Rohit Penumarti
PSTAT 176
6/1/20

HW#4

1. a) Let R - revenue, C - cost, X - # of insurance bought, $x^* = \frac{x}{1000}$

$$E\left(\frac{R-C}{C} \mid X=x\right) = \frac{E(R \mid X=x^*)}{C} - 1 = \frac{0.2x^* + 0.8(150)}{100+0.3x^*} - 1$$

$$\boxed{\text{Expected return} = \frac{0.2x^* + 120}{0.3x^* + 100} - 1}$$

b) $$Var\left(\frac{R-C}{C} \mid X=x^*\right) = Var\left(\frac{R}{C} \mid X=x^*\right) = \frac{Var(R \mid X=x^*)}{C^2}$$

$$= \frac{E(R^2) - E(R)^2}{C^2} = \frac{0.2x^{*2} + 0.8(150^2) - 0.04x^{*2} - 48x^* - 120^2}{(0.3x^*+100)^2}$$

$$= \frac{0.16x^{*2} - 48x^* + 3600}{(0.3x^*+100)^2} = \boxed{\frac{0.16(x^*-150)^2}{(0.3x^*+100)^2}}$$

c) let $f(x^*) = E(K_v) - 0.5\,Var(K_v)$

$$= \frac{0.2x^*+120}{0.3x^*+100} - 1 - \frac{0.08(x^*-150)^2}{(0.3x^*+100)^2}$$

$$f'(x^*) = 0 = \frac{0.2(0.3x^*+100) - 0.3(0.2x^*+120)}{(0.3x^*+100)^2} - \frac{0.16(x^*-150)(100+0.3x^*)^2 - 0.048(0.3x^*+100)(x^*-150)^2}{(0.3x^*+100)^4}$$

$$= (100+0.3x^*)^2(20+\cancel{0.06x^*}-\cancel{0.06x^*}-36) - 0.16(x^*-150)(0.09x^{*2}+60x^*+10000) + 0.048(0.3x^*+100)(x^{*2}-300x^*+22500)$$

$$= -1.44x^{*2} - 960x^* - 160000 - 0.16(0.09x^{*3} + 60x^{*2} + 10000x^* - 13.5x^{*2} - 9000x^* - 1500000) + 0.048(0.3x^{*3} - 90x^{*2} + 6750x^* + 100x^{*2} - 30000x^* + 2250000)$$

$$= -1.44x^{*2} - 960x^* - 160000 - \cancel{0.0144x^{*3}} - 9.6x^{*2} - 1600x^* + 2.16x^{*2} + \cancel{1440x^*} + 240000 + \cancel{0.0144x^{*3}} - 4.32x^{*2} + 384x^* + 4.8x^{*2} - \cancel{1440x^*} + 108000$$

$$0 = -8.4x^{*2} - 2236x^* + 188000$$

$$= 8.4x^{*2} + 2236x^* - 188000$$

$$x^* = \frac{-2236 \pm \sqrt{2236^2 - 4(8.4)(-188000)}}{2(8.4)}$$

$$\boxed{x^* = 67.14}$$

2. $r=0{,}04$

① $\mu_1 = r + \beta(\mu_m - r)$ where $\beta = \frac{Cov(K_1, K_m)}{Var(K_m)}$

We have

$$\mu_1 = E(K_1) = E\left(\frac{S_1(1) - S_1(0)}{S_1(0)}\right) = \frac{1}{S_1(0)}\left(E(S_1(1)) - S_1(0)\right) = \frac{E(S_1(1))}{S_1(0)} - 1$$

$$= 0{,}1\left(\frac{30 - S_1(0)}{S_1(0)}\right) + 0{,}3\left(\frac{15 - S_1(0)}{S_1(0)}\right) + 0{,}4\left(\frac{25 - S_1(0)}{S_1(0)}\right) + 0{,}15\left(\frac{20 - S_1(0)}{S_1(0)}\right) + 0{,}05\left(\frac{10 - S_1(0)}{S_1(0)}\right)$$

$$= \frac{21 - S_1(0)}{S_1(0)}$$

$$\mu_m = 0{,}1\left(\frac{112}{100} - 1\right) + 0{,}3\left(\frac{108}{100} - 1\right) + 0{,}4\left(\frac{104}{100} - 1\right) + 0{,}15\left(\frac{100}{100} - 1\right) + 0{,}05\left(\frac{92}{100} - 1\right)$$

$$= 0{,}048$$

$$Cov(K_1, K_m) = 0{,}1\left(\frac{30}{S_1(0)} - 1 - \frac{21}{S_1(0)} - 1\right)(0{,}12 - 0{,}048) + \left(\frac{15}{S_1(0)} - 1 - \frac{21}{S_1(0)} + 1\right)(0{,}08 - 0{,}048)\,0{,}3$$

$$+ 0{,}4\left(\frac{25}{S_1(0)} - 1 - \frac{21}{S_1(0)} - 1\right)(0{,}04 - 0{,}048) + \left(\frac{20}{S_1(0)} - 1 - \frac{21}{S_1(0)} + 1\right)(0 - 0{,}048)\cdot 0{,}15$$

$$+ 0{,}05\left(\frac{10}{S_1(0)} - 1 - \frac{21}{S_1(0)} - 1\right)(-0{,}08 - 0{,}048)$$

$$= \frac{0{,}072}{S_1(0)}$$

$$Var(K_m) = 0{,}1(0{,}12 - 0{,}048)^2 + 0{,}3(0{,}08 - 0{,}048)^2 + 0{,}4(0{,}04 - 0{,}048)^2 + 0{,}15(0 - 0{,}048)^2 + 0{,}05(-0{,}08 - 0{,}048)^2$$

$$= 0{,}002016$$

$$\beta = \frac{Cov(K_1, K_m)}{Var(K_m)} = \frac{0{,}072}{S_1(0)} \cdot \frac{1}{0{,}002016} = \frac{35{,}714}{S_1(0)}$$

① $$\frac{21}{S_1(0)} - 1 - 0{,}04 = \frac{35{,}714}{S_1(0)}(0{,}048 - 0{,}04)$$

$$\frac{21}{S_1(0)} - \frac{35{,}714}{S_1(0)}(0{,}048 - 0{,}04) = 1{,}04$$

$$\frac{20{,}714}{S_1(0)} = 1{,}04$$

$$S_1(0) = \frac{20{,}714}{1{,}04} = \boxed{19{,}92}$$

In [1]:

```python
import numpy as np
import pandas as pd
from datetime import datetime
```

Problem 3

In [99]:

```python
# Part a)
df = pd.read_csv("IBM-MSFT-HAS.csv")
df['Date'] = pd.to_datetime(df['Date'])

returns = np.zeros((len(df['Date'])-1, 3))
for i in range(3):
    for j in range(len(df['Date'])-1):
        returns[j, i] = (df['IBM'][j+1]/df['IBM'][j] - 1)*(i == 0)+(df['MSFT'][j+1]/df['MSFT'][j] - 1
)*(i == 1) \
        + (df['HAS'][j+1]/df['HAS'][j] - 1)*(i == 2)

df['IBM Returns'] = pd.Series(returns[:, 0])
df['MSFT Returns'] = pd.Series(returns[:, 1])
df['HAS Returns'] = pd.Series(returns[:, 2])

df_nov = df[np.logical_and(df.Date.dt.month == 11, df.Date.dt.year == 2017)]
df_dec = df[np.logical_and(df.Date.dt.month == 12, df.Date.dt.year == 2017)]
df_jan = df[np.logical_and(df.Date.dt.month == 1, df.Date.dt.year == 2018)]
df_feb = df[np.logical_and(df.Date.dt.month == 2, df.Date.dt.year == 2018)]
df_mar = df[np.logical_and(df.Date.dt.month == 3, df.Date.dt.year == 2018)]
df_apr = df[np.logical_and(df.Date.dt.month == 4, df.Date.dt.year == 2018)]

data_nov = df_nov[['IBM Returns', 'MSFT Returns', 'HAS Returns']].to_numpy()
data_dec = df_dec[['IBM Returns', 'MSFT Returns', 'HAS Returns']].to_numpy()
data_jan = df_jan[['IBM Returns', 'MSFT Returns', 'HAS Returns']].to_numpy()
data_feb = df_feb[['IBM Returns', 'MSFT Returns', 'HAS Returns']].to_numpy()
data_mar = df_mar[['IBM Returns', 'MSFT Returns', 'HAS Returns']].to_numpy()
data_apr = df_apr[['IBM Returns', 'MSFT Returns', 'HAS Returns']].to_numpy()
```

In [100]:

```python
daily_avg_returns_half = np.zeros((3, 6))

for i in range(3):
    for j in range(6):
        daily_avg_returns_half[i, j] = np.nanmean(data_nov[:, i])*(j == 0) + np.nanmean(data_dec[:,
i])*(j == 1) \
        + np.nanmean(data_jan[:, i])*(j == 2) + np.nanmean(data_feb[:, i])*(j == 3) + np.nanmean(dat
a_mar[:, i])*(j == 4) \
        + np.nanmean(data_apr[:, i])*(j == 5)

avg_returns_IBM = 20*np.average(daily_avg_returns_half[0])
avg_returns_MSFT = 20*np.average(daily_avg_returns_half[1])
avg_returns_HAS = 20*np.average(daily_avg_returns_half[2])
avg_returns = np.array([avg_returns_IBM, avg_returns_MSFT, avg_returns_HAS])
cov = 400*np.cov(daily_avg_returns_half, ddof=0)
print("The average monthly returns for IBM, Microsoft, and Hasbro are: " + str(avg_returns) + "
respectively. The "
     "covariance matrix of the average returns is: \n" + str(cov) + " \nwhere the variances are giv
en along the diagonal.")
```

```
The average monthly returns for IBM, Microsoft, and Hasbro are: [-0.0053536   0.02347976
0.00113885] respectively. The covariance matrix of the average returns is:
[[0.00099269 0.0008125  0.0005805 ]
 [0.0008125  0.00172114 0.0020343 ]
 [0.0005805  0.0020343  0.00311119]]
where the variances are given along the diagonal.
```

In [101]:

```
# Part b)
r = 0.02
uVec = np.ones(np.size(cov, axis=0))
market_w = np.linalg.multi_dot([(avg_returns-r*uVec), np.linalg.inv(cov)])
market_w /= np.sum(market_w)
mu_market = np.linalg.multi_dot([market_w, np.transpose(avg_returns)])
sigma_market = np.sqrt(np.linalg.multi_dot([market_w, np.transpose(cov), np.transpose(market_w)]))

print("The market portfolio is: " + str(market_w) + " which has expected return: " + str(mu_market
) + " and standard "
      "deviation: " + str(sigma_market))
```

```
The market portfolio is: [ 2.86990494 -4.37265362  2.50274868] which has expected return: -0.11518
295512079778 and standard deviation: 0.06319563574321117
```

In [102]:

```
# Part c)
df_may = df[np.logical_and(df.Date.dt.month == 5, df.Date.dt.year == 2017)]
df_jun = df[np.logical_and(df.Date.dt.month == 6, df.Date.dt.year == 2017)]
df_jul = df[np.logical_and(df.Date.dt.month == 7, df.Date.dt.year == 2017)]
df_aug = df[np.logical_and(df.Date.dt.month == 8, df.Date.dt.year == 2017)]
df_sep = df[np.logical_and(df.Date.dt.month == 9, df.Date.dt.year == 2017)]
df_oct = df[np.logical_and(df.Date.dt.month == 10, df.Date.dt.year == 2017)]

data_may = df_may[['IBM Returns', 'MSFT Returns', 'HAS Returns']].to_numpy()
data_jun = df_jun[['IBM Returns', 'MSFT Returns', 'HAS Returns']].to_numpy()
data_jul = df_jul[['IBM Returns', 'MSFT Returns', 'HAS Returns']].to_numpy()
data_aug = df_aug[['IBM Returns', 'MSFT Returns', 'HAS Returns']].to_numpy()
data_sep = df_sep[['IBM Returns', 'MSFT Returns', 'HAS Returns']].to_numpy()
data_oct = df_oct[['IBM Returns', 'MSFT Returns', 'HAS Returns']].to_numpy()
```

In [103]:

```
daily_avg_returns_other = np.zeros((3, 6))

for i in range(3):
    for j in range(6):
        daily_avg_returns_other[i, j] = np.nanmean(data_may[:, i])*(j == 0) +
np.nanmean(data_jun[:, i])*(j == 1) \
        + np.nanmean(data_jul[:, i])*(j == 2) + np.nanmean(data_aug[:, i])*(j == 3) + np.nanmean(dat
a_sep[:, i])*(j == 4) \
        + np.nanmean(data_oct[:, i])*(j == 5)

avg_returns_IBM = 20*np.average(daily_avg_returns_other[0])
avg_returns_MSFT = 20*np.average(daily_avg_returns_other[1])
avg_returns_HAS = 20*np.average(daily_avg_returns_other[2])
avg_returns = np.array([avg_returns_IBM, avg_returns_MSFT, avg_returns_HAS])
cov = 400*np.cov(daily_avg_returns_other, ddof=0)
print("The average monthly returns for IBM, Microsoft, and Hasbro are: " + str(avg_returns) + "
respectively. The "
     "covariance matrix of the average returns is: \n" + str(cov) + " \nwhere the variances are giv
en along the diagonal.")
```

```
The average monthly returns for IBM, Microsoft, and Hasbro are: [-0.00120631  0.03107525 -
0.00927302] respectively. The covariance matrix of the average returns is:
[[ 1.36745138e-03  4.47166986e-05 -3.14882985e-04]
 [ 4.47166986e-05  1.62915330e-03 -1.29021203e-03]
 [-3.14882985e-04 -1.29021203e-03  2.32567111e-03]]
where the variances are given along the diagonal.
```

In [104]:

```
r = 0.02
uVec = np.ones(np.size(cov, axis=0))
market_w = np.linalg.multi_dot([(avg_returns-r*uVec), np.linalg.inv(cov)])
market_w /= np.sum(market_w)
mu_market = np.linalg.multi_dot([market_w, np.transpose(avg_returns)])
sigma_market = np.sqrt(np.linalg.multi_dot([market_w, np.transpose(cov), np.transpose(market_w)]))

print("The market portfolio is: " + str(market_w) + " which has expected return: " + str(mu_market
```

```
) + " and standard "
      "deviation: " + str(sigma_market))
```

```
The market portfolio is: [0.41073013 0.17555926 0.41371061] which has expected return:
0.0011237340756777557 and standard deviation: 0.019773028055737087
```

> The market portfolio from part c) compared to that of part b) is different because in part c), there is no shorting and each of the weights are fractional amounts of each stock whereas for part b), the weights are larger, and Microsoft is shorted. Furthermore, the return on the portfolio on part c) is very small but positive whereas the return on part b) is negative and has absolute value greater than that of part c). The standard deviation of part c) is much smaller than that of part b) by about 3 times.

In [105]:

```
# Part d)
daily_returns = np.append(daily_avg_returns_half, daily_avg_returns_other, axis=1)
avg_returns_IBM_whole = np.average(daily_returns[0])
avg_returns_MSFT_whole = np.average(daily_returns[1])
avg_returns_HAS_whole = np.average(daily_returns[2])
avg_returns = np.array([avg_returns_IBM_whole, avg_returns_MSFT_whole, avg_returns_HAS_whole])
cov = np.cov(daily_returns, ddof=0)
print("The average monthly returns for the whole year for IBM, Microsoft, and Hasbro are: " + str(
avg_returns) +
      " respectively. The covariance matrix of the average returns is: \n" + str(cov) + " \nwhere t
he variances are given "
      + "along the diagonal.")
```

```
The average monthly returns for the whole year for IBM, Microsoft, and Hasbro are: [-0.000164     0
.00136388 -0.00020335] respectively. The covariance matrix of the average returns is:
[[2.96093084e-06 1.09120462e-06 3.05034588e-07]
 [1.09120462e-06 4.22392180e-06 8.80688086e-07]
 [3.05034588e-07 8.80688086e-07 6.86383535e-06]]
where the variances are given along the diagonal.
```

In [106]:

```
r = 0.02
uVec = np.ones(np.size(cov, axis=0))
market_w = np.linalg.multi_dot([(avg_returns-r*uVec), np.linalg.inv(cov)])
market_w /= np.sum(market_w)
mu_market = np.linalg.multi_dot([market_w, np.transpose(avg_returns)])
sigma_market = np.sqrt(np.linalg.multi_dot([market_w, np.transpose(cov), np.transpose(market_w)]))

print("The market portfolio is: " + str(market_w) + " which has expected return: " + str(mu_market
) + " and standard "
      "deviation: " + str(sigma_market))
```

```
The market portfolio is: [0.53953989 0.23390143 0.22655868] which has expected return:
0.00018445732832092986 and standard deviation: 0.0013742891567943881
```

4. $u=1.1,\ d=0.92,\ S_0=20,\ r=0.04,\ p=0.5,\ U(x)=-2e^{-x/2},\ U'(x)=e^{-x/2}$

a) $\xi_i = \dfrac{z_i/p_i}{(1+r)^N}$ $\quad z=\dfrac{1+r-d}{u-d}=\dfrac{2}{3},\quad p_i=\left(\tfrac{1}{2}\right)^N$

$z_i=\left(\tfrac{2}{3}\right)^{N-n}\left(\tfrac{1}{3}\right)^n$ where $n = \#$ of tails

$X_N^* = \max_{X_N} \sum_i p_i U(X_i) + \lambda\left(\sum_i p_i \xi_i X_i - X_0\right)$ where we are summing over paths

$= \max_{X_N} \sum_i \left(\tfrac{1}{2}\right)^N\left(-2e^{-X_i/2}\right) + \lambda\left(\sum \dfrac{\left(\frac{2}{3}\right)^{N-n}\left(\frac{1}{3}\right)^n}{1.04^N} X_i - X_0\right)$

To find X_N^* that maximizes $E[U(X_N)]$, solve the following lagrange multiplier:

$$\boxed{X_N^* = -\left(\tfrac{1}{2}\right)^{N-1} \max_{X_N} \sum_i e^{-X_i/2} + \lambda\left(\sum_i \dfrac{\left(\frac{2}{3}\right)^{N-n}\left(\frac{1}{3}\right)^n}{1.04^N} X_i - X_0\right)}$$

To find X_3^* explicitly for $N=3$, $X_0=10$, we have

$$X_0(1+r)^N = E^Q(X_N^*) = \sum_i z_i X_i^* = \sum_i z_i\left[-2\ln(-\lambda^* \xi_i)\right]$$

$$10(1.04)^3 = \left(\tfrac{2}{3}\right)^3\left[-2\ln(-2.107\lambda^*)\right] + \left(\tfrac{2}{3}\right)^2\left(\tfrac{1}{3}\right)(3)\left[-2\ln(-1.054\lambda^*)\right] + \tfrac{2}{3}\left(\tfrac{1}{3}\right)^2(3)\left[-2\ln(-0.527\lambda^*)\right] + \left(\tfrac{1}{3}\right)^3\left[-2\ln(-0.263\lambda^*)\right]$$

Using wolfram to solve, we get

$$\lambda^* = -0.00342484$$

Then,

$$\boxed{\begin{aligned} &X_3^*(HHH) = -2\ln(-2.107\lambda^*) = 9.86287\\ &X_3^*(HHT)=X_3^*(HTH)=X_3^*(THH) = -2\ln(-1.054\lambda^*) = 11.2482\\ &X_3^*(HTT)=X_3^*(THT)=X_3^*(TTH) = -2\ln(-0.527\lambda^*) = 12.6345\\ &X_3^*(TTT) = -2\ln(-0.263\lambda^*) = 14.0246 \end{aligned}}$$

$$u(X_0) = E[U(X_3^*)] = \sum_i p_i U(X_3^*) = \left(\tfrac{1}{2}\right)^3\left[-2e^{-9.86287/2} - 3\left(2e^{-11.2482/2}\right) - 3\left(2e^{-12.6345/2}\right) - 2e^{-14.0246/2}\right]$$

$$\boxed{u(X_0) = -0.00609}$$