# 1 빅데이터 기반 AI 응용 솔루션 개발자 전문과정

## 1.1 교과목명 : 통계

- 평가일 : 22.09.08
- 성명 :양주희
- 점수 :70

Q1. df에서 mathematics 점수의 평균값, 중앙값, 최빈값, 분산, 표준편차, 범위, IQR을 구하세요.

In [10]:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats, integrate
from scipy.optimize import minimize_scalar

%precision
%matplotlib inline
```

In [1]:

```
import numpy as np
import pandas as pd
df = pd.read_csv('data/ch2_scores_em.csv',
                 index_col='student number')
df.head()
```

| student number | english | mathematics |
|---|---|---|
| 1 | 42 | 65 |
| 2 | 69 | 80 |
| 3 | 56 | 63 |
| 4 | 41 | 63 |
| 5 | 57 | 76 |

In [2]:

```
#평균
df.mathematics.mean()
```

```
78.88
```

In [3]:

```
#중앙값
df.mathematics.median()
```

```
80.0
```

In [6]:

```
#평균값, 중앙값, 최빈값, 분산, 표준편차, 범위, IQR
print(df.mathematics.mode())
print(df.mathematics.var(ddof= 0))
print(df.mathematics.var(ddof= 1))
print(np.std(df.mathematics, ddof=0))
print(np.std(df.mathematics, ddof=1))
print(np.max(df.mathematics) - np.min(df.mathematics))
```

```
0    77
1    82
2    84
Name: mathematics, dtype: int64
69.38560000000001
70.80163265306123
8.329801918413187
8.414370603500968
37
```

In [5]:

```
scores_Q1 = np.percentile(df.mathematics, 25)
scores_Q3 = np.percentile(df.mathematics, 75)
scores_IQR = scores_Q3 - scores_Q1
scores_IQR
```

```
8.0
```

Q2. df.english를 표준화한 후 배열로 변환하여 처음 5개 원소를 출력하세요.

In [7]:

```
score = df.english
type(score)
```

```
pandas.core.series.Series
```

In [9]:

```
np.mean(z), np.std(z, ddof=0)
```

```
(-2.4424906541753446e-16, 1.0)
```

In [11]:

```
z = (score - np.mean(score)) / np.std(score)
z[:5]
```

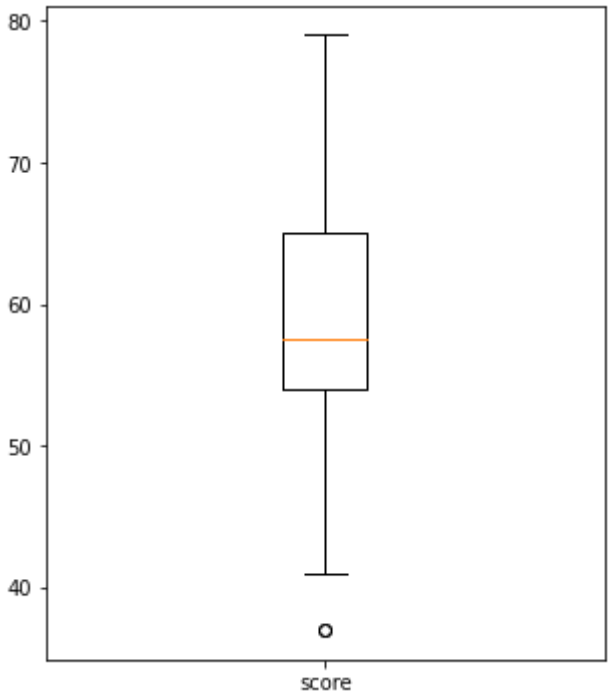
```
student number
1   -1.688430
2    1.094696
3   -0.245327
4   -1.791509
5   -0.142249
Name: english, dtype: float64
```

Q3. score에 대하여 다음사항을 수행하세요.

- 상자그림으로 시각화하여 이상치 여부를 탐색
- 이상치 값 및 인덱스 출력
- 이상치 삭제
- 상자그림으로 시각화하여 이상치 제거 여부 재확인.

In [12]:

```
#상자그림으로 시각화하여 이상치 여부를 탐색
fig = plt.figure(figsize=(5,6))
ax = fig.add_subplot(111)
ax.boxplot(score, labels=['score'])

plt.show()
```




In [13]:

```
scores_Q1 = np.percentile(score, 25)
scores_Q3 = np.percentile(score, 75)
scores_IQR = scores_Q3 - scores_Q1
scores_IQR
```

11.0

In [18]:

```
lower_whisker =  scores_Q1 - 1.5 * scores_IQR
```

In [19]:

```
#이상치 값 및 인덱스 출력
print(score[score < lower_whisker])
```

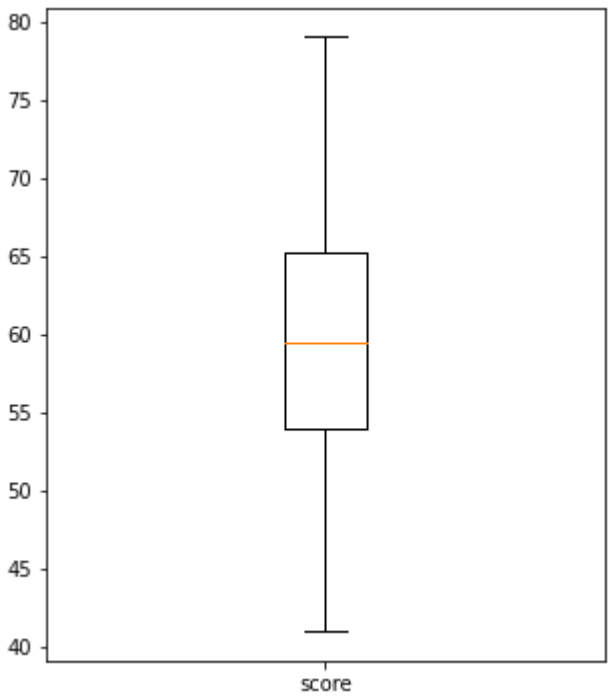
```
student number
20    37
35    37
Name: english, dtype: int64
```

In [20]:

```
#이상치 삭제
score = score[score >= lower_whisker]
```

In [21]:

```
#상자그림으로 시각화하여 이상치 제거 여부 재확인.
fig = plt.figure(figsize=(5,6))
ax = fig.add_subplot(111)
ax.boxplot(score, labels=['score'])

plt.show()
```




Q4. 아래 scores_df에 대해서 아래사항을 수행하세요

- scores_df.english와 scores_df.mathematics에 대한 공분산을 소수점 2째자리까지 출력

- scores_df.english와 scores_df.mathematics에 대한 상관계수를 소수점 2째자리까지 출력
- 두개 변수의 상관관계와 회귀직선을 시각화(회귀직선 포함 및 미포함 비교하여 1행 2열로 출력)
- 두개 변수의 상관관계를 히트맵으로 시각화(칼러바 포함)

In [23]:

```
import numpy as np
import pandas as pd
df = pd.read_csv('data/ch2_scores_em.csv',
                 index_col='student number')
en_scores = np.array(df['english'])[:10]
ma_scores = np.array(df['mathematics'])[:10]

scores_df = pd.DataFrame({'english':en_scores,
                          'mathematics':ma_scores},
                         index=pd.Index(['A', 'B', 'C', 'D', 'E',
                                         'F', 'G', 'H', 'I', 'J'],
                                        name='student'))
scores_df.head()
```

| student | english | mathematics |
|---|---|---|
| A | 42 | 65 |
| B | 69 | 80 |
| C | 56 | 63 |
| D | 41 | 63 |
| E | 57 | 76 |

In [27]:

```
# scores_df.english와 scores_df.mathematics에 대한 공분산을 소수점
cov_mat = np.cov(scores_df.english, scores_df.mathematics, ddof=0)
print(f'{cov_mat[0,1]:.2f}', f'{cov_mat[1,0]:.2f}')
```

```
62.80 62.80
```

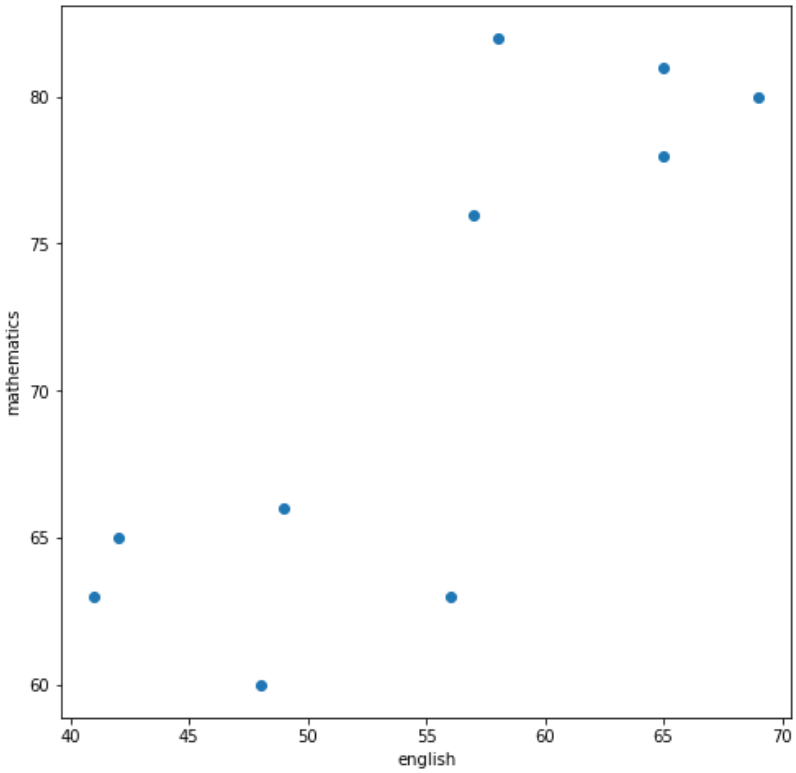
In [28]:

```
#scores_df.english와 scores_df.mathematics에 대한 상관계수를 소수점
cov = np.corrcoef(scores_df.english, scores_df.mathematics)
print(f'{cov[0,1]:.2f}', f'{cov[1,0]:.2f}')
```

```
0.82 0.82
```

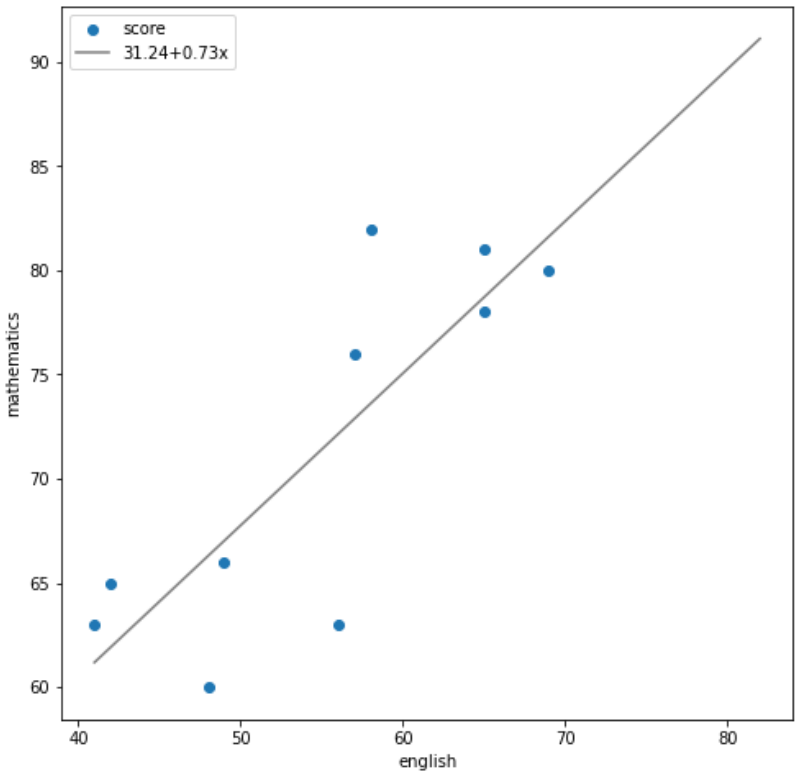
In [31]:

```
#두개 변수의 상관관계와 회귀직선을 시각화(회귀직선 포함 및 미포함

fig = plt.figure(figsize=(8,8))
ax = fig.add_subplot(111)

#산점도
ax.scatter(en_scores, ma_scores)
ax.set_xlabel('english')
ax.set_ylabel('mathematics')

plt.show()
```



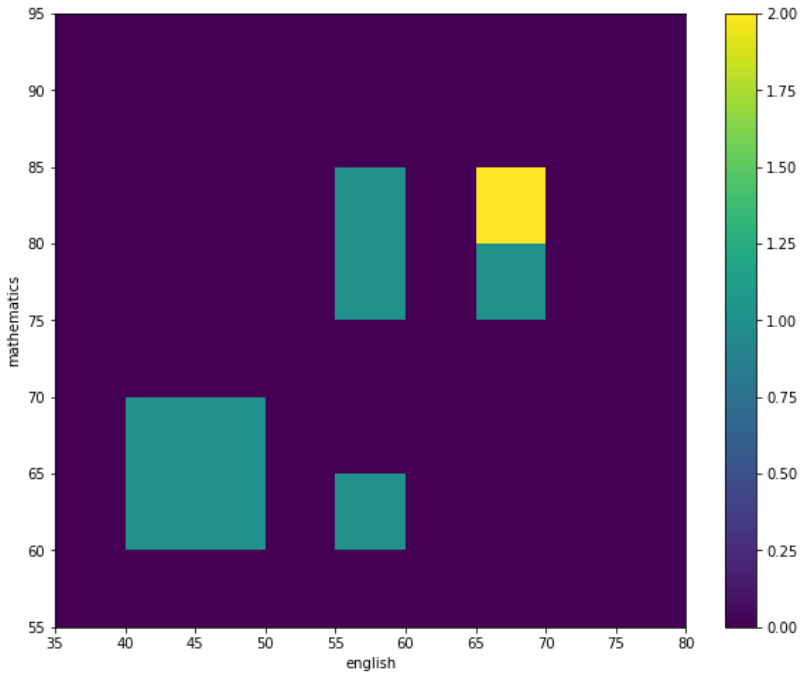
In [32]:

```
# 계수 β_0와 β_1를 구한다
poly_fit = np.polyfit(en_scores, ma_scores, 1)
# β_0+β_1 x를 반환하는 함수를 작성
poly_1d = np.poly1d(poly_fit)
# 직선을 그리기 위해 x좌표를 생성
xs = np.linspace(en_scores.min(), ma_scores.max())
# xs에 대응하는 y좌표를 구한다
ys = poly_1d(xs)

fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(111)
ax.set_xlabel('english')
ax.set_ylabel('mathematics')
ax.scatter(en_scores, ma_scores, label='score')

#plot - label로 범례 지정
ax.plot(xs, ys, color='gray',
        label=f'{poly_fit[1]:.2f}+{poly_fit[0]:.2f}x')
# 범례의 표시
ax.legend(loc='upper left')

plt.show()
```



In [30]:

```
#두개 변수의 상관관계를 히트맵으로 시각화(칼러바 포함)
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111)

c = ax.hist2d(en_scores,ma_scores, bins=[9,8], range=[(35, 80), (5

ax.set_xlabel('english')
ax.set_ylabel('mathematics')
#눈금 설정
ax.set_xticks(c[1]) #[35.000 40.000 45.000 50.000 55.000 60.000 65
ax.set_yticks(c[2]) #[55.000 60.000 65.000 70.000 75.000 80.000 85

#컬러바 표시
fig.colorbar(c[3], ax=ax)
plt.show()
```




Q5. 아래 scores는 전교생의 시험점수이다. 무작위추출로 표본 크기가 20인 표본을 추출하여 표본평균을 계산하는 작업을 10000번 수행해서 그 결과를 히스토그램으로 그려 표본평균이 어떻게 분포되는지 시각화를 수행하세요.

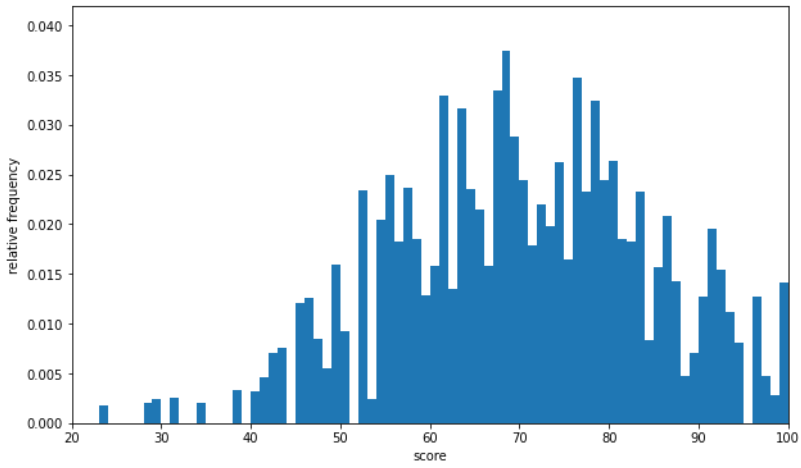
In [34]:

```
df = pd.read_csv('data/ch4_scores400.csv')
scores = np.array(df['score'])
scores[:10]
```

```
array([76, 55, 80, 80, 74, 61, 81, 76, 23, 80], dty
pe=int64)
```

In [35]:

```
sample = np.random.choice(scores, 10000)

fig = plt.figure(figsize=(10,6))
ax = fig.add_subplot(111)

ax.hist(sample, bins=100, range=(0,100), density=True)
ax.set_xlim(20, 100)
ax.set_ylim(0, 0.042)

ax.set_xlabel('score')
ax.set_ylabel('relative frequency')
plt.show()
```




Q6. Bern(0.5)을 따르는 확률변수 X에 대하여 기댓값과 분산을 계산하세요.

In [36]:

```
linestyles = ['-', '--', ':']

def E(X, g=lambda x: x):
    x_set, f = X
    return np.sum([g(x_k) * f(x_k) for x_k in x_set])

def V(X, g=lambda x: x):
    x_set, f = X
    mean = E(X, g)
    return np.sum([(g(x_k)-mean)**2 * f(x_k) for x_k in x_set])

def check_prob(X): # 확률변수를 인수로 가지며 기댓값과 분산 계산 길
    x_set, f = X

    prob = np.array([f(x_k) for x_k in x_set])
    assert np.all(prob >= 0), 'minus probability'
    prob_sum = np.round(np.sum(prob), 6)
    assert prob_sum == 1, f'sum of probability{prob_sum}'

    print(f'expected value {E(X):.4}')
    print(f'variance {(V(X)):.4}')

def plot_prob(X): # 확률변수를 인수로 가지며 그 확률변수의 확률함수
    x_set, f = X
    prob = np.array([f(x_k) for x_k in x_set])

    fig = plt.figure(figsize=(6, 4))
    ax = fig.add_subplot(111)
    ax.bar(x_set, prob, label='prob')
    ax.vlines(E(X), 0, 1, label='mean')
    ax.set_xticks(np.append(x_set, E(X))) #차원이 같은 두 배열 이어
    ax.set_ylim(0, prob.max()*1.2)
    ax.legend()

    plt.show()
```

In [37]:

```
def Bern(p):
    x_set = np.array([0, 1])
    def f(x):
        if x in x_set:
            return p**x*(1-p)**(1-x)
        else:
            return 0
    return x_set, f
```

In [38]:

```
p = 0.5
X = Bern(p)
```

In [39]:

```
check_prob(X)
```

```
expected value 0.5
variance 0.25
```

Q7. Bin(10,0.5)을 따르는 확률변수 X에 대하여 기댓값과 분산을 계산하세요.

In [41]:

```
from scipy.special import comb

def Bin(n, p):
    x_set = np.arange(n + 1)

    def f(x):
        if x in x_set:
            return comb(n, x) *p**x * (1-p)**(n-x)
        else:
            return 0
    return x_set, f
```

In [42]:

```
n = 10
p = 0.5
X =Bin(n, p)
```

In [43]:

```
check_prob(X)
```

```
expected value 5.0
variance 2.5
```

Q8. Poi(2)을 따른 확률변수 X에 대하여 기댓값과 분산을 계산하세요.

In [44]:

```
from scipy.special import factorial

def Poi(lam):
    x_set = np.arange(20) #구현 편의상 x 범위 한정
    def f(x):
        if x in x_set:
            return np.power(lam, x) / factorial(x) * np.exp(-lam)
    return x_set, f
```

In [45]:

```
lam = 2
X = Poi(lam)

#기댓값 , 분산
check_prob(X)
```

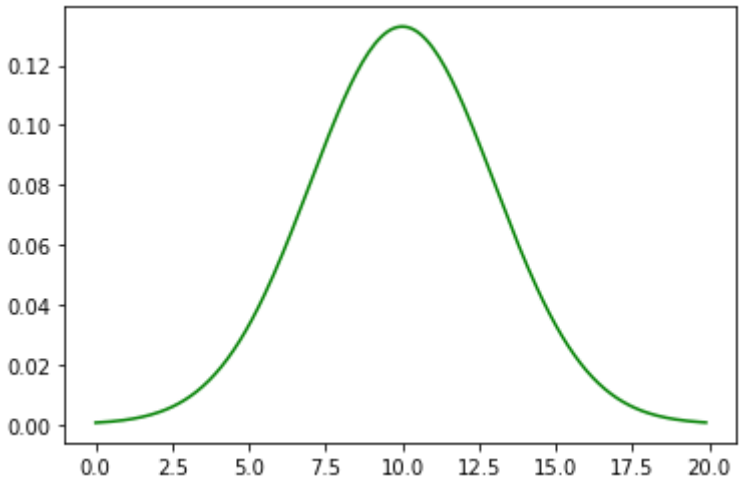
```
expected value 2.0
variance 2.0
```

Q9. 평균이 10, 표준편차가 3인 정규분포의 확률밀도함수를 그래프로 표현하세요.

In [50]:

```
x_plot = np.arange(0, 20, 0.1)
plt.plot(x_plot, stats.norm.pdf(x=x_plot, loc=10, scale=3), color=
```

```
[<matplotlib.lines.Line2D at 0x25469294100>]
```

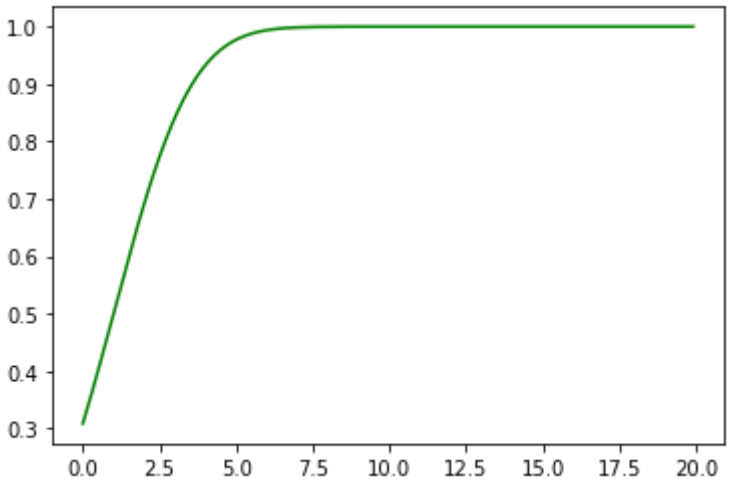



Q10. 평균이 1, 표준편차가 2인 정규분포의 누적분포함수를 그래프로 표현하세요.

In [51]:

```
x_plot = np.arange(0, 20, 0.1)
plt.plot(x_plot, stats.norm.cdf(loc=1, scale=2, x=x_plot), color=
```

```
[<matplotlib.lines.Line2D at 0x254692fb8b0>]
```




Q11. "5_2_fm.csv"을 df1으로 불러와서 다음사항을 수행하세요.

- df1을 df2 이름으로 복사한 후 df2의 species의 A, B를 C,D로 변경하세요.
- df의 length를 species가 C인 것은 2배로 d인 것은 3배로 변경하여 df1과 df2를 행방향으로 결합, df 생성
- df를 species 칼럼을 기준으로 그룹별 평균과 표준편차를 산출

In [9]:

```
import pandas as pd
import numpy as np

df1 = pd.read_csv("5_2_fm.csv")
df1
```

| | species | length |
|---|---|---|
| 0 | A | 2 |
| 1 | A | 3 |
| 2 | A | 4 |
| 3 | B | 6 |
| 4 | B | 8 |
| 5 | B | 10 |

In [10]:

```
df2 = df1.copy()
df2.loc[df2['species'] == 'A','species'] = 'C'
df2.loc[df2['species'] == 'B','species'] = 'D'
df2
# df2.loc[df['species'] == 'C', 'length'] = df['length'] * 2
```

| | species | length |
|---|---|---|
| 0 | C | 2 |
| 1 | C | 3 |
| 2 | C | 4 |
| 3 | D | 6 |
| 4 | D | 8 |
| 5 | D | 10 |

In [13]:

```
df2.loc[df2['species'] == 'C', 'length'] = df2['length'] * 2
df2.loc[df2['species'] == 'D', 'length'] = df2['length'] * 3
df2
```

| | species | length |
|---|---|---|
| 0 | C | 4 |
| 1 | C | 6 |
| 2 | C | 8 |
| 3 | D | 18 |
| 4 | D | 24 |
| 5 | D | 30 |

In [14]:

```
df = pd.concat([df1, df2], axis=0)
df
```

| | species | length |
|---|---|---|
| 0 | A | 2 |
| 1 | A | 3 |
| 2 | A | 4 |
| 3 | B | 6 |
| 4 | B | 8 |
| 5 | B | 10 |
| 0 | C | 4 |
| 1 | C | 6 |
| 2 | C | 8 |
| 3 | D | 18 |
| 4 | D | 24 |
| 5 | D | 30 |

In [15]:

```
function_list = ['size', 'std', 'mean', 'min', 'max', 'sum'] #size
result = df.groupby('species').agg(['mean', 'std'])
result
```

| | length | |
|---|---|---|
| | mean | std |
| species | | |
| A | 3.0 | 1.0 |
| B | 8.0 | 2.0 |
| C | 6.0 | 2.0 |
| D | 24.0 | 6.0 |

Q12. "./dataset/5_2_shoes.csv" 을 데이터프레임으로 불러와서 아래작업을 수행하세요.

- 4행 3열을 복사 후 추가하여 8행 3열로 작성
- 피봇을 이용해서 교차분석표 작성(values='sales',aggfunc='sum', index='store', columns = 'color')
- 독립성 검정을 수행(보너스 문제)

In [19]:

```
import pandas as pd
shoes = pd.read_csv("5_2_shoes.csv")
shoes1=shoes.copy()
shoes1
```

| | store | color | sales |
|---|---|---|---|
| 0 | tokyo | blue | 10 |
| 1 | tokyo | red | 15 |
| 2 | osaka | blue | 13 |
| 3 | osaka | red | 9 |

In [20]:

```
shoes = pd.concat([shoes, shoes1], axis=0)
shoes
```

| | store | color | sales |
|---|---|---|---|
| 0 | tokyo | blue | 10 |
| 1 | tokyo | red | 15 |
| 2 | osaka | blue | 13 |
| 3 | osaka | red | 9 |
| 0 | tokyo | blue | 10 |
| 1 | tokyo | red | 15 |
| 2 | osaka | blue | 13 |
| 3 | osaka | red | 9 |

In [22]:

```
df = pd.pivot_table(shoes, values='sales',aggfunc='sum', index= 's
df
```

| color | blue | red |
|---|---|---|
| store | | |
| osaka | 26 | 18 |
| tokyo | 20 | 30 |

In [24]:

```
#store와 신발의 색상간의 독립성

import scipy as sp
from scipy import stats
sp.stats.chi2_contingency(df, correction=False)

# p값이 0.05보다 크므로 store와 신발의 색상간 유의한 차이가 없다.
```

```
(3.413537549407115,
 0.06466368573255789,
 1,
 array([[21.53191489, 22.46808511],
        [24.46808511, 25.53191489]]))
```

Q13. 'dataset/titanic3.csv'을 불러와서 pclass 와 sex 칼럼을 각각 인덱스, 칼럼으로 하고 values는 survived, 함수는 mean을 적용하여 pivot_table을 만든 후 히트맵으로 시각화 및 인사이트를 기술하세요

In [25]:

```
titanic = pd.read_csv('titanic3.csv')
titanic.head(2)
```

| | pclass | survived | name | sex | age | sibsp | p |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 1 | Allen, Miss. Elisabeth Walton | female | 29.00 | 0 | 0 |
| **1** | 1 | 1 | Allison, Master. Hudson Trevor | male | 0.92 | 1 | 2 |

In [27]:

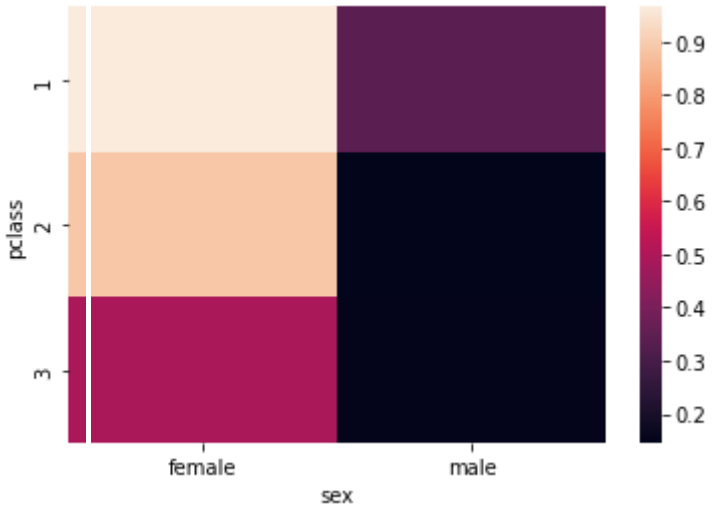
```
titanic3 = pd.pivot_table(titanic, values='survived',aggfunc='mean
titanic3
```

| sex | female | male |
|---|---|---|
| pclass | | |
| 1 | 0.965278 | 0.340782 |
| 2 | 0.886792 | 0.146199 |
| 3 | 0.490741 | 0.152130 |

In [31]:

```
import seaborn as sns

sns.heatmap(titanic3)

#남자에 비해 여자가 생존률이 높고, 남자는 등급이 낮을 수록 생존률(
```

```
<AxesSubplot:xlabel='sex', ylabel='pclass'>
```




Q14. 평균 4, 표준편차 0.8인 정규분포에서 샘플사이즈 10인 표본 10000개의 표본평균을 배열로 저장하고 10개를 출력하세요.(넘파이 zeros 함수 이용)

In [58]:

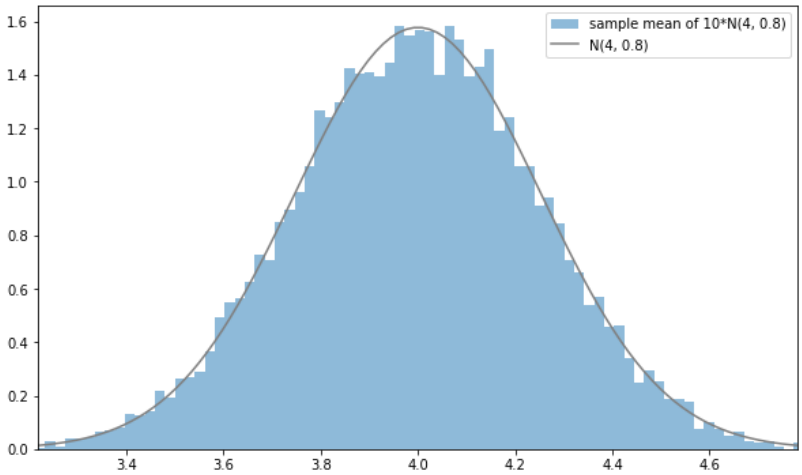
```
rv = stats.norm(4, 0.8)

n = 10
sample_size = 10000
Xs_sample = rv.rvs((n, sample_size))
sample_mean = np.mean(Xs_sample, axis=0)

result = np.zeros(10)
# sample_mean[:10]
real_result = result + sample_mean[:10]
real_result
```

```
array([4.16948363, 4.32791537, 4.20146264, 3.776395
37, 4.11623406,
       4.01484046, 4.1778668 , 4.37089489, 3.952660
65, 4.22255821])
```

Q15. Q14에서 구한 배열의 히스토그램을 시각화하세요.(확률밀도 포함)

In [60]:

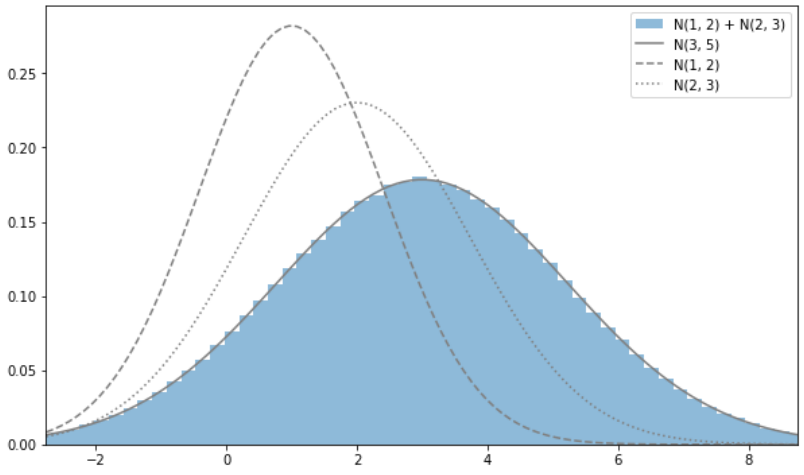
```
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111)

rv_true = stats.norm(4, 0.8/np.sqrt(n))
xs = np.linspace(rv_true.isf(0.999), rv_true.isf(0.001), 100)
ax.hist(sample_mean, bins=100, density=True,
        alpha=0.5, label='sample mean of 10*N(4, 0.8)')
ax.plot(xs, rv_true.pdf(xs), label='N(4, 0.8)', color='gray')

ax.legend()
ax.set_xlim(rv_true.isf(0.999), rv_true.isf(0.001))
plt.show()
```



Q16. 서로 독립인 X~N(1,2), Y~N(2,3)이 있을 때 확률변수 X + Y의 분포는 N(3,5)를 따른다는 것을 시각화하여 출력하세요.

In [66]:

```
rv1 = stats.norm(1, np.sqrt(2))
rv2 = stats.norm(2, np.sqrt(3))
sample_size = int(1e6)
X_sample = rv1.rvs(sample_size)
Y_sample = rv2.rvs(sample_size)
sum_sample = X_sample + Y_sample

fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111)

rv = stats.norm(3, np.sqrt(5))
xs = np.linspace(rv.isf(0.995), rv.isf(0.005), 100)

ax.hist(sum_sample, bins=100, density=True,
        alpha=0.5, label='N(1, 2) + N(2, 3)')
ax.plot(xs, rv.pdf(xs), label='N(3, 5)', color='gray')
ax.plot(xs, rv1.pdf(xs), label='N(1, 2)', ls='--', color='gray')
ax.plot(xs, rv2.pdf(xs), label='N(2, 3)', ls=':', color='gray' )

ax.legend()
ax.set_xlim(rv.isf(0.995), rv.isf(0.005))
plt.show()
```

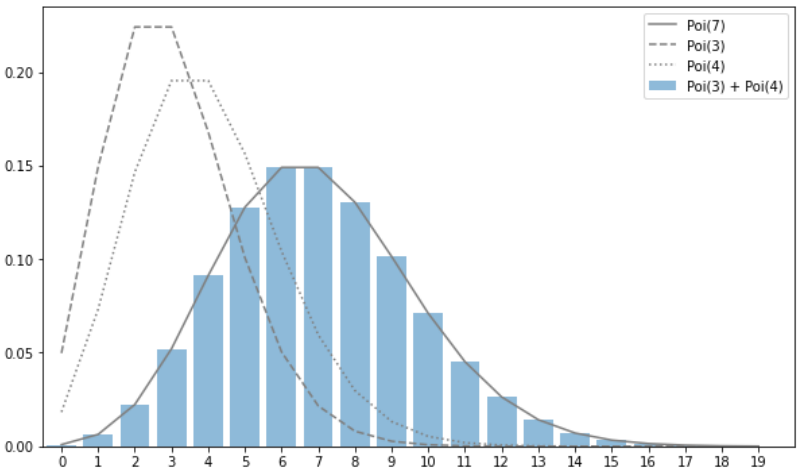



Q17. 서로 독립인 X ~ Poi(3)과 Y ~ Poi(4)가 있을 때 확률변수 X + Y도 포아송 분포를 따른다는 것을 시각화하여 출력하세요.

In [67]:

```
rv1 = stats.poisson(3)
rv2 = stats.poisson(4)

sample_size = int(1e6)
X_sample = rv1.rvs(sample_size)
Y_sample = rv2.rvs(sample_size)
sum_sample = X_sample + Y_sample

fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111)

rv = stats.poisson(7)
xs = np.arange(20)
hist, _ = np.histogram(sum_sample, bins=20,
                       range=(0, 20), normed=True)

ax.bar(xs, hist, alpha=0.5, label='Poi(3) + Poi(4)')
ax.plot(xs, rv.pmf(xs), label='Poi(7)',  color='gray')
ax.plot(xs, rv1.pmf(xs), label='Poi(3)', ls='--', color='gray')
ax.plot(xs, rv2.pmf(xs), label='Poi(4)', ls=':',  color='gray')

ax.legend()
ax.set_xlim(-0.5, 20)
ax.set_xticks(np.arange(20))
plt.show()
```

```
C:\Users\user\AppData\Local\Temp/ipykernel_15404/35
87039884.py:14: VisibleDeprecationWarning: Passing
`normed=True` on non-uniform bins has always been b
roken, and computes neither the probability density
function nor the probability mass function. The res
ult is only correct if the bins are uniform, when d
ensity=True will produce the same result anyway. Th
e argument will be removed in a future version of n
umpy.
  hist, _ = np.histogram(sum_sample, bins=20,
```

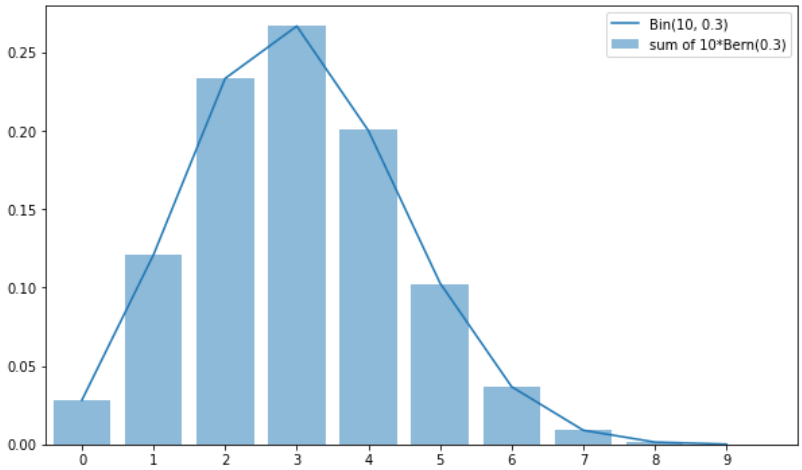


Q18. 베르누이 분포의 합은 이항분포가 되는 성질을 시각화하여 출력하세요

In [68]:

```
p = 0.3
rv = stats.bernoulli(p)

sample_size = int(1e6)
Xs_sample = rv.rvs((10, sample_size))
sum_sample = np.sum(Xs_sample, axis=0)

fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111)

rv = stats.binom(10, p)
xs = np.arange(10)
hist, _ = np.histogram(sum_sample, bins=10,
                       range=(0, 10), normed=True)
ax.bar(xs, hist, alpha=0.5, label='sum of 10*Bern(0.3)')
ax.plot(xs, rv.pmf(xs), label='Bin(10, 0.3)')
ax.legend()
ax.set_xlim(-0.5, 10)
ax.set_xticks(np.arange(10))
plt.show()
```

```
C:\Users\user\AppData\Local\Temp/ipykernel_15404/14
92525424.py:13: VisibleDeprecationWarning: Passing
`normed=True` on non-uniform bins has always been b
roken, and computes neither the probability density
function nor the probability mass function. The res
ult is only correct if the bins are uniform, when d
ensity=True will produce the same result anyway. Th
e argument will be removed in a future version of n
umpy.
  hist, _ = np.histogram(sum_sample, bins=10,
```



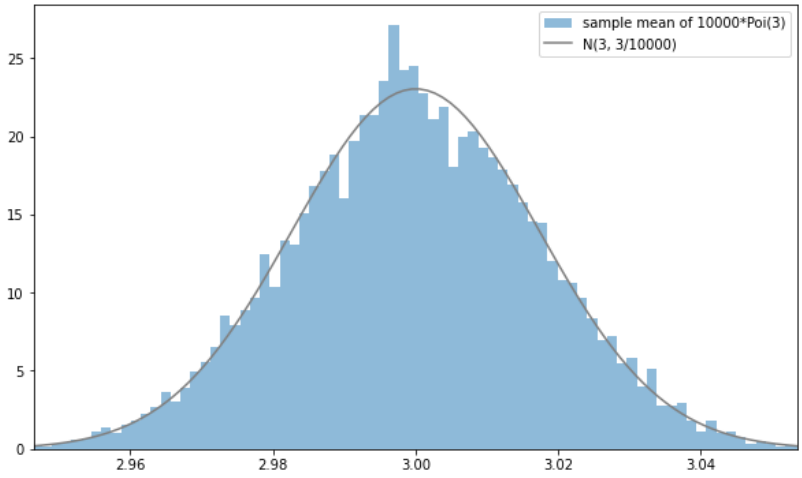

Q19. 포아송 분포의 표본분포는 근사적으로 정규분포를 따른다는 것을 시각화하고 그 핵심 근거인 중심극한정리에 대하여 설명하세요.

In [69]:

```
l = 3
rv = stats.poisson(l)

n = 10000
sample_size = 10000
Xs_sample = rv.rvs((n, sample_size))
sample_mean = np.mean(Xs_sample, axis=0)

rv_true = stats.norm(l, np.sqrt(l/n))
xs = np.linspace(rv_true.isf(0.999), rv_true.isf(0.001), 100)
```

In [70]:

```
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111)

ax.hist(sample_mean, bins=100, density=True,
        alpha=0.5, label='sample mean of 10000*Poi(3)')
ax.plot(xs, rv_true.pdf(xs), label='N(3, 3/10000)', color='gray')

ax.legend()
ax.set_xlim(rv_true.isf(0.999), rv_true.isf(0.001))
plt.show()

#n이 커짐에 따라 표본 평균의 분포는 정규분포와 가까워진다.
```




Q20. 아래 df 데이터셋에서 "무게의 평균이 130kg이다."라는 귀무가설에 대한 유의성 검정을 수행하세요.

In [71]:

```
df = pd.read_csv('data/ch11_potato.csv')
print(df.head(),len(df))
```

```
     무게
0  122.02
1  131.73
2  130.60
3  131.82
4  132.05 14
```

In [72]:

```
#모분산을 모르는 경우
def pmean_test(sample, mean0, alpha=0.05):
    s_mean = np.mean(sample)
    u_var = np.var(sample, ddof=1)
    n = len(sample)
    rv = stats.t(df=n-1)
    interval = rv.interval(1-alpha)

    t = (s_mean - mean0) / np.sqrt(u_var/n)
    if interval[0] <= t <= interval[1]:
        print('귀무가설을 채택')
    else:
        print('귀무가설을 기각')

    if t < 0:
        p = rv.cdf(t) * 2
    else:
        p = (1 - rv.cdf(t)) * 2
    print(f'p값은 {p:.3f}')
```

In [73]:

```
pmean_test(sample, 130)
```

```
귀무가설을 기각
p값은 0.000
```

In [ ]: