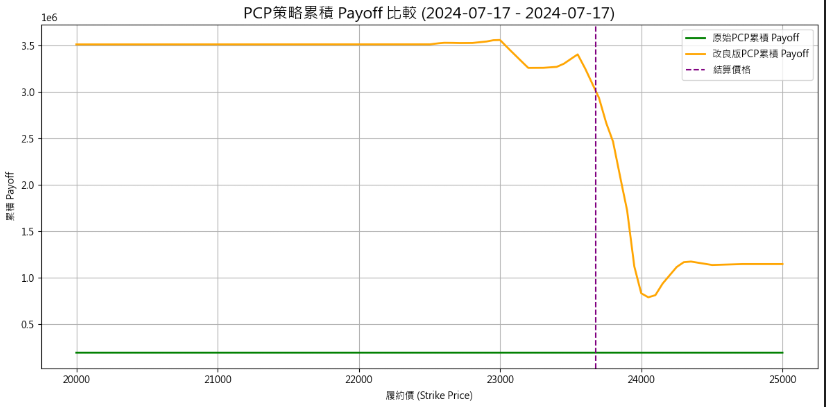
**9. 8月份台股大跌，觀察PCP績效**

現在的重點應該不是績效，而是能否找到有助於實戰下單的策略 (上衣個策略不錯，可是怎麼能確定便宜的是哪一個)

7月份：

7/17表現異常





8月份：

8/13的圖畫不出來

10月數據：

Filitered\_df 少了一些數據

問題點1：如果單以一筆PCP來看，有無風險套利/套損的嫌疑

問題點2：某些履約價、IV會是缺值

沒有找到腳3，主因是起初在篩選時間、履約價範圍內之所有可能性時就已經沒有filtered\_data了，反而不是在根據IV篩選時找不到最好的(代表

其他：

之後可以延伸做週PCP策略，只不過這比較複雜，因為週期貨沒有量，所以起初要用週選call, put + 月期組起來，然後透過價差商品(月期-週選)去做套利