配對Alpha測略

呂偉丞

Contents

配對Alpha策略

- 一種市場中性策略
- 不受市場波動影響，一多一空對沖市場風險
- 從市場上找出歷史走勢相近的商品進行配對，當配對的股票價格差(Spreads)偏離歷史均值時，由做空相對價格較高的商品通時買進相對價格較低的商品，等待他們回歸長期均衡關係，由此賺取價格收斂的報酬。

01 策略說明

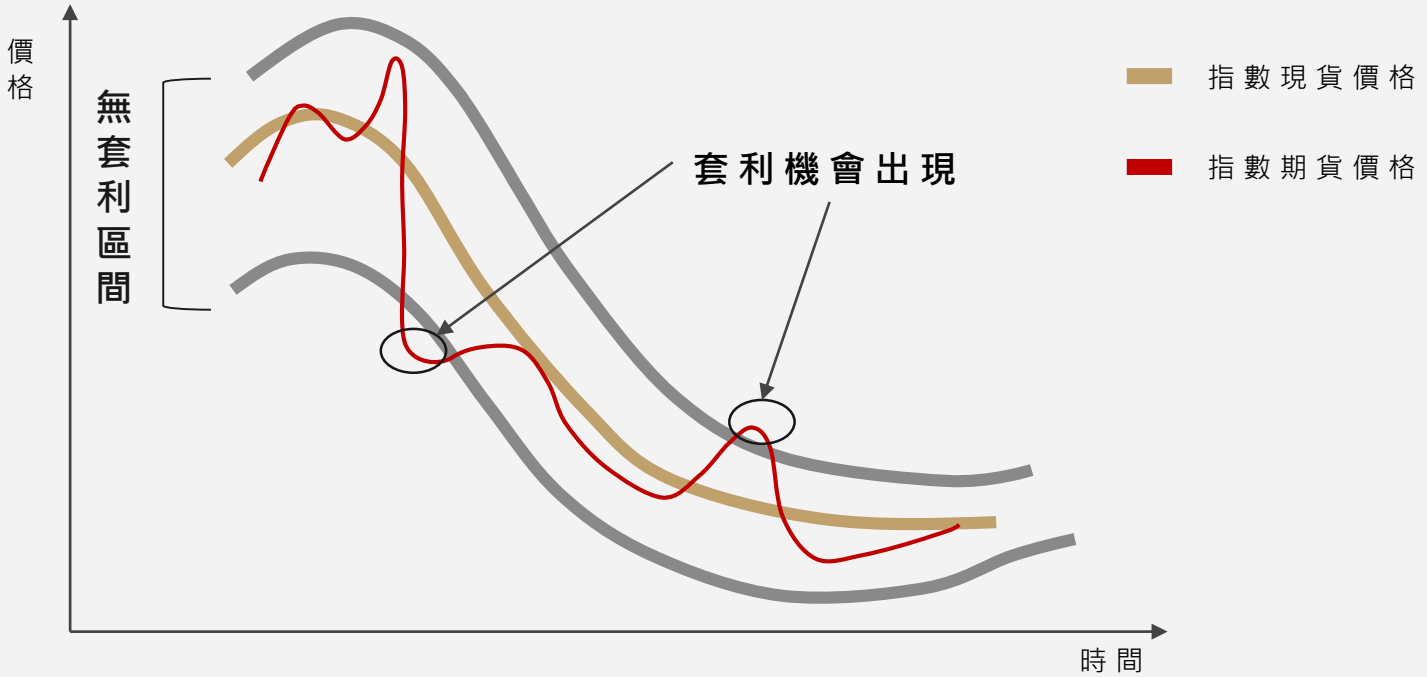
原理

| 因為假設X和Y服從$Y_t = \alpha + \beta X_t + \epsilon_t$的關係，我們重點關注係數$\beta$的變化。$\beta$不會一直是一個固定不變的常數，我們允許$\beta$在一個上下限波動

| 當β超出了上限的時候，說明Y_t“太貴”了、 X_t“太便宜”了，

對應的策略是short Y_t+long X_t

01 策略說明



期現套利原理示意圖

03 CODE說明

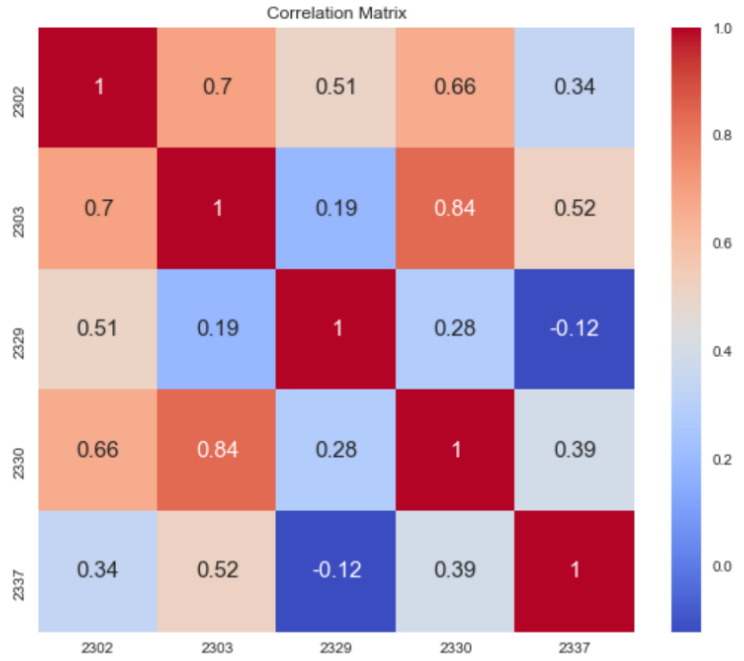
```
In [14]: close_df
```

Out[14]:

Date	2302	2303	2329	2330	2337
2010-01-04	6.222883	10.645722	17.125622	44.052135	26.497847
2010-01-05	5.975026	10.770967	16.549990	43.780632	26.203426
2010-01-06	6.054688	11.522430	16.334118	44.052135	26.350639
2010-01-07	5.842247	11.522430	16.406109	43.576996	25.982613
2010-01-08	5.824543	11.397185	16.693874	43.441250	25.835400
2010-01-11	5.851098	11.647675	16.334118	43.780632	25.614582
2010-01-12	5.895354	11.491119	16.549990	43.169735	26.497847
2010-01-13	5.886503	11.178008	15.974360	42.626728	26.350639
2010-01-14	6.090103	11.240632	16.693874	42.898228	26.350639
2010-01-15	6.222883	11.271941	16.549990	43.101856	26.203426
2010-01-18	6.364504	11.178008	16.262224	42.694599	26.056215

03 CODE說明

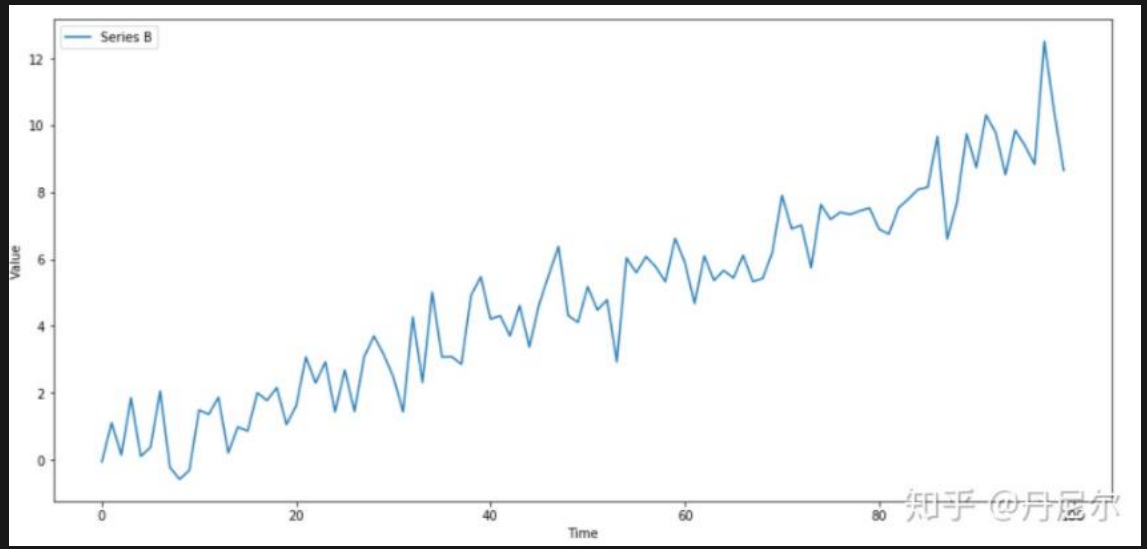
In [18]:
```
## 計算相關係數
import seaborn as sns

plt.figure(figsize=(9,7.5))
plt.title('Correlation Matrix')
sns.heatmap(close_df.corr(),annot=True,annot_kws={"size":15},cmap='coolwarm')
plt.show()
```



03 CODE說明

平穩性（Stationarity）

定義：如果數據產生過程的參數不隨著時間變化，那麼數據平穩



測試平穩性：使用ADF TEST，P值越小，說明越可能是平穩的時間序列。

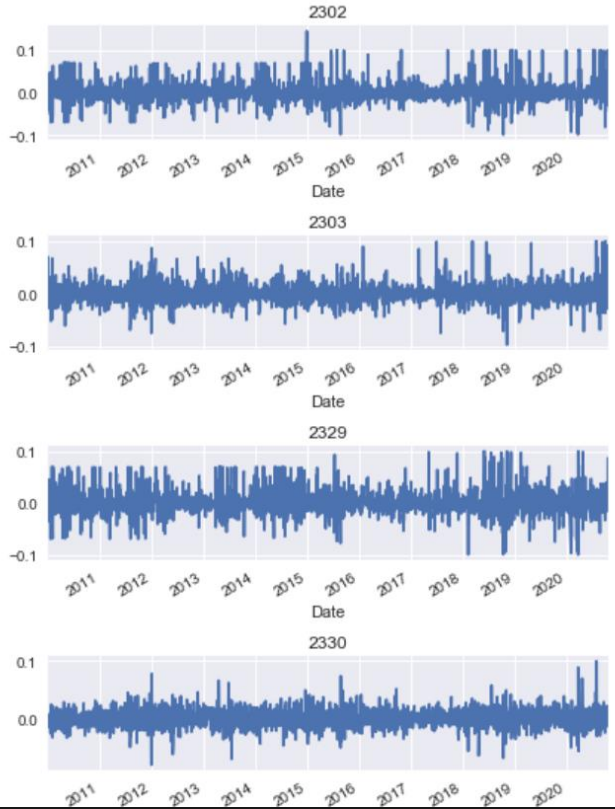
03 CODE說明

```
In [8]: adfuller(df['2330'])

Out[8]: (2.432817510436287,
         0.9990249264195827,
         15,
         2105,
         {'1%': -3.433460352623917,
          '5%': -2.862914023960907,
          '10%': -2.5675014652930193},
         11007.890244125096)
```

2-1. 差分後相關係數

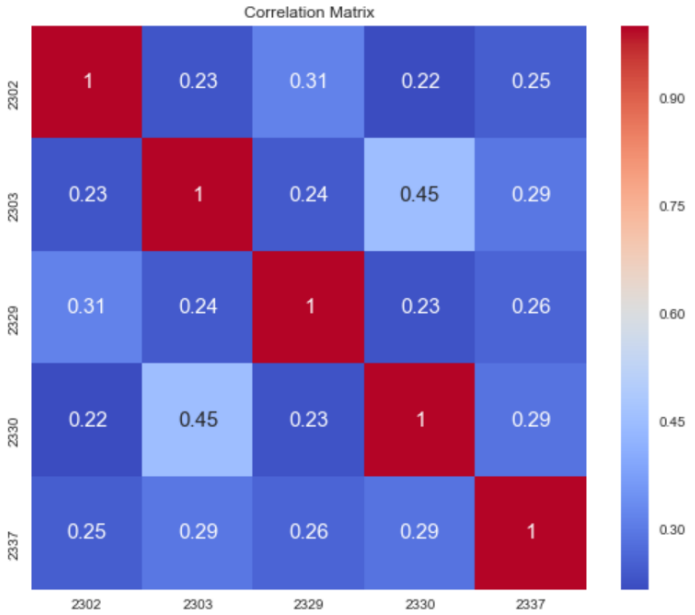
In [19]:
```
plt.figure(figsize=(6,10))
for col in close_df:
    plt.subplot(5,1,close_df.columns.to_list().index(col)+1)
    plt.title(col)
    close_df[col].plot()
plt.tight_layout()
```

In [20]:
```
plt.figure(figsize=(6,10))
for col in close_df:
    plt.subplot(5,1,close_df.columns.to_list().index(col)+1)
    plt.title(col)
    close_df[col].pct_change(periods=1,fill_method="ffill").plot()
plt.tight_layout()
```



03 CODE說明

In [21]:
```
## 解決trend，做一階差分

plt.figure(figsize=(9,7.5))
plt.title('Correlation Matrix');
sns.heatmap(close_df.pct_change(periods=1,fill_method="ffill").corr(),annot=True,annot_kws={"size":15},cmap='coolwarm')
plt.show()
```



03 CODE說明

3. 價差比例

In [24]:
```
stock_1='2302'
stock_2='2329'
```

In [26]:
```
plt.figure(figsize=(12,6))
plt.title('{} & {} Adj Close'.format(stock_1,stock_2),fontsize=16)

ax1 = close_df[stock_1].plot()
ax2 = close_df[stock_2].plot(secondary_y=True)

plt.grid()

lines = ax1.get_lines() + ax2.get_lines()
plt.legend(lines,[l.get_label() for l in lines])
plt.show()
```

03 CODE說明

3-1. 計算價差比例

In [36]:
```
#計算價差比例

Spread_Ratio = close_df[stock_1]/close_df[stock_2]
Spread_Ratio_Mean = Spread_Ratio.mean()
Spread_Ratio_Std = Spread_Ratio.std()

print("Spread_Ratio_Mean :",Spread_Ratio_Mean)
print("Spread_Ratio_Std:",Spread_Ratio_Std)
```

```
Spread_Ratio_Mean : 0.5252937487499353
Spread_Ratio_Std: 0.15018963846376376
```

03 CODE說明

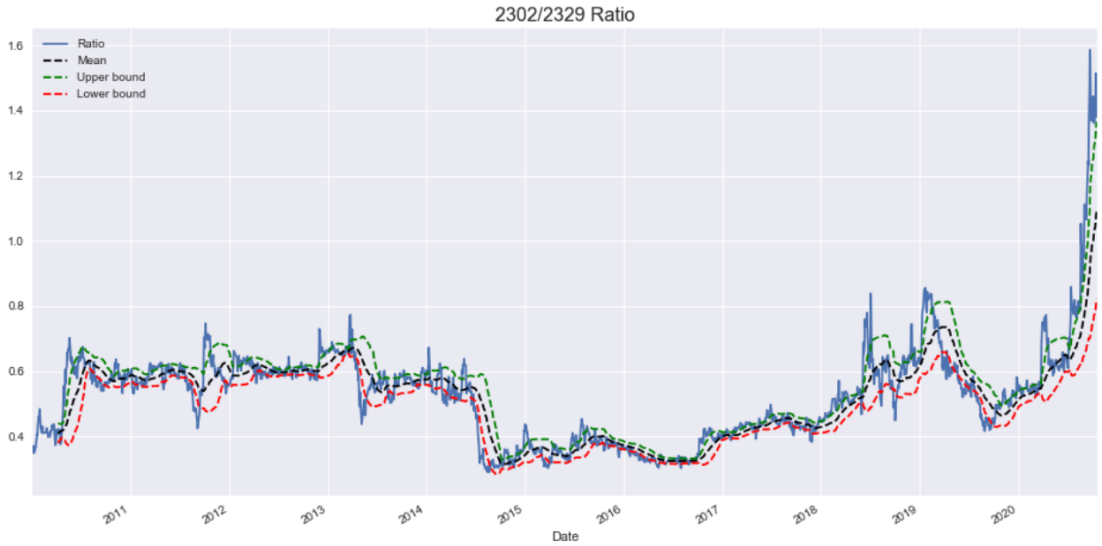
In [37]:
```
crit=1

plt.figure(figsize=(12,6))
plt.title('{}/{} Ratio'.format(stock_1,stock_2),fontsize=16)

Spread_Ratio.plot(label='Ratio')

plt.axhline(Spread_Ratio_Mean,label='Mean',ls='--',c='black')
plt.axhline(Spread_Ratio_Mean+crit*Spread_Ratio_Std,label='Upper bound',ls='--',c='g')
plt.axhline(Spread_Ratio_Mean-crit*Spread_Ratio_Std,label='Lower bound',ls='--',c='r');

plt.legend()
```

Out[37]:
```
<matplotlib.legend.Legend at 0x27d66fcc2e8>
```

03 CODE說明

In [45]:
```
window=60
crit=1

Spread_Ratio_MA = Spread_Ratio.rolling(window=window).mean()
Spread_Ratio_rolling_Std = Spread_Ratio.rolling(window=window).std()
upper_bound = Spread_Ratio_MA+crit*Spread_Ratio_rolling_Std
lower_bound = Spread_Ratio_MA-crit*Spread_Ratio_rolling_Std
```

In [46]:
```
plt.figure(figsize=(16,8))
plt.title('{}/{} Ratio'.format(stock_1,stock_2),fontsize=16)

Spread_Ratio.plot(label='Ratio')
Spread_Ratio_MA.plot(label='Mean',ls='--',c='black')
upper_bound.plot(label='Upper bound',ls='--',c='g')
lower_bound.plot(label='Lower bound',ls='--',c='r');

plt.legend();
```



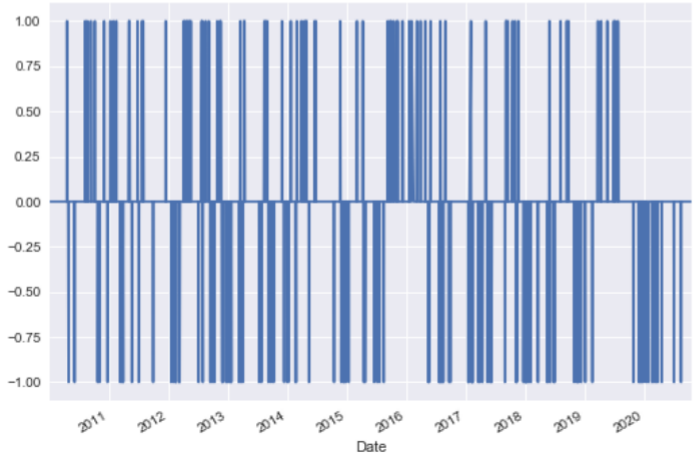
03 CODE說明

4-1. 產出訊號

In [47]:
```
#建立交易訊號
#穿過上通道時定義為放空投資組合
#穿過下通道時定義為買進投資組合
#建立一個list，當達成條件時append進-1與1

signal=[]
```

In [48]:
```
## 組成series
signal_df=pd.Series(signal,index=Spread_Ratio.index)

plt.figure(figsize=(8.5,6))
signal_df.plot()
```

Out[48]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x27d67b52e48>
```



03 CODE說明

4-3. 計算交易股數

In [52]:
```
## 設定總資金，分兩半投入兩邊
## 計算每次"訊號"發生時兩邊要兩邊買幾張

wealth=1000000.0
trade_stock_df=pd.concat(
    [signal_df.shift(1)*(wealth/2/close_df[stock_1].shift(1)),-signal_df.shift(1)*(wealth/2/close_df[stock_2]).shift(1)]
    ,axis=1,keys=[stock_1,stock_2])

trade_stock_df.iloc[67:75]
```

Out[52]:

	2302	2329
Date		
2010-04-19	83681.642400	-31441.855255
2010-04-20	0.000000	-0.000000
2010-04-21	0.000000	-0.000000
2010-04-22	0.000000	-0.000000
2010-04-23	0.000000	-0.000000
2010-04-26	0.000000	-0.000000
2010-04-27	-72139.372109	34399.167682
2010-04-28	0.000000	-0.000000

03 CODE說明

4-4. 計算標的每日變化

In [54]:
```
## 計算每天股價變化
return_df=pd.concat([close_df[stock_1].diff(1),close_df[stock_2].diff(1)],axis=1,keys=[stock_1,stock_2])
return_df.iloc[67:75]
```

Out[54]:

	2302	2329
Date		
2010-04-19	-0.265560	-1.079369
2010-04-20	0.061969	-0.503640
2010-04-21	0.132771	-0.100786
2010-04-22	0.159343	0.014398
2010-04-23	0.416032	0.590029
2010-04-26	0.451447	-0.287766
2010-04-27	0.478001	-0.071991
2010-04-28	0.513416	-0.172679

03 CODE說明

4-5. 計算投組每日變化

In [58]:
```
## 股價變化*持有部位 = 每日損益，且以.cumsum()來計算累積損益
## 建持有單一股票的累積損益和對沖損益df

trade_return_df=(return_df*trade_stock_df)
cum_trade_return_df=(return_df*trade_stock_df).cumsum()
cum_trade_return_df['Total Return']=cum_trade_return_df.sum(axis=1)

cum_trade_return_df.iloc[67:75]
```

Out[58]:

Date	2302	2329	Total Return
2010-04-19	-22222.469618	33937.351014	11714.881397
2010-04-20	-17036.778469	49772.732493	32735.954024
2010-04-21	-5926.321759	52941.637891	47015.316132
2010-04-22	7407.742589	52488.949971	59896.692559
2010-04-23	42222.009941	33937.351014	76159.360955
2010-04-26	79999.797295	42985.232305	122985.029600
2010-04-27	45517.131195	40508.802966	86025.934162
2010-04-28	8479.602384	34568.790898	43048.393281

03 CODE說明

In [59]:
```
## 比較單獨持有和對沖持有的損益
fig,ax=plt.subplots(figsize=(16,6))

cum_trade_return_df.plot(label='Total Return',ax=ax)
plt.legend()
plt.title('Stock & Total Return',fontsize=16)
```

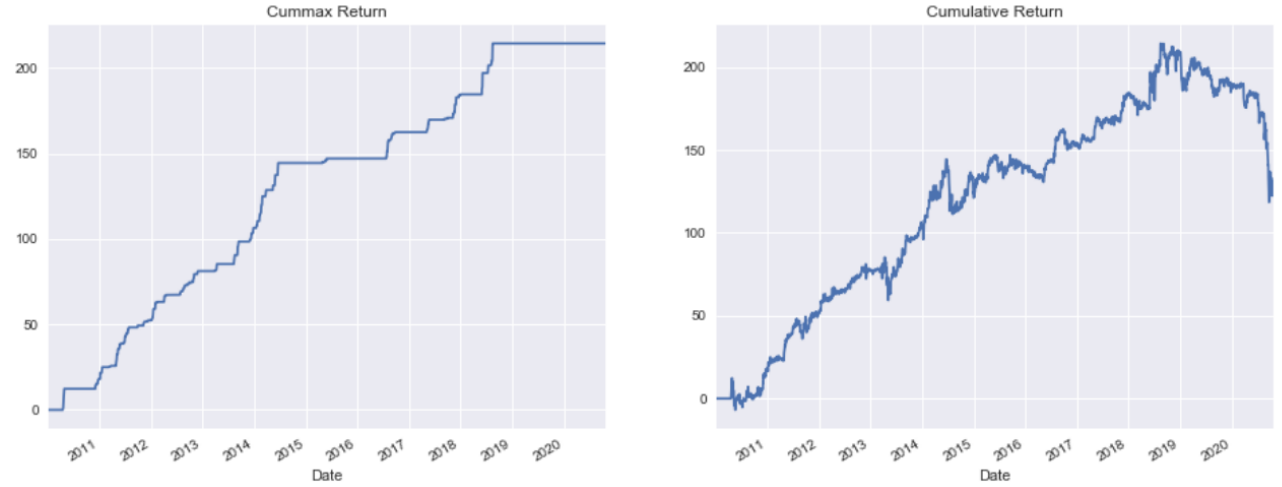
Out[59]:
```
Text(0.5, 1.0, 'Stock & Total Return')
```

03 CODE說明

4-6. 計算累積報酬 (每次都用100萬當本金，不考慮現金收入)

In [60]:
```
## 建累機報酬率的df
## 並且以.cummax()來計算累積報酬創高點

cum_trade_percent_return=(cum_trade_return_df['Total Return']/wealth)*100

fig=plt.figure(figsize=(16,6))
plt.subplot(1,2,1)
cum_trade_percent_return.cummax().plot()
plt.title('Cummax Return')
plt.subplot(1,2,2)
cum_trade_percent_return.plot()
plt.title('Cumulative Return')
```

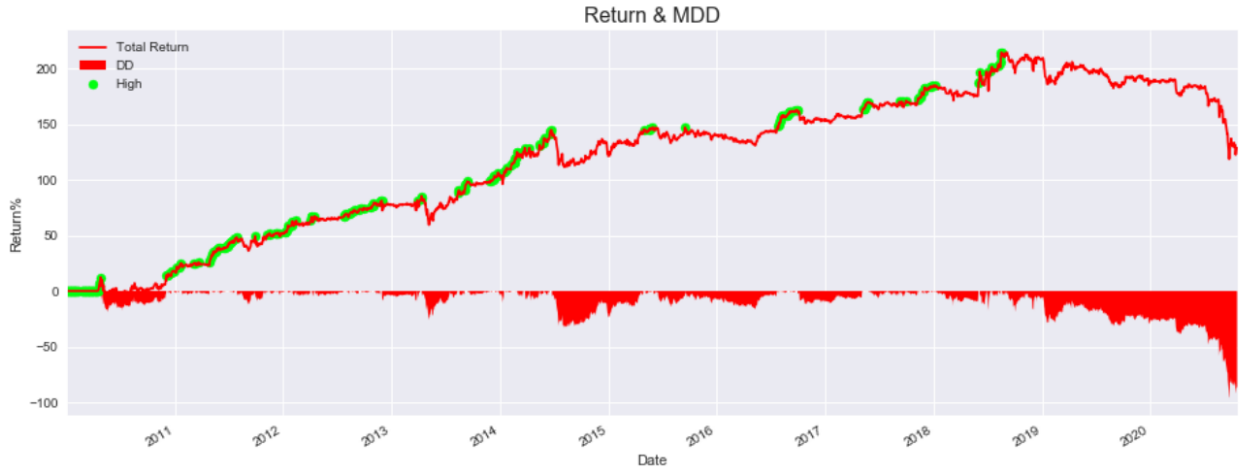
Out[60]: Text(0.5, 1.0, 'Cumulative Return')



03 CODE說明

4-7. 績效指標與畫圖

In [63]:
```
fig,ax=plt.subplots(figsize=(16,6))

(cum_trade_percent_return).plot(label='Total Return',ax=ax,c='r')
plt.fill_between(MDD_series.index,-MDD_series,0,facecolor='r',label='DD')
plt.scatter(high_index,cum_trade_percent_return.loc[high_index],c='#02ff0f',label='High')

plt.legend()
plt.ylabel('Return%')
plt.xlabel('Date')
plt.title('Return & MDD',fontsize=16);
```



03 CODE說明

In [64]:
```
MDD=round(MDD_series.max(),2)
Cumulative_Return=round(cum_trade_percent_return.iloc[-1],2)
Return_on_MDD=round(cum_trade_percent_return.iloc[-1]/MDD_series.max(),2)
daily_return=cum_trade_percent_return.diff(1)

print('Cumulative Return: {}%'.format(Cumulative_Return))
print('MDD: {}%'.format(MDD))
print('Return on MDD: {}'.format(Return_on_MDD))
print('Shapre Ratio: {}'.format(round((daily_return.mean()/daily_return.std())*pow(252,0.5),2)))
```

```
Cumulative Return: 128.37%
MDD: 95.81%
Return on MDD: 1.34
Shapre Ratio: 0.51
```

In [65]:
```
abs(position_df.diff(1)).sum()*0.005650*100
```

Out[65]:
```
71.755
```

Thanks

附　錄

初論Alpha & Beta

- William Sharpe在1964年首次提出，並指出投資者在市場中交易面臨系統性風險和非系統性風險，公式表達如下：
  - $E(R_p)=R_f+\beta(R_m-R_f)$

- Market Model:
  - $R_j=\alpha_j+\beta_j R_m+\varepsilon_j\ ,j=1,\ldots,n$

- Beta策略
  - 利用各種方法交易市場趨勢，獲得報酬率的一種投資方式。

傳統Alpha策略

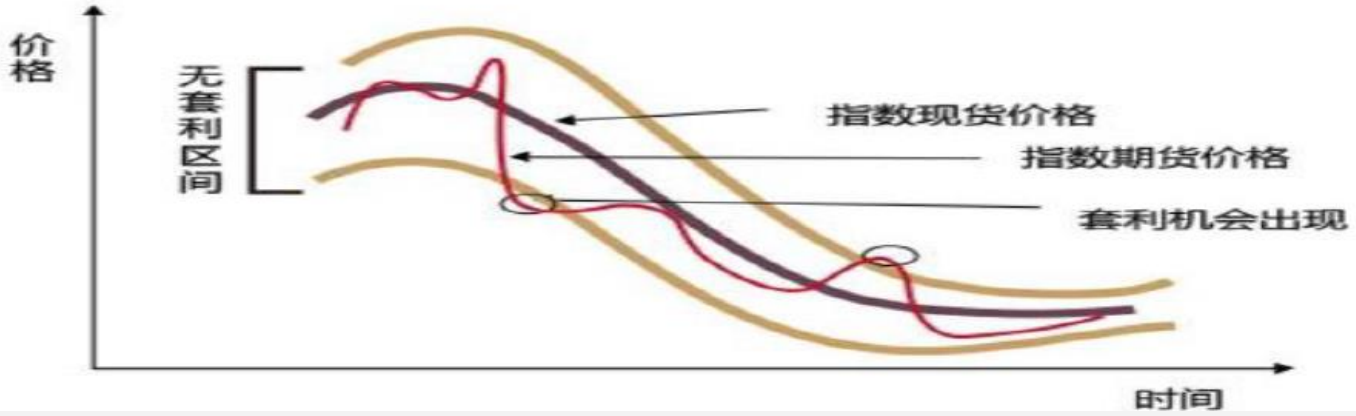
- 是一種市場中性策略:
- 不受市場波動度影響，同時持有多空對沖市場風險賺取超額報酬

- $R_j = \alpha_j + \beta_j R_m + \varepsilon_j \rightarrow +R_j - \beta_j R_m = \alpha_j + \varepsilon_j$
- 多R_j報酬率的商品，放空β_j比例R_m報酬率的商品，獲得超額報酬

- 透過因子的選擇挑選個股，挑選出擁有正Alpha的個股組成投資組合對沖期貨。
  1. 基本面因子：營業利益率、毛利率、EPS成長率、P/E、P/B等
  2. 價量因子：動能 Jegadeesh and Titman (1993)、波動度 、各種技術指標等

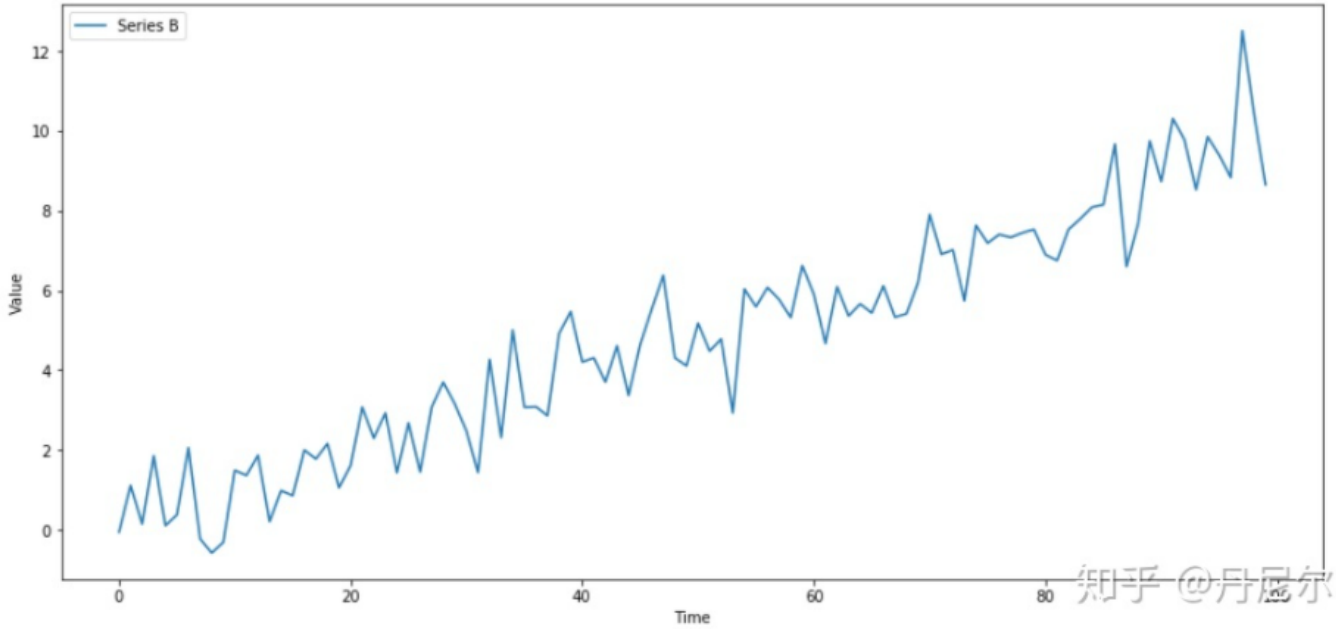
配對Alpha策略

- 是一種市場中性策略:
- 不受市場波動度影響，一多一空對沖市場風險
- 從市場上找出歷史走勢相近的商品進行配對，當配對的股票價格差(Spreads)偏離歷史均值時，由做空相對價格較高的商品通時買進相對價格較低的商品，等待他們回歸長期均衡關係，由此賺取價格收斂的報酬。



2、平穩性（Stationarity）

- 定義：如果數據產生過程的**參數**（例如均值和方差）**不隨著時間變化**，那麼數據**平穩**。
- 例如，時間序列 x_t 的均值和方差是常數、和時間t無關，那麼 x_t 具有平穩性。
- 如果用平穩的分析刻畫一個均值/方差隨時間變化的時間序列，那麼將會導致garbage in garbage out。
  - 例如，下圖series B，是一個隨著時間t而均值不斷增大的曲線。



協整關係與配對交易連結