Statistics Homework 06

B08705034 施芊羽 資管二

Chapter 7

In [1]:
```
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
import statistics
from matplotlib import pyplot as plt
%matplotlib inline
from numpy import random
from scipy.stats import binom
plt.rcParams["figure.dpi"] = 100
```

7.13

a.

Suppose P(n) represents the probability of a university graduate is offered n job(s) while n is from 0 to 3.(P(0) stands for no jobs is received.)

A graduate is offered fewer than two jobs: P(0) + P(1) = 5% + 43% = 48%

b.

A graduate is offered more than one job:P(2) + P(3) = 31% + 21% = 52%

7.27

In [2]:
```
df_c07_27 = pd.read_excel("Xr07-27.xlsx")
display(df_c07_27)
```

	x	P(x)
0	0	0.22
1	5	0.29
2	10	0.12

3	15	0.09
4	20	0.08
5	25	0.05
6	30	0.04
7	40	0.04
8	50	0.03
9	75	0.03
10	100	0.01

a.

In [12]:
```
P = 0
for i in range(df_c07_27["x"].shape[0]):
    if(df_c07_27["x"][i] >= 20):
        P = P + df_c07_27["P(x)"][i]

print("P(X >= 20) = ", P)
```

```
P(X >= 20) =  0.28
```

$P(X \geq 20) = .08 + .05 + .04 + .04 + .03 + .03 + .01 = .28$

b.

In [3]:
```
P = 0
for i in range(df_c07_27["x"].shape[0]):
    if(df_c07_27["x"][i] == 60):
        P = df_c07_27["P(x)"][i]
        break;

print("P(X = 20) = ", P)
```

```
P(X >= 20) =  0
```

There's no x = 60 in the given data, so P(X = 60) = 0.

c.

In [17]:
```
P = 0
for i in range(df_c07_27["x"].shape[0]):
    if(df_c07_27["x"][i] > 50):
        P = P + df_c07_27["P(x)"][i]


print("P(X > 50) = ", P)
```

```
P(X > 50) =  0.04
```

$P(X > 50) = .03 + .01 = .04$

d.

In [18]:
```
P = 0
for i in range(df_c07_27["x"].shape[0]):
    if(df_c07_27["x"][i] > 100):
        P = df_c07_27["P(x)"][i]
        break;

print("P(X > 100) = ", P)
```

```
P(X > 100) =  0
```

There's no probability of X > 100 in the given data, so P(X > 100) = 0

7.35

In [25]:
```
df_c07_35 = pd.read_excel("Xr07-35.xlsx")
display(df_c07_35)

mean = (df_c07_35["x"] * df_c07_35["P(x)"]).sum()
var = (pow((df_c07_35["x"] - mean), 2)  * df_c07_35["P(x)"]).sum()
std = np.sqrt(var)

print("Mean = ", mean)
print("Standard Deviation = ", std)
```

	x	P(x)
0	0	0.10
1	1	0.20
2	2	0.25
3	3	0.25
4	4	0.20

```
Mean =  2.25
Standard Deviation =  1.2599603168354152
```

The mean of the given data is 2.25 customers and the standard deviation is 1.26 customers(round it to the nearest hundredth).

7.55

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

7.59

The data on the textbook is wrong (the sum of the probability on the table ≠ 1). Therefore, the following data are all from the given excel dataset "Xr07-59.xlsx"

In [4]:
```
df_c07_59 = pd.read_excel("Xr07-59.xlsx")
display(df_c07_59)
```

	Unnamed: 0	Carbon Monoxide Detector: 0	Carbon Monoxide Detector: 1	Carbon Monoxide Detector: 2
0	Smoke Detector: 0	0.42	0.03	0.00
1	Smoke Detector: 1	0.15	0.07	0.01
2	Smoke Detector: 2	0.06	0.10	0.05
3	Smoke Detector: 3	0.05	0.04	0.02

a.

In [6]:
```
print("The proportion of homes have no carbon monoxide detectors and two
 smoke detectors is ",
     df_c07_59["Carbon Monoxide Detector: 0"][2])
```

```
The proportion of homes have no carbon monoxide detectors and two smoke
detectors is  0.06
```

The proportion of homes have no carbon monoxide detectors and two smoke detectors is 0.06.

b.

In [7]:
```
print("The proportion of homes have two carbon monoxide detectors and no
 smoke detectors is ",
     df_c07_59["Carbon Monoxide Detector: 2"][0])
```

```
The proportion of homes have two carbon monoxide detectors and no smoke
detectors is  0.0
```

The proportion of homes have two carbon monoxide detectors and no smoke detectors is 0.

c.

In [20]:
```
p = 0

for i in range(1 , 4, 1):
    p = p + df_c07_59["Carbon Monoxide Detector: 1"][i]
    p = p + df_c07_59["Carbon Monoxide Detector: 2"][i]


print("The proportion of homes have at least one carbon monoxide detecto
r and at least one smoke detector is ", p)
```

```
The proportion of homes have at least one carbon monoxide detector and a
t least one smoke detector is  0.29
```

The proportion of homes have at least one carbon monoxide detector and at least one smoke detector is .07 + .10 + .04 + .01 + .05 + .02 = .29

7.79

In the problem description, it doesn't tell which mean to use. Therefore, I choose to use the more simple one **arithematic mean**.

In [2]:
```python
df_c07_79 = pd.read_excel("Xr07-NYSE.xlsx")

def geomean(rate):
    r = rate
    geo_m = np.round(math.exp(np.log(r).mean()), 6)
    return geo_m
```

In [3]:
```python
## Mean of the four stocks
#GE = geomean(df_c07_79["GE"] + 1) - 1
#JNJ =  geomean(df_c07_79["JNJ"] + 1) - 1
#MCD =  geomean(df_c07_79["MCD"] + 1) - 1
#MRK = geomean(df_c07_79["MRK"] + 1) - 1
GE = df_c07_79["GE"].mean()
JNJ =  df_c07_79["JNJ"].mean()
MCD =  df_c07_79["MCD"].mean()
MRK = df_c07_79["MRK"].mean()

## Calculation for variation
var_GE = statistics.variance(df_c07_79["GE"])
var_JNJ = statistics.variance(df_c07_79["JNJ"])
var_MCD = statistics.variance(df_c07_79["MCD"])
var_MRK = statistics.variance(df_c07_79["MRK"])

## Display the statistics in a dataframe
df_c07_79_stat = pd.DataFrame()
df_c07_79_stat["GE"] = [GE, var_GE]
df_c07_79_stat["JNJ"] = [JNJ, var_JNJ]
df_c07_79_stat["MCD"] = [MCD, var_MCD]
df_c07_79_stat["MRK"] = [MRK, var_MRK]

## Calculation for the covariance
cov_mat1 = np.cov(df_c07_79[['GE', 'JNJ']].values, rowvar = False)
cov_mat2 = np.cov(df_c07_79[['GE', 'MCD']].values, rowvar = False)
cov_mat3 = np.cov(df_c07_79[['GE', 'MRK']].values, rowvar = False)
cov_mat4 = np.cov(df_c07_79[['MRK', 'JNJ']].values, rowvar = False)
cov_mat5 = np.cov(df_c07_79[['MCD', 'JNJ']].values, rowvar = False)
cov_mat6 = np.cov(df_c07_79[['MRK', 'MCD']].values, rowvar = False)

cov_ge_jnj = cov_mat1[0][1]
cov_ge_mcd = cov_mat2[0][1]
cov_ge_mrk = cov_mat3[0][1]
cov_mrk_jnj = cov_mat4[0][1]
cov_mcd_jnj = cov_mat5[0][1]
cov_mrk_mcd = cov_mat6[0][1]
```

ddof=0

```
display(df_c07_79_stat)
```

	GE	JNJ	MCD	MRK
0	0.010524	0.012687	0.012042	0.011273
1	0.003090	0.001388	0.001326	0.002157

a.

In [8]:

```
meanA = GE * 0.25 + JNJ * 0.25 + MCD * 0.25 + MRK * 0.25
stdA = np.sqrt(var_GE * 0.25 * 0.25 + var_JNJ * 0.25 * 0.25 + var_MCD * 0.25 * 0.25 + var_MRK * 0.25 * 0.25 +
     2 * 0.25 * 0.25 * cov_ge_jnj + 2 * 0.25 * 0.25 * cov_ge_mcd + 2 * 0.25 * 0.25 * cov_ge_mrk +
     2 * 0.25 * 0.25 * cov_mrk_jnj + 2 * 0.25 * 0.25 * cov_mcd_jnj + 2 * 0.25 * 0.25 * cov_mrk_mcd)

print("Mean of GE: 25% + JNJ: 25% + MCD: 25% + MRK: 25% = ", meanA)
print("Standard Deviation of GE: 25% + JNJ: 25% + MCD: 25% + MRK: 25% = ",  stdB)
```

-5

```
Mean of GE: 25% + JNJ: 25% + MCD: 25% + MRK: 25% =  0.011631337567666475
Standard Deviation of GE: 25% + JNJ: 25% + MCD: 25% + MRK: 25% =  0.030964874957958792
```

b.

In [7]:

```
meanB = GE * 0.05 + JNJ * 0.3 + MCD * 0.4 + MRK * 0.25
stdB = np.sqrt(var_GE * 0.05 * 0.05 + var_JNJ * 0.3 * 0.3
     + var_MCD * 0.4 * 0.4 + var_MRK * 0.25 * 0.25 +
     2 * 0.05 * 0.3 * cov_ge_jnj + 2 * 0.05 * 0.4 * cov_ge_mcd + 2 * 0.05 * 0.25 * cov_ge_mrk +
     2 * 0.25 * 0.3 * cov_mrk_jnj + 2 * 0.4 * 0.3 * cov_mcd_jnj + 2 * 0.25 * 0.4 * cov_mrk_mcd)
print("Mean of GE: 5% + JNJ: 30% + MCD: 40% + MRK: 25% = ", meanB)
print("Standard Deviation of GE: 5% + JNJ: 30% + MCD: 40% + MRK: 25% = ", stdB)
```

-3

```
Mean of GE: 5% + JNJ: 30% + MCD: 40% + MRK: 25% =  0.011967230262297799
Standard Deviation of GE: 5% + JNJ: 30% + MCD: 40% + MRK: 25% =  0.030964874957958792
```

c.

In [9]:

```
meanC = GE * 0.1 + JNJ * 0.5 + MCD * 0.3 + MRK * 0.1
stdC = np.sqrt(var_GE * 0.1 * 0.1 + var_JNJ * 0.5 * 0.5 + var_MCD * 0.3 * 0.3 + var_MRK * 0.1 * 0.1 +
     2 * 0.1 * 0.5 * cov_ge_jnj + 2 * 0.1 * 0.3 * cov_ge_mcd + 2 * 0.1 * 0.1 * cov_ge_mrk +
     2 * 0.3 * 0.5 * cov_mrk_jnj + 2 * 0.1 * 0.5 * cov_mcd_jnj + 2 * 0.1 * 0.3 * cov_mrk_mcd)

print("Mean of GE: 10% + JNJ: 50% + MCD: 30% + MRK: 10% = ", meanC)
print("Standard Deviation of GE: 10% + JNJ: 50% + MCD: 30% + MRK: 10% = ", stdC)
```

3

```
Mean of GE: 10% + JNJ: 50% + MCD: 30% + MRK: 10% =  0.012135638250193525
Standard Deviation of GE: 10% + JNJ: 50% + MCD: 30% + MRK: 10% =  0.0322
5932359442462
```

d.

A gambler will choose the **c.** portfolio since it has the highest mean of return rate.

e.

In [10]:
```
print("Covariance of GE: 25% + JNJ: 25% + MCD: 25% + MRK: 25% = ", stdA/
meanA)
print("Covariance of GE: 5% + JNJ: 30% + MCD: 40% + MRK: 25% = ", stdB/m
eanB)
print("Covariance of GE: 10% + JNJ: 50% + MCD: 30% + MRK: 10% = ", stdC/
meanC)
```

```
Covariance of GE: 25% + JNJ: 25% + MCD: 25% + MRK: 25% =  2.838510897632
9265
Covariance of GE: 5% + JNJ: 30% + MCD: 40% + MRK: 25% =  2.5874721451221
83
Covariance of GE: 10% + JNJ: 50% + MCD: 30% + MRK: 10% =  2.658230488529
121
```

A risk-averse investor will choose **b.** is because that even though its mean of return rate is lowest of all, its covariance is the smallest among all. As a risk-averse investor, he(or she) will rather choose the one that's stable while still profitable.

7.89

9

In the problem description, it doesn't tell which mean to use. Therefore, I choose to use the more simple one **arithematic mean**.

In [12]:
```
df_c07_89 = pd.read_excel("Xr07-TSE.xlsx")
```

In [13]:
```
## Mean of the four stocks
BMO = df_c07_89["BMO"].mean()
MG =  df_c07_89["MG"].mean()
POW =  df_c07_89["POW"].mean()
RCLB = df_c07_89["RCL.B"].mean()

## Calculation for variation
var_BMO = statistics.variance(df_c07_89["BMO"])
var_MG = statistics.variance(df_c07_89["MG"])
var_POW = statistics.variance(df_c07_89["POW"])
var_RCLB = statistics.variance(df_c07_89["RCL.B"])

## Display the statistics in a dataframe
df_c07_89_stat = pd.DataFrame()
df_c07_89_stat["BMO"] = [BMO, var_BMO]
df_c07_89_stat["MG"] = [MG, var_MG]
df_c07_89_stat["POW"] = [POW, var_POW]
df_c07_89_stat["RCL.B"] = [RCLB, var_RCLB]
```

```
## Calculation for the covariance
cov_mat1 = np.cov(df_c07_89[['BMO', 'MG']].values, rowvar = False)
cov_mat2 = np.cov(df_c07_89[['BMO', 'POW']].values, rowvar = False)
cov_mat3 = np.cov(df_c07_89[['BMO', 'RCL.B']].values, rowvar = False)
cov_mat4 = np.cov(df_c07_89[['RCL.B', 'MG']].values, rowvar = False)
cov_mat5 = np.cov(df_c07_89[['POW', 'MG']].values, rowvar = False)
cov_mat6 = np.cov(df_c07_89[['RCL.B', 'POW']].values, rowvar = False)

cov_bmo_mg = cov_mat1[0][1]
cov_bmo_pow = cov_mat2[0][1]
cov_bmo_rclb = cov_mat3[0][1]
cov_rclb_mg = cov_mat4[0][1]
cov_pow_mg = cov_mat5[0][1]
cov_rclb_pow = cov_mat6[0][1]

display(df_c07_89_stat)
```

ddof=0

	BMO	MG	POW	RCL.B
0	0.008698	0.013680	0.006309	0.009613
1	0.001201	0.006967	0.002689	0.002274

a.

In [15]:
```
meanA = BMO * 0.25 + MG * 0.25 + POW * 0.25 + RCLB * 0.25
stdA = np.sqrt(var_BMO * 0.25 * 0.25 + var_MG * 0.25 * 0.25 + var_POW *
0.25 * 0.25 + var_RCLB * 0.25 * 0.25 +
     2 * 0.25 * 0.25 * cov_bmo_mg + 2 * 0.25 * 0.25 * cov_bmo_pow + 2 *
0.25 * 0.25 * cov_bmo_rclb + 2 * 0.25 * 0.25 * cov_rclb_mg +
     2 * 0.25 * 0.25 * cov_pow_mg + 2 * 0.25 * 0.25 * cov_rclb_pow)

print("Mean of BMO: 25% + MG: 25% + POW: 25% + RCL.B: 25% = ", meanA)
print("Standard Deviation of BMO: 25% + MG: 25% + POW: 25% + RCL.B: 25%
= ",  stdA)
```

-3

```
Mean of BMO: 25% + MG: 25% + POW: 25% + RCL.B: 25% =  0.0095747512462921
86
Standard Deviation of BMO: 25% + MG: 25% + POW: 25% + RCL.B: 25% =  0.03
527484125644567
```

b.

In [16]:
```
meanB = BMO * 0.2 + MG * 0.6 + POW * 0.1 + RCLB * 0.1
stdB = np.sqrt(var_BMO * 0.2 * 0.2 + var_MG * 0.6 * 0.6 + var_POW * 0.1
* 0.1 + var_RCLB * 0.1 * 0.1 +
     2 * 0.2 * 0.6 * cov_bmo_mg + 2 * 0.2 * 0.1 * cov_bmo_pow + 2 * 0.2
* 0.1 * cov_bmo_rclb + 2 * 0.1 * 0.6 * cov_rclb_mg +
     2 * 0.1 * 0.6 * cov_pow_mg + 2 * 0.1 * 0.1 * cov_rclb_pow)

print("Mean of BMO: 20% + MG: 60% + POW: 10% + RCL.B: 10% = ", meanB)
print("Standard Deviation of BMO: 20% + MG: 60% + POW: 10% + RCL.B: 10%
= ", stdB)
```

-3

```
Mean of BMO: 20% + MG: 60% + POW: 10% + RCL.B: 10% =  0.0115394531058301
66
Standard Deviation of BMO: 20% + MG: 60% + POW: 10% + RCL.B: 10% =  0.05
40755346331138
```

c.

In [17]:
```
meanC = BMO * 0.1 + MG * 0.2 + POW * 0.3 + RCLB * 0.4
stdC = np.sqrt(var_BMO * 0.1 * 0.1 + var_MG * 0.2 * 0.2 + var_POW * 0.3
* 0.3 + var_RCLB * 0.4 * 0.4 +
     2 * 0.1 * 0.2 * cov_bmo_mg + 2 * 0.1 * 0.3 * cov_bmo_pow + 2 * 0.1
* 0.4 * cov_bmo_rclb + 2 * 0.4 * 0.2 * cov_rclb_mg +
     2 * 0.3 * 0.2 * cov_pow_mg + 2 * 0.4 * 0.3 * cov_rclb_pow)

print("Mean of BMO: 10% + MG: 20% + POW: 30% + RCL.B: 40% = ", meanC)
print("Standard Deviation of BMO: 10% + MG: 20% + POW: 30% + RCL.B: 40%
= ",  stdC)
```

```
Mean of BMO: 10% + MG: 20% + POW: 30% + RCL.B: 40% =  0.0093434869878015
54
Standard Deviation of BMO: 10% + MG: 20% + POW: 30% + RCL.B: 40% =  0.03
600705443257263
```

d.

A gambler will choose the **b.** portfolio since it has the highest mean of return rate.

e.

In [18]:
```
print("Covariance of BMO: 25% + MG: 25% + POW: 25% + RCL.B: 25% = ", std
A/meanA)
print("Covariance of BMO: 20% + MG: 60% + POW: 10% + RCL.B: 10% = ", std
B/meanB)
print("Covariance of BMO: 10% + MG: 20% + POW: 30% + RCL.B: 40% = ", std
C/meanC)
```

```
Covariance of BMO: 25% + MG: 25% + POW: 25% + RCL.B: 25% =  3.6841522405
196505
Covariance of BMO: 20% + MG: 60% + POW: 10% + RCL.B: 10% =  4.6861436271
87393
Covariance of BMO: 10% + MG: 20% + POW: 30% + RCL.B: 40% =  3.8537062747
111284
```

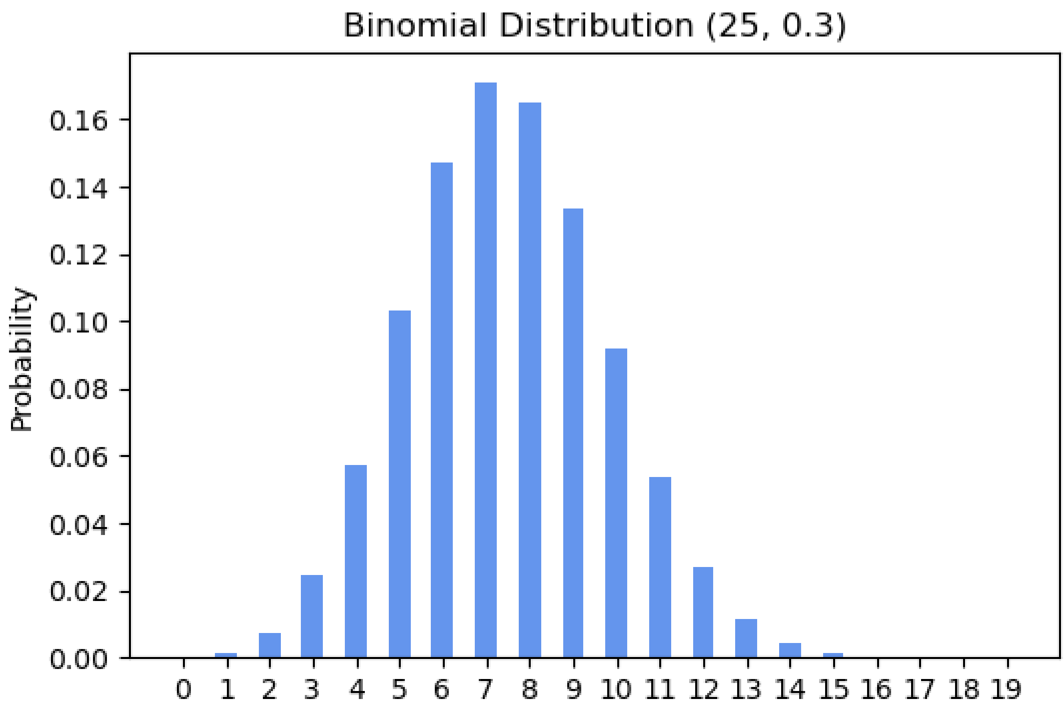
A risk-averse investor will choose **a.** is because that even though its mean of return rate is not the highest, its covariance is the smallest of the three. As a risk-averse investor, he(or she) will rather choose the one that's stable while profitable.

7.109

All answers of this problem are rounded to the nearest thousandth

In [20]:
```
def factor(n):
    ans = 1
    for i in range(1 , n + 1):
        ans = ans * i
    return ans
```

```python
x = np.arange(20)
hh = stats.binom(25, 0.3)
y = hh.pmf(x)
fig_hh, ax_hh = plt.subplots()
rect_hh = ax_hh.bar(x, y, width=0.5, bottom=None, align='center', color=
'cornflowerblue')
plt.ylabel('Probability')
plt.title('Binomial Distribution (25, 0.3)')
plt.xticks(x)
plt.show()
```



a.

In [47]:
```python
p = 0
for i in range(11):
    p = p + np.power(0.3 , i) * np.power((1 - 0.3) , 25 - i) * factor(25
) /factor(i)/factor(25 - i)

print("The probability that 10 or fewer customers chose the leading bran
d is ", p)
```

```
The probability that 10 or fewer customers chose the leading brand is  0
.9021999888782667
```

The probability that 10 or fewer customers chose the leading brand is

$\sum_{10i=1} 25!i!(25-i)!(0.3)^{i}(1-0.3)^{25-i}$ 0.902.

b.

In [49]:
```python
p = 0
for i in range(11, 26):
    p = p + np.power(0.3 , i) * np.power((1 - 0.3) , 25 - i) * factor(25
) /factor(i)/factor(25 - i)
```

```
print("The probability that  11  or more customers chose the leading brand is", p)
```

```
The probability that  11  or more customers chose the leading brand is 0.09780001112173187
```

The probability that 11 or more customers chose the leading brand is

$\sum_{i=11}^{25} \frac{25!}{i!(25-i)!}(0.3)^{i}(1-0.3)^{25-i}$ 0.098.

c.

In [21]:
```
p = 0
p = p + np.power(0.3 , 10) * np.power((1 - 0.3) , 25 - 10) * factor(25)/factor(10)/factor(25 - 10)
print("The probability that 10 customers chose the leading brand is", p)
```

```
The probability that 10 customers chose the leading brand is 0.09163601238321353
```

The probability that 10 customers chose the leading brand is

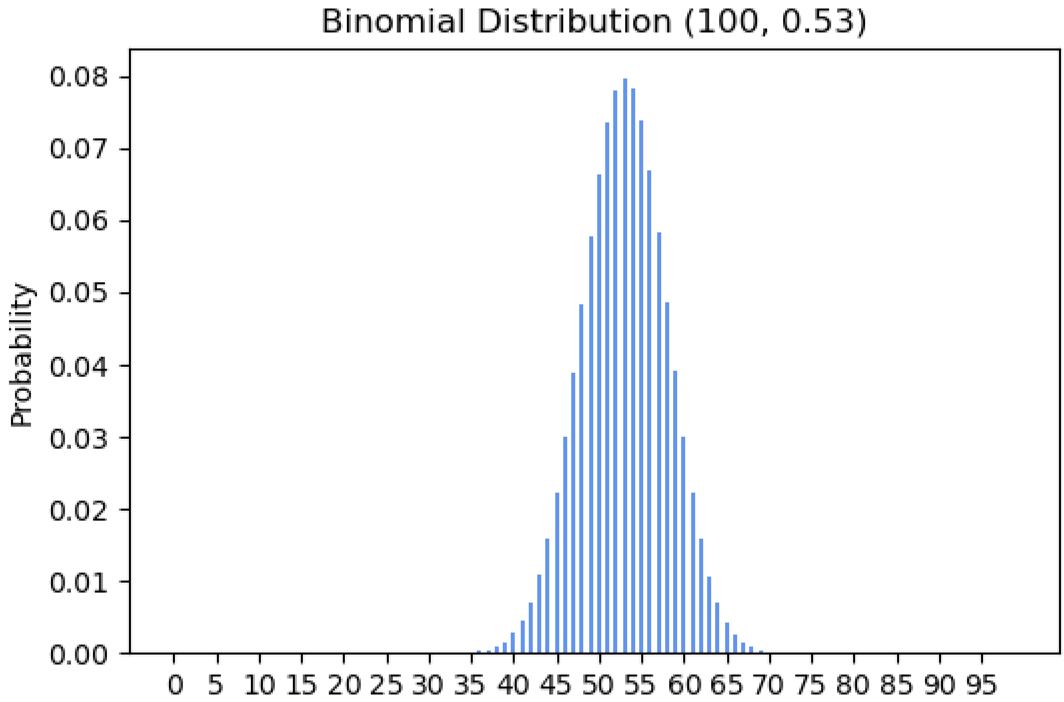
$\frac{25!}{10!(25-10)!}(0.3)^{10}(1-0.3)^{(25-10)}$ 0.0916

7.127

10

All answers of this problem are round to the nearest thousandth

In [119]:
```
x = np.arange(100)
hh = stats.binom(100, 0.53)
y = hh.pmf(x)
fig_hh, ax_hh = plt.subplots()
rect_hh = ax_hh.bar(x, y, width=0.5, bottom=None, align='center', color='cornflowerblue')
plt.ylabel('Probability')
plt.title('Binomial Distribution (100, 0.53)')
plt.xticks(np.arange(0, 100, 5), np.arange(0, 100, 5), fontsize=10)
plt.show()
```



a.

In [57]:
```
p = 0
for i in range(51, 101):
    p = p + np.power(0.53 , i) * np.power((1 - 0.53) ,  100 - i) * facto
r(100) /factor(i)/factor(100 - i)

print("The probability that more than half say that Congress is doing a
poor or bad job is", p)
```

```
The probability that more than half say that Congress is doing a poor or
 bad job is 0.6921991139282937
```

The probability that more than half say that Congress is doing a poor or bad job is

$\Sigma_{100i=51} 100!i!(100-i)!(0.53)^{i}(1-0.53)^{(100-i)}$ 0.692

b.

In [58]:
```
p = 0
for i in range(61, 101):
    p = p + np.power(0.53 , i) * np.power((1 - 0.53) ,  100 - i) * facto
r(100) /factor(i)/factor(100 - i)

print("The probability that more than 60% say that Congress is doing a p
oor or bad job is", p)
```

```
The probability that more than 60% say that Congress is doing a poor or
bad job is 0.06592297187070088
```

The probability that more than 60% say that Congress is doing a poor or bad job is

$\Sigma_{100i=61} 100!i!(100-i)!(0.53)^{i}(1-0.53)^{(100-i)}$ 0.066

c.

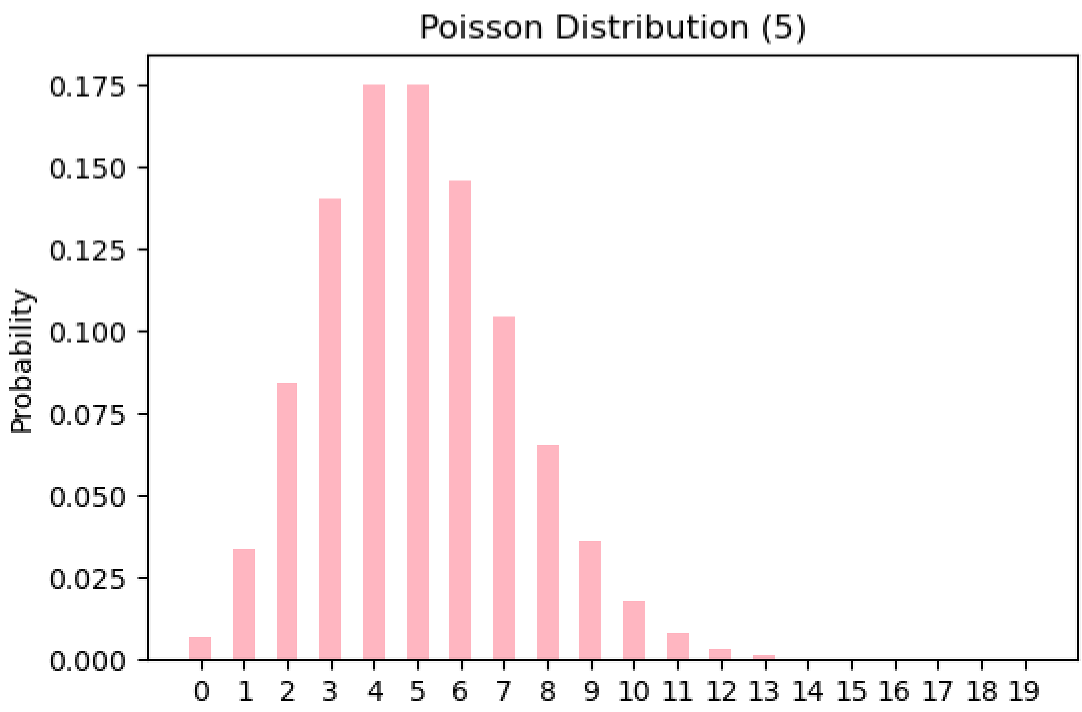
Since it's a Binomial Distribution, we can know that the expected number of American adults in the sample would say that Congress is doing a poor or bad job is np while n represents for the number of trials and p represents the succeess rate(in this case, is that 53% believed that Congress is doing a poor or bad job). Hence the expected number of American adults in the sample would say that Congress is doing a poor or bad job is 100 × 53% = 53.

Therefore, it's expected that 53 adults out of the 100 will say that Congress is doing a poor or bad job.

7.133

All the answers to this quesion are rounded off to the 6th decimal place.

In [121]:
```
x = np.arange(20)
pp = stats.poisson(5)
y = pp.pmf(x)
fig_pp, ax_pp = plt.subplots()
rect_pp = ax_pp.bar(x, y, width=0.5, bottom=None, align='center', color=
'lightpink')
plt.ylabel('Probability')
plt.title('Poisson Distribution (5)')
plt.xticks(x)
plt.show()
```



a.

In [61]:
```
pp = stats.poisson(5)
print("p(mu = 5,", "x>=", 10, ")=", 1 - pp.cdf(9))
```

```
p(mu = 5, x>= 10 )= 0.03182805730620486
```

The probability that the site gets 10 or more hits in a week is 0.031828 after using solving it through the `poisson` functions in python.

b.

In [59]:
```
pp = stats.poisson(10)
print("p(mu = 10,", "x>=", 20, ")=", 1 - pp.cdf(19))
```

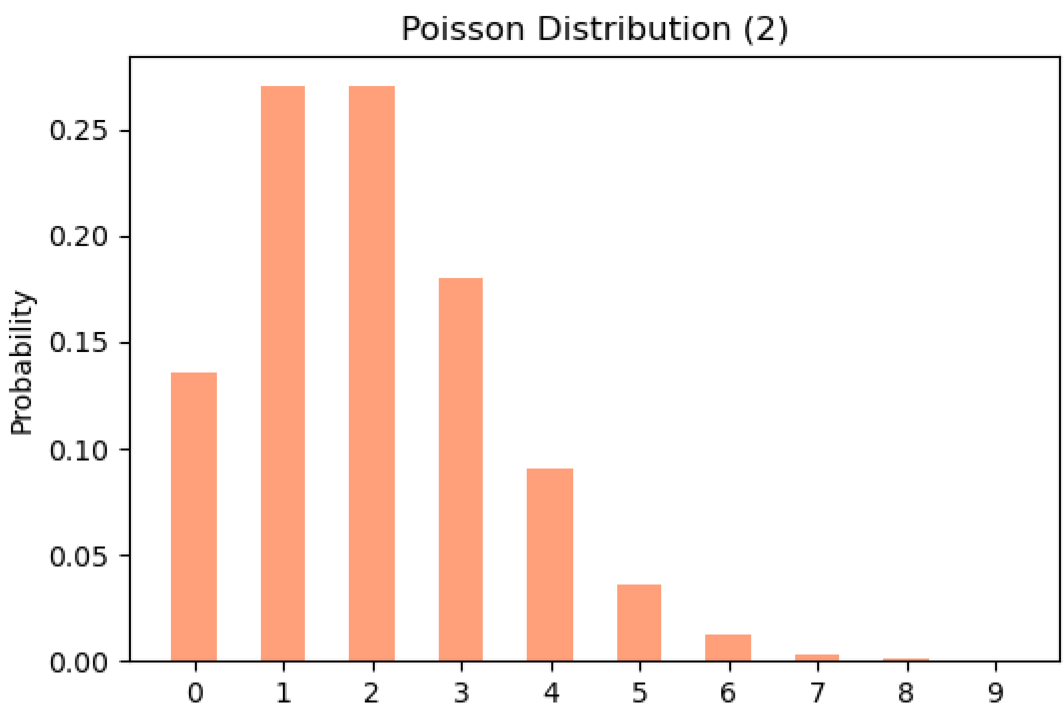
```
p(mu = 10, x>= 20 )= 0.0034543419758568117
```

The probability that the site gets 20 or more hits in two weeks is 0.003454 after using solving it through the poisson functions in python.

7.141

All the answers to this quesion are rounded off to the 6th decimal place.

In [122]:
```
x = np.arange(10)
pp = stats.poisson(2)
y = pp.pmf(x)
fig_pp, ax_pp = plt.subplots()
rect_pp = ax_pp.bar(x, y, width=0.5, bottom=None, align='center', color=
 'lightsalmon')
plt.ylabel('Probability')
plt.title('Poisson Distribution (2)')
plt.xticks(x)
plt.show()
```



a.

In [70]:
```
pp = stats.poisson(2)
print("p(mu = 2,", "x = ", 0, ")=", pp.pmf(0))
```

```
p(mu = 2, x =  0 )= 0.1353352832366127
```

The probability that a golfer loses no golf balls is 0.135335.

b.

In the description, I don't know what's the meaning of "A golfer **loses 4 or more no golf balls**"; therefore, I predicted that the 'no' is accidentally typed. Hence, I answered this question with the thinking that the problem is asking "A golfer **loses 4 or more golf balls**".

In [81]:
```
pp = stats.poisson(2)
print("p(mu = 2,", "x >= ", 4, ")=", 1 - pp.cdf(3))
```

```
p(mu = 2, x >=  4 )= 0.14287653950145296
```

The probability that a golfer **loses 4 or more golf balls** is 0.142877.

c.

In [82]:
```
pp = stats.poisson(2)
print("p(mu = 2,", "x <= ", 2, ")=", pp.cdf(2))
```

```
p(mu = 2, x <=  2 )= 0.6766764161830634
```

The probability that a golfer loses 2 or fewer golf balls is 0.676676.