Response for Project Week 7

Course Code: Fintech545

Course Title: Quantitative Risk Management

Student Name: Wanglin (Steve) Cai

Student Net ID: WC191

Problem 1

Problem 1

- Current Stock Price $165
- Strike Price $165
- Current Date 03/13/2022
- Options Expiration Date 04/15/2022
- Risk Free Rate of 4.25%
- Continuously Compounding Coupon of 0.53%

Implement the closed form greeks for GBSM. Implement a finite difference derivative calculation. Compare the values between the two methods for both a call and a put.

Implement the binomial tree valuation for American options with and without discrete dividends. Assume the stock above:

- Pays dividend on 4/11/2022 of $0.88

Calculate the value of the call and the put. Calculate the Greeks of each.

What is the sensitivity of the put and call to a change in the dividend amount?

1.1 Implementing closed form greeks for GBSM

In this question, I implemented the closed form greeks for GBSM by using functions. Then I defined other functions used to implement a finite difference derivative calculation. Then I ran these functions and compare the values:

```
# delta
delta_call = gbsm_delta("Call", S, X, T, implied_vol, r, b)
delta_put = gbsm_delta("Put", S, X, T, implied_vol, r, b)
gbsm_delta_num = cal_partial_derivative(gbsm, 1, 'S')
delta_call_num = gbsm_delta_num("Call", S, X, T, implied_vol, r, b)
delta_put_num = gbsm_delta_num("Put", S, X, T, implied_vol, r, b)
print(delta_call, delta_put)
print(delta_call_num, delta_put_num)
✓ 0.1s

0.5342650692693474 -0.46573493073065264
0.5340091223970944 -0.46551181431908617
```

```
# gamma
gamma_call = gbsm_gamma("Call", S, X, T, implied_vol, r, b)
gamma_put = gbsm_gamma("Put", S, X, T, implied_vol, r, b)
gbsm_gamma_num = cal_partial_derivative(gbsm, 2, 'S')
gamma_call_num = gbsm_gamma_num("Call", S, X, T, implied_vol, r, b)
gamma_put_num = gbsm_gamma_num("Put", S, X, T, implied_vol, r, b)
print(gamma_call, gamma_put)
print(gamma_call_num, gamma_put_num)
✓ 0.1s

0.04005712070020568 0.04005712070020568
0.040037932080849714 0.040037960502559145
```

```
# vega
vega_call = gbsm_vega("Call", S, X, T, implied_vol, r, b)
vega_put = gbsm_vega("Put", S, X, T, implied_vol, r, b)
gbsm_vega_num = cal_partial_derivative(gbsm, 1, 'implied_vol')
vega_call_num = gbsm_vega_num("Call", S, X, T, implied_vol, r, b)
vega_put_num = gbsm_vega_num("Put", S, X, T, implied_vol, r, b)
print(vega_call, vega_put)
print(vega_call_num, vega_put_num)
✓ 0.1s

19.71962666579851 19.71962666579851
19.71017887198201 19.71017887198201
```

```
# theta
theta_call = gbsm_theta("Call", S, X, T, implied_vol, r, b)
theta_put = gbsm_theta("Put", S, X, T, implied_vol, r, b)
gbsm_theta_num = cal_partial_derivative(gbsm, 1, 'T')
theta_call_num = -gbsm_theta_num("Call", S, X, T, implied_vol, r, b)
theta_put_num = -gbsm_theta_num("Put", S, X, T, implied_vol, r, b)
print(theta_call, theta_put)
print(theta_call_num, theta_put_num)
✓ 0.2s

-24.898522316969515 -18.786996965277233
-24.898857359268334 -18.78733200548055
```

```
# rho
rho_call = gbsm_rho("Call", S, X, T, implied_vol, r, b)
rho_put = gbsm_rho("Put", S, X, T, implied_vol, r, b)
gbsm_rho_num = cal_partial_derivative(gbsm, 1, 'r')
rho_call_num = gbsm_rho_num("Call", S, X, T, implied_vol, r, b)
rho_put_num = gbsm_rho_num("Put", S, X, T, implied_vol, r, b)
print(rho_call, rho_put)
print(rho_call_num, rho_put_num)
✓ 0.1s

7.583586080244792 -7.277010958127815
-0.3826595967950652 -0.3325949902830416
```

```
# carry rho
carry_rho_call = gbsm_carry_rho("Call", S, X, T, implied_vol, r, b)
carry_rho_put = gbsm_carry_rho("Put", S, X, T, implied_vol, r, b)
gbsm_carry_rho_num = cal_partial_derivative(gbsm, 1, 'b')
carry_rho_call_num = gbsm_carry_rho_num("Call", S, X, T, implied_vol, r, b)
carry_rho_put_num = gbsm_carry_rho_num("Put", S, X, T, implied_vol, r, b)
print(carry_rho_call, carry_rho_put)
print(carry_rho_call_num, carry_rho_put_num)
✓ 0.1s

7.966245676523029 -6.944415968299725
7.96624576389604 -6.944415901223522
```

As we can see from the above graph, the value for the greeks are very close to each other except Rho, which measuring how values of options change with respect to risk free rate. Difference between results of Rho is because the closed-form formulas for Rho are for GBSM when risk free rate equal to b, cost of carry. In this case, the risk free rate differs from b, so results using closed-form formulas are incorrect.

Then, I defined the functions of binomial tree (assuming N=300) which are used to calculate the values of American options with or without dividends. After applying the defined functions, I get the following results:

```
# Assume N is 300
N = 300
value_no_div_call = binomial_tree_no_div("Call", S, X, T, implied_vol, r, N)
value_no_div_put = binomial_tree_no_div("Put", S, X, T, implied_vol, r, N)
print("Binomial tree value without dividend for call: " + str(value_no_div_call))
print("Binomial tree value without dividend for put: " + str(value_no_div_put))

✓ 0.5s

Binomial tree value without dividend for call: 4.271506155124947
Binomial tree value without dividend for put: 3.685242343269967
```

```python
div_date = datetime(2022, 4, 11)
div = 0.88
div_time = int((div_date - current_date).days / (expire_date - current_date).days * N)

value_call = binomial_tree("Call", S, X, T, div_time, div, implied_vol, r, N)
value_put = binomial_tree("Put", S, X, T, div_time, div, implied_vol, r, N)
print("Binomial tree value with dividend for call: " + str(value_call))
print("Binomial tree value with dividend for put: " + str(value_put))
```

```
Binomial tree value with dividend for call: 4.1140138117912395
Binomial tree value with dividend for put: 4.107373800666709
```

```python
# delta
cal_amr_delta_num = cal_partial_derivative(binomial_tree, 1, 'S0')
delta_call_amr = cal_amr_delta_num("Call", S, X, T, div_time, div, implied_vol, r, N)
delta_put_amr = cal_amr_delta_num("Put", S, X, T, div_time, div, implied_vol, r, N)
print(delta_call_amr, delta_put_amr)
```

```
0.54049284378043779 -0.4888376347138568
```

```python
# gamma
cal_amr_gamma_num = cal_partial_derivative(binomial_tree, 2, 'S0', delta=1)
gamma_call_amr = cal_amr_gamma_num("Call", S, X, T, div_time, div, implied_vol, r, N)
gamma_put_amr = cal_amr_gamma_num("Put", S, X, T, div_time, div, implied_vol, r, N)
print(gamma_call_amr, gamma_put_amr)
```

```
0.04041911053184144 0.0395121232373139
```

```python
# vega
cal_amr_vega_num = cal_partial_derivative(binomial_tree, 1, 'implied_vol')
vega_call_amr = cal_amr_vega_num("Call", S, X, T, div_time, div, implied_vol, r, N)
vega_put_amr = cal_amr_vega_num("Put", S, X, T, div_time, div, implied_vol, r, N)
print(vega_call_amr, vega_put_amr)
```

```
19.531465248245006 19.827352392823183
```

```python
# theta
cal_amr_theta_num = cal_partial_derivative(binomial_tree, 1, 'T')
theta_call_amr = -cal_amr_theta_num("Call", S, X, T, div_time, div, implied_vol, r, N)
theta_put_amr = -cal_amr_theta_num("Put", S, X, T, div_time, div, implied_vol, r, N)
print(theta_call_amr, theta_put_amr)
```

```
-24.808805697263736 -18.542829430221897
```

```python
# rho
cal_amr_rho_num = cal_partial_derivative(binomial_tree, 1, 'r')
rho_call_amr = cal_amr_rho_num("Call", S, X, T, div_time, div, implied_vol, r, N)
rho_put_amr = cal_amr_rho_num("Put", S, X, T, div_time, div, implied_vol, r, N)
print(rho_call_amr, rho_put_amr)
```

```
6.819090304833075 -7.207250065234927
```

Note that Carry Rhos are not applicable to American options with dividends, since we are not using b, costs of carry, as a parameter while pricing these options. Therefore, partial derivatives do not exist.

Eventually, I am able to calculate sensitivity of both of those American options to cha nges in dividends with applications of finite difference derivative methods using central differences, and results are:

```
Sensitivity to dividend amount: Call: -0.104, Put: 0.508
```

We may conclude from these numbers that as dividends increase, values of American call options decrease, while values of American put options increase.

Problem 2

Problem 2

Using the options portfolios from Problem3 last week (named problem2.csv in this week's repo) and assuming :

- American Options
- Current Date 03/03/2023
- Current AAPL price is 165
- Risk Free Rate of 4.25%
- Dividend Payment of $1.00 on 3/15/2023

Using DailyPrices.csv. Fit a Normal distribution to AAPL returns – assume 0 mean return. Simulate AAPL returns 10 days ahead and apply those returns to the current AAPL price (above). Calculate Mean, VaR and ES.

Calculate VaR and ES using Delta-Normal.

Present all VaR and ES values a $ loss, not percentages.

Compare these results to last week's results.

2.1 Assume Normal Distribution, Calculate Mean, VaR, ES

First, we need to transform the price data to return data.

We can get the return for the price dataset:

```
all_returns
```

✓ 0.2s Python

	Date	SPY	AAPL	MSFT	AMZN	TSLA	GOOGL	GOOG	META	NVDA	...	PNC	MDLZ
1	2/15/2022 0:00	0.015998	0.022888	0.018372	0.008621	0.051919	0.007956	0.008284	0.015045	0.087839	...	0.012726	-0.004091
2	2/16/2022 0:00	0.001120	-0.001390	-0.001168	0.010108	0.001040	0.008234	0.007754	-0.020387	0.000604	...	0.006734	-0.002432
3	2/17/2022 0:00	-0.021593	-0.021499	-0.029719	-0.022050	-0.052286	-0.038476	-0.038397	-0.041632	-0.078601	...	-0.035574	0.005312
4	2/18/2022 0:00	-0.006496	-0.009400	-0.009678	-0.013351	-0.022351	-0.016247	-0.014012	-0.007490	-0.035934	...	-0.000646	-0.000909
5	2/22/2022 0:00	-0.010790	-0.017973	-0.000729	-0.015879	-0.042246	-0.004531	-0.008196	-0.019989	-0.010716	...	0.009450	0.007096
...	...	...	...	...	...	...	...	...	...	...	...	...	...
244	2/3/2023 0:00	-0.010686	0.024107	-0.023904	-0.088083	0.009042	-0.027858	-0.033458	-0.011937	-0.028454	...	-0.004705	-0.011315
245	2/6/2023 0:00	-0.006130	-0.018091	-0.006135	-0.011772	0.024849	-0.018105	-0.016772	-0.002523	-0.000521	...	-0.014556	0.003937
246	2/7/2023 0:00	0.012994	0.019062	0.041163	-0.000685	0.010471	0.045035	0.043220	0.029445	0.050124	...	-0.000368	-0.016610
247	2/8/2023 0:00	-0.010995	-0.017810	-0.003107	-0.020381	0.022508	-0.079942	-0.077331	-0.043681	0.001442	...	-0.008505	-0.004466
248	2/9/2023 0:00	-0.008707	-0.006936	-0.011728	-0.018257	0.029517	-0.044868	-0.046463	-0.030499	0.005927	...	-0.016727	-0.007747

Then, we can fit normal distribution to AAPL returns. With return simulated, we can calculate the price which is also a simulated value because of return.

The simulated price is shown below:

```
sim_values
✓ 0.0s                                                    Python
```

	0	1	2	3	4	5	6	7	8	9	...
Portfolio											
Call	3.696995	3.721341	8.086201	17.287494	3.883030	11.021917	4.405491	0.183748	10.295121	13.419215	...
CallSpread	2.923976	2.939718	5.342283	8.301017	3.044266	6.597863	3.382089	0.177535	6.335391	7.345453	...
CoveredCall	144.273758	144.322229	149.667953	153.636933	144.644146	151.588958	145.684345	127.585147	151.135266	152.528589	...
ProtectedPut	150.518726	150.552936	156.201637	166.739658	150.780140	159.688161	151.516136	145.948577	158.796534	162.448407	...
Put	7.103832	7.063735	2.875651	0.527067	6.798004	1.644711	5.942518	23.218907	1.884691	1.076594	...
PutSpread	4.419738	4.399541	2.028963	0.408892	4.265959	1.209720	3.836515	9.445319	1.348396	0.799188	...
Stock	146.074367	146.138250	154.624186	166.492574	146.562522	158.852543	147.933455	127.624182	157.810378	161.910833	...
Straddle	10.800827	10.785076	10.961852	17.814561	10.681034	12.666629	10.348009	23.402655	12.179812	14.495809	...
SynLong	-3.406837	-3.342395	5.210549	16.760427	-2.914974	9.377206	-1.537027	-23.035159	8.410430	12.342621	...

rows × 100 columns

As a result, the Mean, VaR, and ES can be calculated.

	Mean	VaR($)	ES($)
Call	6.672356	6.192209	6.562053
CallSpread	4.064176	4.022207	4.361915
CoveredCall	145.248033	13.841819	18.791813
ProtectedPut	154.082517	7.642252	8.040817
Put	6.802323	4.351499	4.649564
PutSpread	3.828268	2.624612	2.853856
Stock	149.520044	17.729656	22.788306
Straddle	13.474678	1.346927	1.385833
SynLong	-0.129967	18.995445	24.324425

2.2 Using Delta-Normal

After finishing the Monte-Carlo normal method. We can go on to use delta normal method to calculate VaR and ES. The delta for calculating the gradients used in Delta Normal method is shown below:

```
deltas
✓ 0.0s
[0.540068567444596,
 -0.4644847553265663,
 0.540068567444596,
 -0.4644847553265663,
 0.540068567444596,
 0.24144707643669605,
 -0.4644847553265663,
 -0.18554087454214496,
 1,
 0.540068567444596,
 -0.4644847553265663,
 1,
 0.3815521521051135,
 1,
 -0.31121246555421145]
```

The results of VaR and ES is shown below:

	Mean	VaR	ES
Portfolio			
Call	0	9.514631	11.931731
CallSpread	0	5.260949	6.597442
CoveredCall	0	10.895474	13.663364
ProtectedPut	0	12.13468	15.217378
Put	0	8.183037	10.261858
PutSpread	0	4.91428	6.162705
Stock	0	17.61745	22.092993
Straddle	0	1.331594	1.669873
SynLong	0	17.697668	22.193589

2.3 Comparison

With these results, a comparison could be done. The value of last week's result is shown below:

	Mean	VaR	ES
Straddle	1.326614	1.380597	1.387930
SynLong	-0.424802	16.414656	20.073559
CallSpread	-0.187994	3.910205	4.199130
PutSpread	0.359855	2.553987	2.748505
Stock	-0.223259	16.170964	19.807763
Call	0.450906	6.060526	6.382465
Put	0.875708	4.265698	4.520131
CoveredCall	-0.844624	12.346409	15.870225
ProtectedPut	0.495880	8.112426	8.718021

As seen above, values of American options, calculated in this week's problem, are typically higher than European options, derived from last week's assignment (there are exceptions puts that are highly likely to not be available for exercise are having lower values). High returns/values are usually associated with higher risks, explaining why VaRs and ES are typically higher for American options, note that there are exceptions.

Problem 3

Problem 3

Use the Fama French 3 factor return time series (F-F_Research_Data_Factors_daily.CSV) as well as the Carhart Momentum time series (F-F_Momentum_Factor_daily.CSV) to fit a 4 factor model to the following stocks.

AAPL	FB	UNH	MA
MSFT	NVDA	HD	PFE
AMZN	BRK-B	PG	XOM
TSLA	JPM	V	DIS
GOOGL	JNJ	BAC	CSCO

Fama stores values as percentages, you will need to divide by 100 (or multiply the stock returns by 100) to get like units.

Based on the past 10 years of factor returns, find the expected **annual** return of each stock.

Construct an annual covariance matrix for the 10 stocks.

Assume the risk free rate is 0.0425. Find the super efficient portfolio.

3.1 merge data

For this question, since we have data from different time, we need to merge them together and include the data with their time is overlapping.

```
dataset
```

Date	AAPL	META	UNH	MA	MSFT	NVDA	HD	PFE	AMZN	BRK-B	...	DIS	GOOG
2022-02-15	0.022888	0.015045	0.008041	0.019532	0.018372	0.087839	0.004824	-0.000201	0.008621	0.006091	...	0.025331	0.00795
2022-02-16	-0.001390	-0.020387	0.003798	0.003636	-0.001168	0.000604	-0.009014	-0.002212	0.010108	-0.001741	...	0.010480	0.00823
2022-02-17	-0.021499	-0.041632	-0.020434	-0.024399	-0.029719	-0.078601	-0.006160	-0.015825	-0.022050	-0.006675	...	-0.021986	-0.03847
2022-02-18	-0.009400	-0.007490	-0.005394	-0.010086	-0.009678	-0.035934	-0.003080	-0.007595	-0.013351	0.003979	...	-0.010450	-0.01624
2022-02-22	-0.017973	-0.019989	-0.011394	-0.004497	-0.000729	-0.010716	-0.092670	-0.020821	-0.015879	-0.002035	...	-0.021841	-0.00453
...	...	...	...	...	...	...	...	...	...	...	...	...	
2023-01-25	-0.004712	-0.011523	0.001829	0.006243	-0.005926	0.003006	-0.001354	0.008020	0.008889	0.001926	...	0.019803	-0.02571
2023-01-26	0.014695	0.040172	-0.000041	-0.013560	0.030251	0.024487	-0.010934	-0.009223	0.020775	-0.003050	...	0.014508	0.02386
2023-01-27	0.013591	0.029697	-0.013142	-0.008546	0.000645	0.028035	0.009136	-0.010450	0.029983	-0.005741	...	-0.001460	0.01879
2023-01-30	-0.020282	-0.031328	-0.000535	-0.007811	-0.022206	-0.060889	-0.007766	-0.005496	-0.016668	-0.005969	...	-0.017962	-0.02475
2023-01-31	0.008980	0.012904	0.027210	-0.001402	0.020795	0.019381	0.031111	0.013910	0.025335	0.013541	...	0.008330	0.01941

Since the datafile F-F_Research_data_factors_daily and F-F Momentum_Factor_daily only has data until 1/31/2023. We need to exclude the data since February 2023.

3.2 expected annual return and covariance matrix

After doing the merging, I filtered and processed these data so that they could be applicable later to fit the model as I would like to preparing all factor data as Xs and stock returns risk free rate s) as Ys that would be used in regression. Note that I multiply returns by 100 so that they are in the format of percentage. Then, I use OLS function to run linear regression over my Xs and Ys. With estimated betas, I am able to calculate daily expected returns of each stock based on average factor returns of the past 10 years, and annualize them by multiplying daily returns by the number of trading days, which is 255; therefore, resulting returns are note that returns are expressed in percentages here)

Here, I use both geometric mean and arithmetic mean for the results.

```
# geometric annu
geo_means = np.l
geo_covariance =
print(geo_means)
```

✓ 0.1s

```
AAPL     0.094815
META     0.364139
UNH     -0.067104
MA       0.196962
MSFT     0.112007
NVDA     0.477393
HD       0.202864
PFE     -0.028339
AMZN     0.160461
BRK-B    0.111659
PG       0.115485
XOM      0.050076
TSLA    -0.115382
JPM      0.304899
V        0.194757
DIS      0.304078
GOOGL    0.153048
JNJ     -0.053340
BAC      0.270332
CSCO     0.197534
dtype: float64
```

```
display(geo_covariance)
```

✓ 0.1s Python

	AAPL	META	UNH	MA	MSFT	NVDA	HD	PFE	AMZN	BRK-B	PG	XOM	TSLA
AAPL	0.128420	0.141767	0.038210	0.082102	0.103863	0.172784	0.067124	0.033264	0.124313	0.055957	0.037377	0.038211	0.159622
META	0.141767	0.432568	0.019368	0.102816	0.142734	0.237412	0.096745	0.046574	0.197253	0.062079	0.033342	0.022000	0.174324
UNH	0.038210	0.019368	0.061663	0.031739	0.037093	0.047934	0.026776	0.032249	0.036749	0.028494	0.028296	0.027016	0.040789
MA	0.082102	0.102816	0.031739	0.097118	0.080303	0.138967	0.057322	0.034195	0.098782	0.048143	0.031232	0.031746	0.100198
MSFT	0.103863	0.142734	0.037093	0.080303	0.129307	0.177341	0.071646	0.035847	0.136562	0.053091	0.034098	0.031595	0.133427
NVDA	0.172784	0.237412	0.047934	0.138967	0.177341	0.407973	0.112821	0.047719	0.225900	0.085444	0.041971	0.057013	0.299003
HD	0.067124	0.096745	0.026776	0.057322	0.071646	0.112821	0.100060	0.034232	0.098874	0.042729	0.035081	0.015990	0.079521
PFE	0.033264	0.046574	0.032249	0.034195	0.035847	0.047719	0.034232	0.071127	0.039182	0.031562	0.027795	0.020266	0.023446
AMZN	0.124313	0.197253	0.036749	0.098782	0.136562	0.225900	0.098874	0.039182	0.251933	0.067084	0.031346	0.038389	0.190833
BRK-B	0.055957	0.062079	0.028494	0.048143	0.053091	0.085444	0.042729	0.031562	0.067084	0.050435	0.025046	0.034178	0.063062
PG	0.037377	0.033342	0.028296	0.031232	0.034098	0.041971	0.035081	0.027795	0.031346	0.025046	0.049875	0.004500	0.024236
XOM	0.038211	0.022000	0.027016	0.031746	0.031595	0.057013	0.015990	0.020266	0.038389	0.034178	0.004500	0.123199	0.045261
TSLA	0.159622	0.174324	0.040789	0.100198	0.133427	0.299003	0.079521	0.023446	0.190833	0.063062	0.024236	0.045261	0.479860
JPM	0.058778	0.074273	0.033466	0.058897	0.056915	0.099061	0.044064	0.032162	0.072108	0.047173	0.029681	0.029328	0.068497
V	0.072251	0.085601	0.030224	0.083473	0.068638	0.119867	0.050735	0.031222	0.085893	0.042603	0.029603	0.024802	0.093329
DIS	0.089545	0.127324	0.023519	0.079138	0.089340	0.161030	0.065139	0.025470	0.127926	0.052099	0.028082	0.036918	0.133000
GOOGL	0.112908	0.182912	0.030250	0.079717	0.121516	0.188639	0.070063	0.029762	0.152397	0.056737	0.028229	0.031503	0.144687
JNJ	0.023170	0.021635	0.023078	0.017602	0.020631	0.022461	0.022924	0.027893	0.023550	0.019375	0.023772	0.006469	0.013102
BAC	0.066545	0.089046	0.034686	0.064276	0.065558	0.114942	0.047222	0.031251	0.086893	0.050972	0.028716	0.033007	0.084756
CSCO	0.067601	0.076115	0.028918	0.052413	0.061536	0.098911	0.049247	0.029735	0.073439	0.040962	0.035745	0.023250	0.068103

```
print(arith_means)
```

✓ 0.1s

```
AAPL     0.172367
META     0.786795
UNH     -0.035623
MA       0.278287
MSFT     0.193225
NVDA     0.976604
HD       0.287746
PFE      0.007251
AMZN     0.331662
BRK-B    0.146686
PG       0.150760
XOM      0.118150
TSLA     0.132637
JPM      0.414015
V        0.267066
DIS      0.456722
GOOGL    0.265048
JNJ     -0.036789
BAC      0.378313
CSCO     0.276087
dtype: float64
```

```
display(arith_covariance)
```

✓ 0.1s Python

	AAPL	META	UNH	MA	MSFT	NVDA	HD	PFE	AMZN	BRK-B	PG	XOM	TSLA
AAPL	0.188340	0.319052	0.044037	0.128232	0.153107	0.437065	0.104816	0.039941	0.206655	0.077369	0.051380	0.051059	0.229810
META	0.319052	1.727888	0.033699	0.247333	0.327105	0.946391	0.233729	0.085805	0.518836	0.131225	0.069712	0.044441	0.385413
UNH	0.044037	0.033699	0.059153	0.039753	0.043485	0.093597	0.033701	0.031836	0.048071	0.031962	0.031850	0.029528	0.045475
MA	0.128232	0.247333	0.039753	0.166653	0.127537	0.376690	0.097115	0.044789	0.176738	0.072293	0.046667	0.046103	0.152587
MSFT	0.153107	0.327105	0.043485	0.127537	0.196538	0.457646	0.114128	0.043865	0.232507	0.074605	0.047629	0.042828	0.192909
NVDA	0.437065	0.946391	0.093597	0.376690	0.457646	1.968199	0.303998	0.097309	0.667123	0.202177	0.097498	0.129668	0.780242
HD	0.104816	0.233729	0.033701	0.097115	0.114128	0.303998	0.174514	0.045171	0.178219	0.064463	0.052909	0.023209	0.120721
PFE	0.039941	0.085805	0.031836	0.044789	0.043865	0.097309	0.045171	0.074790	0.053598	0.037036	0.032669	0.023058	0.027065
AMZN	0.206655	0.518836	0.048071	0.176738	0.232507	0.667123	0.178219	0.053598	0.508076	0.105951	0.048796	0.058272	0.317128
BRK-B	0.077369	0.131225	0.031962	0.072293	0.074605	0.202177	0.064463	0.037036	0.105951	0.068017	0.033468	0.044579	0.084542
PG	0.051380	0.069712	0.031850	0.046667	0.047629	0.097498	0.052909	0.032669	0.048796	0.033468	0.067722	0.005803	0.031976
XOM	0.051059	0.044441	0.029528	0.046103	0.042828	0.129668	0.023209	0.023058	0.058272	0.044579	0.005803	0.163921	0.058639
TSLA	0.229810	0.385413	0.045475	0.152587	0.192909	0.780242	0.120721	0.027065	0.317128	0.084542	0.031976	0.058639	0.790052
JPM	0.100360	0.194800	0.046407	0.109655	0.098814	0.291048	0.082029	0.046553	0.140795	0.078320	0.049021	0.047056	0.113547
V	0.111299	0.202335	0.037495	0.141003	0.107418	0.318938	0.084918	0.040476	0.151333	0.063237	0.043809	0.035578	0.140388
DIS	0.159982	0.353429	0.033432	0.153351	0.162438	0.503082	0.126262	0.037852	0.264731	0.089334	0.047742	0.061258	0.234704
GOOGL	0.177273	0.453678	0.037468	0.134188	0.195038	0.519115	0.118231	0.038494	0.277326	0.084684	0.041680	0.045271	0.223062
JNJ	0.026469	0.037640	0.021687	0.021865	0.023958	0.043247	0.028763	0.027442	0.030566	0.021609	0.026665	0.006990	0.014388
BAC	0.111187	0.229360	0.046914	0.116965	0.111431	0.331852	0.085826	0.044071	0.166623	0.082650	0.046208	0.051719	0.138084
CSCO	0.104630	0.180327	0.036107	0.087777	0.096642	0.262240	0.082952	0.038794	0.129492	0.061182	0.053439	0.033563	0.101861

3.3 Super efficient portfolio

For the geometric annualized return:

The most efficient portfolio consists of:

	weights(%)
AAPL	0.00
META	1.40
UNH	0.00
MA	0.00
MSFT	0.00
NVDA	4.95
HD	3.08
PFE	0.00
AMZN	0.00
BRK-B	0.00
PG	0.00
XOM	0.00
TSLA	0.00
JPM	76.86
V	0.00
DIS	13.72
GOOGL	0.00
JNJ	0.00
BAC	0.00
CSCO	0.00

The Portfolio's Sharpe Ratio is: 1.081876690905608

The most efficient portfolio is shown above, and the portfolio’s sharpe ratio is 1.082

For the arithmetic annualized return:

The most efficient portfolio consists of:

	weights(%)
AAPL	0.00
META	4.16
UNH	0.00
MA	0.00
MSFT	0.00
NVDA	1.08
HD	7.84
PFE	0.00
AMZN	0.00
BRK-B	0.00
PG	9.06
XOM	0.00
TSLA	0.00
JPM	58.13
V	0.00
DIS	12.92
GOOGL	0.00
JNJ	0.00
BAC	0.00
CSCO	6.81

The Portfolio's Sharpe Ratio is: 1.0561289553450046

The most efficient portfolio is shown above, and the portfolio’s sharpe ratio is 1.056