Asset Pricing Coursework

Group 8:

Prachin Patel

Lachezar Rusakov

Zhao Shen

Jinda Wan

```
# Importing required libraries & functions

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import cvxpy
!pip install PyPortfolioOpt
from pypfopt.efficient_frontier import EfficientFrontier
from pypfopt import risk_models
from pypfopt import expected_returns
from pypfopt import plotting
from pypfopt import objective_functions
from scipy.optimize import minimize
```

```
    Collecting PyPortfolioOpt
      Downloading pyportfolioopt-1.5.5-py3-none-any.whl (61 kB)
         ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 61.9/61.9 kB 1.5 MB/s eta 0:
    Requirement already satisfied: cvxpy<2.0.0,>=1.1.19 in /usr/local/lib/pyth
    Requirement already satisfied: numpy<2.0.0,>=1.22.4 in /usr/local/lib/pyth
    Requirement already satisfied: pandas>=0.19 in /usr/local/lib/python3.10/d
    Requirement already satisfied: scipy<2.0,>=1.3 in /usr/local/lib/python3.1
    Requirement already satisfied: osqp>=0.4.1 in /usr/local/lib/python3.10/di
    Requirement already satisfied: ecos>=2 in /usr/local/lib/python3.10/dist-p
    Requirement already satisfied: scs>=1.1.6 in /usr/local/lib/python3.10/dis
    Requirement already satisfied: setuptools>65.5.1 in /usr/local/lib/python3
    Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/py
    Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/d
    Requirement already satisfied: qdldl in /usr/local/lib/python3.10/dist-pac
    Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
    Installing collected packages: PyPortfolioOpt
    Successfully installed PyPortfolioOpt-1.5.5
```

```
# Importing the excel file

df = pd.read_excel('cw2023AP.xlsx')
df
```

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6
0	NaT	NaN	NaN	NaN	NaN	NaN	NaN
1	NaT	Stock Price	NaN	NaN	NaN	NaN	NaN
2	NaT	1	2	3	4	5	6
3	NaT	BP	HSBC HOLDINGS	TESCO	VODAFONE GROUP	BUNZL	ROLLS-ROYCE HOLDINGS
4	NaT	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...
274	2023-05-31	453.3	590.5	260.8	76.37	3144	143.3
275	2023-06-30	458.35	621.7	248.4	73.97	2999	151.05
276	2023-07-31	483	646.3	258	74.43	2888	184.85
277	2023-08-31	487.5	583.1	265.7	73.21	2830	222.3
278	2023-09-29	531.4	644.9	264.2	76.82	2926	220.9

279 rows × 20 columns

```
# Data pre-processing & cleaning

df = df.iloc[3:]
df.columns = df.iloc[0]
df.reset_index(drop=True, inplace=True)
df = df.iloc[1:]
df = df.rename(columns={df.columns[0]: 'Date'})
```

```
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)
df = df.dropna(axis=1, how='all')
df = df.dropna(axis=0, how='all')
df
```

3	BP	HSBC HOLDINGS	TESCO	VODAFONE GROUP	BUNZL	ROLLS-ROYCE HOLDINGS	EASYJET	LEGAL & GENERAL
Date								
2000-12-29	540	858.36	272.35	251.39	416.11	46.28	294.04	169.16
2001-01-31	588.5	927.21	240.65	247.8	455.14	46.69	307.15	153.8
2001-02-28	573	802.59	262.12	192.25	460.71	46.8	325.88	159.07
2001-03-30	582	735.49	250.63	197.63	428.78	51.01	265.57	148.53
2001-04-30	627	802.59	249.63	217.34	417.12	49.72	297.79	151.05
...	...	...	...	...	...	...	...	...
2023-05-31	453.3	590.5	260.8	76.37	3144	143.3	471.3	228.1
2023-06-30	458.35	621.7	248.4	73.97	2999	151.05	482.6	227.3
2023-07-31	483	646.3	258	74.43	2888	184.85	452.3	233.4
2023-08-31	487.5	583.1	265.7	73.21	2830	222.3	423.2	218.7
2023-09-29	531.4	644.9	264.2	76.82	2926	220.9	427.3	222.5

274 rows × 18 columns

```
# Creating new pandas dataframe with the desired stocks

selected_stocks = ['BUNZL', 'EASYJET', 'ROLLS-ROYCE HOLDINGS', 'BAE SYSTEMS',
ss_df = df[selected_stocks]
ss_df
```

3	BUNZL	EASYJET	ROLLS-ROYCE HOLDINGS	BAE SYSTEMS	SEVERN TRENT
Date					
2000-12-29	416.11	294.04	46.28	382	781.28
2001-01-31	455.14	307.15	46.69	296	735.07
2001-02-28	460.71	325.88	46.8	294	744.53
2001-03-30	428.78	265.57	51.01	314	729.3
2001-04-30	417.12	297.79	49.72	331	724.57
...	...	...	...	...	...
2023-05-31	3144	471.3	143.3	928.2	2773
2023-06-30	2999	482.6	151.05	927	2566

```
# Slicing the dataframe into 2 time periods as required

ss_df18 = ss_df.loc['2000-12-29':'2018-09-30']
ss_df23 = ss_df.loc['2018-10-01':'2023-09-29']
```

```
ss_df18
```

3	BUNZL	EASYJET	ROLLS-ROYCE HOLDINGS	BAE SYSTEMS	SEVERN TRENT
Date					
2000-12-29	416.11	294.04	46.28	382	781.28
2001-01-31	455.14	307.15	46.69	296	735.07
2001-02-28	460.71	325.88	46.8	294	744.53
2001-03-30	428.78	265.57	51.01	314	729.3
2001-04-30	417.12	297.79	49.72	331	724.57
...	...	...	...	...	...
2018-05-31	2290	1438.83	277.49	639.4	1989
2018-06-29	2295	1408.52	332.55	646.8	1979.5

```
ss_df23
```

3	BUNZL	EASYJET	ROLLS-ROYCE HOLDINGS	BAE SYSTEMS	SEVERN TRENT
Date					
2018-10-31	2311	1010.3	283.89	525.8	1863
2018-11-30	2416	936.63	287.48	491.3	1830
2018-12-31	2369	930.31	280.85	459.2	1815.5
2019-01-31	2400	1064.18	299.25	511.8	1998.5
2019-02-28	2373	1033.45	323.14	466	2019
2019-03-29	2532	940.84	305.62	482.4	1976

2019-04-30	2309	979.57	311.87	494	2036
2019-05-31	2115	733.47	293.26	452.4	1990
2019-06-28	2077	802.51	286.57	495.4	2048
2019-07-31	2150	811.6	293.94	548.6	2015
2019-08-30	2013	811.1	262.98	545.6	2071
2019-09-30	2125	968.2	270.2	570	2165
2019-10-31	2008	1042.29	243.31	576.2	2255
2019-11-29	2123	1126.48	244.2	573.4	2243
2019-12-31	2065	1199.3	234.45	564.8	2515
2020-01-31	1965	1172.78	229.92	631.4	2579
2020-02-28	1870	926.52	212.76	608.4	2465
2020-03-31	1629.5	480.73	116.88	521.8	2280
2020-04-30	1727.5	507.84	113.25	508.4	2390
2020-05-29	1886.5	572.5	93.2	496.2	2441
2020-06-30	2166	572.5	97.94	483.4	2479
2020-07-31	2198	415.23	79.48	490.2	2450
2020-08-31	2422	533.1	82.74	519.6	2332
2020-09-30	2507	422.3	44.61	482	2439
2020-10-30	2400	425.84	71.32	397	2430

2020-11-30	2359	678.08	105.7	504	2389
2020-12-31	2443	698.79	111.25	488.8	2289
2021-01-29	2350	613.25	91.7	462.1	2312
2021-02-26	2237	829.79	107.7	483.9	2190
2021-03-31	2323	823.56	105.3	505	2306
2021-04-30	2327	872.22	104.62	506.2	2477
2021-05-31	2285	847.39	107.02	526.6	2453
2021-06-30	2389	753.34	98.92	522	2501
2021-07-30	2666	712.43	99.71	576	2800
2021-08-31	2636	670.16	114.86	568.4	2762
2021-09-30	2455	662.8	140.14	565.2	2604
2021-10-29	2702	623	131.84	551.8	2738
2021-11-30	2868	496.5	122.5	548.2	2886
2021-12-	2885	556	122.88	549.8	2947

```
# Calculating log returns
ss_df18 = ss_df18.astype('float')
ss_df23 = ss_df23.astype('float')
ret_df18 = np.log(ss_df18 / ss_df18.shift(1)).dropna()
ret_df23 = np.log(ss_df23 / ss_df23.shift(1)).dropna()
```

```
ret_df18
```

3	BUNZL	EASYJET	ROLLS-ROYCE HOLDINGS	BAE SYSTEMS	SEVERN TRENT
Date					
2001-01-31	0.089655	0.043620	0.008820	-0.255061	-0.060968
2001-02-28	0.012164	0.059193	0.002353	-0.006780	0.012787
2001-03-30	-0.071825	-0.204651	0.086138	0.065813	-0.020668
2001-04-30	-0.027570	0.114510	-0.025614	0.052725	-0.006507
2001-05-31	0.091659	0.093602	0.108894	0.055815	0.032800
...	...	...	...	...	...
2018-05-31	0.081390	0.072176	-0.018531	0.045433	0.024944
2018-06-29	0.002181	-0.021291	0.181005	0.011507	-0.004788

```
ret_df23
```

3	BUNZL	EASYJET	ROLLS-ROYCE HOLDINGS	BAE SYSTEMS	SEVERN TRENT
Date					
2018-11-30	0.044433	-0.075714	0.012566	-0.067866	-0.017872
2018-12-31	-0.019645	-0.006770	-0.023333	-0.067569	-0.007955
2019-01-31	0.013001	0.134442	0.063459	0.108448	0.096036
2019-02-28	-0.011314	-0.029302	0.076806	-0.093748	0.010205
2019-03-29	0.064855	-0.093885	-0.055743	0.034588	-0.021528
2019-04-30	-0.092195	0.040341	0.020244	0.023762	0.029912

2019-05-31	-0.087760	-0.289327	-0.061527	-0.087969	-0.022852
2019-06-28	-0.018130	0.089958	-0.023077	0.090799	0.028729
2019-07-31	0.034543	0.011263	0.025393	0.102004	-0.016245
2019-08-30	-0.065842	-0.000616	-0.111298	-0.005483	0.027412
2019-09-30	0.054146	0.177047	0.027084	0.043750	0.044389
2019-10-31	-0.056633	0.073737	-0.104826	0.010818	0.040730
2019-11-29	0.055691	0.077678	0.003651	-0.004871	-0.005336
2019-12-31	-0.027700	0.062640	-0.040745	-0.015112	0.114459
2020-01-31	-0.049638	-0.022361	-0.019511	0.111468	0.025129
2020-02-28	-0.049554	-0.235697	-0.077567	-0.037107	-0.045210
2020-03-31	-0.137665	-0.656130	-0.599017	-0.153548	-0.078016
2020-04-30	0.058402	0.054861	-0.031550	-0.026016	0.047118
2020-05-29	0.088048	0.119846	-0.194850	-0.024289	0.021114
2020-06-30	0.138159	0.000000	0.049607	-0.026135	0.015447
2020-07-31	0.014666	-0.321180	-0.208850	0.013969	-0.011767
2020-08-31	0.097046	0.249876	0.040198	0.058246	-0.049362
2020-09-30	0.034493	-0.232993	-0.617745	-0.075115	0.044862
2020-10-30	-0.043618	0.008348	0.469219	-0.194008	-0.003697
2020-11-30	-0.017231	0.465202	0.393428	0.238640	-0.017016

2020-12-31	0.034989	0.030085	0.051175	-0.030623	-0.042760
2021-01-29	-0.038811	-0.130578	-0.193258	-0.056172	0.009998
2021-02-26	-0.049280	0.302400	0.160827	0.046097	-0.054211
2021-03-31	0.037724	-0.007536	-0.022536	0.042680	0.051613
2021-04-30	0.001720	0.057405	-0.006479	0.002373	0.071534
2021-05-31	-0.018214	-0.028881	0.022681	0.039509	-0.009736
2021-06-30	0.044509	-0.117644	-0.078704	-0.008774	0.019379
2021-07-30	0.109704	-0.055835	0.007955	0.098440	0.112929
2021-08-31	-0.011317	-0.061165	0.141448	-0.013282	-0.013664
2021-09-30	-0.071136	-0.011043	0.198928	-0.005646	-0.058906
2021-10-29	0.095865	-0.061927	-0.061053	-0.023994	0.050179
2021-11-30	0.059623	-0.226963	-0.073478	-0.006545	0.052644
2021-12-31	0.005910	0.113185	0.003097	0.002914	0.020916

```
# Creating a new dataframe for the risk free asset

rf_df = df['UK STERLING 1M DEPOSIT (FT/RFV) - MIDDLE RATE']
rf_df
```

```
Date
2000-12-29    5.8906
2001-01-31    5.7813
2001-02-28    5.6719
2001-03-30    5.5781
2001-04-30    5.3906
                ...
2023-05-31      4.71
2023-06-30     5.115
2023-07-31     5.345
2023-08-31     5.265
2023-09-29      5.33
Name: UK STERLING 1M DEPOSIT (FT/RFV) - MIDDLE RATE, Length: 274, dtype:
object
```

```
# Slicing the risk free asset dataframe for 2 different time periods

rf_df18 = rf_df.loc['2000-12-29':'2018-09-30']
rf_df23 = rf_df.loc['2018-09-30':'2023-09-29']
```

```
rf_df18
```

```
Date
2000-12-29    5.8906
2001-01-31    5.7813
2001-02-28    5.6719
2001-03-30    5.5781
2001-04-30    5.3906
                ...
2018-05-31      0.52
2018-06-29      0.52
2018-07-31      0.68
2018-08-31      0.73
2018-09-28      0.77
Name: UK STERLING 1M DEPOSIT (FT/RFV) - MIDDLE RATE, Length: 214, dtype:
object
```

```
rf_df23
```

```
Date
2018-10-31     0.87
2018-11-30    1.055
2018-12-31    0.915
2019-01-31     0.77
2019-02-28     0.77
```

```
2019-02-28     0.77
2019-03-29     0.77
2019-04-30     0.77
2019-05-31     0.75
2019-06-28     0.75
2019-07-31     0.75
2019-08-30     0.75
2019-09-30     0.77
2019-10-31     0.76
2019-11-29     0.73
2019-12-31     0.79
2020-01-31     0.75
2020-02-28     0.72
2020-03-31    0.325
2020-04-30    0.185
2020-05-29     0.23
2020-06-30    0.195
2020-07-31    0.095
2020-08-31     0.42
2020-09-30     0.09
2020-10-30     0.09
2020-11-30     0.09
2020-12-31     0.09
2021-01-29     0.09
2021-02-26     0.09
2021-03-31     0.09
2021-04-30     0.09
2021-05-31      0.1
2021-06-30      0.1
2021-07-30     0.09
2021-08-31      0.1
2021-09-30      0.1
2021-10-29     0.13
2021-11-30      0.1
2021-12-31     0.24
2022-01-31     0.42
2022-02-28    0.735
2022-03-31     0.75
2022-04-29     0.98
2022-05-31      1.1
2022-06-30    1.345
2022-07-29     1.65
2022-08-31     1.95
2022-09-30     2.52
2022-10-31     2.95
2022-11-30     3.15
2022-12-30    3.435
2023-01-31     3.98
2023-02-28     4.19
2023-03-31    4.435
2023-04-28     4.46
2023-05-31     4.71
2023-06-30    5.115
2023-07-31    5.345
2023-08-31    5.365
```

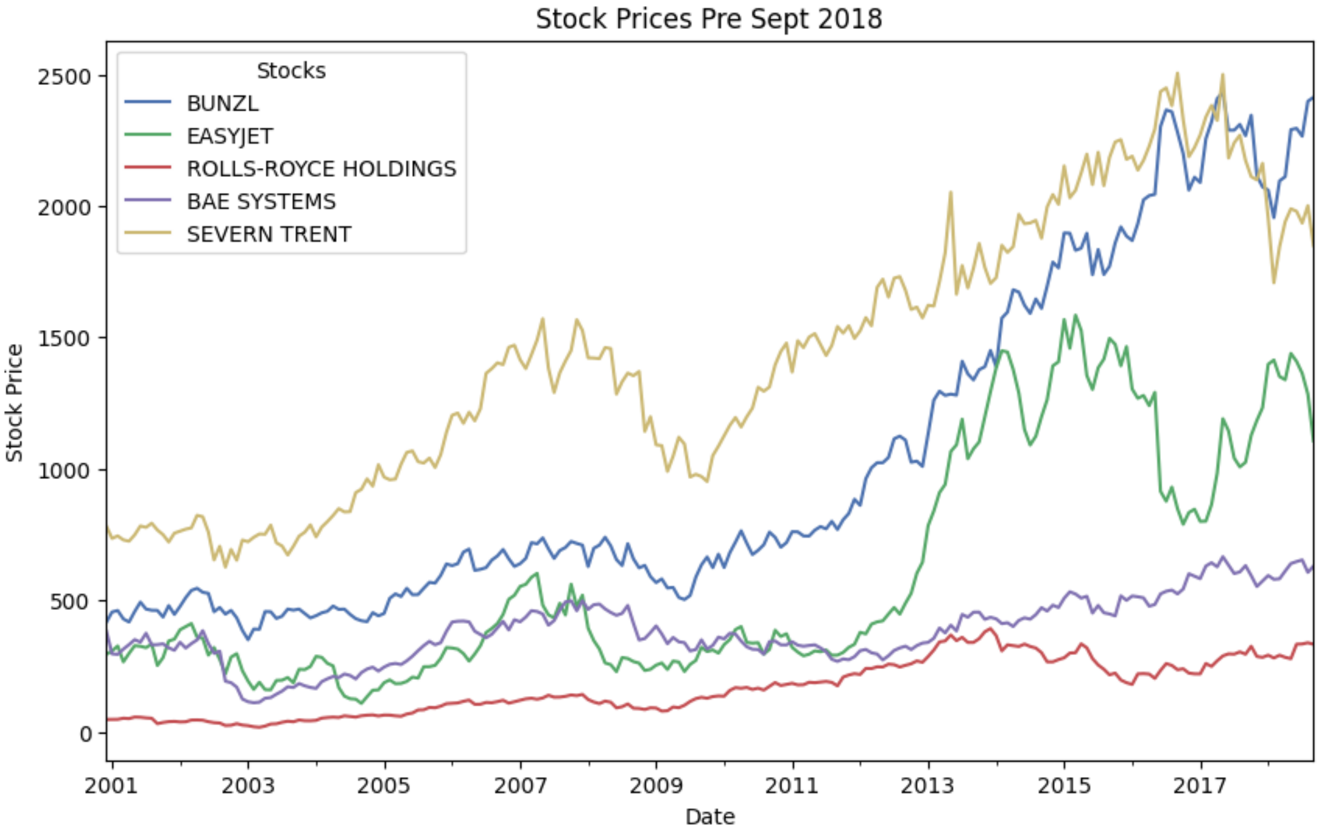
```
# Computing the risk free rate for the respective time periods

risk_free_rate_df18 = rf_df18.mean() / 100
print(f"Pre Sept 2018 Risk Free Rate:  {risk_free_rate_df18}")
risk_free_rate_df23 = rf_df23.mean() / 100
print(f"Post Sept 2018 Risk Free Rate: {risk_free_rate_df23}")
```

```
    Pre Sept 2018 Risk Free Rate:  0.024231537383117756
    Post Sept 2018 Risk Free Rate: 0.013397499999999998
```

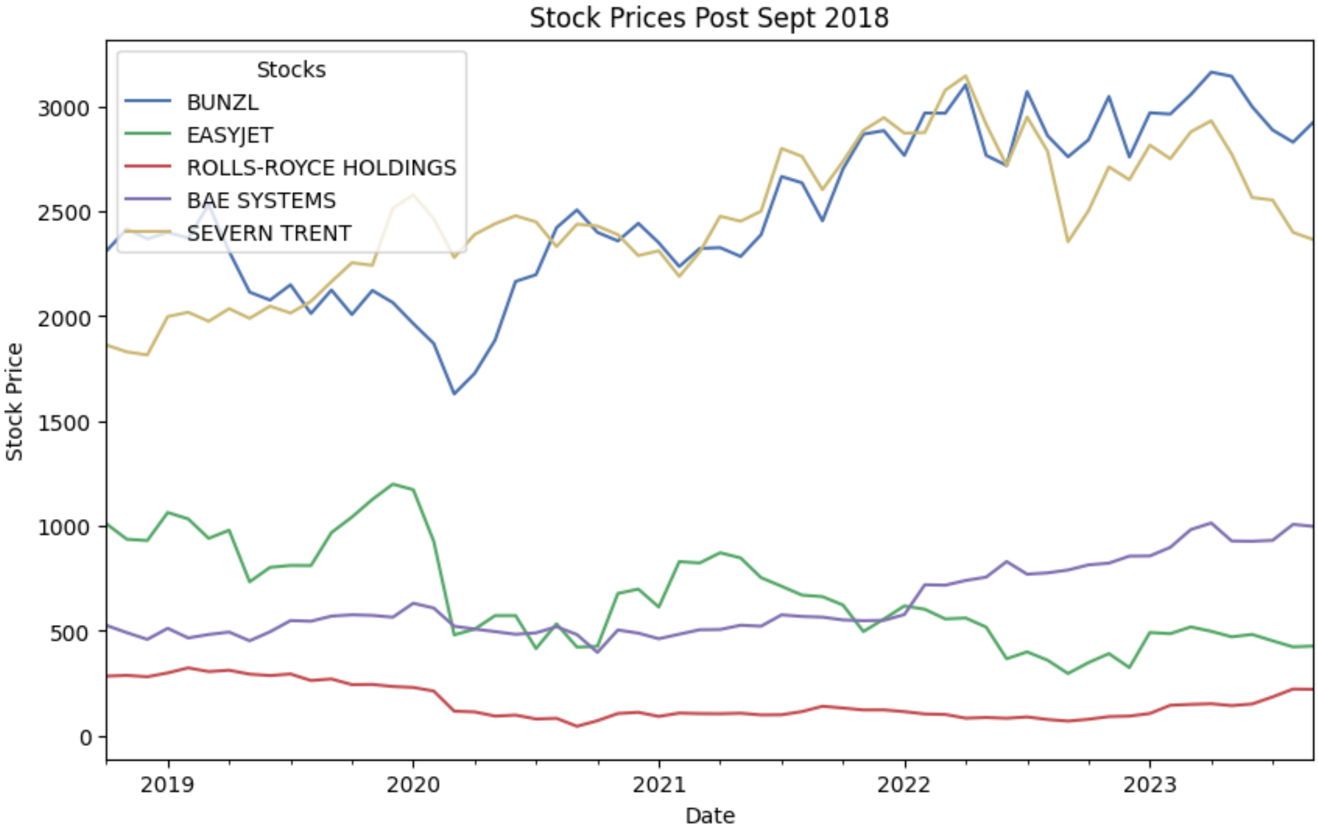
Data Analysis

Plotting prices, returns and computing descriptive and financial statistics like Sharpe ratio

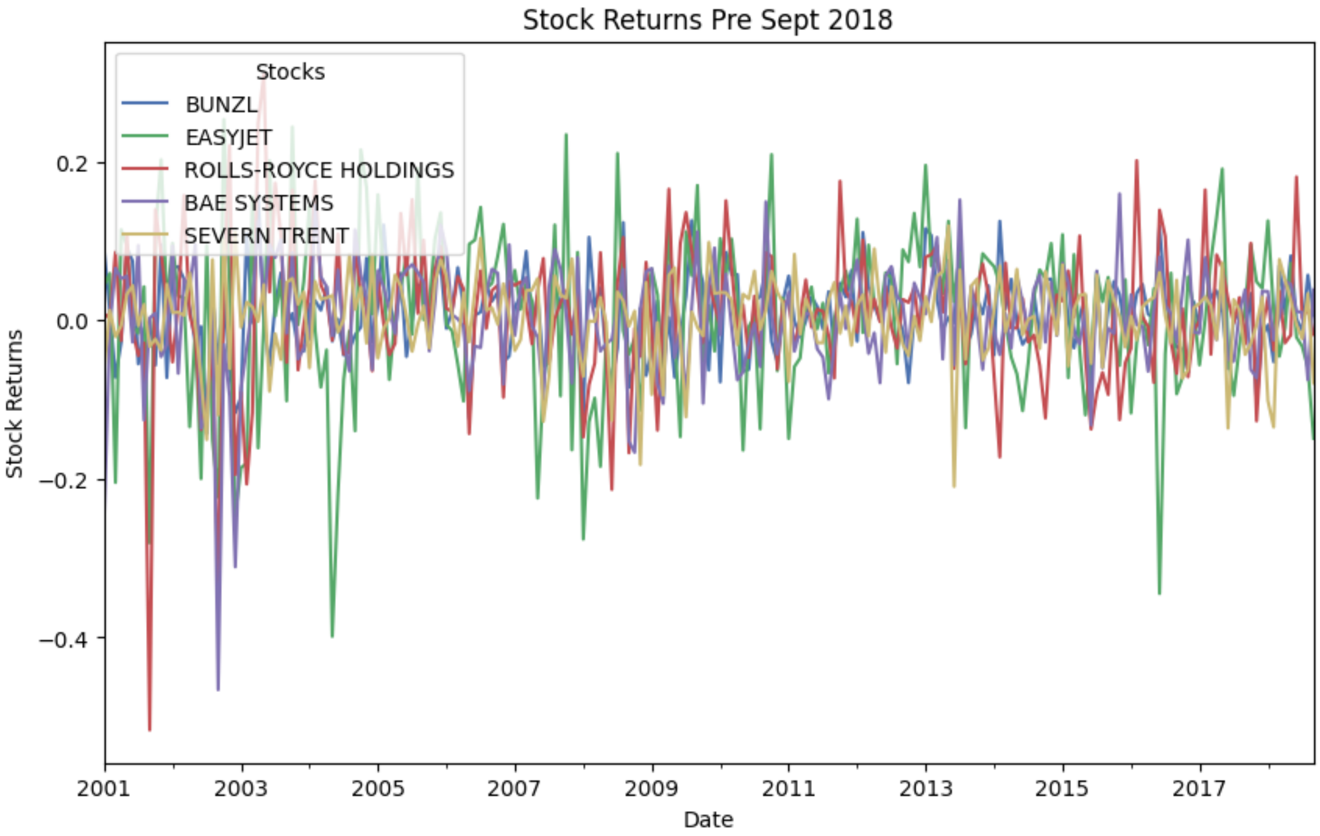
```
ss_df18.plot(figsize=(10, 6))
plt.title('Stock Prices Pre Sept 2018')
plt.xlabel('Date')
plt.ylabel('Stock Price')
plt.legend(title='Stocks', loc='upper left')
plt.show()
```



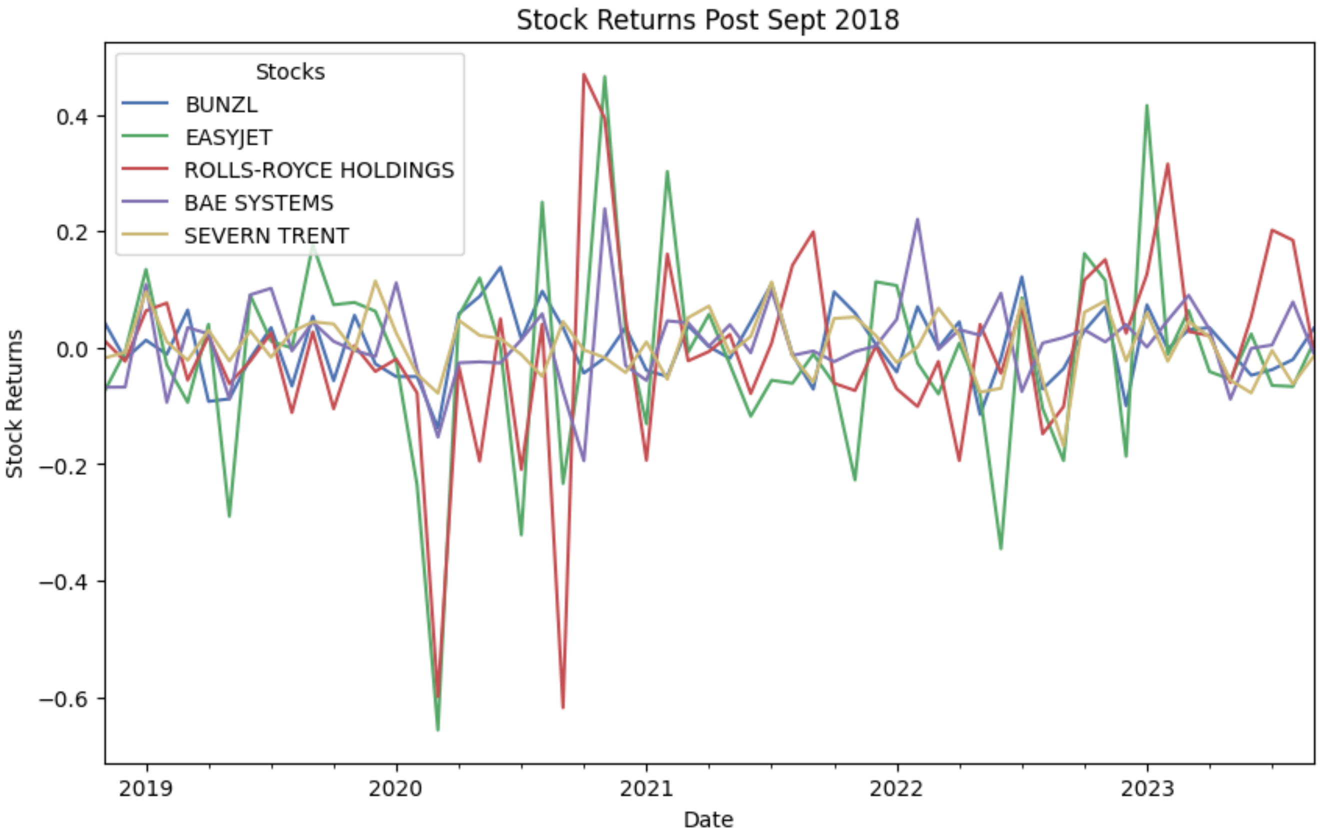
```
ss_df23.plot(figsize=(10, 6))
plt.title('Stock Prices Post Sept 2018')
plt.xlabel('Date')
plt.ylabel('Stock Price')
plt.legend(title='Stocks', loc='upper left')
plt.show()
```



```
ret_df18.plot(figsize=(10, 6))
plt.title('Stock Returns Pre Sept 2018')
plt.xlabel('Date')
plt.ylabel('Stock Returns')
plt.legend(title='Stocks', loc='upper left')
plt.show()
```



```
ret_df23.plot(figsize=(10, 6))
plt.title('Stock Returns Post Sept 2018')
plt.xlabel('Date')
plt.ylabel('Stock Returns')
plt.legend(title='Stocks', loc='upper left')
plt.show()
```



```
mu_18
```

```
    3
    BUNZL                    0.085443
    EASYJET                  0.001240
    ROLLS-ROYCE HOLDINGS     0.057635
    BAE SYSTEMS             -0.008523
    SEVERN TRENT             0.031636
    dtype: float64
```

```
mu_23
```

```
3
BUNZL                  0.026349
EASYJET               -0.327809
ROLLS-ROYCE HOLDINGS  -0.241108
BAE SYSTEMS            0.103707
SEVERN TRENT           0.031883
dtype: float64
```

```
std_18
```

3	BUNZL	EASYJET	ROLLS-ROYCE HOLDINGS	BAE SYSTEMS	SEVERN TRENT
3					
BUNZL	0.033829	0.009113	0.018960	0.016267	0.007781
EASYJET	0.009113	0.139258	0.031635	0.023110	0.005598
ROLLS-ROYCE HOLDINGS	0.018960	0.031635	0.100677	0.034641	0.007021
BAE SYSTEMS	0.016267	0.023110	0.034641	0.065650	0.008961

```
std_23
```

3	BUNZL	EASYJET	ROLLS-ROYCE HOLDINGS	BAE SYSTEMS	SEVERN TRENT
3					
BUNZL	0.044645	0.042953	0.006768	0.007526	0.018325
EASYJET	0.042953	0.365992	0.210224	0.059527	0.036442
ROLLS-ROYCE HOLDINGS	0.006768	0.210224	0.356531	0.035794	0.001529
BAE SYSTEMS	0.007526	0.059527	0.035794	0.064516	0.002959

```
np.sqrt(std_18)
```

3	BUNZL	EASYJET	ROLLS-ROYCE HOLDINGS	BAE SYSTEMS	SEVERN TRENT
3					
BUNZL	0.183926	0.095461	0.137697	0.127542	0.088211
EASYJET	0.095461	0.373173	0.177861	0.152019	0.074822
ROLLS-ROYCE HOLDINGS	0.137697	0.177861	0.317297	0.186122	0.083793
BAE SYSTEMS	0.127542	0.152019	0.186122	0.256223	0.094665

```
np.sqrt(std_23)
```

3	BUNZL	EASYJET	ROLLS-ROYCE HOLDINGS	BAE SYSTEMS	SEVERN TRENT
3					
BUNZL	0.211294	0.207251	0.082269	0.086755	0.135368
EASYJET	0.207251	0.604973	0.458502	0.243982	0.190897
ROLLS-ROYCE HOLDINGS	0.082269	0.458502	0.597102	0.189193	0.039106
BAE SYSTEMS	0.086755	0.243982	0.189193	0.254000	0.054400

```
# Computing Sharpe Ratios
bunzl_sr_18 = (0.085443 - 0.02423153738317756) / np.sqrt(0.033829)
easyjet_sr_18 = (0.001240 - 0.02423153738317756) / np.sqrt(0.139258)
rr_sr_18 = (0.057635 - 0.02423153738317756) / np.sqrt(0.100677)
bae_sr_18 = (-0.008523 - 0.02423153738317756) / np.sqrt(0.065650)
st_sr_18 = (0.031636 - 0.02423153738317756) / np.sqrt(0.033689)
print(f"BUNZL Sharpe Ratio Pre Sept 2018: {bunzl_sr_18}")
print(f"EASYJET Sharpe Ratio Pre Sept 2018: {easyjet_sr_18}")
print(f"ROLLS-ROYCE HOLDINGS Sharpe Ratio Pre Sept 2018: {rr_sr_18}")
print(f"BAE SYSTEMS Sharpe Ratio Pre Sept 2018: {bae_sr_18}")
print(f"SEVERN TRENT Sharpe Ratio Pre Sept 2018: {st_sr_18}")
```

```
    BUNZL Sharpe Ratio Pre Sept 2018: 0.3328037236495355
    EASYJET Sharpe Ratio Pre Sept 2018: -0.06161095428053072
    ROLLS-ROYCE HOLDINGS Sharpe Ratio Pre Sept 2018: 0.10527526792436634
    BAE SYSTEMS Sharpe Ratio Pre Sept 2018: -0.12783627423861588
    SEVERN TRENT Sharpe Ratio Pre Sept 2018: 0.04034126243775734
```

```
bunzl_sr_23 = (0.026349 - 0.013397499999999998) / np.sqrt(0.044645)
easyjet_sr_23 = ( -0.327809 - 0.013397499999999998) / np.sqrt(0.365992)
rr_sr_23 = (-0.241108 - 0.013397499999999998) / np.sqrt(0.356531)
bae_sr_23 = (0.103707 - 0.013397499999999998) / np.sqrt(0.064516)
st_sr_23 = (0.031883 - 0.013397499999999998) / np.sqrt(0.034379)
print(f"BUNZL Sharpe Ratio Post Sept 2018: {bunzl_sr_23}")
print(f"EASYJET Sharpe Ratio Post Sept 2018: {easyjet_sr_23}")
print(f"ROLLS-ROYCE HOLDINGS Sharpe Ratio Post Sept 2018: {rr_sr_23}")
print(f"BAE SYSTEMS Sharpe Ratio Post Sept 2018: {bae_sr_23}")
print(f"SEVERN TRENT Sharpe Ratio Post Sept 2018: {st_sr_23}")
```

```
    BUNZL Sharpe Ratio Post Sept 2018: 0.061296214757150476
    EASYJET Sharpe Ratio Post Sept 2018: -0.5640031111566062
    ROLLS-ROYCE HOLDINGS Sharpe Ratio Post Sept 2018: -0.42623442568692127
    BAE SYSTEMS Sharpe Ratio Post Sept 2018: 0.3555492125984252
    SEVERN TRENT Sharpe Ratio Post Sept 2018: 0.09969757204323881
```

▾ Pre Sept 2018 Optimal Portfolio

```
mu_18 = expected_returns.mean_historical_return(ss_df18,frequency = 12,log_ret
std_18 = risk_models.sample_cov(ss_df18,frequency = 12,log_returns=True)
```

```
ef_18 = EfficientFrontier(mu_18, std_18)
```

```
opt18 = ef_18.max_sharpe(risk_free_rate_df18)
opt18
```

```
    OrderedDict([('BUNZL', 0.9999988878818848),
                 ('EASYJET', 1.9442662763e-06),
                 ('ROLLS-ROYCE HOLDINGS', -1.7639282742e-06),
                 ('BAE SYSTEMS', 1.7684984203e-06),
                 ('SEVERN TRENT', -8.356937556e-07)])
```

```
ef_18.portfolio_performance(verbose=True,risk_free_rate = risk_free_rate_df18)
```

```
    Expected annual return: 8.5%
    Annual volatility: 18.4%
    Sharpe Ratio: 0.33
    (0.0854432340286779, 0.18392601363220387, 0.33280608564650965)
```

Pre Sept 18 Minimum Variance Portfolio

```
ef_min_18 = EfficientFrontier(mu_18, std_18)
```

```
min18 = ef_min_18.min_volatility()
min18
```

```
    OrderedDict([('BUNZL', 0.3758070645351279),
                 ('EASYJET', 0.0705404107068954),
                 ('ROLLS-ROYCE HOLDINGS', 0.0319494601717527),
                 ('BAE SYSTEMS', 0.0932683193190816),
                 ('SEVERN TRENT', 0.4284347452671422)])
```

```
ef_min_18.portfolio_performance(verbose=True, risk_free_rate = risk_free_rate_
```

```
    Expected annual return: 4.7%
    Annual volatility: 13.7%
    Sharpe Ratio: 0.16
    (0.046797950580979145, 0.13715915390767794, 0.16452721203712786)
```

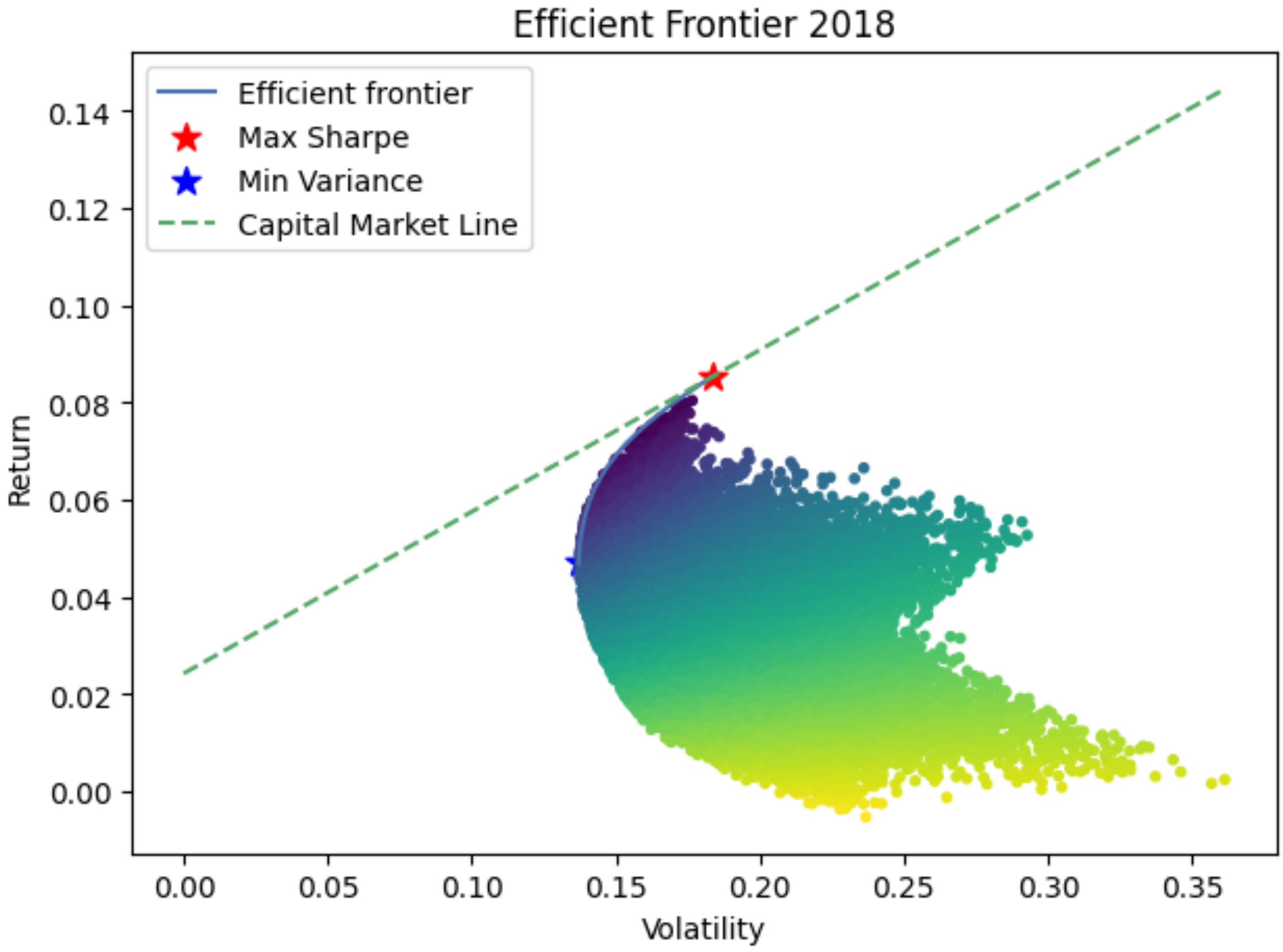
Plotting the Pre 2018 Efficient Frontier, CML with the Minimum Variance Portfolio & the Optimal Portfolio

```
port_18 = EfficientFrontier(mu_18,std_18)
```

```
# Setting up the plots & portfolio
```

```
fig, ax = plt.subplots()
ef_max_sharpe = port_18.deepcopy()
plotting.plot_efficient_frontier(port_18, ax=ax, show_assets=False)

# Find the tangency portfolio
ef_max_sharpe.max_sharpe(risk_free_rate_df18)
ret_tangent, std_tangent, _ = ef_max_sharpe.portfolio_performance()
ax.scatter(std_tangent, ret_tangent, marker="*", s=100, c="r", label="Max Shar
ax.scatter(0.137,0.047,marker = "*",s=100,c = "b",label = "Min Variance")

# Generate random portfolios
n_samples = 25000
w = np.random.dirichlet(np.ones(port_18.n_assets), n_samples)
rets = w.dot(port_18.expected_returns)
stds = np.sqrt(np.diag(w @ port_18.cov_matrix @ w.T))
sharpes = rets / stds
ax.scatter(stds, rets, marker=".", c=sharpes, cmap="viridis_r")

# Plot the tangent line
slope_tangent = ((ret_tangent-risk_free_rate_df18)/std_tangent)
intercept_tangent = risk_free_rate_df18
x_tangent = np.linspace(0,max(stds),100)
y_tangent = slope_tangent*x_tangent+intercept_tangent
ax.plot(x_tangent,y_tangent,linestyle="--", label = "Capital Market Line")

# Output
ax.set_title("Efficient Frontier 2018")
ax.legend()
plt.tight_layout()
plt.savefig("ef_scatter_1.png", dpi=200)
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/pypfopt/efficient_frontier/efficier
  warnings.warn(
```



Post Sept 18 Optimal Portfolio

```
mu_23 = expected_returns.mean_historical_return(ss_df23,frequency = 12,log_ret
std_23 = risk_models.sample_cov(ss_df23,frequency = 12,log_returns=True)


ef_23 = EfficientFrontier(mu_23, std_23)


opt23 = ef_23.max_sharpe(risk_free_rate_df23)
opt23
```

```
OrderedDict([('BUNZL', 0.0),
             ('EASYJET', 0.0),
             ('ROLLS-ROYCE HOLDINGS', 0.0),
             ('BAE SYSTEMS', 0.7672254160392292),
             ('SEVERN TRENT', 0.2327745839607708)])
```

```
ef_23.portfolio_performance(verbose=True,risk_free_rate = risk_free_rate_df23)

    Expected annual return: 8.7%
    Annual volatility: 20.2%
    Sharpe Ratio: 0.36
    (0.0869881004664422, 0.20222833389412748, 0.363898564802344)
```

Post Sept 18 Minimum Variance Portfolio

```
ef_min_23 = EfficientFrontier(mu_23, std_23)
```

```
min23 = ef_min_23.min_volatility()
min23

    OrderedDict([('BUNZL', 0.2356326216686248),
                 ('EASYJET', 0.0),
                 ('ROLLS-ROYCE HOLDINGS', 0.0265584171409459),
                 ('BAE SYSTEMS', 0.2669082774731772),
                 ('SEVERN TRENT', 0.4709006837172521)])
```

```
ef_min_23.portfolio_performance(verbose=True, risk_free_rate = risk_free_rate_

    Expected annual return: 4.2%
    Annual volatility: 14.6%
    Sharpe Ratio: 0.20
    (0.04249937179276714, 0.14607388592712758, 0.19922706654963157)
```

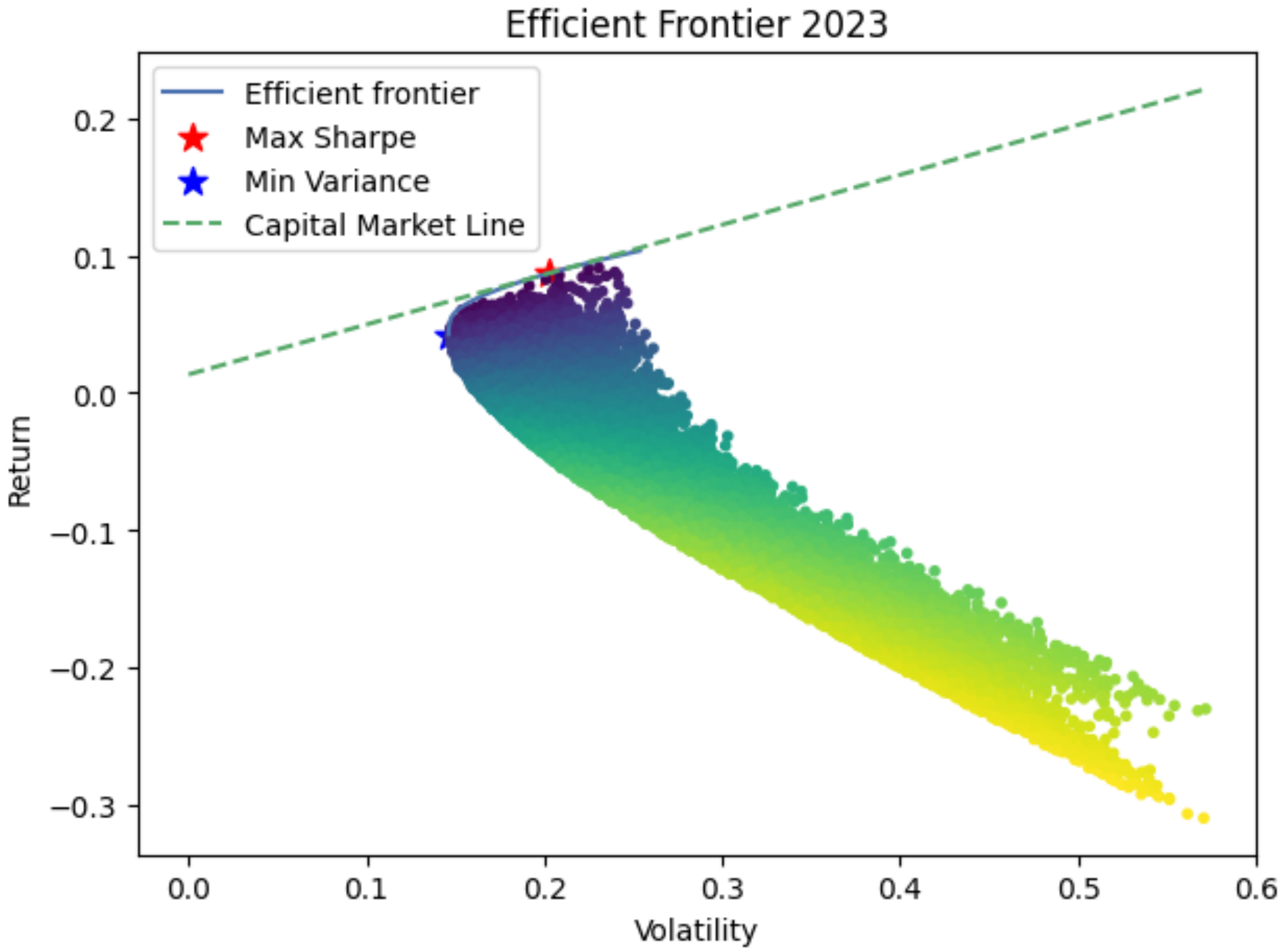
Plotting the Post 2018 Efficient Frontier, CML with the Minimum Variance Portfolio & the Optimal Portfolio

```
port_23 = EfficientFrontier(mu_23,std_23)
```

```
fig, ax = plt.subplots()
ef_max_sharpe = port_23.deepcopy()
plotting.plot_efficient_frontier(port_23, ax=ax, show_assets=False)

# Find the tangency portfolio
ef_max_sharpe.max_sharpe(risk_free_rate_df23)
ret_tangent, std_tangent, _ = ef_max_sharpe.portfolio_performance()
ax.scatter(std_tangent, ret_tangent, marker="*", s=100, c="r", label="Max Shar
ax.scatter(0.146,0.042,marker = "*",s=100,c = "b",label = "Min Variance")
```

```
# Generate random portfolios
n_samples = 25000
w = np.random.dirichlet(np.ones(port_23.n_assets), n_samples)
rets = w.dot(port_23.expected_returns)
stds = np.sqrt(np.diag(w @ port_23.cov_matrix @ w.T))
sharpes = rets / stds
ax.scatter(stds, rets, marker=".", c=sharpes, cmap="viridis_r")

# Plot the tangent line
slope_tangent = ((ret_tangent-risk_free_rate_df23)/std_tangent)
intercept_tangent = risk_free_rate_df23
x_tangent = np.linspace(0,max(stds),100)
y_tangent = slope_tangent*x_tangent+intercept_tangent
ax.plot(x_tangent,y_tangent,linestyle="--", label = "Capital Market Line")

# Output
ax.set_title("Efficient Frontier 2023")
ax.legend()
plt.tight_layout()
plt.savefig("ef_scatter_2.png", dpi=200)
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/pypfopt/efficient_frontier/efficier
  warnings.warn(
```


Efficient Frontier 2023
Efficient frontier
Max Sharpe
Min Variance
Capital Market Line
Return
Volatility