

```
In [1]: #Disclaimer: The relative arbitrage strategy was
#not fully implemented until October, 2022.
#Prior to October, 2022, it was a mixture of mostly
#put spread and a few ITM call as well as futures
#for quick delta adjustment.
#Since then, this relative arbitrage strategy has
#been fully and consistently implemented.
```

```
In [2]: import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

```
In [3]: # Load the Excel file
excel_file = pd.ExcelFile('E:\Derivatives Trading\TAIEX derivatives trading record

# Get the sheet you want to read
sheet_name = 'ForPython' # Replace with the name of the sheet you want to read
df = excel_file.parse(sheet_name)
```

```
In [4]: # Output data information
print(df.head())
```

	Date	PnL Index	TAIEX	VIX	Returns	Unnamed: 5	Unnamed: 6	\
0	2022-07-01	100.000000	14343.08	27.01	0.000000	NaN	NaN	
1	2022-07-04	95.577858	14217.06	27.56	-0.044221	NaN	NaN	
2	2022-07-05	93.953178	14349.20	27.18	-0.016998	NaN	NaN	
3	2022-07-06	92.057052	13985.51	29.40	-0.020182	NaN	NaN	
4	2022-07-07	92.698962	14335.27	28.26	0.006973	NaN	NaN	

	Base
0	100.0
1	NaN
2	NaN
3	NaN
4	NaN

```
In [5]: #*****Plotting setup*****#
# Generate some data
Date = df["Date"]
Date
y1 =df["PnL Index"]
y1
y2 = df["TAIEX"]
y2
```

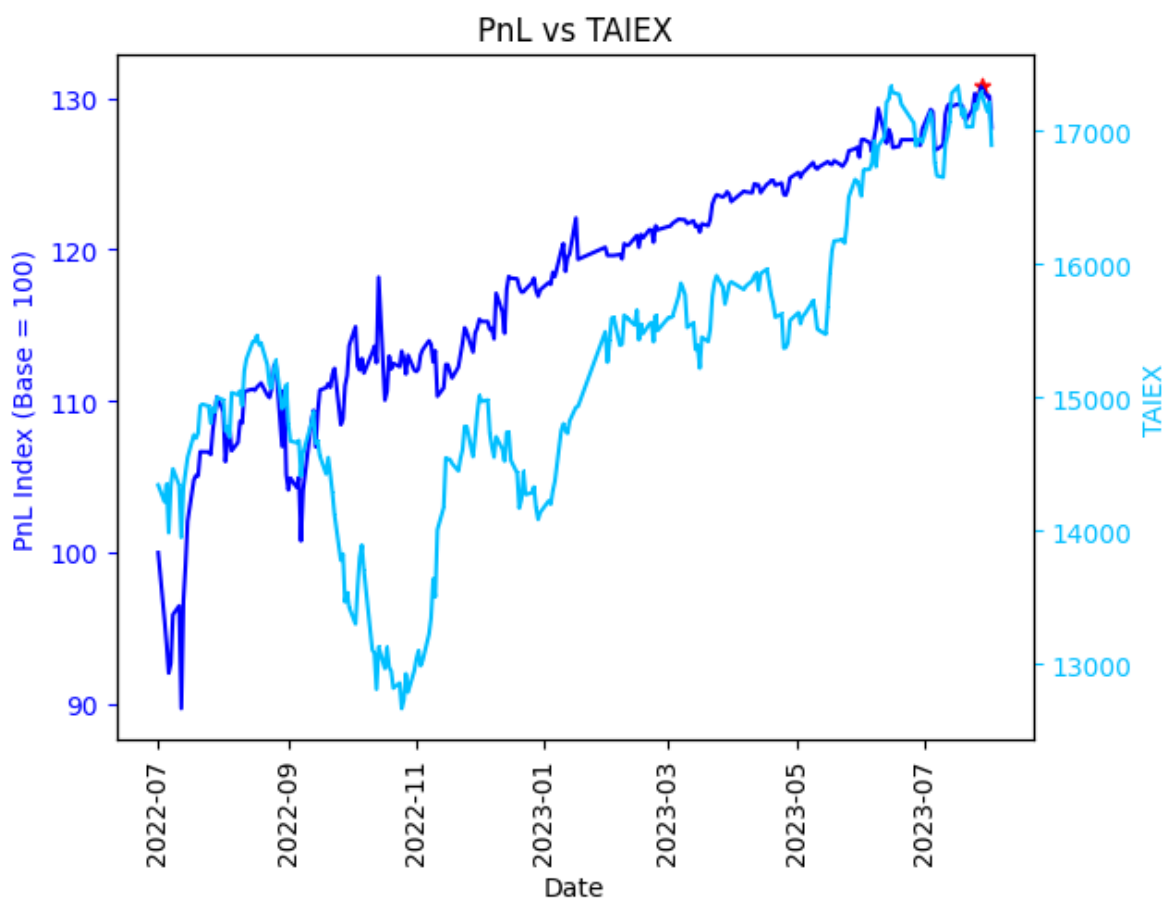
```
Out[5]: 0      14343.08
1      14217.06
2      14349.20
3      13985.51
4      14335.27
...
259    17241.82
260    17292.93
261    17145.43
262    17212.87
263    16893.73
Name: TAIEX, Length: 264, dtype: float64
```

```
In [6]: # Get the maximum PnL value
max_pnl = df['PnL Index'].max()
max_pnl_date = df.loc[df['PnL Index']==max_pnl, 'Date'].values[0]
```

```
In [7]: # Create the plot and set the first y-axis (left)
fig, ax1 = plt.subplots()
plt.xticks(rotation=90)
ax1.plot(Date, y1, 'b-')
ax1.scatter(max_pnl_date, max_pnl, color='red', marker='*')
ax1.set_xlabel('Date')
ax1.set_ylabel('PnL Index (Base = 100)', color='b')
ax1.tick_params('y', colors='b')

# Set the second y-axis (right)
ax2 = ax1.twinx()
ax2.plot(Date, y2, color='deepskyblue', marker=',')
ax2.set_ylabel('TAIEX', color='deepskyblue')
ax2.tick_params('y', colors='deepskyblue')

# Show the plot
plt.title('PnL vs TAIEX')
plt.show()
```

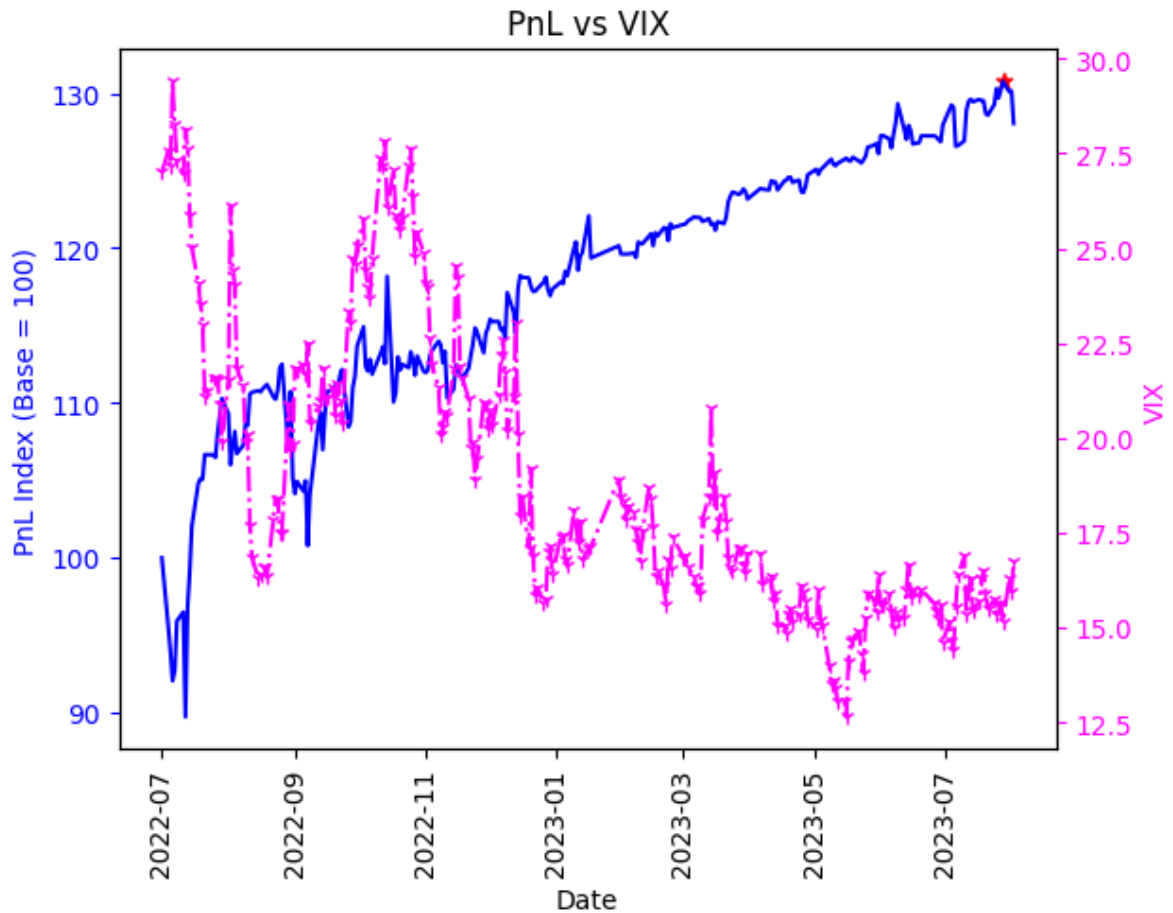


```
In [8]: #PnL vs VIX
y3 = df["VIX"]
y3

# Create the plot and set the first y-axis (left)
fig, ax1 = plt.subplots()
plt.xticks(rotation=90)
ax1.plot(Date, y1, 'b-')
ax1.scatter(max_pnl_date, max_pnl, color='red', marker='*')
ax1.set_xlabel('Date')
ax1.set_ylabel('PnL Index (Base = 100)', color='b')
ax1.tick_params('y', colors='b')
```

```
# Set the second y-axis (right)
ax3 = ax1.twinx()
ax3.plot(Date, y3, 'fuchsia', marker='1', linestyle='-.')
ax3.set_ylabel('VIX', color='fuchsia')
ax3.tick_params('y', colors='fuchsia')

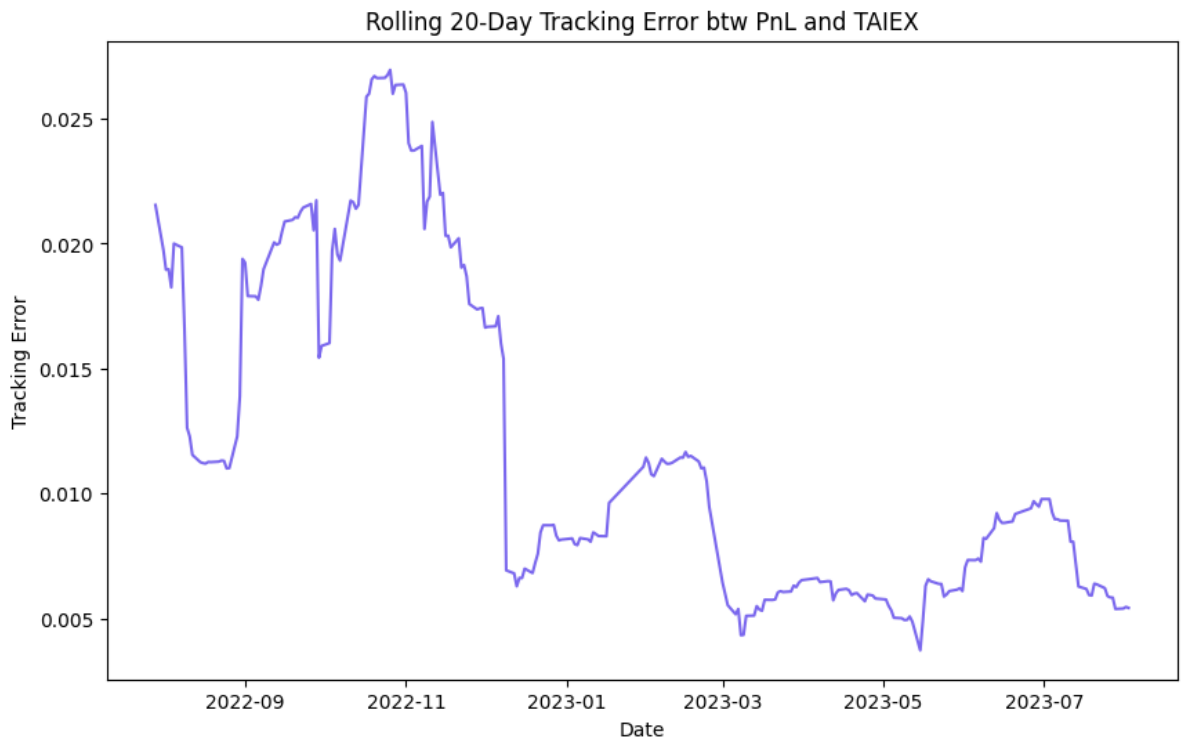
# Show the plot
plt.title('PnL vs VIX')
plt.show()
```



```
In [9]: #Tracking error between PnL and TAIEX
PNL_returns = df['PnL Index'].pct_change()
TAIEX_returns = df['TAIEX'].pct_change()
diff_returns = PNL_returns - TAIEX_returns
tracking_error = diff_returns.std()

roll_te = diff_returns.rolling(20).std()

plt.figure(figsize=(10, 6))
plt.title('Rolling 20-Day Tracking Error btw PnL and TAIEX')
plt.plot(df['Date'], roll_te, color='mediumslateblue')
plt.xlabel('Date')
plt.ylabel('Tracking Error')
plt.show()
```



```
In [10]: #Least Squares algo
from scipy.optimize import least_squares

# Set lower and upper bounds
bounds =(10, 45)

# Objective function
def f(vix, PNL_returns , TAIEX_returns):
    diff = (TAIEX_returns* vix)-(PNL_returns*vix)
    return diff.std()

# Set initial guess within bounds
x0 = [15.0]

# Minimize tracking error
result2 = least_squares(f, x0, bounds=bounds, args=(TAIEX_returns, PNL_returns))
optimal_vix = result2.x[0]

print("Optimal VIX:", optimal_vix)
print("Minimum Tracking Error:", f(optimal_vix, TAIEX_returns, PNL_returns))

Optimal VIX: 10.000000232305702
Minimum Tracking Error: 0.14072798052001095
```

```
In [11]: #Sharpe ratio
# Read in the portfolio returns data from a CSV file
R_first=df["PnL Index"].iloc[0,]
R_first
R_last=df["PnL Index"].iloc[165,] #Always excel's actual row-2
R_last
```

```
Out[11]: 121.98400800102736
```

```
In [12]: portfolio_returns=(R_last-R_first)/R_first
portfolio_returns
```

```
Out[12]: 0.21984008001027364
```

```
In [13]: daily_returns=df["Returns"]
daily_returns
```

```
Out[13]: 0      0.000000
1     -0.044221
2     -0.016998
3     -0.020182
4      0.006973
...
259    0.003840
260    0.004398
261   -0.005165
262    0.000384
263   -0.016087
Name: Returns, Length: 264, dtype: float64
```

```
In [14]: # Max Drawdown Calculation for PnL Index
cumulative_returns = (1 + df["Returns"]).cumprod()
cumulative_max = cumulative_returns.cummax()
drawdown = (cumulative_returns / cumulative_max) - 1
max_drawdown = drawdown.min()

print("Max Drawdown:", max_drawdown)
```

Max Drawdown: -0.10420949154156467

```
In [15]: # Calculate the Profit Factor
positive_returns = daily_returns[daily_returns > 0].sum()
negative_returns = daily_returns[daily_returns < 0].sum()

# Avoid division by zero
if negative_returns != 0:
    profit_factor = abs(positive_returns / negative_returns)
else:
    profit_factor = float('inf')

print("Profit Factor:", profit_factor)
```

Profit Factor: 1.3060785904133356

```
In [16]: # Calculate the excess returns and standard deviation
risk_free_rate = 0.0145 # Taiwan savings rate
excess_returns = portfolio_returns - risk_free_rate
std_dev = np.std(daily_returns)
print("Standard Deviation of Daily Return:", std_dev)
```

Standard Deviation of Daily Return: 0.013878500996129924

```
In [17]: # Calculate the Sharpe ratio
Sharpe_Ratio = excess_returns / std_dev
print("Sharpe Ratio:", Sharpe_Ratio)
```

Sharpe Ratio: 14.795551772308373

```
In [18]: #Annualized Sharpe ratio
Annualized_Sharpe_Ratio=Sharpe_Ratio*np.sqrt(250)
print("Annualized Sharpe Ratio:", Annualized_Sharpe_Ratio)
```

Annualized Sharpe Ratio: 233.9382141971772

```
In [ ]:
```

```
In [ ]:
```

