Assignment 3 - Q1 & Q2

Qian Zhang, CID: 01939418

Yijie Wu, CID: 01894265

Dennis Wen, CID: 01973771

In [1]:
```
import pandas as pd
import numpy as np
```

Q1

In [2]:
```
data=pd.read_csv('StressData.csv')
```

In [3]:
```
data
```

Out[3]:

	Underlying Stress	Volatility Stress	Underlying	Underlying Price	Div. Yield	Security Type	Currency	Position	Strike
0	-20	-10	SPX Index	1,100	2.00%	Future	USD	25	NaN
1	-20	-10	SPX Index	1,100	2.00%	Option	USD	200	935
2	-20	-10	SPX Index	1,100	2.00%	Option	USD	-50	1,001
3	-20	-10	SPX Index	1,100	2.00%	Option	USD	25	1,100
4	-20	-10	SPX Index	1,100	2.00%	Option	USD	-25	1,034
...	...	...	...	...	...	...	...	...	...
8212	20	10	NKY Index	10,022	0.50%	Option	JPY	-10	9,721
8213	20	10	NKY Index	10,022	0.50%	Option	JPY	-5	7,617
8214	20	10	NKY Index	10,022	0.50%	Option	JPY	-10	9,721
8215	20	10	NKY Index	10,022	0.50%	Option	JPY	-10	8,819
8216	20	10	NKY Index	10,022	0.50%	Option	JPY	-10	8,819

8217 rows × 30 columns

In [9]:
```
for i in range(len(data['PnL'])):

    if int(data['PnL'][i].split(',')[0])<0 and len(data['PnL'][i].split(
','))==2:
        data['PnL'][i]=int(data['PnL'][i].split(',')[0])*1000-int(data[
'PnL'][i].split(',')[1])
    elif int(data['PnL'][i].split(',')[0])>0 and len(data['PnL'][i].spli
t(','))==2:
        data['PnL'][i]=int(data['PnL'][i].split(',')[0])*1000+int(data[
'PnL'][i].split(',')[1])
    else:
        data['PnL'][i]=int(data['PnL'][i])
```

```
/Users/YijieWu/Desktop/anaconda/anaconda3/lib/python3.7/site-packages/i
pykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
/Users/YijieWu/Desktop/anaconda/anaconda3/lib/python3.7/site-packages/i
pykernel_launcher.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/Users/YijieWu/Desktop/anaconda/anaconda3/lib/python3.7/site-packages/i
pykernel_launcher.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

In [10]:
```
df=data.groupby(['Underlying Stress','Volatility Stress'])['PnL'].sum()
```

In [13]:
```
NKY=data[data['Underlying']=='NKY Index']
SPX=data[data['Underlying']=='SPX Index']
SX5E=data[data['Underlying']=='SX5E Index']
NKY2=NKY.groupby(['Underlying Stress','Volatility Stress'])['PnL'].sum()
.to_frame().unstack()
SPX2=SPX.groupby(['Underlying Stress','Volatility Stress'])['PnL'].sum()
.to_frame().unstack()
SX5E2=SX5E.groupby(['Underlying Stress','Volatility Stress'])['PnL'].sum
().to_frame().unstack()
```

In [1]:
```
#NKY2
```

In [15]:
```
SPX2
```

Out[15]:

	PnL								
Volatility Stress	**-10**	**-5**	**-2**	**-1**	**0**	**1**	**2**	**5**	**10**
Underlying Stress									
-20	-1276	-5350	-7886	-8740	-9595	-10454	-11315	-13892	-18182
-10	6994	1614	-1511	-2535	-3555	-4569	-5576	-8563	-13440
-5	9570	3904	640	-429	-1493	-2545	-3590	-6690	-11737
-2	10652	4923	1626	544	-531	-1596	-2654	-5798	-10914
-1	10942	5210	1907	820	-252	-1324	-2387	-5532	-10670
0	11193	5471	2164	1079	0	-1071	-2135	-5291	-10443
1	11412	5702	2397	1312	233	-838	-1905	-5065	-10226
2	11597	5912	2611	1527	447	-626	-1688	-4853	-10028
5	11971	6387	3122	2047	972	-99	-1159	-4324	-9513
10	12030	6748	3585	2536	1483	433	-614	-3741	-8912
20	10502	6081	3259	2299	1332	360	-619	-3593	-8597

In [2]:
```
#SX5E2
```

In [3]:
```
#df.to_frame().unstack()
```

Q2

a)

In [18]:
```
data2=pd.read_csv('OptionsData.csv')
```

In [50]:
```
data2
```

Out[50]:

	Underlying	Underlying Price	Div. Yield	Security Type	Currency	Position	Strike	CallPut	Maturity	Im V
0	SPX Index	1100	2.00%	Future	USD	25	NaN	NaN	19-Mar-10	Na
1	SPX Index	1100	2.00%	Option	USD	200	935.0	Put	19-Mar-10	44.40'
2	SPX Index	1100	2.00%	Option	USD	-50	1001.0	Put	19-Mar-10	38.40'
3	SPX Index	1100	2.00%	Option	USD	25	1100.0	Put	19-Mar-10	32.50'
4	SPX Index	1100	2.00%	Option	USD	-25	1034.0	Put	19-Mar-10	36.00'
...	...	...	...	...	...	...	...	...	...	
78	NKY Index	10022	0.50%	Option	JPY	-10	9721.0	Put	17-Sep-10	34.90'
79	NKY Index	10022	0.50%	Option	JPY	-5	7617.0	Put	17-Sep-10	57.80'
80	NKY Index	10022	0.50%	Option	JPY	-10	9721.0	Put	17-Sep-10	34.90'
81	NKY Index	10022	0.50%	Option	JPY	-10	8819.0	Put	17-Sep-10	42.50'
82	NKY Index	10022	0.50%	Option	JPY	-10	8819.0	Put	17-Sep-10	42.50'

83 rows × 27 columns

In [60]:
```
data2.groupby(['Underlying','Currency'])['Total $Delta'].sum()
```

Out[60]:
```
Underlying  Currency
NKY Index   JPY        -164156
SPX Index   USD          24442
SX5E Index  EUR         -17486
Name: Total $Delta, dtype: int64
```

In [61]:
```
-164156/132.71
```

Out[61]:
```
-1236.9527541255368
```

In [62]:
```
24442/1.22
```

Out[62]:
```
20034.426229508197
```

eur=1.22usd

eur=132.71jpy

for NKY index the total Delta is -164156/132.71=-1236.95

for SPX index the total Delta is 24442/1.22=20034.43

for SX5E index the total Delta is -17486

The number is consistent because we can see that Delta is 24442 and is positive, which means when volatility fixed, as s increases, option price will also increase. It is consistent with Question 1.

b)

In [19]:
```
data2.groupby(['Underlying','Currency'])['Total $Gamma'].sum()
```

Out[19]:
```
Underlying  Currency
NKY Index   JPY        -1508038
SPX Index   USD         -103722
SX5E Index  EUR         -458146
Name: Total $Gamma, dtype: int64
```

In [21]:
```
-1508038/132.71
```

Out[21]:
```
-11363.408936779444
```

In [22]:
```
-103722/1.22
```

Out[22]:
```
-85018.03278688525
```

for NKY index the total Gamma is -11363.41

for SPX index the total Gamma is -85018.03

for SX5E index the total Gamma is -458146

The SPX number is consistent because we can see that although Gamma is -85018, but the option price is also influenced by Delta and Delta is positive. which means when volatility fixed, as s increases, option price will increase with decreasing growth rate. It is consistent with Question 1.

c)

In [20]:
```
data2.groupby(['Underlying','Currency'])['Total Vega 1%'].sum()
```

Out[20]:
```
Underlying  Currency
NKY Index   JPY         -13304
SPX Index   USD          -1074
SX5E Index  EUR          -3809
Name: Total Vega 1%, dtype: int64
```

In [23]:
```
-13304/132.71
```

Out[23]:
```
-100.24866249717428
```

In [24]:
```
-1074/1.22
```

Out[24]:
```
-880.327868852459
```

for NKY index the total Gamma is -100.25

for SPX index the total Gamma is -880.33

for SX5E index the total Gamma is -3809

In the case of the SPX, the numbers are consistent with the result since as implied volatility increases, because vega is a negative number, the P&L will decrease. We can see from the P&L table, when keep S, increase implied volatilty leads to lower P&L

In [ ]:
```

```

Assignment3 - Q3

Qian Zhang-01939418, Yijie Wu-01894265, Dennis Wen-01973771

24/05/2021

```r
# Load the data and calculate true_mean and true_sigma
ER_and_SD <- read_excel("Return-Covariance-Data.xlsx", sheet = "ER and SD")
Covariance_matrix <- read_excel("Return-Covariance-Data.xlsx", sheet = "Covariance matrix")
true_mu = ER_and_SD$'Expected Return (%)'
true_sigma = data.matrix(Covariance_matrix[,-1])
```

```r
# Generate monthly return for the next 5 years
set.seed(10)
monthly_return = mvrnorm(n = 60, true_mu, true_sigma, tol = 1e-06, empirical = FALSE)
```

```r
sample_mu = colMeans(monthly_return)
sample_Sigma = cov(monthly_return)
```

```r
sample_mu
```

Q(a). Compute the sample mean and the sample covariance matrix of the returns generated

```
## [1]  0.0005749346 -0.0007161317 -0.0037812598  0.0086494588  0.0023125916
## [6]  0.0023554647 -0.0006760876  0.0077029283
```

```r
sample_Sigma
```

```
##               [,1]          [,2]          [,3]         [,4]         [,5]
## [1,]  1.703894e-04  2.055828e-04  9.197315e-05 1.750186e-04 5.760712e-05
## [2,]  2.055828e-04  3.463092e-04 -2.182164e-05 4.604476e-05 9.445135e-05
## [3,]  9.197315e-05 -2.182164e-05  2.139421e-03 1.457947e-03 1.040427e-03
## [4,]  1.750186e-04  4.604476e-05  1.457947e-03 4.827195e-03 2.895509e-03
## [5,]  5.760712e-05  9.445135e-05  1.040427e-03 2.895509e-03 3.914300e-03
## [6,] -1.082656e-04 -1.208449e-04  1.221430e-03 1.347107e-03 1.517789e-03
## [7,] -7.211613e-05 -1.986048e-04  1.643860e-03 1.827126e-03 1.688894e-03
## [8,]  1.917602e-04  1.998457e-04  1.291927e-03 1.201110e-03 9.876792e-04
##               [,6]          [,7]         [,8]
## [1,] -0.0001082656 -7.211613e-05 0.0001917602
## [2,] -0.0001208449 -1.986048e-04 0.0001998457
## [3,]  0.0012214303  1.643860e-03 0.0012919269
```

```
## [4,]  0.0013471074  1.827126e-03 0.0012011098
## [5,]  0.0015177894  1.688894e-03 0.0009876792
## [6,]  0.0041662415  1.537358e-03 0.0007934992
## [7,]  0.0015373577  3.461959e-03 0.0009513552
## [8,]  0.0007934992  9.513552e-04 0.0018934810
```

```
## Form problem
SAMPLES <- 100
n <- 8
w <- Variable(n)
ret <- t(sample_mu) %*% w
risk <- quad_form(w, sample_Sigma)
constraints <- list(w >= 0, sum(w) == 1)

## Risk aversion parameters
gammas <- 10^seq(-2, 3, length.out = SAMPLES)
ret_data <- rep(0, SAMPLES)
risk_data <- rep(0, SAMPLES)
w_data <- matrix(0, nrow = SAMPLES, ncol = n)

## Compute trade-off curve
for(i in seq_along(gammas)) {
    gamma <- gammas[i]
    objective <- ret - gamma * risk
    prob <- Problem(Maximize(objective), constraints)
    result <- solve(prob)

    ## Evaluate risk/return for current solution
    risk_data[i] <- result$getValue(sqrt(risk))
    ret_data[i] <- result$getValue(ret)
    w_data[i,] <- result$getValue(w)
}
w_data_rounded <- round(w_data,2)
```

```
# Obtain portfolio's estimated efficient frontier
index = round(seq(1, 100, length.out = 10))

portfolio <- data.frame(w_data_rounded,risk_data,ret_data)
names(portfolio)[1:8] <- names(Covariance_matrix)[-1]
names(portfolio)[9] <- 'estimate risk'
names(portfolio)[10] <- 'estimate return'
```

Q(b). Compute at least ten long-only efficient portfolios along the efficient frontier based on the estimates in part (a)

```
choosen_portfolio <- portfolio[index,]
choosen_portfolio_weight = data.matrix(choosen_portfolio[0:8], rownames.force = NA)
```

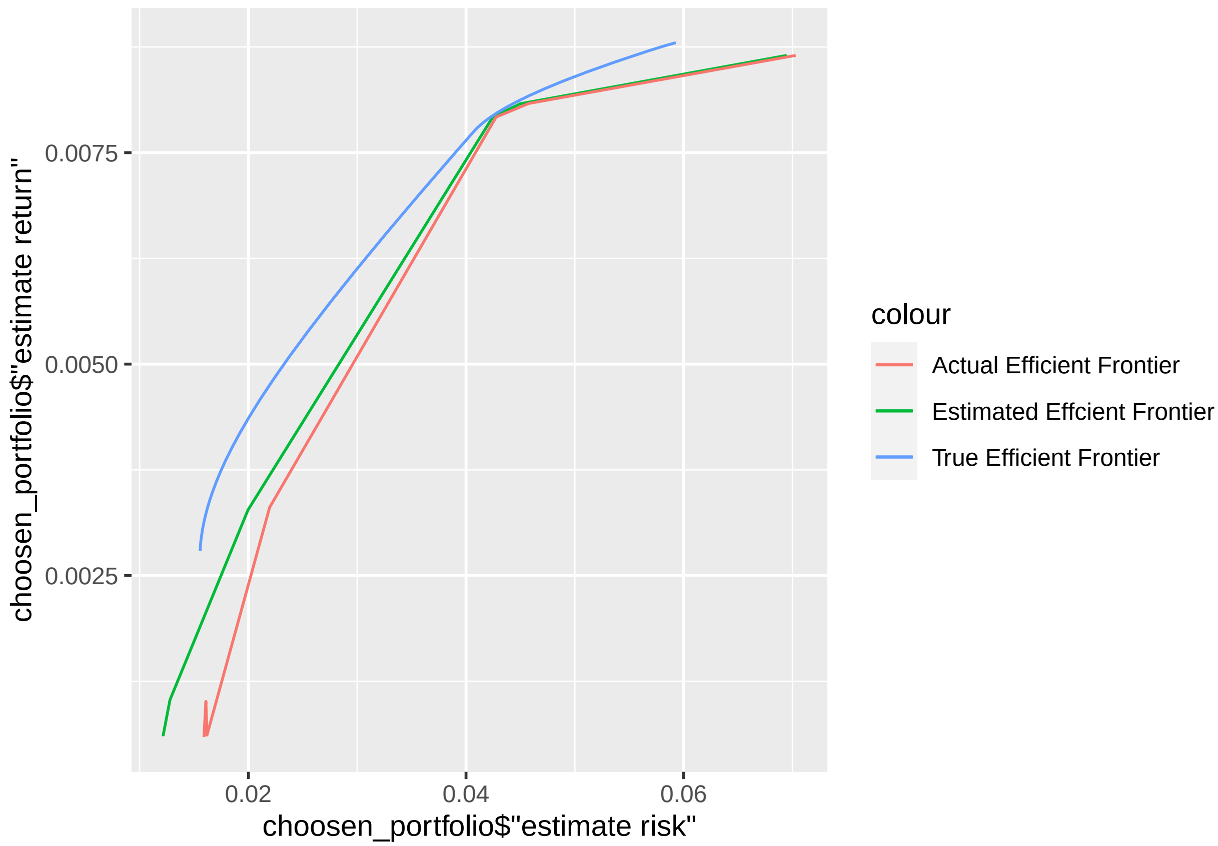
```
actual_return <- monthly_return%*%t(choosen_portfolio_weight)
actual_mu <- colMeans(actual_return)
actual_sigma <- cov(actual_return)

actual_variance <- rep(0, 10)
for (i in seq_along(actual_variance)){
  actual_variance[i] = t(choosen_portfolio_weight[i,])%*%true_sigma%*%choosen_portfolio_weight[i,]
}
```

Q(c). Compute the actual expected returns and standard deviations for the portfolios found in step (b)

```
# Calculate the mean and standard deviation for true frontier

## Form problem
n <- 8
w <- Variable(n)
ret <- t(true_mu) %*% w
risk <- quad_form(w, true_sigma)
constraints <- list(w >= 0, sum(w) == 1)

## Risk aversion parameters
gammas <- 10^seq(-2, 3, length.out = SAMPLES)
ret_data <- rep(0, SAMPLES)
risk_data <- rep(0, SAMPLES)
w_data <- matrix(0, nrow = SAMPLES, ncol = n)

## Compute trade-off curve
for(i in seq_along(gammas)) {
    gamma <- gammas[i]
    objective <- ret - gamma * risk
    prob <- Problem(Maximize(objective), constraints)
    result <- solve(prob)

    ## Evaluate risk/return for current solution
    risk_data[i] <- result$getValue(sqrt(risk))
    ret_data[i] <- result$getValue(ret)
    w_data[i,] <- result$getValue(w)
}
```

```
# Plot the estimated efficient frontier, the actual frontier, and the true frontier
ggplot() +
  geom_line(mapping = aes(x = choosen_portfolio$'estimate risk', y = choosen_portfolio$'estimate return
  geom_line(mapping = aes(x = risk_data, y = ret_data, colour  = "True Efficient Frontier")) +
  geom_line(mapping = aes(x = sqrt(actual_variance), y = actual_mu, colour  = "Actual Efficient Frontie
```

Q(d). Plot the estimated efficient frontier, the actual frontier, and the true frontier



Q(e). After running all the above steps several times, we find that estimated frontiers and actual frontiers don't fully overlap each other for a single time. That is to say, the portfolios we estimated cannot be completely realized. And on average, the realized frontiers have worse performance than what we believe we can obtain.