Stat_HW6_Solution

In [1]:
```
#載入所需函示庫
from matplotlib import pyplot as plt
%matplotlib inline

# 設定圖形大小; DPI越大圖越大
plt.rcParams["figure.dpi"] = 100
import seaborn as sns
import pandas as pd
import numpy as np
import scipy.stats as stats
import statsmodels.api as sm
import statsmodels.stats.api as sms
import statsmodels.formula.api as smf
import math as math
```

Exercise 7.13

a. (2%)

X: the number of jobs a university graduate will be offered
P(X<2)=P(0)+P(1)=.05+.43=.48

b. (2%)

X: the number of jobs offered
P(X>1)=P(2)+P(3)=.31+.21=.52

Exercise 7.27

a. (2%)

X:the number of times a student walks into the library
P(X>=20) = P(20) + P(25) + P(30) + P(40) + P(50) + P(75) + P(100) = .08 + .05 + .04 + .04 + .03 + .03 + .01 = .28

b. (2%)

X:the number of times a student walks into the library
P(X=60)=0

c. (2%)

X:the number of times a student walks into the library
P(X>50)=P(75) + P(100) = .03 + .01 = .04

d. (2%)

X:the number of times a student walks into the library
P(X>100)=0

Exercise 7.35 (2% for mean, 2% for standard deviation)

mean = E(X) = 0(.10) + 1(.20) + 2(.25) + 3(.25) + 4(.20) = 2.25
variance = (.10)(0–2.25)^2 + (.20)(1–2.25)^2 + (.25)(2–2.25)^2 + (.13)(3–2.25)^2 + (.20)(4–2.25)^2 = 1.59
standard deviation = √1.59 = 1.26

Exercise 7.55 (4%)

Y/X	0	1
1	0.04	0.16
2	0.08	0.32
3	0.08	0.32

Exercise 7.59

a. (2%)

P(CMD = 0 and SD = 2) = .06

b. (2%)

P(CMD = 2 and SD = 0) = 0

c. (2%)

P(CMD ≥ 1 and SD ≥ 1) = .07 + .01 + .10 + .05 + .04 + .02 = .29

Exercise 7.79

In [2]:
```
df_xr07nyse = pd.read_excel('Xr07-NYSE.xlsx')
display(df_xr07nyse.head())

GE_Mean=df_xr07nyse["GE"].mean()
JNJ_Mean=df_xr07nyse["JNJ"].mean()
MCD_Mean=df_xr07nyse["MCD"].mean()
MRK_Mean=df_xr07nyse["MRK"].mean()

cov_mat = np.cov(df_xr07nyse[['GE','JNJ','MCD','MRK']].values,rowvar=False,ddof=0)
print(cov_mat)
```

	Year	Month	MMM	AXP	BA	CAT	CVX	KO	DIS	DD	...	MRK	NKE	PFE	PG	
0	2011.0	January	0.055331	0.004380	0.042435	0.061025	0.101130	0.017025	0.125290	0.090908	...	-0.018089	0.079413	0.067086	-0.001267	0.06
1	NaN	February	0.013770	0.037411	0.026663	0.081803	0.036048	0.045385	-0.014861	0.001823	...	0.025352	-0.146730	0.055613	-0.022998	-0.00
2	NaN	March	0.039679	0.090161	0.079129	0.040778	0.018141	0.016883	0.000232	0.033109	...	0.089064	0.087451	0.032496	0.062309	0.06
3	NaN	April	-0.023390	0.051345	-0.016770	-0.083268	-0.033980	-0.009635	-0.034107	-0.054519	...	0.022253	0.025875	0.032792	0.032357	-0.01
4	NaN	May	0.004980	0.005571	-0.052544	0.006238	-0.019731	0.014476	-0.062215	0.014071	...	-0.029323	0.069519	-0.039627	-0.051194	-0.05

5 rows × 28 columns

```
[[0.00303843 0.0009583  0.0008295  0.00086191]
 [0.0009583  0.0013647  0.00069022 0.00084199]
 [0.0008295  0.00069022 0.0013038  0.00047924]
 [0.00086191 0.00084199 0.00047924 0.00212063]]
```

a. (3% for mean, 3% for standard deviation)

In [3]:
```
weight=[0.25,0.25,0.25,0.25]
portfolio_mean=GE_Mean*weight[0]+JNJ_Mean*weight[1]+MCD_Mean*weight[2]+MRK_Mean*weight[3]

portfolio_variance=0
for i in range(4):
    for j in range(4):
        portfolio_variance=portfolio_variance+weight[i]*weight[j]*cov_mat[i][j]

portfolio_std=math.sqrt(portfolio_variance)
portfolio_std
print("MEAN:",portfolio_mean)
print("STANDARD DEVIATION:",portfolio_std)
```

```
MEAN: 0.011631337567666477
STANDARD DEVIATION: 0.03273939175464404
```

b. (3% for mean, 3% for standard deviation)

In [4]:
```
weight=[0.05,0.3,0.4,0.25]
portfolio_mean=GE_Mean*weight[0]+JNJ_Mean*weight[1]+MCD_Mean*weight[2]+MRK_Mean*weight[3]

portfolio_variance=0
for i in range(4):
    for j in range(4):
        portfolio_variance=portfolio_variance+weight[i]*weight[j]*cov_mat[i][j]

portfolio_std=math.sqrt(portfolio_variance)
portfolio_std
print("MEAN:",portfolio_mean)
print("STANDARD DEVIATION:",portfolio_std)
```

```
MEAN: 0.011967230262297802
STANDARD DEVIATION: 0.030705750109862536
```

c. (3% for mean, 3% for standard deviation)

In [5]:
```
weight=[0.1,0.5,0.3,0.1]
portfolio_mean=GE_Mean*weight[0]+JNJ_Mean*weight[1]+MCD_Mean*weight[2]+MRK_Mean*weight[3]

portfolio_variance=0
for i in range(4):
    for j in range(4):
        portfolio_variance=portfolio_variance+weight[i]*weight[j]*cov_mat[i][j]

portfolio_std=math.sqrt(portfolio_variance)
portfolio_std
print("MEAN:",portfolio_mean)
print("STANDARD DEVIATION:",portfolio_std)
```

```
MEAN: 0.012135638250193528
STANDARD DEVIATION: 0.03151132959681168
```

d. (2% for portfolio chosen, 2% for explaination)

(c)

e. (2% for portfolio chosen, 2% for explaination)

(b)

Exercise 7.89

In [6]:
```
df_xr07tse = pd.read_excel('Xr07-TSE.xlsx')
display(df_xr07tse.head())

BMO_Mean=df_xr07tse["BMO"].mean()
MG_Mean=df_xr07tse["MG"].mean()
POW_Mean=df_xr07tse["POW"].mean()
RCLB_Mean=df_xr07tse["RCL.B"].mean()

cov_mat = np.cov(df_xr07tse[['BMO','MG','POW','RCL.B']].values,rowvar=False,ddof=0)
print(cov_mat)
```

	Year	Month	AEM	ABX	BBD.B	BCE	BMO	BNS	CM	CNR	...	L	MFC	MG	POT	
0	2011.0	January	-0.001126	0.082743	0.096491	-0.009623	0.072343	0.062699	0.077619	0.046693	...	0.010222	0.059382	-0.181632	0.012289	0.08
1	NaN	February	-0.056369	-0.017356	0.140800	-0.008416	0.016462	-0.008500	0.027613	0.034082	...	-0.011872	-0.064270	-0.023720	-0.044604	-0.02
2	NaN	March	0.022808	-0.041080	-0.011220	0.006814	-0.002158	-0.021706	-0.020215	0.003281	...	0.026512	-0.011059	0.046041	-0.064199	-0.00
3	NaN	April	-0.047643	-0.037575	-0.039046	0.099831	-0.003540	0.028948	-0.019534	0.032429	...	0.043129	0.024044	-0.030499	0.021890	-0.02
4	NaN	May	-0.026345	-0.056047	0.029630	-0.016954	-0.009205	-0.012378	-0.041051	0.022568	...	-0.058778	-0.011002	0.111940	0.008422	-0.02

5 rows × 25 columns

```
[[0.0011811  0.00036295 0.00070929 0.00033089]
 [0.00036295 0.00685094 0.00091688 0.00054053]
 [0.00070929 0.00091688 0.00264461 0.00047174]
 [0.00033089 0.00054053 0.00047174 0.00223601]]
```

a. (3% for mean, 3% for standard deviation)

In [7]:
```
weight=[0.25,0.25,0.25,0.25]
portfolio_mean=BMO_Mean*weight[0]+MG_Mean*weight[1]+POW_Mean*weight[2]+RCLB_Mean*weight[3]

portfolio_variance=0
for i in range(4):
    for j in range(4):
        portfolio_variance=portfolio_variance+weight[i]*weight[j]*cov_mat[i][j]

portfolio_std=math.sqrt(portfolio_variance)
portfolio_std
print("MEAN:",portfolio_mean)
print("STANDARD DEVIATION:",portfolio_std)
```

```
MEAN: 0.009574751246292187
STANDARD DEVIATION: 0.03497964911068031
```

b. (3% for mean, 3% for standard deviation)

In [8]:
```
weight=[0.2,0.6,0.1,0.1]
portfolio_mean=BMO_Mean*weight[0]+MG_Mean*weight[1]+POW_Mean*weight[2]+RCLB_Mean*weight[3]

portfolio_variance=0
for i in range(4):
    for j in range(4):
        portfolio_variance=portfolio_variance+weight[i]*weight[j]*cov_mat[i][j]

portfolio_std=math.sqrt(portfolio_variance)
portfolio_std
print("MEAN:",portfolio_mean)
print("STANDARD DEVIATION:",portfolio_std)
```

```
MEAN: 0.011539453105830171
STANDARD DEVIATION: 0.05362301174333776
```

c. (3% for mean, 3% for standard deviation)

In [9]:
```
weight=[0.1,0.2,0.3,0.4]
portfolio_mean=BMO_Mean*weight[0]+MG_Mean*weight[1]+POW_Mean*weight[2]+RCLB_Mean*weight[3]

portfolio_variance=0
for i in range(4):
    for j in range(4):
        portfolio_variance=portfolio_variance+weight[i]*weight[j]*cov_mat[i][j]

portfolio_std=math.sqrt(portfolio_variance)
portfolio_std
print("MEAN:",portfolio_mean)
print("STANDARD DEVIATION:",portfolio_std)
```

```
MEAN: 0.009343486987801556
STANDARD DEVIATION: 0.03570573487217066
```

d. (2% for portfolio chosen, 2% for explaination)

(b)

e. (2% for portfolio chosen, 2% for explaination)

(a)

Exercise 7.109

a. (2%)

In [10]:
```
n = 25          # Number of total bets
p = 0.3         # Probability of success
hh = stats.binom(n, p)

print("The probability that 10 or fewer customers chose the leading brand")
print("p(n =", n, ", p =", p, ", x<=10)=", hh.cdf(10))
```

```
The probability that 10 or fewer customers chose the leading brand
p(n = 25 , p = 0.3 , x<=10)= 0.9021999888782679
```

b. (2%)

In [11]:
```
print("The probability that 11 or more customers chose the leading brand")
print("p(n =", n, ", p =", p, ", x>=11)=", 1-hh.cdf(10))
```

```
The probability that 11 or more customers chose the leading brand
p(n = 25 , p = 0.3 , x>=11)= 0.09780001112173209
```

c. (2%)

In [12]:
```
print("The probability that exactly 10 customers chose the leading brand")
print("p(n =", n, ", p =", p, ", x=10)=", hh.pmf(10))
```

```
The probability that exactly 10 customers chose the leading brand
p(n = 25 , p = 0.3 , x=10)= 0.09163601238321294
```

Exercise 7.127

a. (2%)

In [13]:
```
n = 100         # Number of total bets
p = 0.53        # Probability of success
hh = stats.binom(n, p)

print("The probability that more than half say that Congress is doning a poor or bad job")
print("p(n =", n, ", p =", p, ", x>=50)=", 1-hh.cdf(50))
```

```
The probability that more than half say that Congress is doning a poor or bad job
p(n = 100 , p = 0.53 , x>=50)= 0.6921991139282936
```

b. (2%)

In [14]:
```
print("The probability that more than half say that Congress is doning a poor or bad job")
print("p(n =", n, ", p =", p, ", x>=60)=", 1-hh.cdf(60))
```

```
The probability that more than half say that Congress is doning a poor or bad job
p(n = 100 , p = 0.53 , x>=60)= 0.06592297187070095
```

c. (2%)

In [15]:
```
print("The expected number of American adults in the sample say that Congress is doing a poor or bad job")
print("mean of p(n =", n, ", p =", p, ")=", hh.mean())
```

```
The expected number of American adults in the sample say that Congress is doing a poor or bad job
mean of p(n = 100 , p = 0.53 )= 53.0
```

Exercise 7.133

a. (2%)

In [16]:
```
mu = 5
pp = stats.poisson(mu)

print("The probability that the site gets 10 or more hits in a week")
print("p(mu =", mu, ", x >= 10)=", 1-pp.cdf(9))
```

```
The probability that the site gets 10 or more hits in a week
p(mu = 5 , x >= 10)= 0.03182805730620486
```

b. (2%)

In [17]:
```
mu = 5*2
pp = stats.poisson(mu)

print("The probability that the site gets 20 or more hits in 2 week")
print("p(mu =", mu, ", x >= 20)=", 1-pp.cdf(19))
```

```
The probability that the site gets 20 or more hits in 2 week
p(mu = 10 , x >= 20)= 0.0034543419758568117
```

Exercise 7.141

a. (2%)

In [18]:
```
mu = 2
pp = stats.poisson(mu)

print("The probability that a golfer loses no golf balls")
print("p(mu =", mu, ", x = 0)=", pp.pmf(0))
```

```
The probability that a golfer loses no golf balls
p(mu = 2 , x = 0)= 0.1353352832366127
```

b. (2%)

In [19]:
```
print("The probability that a golfer loses 4 or more balls")
print("p(mu =", mu, ", x >= 4)=", 1-pp.cdf(3))
```

```
The probability that a golfer loses 4 or more balls
p(mu = 2 , x >= 4)= 0.14287653950145296
```

c. (2%)

In [20]:
```
print("The probability that a golfer loses 4 or more balls")
print("p(mu =", mu, ", x <= 2)=", pp.cdf(2))
```

```
The probability that a golfer loses 4 or more balls
p(mu = 2 , x <= 2)= 0.6766764161830634
```