# **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
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:

```
# 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, '5')
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
```

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", 5, 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
```

```
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))

1.8s
Binomial tree value with dividend for put: 4.1140138117912395
Binomial tree value with dividend for put: 4.107373800666709
```

```
# 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)

$\square 5.3s$

0.04041911053184144 0.0395121232373139$
```

```
# 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)

$\square 2.9s$
19.531465248245006 19.827352392823183
```

```
# theta
cal_amr_theta_num = cal_partial_derivative(binomial_tree, 1, 'T')
theta_call_amr = -cal_amr_theta_num("Call", 5, X, T, div_time, div, implied_vol, r, N)
theta_put_amr = -cal_amr_theta_num("Put", 5, X, T, div_time, div, implied_vol, r, N)
print(theta_call_amr, theta_put_amr)

3.6s
-24.808805697263736 -18.542829430221897
```

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,

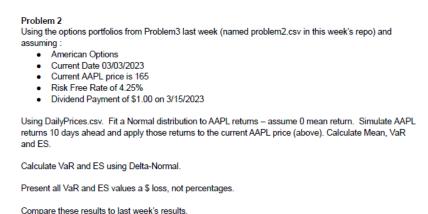
partial derivatives do not exist.

Eventually, I am able to calculate sensitivity of both of those American options to changes 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.

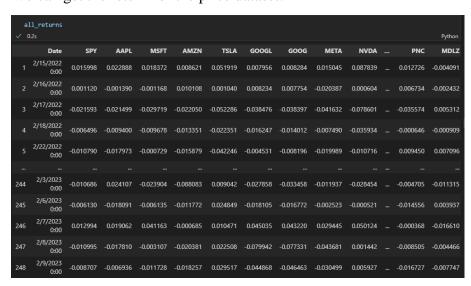
### **Problem 2**



# 2.1 Assume Normal Distribution, Calculate Mean, VaR, ES

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

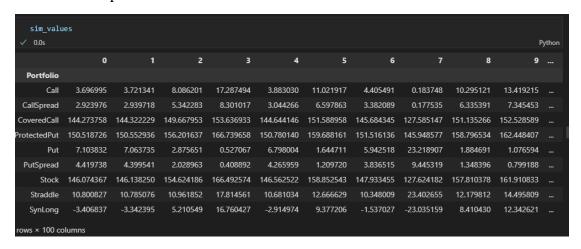
We can get the return for the price dataset:



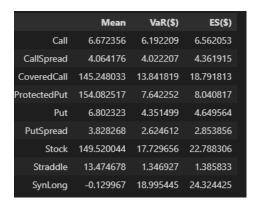
Then, we can fit normal distribution to AAPL returns. With return simulated, we can

calculate the price which is also a simulated value because of return.

The simulated price is shown below:

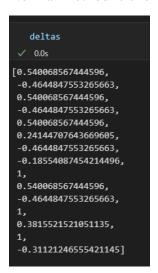


As a result, the Mean, VaR, and ES can be calculated.

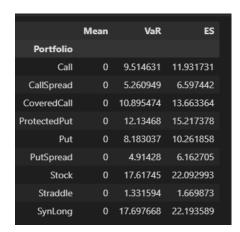


# 2.2 Using Delta-Normal

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:

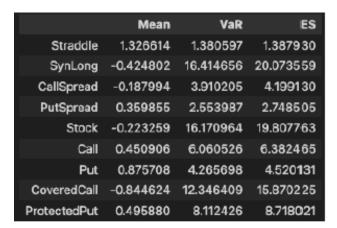


The results of VaR and ES is shown below:



# 2.3 Comparison

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



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	٧	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.

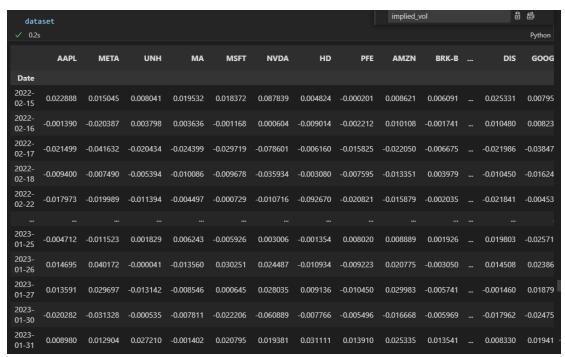
Based on the past 10 years of factor returns, find the expected annual return of each stock.

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.



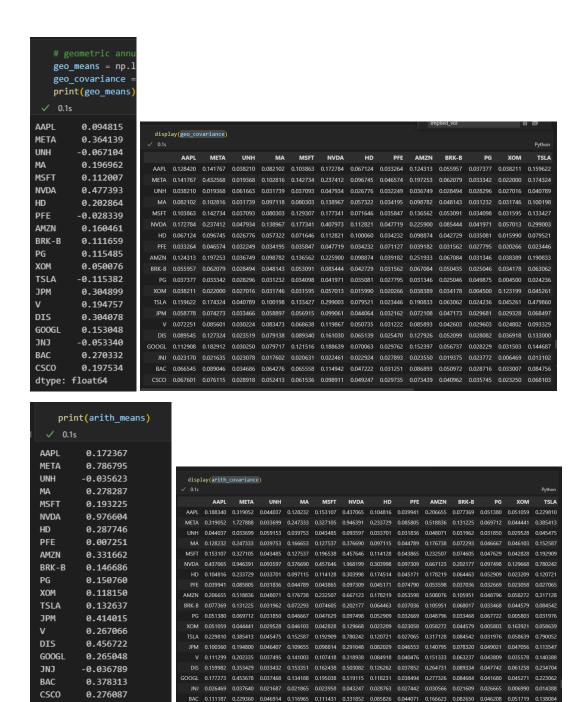
Since the datafile F-F\_Research\_data\_factors\_daily and F-F Momentum\_Factor\_daily only has data until 1/31/2023. We need to exclude the data since February 2023.

# 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)

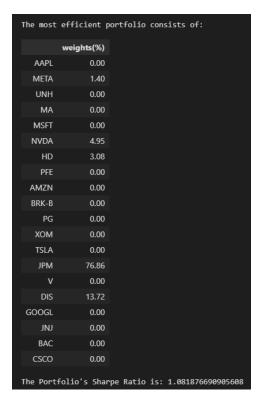
Here, I use both geometric mean and arithmetic mean for the results.



# 3.3 Super efficient portfolio

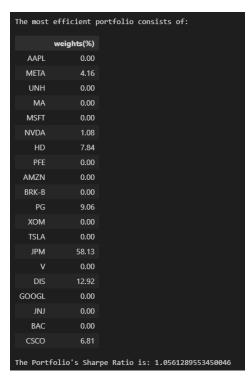
dtype: float64

For the geometric annualized return:



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 is shown above, and the portfolio's sharpe ratio is 1.056