2장 연습문제

[1]:
```
import numpy as np
import math as m
import pandas as pd
```

1 번

[2]:
```
#(a)
1+2*(3+4)
```

[2]: 15

[3]:
```
#(b)
1+1/2+1/3+1/4+1/5+1/6
```

[3]: 2.4499999999999997

[4]:
```
#(c)
np.sqrt((4+3)*(2+5))
```

[4]: 7.0

[5]:
```
#(d)
x=(1+2)/(4+5)
x*x*x
```

[5]: 0.037037037037037035

[6]:
```
#(e)
122+12*23
```

[6]: 398

[7]:
```
#(f)
m.factorial(10)
```

[7]: 3628800

[8]:
```
#(g)
np.sqrt(6*6+4)
```

[8]: 6.324555320336759

[9]:
```
#(h)
np.sin(np.pi/3)+np.cos(np.pi/6)
```

[9]: 1.7320508075688772

[10]:
```
#(i)
np.log10(24)+np.log(10)
```

[10]: 3.6827963347056852

[11]:
```
#(j)
np.sin(np.pi/4)
```

[11]: 0.7071067811865476

[12]:
```
#(k)
np.cos(np.pi/3)
```

[12]: 0.5000000000000001

[13]:
```
#(l)
(1+2+3)/(4+5+6)
```

[13]: 0.4

[14]:
```
#(m)
sum=0
for i in range(5,12):
    sum=sum+i*i
print(sum)
```

476

[15]:
```
#(n)
x=15
y=3

np.sqrt((3*x*x+2*y*y*y)/((x+y)*(x-y)))
```

[15]: 1.8371173070873836

2 번

[16]:
```
#(a)
x=np.array([2,3,5,7,9,10])
x
```

[16]:
```
array([ 2,  3,  5,  7,  9, 10])
```

[17]:
```
#(b)
x2=x*x
x2
```

[17]:
```
array([  4,   9,  25,  49,  81, 100])
```

[18]:
```
#(c)
sum=0
for i in range(len(x2)):
    sum=sum+x2[i]
print(sum)
```

```
268
```

[19]:
```
#(d)
x-2
```

[19]:
```
array([0, 1, 3, 5, 7, 8])
```

[20]:
```
#(e)
print("max :" ,max(x))
print("min :" ,min(x))
```

```
max : 10
min : 2
```

[21]:
```
#(f)
x_up=x[np.where(x>5)]
x_up
```

[21]:
```
array([ 7,  9, 10])
```

[22]:
```
#(g)
len(x)
```

[22]:
```
6
```

[23]:
```
#(h)
np.matmul(x.transpose(),x)
```

[23]:
```
268
```

[24]:
```
#(i)
np.outer(x,x.transpose())
```

[24]:
```
array([[  4,   6,  10,  14,  18,  20],
       [  6,   9,  15,  21,  27,  30],
       [ 10,  15,  25,  35,  45,  50],
       [ 14,  21,  35,  49,  63,  70],
       [ 18,  27,  45,  63,  81,  90],
       [ 20,  30,  50,  70,  90, 100]])
```

[25]:
```
#(j)
xc=np.column_stack([x,x2])
xc
```

[25]:
```
array([[  2,   4],
       [  3,   9],
       [  5,  25],
       [  7,  49],
       [  9,  81],
       [ 10, 100]])
```

[26]:
```
#(k)
xr=np.vstack([x,x2])
xr
```

[26]:
```
array([[  2,   3,   5,   7,   9,  10],
       [  4,   9,  25,  49,  81, 100]])
```

3 번

[27]:
```
A=np.matrix([[1,-1,4], [-1,1,3], [4,3,2]])
B=np.matrix([[3,-2,4], [-2,1,0], [4,0,5]])
x=np.array([1,-2,4]).transpose()
y=np.array([3,2,1]).transpose()
print("A: ",A)
print("B: ",B)
print("x: ",x)
print("y: ",y)
```

```
A:  [[ 1 -1  4]
 [-1  1  3]
 [ 4  3  2]]
B:  [[ 3 -2  4]
 [-2  1  0]
 [ 4  0  5]]
x:  [ 1 -2  4]
y:  [3 2 1]
```

[28]:
```python
#(a)
A+B
```

[28]:
```
matrix([[ 4, -3,  8],
        [-3,  2,  3],
        [ 8,  3,  7]])
```

[29]:
```python
#(b)
A.transpose()
```

[29]:
```
matrix([[ 1, -1,  4],
        [-1,  1,  3],
        [ 4,  3,  2]])
```

[30]:
```python
#(c)
a1=np.matmul(x.transpose(),A)
np.matmul(a1,y)
```

[30]:
```
matrix([[81]])
```

[31]:
```python
#(d)
np.matmul(x.transpose(),x)
```

[31]:
```
21
```

[32]:
```python
#(e)
a2=np.matmul(x.transpose(),A)
np.matmul(a1,x)
```

[32]:
```
matrix([[25]])
```

[33]:
```python
#(f)
np.matmul(x.transpose(),y)
```

[33]:
```
3
```

[34]:
```python
#(g)
np.matmul(A.transpose(),A)
```

[34]:
```
matrix([[18, 10,  9],
        [10, 11,  5],
        [ 9,  5, 29]])
```

[35]:
```python
#(h)
np.matmul(A,B)
```

[35]:
```
matrix([[21, -3, 24],
        [ 7,  3, 11],
        [14, -5, 26]])
```

[36]:
```python
#(i)
np.matmul(y.transpose(),B)
```

[36]:
```
matrix([[ 9, -4, 17]])
```

[37]:
```python
#(j)
np.outer(x,x.transpose())
```

[37]:
```
array([[ 1, -2,  4],
       [-2,  4, -8],
       [ 4, -8, 16]])
```

[38]:
```python
#(k)
x+y
```

[38]:
```
array([4, 0, 5])
```

[39]:
```python
#(l)
x-y
```

[39]:
```
array([-2, -4,  3])
```

[40]:
```python
#(m)
(x-y).transpose()
```

[40]:
```
array([-2, -4,  3])
```

[41]:
```python
#(n)
np.outer(x,y.transpose())
```

[41]:
```
array([[ 3,  2,  1],
       [-6, -4, -2],
       [12,  8,  4]])
```

[42]:
```python
#(o)
A-B
```

[42]:
```
matrix([[-2,  1,  0],
        [ 1,  0,  3],
        [ 0,  3, -3]])
```

[43]:
```
#(p)
A.transpose()+B.transpose()
```

[43]:
```
matrix([[ 4, -3,  8],
        [-3,  2,  3],
        [ 8,  3,  7]])
```

[44]:
```
#(q)
(A+B).transpose()
```

[44]:
```
matrix([[ 4, -3,  8],
        [-3,  2,  3],
        [ 8,  3,  7]])
```

[45]:
```
#(r)
3*x
```

[45]:
```
array([ 3, -6, 12])
```

[46]:
```
#(s)
(np.matmul(x.transpose(),y)*np.matmul(x.transpose(),y))
```

[46]:
```
9
```

[47]:
```
#(t)
np.matmul(B,A)
```

[47]:
```
matrix([[21,  7, 14],
        [-3,  3, -5],
        [24, 11, 26]])
```

[48]:
```
#(u)
np.linalg.inv(A)
```

[48]:
```
matrix([[ 0.14285714, -0.28571429,  0.14285714],
        [-0.28571429,  0.28571429,  0.14285714],
        [ 0.14285714,  0.14285714,  0.        ]])
```

4 번

[49]:
```
#(a)
np.tile("a",8)
```

[49]:
```
array(['a', 'a', 'a', 'a', 'a', 'a', 'a', 'a'], dtype='<U1')
```

[50]:
```
#(b)
one=np.tile(1,3)
two=np.tile(2,3)
```

```
three=np.tile(3,3)
four=np.tile(4,3)
five=np.tile(5,3)
np.concatenate((one,two,three,four,five),axis=0)
```

[50]:
```
array([1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 5, 5, 5])
```

[51]:
```
#(c)
np.arange(1,100,step=2)
```

[51]:
```
array([ 1,  3,  5,  7,  9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33,
       35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67,
       69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99])
```

[52]:
```
#(d)
np.array([1,5,19,30])
```

[52]:
```
array([ 1,  5, 19, 30])
```

[53]:
```
np.arange(-10,11)
```

[53]:
```
array([-10,  -9,  -8,  -7,  -6,  -5,  -4,  -3,  -2,  -1,   0,   1,   2,
         3,   4,   5,   6,   7,   8,   9,  10])
```

5 번

[54]:
```
x=np.arange(1,11)
x
```

[54]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

[55]:
```
#(a)
len(x)
```

[55]:
```
10
```

[56]:
```
#(b)
np.sum(x)
```

[56]:
```
55
```

[57]:
```
#(c)
np.mean(x)
```

[57]:
```
5.5
```

[58]:
```
#(d)
print(np.var(x,ddof=1))
print(np.std(x,ddof=1))
```

```
9.166666666666666
3.0276503540974917
```

[59]:
```
#(e)
odd=x[np.where(x%2==1)]
odd
```

[59]:
```
array([1, 3, 5, 7, 9])
```

[60]:
```
#(f)
sum=0.0
for i in range(len(x)):
    sum=sum+x[i]/(i+1)
sum
```

[60]:
```
10.0
```

6 번

[62]:
```
brother=pd.read_csv("C:/resultfile/brother_e.csv")
print(brother)
```

```
   id  elder  younger
0   1     86       88
1   2     71       77
2   3     77       76
3   4     68       64
4   5     91       96
5   6     72       72
```

[63]:
```
#(c)
elder=brother.loc[:,['elder']]
print(elder)
mean_e=np.around(np.mean(elder).values,3)
print("elder mean: ",mean_e)
sd_e=np.around(np.sqrt(np.var(elder).values),3)
print("elder sd: ",sd_e)

younger=brother.loc[:,['younger']]
print(younger)
mean_y=np.around(np.mean(younger).values,3)
print("younger mean: ",mean_y)
sd_y=np.around(np.sqrt(np.var(younger).values),3)
print("younger sd: ",sd_y)
```

```
   elder
0     86
1     71
2     77
3     68
4     91
5     72
elder mean:  [77.5]
elder sd:  [8.342]
   younger
0       88
1       77
2       76
3       64
4       96
5       72
younger mean:  [78.833]
younger sd:  [10.463]
```

[65]:
```
#(d)
data={'elder_mean' : mean_e,
      'elder_sd' : sd_e,
      'younger_mean': mean_y,
      'younger_sd': sd_y}
data=pd.DataFrame(data)
data

data.to_csv("C:/resultfile/attackout.txt",header=True, index=False)
```

7 번

[66]:
```
#(a)
x=np.array([-4.123,-3.556,1.634,2.213,3.875])
x
```

[66]:
```
array([-4.123, -3.556,  1.634,  2.213,  3.875])
```

[67]:
```
#(b)
y=np.around(x,2)
y
```

[67]:
```
array([-4.12, -3.56,  1.63,  2.21,  3.88])
```

[68]:
```
#(c)
x-y
```

[68]:
```
array([-0.003,  0.004,  0.004,  0.003, -0.005])
```

[69]:
```python
#(d)
np.around(x,1)
```

[69]: array([-4.1, -3.6, 1.6, 2.2, 3.9])

[70]:
```python
#(e)
np.ceil(x)
```

[70]: array([-4., -3., 2., 3., 4.])

[71]:
```python
#(f)
np.trunc(x)
```

[71]: array([-4., -3., 1., 2., 3.])

8 번

[72]:
```python
data1={'name' : ['kim','lee','park','oh','yang','min','jung','moon'],
      'Korean': [93,76,87,92,98,75,82,92]}
data1=pd.DataFrame(data1)
print("data1\n",data1)

data2={'name2' : ['kim','lee','park','oh','yang','min','jung','choi'],
      'English': [90,94,88,75,79,87,88,90]}
data2=pd.DataFrame(data2)
print("data2\n",data2)
```

```
data1
    name  Korean
0   kim      93
1   lee      76
2  park      87
3    oh      92
4  yang      98
5   min      75
6  jung      82
7  moon      92
data2
   name2  English
0   kim       90
1   lee       94
2  park       88
3    oh       75
4  yang       79
5   min       87
6  jung       88
7  choi       90
```

[75]:
```python
#(a)
data_merge=pd.merge(data1,data2,left_on=['name'],right_on=['name2'],how='outer')
data_merge.drop(['name2'],axis=1,inplace=True)
data_merge.loc[8,'name']='choi'
data_merge
```

[75]:

	name	Korean	English
0	kim	93.0	90.0
1	lee	76.0	94.0
2	park	87.0	88.0
3	oh	92.0	75.0
4	yang	98.0	79.0
5	min	75.0	87.0
6	jung	82.0	88.0
7	moon	92.0	NaN
8	choi	NaN	90.0

[76]:
```python
#(b)
data_merge.sort_values(by='name')
```

[76]:

	name	Korean	English
8	choi	NaN	90.0
6	jung	82.0	88.0
0	kim	93.0	90.0
1	lee	76.0	94.0
5	min	75.0	87.0
7	moon	92.0	NaN
3	oh	92.0	75.0
2	park	87.0	88.0
4	yang	98.0	79.0

9 번

[77]:
```python
class1={'class': ['1','1','1','1','1','1','1','1'],
        'name' : ['kim','lee','park','oh','ang','min','jung','moon'],
        'Korean': [93,84,87,95,98,77,82,92]}
class1=pd.DataFrame(class1)
print("class1\n",class1)

class2={'class': ['2','2','2','2','2','2','2','2'],
        'name' : ['kang','yun','park','cho','yang','min','jung','choi'],
        'Korean': [90,95,88,75,79,87,90,90]}
class2=pd.DataFrame(class2)
print("class2\n",class2)
```

```
class1
   class  name  Korean
```

```
0      1   kim      93
1      1   lee      84
2      1  park      87
3      1    oh      95
4      1   ang      98
5      1   min      77
6      1  jung      82
7      1  moon      92
class2
   class  name  Korean
0      2  kang      90
1      2   yun      95
2      2  park      88
3      2   cho      75
4      2  yang      79
5      2   min      87
6      2  jung      90
7      2  choi      90
```

[78]:
```
#(a)
class1_1=class1.loc[:,['Korean']]
mean_1=np.around(np.mean(class1_1).values,3)
print("class1 mean: ",mean_1)
sd_1=np.around(np.sqrt(np.var(class1_1).values),3)
print("class1 sd: ",sd_1)

class2_1=class2.loc[:,['Korean']]
mean_2=np.around(np.mean(class2_1).values,3)
print("class2 mean: ",mean_2)
sd_2=np.around(np.sqrt(np.var(class2_1).values),3)
print("class2 sd: ",sd_2)
```

```
class1 mean:  [88.5]
class1 sd:  [6.727]
class2 mean:  [86.75]
class2 sd:  [6.119]
```

[79]:
```
#(b)
korean=np.concatenate([class1,class2],axis=0)
korean=pd.DataFrame(korean)
korean.columns=['class','name',"Korean"]
print(korean)

korean1=korean.loc[:,['Korean']] print
("mean:",korean1.mean().values)
print("sd:",np.sqrt(korean1.var().values))
```

```
   class  name Korean
```

```
0       1   kim     93
1       1   lee     84
2       1  park     87
3       1    oh     95
4       1   ang     98
5       1   min     77
6       1  jung     82
7       1  moon     92
8       2  kang     90
9       2   yun     95
10      2  park     88
11      2   cho     75
12      2  yang     79
13      2   min     87
14      2  jung     90
15      2  choi     90
mean: [87.625]
sd: [6.70198975]
```

[80]:
```
korean.sort_values(by='name')
```

[80]:

	class	name	Korean
4	1	ang	98
11	2	cho	75
15	2	choi	90
6	1	jung	82
14	2	jung	90
8	2	kang	90
0	1	kim	93
1	1	lee	84
5	1	min	77
13	2	min	87
7	1	moon	92
3	1	oh	95
2	1	park	87
10	2	park	88
12	2	yang	79
9	2	yun	95

10 번

[81]:
```
def ctof(data):
    return (9*data/5+32)

c=np.arange(20,31)

for i in range(len(c)):
```

```
    print(c[i],"->",ctof(c[i]))
```

```
20 -> 68.0
21 -> 69.8
22 -> 71.6
23 -> 73.4
24 -> 75.2
25 -> 77.0
26 -> 78.8
27 -> 80.6
28 -> 82.4
29 -> 84.2
30 -> 86.0
```

11 번

[82]:
```
def area(radius):
    return(np.pi*radius*radius)

r=np.array([1,2,4,9,10])

for i in range(len(r)):
    print('radius:',r[i],'-> area:',area(r[i]))
```

```
radius: 1 -> area: 3.141592653589793
radius: 2 -> area: 12.566370614359172
radius: 4 -> area: 50.26548245743669
radius: 9 -> area: 254.46900494077323
radius: 10 -> area: 314.1592653589793
```

12 번

[83]:
```
def volumne(radius):
    return((4/3)*np.pi*radius*radius)

r=np.array([1.5,3])

for i in range(len(r)):
    print('radius: ',r[i],'-> volume: ',volumne(r[i]))
```

```
radius:  1.5 -> volume:  9.42477796076938
radius:  3.0 -> volume:  37.69911184307752
```

13번

[84]:
```python
#(a)
a=8
type(a)
```

[84]:
```
int
```

[85]:
```python
#(b)
a=8.0
type(a)
```

[85]:
```
float
```

[86]:
```python
#(c)
a='8'
type(a)
```

[86]:
```
str
```

[87]:
```python
#(d)
a=(8)
type(a)
```

[87]:
```
int
```

[88]:
```python
#(e)
a=[1,2,3]
type(a)
```

[88]:
```
list
```

[89]:
```python
#(f)
a=(1,2,3)
type(a)
```

[89]:
```
tuple
```

[90]:
```python
#(g)
a=False
type(a)
```

[90]:
```
bool
```

3장 연습문제

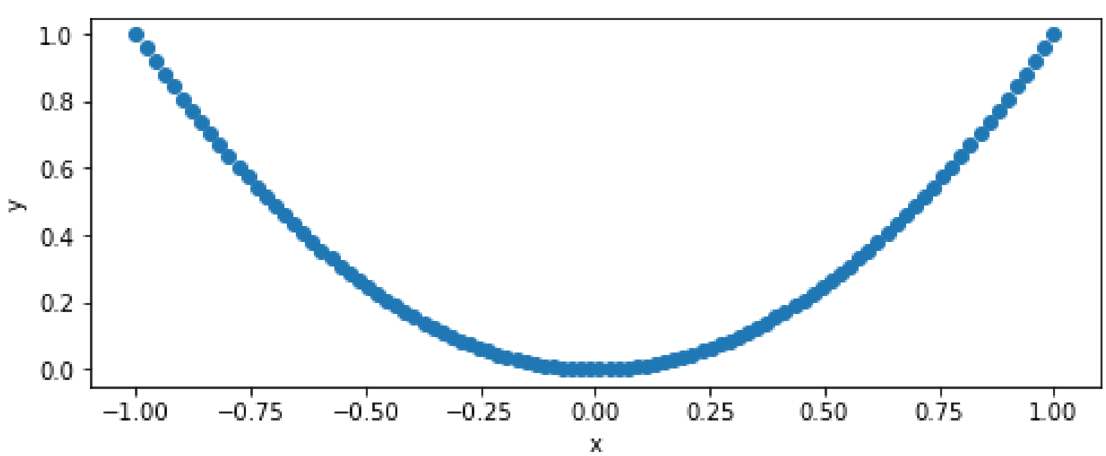
[5]:
```
import numpy as np
import math as m
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D
```

1 번

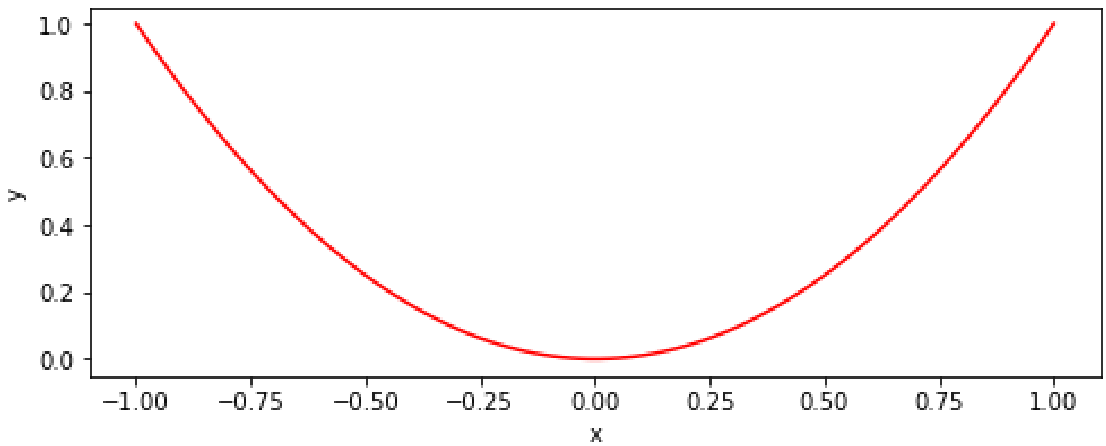
[2]:
```
#(a)
x=np.linspace(-1,1,100)
len(x)
```

[2]:
```
100
```

[3]:
```
#(b)
y=x*x
plt.figure(figsize=(8,3))
plt.scatter(x,y)
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



[4]:
```
#(c)
y=x*x
plt.figure(figsize=(8,3))
plt.plot(x,y,color="red")
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

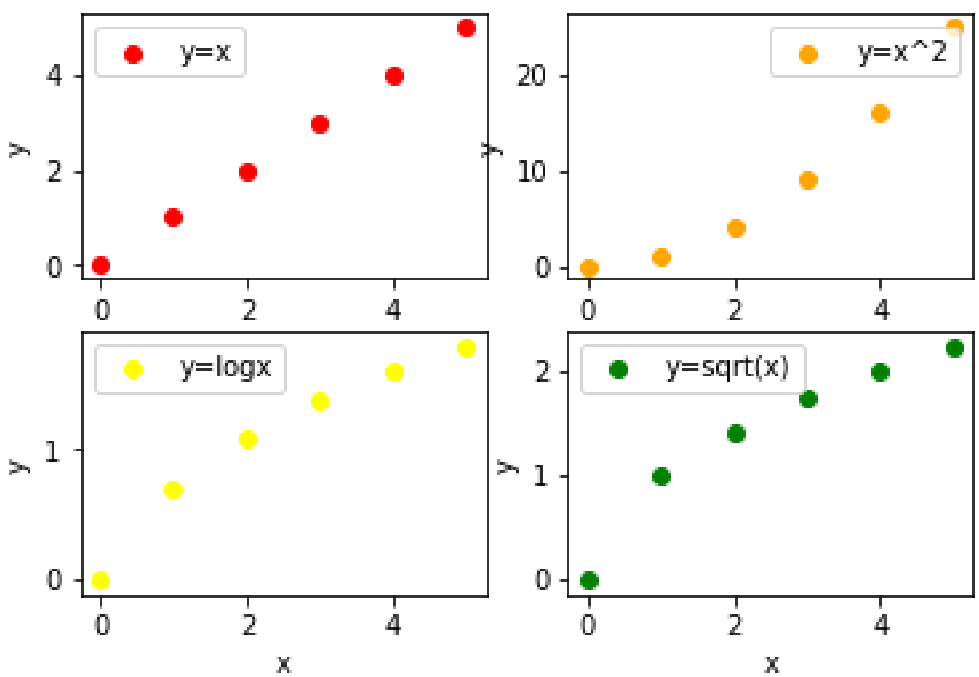


2번

[101]:
```
#(a)
x=np.arange(0,6)
y1=x
y2=x*x
y3=np.log(x+1)
y4=np.sqrt(x)

plt.subplot(221)
plt.scatter(x,y1,color="red",label="y=x")
plt.legend()
plt.xlabel('x')
plt.ylabel('y')

plt.subplot(222)
plt.scatter(x,y2,color="orange",label="y=x^2")
plt.legend()
plt.xlabel('x')
```

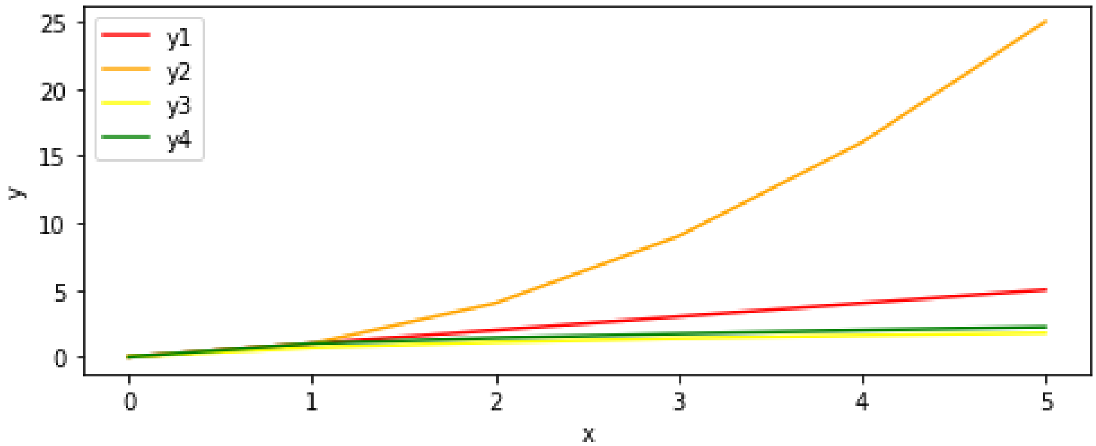
```
plt.ylabel('y')

plt.subplot(223)
plt.scatter(x,y3,color="yellow",label='y=logx')
plt.legend()
plt.xlabel('x')
plt.ylabel('y')

plt.subplot(224)
plt.scatter(x,y4,color="green",label='y=sqrt(x)')
plt.legend()
plt.xlabel('x')
plt.ylabel('y')
```

[101]: Text(0, 0.5, 'y')

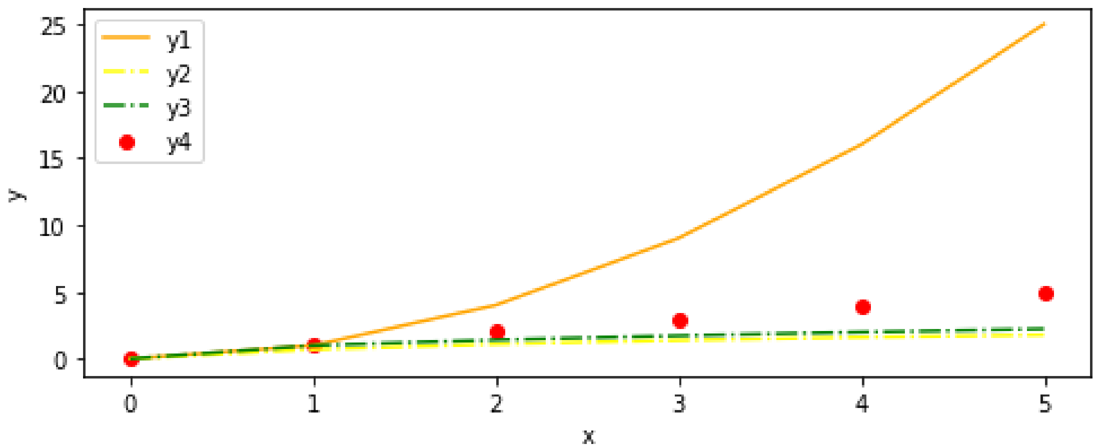


[8]:
```
#(b)
x=np.arange(0,6)
y1=x
y2=x*x
y3=np.log(x+1)
y4=np.sqrt(x)

plt.figure(figsize=(8,3))
```

```
plt.plot(x,y1,color="red")
plt.plot(x,y2,color="orange")
plt.plot(x,y3,color="yellow")
plt.plot(x,y4,color="green")
plt.legend(labels=('y1','y2','y3','y4'),loc='upper left')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

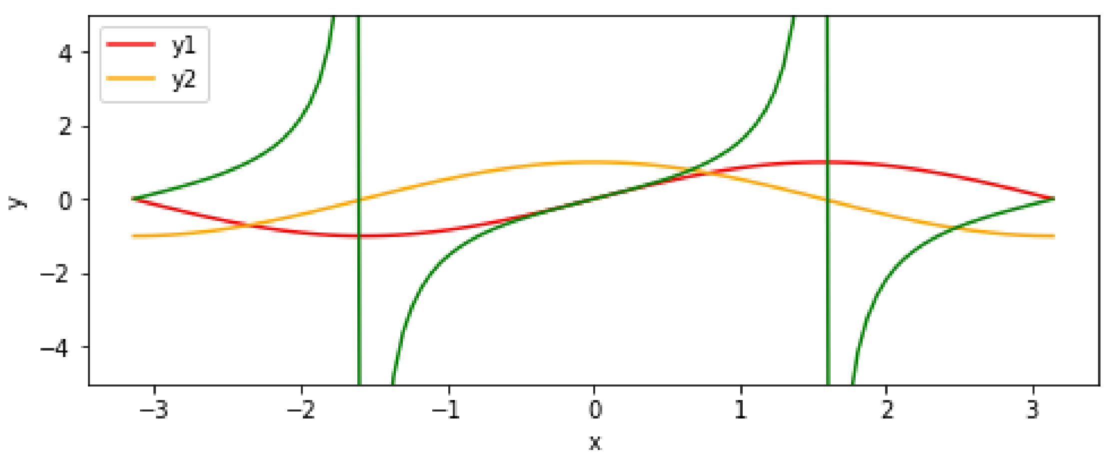


[9]:
```
#(c)
x=np.arange(0,6)
y1=x
y2=x*x
y3=np.log(x+1)
y4=np.sqrt(x)

plt.figure(figsize=(8,3))
plt.scatter(x,y1,color="red")
plt.plot(x,y2,color="orange")
plt.plot(x,y3,color="yellow",linestyle='-.')
plt.plot(x,y4,color="green",linestyle='-.')
plt.legend(labels=('y1','y2','y3','y4'),loc='upper left')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



3번

[25]:
```
x=np.linspace(-np.pi,np.pi,100)
y1=np.sin(x)
y2=np.cos(x)
y3=np.tan(x)

plt.figure(figsize=(8,3))
plt.plot(x,y1,color="red")
plt.plot(x,y2,color="orange")
plt.plot(x,y3,color="green")
plt.ylim(-5,5)
plt.legend(labels=('y1','y2'),loc='upper left')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

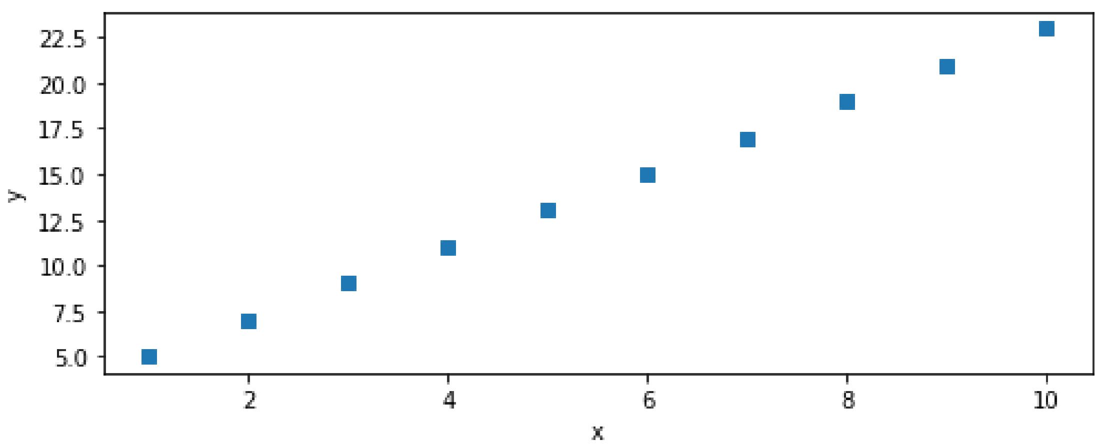


4번

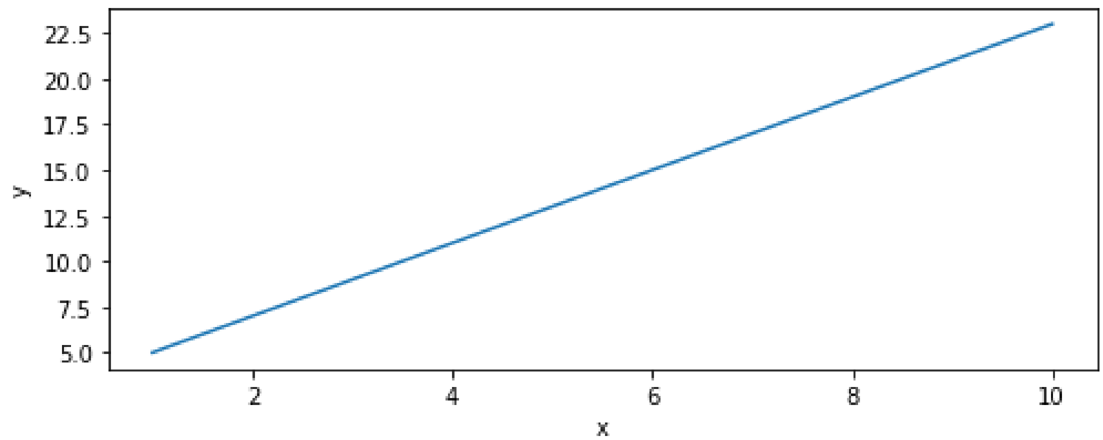
[12]:
```
x=np.arange(1,11)
print("x:",x)
y=2*x+3
print("y:",y)
```

```
x: [ 1  2  3  4  5  6  7  8  9 10]
y: [ 5  7  9 11 13 15 17 19 21 23]
```

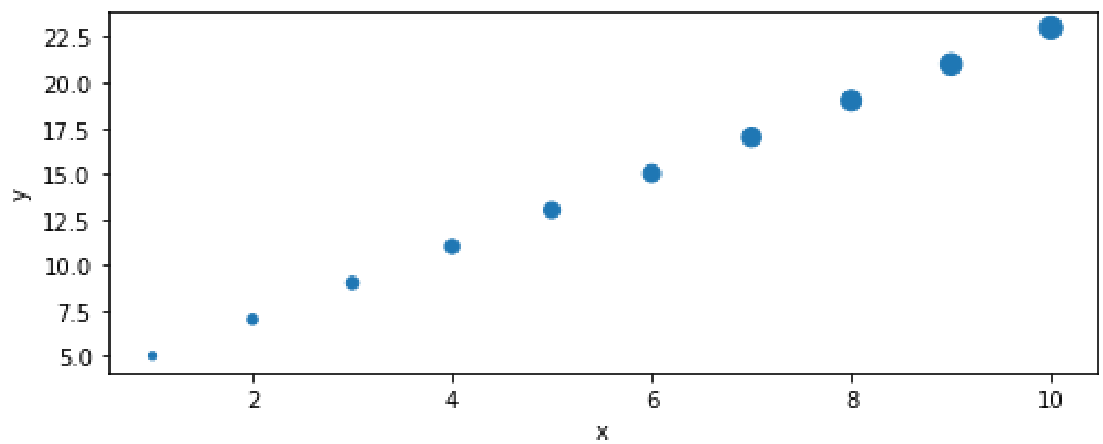
[13]:
```
#(a)
plt.figure(figsize=(8,3))
plt.scatter(x,y,marker='s')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



[14]:
```
#(b)
plt.figure(figsize=(8,3))
plt.plot(x,y)
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



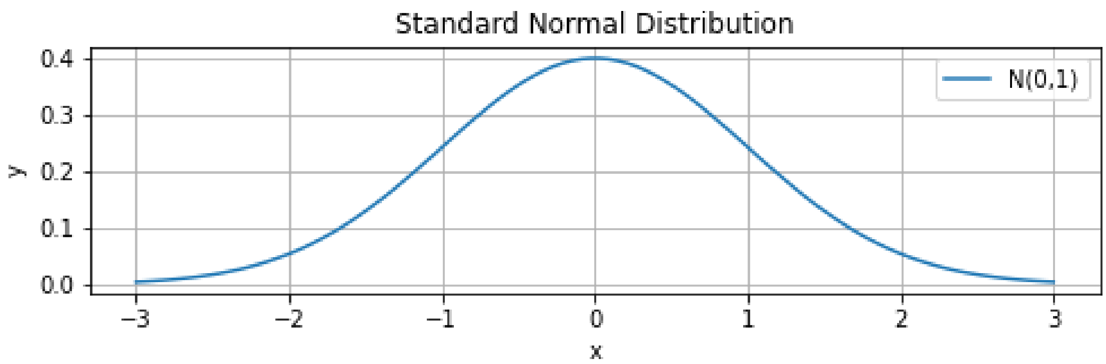
[15]:
```
#(c)
s=x*10
plt.figure(figsize=(8,3))
plt.scatter(x,y,s)
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



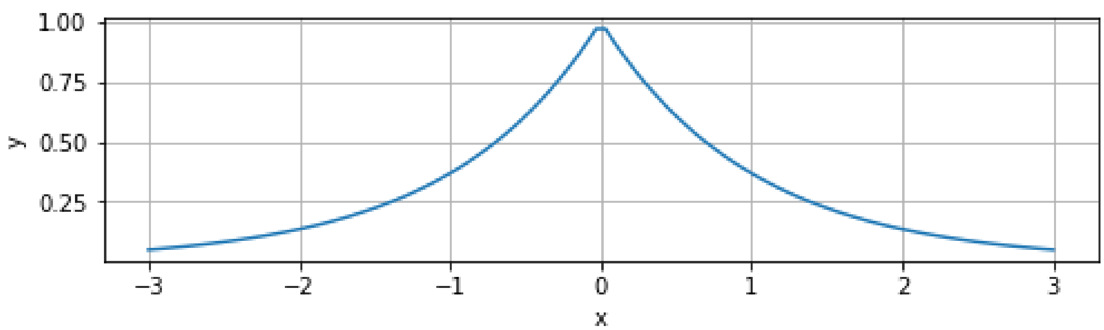
5번

[9]:
```
#(a)
x=np.linspace(-3,3,100)
y=(1/np.sqrt(2*np.pi))*np.exp(-x**2/2)
plt.figure(figsize=(8,2))
```

```
plt.plot(x,y)
plt.xlabel('x')
plt.ylabel('y')
plt.grid()
plt.title("Standard Normal Distribution")
plt.legend(["N(0,1)"])
plt.show()
```

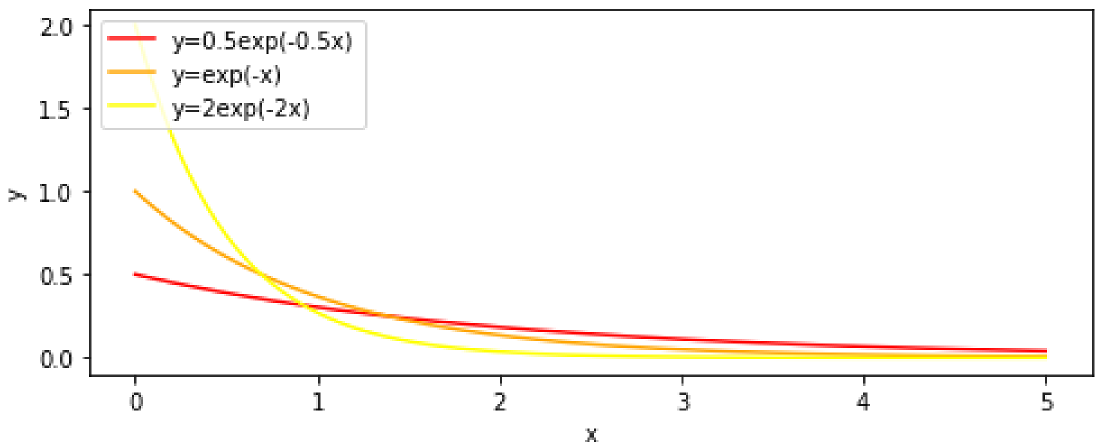


[10]:
```
#(b)
x=np.linspace(-3,3,100)
y=np.exp(-np.abs(x))
plt.figure(figsize=(8,2))
plt.plot(x,y)
plt.xlabel('x')
plt.ylabel('y')
plt.grid()
plt.show()
```



6번

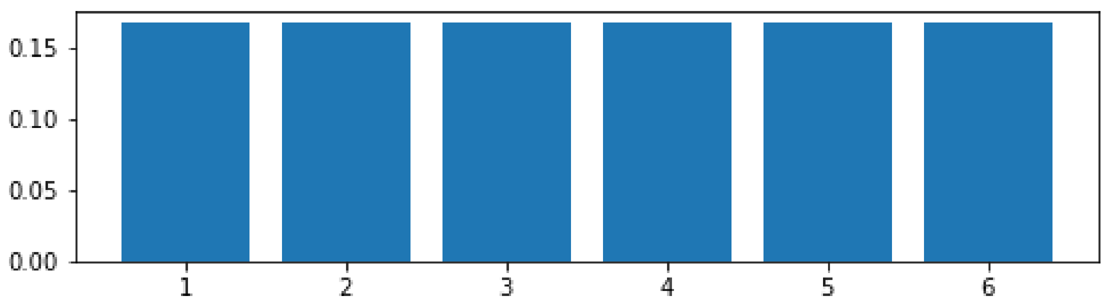
[18]:
```
x=np.linspace(0,5,100)
y1=0.5*np.exp(-0.5*x)
y2=np.exp(-x)
y3=2*np.exp(-2*x)

plt.figure(figsize=(8,3))
plt.plot(x,y1,color="red")
plt.plot(x,y2,color="orange")
plt.plot(x,y3,color="yellow")
plt.legend(labels=('y=0.5exp(-0.5x)','y=exp(-x)','y=2exp(-2x)'),loc='upper␣
↪left')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



7번

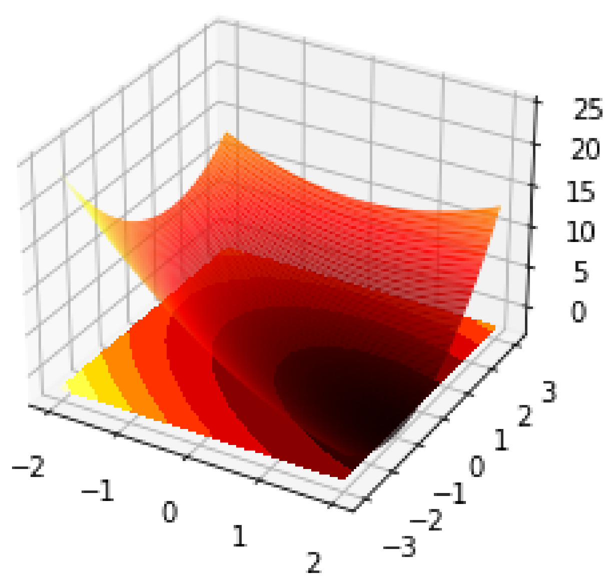
[11]:
```
x=np.arange(1,7)
y=1/6

plt.figure(figsize=(8,2))
plt.bar(x,y)
```

[11]: <BarContainer object of 6 artists>

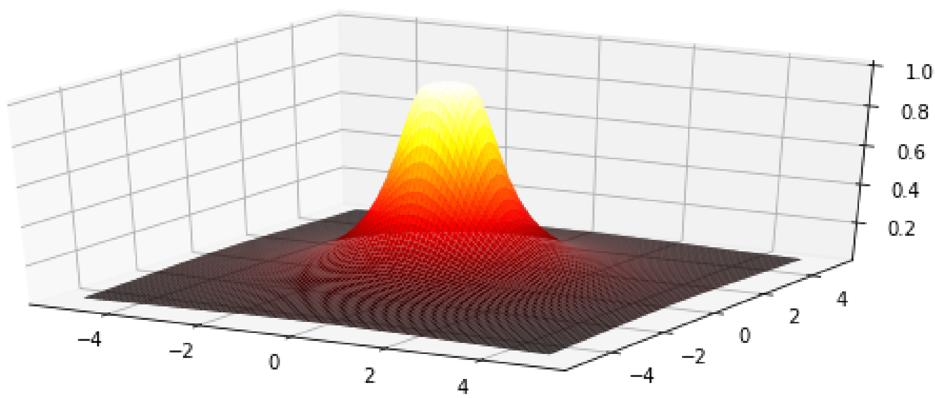


8번

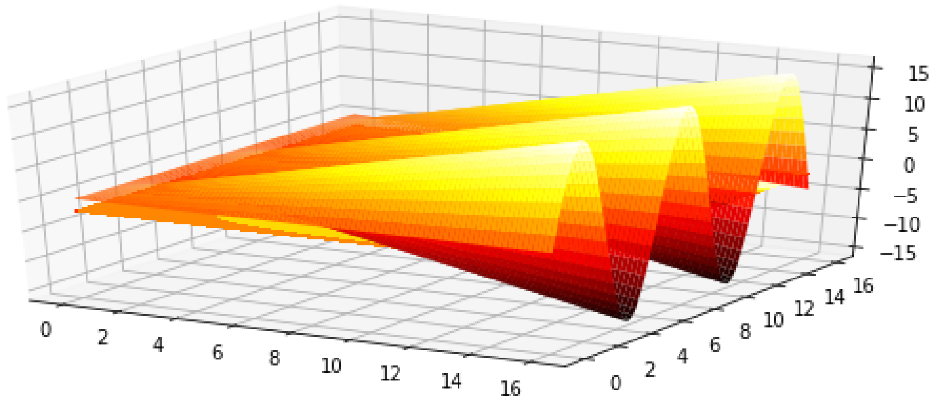
[8]:
```
#(a)
fig=plt.figure(figsize=(10,3))
ax=Axes3D(fig)

x=np.linspace(-2,2,100)
y=np.linspace(-3,3,100)

x,y=np.meshgrid(x,y)

z=x**2+y**2+x*(y-3)

ax.plot_surface(x,y,z,rstride=1,cstride=1,cmap=plt.cm.hot)
ax.contourf(x,y,z,zdir='z',offset=-2,cmap=plt.cm.hot)
plt.show()
```



[20]:
```
#(b)
fig=plt.figure(figsize=(8,3))
ax=Axes3D(fig)

x=np.linspace(-5,5,100)
y=np.linspace(-5,5,100)

x,y=np.meshgrid(x,y)

z=1-np.exp(-1/(x**2+y**2))

ax.plot_surface(x,y,z,rstride=1,cstride=1,cmap=plt.cm.hot)
ax.contourf(x,y,z,zdir='z',offset=-2,cmap=plt.cm.hot)
plt.show()
```



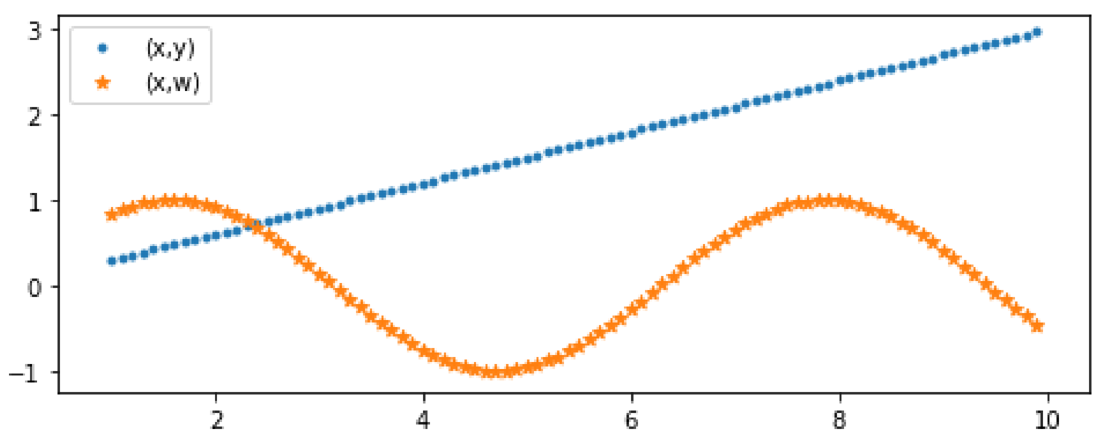
[21]:
```
#(c)
fig=plt.figure(figsize=(8,3))
ax=Axes3D(fig)

x=np.linspace(0,16,100)
y=np.linspace(0,16,100)

x,y=np.meshgrid(x,y)

z=x*np.sin(y)

ax.plot_surface(x,y,z,rstride=1,cstride=1,cmap=plt.cm.hot)
ax.contourf(x,y,z,zdir='z',offset=-2,cmap=plt.cm.hot)
plt.show()
```



9번

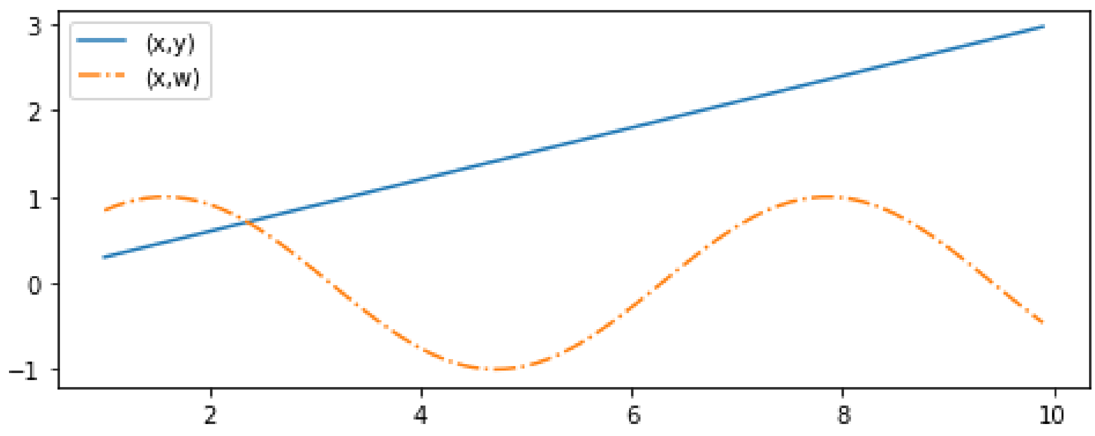
[22]:
```
x=np.arange(1,10,0.1)
y=x*0.3
w=np.sin(x)
```

[23]:
```
#(a)
plt.figure(figsize=(8,3))
plt.scatter(x,y,marker=".")
plt.scatter(x,w,marker="*")
plt.legend(labels=('(x,y)','(x,w)'))
plt.show()
```

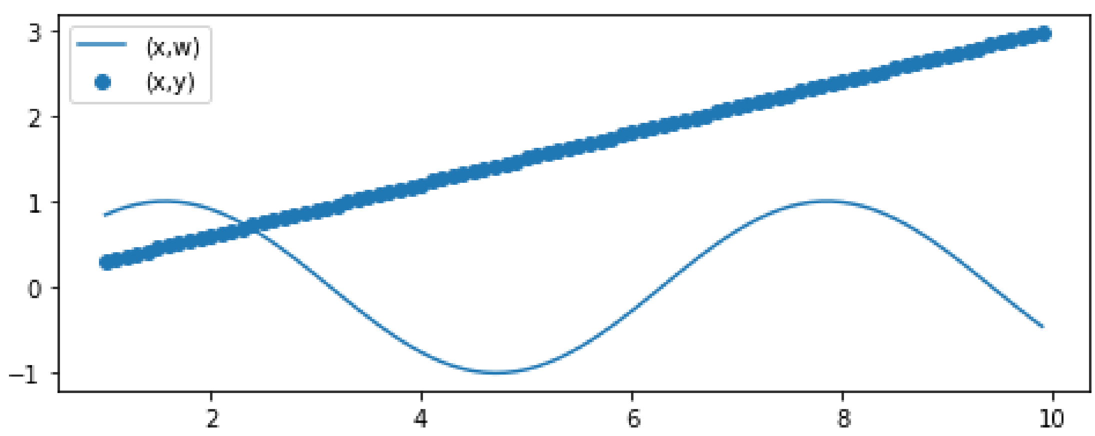


[24]:
```
#(b)
plt.figure(figsize=(8,3))
```

```
plt.plot(x,y)
plt.plot(x,w,linestyle='-.')
plt.legend(labels=('(x,y)','(x,w)'))
plt.show()
```



[25]:
```
#(c)
plt.figure(figsize=(8,3))
plt.scatter(x,y)
plt.plot(x,w)
plt.legend(labels=('(x,w)','(x,y)'))
plt.show()
```



10번

[26]:
```
iris=sns.load_dataset("iris")
iris.head()
```

[26]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

[27]:
```
#(a)
iris.mean(axis=0)
```

[27]:
```
sepal_length    5.843333
sepal_width     3.057333
petal_length    3.758000
petal_width     1.199333
dtype: float64
```

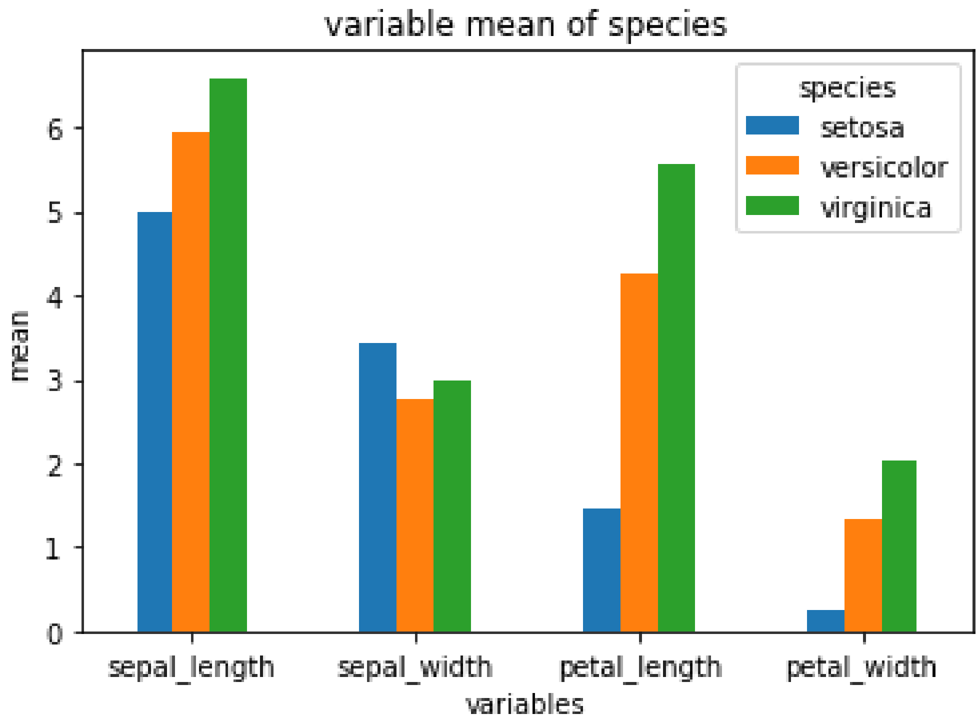
[28]:
```
#(b)
iris.groupby("species").mean()
```

[28]:

	sepal_length	sepal_width	petal_length	petal_width
species				
setosa	5.006	3.428	1.462	0.246
versicolor	5.936	2.770	4.260	1.326
virginica	6.588	2.974	5.552	2.026

[31]:
```
#(c)
st=iris.groupby(iris.species).mean()
print(st)

plt.figure(figsize=(8,3))
st.T.plot.bar(rot=0)
plt.title("variable mean of species")
plt.xlabel("variables")
plt.ylabel("mean")
plt.show()
```

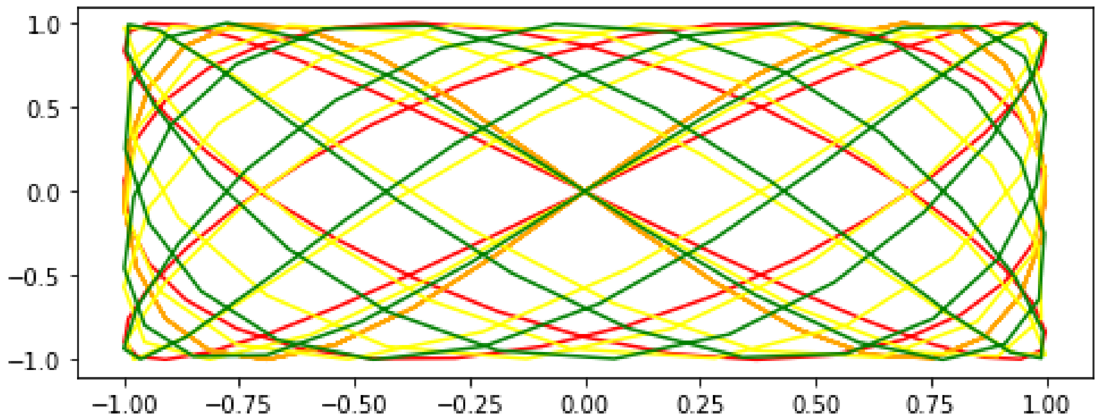
```
            sepal_length  sepal_width  petal_length  petal_width
species
setosa             5.006        3.428         1.462        0.246
versicolor         5.936        2.770         4.260        1.326
virginica          6.588        2.974         5.552        2.026
```



11번

[33]:
```
x=np.linspace(0,2*np.pi,100)

a1=3; b1=4
a2=3; b2=6
a3=5; b3=8
a4=4; b4=7

z1_1=np.sin(a1*x); z2_1=np.sin(b1*x)
z1_2=np.sin(a2*x); z2_2=np.sin(b2*x)
z1_3=np.sin(a3*x); z2_3=np.sin(b3*x)
z1_4=np.sin(a4*x); z2_4=np.sin(b4*x)


#z1_1,z2_1=np.meshgrid(np.sin(a1*x),np.sin(b1*x))
#z1_2,z2_2=np.meshgrid(np.sin(a2*x),np.sin(b2*x))
#z1_3,z2_3=np.meshgrid(np.sin(a3*x),np.sin(b3*x))
#z1_4,z2_4=np.meshgrid(np.sin(a4*x),np.sin(b4*x))

plt.figure(figsize=(8,3))
plt.plot(z1_1,z2_1,color="red")
plt.plot(z1_2,z2_2,color="orange")
```

```
plt.plot(z1_3,z2_3,color="yellow")
plt.plot(z1_4,z2_4,color="green")

plt.show()
```



12번

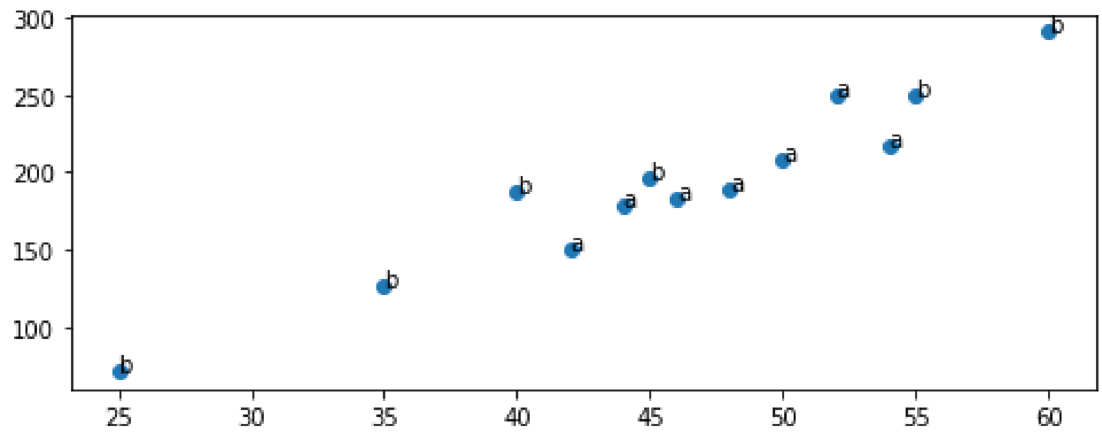
[34]:
```
group=np.array(["a","a","a","a","a","a","a","b","b","b","b","b","b"])
stretch=np.array([46,54,48,50,44,42,52,25,45,35,40,55,60])
distance=np.array([183,217,189,208,178,150,249,71,196,127,187,249,291])
band=pd.DataFrame({'group':group, 'stretch':stretch, 'distance':distance})
band
```

[34]:

	group	stretch	distance
0	a	46	183
1	a	54	217
2	a	48	189
3	a	50	208
4	a	44	178
5	a	42	150
6	a	52	249
7	b	25	71
8	b	45	196
9	b	35	127
10	b	40	187
11	b	55	249
12	b	60	291

[35]:
```
plt.figure(figsize=(8,3))
plt.scatter(x="stretch",y="distance",marker='o',data=band)
```

```
for i in range(len(band)):
    plt.annotate(group[i],(stretch[i],distance[i]))

plt.show()
```



4장 연습문제

[1]:
```
import numpy as np
import math as m
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
import scipy as sp
import statsmodels.formula.api as smf
import statsmodels.api as sm
import pylab as py
```

1번

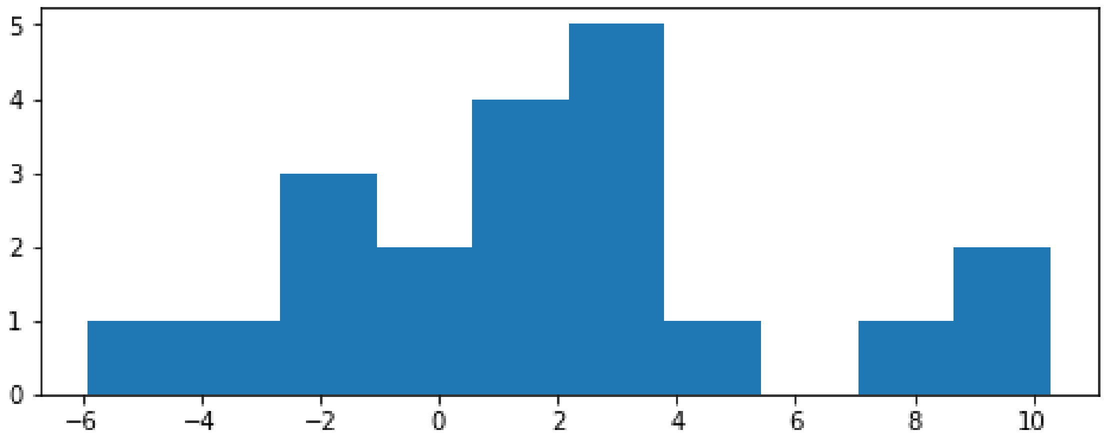
[2]:
```
#(a)
rand=sp.stats.norm.rvs(loc=3,scale=5,size=20)
rand
```

[2]:
```
array([ 0.50570091, -5.90677985,  2.19524141,  2.54241164,  4.68881525,
        3.21084206, -2.32730223,  1.34133166,  2.64892605,  3.00587893,
        8.46573057,  2.062368  , -2.73742365,  0.23734731, 10.27703726,
        9.28888269, -1.69906732,  2.06873192,  1.67734762, -1.0767674 ])
```

[3]:
```
#(b)
print("mean:",rand.mean())
print("sd:",np.sqrt(rand.var()))
```

```
mean: 2.0234626420924355
sd: 3.9154491438674213
```

[4]:
```
#(c)
plt.figure(figsize=(8,3))
plt.hist(rand)
plt.show()
```



2번

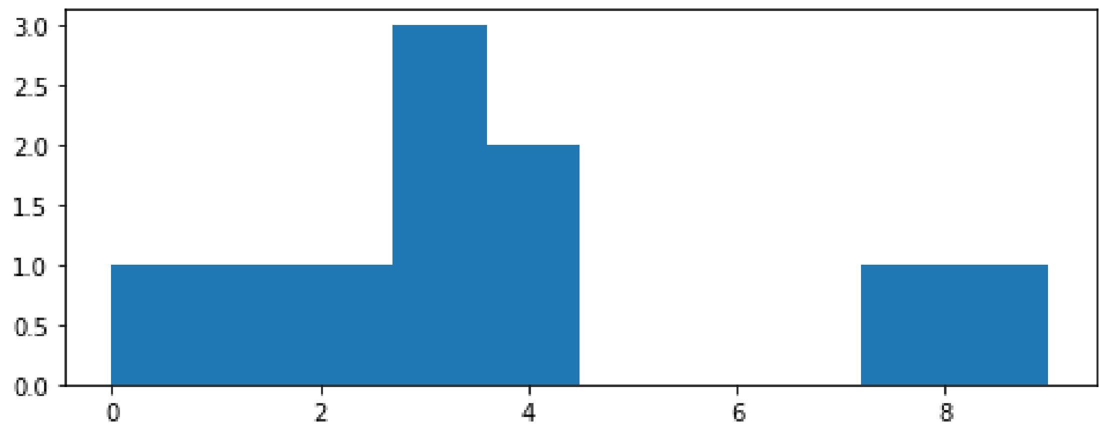
[5]:
```python
#(a)
rand_p=sp.stats.poisson.rvs(mu=3, size=10)
rand_p
```

[5]:
```
array([4, 2, 3, 8, 3, 3, 9, 1, 4, 0])
```

[6]:
```python
#(b)
print("mean:",rand_p.mean())
print("sd:",np.sqrt(rand_p.var()))
```

```
mean: 3.7
sd: 2.685144316419511
```

[7]:
```python
#(c)
plt.figure(figsize=(8,3))
plt.hist(rand_p)
plt.show()
```



3번

[8]:
```
#(a)
sp.stats.norm.ppf(loc=10,scale=3,q=0.6)
```

[8]: 10.760041309407399

[9]:
```
#(b)
sp.stats.norm.sf(loc=10,scale=3,x=12)
```

[9]: 0.2524925375469229

[2]:
```
#(c)
sp.stats.norm.cdf(loc=10,scale=3,x=11.5)-sp.stats.norm.cdf(loc=10,scale=3,x=-11.
↪5)
```

[2]: 0.6914624612736289

4번

[11]:
```
#(a)
#P(-b<=Z<=b)=2P(0<Z<=b)=0.90
#P(0<=Z<=b)=0.45  =>  P(Z<=b)=0.95
sp.stats.norm.ppf(loc=0,scale=1,q=0.95)
```

[11]: 1.6448536269514722

[12]:
```
#(b)
#P(-c<=Z<=c)=2P(0<Z<=c)=0.95
#P(0<=Z<=c)=0.475  =>  P(Z<=c)=0.975
sp.stats.norm.ppf(loc=0,scale=1,q=0.975)
```

[12]: 1.959963984540054

5번

[13]:
```
1-sp.stats.binom.cdf(n=5,p=1/6,k=2)
```

[13]: 0.03549382716049376

6번

[14]:
```
sp.stats.binom.cdf(n=10000,p=0.5,k=5020)-sp.stats.binom.cdf(n=10000,p=0.
 ↪5,k=4979)
```

[14]: 0.3181919670623352

7번

[15]:
```
#(a)
sp.stats.t.sf(df=10,x=2.5)
```

[15]: 0.015723422118304388

[16]:
```
#(b)
sp.stats.t.cdf(df=10,x=-2.5)
```

[16]: 0.015723422118304388

[17]:
```
#(c)
2*sp.stats.t.sf(df=10,x=1.8)
```

[17]: 0.10205224313467903

[18]:
```
#(d)
sp.stats.t.sf(df=10,x=2.5)
```

[18]: 0.015723422118304388

[19]:
```
#(e)
sp.stats.t.cdf(df=10,x=-2.5)
```

[19]: 0.015723422118304388

[20]:
```
#(f)
sp.stats.t.cdf(df=10,x=1.5)-sp.stats.t.cdf(df=10,x=-1)
```

[20]: 0.7472997706262499

8번

[3]:
```
#(a)
1-sp.stats.poisson.cdf(mu=3,k=2)
```

[3]: 0.5768099188731564

[4]:
```
#(b)
sp.stats.poisson.cdf(mu=3,k=3)
```

[4]: 0.6472318887822313

[5]:
```
#(c)
sp.stats.poisson.cdf(mu=3,k=7)-sp.stats.poisson.cdf(mu=3,k=1)
```

[5]: 0.7889472226721868

[6]:
```
#(d)
sp.stats.poisson.cdf(mu=3,k=4)
```

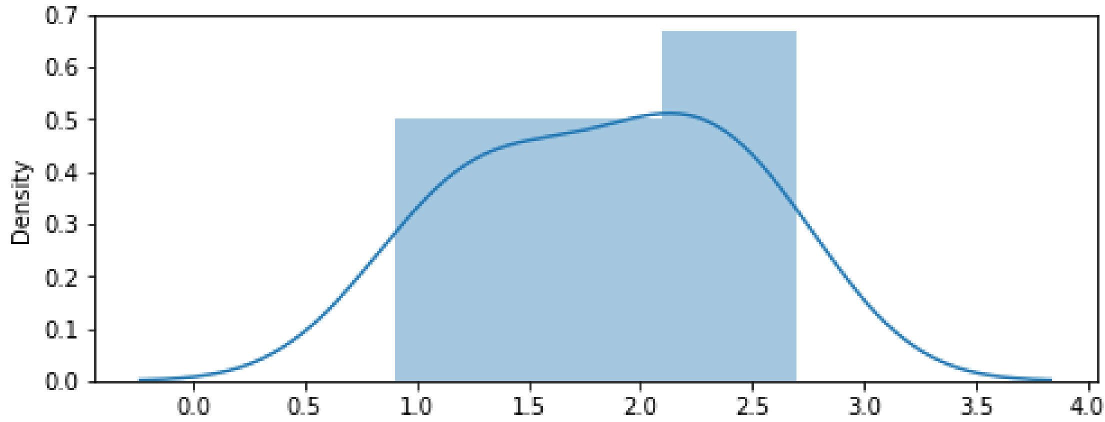
[6]: 0.8152632445237722

9번

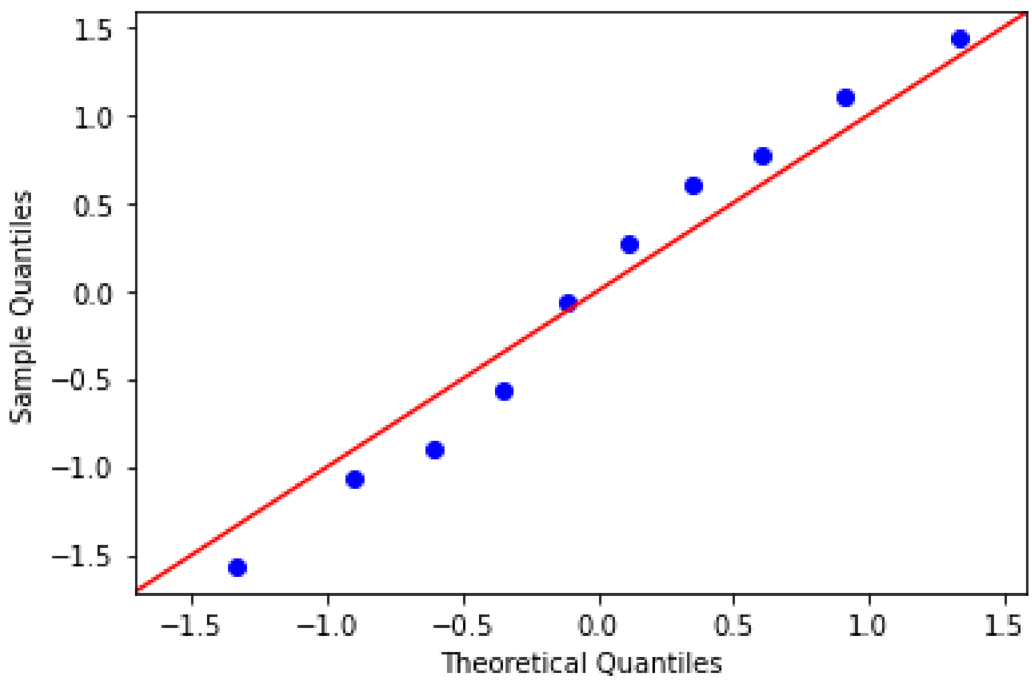
[19]:
```
data=np.array([1.5,2.2,0.9,1.3,2.0,1.2,2.5,2.7,1.8,2.3])
```

[20]:
```
#(a)
plt.figure(figsize=(8,3))
sns.distplot(data)
plt.show()
```

```
C:\Users\DS\anaconda3\lib\site-packages\seaborn\distributions.py:2551:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```



[22]:
```
#(b)
data1=(data-np.mean(data))/np.std(data,ddof=1)
sm.qqplot(data1,line='45')

py.show()
```



[29]:
```
#(c)
sp.stats.shapiro(data)
```

```
# H0:정규분포를 따른다/ H1:정규분포를 따르지 않는다
# 검정통계량=0.96이며 pvalue=0.834로 pvalue>0.05이므로 유의수준 0.05에서 귀무가설을 기각하지 못한다.
# 따라서 정규분포를 따른다고 할 수 있다.
```

[29]: (0.9642989635467529, 0.8335516452789307)

[30]:
```
#(d)
print("mean:",data.mean())
print("var:",data.var())
print("sd:",np.sqrt(data.var()))
```

```
mean: 1.8400000000000003
var: 0.32439999999999997
sd: 0.5695612346359257
```

10 번

[23]:
```
data1=np.array([182,167,166,159,178,176,169,163,166,181,
                171,182,172,186,171,166,170,168,154,173,
                174,166,160,162,161,179,147,162,170,166,
                165,178,171,169,183,149,168,177,170,163])
```

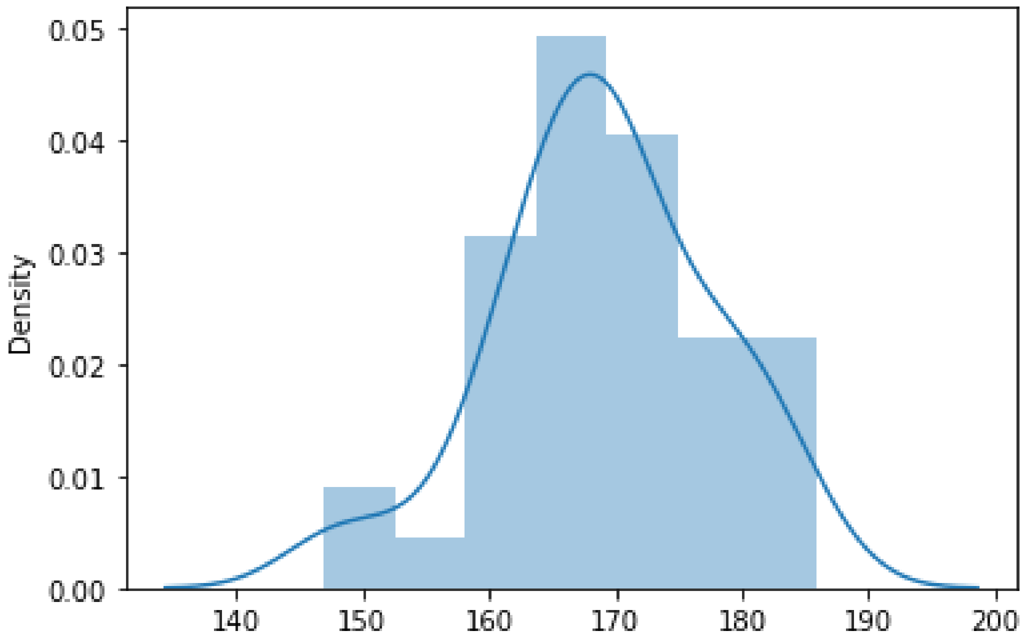
[24]:
```
#(a)
print("mean:",data1.mean())
print("var:",data1.var())
print("sd:",np.sqrt(data1.var()))
print("range:",(data1.max()-data1.min()))
```

```
mean: 169.0
var: 74.95
sd: 8.657366805212773
range: 39
```

[25]:
```
#(b)
sns.distplot(data1)
```

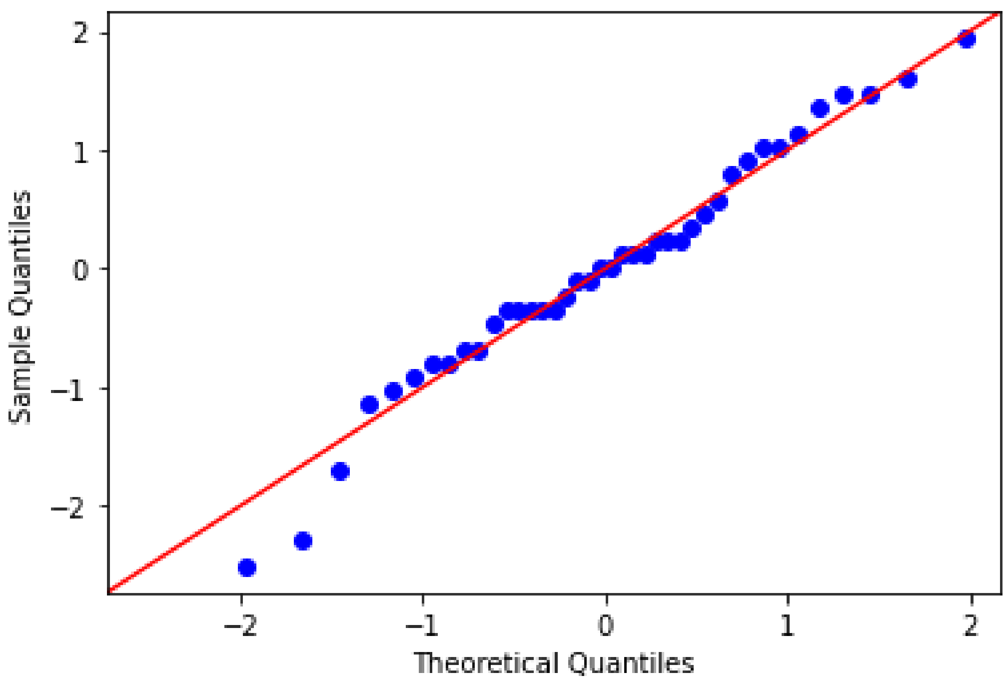
```
C:\Users\DS\anaconda3\lib\site-packages\seaborn\distributions.py:2551:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

[25]: <AxesSubplot:ylabel='Density'>



[27]:
```
#(c)
data1_1=(data1-np.mean(data1))/np.std(data1,ddof=1)
sm.qqplot(data1_1,line='45')
sp.stats.shapiro(data1)
# H0:정규분포를 따른다/ H1:정규분포를 따르지 않는다
# 검정통계량=0.97이며 pvalue=0.499>0.05이므로 유의수준 0.05에서 귀무가설을 기각하지 못한다.
# 따라서 정규분포를 따른다고 할 수 있다.
```

[27]: ShapiroResult(statistic=0.9746835231781006, pvalue=0.4994093179702759)



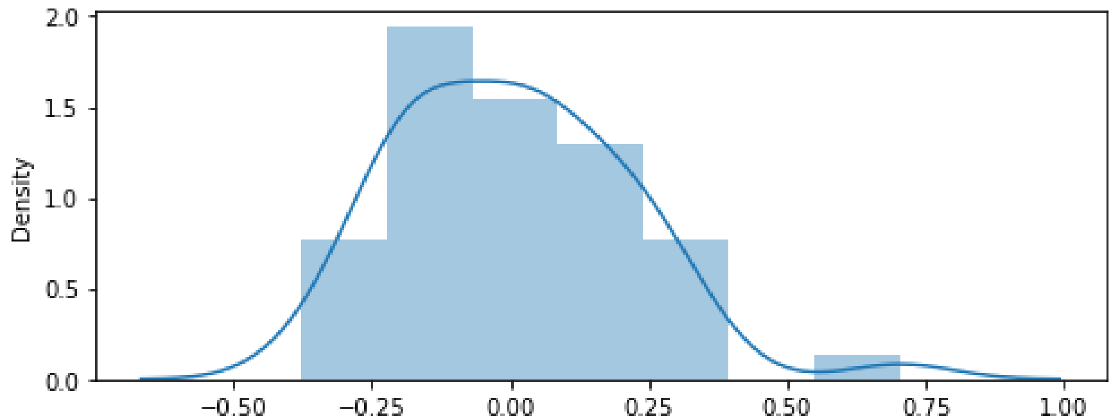
11번

[5]:
```
#(a)
x=sp.stats.norm.rvs(loc=0,scale=np.sqrt(2),size=30)
x
```

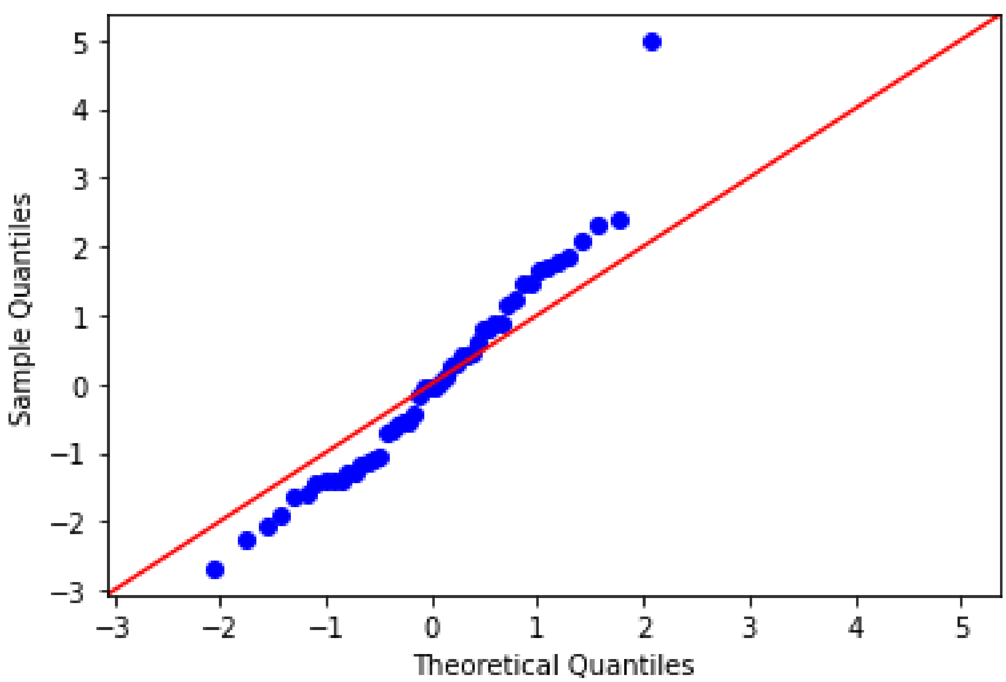
[5]:
```
array([ 0.08100433, -0.83719537,  0.18487571,  0.17713111,  0.75864781,
       -0.94479036,  2.58963453,  0.07389862,  1.64328447, -1.10102411,
        1.63087002, -0.94247968, -0.8353216 , -3.9031779 ,  0.48416643,
        2.3534267 ,  0.04156127, -2.31850429, -0.6928643 ,  0.30735224,
       -0.7739488 ,  0.67041244,  0.41471911,  0.27686362,  1.40346222,
        0.1553217 , -1.59487097, -1.74281854,  0.04666822, -1.55823161])
```

[146]:
```
#(b)
np.random.seed(234)
mean1=np.zeros(50)

for i in range(50):
    x=sp.stats.norm.rvs(loc=0,scale=np.sqrt(2),size=30)
    mean1[i]=(np.mean(x))

plt.figure(figsize=(8,3))
sns.distplot(mean1)
plt.show()
```

```
C:\Users\DS\anaconda3\lib\site-packages\seaborn\distributions.py:2551:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```



[147]:
```
#(c)
#xbar~N(0,1/50)
data1_1=mean1*np.sqrt(len(mean1))
sm.qqplot(data1_1,line='45')
py.show()
```

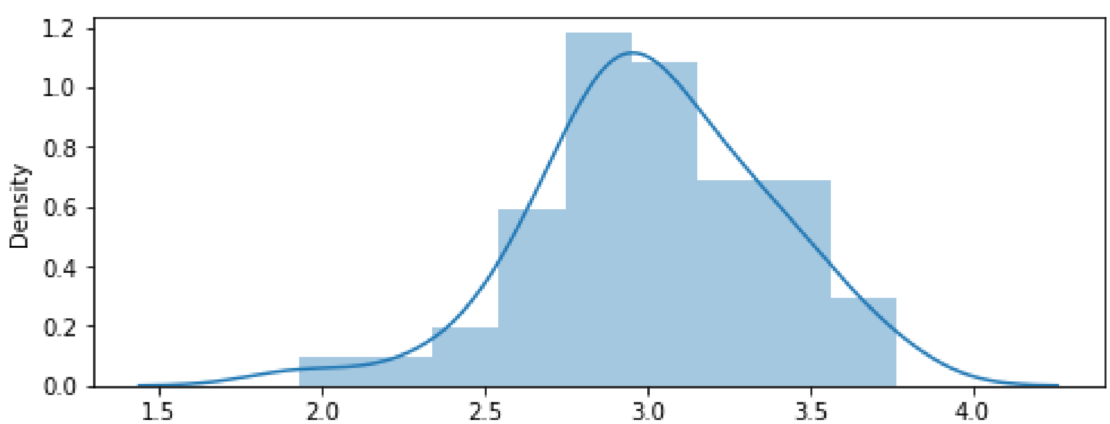


[143]:
```
#(d)
y=sp.stats.poisson.rvs(mu=3,size=30)
y
```

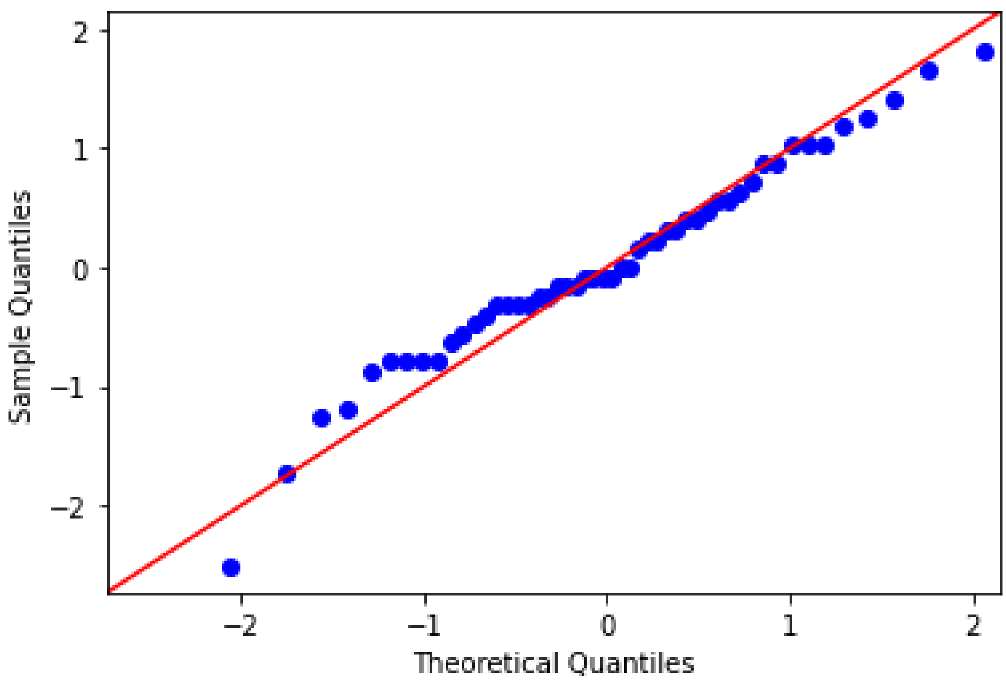
[143]:
```
array([1, 0, 2, 6, 2, 3, 5, 3, 3, 3, 1, 2, 8, 3, 4, 3, 3, 2, 3, 2, 0, 2,
       5, 3, 5, 2, 2, 1, 1, 4])
```

[144]:
```
#(e)
mean_p=np.zeros(50)

for i in range(50):
    y=sp.stats.poisson.rvs(mu=3,size=30)
    mean_p[i]=y.mean()

plt.figure(figsize=(8,3))
sns.distplot(mean_p)
plt.show()
```

```
C:\Users\DS\anaconda3\lib\site-packages\seaborn\distributions.py:2551:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

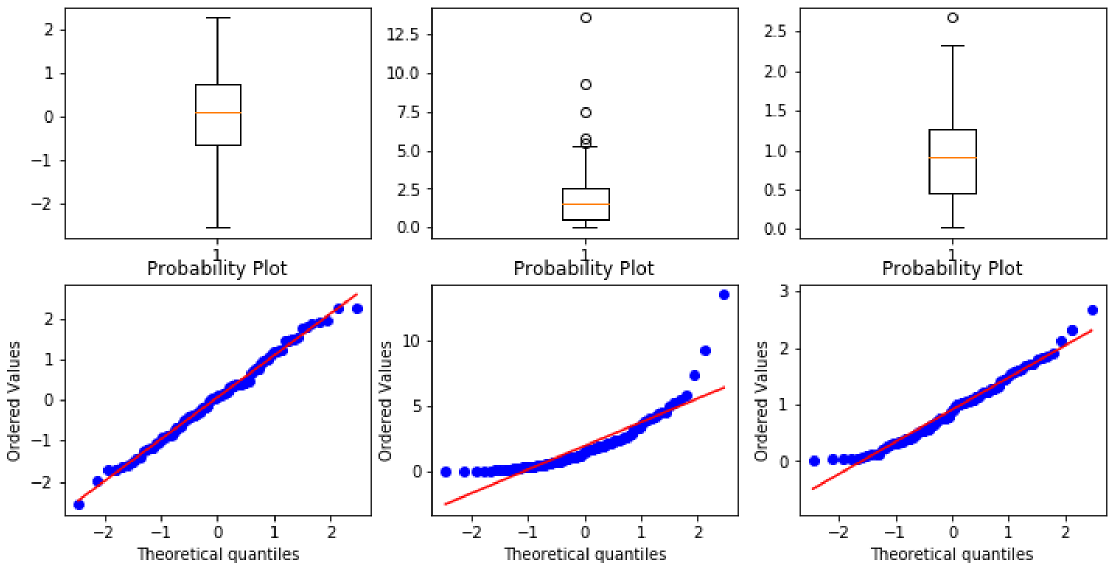


[145]:
```
#(f)
datap_1=(np.sqrt(len(mean_p))*(mean_p-3))/3
sm.qqplot(datap_1,line='45')
plt.show()
```



12번

[45]:
```
from scipy.stats import probplot
np.random.seed(0)
n=100
x1=np.random.normal(0,1,n)
x2=np.random.exponential(2,n)
x3=np.log1p(x2)

f,axes=plt.subplots(2,3,figsize=(12,6))
axes[0][0].boxplot(x1)
probplot(x1,plot=axes[1][0])
axes[0][1].boxplot(x2)
probplot(x2,plot=axes[1][1])
axes[0][2].boxplot(x3)
probplot(x3,plot=axes[1][2])

plt.axis("equal")
plt.show()
```


Probability Plot
Probability Plot
Probability Plot
Ordered Values
Theoretical quantiles

5장 연습문제

[1]:
```
import numpy as np
import math as m
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
import scipy as sp
import statsmodels.formula.api as smf
import statsmodels.api as sm
import pylab as py
```

[2]:
```
# 줄기 잎 그림을 위한 import
import sys
!{sys.executable} -m pip install stemgraphic
import stemgraphic
```

```
Collecting stemgraphic
  Downloading stemgraphic-0.9.1-py3-none-any.whl (61 kB)
Requirement already satisfied: matplotlib in c:\users\ds\anaconda3\lib\site-
packages (from stemgraphic) (3.3.2)
Requirement already satisfied: seaborn in c:\users\ds\anaconda3\lib\site-
packages (from stemgraphic) (0.11.0)
Collecting docopt
  Downloading docopt-0.6.2.tar.gz (25 kB)
Requirement already satisfied: pandas in c:\users\ds\anaconda3\lib\site-packages
(from stemgraphic) (1.1.3)
Requirement already satisfied: certifi>=2020.06.20 in
c:\users\ds\anaconda3\lib\site-packages (from matplotlib->stemgraphic)
(2020.6.20)
Requirement already satisfied: kiwisolver>=1.0.1 in
c:\users\ds\anaconda3\lib\site-packages (from matplotlib->stemgraphic) (1.3.0)
Requirement already satisfied: numpy>=1.15 in c:\users\ds\anaconda3\lib\site-
packages (from matplotlib->stemgraphic) (1.19.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in
c:\users\ds\anaconda3\lib\site-packages (from matplotlib->stemgraphic) (2.4.7)
Requirement already satisfied: cycler>=0.10 in c:\users\ds\anaconda3\lib\site-
packages (from matplotlib->stemgraphic) (0.10.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\ds\anaconda3\lib\site-
packages (from matplotlib->stemgraphic) (8.0.1)
```

```
Requirement already satisfied: python-dateutil>=2.1 in
c:\users\ds\anaconda3\lib\site-packages (from matplotlib->stemgraphic) (2.8.1)
Requirement already satisfied: scipy>=1.0 in c:\users\ds\anaconda3\lib\site-
packages (from seaborn->stemgraphic) (1.5.2)
Requirement already satisfied: pytz>=2017.2 in c:\users\ds\anaconda3\lib\site-
packages (from pandas->stemgraphic) (2020.1)
Requirement already satisfied: six in c:\users\ds\anaconda3\lib\site-packages
(from cycler>=0.10->matplotlib->stemgraphic) (1.15.0)
Building wheels for collected packages: docopt
  Building wheel for docopt (setup.py): started
  Building wheel for docopt (setup.py): finished with status 'done'
  Created wheel for docopt: filename=docopt-0.6.2-py2.py3-none-any.whl
size=13709
sha256=ef253f8e00522abe529e28dd3b58c3bfb3d981499420bc0ed1f37b32df1d57d8
  Stored in directory: c:\users\ds\appdata\local\pip\cache\wheels\56\ea\58\ead13
7b087d9e326852a851351d1debf4ada529b6ac0ec4e8c
Successfully built docopt
Installing collected packages: docopt, stemgraphic
Successfully installed docopt-0.6.2 stemgraphic-0.9.1
```

1번

[3]:
```
experiment=np.arange(1,21)
count=np.array([10,12,20,14,17,20,14,13,11,17,21,11,16,14,17,2,0,1,7,2])
A=np.tile('A',8)
B=np.tile('B',7)
C=np.tile('C',5)
spray=np.concatenate((A,B,C),axis=0)

data=pd.DataFrame({'experiment':experiment, 'count':count,'spray':spray})
data.head()
```

[3]:

	experiment	count	spray
0	1	10	A
1	2	12	A
2	3	20	A
3	4	14	A
4	5	17	A

[4]:
```
#(a)
pd.crosstab(index=data["spray"],columns="count")
```

[4]:

col_0	count
spray	
A	8
B	7
C	5

[5]:
```python
#(b)
data.loc[:,['count']].mean()
```

[5]:
```
count    11.95
dtype: float64
```

[6]:
```python
#(c)
data.groupby(by="spray").sum()
```

[6]:

spray	experiment	count
A	36	120
B	84	107
C	90	12

2번

[7]:
```python
river=np.array([735,320,325,392,524,450,1459,135,465,600,330,336,280,
                315,870,906,202,329,290,1000,600,505,1450,840,1243,890,
                350,407,286,280])
```

[8]:
```python
#(a)
print("mean:",river.mean())
print("median",np.median(river))
```

```
mean: 570.4666666666667
median 428.5
```

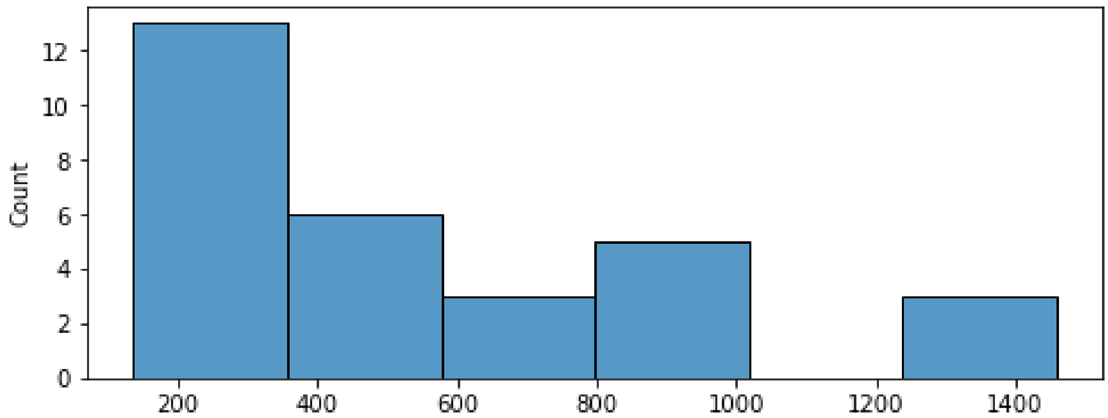
[9]:
```python
#(b)
print("var:",river.var())
print("sd:",np.sqrt(river.var()))
print("quantile range:",(np.percentile(river,75)-np.percentile(river,25)))
```

```
var: 125269.18222222221
sd: 353.93386701786847
quantile range: 492.5
```

[10]:
```python
#(c)
np.percentile(river,[15,45,80])
```

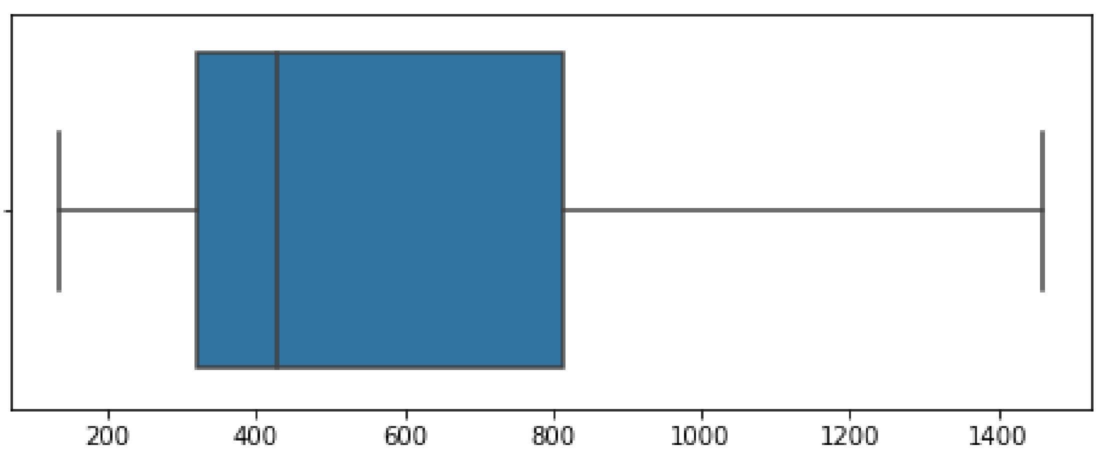
[10]:
```
array([287.4 , 392.75, 874.  ])
```

[37]:
```python
#(d)
plt.figure(figsize=(8,3))
sns.histplot(river)
plt.show()
```



[12]:
```
#(e)
plt.figure(figsize=(8,3))
sns.boxplot(river)
plt.show()
```

```
C:\Users\DS\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```



3번

[13]:
```
bulb=np.array([25,16,44,62,36,58,38])
```

[14]:
```
#(a)
bulb.mean()
```

[14]: 39.857142857142854

[15]:
```
#(b)
bulb.var()
```

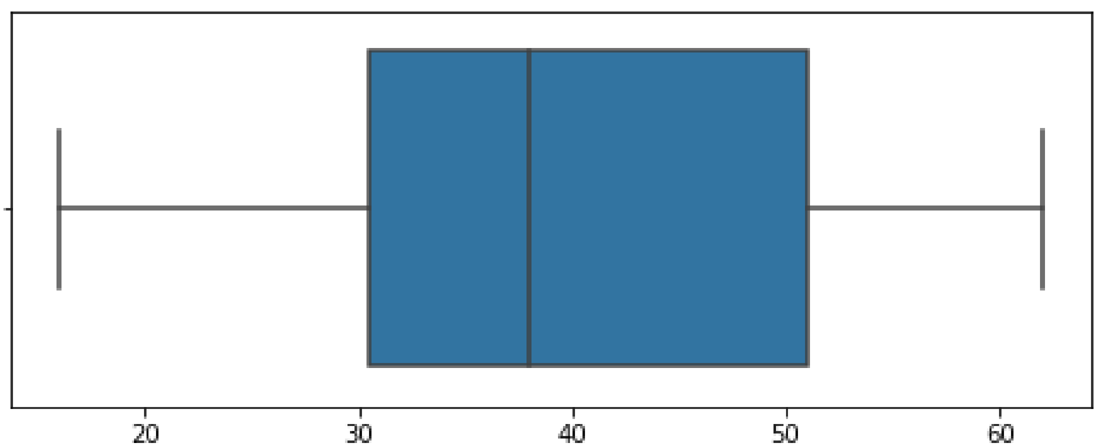
[15]: 234.9795918367347

[16]:
```
#(c)
np.sqrt(bulb.var())
```

[16]: 15.32904406141279

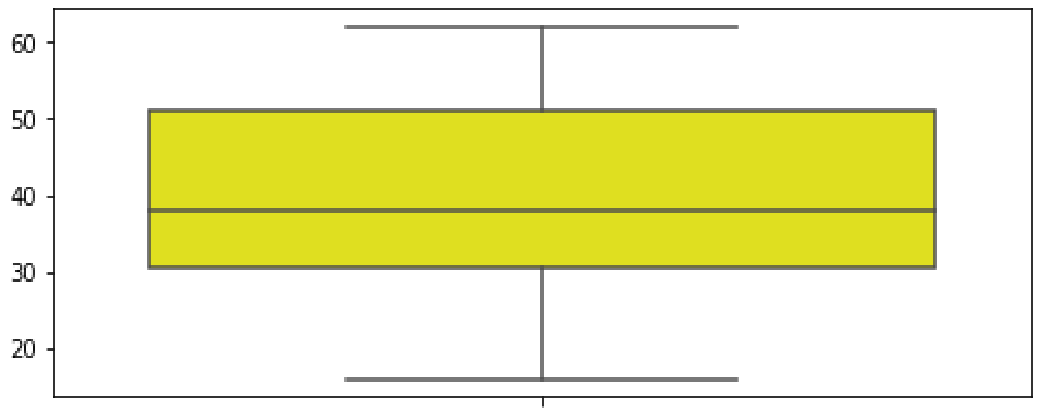
[17]:
```
#(d)
plt.figure(figsize=(8,3))
sns.boxplot(bulb)
plt.show()
```

```
C:\Users\DS\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```



[18]:
```
#(e)
plt.figure(figsize=(8,3))
sns.boxplot(bulb,orient="v",color="yellow")
plt.show()
```

```
C:\Users\DS\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```



[19]:
```
#(f)
stemgraphic.stem_graphic(bulb,scale=10)
```

[19]:
```
(<Figure size 540x162 with 1 Axes>,
 <matplotlib.axes._axes.Axes at 0x160c299e280>)
```

4번

[20]:
```
data=sp.stats.uniform.rvs(loc=0,scale=1,size=30)
data
```

[20]:
```
array([0.18228122, 0.47543264, 0.49732022, 0.73170902, 0.24995905,
       0.91639377, 0.76573231, 0.03886928, 0.05193703, 0.06340538,
       0.93126553, 0.68976889, 0.51051159, 0.62209063, 0.76018355,
       0.70127616, 0.10787085, 0.12634677, 0.82234153, 0.34757319,
       0.83915286, 0.93330428, 0.57712905, 0.9909705 , 0.9962126 ,
       0.15061687, 0.47894984, 0.98444165, 0.96034698, 0.9135294 ])
```

[21]:
```
#(a)
m=data.mean()
s=np.sqrt(data.var())

print("mean:",m)
print("sd:",s)
```

```
mean: 0.5805640873604927
sd: 0.32521456564625867
```

[22]:
```
#(b)
n=len(data)

lower=m-2*s/np.sqrt(n)
upper=m+2*s/np.sqrt(n)

print(lower); print(upper)
```

```
0.461812518271359
0.6993156564496263
```

5번

[39]:
```
data1=np.array([19,21,15,23,24,15,15,15,16,29,
               18,32,20,23,24,24,25,25,25,25,
               25,25,25,36,26,28,30])
```

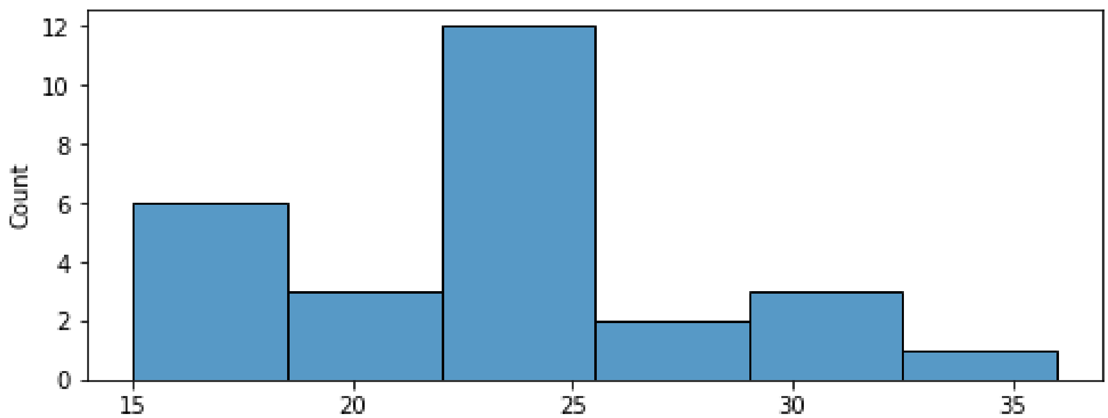
[24]:
```
#(a)
print("mean:",np.mean(data1))
print("median:",np.median(data1))
```

```
mean: 23.25925925925926
median: 24.0
```

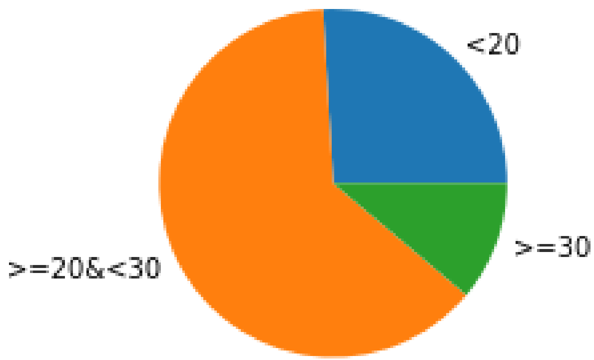
[25]:
```
#(b)
print("var:",np.var(data1,ddof=1))
print("std:",np.std(data1,ddof=1))
print("range:",(np.max(data1)-np.min(data1)))
```

```
var: 29.122507122507122
std: 5.396527320648634
range: 21
```

[40]:
```
#(c)
plt.figure(figsize=(8,3))
sns.histplot(data1)
plt.show()
```



[27]:
```
#(d)
pi1=data1[np.where(data1<20)]
pi2=data1[np.where((data1>=20)&(data1<30))]
pi3=data1[np.where(data1>=30)]

cat=('<20','>=20&<30','>=30')
freq=[len(pi1),len(pi2),len(pi3)]

plt.figure(figsize=(8,3))
plt.pie(freq,labels=cat)
plt.show()
```



6번

[28]:
```
data=[2.3,2.4,3.1,2.2,1.0,2.3,2.1,1.1,1.2,0.9,1.5,1.1]
```

[29]:
```
#(a)
print("mean:",np.mean(data))
print("median:",np.median(data))
```

```
mean: 1.7666666666666666
median: 1.8
```

[30]:
```
#(b)
print("var:",np.var(data,ddof=1))
print("std:",np.std(data,ddof=1))
print("range:",(np.max(data)-np.min(data)))
```

```
var: 0.5151515151515151
std: 0.7177405625652734
range: 2.2
```

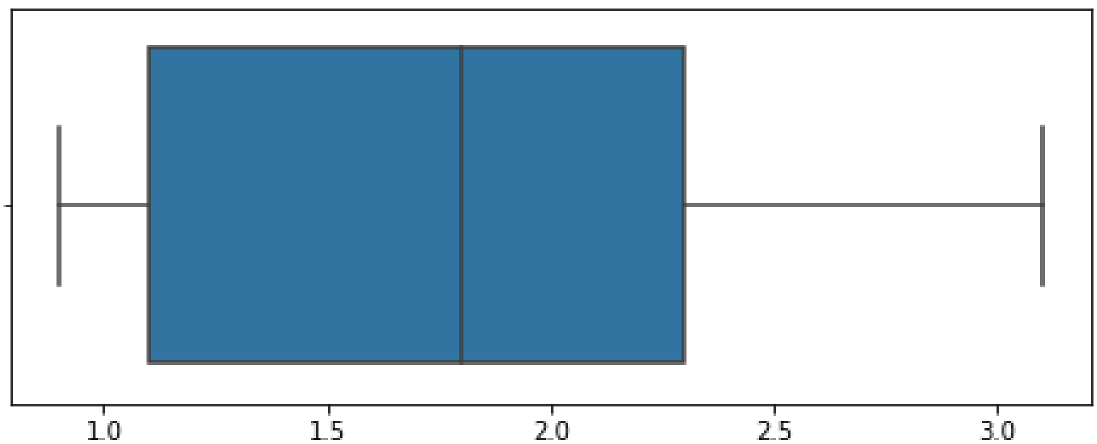
[31]:
```
#(c)
n=len(data)
m=np.mean(data)
sd=np.std(data,ddof=1)
cri=sp.stats.norm.ppf(loc=0,scale=1,q=0.975)

lower=m-cri*sd/np.sqrt(n)
upper=m+cri*sd/np.sqrt(n)

print(lower); print(upper)
```

```
1.3605741759833319
2.172759157350001
```

[33]:

```
#(d)
plt.figure(figsize=(8,3))
sns.boxplot(data)
plt.show()
```

```
C:\Users\DS\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```



[37]:

```
#(e)
stemgraphic.stem_graphic(data,scale=1)
plt.show()
```

7번

[38]:
```
smoke=np.array(['y','y','n','n'])
wrinkle=np.array(['y','n','y','n'])
freq=np.array([60,10,30,40])
df=pd.DataFrame({'smoke':smoke, 'wrinkle':wrinkle,'freq':freq}); df
```

[38]:

	smoke	wrinkle	freq
0	y	y	60
1	y	n	10
2	n	y	30
3	n	n	40

[39]:
```
#(a)
df1=df.loc[smoke=='y',:]; df1
print(df1)
p1=60/70
print(p1)
```

```
  smoke wrinkle  freq
0     y       y    60
1     y       n    10
0.8571428571428571
```

[40]:
```
#(b)
df2=df.loc[smoke=='n',:]; df2
print(df2)
p2=30/70
print(p2)
```

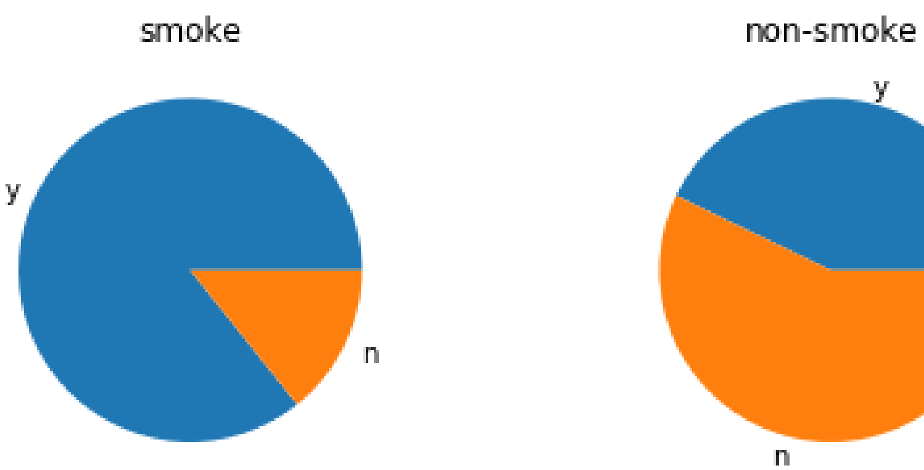
```
  smoke wrinkle  freq
2     n       y    30
3     n       n    40
0.42857142857142855
```

[41]:
```
#(c)
p1-p2
```

[41]: 0.42857142857142855

[42]:
```
#(d)
wrinkle1=np.array(['y','n'])
freq1=np.array([60,10])
freq2=np.array([30,40])

plt.figure(figsize=(8,3))
plt.subplot(1,2,1)
plt.pie(freq1,labels=wrinkle1)
plt.title("smoke")
```

```
plt.subplot(1,2,2)
plt.pie(freq2,labels=wrinkle1)
plt.title("non-smoke")

plt.show()
```



8번

[43]:
```
tv=[5.7,6.7,6.8,7.9,10.6,11.3,9.8,8.4,8.3,9.5,
   6.7,6.9,9.8,8.8,12.1,10.2,9.5,9.4,9.3,5.9]
```

[44]:
```
#(a)
m=np.mean(tv)
print("mean:",m)
```

```
mean: 8.680000000000001
```

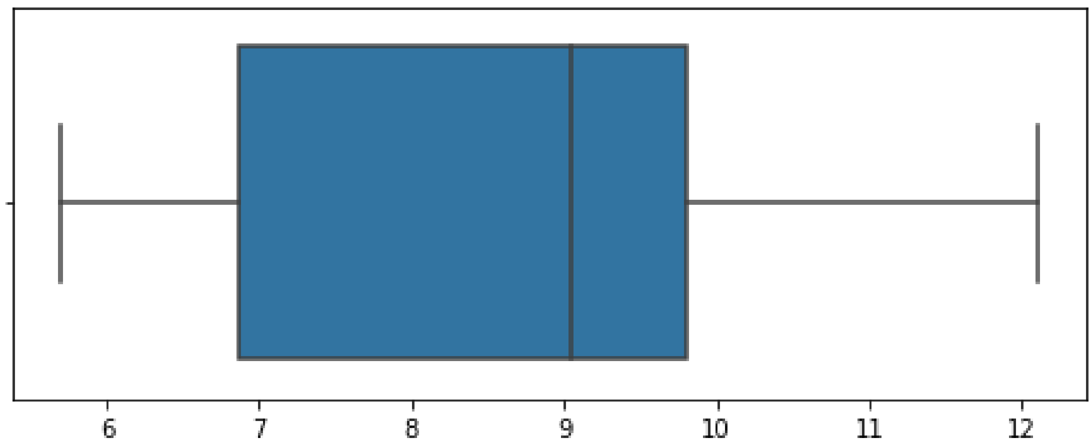
[45]:
```
#(b)
n=len(tv)
sd=np.std(tv,ddof=1)
cri=sp.stats.norm.ppf(loc=0,scale=1,q=0.975)

lower=m-cri*sd/np.sqrt(n)
upper=m+cri*sd/np.sqrt(n)

print(lower); print(upper)
```

```
7.892720334559037
9.467279665440966
```

[46]:
```
#(c)
print("var:",np.var(tv,ddof=1))
print("std:",np.std(tv,ddof=1))
print("range:",(np.max(tv)-np.min(tv)))
```

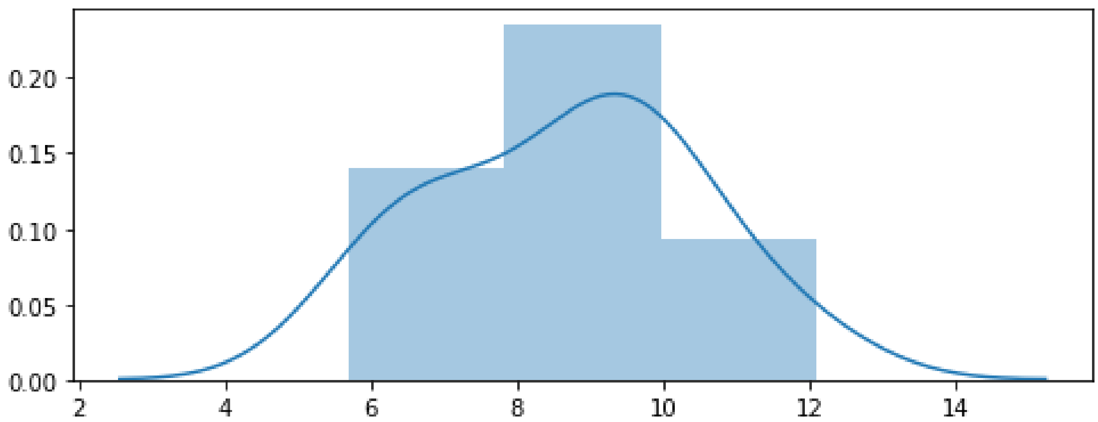
```
var: 3.2269473684210523
std: 1.7963706099858827
range: 6.3999999999999995
```

[47]:
```
#(d)
plt.figure(figsize=(8,3))
sns.boxplot(tv)
plt.show()
```



[48]:
```
#(e)
plt.figure(figsize=(8,3))
sns.distplot(tv)
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713:
FutureWarning: Using a non-tuple sequence for multidimensional indexing is
deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will
be interpreted as an array index, `arr[np.array(seq)]`, which will result either
in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



9번

[49]:
```
phone=[20870,39400,65000,45000,35890,29000,56770,
       23000,38550,59800,39880,56780,35200,48990]
```

[50]:
```
#(a)
m=np.mean(phone)
print("mean:",m)
```

```
mean: 42437.857142857145
```

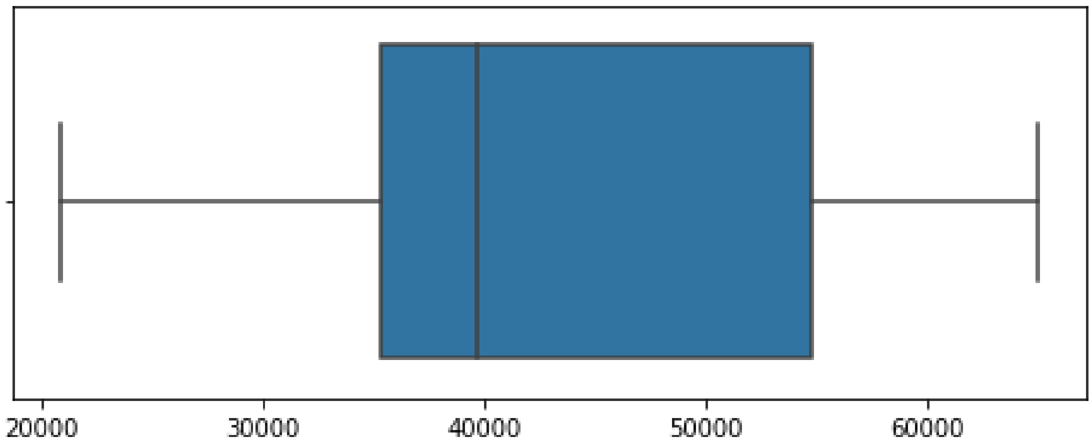
[51]:
```
#(b)
n=len(phone)
sd=np.std(phone,ddof=1)
cri=sp.stats.norm.ppf(loc=0,scale=1,q=0.975)

lower=m-cri*sd/np.sqrt(n)
upper=m+cri*sd/np.sqrt(n)

print(lower); print(upper)
```

```
35289.702546541055
49586.011739173235
```

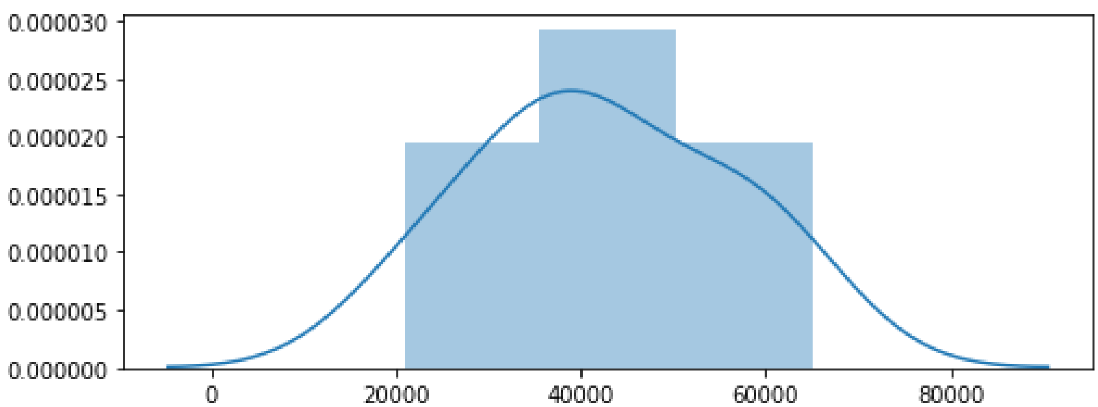
[52]:
```
#(c)
print("var:",np.var(phone,ddof=1))
print("std:",np.std(phone,ddof=1))
print("range:",(np.max(phone)-np.min(phone)))
```

```
var: 186217171.97802195
std: 13646.141285287278
range: 44130
```

[53]:
```
#(d)
plt.figure(figsize=(8,3))
sns.boxplot(phone)
plt.show()
```



[54]:
```
#(e)
plt.figure(figsize=(8,3))
sns.distplot(phone)
plt.show()
```



10번

[55]:
```
#(a)
80/200
```

[55]: 0.4

[56]:
```
#(b)
120/200
```

[56]: 0.6

11번

[57]:
```
sleep=[5.6,7.8,6.5,7.2,6.9,7.3,5.8,7.5,8.2,7.8]
```

[58]:
```
#(a)
m=np.mean(sleep)
print("mean:",m)
```

```
mean: 7.06
```

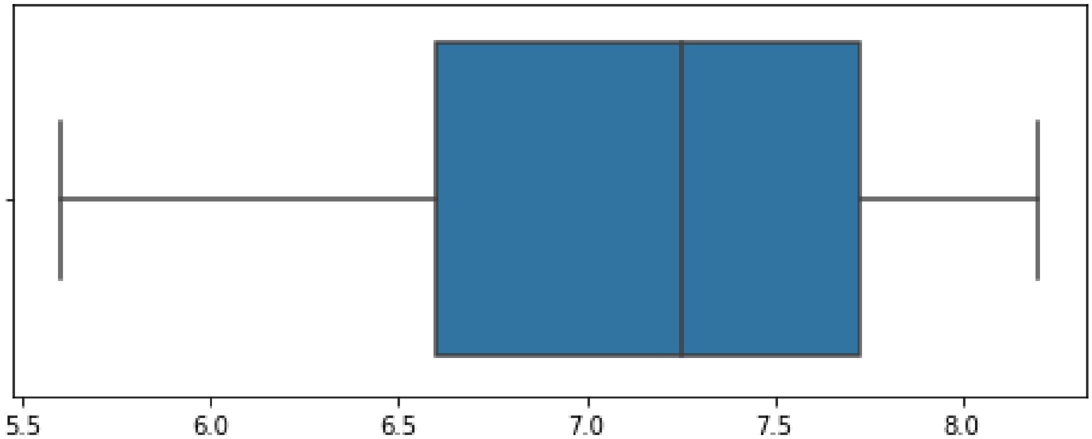
[59]:
```
#(b)
n=len(sleep)
sd=np.std(sleep,ddof=1)
cri=sp.stats.t.ppf(df=n-1,q=0.975)

lower=m-cri*sd/np.sqrt(n)
upper=m+cri*sd/np.sqrt(n)

print(lower); print(upper)
```

```
6.4416770421841285
7.678322957815871
```

[60]:
```
#(c)
print("var:",np.var(sleep,ddof=1))
print("std:",np.std(sleep,ddof=1))
print("range:",(np.max(sleep)-np.min(sleep)))
```

```
var: 0.7471111111111111
std: 0.864355893779357
range: 2.5999999999999996
```

[61]:
```
#(d)
plt.figure(figsize=(8,3))
sns.boxplot(sleep)
plt.show()
```



[62]:
```
#(e)
z=(sleep-np.mean(sleep))/np.std(sleep,ddof=1); z
```

[62]:
```
array([-1.68911904,  0.85612883, -0.64788128,  0.16197032, -0.18510894,
        0.2776634 , -1.45773287,  0.50904957,  1.31890117,  0.85612883])
```

6장 연습문제

[14]:
```
import numpy as np
import math as m
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
import scipy as sp
import statsmodels.formula.api as smf
import statsmodels.api as sm
```

1번

[3]:
```
fracture=pd.read_csv("C:/PythonbookData/table6.5_fracture.csv")
fracture.head()
```

[3]:

	gender	fracture	age	blood
0	1	1	42	35
1	1	1	42	43
2	1	1	38	35
3	1	1	35	33
4	1	1	25	31

[4]:
```
#(a)
frac_tab=pd.crosstab(index=fracture["gender"],columns=fracture["fracture"])
frac_tab
```

[4]:

fracture	1	2	3
gender			
1	6	4	0
2	0	7	5

[5]:
```
#(b)
fracture.groupby("gender").mean().loc[:,"blood"]
```

[5]:
```
gender
1    32.900000
2    38.833333
Name: blood, dtype: float64
```

[6]:
```
#(c)
fracture.groupby("fracture").mean().loc[:,"blood"]
```

[6]:
```
fracture
1    35.500000
2    32.545455
3    44.800000
Name: blood, dtype: float64
```

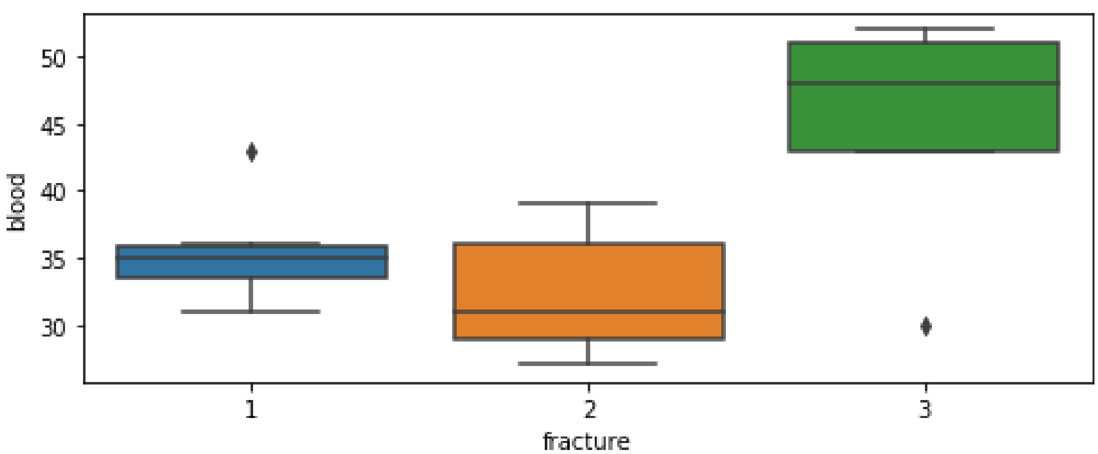
[53]:
```
#(d)
sp.stats.pearsonr(fracture['age'],fracture['blood'])[0]
```

[53]: -0.04325692537768141

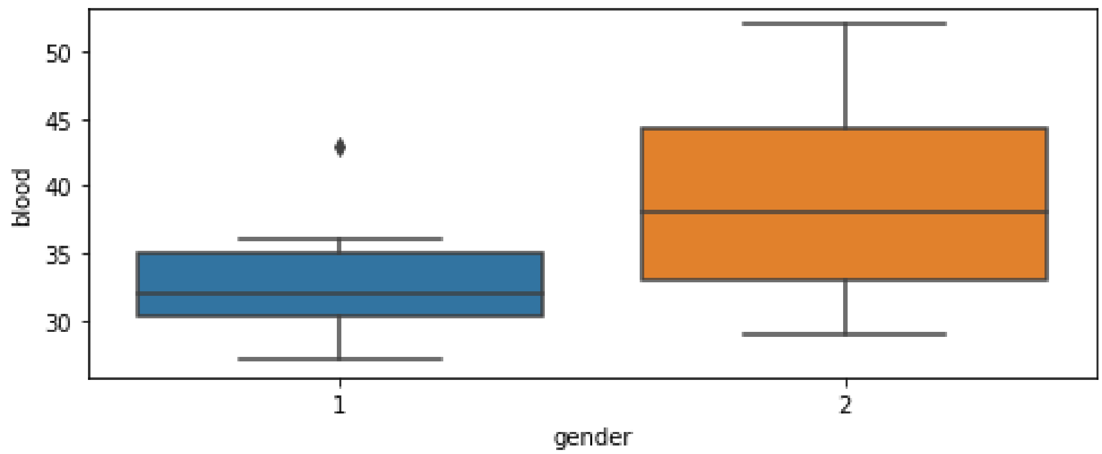
[54]:
```
#(e)
sp.stats.spearmanr(fracture['age'],fracture['blood'])[0]
```

[54]: -0.049645392068624034

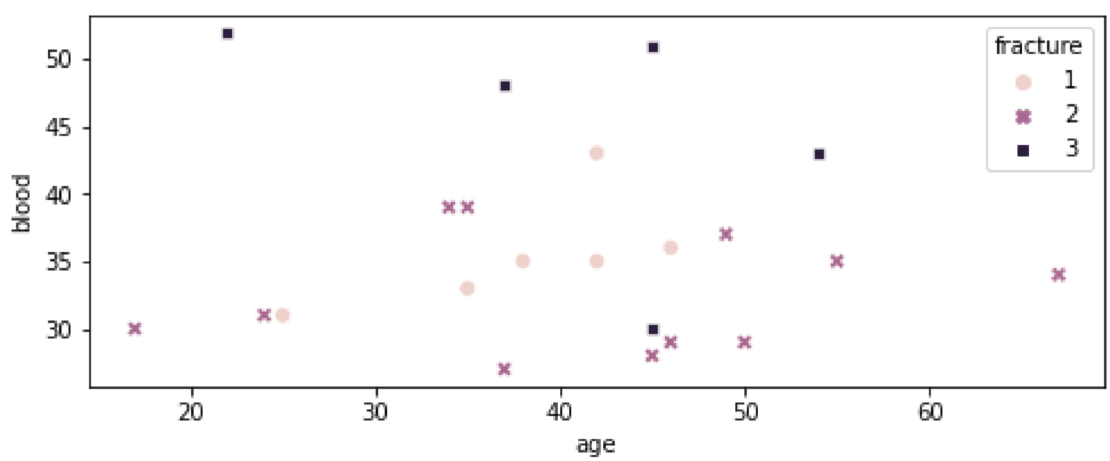
[55]:
```
#(f)
plt.figure(figsize=(8,3))
sns.boxplot(x='fracture',y='blood',data=fracture)
plt.show()
```



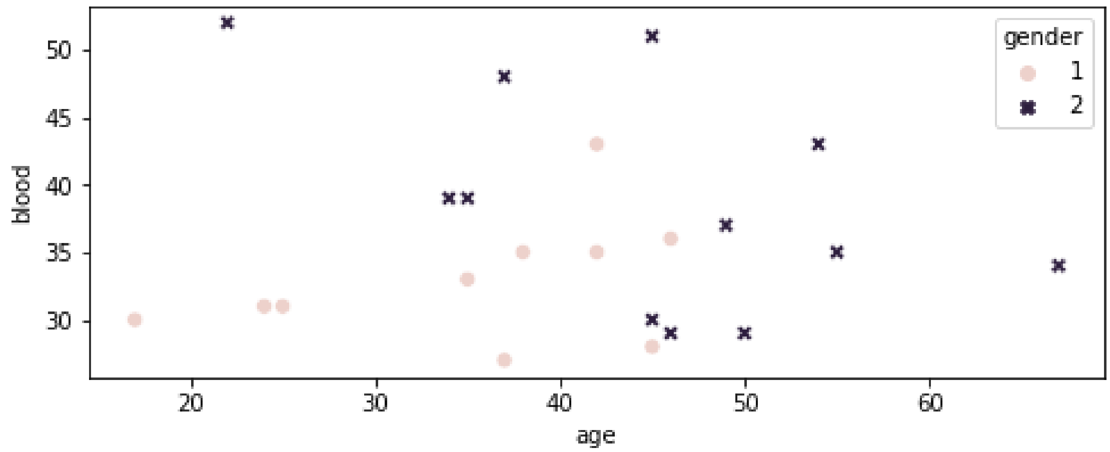
[56]:
```
#(g)
plt.figure(figsize=(8,3))
sns.boxplot(x='gender',y='blood',data=fracture)
plt.show()
```



[13]:
```
#(h)
plt.figure(figsize=(8,3))
sns.
↪scatterplot(x='age',y='blood',hue='fracture',style='fracture',s=50,data=fracture)
plt.legend(loc='upper right',title='fracture')
plt.show()
```



[12]:
```
#(i)
plt.figure(figsize=(8,3))
sns.
↪scatterplot(x='age',y='blood',hue='gender',style='gender',s=50,data=fracture)
plt.legend(loc='upper right',title='gender')
plt.show()
```



2번

[78]:
```
president=pd.read_csv("C:/PythonbookData/table6.6_president_election.csv")
president.head()
```

[78]:
```
            ism candidate  freq
0      progress         M   426
1        middle         M   543
2  conservative         M   130
3      progress         H    59
4        middle         H   373
```

[97]:
```
#(a)
cross=pd.
↪pivot_table(data=president,values="freq",aggfunc="sum",index="ism",columns="candidate")
cross
```

[97]:
```
candidate       A    H    M
ism
conservative  245  625  130
middle        674  373  543
progress      215   59  426
```

[84]:
```
#(b)
president['rate']=president['freq']/np.sum(president['freq'])
cross_p=pd.pivot_table(data=president,values="rate",
↪aggfunc="sum",index="ism",columns= "candidate",margins=True)
 cross_p
```

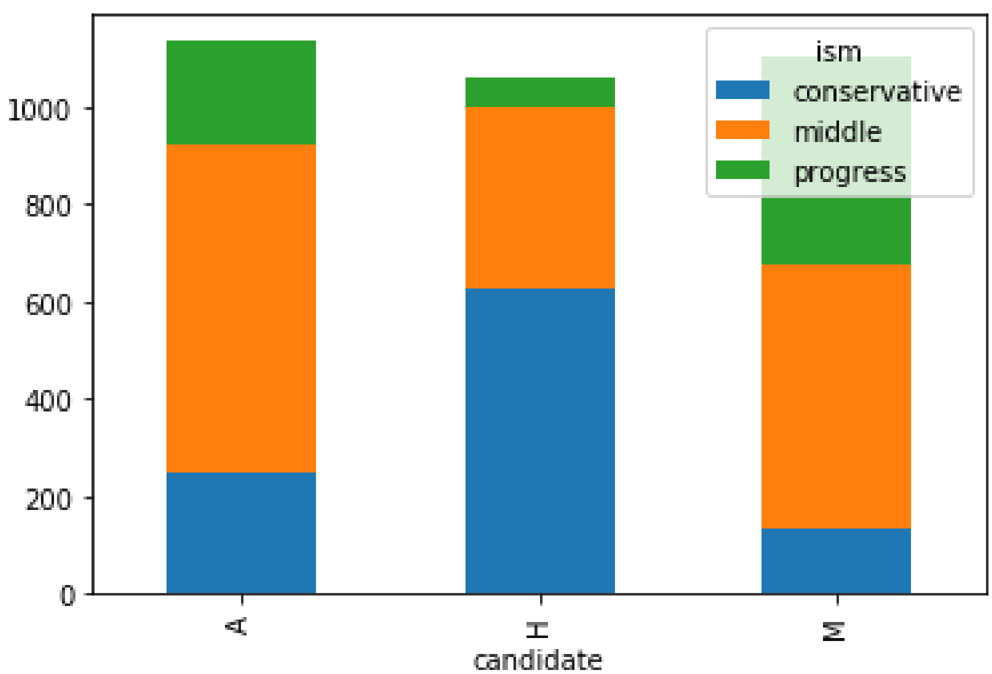
[84]:
```
candidate            A         H         M       All
ism
conservative  0.074468  0.189970  0.039514  0.303951
middle        0.204863  0.113374  0.165046  0.483283
progress      0.065350  0.017933  0.129483  0.212766
All           0.344681  0.321277  0.334043  1.000000
```

[98]:
```
#(c)
cross_c=pd.
↪pivot_table(data=president,values="freq",aggfunc="sum",index="ism",columns="candidate",
 margins=True)
cross_c
```

[98]:
```
candidate        A     H     M   All
ism
conservative   245   625   130  1000
middle         674   373   543  1590
progress       215    59   426   700
All           1134  1057  1099  3290
```

[101]:
```
#(d)
cross_b=pd.
 ↪pivot_table(data=president,values="freq",aggfunc="sum",index="candidate",columns="ism")
plt.figure(figsize=(8,3))
cross_b.plot(kind="bar",stacked=True)
plt.show()
```

```
&lt;Figure size 576x216 with 0 Axes&gt;
```



3번

[102]:
```
x=[147,158,131,142,180]
y=[122,128,125,123,115]
```

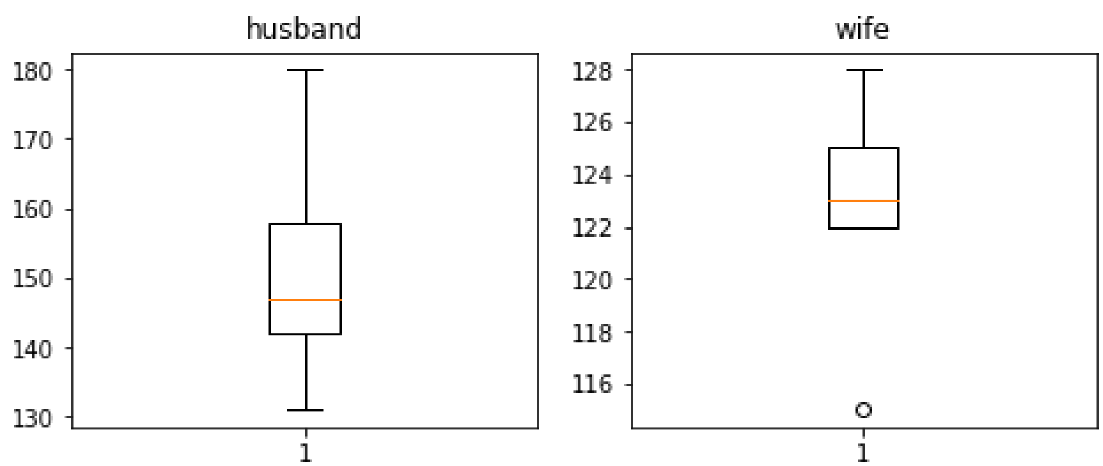
[103]:
```
#(a)
sp.stats.pearsonr(x,y)[0]
#약한 음의 상관관계를 갖는다.
```

[103]: -0.6451329649617566

[105]:
```
#(b)
sp.stats.spearmanr(x,y)[0]
#약한 음의 순위상관성을 갖는다.
```

[105]: -0.39999999999999997

[106]:
```
#(c)
plt.figure(figsize=(8,3))
plt.subplot(1,2,1)
plt.boxplot(x)
plt.title("husband")

plt.subplot(1,2,2)
plt.boxplot(y)
plt.title("wife")
plt.show()
```



4번

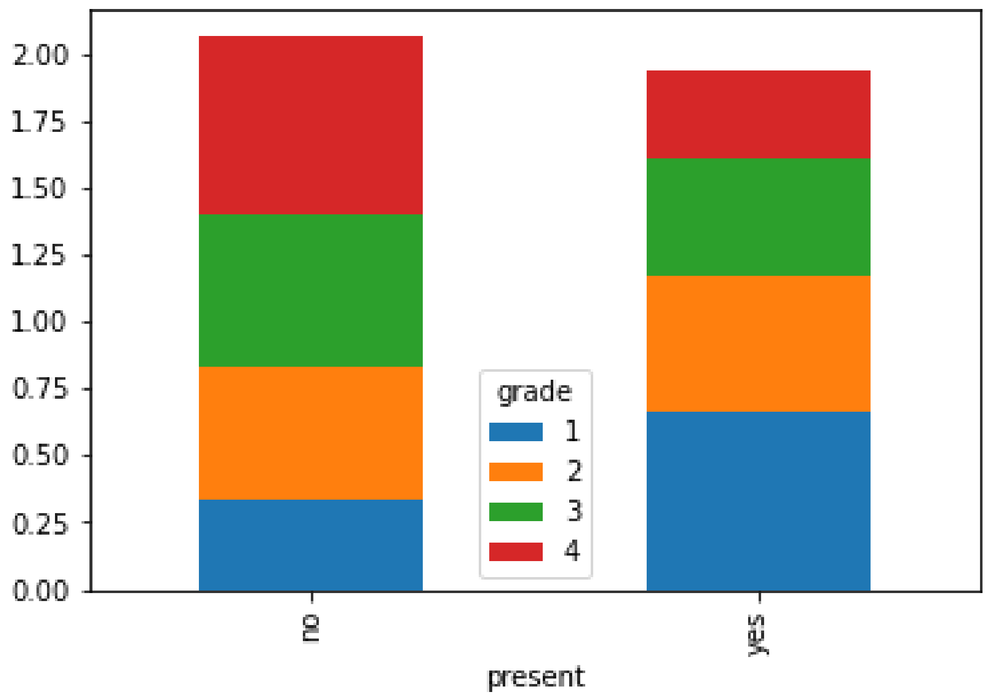
[16]:
```
event=pd.read_csv("C:/PythonbookData/problem8.3_hangsa.csv")
event.head()

#주어진 데이터를 사용하지 않을 때
#present=['y','y','y','y','n','n','n','n']
#freq=[40,30,35,20,20,30,45,40]
#grade=[1,2,3,4,1,2,3,4]
#event=pd.DataFrame({'grade':grade,'present':present,'freq':freq})
```

[16]:
```
   grade present  freq
0      1     yes    40
1      2     yes    30
2      3     yes    35
3      4     yes    20
4      1      no    20
```

[17]:
```
#(a)
event['rate1']=[40/60,30/60,35/80,20/60,20/60,30/60,45/80,40/60]

e_cross=pd.
 ↪pivot_table(data=event,values="rate1",aggfunc="sum",index="present",columns="grade")
print(e_cross)

plt.figure(figsize=(8,3))
e_cross.plot(kind="bar",stacked=True)
plt.show()
```

```
grade           1    2       3         4
present
no       0.333333  0.5  0.5625  0.666667
yes      0.666667  0.5  0.4375  0.333333
```

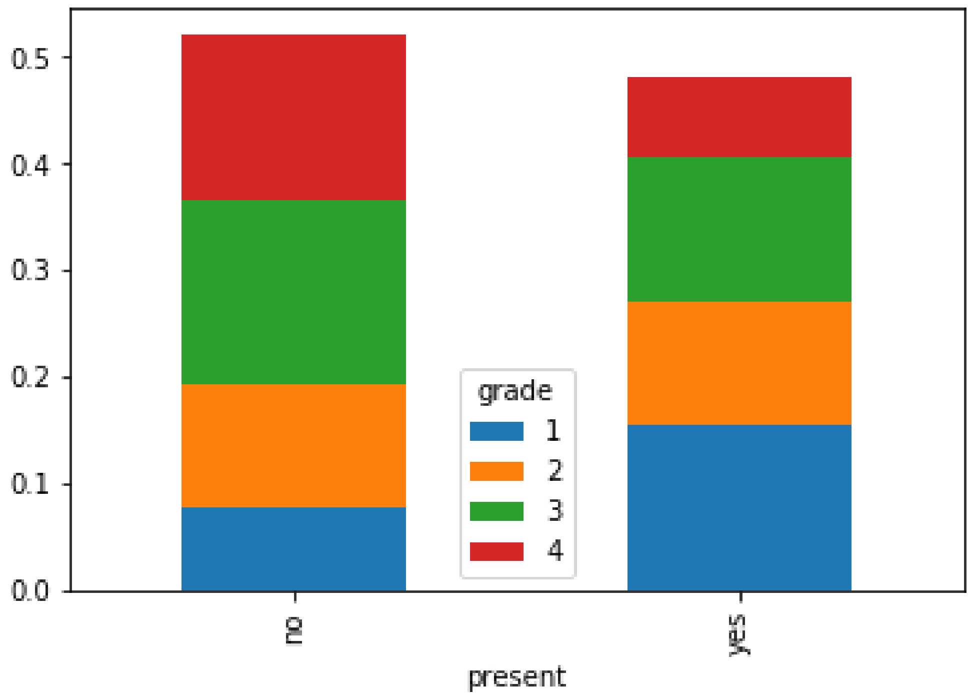
```
<Figure size 576x216 with 0 Axes>
```



[15]:
```
#(b)
event['rate2']=event['freq']/np.sum(event['freq'])

e_cross1=pd.
↪pivot_table(data=event,values="rate2",aggfunc="sum",index="present",columns="grade")
print(e_cross1)

e_cross1.plot(kind="bar",stacked=True)
```

```
grade           1         2         3         4
present
no       0.076923  0.115385  0.173077  0.153846
yes      0.153846  0.115385  0.134615  0.076923
```

[15]: <matplotlib.axes._subplots.AxesSubplot at 0x18dc811b400>



5번

[16]:
```
diabetes=pd.read_csv("C:/PythonbookData/table6.7_diabetes.csv")
diabetes.head()
```

[16]:

	patient	Y1	Y2	X1	X2	X3
0	1	0.81	80	356	124	55
1	2	0.95	97	289	117	76
2	3	0.94	105	319	143	105
3	4	1.04	90	356	199	108
4	5	1.00	90	323	240	143

[17]:
```
#(a)
sp.stats.pearsonr(diabetes['Y1'],diabetes['Y2'])[0]
# 상관성이 거의 없다고 할 수 있다.
```

[17]: 0.0831388509401317

[18]:
```
#(b)
sp.stats.pearsonr(diabetes['X1'],diabetes['Y2'])[0]
# 상관성이 거의 없다고 할 수 있다.
```

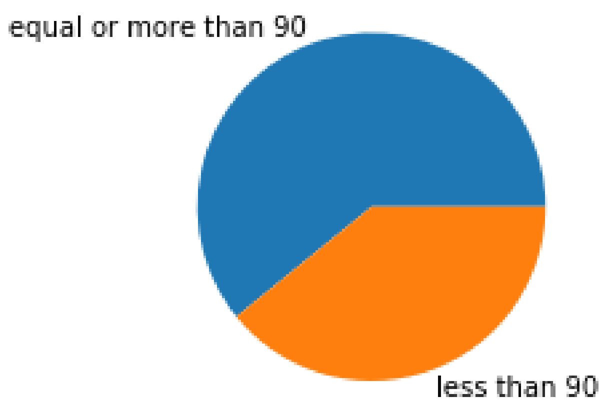
[18]: 0.01452643842796153

[20]:
```
#(c)
# H0: 두 변수 간 pearson 상관계수가 0이다.
# H1: 두 변수 간 pearson 상관계수가 0이 아니다..

sp.stats.pearsonr(diabetes['X1'],diabetes['Y2'])

# 상관성에 대한 유의성 검정 결과 pvalue=0.948>0.05이므로 유의수준 0.05에서 귀무가설을 기각하지 못한다.
# 따라서 유의수준 0.05에서 두 변수 간 pearson상관계수는 0이 아니라고 할 수 없다.
```

[20]:
```
(0.014526438427961 53, 0.9475493968222256)
```

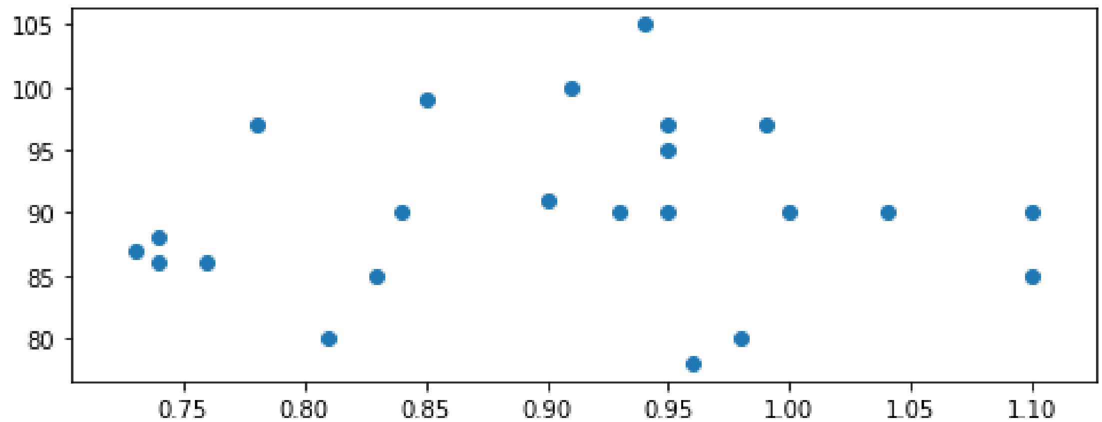
[32]:
```
#(d)
l1=diabetes.loc[(diabetes['Y2']>=90),:]
print("equal or more than 90:",len(l1))

l2=diabetes.loc[(diabetes['Y2']<90),:]
print("less than 90:",len(l2))

cat=('equal or more than 90','less than 90')
freq=[len(l1),len(l2)]

plt.figure(figsize=(8,3))
plt.pie(freq,labels=cat)
plt.show()
```

```
equal or more than 90: 14
less than 90: 9
```

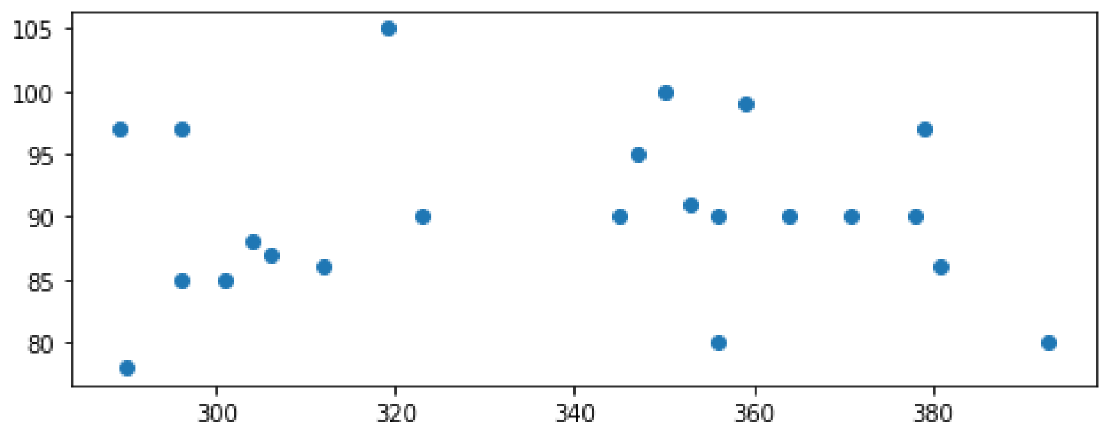


[33]:
```
#(e)
plt.figure(figsize=(8,3))
plt.scatter(x='Y1',y='Y2',data=diabetes)
```

```
plt.show()
```



[133]:
```
#(f)
plt.figure(figsize=(8,3))
plt.scatter(x='X1',y='Y2',data=diabetes)
plt.show()
```



[135]:
```
#(g)
print("Y1 & Y2 :",sp.stats.pearsonr(diabetes['Y1'],diabetes['Y2'])[0])
print("Y1 & X1 :",sp.stats.pearsonr(diabetes['Y1'],diabetes['X1'])[0])
print("Y1 & X2 :",sp.stats.pearsonr(diabetes['Y1'],diabetes['X2'])[0])
print("Y1 & X3 :",sp.stats.pearsonr(diabetes['Y1'],diabetes['X3'])[0])
print("Y2 & X1 :",sp.stats.pearsonr(diabetes['Y2'],diabetes['X1'])[0])
print("Y2 & X2 :",sp.stats.pearsonr(diabetes['Y2'],diabetes['X2'])[0])
print("Y2 & X3 :",sp.stats.pearsonr(diabetes['Y2'],diabetes['X3'])[0])
```

```python
print("X1 & X2 :",sp.stats.pearsonr(diabetes['X1'],diabetes['X2'])[0])
print("X1 & X3 :",sp.stats.pearsonr(diabetes['X1'],diabetes['X3'])[0])
print("X2 & X3 :",sp.stats.pearsonr(diabetes['X2'],diabetes['X3'])[0])

#상관계수가 가장 높은 쌍은 (X2, X3)이며 상관계수는 0.279이다.
```

```
Y1 & Y2 : 0.0831388509401317
Y1 & X1 : 0.19590133846507857
Y1 & X2 : 0.14485473447465502
Y1 & X3 : 0.24846676405154156
Y2 & X1 : 0.01452643842796153
Y2 & X2 : 0.039731030674018744
Y2 & X3 : -0.07244620805301141
X1 & X2 : 0.2627762919493086
X1 & X3 : 0.026349534455835287
X2 & X3 : 0.2791632202290843
```

6번

[136]:
```python
elder=[86,71,77,68,91,72,60]
younger=[88,77,76,64,96,72,65]

print("pearson: ",sp.stats.pearsonr(elder,younger)[0])
print("spearman: ",sp.stats.spearmanr(elder,younger)[0])
print("kendall: ",sp.stats.kendalltau(elder,younger)[0])

# 피어슨 상관계수, 스피어만 상관계수, 켄달의 타우를 구한 결과
#형과 동생사이의 공격성은 강한 양의 상관성을 보임을 알 수 있다.
```

```
pearson:  0.9489820285560242
spearman:  0.8571428571428573
kendall:  0.7142857142857143
```

7장 연습문제

[1]:
```
import numpy as np
import math as m
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
import scipy as sp
import statsmodels.formula.api as smf
import statsmodels.api as sm
from statsmodels.stats.proportion import proportions_ztest
```

1번

[3]:
```
data=[159,280,101,121,224,222,379,179,250,170]
```

[4]:
```
#(a)
# H0 : 전기기구의 평균 수리시간은 225시간이다.
# H1 : 전기기구의 평균 수리시간은 225시간이 아니다.
```

[5]:
```
#(b)
m=np.mean(data)
print("mean:",m)
print("median:",np.median(data))
print("var:",np.var(data,ddof=1))
print("std:",np.std(data,ddof=1))
```

```
mean: 208.5
median: 200.5
var: 6704.722222222223
std: 81.88236820111044
```

[6]:
```
#(c)
n=len(data)
sd=np.std(data,ddof=1)
cri=sp.stats.norm.ppf(loc=0,scale=1,q=0.975)

lower=m-cri*sd/np.sqrt(n)
upper=m+cri*sd/np.sqrt(n)
```

```
print(lower); print(upper)
```

```
157.74971495561874
259.25028504438126
```

[8]:
```
#(d)
sp.stats.ttest_1samp(data,225)

# 검정통계량=-0.637 이고pvalue=0.54>0.05 이므로 유의수준 0.05에서 귀무가설을 기각하지 못한다.
# 따라서 유의수준 0.05에서 전기기구의 평균 수리시간은 225시간이라고 할 수 있다.
```

[8]:
```
Ttest_1sampResult(statistic=-0.6372260907821502, pvalue=0.5398469302862716)
```

[10]:
```
#(e)
cri=sp.stats.norm.ppf(loc=0,scale=1,q=0.95)

lower=m-cri*sd/np.sqrt(n)
upper=m+cri*sd/np.sqrt(n)

print(lower); print(upper)
```

```
165.9090181847596
251.0909818152404
```

[13]:
```
#(f)
y=np.log1p(data)
y
```

[13]:
```
array([5.07517382, 5.63835467, 4.62497281, 4.80402104, 5.4161004 ,
       5.40717177, 5.94017125, 5.19295685, 5.52545294, 5.14166356])
```

[14]:
```
#(g)
print("mean:",np.mean(y))
print("var:",np.var(y,ddof=1))
```

```
mean: 5.276603911549472
var: 0.1537061967349907
```

3 번

[20]:
```
A=[501,502,495,498,499,506]
B=[508,510,503,504,500,504,505]
```

[21]:
```
#(a)
# H0: 두 기계의 분산은 같다.
# H1: 두 기계의 분산은 다르다.

F=np.var(A,ddof=1)/np.var(B,ddof=1)
```

```
df1=len(A)-1
df2=len(B)-1
pvalue=2*(1-sp.stats.f.cdf(F,df1,df2))
print('F=',F,'pvalue=',pvalue)
# 검정통계량 F=1.311 이며  pvalue=0.742>0.05 이므로 유의수준 0.05에서 귀무가설을 기각하지 못한다.
# 따라서 유의수준 0.05에서 두 기계의 분산은 같다고 할 수 있다.
```

```
F= 1.3105726872246697 pvalue= 0.7420881828204362
```

[25]:
```
#(b)
# H0: 두 기계의 평균은 같다.
# H1:  두 기계의 평균은 다르다.

sp.stats.ttest_ind(A,B,equal_var=True)


#  검정통계량 t=-2.4이고 pvalue=0.035<0.05   이므로 유의수준0.05에서 귀무가설을 기각한다.
# 따라서 유의수준 0.05에서 두 기계의 평균은 같다고 할 수 없다.
```

[25]:
```
Ttest_indResult(statistic=-2.400443918973941, pvalue=0.03520490080497176)
```

[21]:
```
#(c)
print("A mean:",np.mean(A))
print("A median:",np.median(A))
print("A var:",np.var(A,ddof=1))
print("A std:",np.std(A,ddof=1))

print("B mean:",np.mean(B))
print("B median:",np.median(B))
print("B var:",np.var(B,ddof=1))
print("B std:",np.std(B,ddof=1))
```

```
A mean: 500.1666666666667
A median: 500.0
A var: 14.166666666666668
A std: 3.7638632635454052
B mean: 504.85714285714283
B median: 504.0
B var: 10.80952380952381
B std: 3.2877840272018797
```

4 번

[22]:
```
diet=pd.read_csv("C:/PythonbookData/table7.7_diet.csv")
diet.head()
```

[22]:

	id	before	after
0	1	55	54
1	2	60	55
2	3	70	64
3	4	75	73
4	5	66	61

[25]:

```
#(a)
diet['difference']=diet['before']-diet['after']
print("difference:",diet['difference'])

print("before mean:",np.mean(diet['before']))
print("before var:",np.var(diet['before'],ddof=1))
print("before std:",np.std(diet['before'],ddof=1))

print("after mean:",np.mean(diet['after']))
print("after var:",np.var(diet['after'],ddof=1))
print("after std:",np.std(diet['after'],ddof=1))

print("difference mean:",np.mean(diet['difference']))
print("difference var:",np.var(diet['difference'],ddof=1))
print("difference std:",np.std(diet['difference'],ddof=1))
```

```
difference: 0     1
1     5
2     6
3     2
4     5
5     8
6     4
7    18
8    10
9     1
Name: difference, dtype: int64
before mean: 72.8
before var: 105.95555555555556
before std: 10.293471501663351
after mean: 66.8
after var: 70.4
after std: 8.390470785361213
difference mean: 6.0
difference var: 26.222222222222222
difference std: 5.120763831912406
```

[27]:

```
#(b)
# H0 : 약의 효과가 없다.
# H1 : 약의 효과가 있다.
```

```python
sp.stats.ttest_1samp(diet['difference'],0)

# 검정통계량 t=3.705, pvalue=0.005/2=0.0025<0.05 이므로 유의수준 0.05 에서 귀무가설을 기각한다.
# 따라서 유의수준 0.05 에서 약의 효과가 있다고 할 수 있다.
```

[27]: Ttest_1sampResult(statistic=3.705241363166782, pvalue=0.004880794471499406)

5 번

[28]:
```python
words=pd.read_csv("C:/PythonbookData/table7.8_words.csv")
words.head()
```

[28]:

	student_num	X1	X2
0	1	51	36
1	2	27	20
2	3	37	22
3	4	42	36
4	5	27	18

[29]:
```python
#(a)
print("x1 mean:",np.mean(words['X1']))
print("x1 var:",np.var(words['X1'],ddof=1))
print("x1 std:",np.std(words['X1'],ddof=1))

print("x2 mean:",np.mean(words['X2']))
print("x2 var:",np.var(words['X2'],ddof=1))
print("x2 std:",np.std(words['X2'],ddof=1))
```

```
x1 mean: 36.0
x1 var: 59.27272727272727
x1 std: 7.6988783126327744
x2 mean: 25.916666666666668
x2 var: 43.53787878787879
x2 std: 6.598323937779865
```

[32]:
```python
#(b)
sp.stats.ttest_1samp(words['X1'],30)

# 검정통계량 t=2.7, pvalue=0.021<0.05 이므로 유의수준 0.05 에서 귀무가설을 기각한다.
# 따라서 유의수준 0.05 에서 mu(x1)=30 이라고 할 수 없다.
```

[32]: Ttest_1sampResult(statistic=2.6996932341068325, pvalue=0.02066807131079545)

[34]:
```python
#(c)
sp.stats.ttest_1samp(words['X2'],25)
# 검정통계량 t=0.48, pvalue=0.64>0.05 이므로 유의수준 0.05 에서 귀무가설을 기각하지 못한다.
```

```
# 따라서 유의수준 0.05에서 mu(x2)=25라고 할 수 있다.
```

[34]: Ttest_1sampResult(statistic=0.4812474365439203, pvalue=0.6397711672994958)

6번

[36]:
```
count=120; nobs=300; value=0.5
prop=count/nobs
```

[46]:
```
#(a)
#H0: B정당의 지지도는 50%이다.
#H1: B정당의 지지도는 50%가 아니다.

proportions_ztest(count,nobs,value)

# 검정통계량 z=-3.54, pvalue=0.0004<0.05이므로 유의수준 0.05에서 귀무가설을 기각한다.
# 따라서 유의수준 0.05에서 B정당의 지지도는 50%라고 할 수 없다.
```

[46]: (-3.535533905932737, 0.00040695201744495973)

[47]:
```
#(b)
#H0: B정당의 지지도는 50%이다.
#H1: B정당의 지지도는 50%보다 낮다.

proportions_ztest(count,nobs,value)

# 검정통계량 z=-3.54, pvalue=0.0004/2=0.0002<0.05 이므로 유의수준 0.05에서 귀무가설을 기각한다.
# 따라서 유의수준 0.05에서 B정당의 지지도는 50%보다 낮다고 할 수 있다 .
```

[47]: (-3.535533905932737, 0.00040695201744495973)

7번

[8]:
```
count=np.array([500,200])
nobs=np.array([3500,2800])
prop=count/nobs

#H0: 두 생명보험의 계약 해지 비율은 같다
#H1:두 생명보험의 계약 해지 비율은 다르다

proportions_ztest(count,nobs)

# 검정통계량 z=8.964, pvalue=3.125e-19<0.05이므로 유의수준 0.05에서 귀무가설을 기각한다.
# 따라서 유의수준 0.05에서 두 생명보험의 계약 해지 비율은 다르다고 할 수 있다.
```

[8]: (8.964214570007952, 3.1250001860770074e-19)

8번

[9]:
```
count=50; nobs=200; value=0.2
prop=count/nobs

#H0: k 음료의 선호도는 20%이다,
#H1: k 음료의 선호도는 20%가 아니다.

proportions_ztest(count,nobs,value)

# 검정통계량 z=1.633, pvalue=0.102>0.05이므로 유의수준 0.05에서 귀무가설을 기각하지 못한다.
# 따라서 유의수준 0.05에서 k음료의 선호도는 20%라고 할 수 있다.
```

[9]:
```
(1.6329931618554518, 0.10247043485974941)
```

9번

[13]:
```
count=np.array([2000,2000])
nobs=np.array([5500,3000])
prop=count/nobs

#H0: 두 도시의 지하철 이용비율은 같다.
#H1: B 도시의 지하철 이용비율이 A도시의 지하철 이용비율보다 높다.

proportions_ztest(count,nobs)
# 검정통계량 z=-26.749, pvalue=(1.28e-157)/2<0.05이므로 유의수준 0.05에서 귀무가설을 기각한다.
# 따라서 유의수준 0.05에서 B 도시의 지하철 이용비율이 A도시의 지하철 이용비율보다
# 높다고 할 수 있다.
```

[13]:
```
(-26.748611468414865, 1.2815690431381088e-157)
```

10번

[14]:
```
g1=np.array([14,15,16,13,12,17,15,13,16,13])
g2=np.array([8,11,9,8,10,11,7,9,6,8,7,10])
```

[16]:
```
#(a)
print("mean:",np.mean(g1))
print("std:",np.std(g1,ddof=1))
```

```
mean: 14.4
std: 1.6465452046971292
```

[17]:
```
#(b)
print("mean:",np.mean(g2))
print("std:",np.std(g2,ddof=1))
```

```
mean: 8.666666666666666
std: 1.6143297699232972
```

[26]:
```python
#(c)
# 분산동일성검정
# H0: 두 집단의 분산은 같다.
# H1: 두 집단의 분산은 다르다.

F=np.var(g1,ddof=1)/np.var(g2,ddof=1)
df1=len(g1)-1
df2=len(g2)-1
pvalue=2*(1-sp.stats.f.cdf(F,df1,df2))
print('F=',F,'pvalue=',pvalue)

# 검정통계량 F=1.04이며 pvalue=0.935>0.05이므로 유의수준 0.05에서 귀무가설을 기각하지 못한다.
# 따라서 유의수준 0.05에서 두 집단의 분산은 같다고 할 수 있다.


# t검정
# H0: 두 집단의 헤모글로빈 평균은 같다.
# H1: 두 집단의 헤모글로빈 평균은 다르다.

sp.stats.ttest_ind(g1,g2,equal_var=True)

# 검정통계량 t=8.22이며 pvalue=7.641e-08<0.05이므로 유의수준 0.05에서 귀무가설을 기각한다.
# 따라서 유의수준 0.05에서 두 집단의 헤모글로빈 평균은 같다고 할 수 없다.
```

```
F= 1.0403100775193799 pvalue= 0.9345384814678361
```

[26]:
```
Ttest_indResult(statistic=8.220354907813636, pvalue=7.641133820875184e-08)
```

11번

[46]:
```python
g1=np.array([67,79,57,66,71,78])
g2=np.array([42,61,64,76,45,58])

 # H0: 두 그룹의 심장박동비율에 차이가 없다.
 # H1: 두 그룹의 심장박동비율에 차이가 있다.

sp.stats.ttest_ind(g1,g2)

# 검정통계량 t=1.95이며 pvalue=0.08>0.05이므로 유의수준 0.05에서 귀무가설을 기각하지 못한다.
# 따라서 유의수준 0.05에서 두 그룹의 심장박동비율에 차이가 있다고 할 수 없다.
```

[46]:
```
Ttest_indResult(statistic=1.9529486015103585, pvalue=0.07935931558149617)
```

12번

[48]:
```python
g1=np.array([2.1,5.0,1.4,4.6,3.0,4.3,3.2])
g2=np.array([1.9,0.5,2.8,3.1,2.7,1.8])

# 분산동일성검정
# H0: 두 집단의 분산은 같다.
# H1: 두 집단의 분산은 다르다.

F=np.var(g1,ddof=1)/np.var(g2,ddof=1)
df1=len(g1)-1
df2=len(g2)-1
pvalue=2*(1-sp.stats.f.cdf(F,df1,df2))
print('F=',F,'pvalue=',pvalue)

# 검정통계량 F=1.97이며 pvalue=0.475>0.05이므로 유의수준 0.05에서 귀무가설을 기각하지 못한다.
# 따라서 유의수준 0.05에서 두 집단의 분산은 같다고 할 수 있다.


# t검정
# H0: 혈청 주입 여부에 따라 생존연수 간에 차이가 없다.
# H1: 혈청 주입 여부에 따라 생존연수 간에 차이가 있다.

sp.stats.ttest_ind(g1,g2,equal_var=True)

# 검정통계량 t=1.89이며 pvalue=0.085>0.05이므로 유의수준 0.05에서 귀무가설을 기각하지 못한다.
# 따라서 유의수준 0.05에서 혈청 주입 여부에 따라 생존연수 간에 차이가 있다고 할 수 없다.
```

```
F= 1.9658613445378152 pvalue= 0.4752162590088296
```

[48]:
```
Ttest_indResult(statistic=1.8914173266121344, pvalue=0.08517807433726889)
```

13번

[50]:
```python
A=np.array([90,88,78,65,78,60,89,73])
B=np.array([80,78,75,69,73,62,79,70])
D=A-B

# H0 : 두 음료의 맛에 차이가 없다.
# H1 : 두 음료의 맛에 차이가 있다.

sp.stats.ttest_1samp(D,0)

# 검정통계량 t=2.26, pvalue=0.058>0.05이므로 유의수준 0.05에서 귀무가설을 기각하지 못한다.
# 따라서 유의수준 0.05에서 두 음료의 맛에 차이가 있다고 할 수 없다.
```

[50]:
```
Ttest_1sampResult(statistic=2.2599129785541043, pvalue=0.05833886818381974)
```

14번

[53]:
```
E=np.array([80,88,78,65,78,60,89,73])
F=np.array([70,78,75,69,73,62,79,70])
D=E-F

# H0 : 두 로션 간의 효과 차이가 없다.
# H1 : 두 로션 간의 효과 차이가 있다.

sp.stats.ttest_1samp(D,0)

# 검정통계량 t=2.26, pvalue=0.058>0.05이므로 유의수준 0.05에서 귀무가설을 기각하지 못한다.
# 따라서 유의수준 0.05에서 두 로션 간의 효과 차이가 있다고 할 수 없다.
```

[53]:
```
Ttest_1sampResult(statistic=2.2599129785541043, pvalue=0.05833886818381974)
```

8 장 연습문제

[1]:
```
import numpy as np
import math as m
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
import scipy as sp
import statsmodels.formula.api as smf
import statsmodels.api as sm
```

1번

[2]:
```
apply=pd.read_csv("C:/PythonbookData/problem8.1_apply.csv")
apply
```

[2]:

	gender	apply	freq
0	male	yes	30
1	female	yes	10
2	male	no	30
3	female	no	10

[3]:
```
apply_tab=pd.
↪pivot_table(data=apply,values="freq",aggfunc="sum",index="apply",columns="gender")
print(apply_tab)

sp.stats.chi2_contingency(apply_tab)[3]
```

```
gender  female  male
apply
no          10    30
yes         10    30
```

[3]:
```
array([[10., 30.],
       [10., 30.]])
```

[6]:
```
#(b)
 # H0: 입학여부와 성별 간에 관계가 없다.
 # H1: 입학여부와 성별 간에 관계가 있다.
```

```
sp.stats.chi2_contingency(apply_tab)

# 검정통계량은 0이며 pvalue=1.0>0.05이므로 유의수준 0.05에서 귀무가설을 기각하지 못한다.
# 따라서 유의수준 0.05에서 입학여부와 성별 간에 관계가 있다고 할 수 없다.
```

[6]:
```
(0.0, 1.0, 1, array([[10., 30.],
        [10., 30.]]))
```

2번

[7]:
```
apply2=pd.read_csv("C:/PythonbookData/problem8.2_apply.csv")
apply2
```

[7]:

	gender	apply	freq
0	male	yes	30
1	female	yes	20
2	male	no	30
3	female	no	10

[8]:
```
apply_tab2=pd.
 ↪pivot_table(data=apply2,values="freq",aggfunc="sum",index="apply",columns="gender")
print(apply_tab2)

sp.stats.chi2_contingency(apply_tab2)[3]
```

```
gender  female  male
apply
no          10    30
yes         20    30
```

[8]:
```
array([[13.33333333, 26.66666667],
       [16.66666667, 33.33333333]])
```

[9]:
```
#(b)
# H0: 입학여부와 성별 간에 관계가 없다.
# H1: 입학여부와 성별 간에 관계가 있다.

sp.stats.chi2_contingency(apply_tab2)

# 검정통계량은 1.626이며 pvalue=0.202>0.05이므로 유의수준 0.05에서 귀무가설을 기각하지 못한다.
# 따라서 유의수준 0.05에서 입학여부와 성별 간에 관계가 있다고 할 수 없다.
```

[9]:
```
(1.6256249999999999, 0.20230924199116818, 1, array([[13.33333333, 26.66666667],
        [16.66666667, 33.33333333]]))
```

3번

[24]:
```
event=pd.read_csv("C:/PythonbookData/problem8.3_hangsa.csv")
print(event)

event_tab=pd.
↪pivot_table(data=event,values="freq",aggfunc="sum",index="present",columns="grade")
print(event_tab)

# H0: 학년과 학과 행사 참석 여부는 독립이다.
# H1: 학년과 학과 행사 참석 여부는 독립이 아니다.

sp.stats.chi2_contingency(event_tab)

# 검정통계량은 14.22이며 pvalue=0.0026<0.05이므로 유의수준 0.05에서 귀무가설을 기각한다.
# 따라서 유의수준 0.05에서 학년과 학과 행사 참석 여부는 독립이라고 할 수 없다.
```

```
   grade present  freq
0      1     yes    40
1      2     yes    30
2      3     yes    35
3      4     yes    20
4      1      no    20
5      2      no    30
6      3      no    45
7      4      no    40
grade     1   2   3   4
present
no       20  30  45  40
yes      40  30  35  20
```

[24]:
```
(14.219753086419752,
 0.0026207903035146093,
 3,
 array([[31.15384615, 31.15384615, 41.53846154, 31.15384615],
        [28.84615385, 28.84615385, 38.46153846, 28.84615385]]))
```

4번

[26]:
```
dice=np.array([30,20,40,10,40,60])
p0=np.array([1/6,1/6,1/6,1/6,1/6,1/6])
n=200
expected=n*p0

# H0: p1=1/6, p2=1/6, p3=1/6, p4=1/6, p5=1/6, p6=1/6(각 수가 나올 확률이 각각 1/6이다.)
# H1: not H0
```

```
sp.stats.chisquare(f_obs=dice,f_exp=expected)

# 검정통계량은 46이며 pvalue=9.082e-09<0.05이므로 유의수준 0.05에서 귀무가설을 기각한다.
# 따라서 유의수준 0.05에서 각 수가 나올 확률이 각각 1/6이라고 할 수 없다.
```

[26]:

```
Power_divergenceResult(statistic=46.00000000000001,
pvalue=9.082105818889259e-09)
```

5번

[31]:

```
observed=np.array([53,42,51,45,36,37,65])
p0=np.array([1/7,1/7,1/7,1/7,1/7,1/7,1/7])
n=np.sum(observed)
expected=n*p0

# H0: p1=1/7, p2=1/7, p3=1/7, p4=1/7, p5=1/7, p6=1/7, p7=1/7
# H1: not H0

sp.stats.chisquare(f_obs=observed,f_exp=expected)

# 검정통계량은 13.319이며 pvalue=0.0382<0.05이므로 유의수준 0.05에서 귀무가설을 기각한다.
# 따라서 유의수준 0.05에서 요일별 살인사건 발생 확률은 같다고 할 수 없다.
```

[31]:

```
Power_divergenceResult(statistic=13.319148936170212,
pvalue=0.038238681612064264)
```

6번

[33]:

```
seatbelt=pd.read_csv("C:/PythonbookData/problem8.6_seatbelt.csv")
seatbelt
```

[33]:

	parent_seatbelt	child_seatbelt	freq
0	yes	yes	6
1	yes	no	10
2	no	yes	5
3	no	no	20

[36]:

```
#(a)
seatbelt_tab=pd.
↪pivot_table(data=seatbelt,values="freq",aggfunc="sum",index="parent_seatbelt",
 columns="child_ seatbelt ")
print(seatbelt_tab)
# H0: 부모의 안전벨트 착용여부와 자녀의 안전벨트 착용여부는 독립이다.
# H1: 부모의 안전벨트 착용여부와 자녀의 안전벨트 착용여부는 독립이 아니다.

sp.stats.chi2_contingency(seatbelt_tab)
```

```
# 검정통계량은 0.761이며 pvalue=0.383>0.05이므로 유의수준 0.05에서 귀무가설을 기각하지 못한다.
# 따라서 유의수준 0.05에서 부모의 안전벨트 착용여부와 자녀의 안전벨트 착용여부는
# 독립이라고 할 수 있다.
```

```
child_seatbelt   no  yes
parent_seatbelt
no               20    5
yes              10    6
```

[36]: (0.7610624999999995, 0.38299622041867754, 1, array([[18.29268293, 6.70731707],
        [11.70731707, 4.29268293]]))

[39]:
```
#(b)
sp.stats.chi2_contingency(seatbelt_tab)[3]
```

[39]: array([[18.29268293, 6.70731707],
       [11.70731707, 4.29268293]])

7번

[48]:
```
sugang=pd.read_csv("C:/PythonbookData/problem8.7_sugang.csv")
print(sugang)

tab=pd.crosstab(index=sugang["gender"],columns=sugang["result"])
print(tab)

 # H0: 수강결과 성적과 성별 간에 관계가 없다.
# H1: 수강결과 성적과 성별 간에 관계가 있다.        .

sp.stats.chi2_contingency(tab)

# 검정통계량은 0.292이며 pvalue=0.589>0.05이므로 유의수준 0.05에서 귀무가설을 기각하지 못한다.
# 따라서 유의수준 0.05에서 수강결과 성적과 성별 간에 관계가 있다고 할 수 없다.
```

```
    id gender result
0    1      M      P
1    2      M      P
2    3      M      F
3    4      M      F
4    5      M      F
5    6      M      F
6    7      F      P
7    8      F      P
8    9      F      P
9   10      F      F
10  11      F      P
11  12      M      P
12  13      F      P
```

```
13  14       F       F
result  F  P
gender
F       2  5
M       4  3
```

[48]:
```
(0.29166666666666663, 0.5891544654500582, 1, array([[3., 4.],
        [3., 4.]]))
```

8번

[47]:
```python
observed=np.array([20,55,30])
p0=np.array([1/3,1/3,1/3])
n=np.sum(observed)
expected=n*p0

# H0: p1=1/3, p2=1/3, p3=1/3 (유전자형 분포는 동일하다.)
 # H1: not H0

sp.stats.chisquare(f_obs=observed,f_exp=expected)

# 검정통계량은 18.57이며 pvalue=9.274e-05<0.05이므로 유의수준 0.05에서 귀무가설을 기각한다.
# 따라서 유의수준 0.05에서 유전자형 분포는 동일하다고 할 수 없다.
```

[47]:
```
Power_divergenceResult(statistic=18.571428571428573,
pvalue=9.273966551400925e-05)
```

9번

[49]:
```python
hand=pd.read_csv("C:/PythonbookData/problem8.9_hand.csv")
print(hand)
```

```
    hand  gender  freq
0  right    male    27
1  right  female    18
2   left    male     7
3   left  female    10
```

[50]:
```python
#(a)
hand['rate']=hand['freq']/np.sum(hand['freq'])
hand_p=pd.
↪pivot_table(data=hand,values="rate",aggfunc="sum",index="hand",columns="gender",margins=True)
hand_p
```

[50]:
```
gender    female      male       All
hand
left    0.161290  0.112903  0.274194
```

```
right    0.290323  0.435484  0.725806
All      0.451613  0.548387  1.000000
```

[51]:
```
#(b)
hand_tab=pd.
 ↪pivot_table(data=hand,values="freq",aggfunc="sum",index="hand",columns="gender")
print(hand_tab)

sp.stats.chi2_contingency(hand_tab)[3]
```

```
gender  female  male
hand
left        10     7
right       18    27
```

[51]:
```
array([[ 7.67741935,  9.32258065],
       [20.32258065, 24.67741935]])
```

[53]:
```
#(c)
# H0: 성별과 양손을 사용하는 빈도 간에 관계가 없다.
  # H1: 성별과 양손을 사용하는 빈도 간에 관계가 있다.

sp.stats.chi2_contingency(hand_tab)

# 검정통계량은 1.087이며 pvalue=0.297>0.05이므로 유의수준 0.05에서 귀무가설을 기각하지 못한다.
# 따라서 유의수준 0.05에서 성별과 양손을 사용하는 빈도 간에 관계가 있다고 할 수 없다.
```

[53]:
```
(1.0870516834184656, 0.29712538491315754, 1, array([[ 7.67741935,  9.32258065],
        [20.32258065, 24.67741935]]))
```

10번

[61]:
```
president=pd.read_csv("C:/PythonbookData/problem8.10_president.csv")
print(president)

president_tab=pd.
 ↪pivot_table(data=president,values="freq",aggfunc="sum",index="tendency",
   columns="candidate")
print(president_tab)
# H0: 대통령 후보와 이념 성향은 독립이다.
# H1: 대통령 후보와 이념 성향은 독립이 아니다.

sp.stats.chi2_contingency(president_tab)

# 검정통계량은 151.386이며 pvalue=1.34e-33<0.05이므로 유의수준 0.05에서 귀무가설을 기각한다.
# 따라서 유의수준 0.05에서 대통령 후보와 이념 성향은 독립이라고 할 수 없다.
```

```
  candidate tendency  freq
0         A    jinbo   115
1         A   jungdo   169
2         A     bosu   225
3         B    jinbo   395
4         B   jungdo   221
5         B     bosu   125
candidate     A    B
tendency
bosu        225  125
jinbo       115  395
jungdo      169  221
```

[61]:
```
(151.38589657922776, 1.3395848857156639e-33, 2, array([[142.52 , 207.48 ],
        [207.672, 302.328],
        [158.808, 231.192]]))
```

11번

[64]:
```python
product=pd.read_csv("C:/PythonbookData/problem8.11_newproduct.csv")
print(product)

product_tab=pd.
 ↪pivot_table(data=product,values="freq",aggfunc="sum",index="buy",columns="age")
print(product_tab)

# H0: 여성나이대와 구매 여부는 독립이다.
# H1: 여성나이대와 구매 여부는 독립이 아니다.

sp.stats.chi2_contingency(product_tab)

# 검정통계량은 48.096이며 pvalue=9.014e-10<0.05이므로 유의수준 0.05에서 귀무가설을 기각한다.
# 따라서 유의수준 0.05에서 여성나이대와 구매여부는 독립이라고 할 수 없다.
```

```
     buy  age  freq
0     no   20    24
1     no   30    20
2     no   40    50
3  think   20    32
4  think   30    30
5  think   40    20
6    yes   20    54
7    yes   30    60
8    yes   40    15
age     20  30  40
buy
no      24  20  50
think   32  30  20
```

```
     yes    54  60  15
```

[64]:
```
(48.09571270280807,
 9.014035184801174e-10,
 4,
 array([[33.90163934, 33.90163934, 26.19672131],
        [29.57377049, 29.57377049, 22.85245902],
        [46.52459016, 46.52459016, 35.95081967]]))
```

9장 연습문제

[2]:
```python
import numpy as np
import math as m
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
import scipy as sp
import statsmodels.formula.api as smf
import statsmodels.api as sm
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
import pylab
```

1번

[2]:
```python
x=np.array([10,5,7,19,11,8])
y=np.array([15,9,3,25,7,13])
```

[5]:
```python
#(a)
df=pd.DataFrame()
df['x']=x
df['y']=y

lm=smf.ols("y~x",df).fit()
lm.params

#y=-0.67+1.27x
```

[5]:
```
Intercept   -0.666667
x            1.266667
dtype: float64
```

[19]:
```python
#(b)
y_pred=lm.predict(df['x'])
y_pred
```

[19]:
```
0    12.000000
1     5.666667
```

```
2     8.200000
3    23.400000
4    13.266667
5     9.466667
dtype: float64
```

[20]:
```
#(c)
resid=y-y_pred
resid
```

[20]:
```
0    3.000000
1    3.333333
2   -5.200000
3    1.600000
4   -6.266667
5    3.533333
dtype: float64
```

[26]:
```
#(d)
sse=np.sum(resid**2)
sse
```

[26]:
```
101.46666666666663
```

[34]:
```
#(e)
sst=np.sum((y-np.mean(y))**2)
ssr=sst-sse
ssr
```

[34]:
```
192.53333333333336
```

[36]:
```
#(f)
lm.summary()

#R-squared=0.655
#추정된 회귀직선의 설명력이 어느정도 있다고 할 수 있다.
```

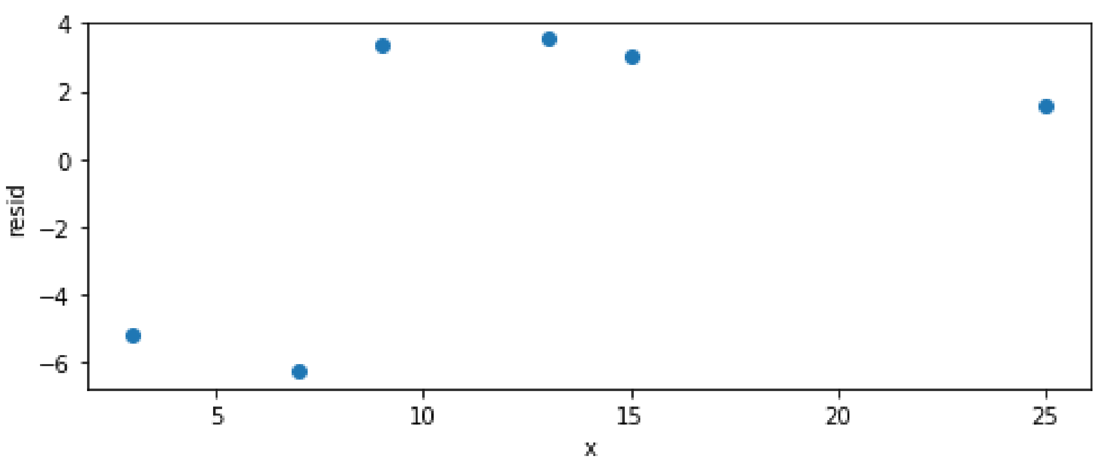
```
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\stats\stattools.py:72:
ValueWarning: omni_normtest is not valid with less than 8 observations; 6
samples were given.
  "samples were given." % int(n), ValueWarning)
```

[36]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.655
Model:                            OLS   Adj. R-squared:                  0.569
```

```
Method:                 Least Squares   F-statistic:                     7.590
Date:                Fri, 25 Jun 2021   Prob (F-statistic):             0.0511
Time:                        17:46:45   Log-Likelihood:                -16.998
No. Observations:                   6   AIC:                             38.00
Df Residuals:                       4   BIC:                             37.58
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.6667      5.037     -0.132      0.901     -14.650      13.317
x              1.2667      0.460      2.755      0.051      -0.010       2.543
==============================================================================
Omnibus:                          nan   Durbin-Watson:                   2.731
Prob(Omnibus):                    nan   Jarque-Bera (JB):                0.967
Skew:                          -0.658   Prob(JB):                        0.617
Kurtosis:                       1.539   Cond. No.                         27.0
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

[40]:

```python
#(g)
plt.figure(figsize=(8,3))
plt.scatter(y,resid)
plt.xlabel('x');plt.ylabel('resid')
plt.show()
```



2번

[3]:
```python
cotton=pd.read_csv("C:/PythonbookData/table9.5_cotton.csv")
print(cotton.head())

lm_cotton=smf.ols("Yield~Irrigation",data=cotton).fit()
lm_cotton.summary()

#Yield=-24.49+167.86Irrigation
#R-Squared=0.717                    .
```

```
   Irrigation  Yield
0         1.8    260
1         1.9    370
2         2.5    450
3         1.4    160
4         1.3     90
```

```
C:\ProgramData\Anaconda3\lib\site-packages\scipy\stats\stats.py:1394:
UserWarning: kurtosistest only valid for n>=20 … continuing anyway, n=14
  "anyway, n=%i" % int(n))
```

[3]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Yield   R-squared:                       0.717
Model:                            OLS   Adj. R-squared:                  0.693
Method:                 Least Squares   F-statistic:                     30.41
Date:                Wed, 30 Jun 2021   Prob (F-statistic):           0.000133
Time:                        17:27:00   Log-Likelihood:                -80.625
No. Observations:                  14   AIC:                             165.3
Df Residuals:                      12   BIC:                             166.5
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    -24.4871     63.967     -0.383      0.709    -163.859     114.885
Irrigation   167.8558     30.441      5.514      0.000     101.531     234.180
==============================================================================
Omnibus:                        1.281   Durbin-Watson:                   1.999
Prob(Omnibus):                  0.527   Jarque-Bera (JB):                0.974
Skew:                          -0.583   Prob(JB):                        0.614
Kurtosis:                       2.442   Cond. No.                         7.31
==============================================================================

Warnings:
```

```
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

3번

[4]:
```
x1=[10,5,7,19,11,18]
x2=[2,3,3,6,7,9]
y=[15,9,3,25,7,13]

df=pd.DataFrame({'y':y,'x1':x1,'x2':x2})
df
```

[4]:

	y	x1	x2
0	15	10	2
1	9	5	3
2	3	7	3
3	25	19	6
4	7	11	7
5	13	18	9

[5]:
```
#(a)
lm=smf.ols("y~0+x1+x2",data=df).fit()
lm.params

#y=1.9x1-2.09*x2
```

[5]:
```
x1    1.895041
x2   -2.090807
dtype: float64
```

[6]:
```
#(b)
y_pred=lm.predict(pd.DataFrame({'x1':x1,'x2':x2}))
y_pred
```

[6]:
```
0    14.768795
1     3.202783
2     6.992865
3    23.460935
4     6.209800
5    15.293472
dtype: float64
```

[7]:
```
#(c)
resid=y-y_pred
resid
```

[7]:
```
0    0.231205
1    5.797217
2   -3.992865
3    1.539065
4    0.790200
5   -2.293472
dtype: float64
```

[8]:
```
#(d)
sse=np.sum(resid**2)
print('sse:',sse)
sst=np.sum((y-np.mean(y))**2)
ssr=sst-sse
print('ssr',ssr)
```

```
sse: 57.85730290701103
ssr 236.14269709298895
```

[9]:
```
#(e)
lm.summary()

#R-squared=0.95
# 추정된 회귀직선의 설명력이 높다고 할 수 있다.
```

```
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\stats\stattools.py:72:
ValueWarning: omni_normtest is not valid with less than 8 observations; 6
samples were given.
  "samples were given." % int(n), ValueWarning)
```

[9]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.950
Model:                            OLS   Adj. R-squared:                  0.925
Method:                 Least Squares   F-statistic:                     38.03
Date:                Wed, 30 Jun 2021   Prob (F-statistic):            0.00250
Time:                        17:28:37   Log-Likelihood:                -15.312
No. Observations:                   6   AIC:                             34.62
Df Residuals:                       4   BIC:                             34.21
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1             1.8950      0.400      4.733      0.009       0.783       3.007
x2            -2.0908      0.914     -2.287      0.084      -4.629       0.448
==============================================================================
```

```
Omnibus:                          nan   Durbin-Watson:                   2.895
Prob(Omnibus):                    nan   Jarque-Bera (JB):                0.231
Skew:                           0.362   Prob(JB):                        0.891
Kurtosis:                       2.367   Cond. No.                         8.86
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

4번

[3]:
```python
x=[4,6,6,8,8,8,9,9,10,12]
y=[9,10,18,20,15,17,20,22,25,30]
df=pd.DataFrame({'y':y,'x':x})
df
```

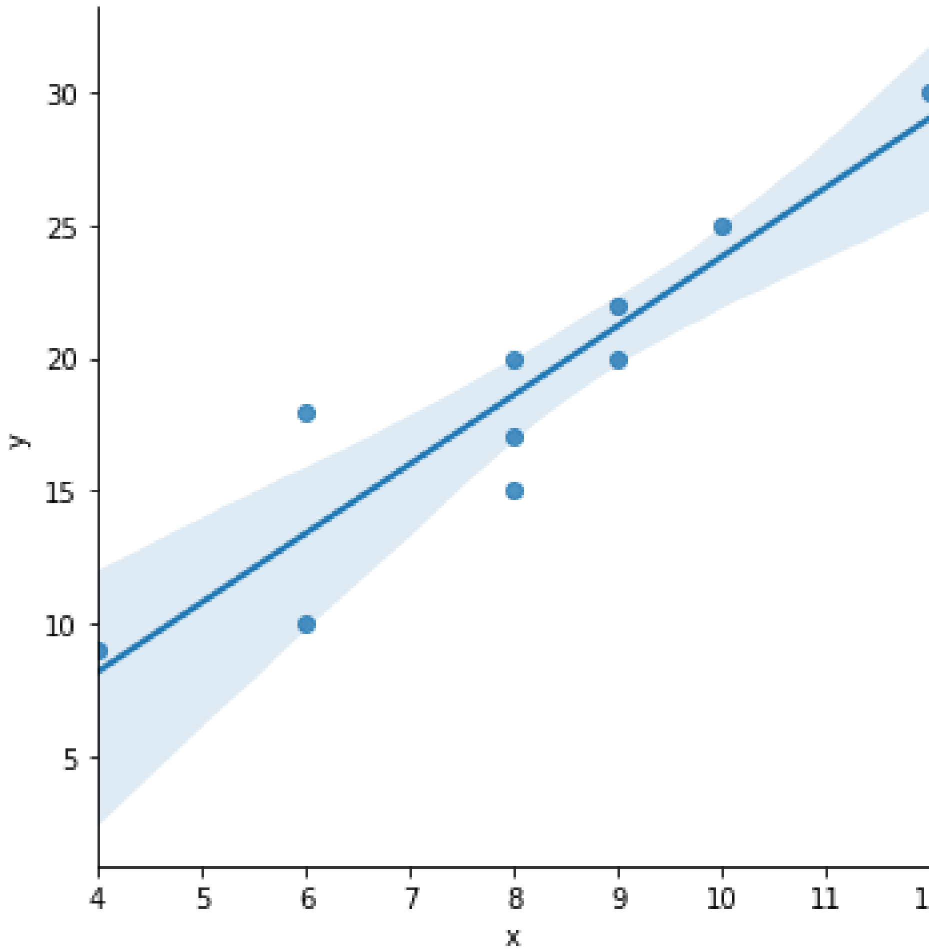
[3]:

	y	x
0	9	4
1	10	6
2	18	6
3	20	8
4	15	8
5	17	8
6	20	9
7	22	9
8	25	10
9	30	12

[4]:
```python
#(a)
lm=smf.ols("y~x",df).fit()
print(lm.params)

plt.figure(figsize=(8,3))
sns.lmplot(x='x',y='y',data=df)
plt.show()
```

```
Intercept   -2.269565
x            2.608696
dtype: float64
```

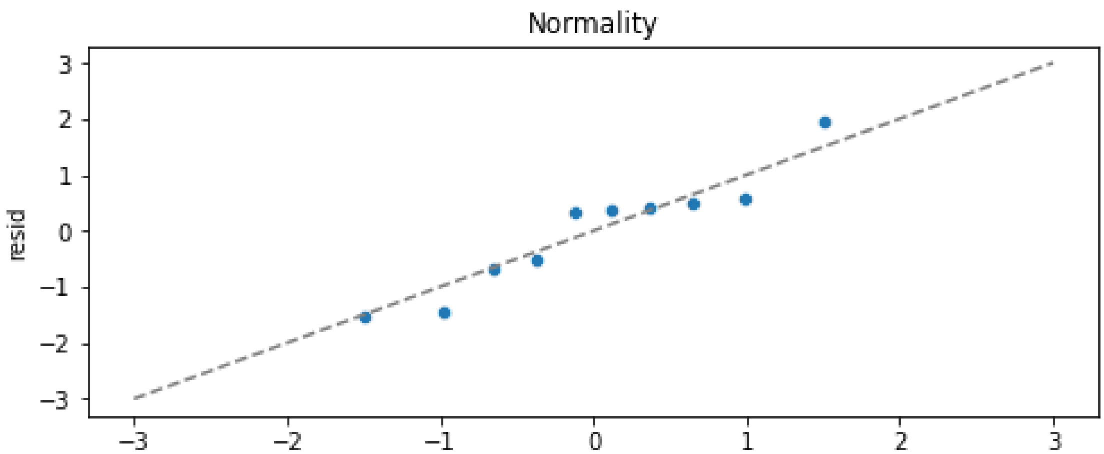
```
<Figure size 576x216 with 0 Axes>
```



[5]:
```
#(b)
y_pred=lm.predict(df['x'])
resid=y-y_pred

plt.figure(figsize=(8,3))
sr=sp.stats.zscore(resid)
(a,b),_=sp.stats.probplot(sr)
sns.scatterplot(a,b)
plt.plot([-3,3],[-3,3],'--',color='grey')
plt.title("Normality"); plt.ylabel('resid')
plt.show()

lm.summary()

#R-squared=0.85
```

```
C:\Users\DS\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
```

```
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```




```
C:\Users\DS\anaconda3\lib\site-packages\scipy\stats\stats.py:1603: UserWarning:
kurtosistest only valid for n>=20 … continuing anyway, n=10
  warnings.warn("kurtosistest only valid for n>=20 … continuing "
```

[5]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.850
Model:                            OLS   Adj. R-squared:                  0.831
Method:                 Least Squares   F-statistic:                     45.24
Date:                Thu, 01 Jul 2021   Prob (F-statistic):           0.000149
Time:                        17:40:21   Log-Likelihood:                -22.745
No. Observations:                  10   AIC:                             49.49
Df Residuals:                       8   BIC:                             50.10
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -2.2696      3.212     -0.707      0.500      -9.677       5.138
x              2.6087      0.388      6.726      0.000       1.714       3.503
==============================================================================
Omnibus:                        0.113   Durbin-Watson:                   2.267
Prob(Omnibus):                  0.945   Jarque-Bera (JB):                0.118
```

```
Skew:                           0.104   Prob(JB):                        0.943
Kurtosis:                       2.512   Cond. No.                         32.4
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

[6]:
```
#(c)
t=(2.6087-1.5)/0.388
print('t:',t)
print('pvalue:',2*sp.stats.t.sf(df=8,x=t))

# 검정통계량 t=2.86, pvalue=0.021>0.01 이므로 유의수준 0.01 에서 귀무가설을 기각하지 못한다
# 따라서 유의수준 0.01 에서 기울기 회귀계수=1.5라고 할 수 있다.
```

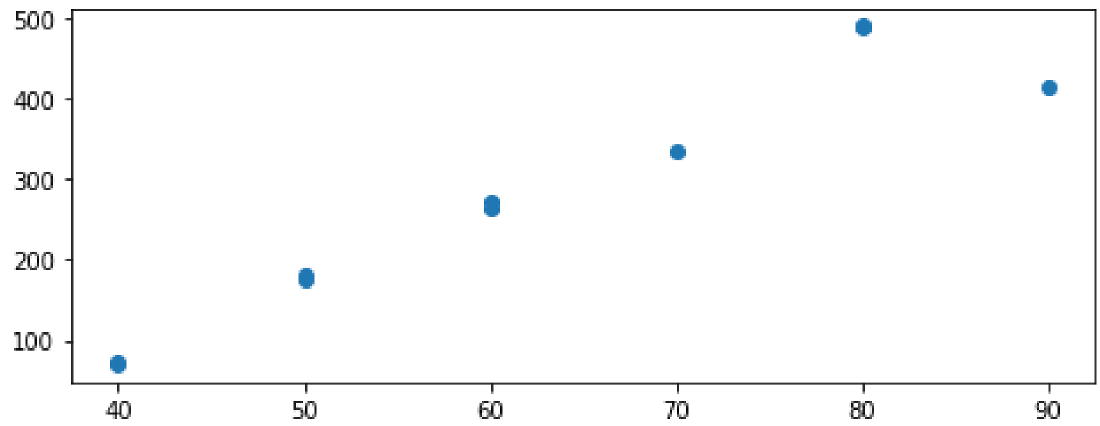
```
t: 2.857474226804123
pvalue: 0.0212336746680744
```

5번

[2]:
```
water=pd.read_csv("C:/PythonbookData/table9.6_water.csv")
print(water.head())
```

```
    X    Y
0  40   69
1  50  175
2  60  272
3  70  335
4  80  490
```

[5]:
```
#(a)
plt.figure(figsize=(8,3))
plt.scatter(water['X'],water['Y'])
plt.show()
```



[7]:
```
#(b)
lm_water=smf.ols("Y~X",water).fit()
print(lm_water.params)

#Y=-252.297+8.53X
```

```
Intercept   -252.297101
X              8.528986
dtype: float64
```

[9]:
```
#(c)
lm_water.summary()

# 검정통계량 9.318이며 pavlue=0<0.05이므로 유의수준 0.05에서 귀무가설을 기각한다.
# 따라서 유의수준 0.05에서 beta1=0이라고 할 수 없다.
```

```
C:\ProgramData\Anaconda3\lib\site-packages\scipy\stats\stats.py:1394:
UserWarning: kurtosistest only valid for n>=20 … continuing anyway, n=10
  "anyway, n=%i" % int(n))
```

[9]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.916
Model:                            OLS   Adj. R-squared:                  0.905
Method:                 Least Squares   F-statistic:                     86.83
Date:                Mon, 28 Jun 2021   Prob (F-statistic):           1.43e-05
Time:                        16:11:35   Log-Likelihood:                -51.804
No. Observations:                  10   AIC:                             107.6
Df Residuals:                       8   BIC:                             108.2
Df Model:                           1
```

```
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   -252.2971     58.751     -4.294      0.003    -387.777    -116.818
X              8.5290      0.915      9.318      0.000       6.418      10.640
==============================================================================
Omnibus:                        4.063   Durbin-Watson:                   2.455
Prob(Omnibus):                  0.131   Jarque-Bera (JB):                1.097
Skew:                          -0.712   Prob(JB):                        0.578
Kurtosis:                       3.776   Cond. No.                         248.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

[11]:
```python
#(d)
lm_water.summary()

# R-Squared=0.916
```

```
C:\ProgramData\Anaconda3\lib\site-packages\scipy\stats\stats.py:1394:
UserWarning: kurtosistest only valid for n>=20 … continuing anyway, n=10
  "anyway, n=%i" % int(n))
```

[11]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.916
Model:                            OLS   Adj. R-squared:                  0.905
Method:                 Least Squares   F-statistic:                     86.83
Date:                Mon, 28 Jun 2021   Prob (F-statistic):           1.43e-05
Time:                        16:13:33   Log-Likelihood:                -51.804
No. Observations:                  10   AIC:                             107.6
Df Residuals:                       8   BIC:                             108.2
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   -252.2971     58.751     -4.294      0.003    -387.777    -116.818
X              8.5290      0.915      9.318      0.000       6.418      10.640
==============================================================================
Omnibus:                        4.063   Durbin-Watson:                   2.455
Prob(Omnibus):                  0.131   Jarque-Bera (JB):                1.097
```

```
Skew:                          -0.712   Prob(JB):                        0.578
Kurtosis:                       3.776   Cond. No.                         248.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

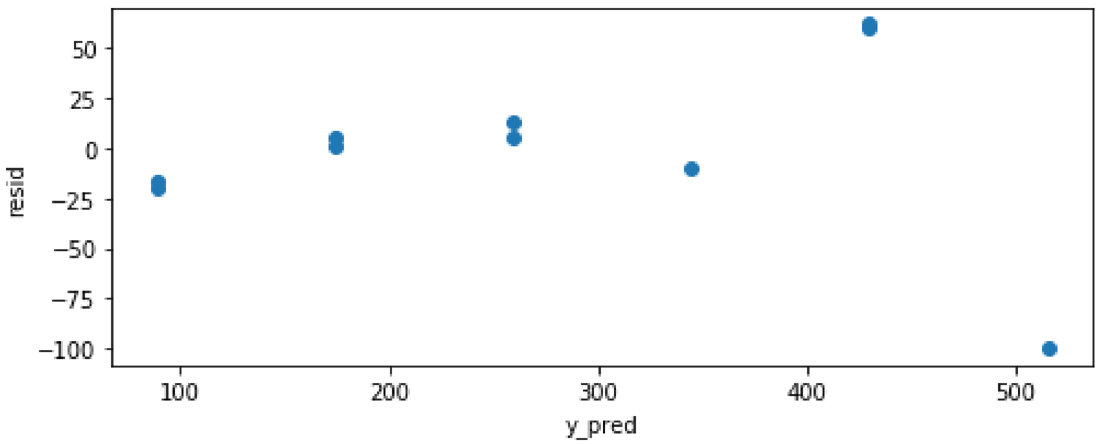
[14]:
```
#(e)
y_pred=lm_water.predict(water['X'])
y_pred
```

[14]:
```
0     88.862319
1    174.152174
2    259.442029
3    344.731884
4    430.021739
5    515.311594
6     88.862319
7    259.442029
8    430.021739
9    174.152174
dtype: float64
```

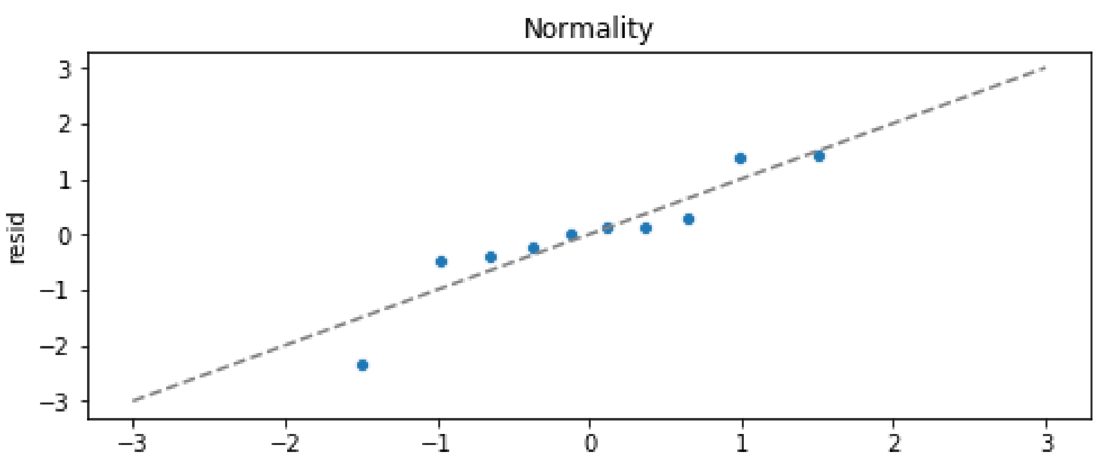
[16]:
```
#(f)
resid=water['Y']-y_pred
resid
```

[16]:
```
0    -19.862319
1      0.847826
2     12.557971
3     -9.731884
4     59.978261
5   -100.311594
6    -16.862319
7      5.557971
8     61.978261
9      5.847826
dtype: float64
```

[18]:
```
#(g)
plt.figure(figsize=(8,3))
plt.scatter(y_pred,resid)
plt.xlabel('y_pred')
plt.ylabel('resid')
plt.show()
```



[19]:
```
#(h)
plt.figure(figsize=(8,3))
sr=sp.stats.zscore(resid)
(a,b),_=sp.stats.probplot(sr)
sns.scatterplot(a,b)
plt.plot([-3,3],[-3,3],'--',color='grey')
plt.title("Normality"); plt.ylabel('resid')
plt.show()
```



[29]:
```
#(i)
lm_water.predict(pd.DataFrame({'X':[np.mean(water['X'])]}))
```

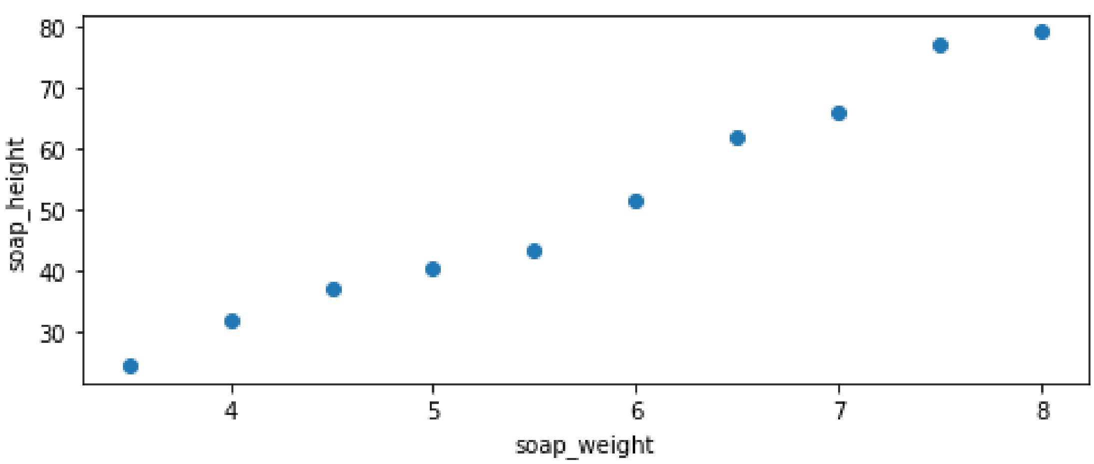
[29]:
```
0    276.5
dtype: float64
```

6번

[30]:
```
soap=pd.read_csv("C:/PythonbookData/table9.7_soap.csv")
print(soap.head())
```

```
   soap_weight  soap_height
0          3.5         24.4
1          4.0         32.1
2          4.5         37.1
3          5.0         40.4
4          5.5         43.3
```

[49]:
```
#(a)
plt.figure(figsize=(8,3))
plt.scatter(soap['soap_weight'],soap['soap_height'])
plt.xlabel('soap_weight')
plt.ylabel('soap_height')
plt.show()
```



[51]:
```
#(b)
lm_soap=smf.ols("soap_height~soap_weight+0",soap).fit()
print(lm_soap.params)

#Y=9.13soap_weight
```

```
soap_weight    9.130107
dtype: float64
```

[52]:
```
#(c)
lm_soap.summary()

# 검정통계량 29.409이며 pavlue=0<0.05이므로 유의수준 0.05에서 귀무가설을 기각한다.
```

```
# 따라서 유의수준 0.05에서 beta1=0이라고 할 수 없다.
```

```
C:\ProgramData\Anaconda3\lib\site-packages\scipy\stats\stats.py:1394:
UserWarning: kurtosistest only valid for n>=20 … continuing anyway, n=10
  "anyway, n=%i" % int(n))
```

[52]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:            soap_height   R-squared:                       0.990
Model:                            OLS   Adj. R-squared:                  0.989
Method:                 Least Squares   F-statistic:                     864.9
Date:                Mon, 28 Jun 2021   Prob (F-statistic):           2.97e-10
Time:                        16:44:14   Log-Likelihood:                -31.273
No. Observations:                  10   AIC:                             64.55
Df Residuals:                       9   BIC:                             64.85
Df Model:                           1
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
soap_weight     9.1301      0.310     29.409      0.000       8.428       9.832
==============================================================================
Omnibus:                        1.511   Durbin-Watson:                   0.366
Prob(Omnibus):                  0.470   Jarque-Bera (JB):                1.029
Skew:                           0.563   Prob(JB):                        0.598
Kurtosis:                       1.904   Cond. No.                         1.00
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

[54]:
```
#(d)
lm_soap.summary()

# R-Squared=0.99
# 추정된 회귀직선의 설명력이 높다고 할 수 있다.
```

```
C:\ProgramData\Anaconda3\lib\site-packages\scipy\stats\stats.py:1394:
UserWarning: kurtosistest only valid for n>=20 … continuing anyway, n=10
  "anyway, n=%i" % int(n))
```

[54]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
```

```
==============================================================================
Dep. Variable:            soap_height   R-squared:                       0.990
Model:                            OLS   Adj. R-squared:                  0.989
Method:                 Least Squares   F-statistic:                     864.9
Date:                Mon, 28 Jun 2021   Prob (F-statistic):           2.97e-10
Time:                        16:44:39   Log-Likelihood:                -31.273
No. Observations:                  10   AIC:                             64.55
Df Residuals:                       9   BIC:                             64.85
Df Model:                           1
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
soap_weight     9.1301      0.310     29.409      0.000       8.428       9.832
==============================================================================
Omnibus:                        1.511   Durbin-Watson:                   0.366
Prob(Omnibus):                  0.470   Jarque-Bera (JB):                1.029
Skew:                           0.563   Prob(JB):                        0.598
Kurtosis:                       1.904   Cond. No.                         1.00
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

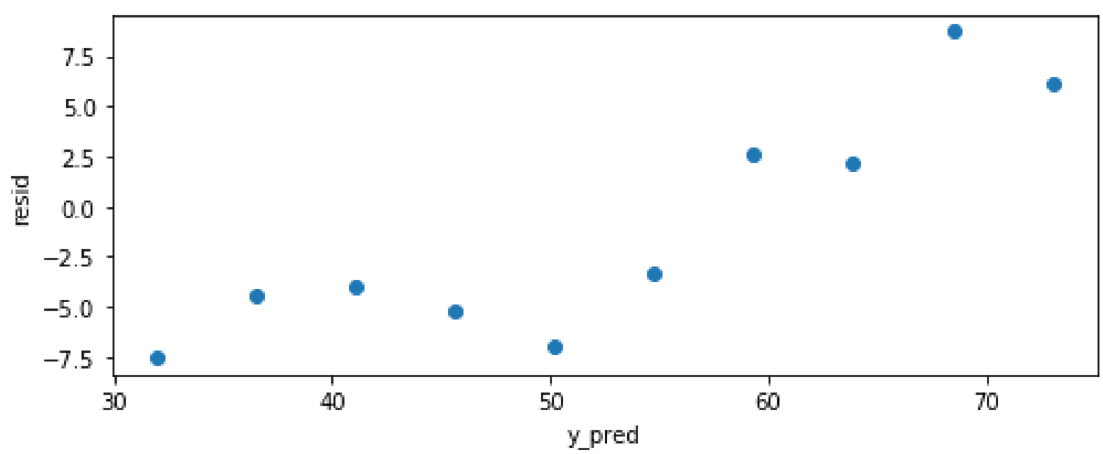
[55]:
```
#(e)
y_pred=lm_soap.predict(soap['soap_weight'])
y_pred
```

[55]:
```
0    31.955374
1    36.520427
2    41.085480
3    45.650534
4    50.215587
5    54.780641
6    59.345694
7    63.910747
8    68.475801
9    73.040854
dtype: float64
```

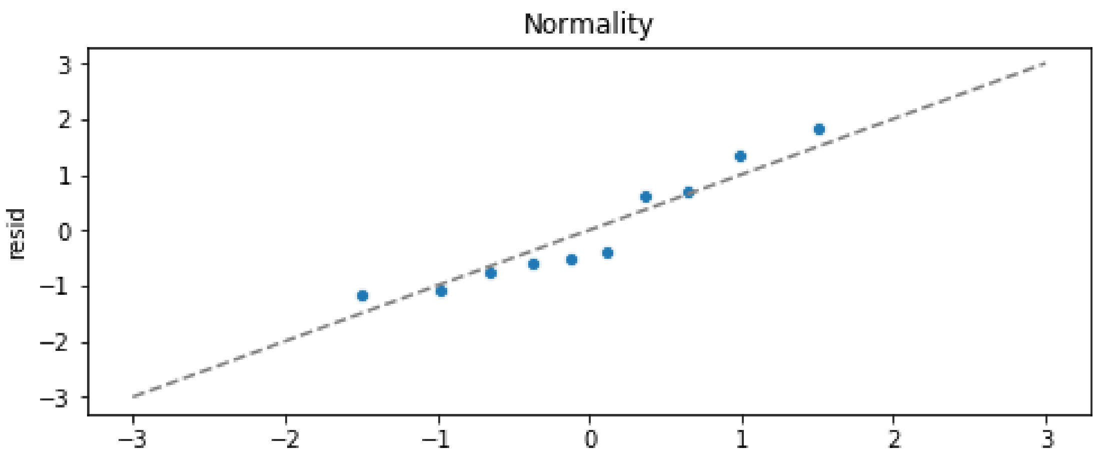
[56]:
```
#(f)
resid=soap['soap_height']-y_pred
resid
```

[56]:
```
0   -7.555374
1   -4.420427
2   -3.985480
3   -5.250534
4   -6.915587
5   -3.380641
6    2.554306
7    2.189253
8    8.724199
9    6.159146
dtype: float64
```

[57]:
```
#(g)
plt.figure(figsize=(8,3))
plt.scatter(y_pred,resid)
plt.xlabel('y_pred')
plt.ylabel('resid')
plt.show()
```



[58]:
```
#(h)
plt.figure(figsize=(8,3))
sr=sp.stats.zscore(resid)
(a,b),_=sp.stats.probplot(sr)
sns.scatterplot(a,b)
plt.plot([-3,3],[-3,3],'--',color='grey')
plt.title("Normality"); plt.ylabel('resid')
plt.show()
```




```
[61]: #(i)
lm_soap.predict(pd.DataFrame({'soap_weight':[5.3]}))
```

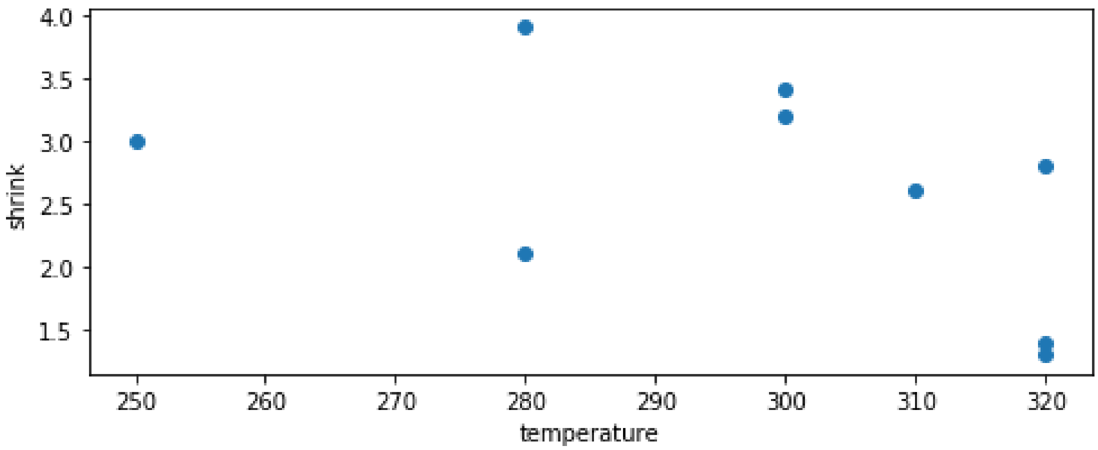
```
[61]: 0    48.389566
dtype: float64
```

7번

```
[10]: fiber=pd.read_csv("C:/PythonbookData/table9.8_fiber.csv")
print(fiber.head())
```

```
   batch_no  temperature  shrink
0         1          280     2.1
1         2          250     3.0
2         3          300     3.2
3         4          320     1.4
4         5          310     2.6
```

```
[11]: #(a)
plt.figure(figsize=(8,3))
plt.scatter(fiber['temperature'],fiber['shrink'])
plt.xlabel('temperature')
plt.ylabel('shrink')
plt.show()
```



[12]:
```
#(b)
lm_fiber=smf.ols("shrink~temperature",fiber).fit()
print(lm_fiber.params)

#Y=7.95-0.02temperature
```

```
Intercept       7.949756
temperature    -0.017854
dtype: float64
```

[13]:
```
#(c)
lm_fiber.summary()

# 검정통계량 t=-1.454이며 pavlue=0.189>0.05이므로 유의수준 0.05에서 귀무가설을 기각하지 못한다.
# 따라서 유의수준 0.05에서 추정된 회귀계수는 유의하지 않다고 할 수 있다.
```

```
C:\ProgramData\Anaconda3\lib\site-packages\scipy\stats\stats.py:1394:
UserWarning: kurtosistest only valid for n>=20 … continuing anyway, n=9
  "anyway, n=%i" % int(n))
```

[13]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                 shrink   R-squared:                       0.232
Model:                            OLS   Adj. R-squared:                  0.122
Method:                 Least Squares   F-statistic:                     2.114
Date:                Wed, 30 Jun 2021   Prob (F-statistic):              0.189
Time:                        17:34:03   Log-Likelihood:                -9.9491
No. Observations:                   9   AIC:                             23.90
Df Residuals:                       7   BIC:                             24.29
Df Model:                           1
```

```
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      7.9498      3.667      2.168      0.067      -0.721      16.620
temperature   -0.0179      0.012     -1.454      0.189      -0.047       0.011
==============================================================================
Omnibus:                        4.645   Durbin-Watson:                   2.431
Prob(Omnibus):                  0.098   Jarque-Bera (JB):                1.113
Skew:                          -0.100   Prob(JB):                        0.573
Kurtosis:                       1.289   Cond. No.                     3.96e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 3.96e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

[14]:

```python
#(d)
lm_fiber.summary()

# R-Squared=0.232
# 추정된 회귀계수의 설명력이 낮은 편이라고 할 수 있다.
```

```
C:\ProgramData\Anaconda3\lib\site-packages\scipy\stats\stats.py:1394:
UserWarning: kurtosistest only valid for n>=20 … continuing anyway, n=9
  "anyway, n=%i" % int(n))
```

[14]:

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                 shrink   R-squared:                       0.232
Model:                            OLS   Adj. R-squared:                  0.122
Method:                 Least Squares   F-statistic:                     2.114
Date:                Wed, 30 Jun 2021   Prob (F-statistic):              0.189
Time:                        17:34:07   Log-Likelihood:                -9.9491
No. Observations:                   9   AIC:                             23.90
Df Residuals:                       7   BIC:                             24.29
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      7.9498      3.667      2.168      0.067      -0.721      16.620
temperature   -0.0179      0.012     -1.454      0.189      -0.047       0.011
```

```
==============================================================================
Omnibus:                        4.645   Durbin-Watson:                   2.431
Prob(Omnibus):                  0.098   Jarque-Bera (JB):                1.113
Skew:                          -0.100   Prob(JB):                        0.573
Kurtosis:                       1.289   Cond. No.                     3.96e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 3.96e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

[15]:
```python
#(e)
y_pred=lm_fiber.predict(fiber['temperature'])
y_pred
```

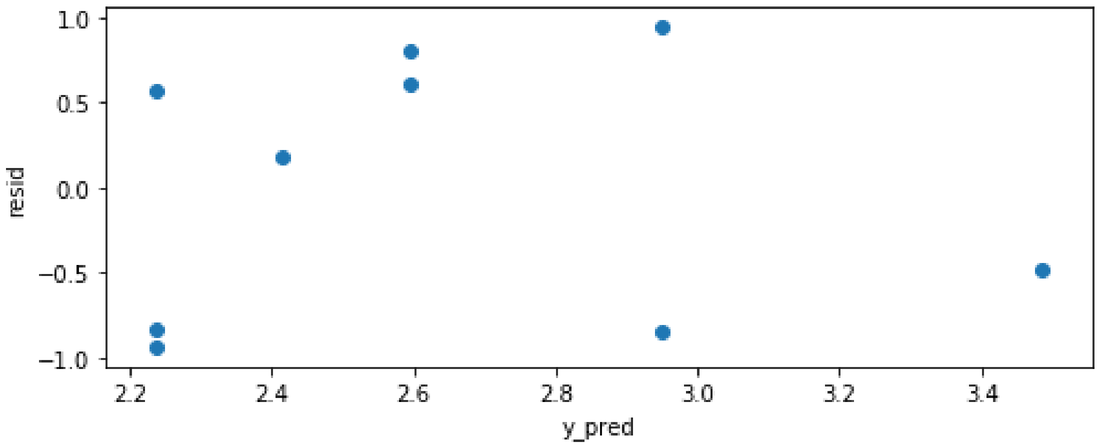
[15]:
```
0    2.950732
1    3.486341
2    2.593659
3    2.236585
4    2.415122
5    2.950732
6    2.236585
7    2.593659
8    2.236585
dtype: float64
```

[16]:
```python
#(f)
resid=fiber['shrink']-y_pred
resid
```

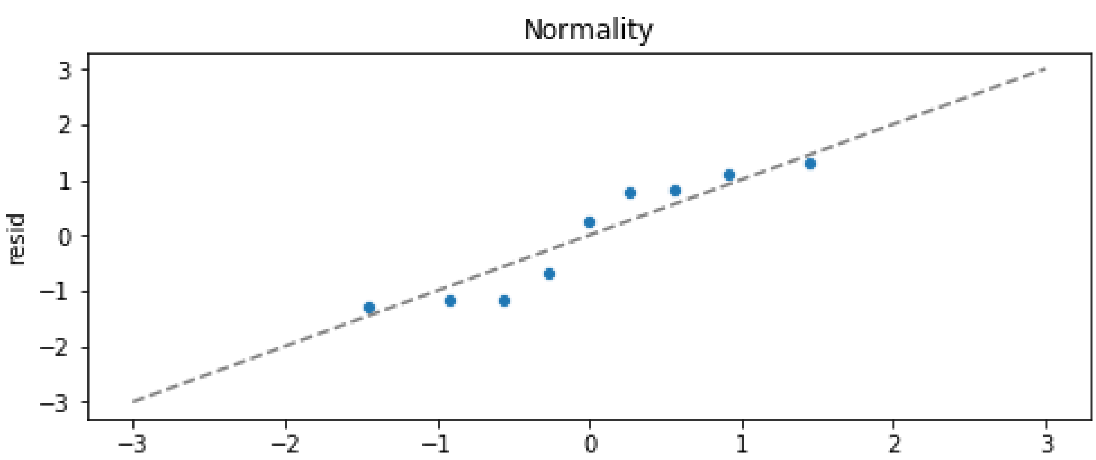
[16]:
```
0   -0.850732
1   -0.486341
2    0.606341
3   -0.836585
4    0.184878
5    0.949268
6   -0.936585
7    0.806341
8    0.563415
dtype: float64
```

[17]:
```python
#(g)
plt.figure(figsize=(8,3))
plt.scatter(y_pred,resid)
```

```
plt.xlabel('y_pred')
plt.ylabel('resid')
plt.show()
```



[18]:
```
#(h)
plt.figure(figsize=(8,3))
sr=sp.stats.zscore(resid)
(a,b),_=sp.stats.probplot(sr)
sns.scatterplot(a,b)
plt.plot([-3,3],[-3,3],'--',color='grey')
plt.title("Normality"); plt.ylabel('resid')
plt.show()
```



[19]:
```
#(i)
lm_fiber.predict(pd.DataFrame({'temperature':[400]}))
```

[19]:
```
0    0.808293
dtype: float64
```

8번

[80]:
```
x=[4,6,6,8,8,8,9,9,10,12]
y=[9,10,18,20,15,17,20,22,25,30]
df=pd.DataFrame({'y':y,'x':x})
df
```

[80]:

	y	x
0	9	4
1	10	6
2	18	6
3	20	8
4	15	8
5	17	8
6	20	9
7	22	9
8	25	10
9	30	12

[83]:
```
#(a)
lm=smf.ols("y~x",df).fit()
print(lm.params)

#y=-2.27+2.61x
```

```
Intercept   -2.269565
x            2.608696
dtype: float64
```

[86]:
```
#(b)
df['xx']=df['x']**2
lm=smf.ols("y~x+xx",df).fit()
print(lm.params)

#y=4.08+0.88x+0.11x^2
```

```
Intercept    4.081928
x            0.881460
xx           0.108839
dtype: float64
```

[89]:
```python
#(c)
lm=smf.ols("y~x+0",df).fit()
print(lm.params)

#y=2.344x
```

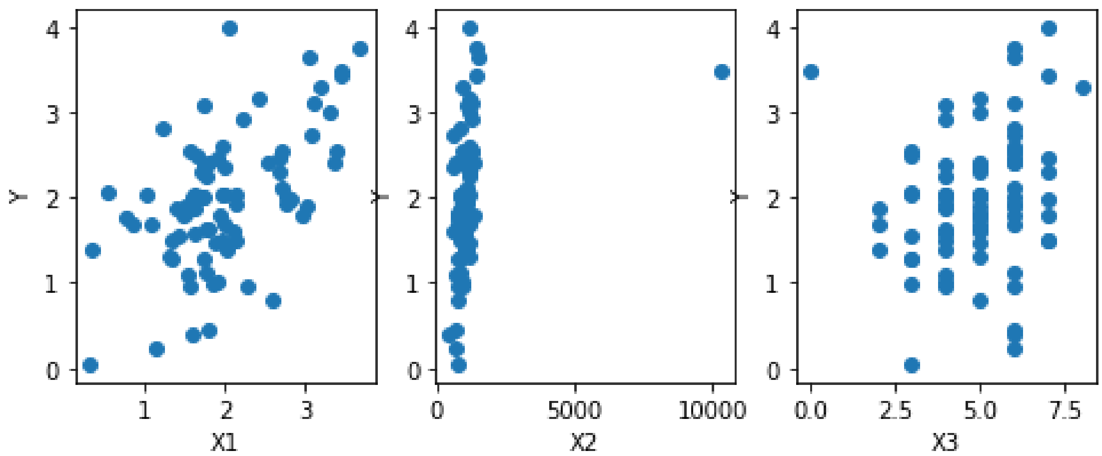
```
x    2.344023
dtype: float64
```

9번

[90]:
```python
univ=pd.read_csv("C:/PythonbookData/table9.9_Univ.csv")
print(univ.head())
```

```
      Y    X1    X2  X3
0  2.04  2.01  1070   5
1  2.56  3.40  1254   6
2  3.75  3.68  1466   6
3  1.10  1.54   706   4
4  3.00  3.32  1160   5
```

[92]:
```python
#(a)
plt.figure(figsize=(8,3))
plt.subplot(1,3,1)
plt.scatter(univ['X1'],univ['Y'])
plt.xlabel('X1')
plt.ylabel('Y')

plt.subplot(1,3,2)
plt.scatter(univ['X2'],univ['Y'])
plt.xlabel('X2')
plt.ylabel('Y')

plt.subplot(1,3,3)
plt.scatter(univ['X3'],univ['Y'])
plt.xlabel('X3')
plt.ylabel('Y')

plt.show()
```



[94]:
```python
#(b)
lm_univ=smf.ols("Y~X1+X2+X3",univ).fit()
print(lm_univ.params)

#Y=0.696+0.505x1+0.0002x2+0.024x3
```

```
Intercept    0.695528
X1           0.505467
X2           0.000151
X3           0.024384
dtype: float64
```

[99]:
```python
#(c)
lm_univ.summary()

# 검정통계량 3.78이며 pavlue=0<0.05이므로 유의수준 0.05에서 귀무가설을 기각한다.
# 따라서 유의수준 0.05에서 beta1=0이라고 할 수 없다.
```

[99]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.331
Model:                            OLS   Adj. R-squared:                  0.305
Method:                 Least Squares   F-statistic:                     12.56
Date:                Mon, 28 Jun 2021   Prob (F-statistic):           9.38e-07
Time:                        17:06:45   Log-Likelihood:                -78.435
No. Observations:                  80   AIC:                             164.9
Df Residuals:                      76   BIC:                             174.4
Df Model:                           3
Covariance Type:            nonrobust
```

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.6955      0.315      2.205      0.030       0.067       1.324
X1             0.5055      0.134      3.780      0.000       0.239       0.772
X2             0.0002   8.78e-05      1.721      0.089   -2.38e-05       0.000
X3             0.0244      0.069      0.354      0.725      -0.113       0.162
==============================================================================
Omnibus:                        0.489   Durbin-Watson:                   2.134
Prob(Omnibus):                  0.783   Jarque-Bera (JB):                0.118
Skew:                           0.017   Prob(JB):                        0.943
Kurtosis:                       3.185   Cond. No.                     6.67e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 6.67e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

[101]:
```python
#(d)
lm_univ.summary()

# 검정통계량 1.721이며 pavlue=0.089>0.05이므로 유의수준 0.05에서 귀무가설을 기각하지 못한다.
# 따라서 유의수준 0.05에서 beta2=0이라고 할 수 있다.
```

[101]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.331
Model:                            OLS   Adj. R-squared:                  0.305
Method:                 Least Squares   F-statistic:                     12.56
Date:                Mon, 28 Jun 2021   Prob (F-statistic):           9.38e-07
Time:                        17:07:37   Log-Likelihood:                -78.435
No. Observations:                  80   AIC:                             164.9
Df Residuals:                      76   BIC:                             174.4
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.6955      0.315      2.205      0.030       0.067       1.324
X1             0.5055      0.134      3.780      0.000       0.239       0.772
X2             0.0002   8.78e-05      1.721      0.089   -2.38e-05       0.000
X3             0.0244      0.069      0.354      0.725      -0.113       0.162
```

```
==============================================================================
Omnibus:                        0.489   Durbin-Watson:                   2.134
Prob(Omnibus):                  0.783   Jarque-Bera (JB):                0.118
Skew:                           0.017   Prob(JB):                        0.943
Kurtosis:                       3.185   Cond. No.                     6.67e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 6.67e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

[102]:
```
#(e)
lm_univ.summary()

# 검정통계량 0.354이며 pavlue=0.725>0.05이므로 유의수준 0.05에서 귀무가설을 기각하지 못한다.
# 따라서 유의수준 0.05에서 beta3=0이라고 할 수 있다.
```

[102]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.331
Model:                            OLS   Adj. R-squared:                  0.305
Method:                 Least Squares   F-statistic:                     12.56
Date:                Mon, 28 Jun 2021   Prob (F-statistic):           9.38e-07
Time:                        17:08:32   Log-Likelihood:                -78.435
No. Observations:                  80   AIC:                             164.9
Df Residuals:                      76   BIC:                             174.4
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.6955      0.315      2.205      0.030       0.067       1.324
X1             0.5055      0.134      3.780      0.000       0.239       0.772
X2             0.0002   8.78e-05      1.721      0.089   -2.38e-05       0.000
X3             0.0244      0.069      0.354      0.725      -0.113       0.162
==============================================================================
Omnibus:                        0.489   Durbin-Watson:                   2.134
Prob(Omnibus):                  0.783   Jarque-Bera (JB):                0.118
Skew:                           0.017   Prob(JB):                        0.943
Kurtosis:                       3.185   Cond. No.                     6.67e+03
==============================================================================
```

```
Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 6.67e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

[104]:
```python
#(f)
lm_univ.summary()

#R-squared=0.331
#추정된 회귀직선의 설명력이 낮은편이라고 할 수 있다.
```

[104]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.331
Model:                            OLS   Adj. R-squared:                  0.305
Method:                 Least Squares   F-statistic:                     12.56
Date:                Mon, 28 Jun 2021   Prob (F-statistic):           9.38e-07
Time:                        17:09:03   Log-Likelihood:                -78.435
No. Observations:                  80   AIC:                             164.9
Df Residuals:                      76   BIC:                             174.4
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.6955      0.315      2.205      0.030       0.067       1.324
X1             0.5055      0.134      3.780      0.000       0.239       0.772
X2             0.0002   8.78e-05      1.721      0.089   -2.38e-05       0.000
X3             0.0244      0.069      0.354      0.725      -0.113       0.162
==============================================================================
Omnibus:                        0.489   Durbin-Watson:                   2.134
Prob(Omnibus):                  0.783   Jarque-Bera (JB):                0.118
Skew:                           0.017   Prob(JB):                        0.943
Kurtosis:                       3.185   Cond. No.                     6.67e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 6.67e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

10번

[105]:
```
la=pd.read_csv("C:/PythonbookData/table9.10_LA_Olympic.csv")
print(la.head())
```

```
   country   m100   m200   m400   m800  m1500  m3000  marathon
0  Austria  11.43  23.09  50.62  1.99   4.22   9.34    159.37
1  Belgium  11.41  23.04  52.00  2.00   4.14   8.88    157.85
2   Brazil  11.31  23.17  52.80  2.10   4.49   9.77    168.75
3   Canada  11.00  22.25  50.06  2.00   4.06   8.81    149.45
4    China  11.95  24.41  54.97  2.08   4.33   9.31    168.48
```

[106]:
```
#(a)
la.describe()
```

[106]:
```
            m100       m200       m400       m800      m1500      m3000  \
count  24.000000  24.000000  24.000000  24.000000  24.000000  24.000000
mean   11.578333  23.489583  52.886250   2.046250   4.235417   9.255417
std     0.388572   0.804982   1.795186   0.073118   0.202334   0.548722
min    10.790000  21.830000  49.290000   1.920000   3.950000   8.500000
25%    11.305000  23.030000  51.672500   2.000000   4.135000   8.840000
50%    11.590000  23.530000  53.195000   2.035000   4.170000   9.110000
75%    11.830000  24.112500  54.375000   2.092500   4.350000   9.657500
max    12.300000  25.000000  55.700000   2.190000   4.690000  10.460000

         marathon
count   24.000000
mean   164.443750
std     14.566807
min    142.720000
25%    153.815000
50%    160.095000
75%    171.825000
max    200.370000
```

[108]:
```
#(b)
sp.stats.pearsonr(la['m100'],la['m200'])[0]
```

[108]: 0.9517436599463605

[110]:
```
#(c)
sp.stats.pearsonr(la['m100'],la['marathon'])[0]
```

[110]: 0.6395764929127904

[111]:
```
#(d)
sp.stats.pearsonr(la['m3000'],la['marathon'])[0]
```

[111]: 0.8152014266830425

11번

[130]:
```
company=pd.read_csv("C:/PythonbookData/table9.11_1999_company.csv")
company.head()
```

[130]:

	firm	salary	tenure	age	sales	profits	assets
0	1	3030	7	61	161315	2956	257389
1	2	6050	0	51	144416	22071	237545
2	3	3571	11	63	139208	4430	49271
3	4	3300	6	60	100697	6370	92630
4	5	0	18	63	100469	9296	355935

[131]:
```
#(a)
lm_company=smf.ols("salary~profits",company).fit()
print(lm_company.params)

#salary=3311.911+0.135profits
```

```
Intercept    3311.911072
profits         0.135036
dtype: float64
```

[134]:
```
#(b)
lm_company1=smf.ols("salary~profits+age",company).fit()
print(lm_company1.params)

#salary=23389.96+0.000431profits-321.596age
```

```
Intercept    23389.960409
profits          0.000431
age           -321.575606
dtype: float64
```

[135]:
```
#(c)
lm_company2=smf.ols("salary~profits+age+sales",company).fit()
print(lm_company2.params)

#salary=23248.064+0.018profits-314.154age-0.00465sales
```

```
Intercept    23248.064386
profits          0.018070
age           -314.154422
sales           -0.004650
dtype: float64
```

[138]:
```python
#(d)
lm_company3=smf.ols("salary~profits+age+sales+tenure+assets",company).fit()
print(lm_company3.params)

#salary=35367.269-0.111profits-529.79age-0.0008sales+24.462tenure+0.005assets
```

```
Intercept    35367.269071
profits         -0.111277
age           -529.789738
sales           -0.000783
tenure          24.462120
assets           0.004955
dtype: float64
```

12번

[142]:
```python
pizza=pd.read_csv("C:/PythonbookData/table9.12_pizza_delivery.csv")
print(pizza.head())

lm_pizza=smf.ols("delivery_time~dist+0",pizza).fit()
lm_pizza.params

#delivery_time=0.122dist
```

```
   order  dist  delivery_time
0      1   100             21
1      2   150             30
2      3   300             32
3      4   400             47
4      5   130             26
```

[142]:
```
dist    0.122462
dtype: float64
```

10장 연습문제

```
[1]: import numpy as np
import math as m
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
import scipy as sp
import statsmodels.formula.api as smf
import statsmodels.api as sm
from sklearn.linear_model import LinearRegression
from statsmodels.graphics.factorplots import interaction_plot
```

1번

(b)

$Y_{ij} = \mu + \tau_i + \epsilon_{ij} i = 1, 2, 3, j = 1, 2, \ldots 6$

$\sum_{i=1}^{3} n_i \tau_i = 0$

```
[2]: A=[100,96,98,96,92,100]
B=[76,80,84,84,78,80]
c=[108,100,101,103,104,110]
```

```
[3]: #(a)
# H0: tauA=tauB=tauC=0 (세 회사 제품의 전구 수명 간에 차이가 없다.)
 # H1: not H0(세 회사 제품의 전구 수명 간에 차이가 있다.)
```

```
[4]: #(c)
data=pd.DataFrame({'group':np.hstack([np.tile("A",6),np.tile("B",6),np.
 ↪tile("C",6)]),'freq': np.hstack([A,B,c])})
data.head()

aov=smf.ols("freq~group",data=data).fit()
sm.stats.anova_lm(aov,typ=2)


# 검정통계량 F=77.94이고 pvalue=1.19e-08<0.05이므로 유의수준 0.05에서 귀무가설을 기각한다,
# 따라서 유의수준 0.05에서 세 회사 제품의 전구 수명간에 차이가 있다고 할 수 있다.
```

[4]:

	sum_sq	df	F	PR(>F)
group	1815.111111	2.0	77.938931	1.189983e-08
Residual	174.666667	15.0	NaN	NaN

2번

[5]:
```
A=[10,12,12,10] B=[8
,9,10,12] c=[10,12,
14,12] D=[20,18,17,
19] E=[18,20,22,23]

data1=pd.DataFrame({'group':np.hstack([np.tile("A",4),np.tile("B",4),np.tile
 ("C",4),np.tile("D",4),np.tile("E",4)]),'freq': np.hstack([A,B,c,D,E])})
data1.head()
```

[5]:

	group	freq
0	A	10
1	A	12
2	A	12
3	A	10
4	B	8

[6]:
```
aov1=smf.ols("freq~group",data=data1).fit()
sm.stats.anova_lm(aov1)

# H0: tauA=tauB=tauC=tauD=tauE=0 (5가지 배양법에 따라 클로버 잎새 질소 성분 함량에 차이가 없다.)
# H1: not H0 (5가지 배양법에 따라 클로버 잎새 질소 성분 함량에 차이가 있다.)

# 검정통계량 F=35.58이고 pvalue=1.72e-07<0.05이므로 유의수준 0.05에서 귀무가설을 기각한다,
# 따라서 유의수준 0.05에서 5가지 배양법에 따라 클로버 잎새 질소 성분 함량에 차이가 있다고 할 수 있다.
```

[6]:

	df	sum_sq	mean_sq	F	PR(>F)
group	4.0	384.3	96.075	35.583333	1.721017e-07
Residual	15.0	40.5	2.700	NaN	NaN

3번

[7]:
```
A=[26,23,27]
B=[19,20,20,17]
c=[28,28,29]
D=[29,29,27,34]

data2=pd.DataFrame({'group':np.hstack([np.tile("A",3),np.tile("B",4),np.
↪tile("C",3),np.tile("D",4)]),'freq': np.hstack([A,B,c,D])})
data2.head()
```

[7]:

	group	freq
0	A	26

```
1      A     23
2      A     27
3      B     19
4      B     20
```

[8]:

```python
aov2=smf.ols("freq~group",data=data2).fit()
sm.stats.anova_lm(aov2)

# H0: tauA=tauB=tauC=tauD=0 (4가지 필터 종류에 따라 여과능력에 차이가 없다.)
# H1: not H0 (4가지 필터 종류에 따라 여과능력에 차이가 있다.)

# 검정통계량 F=21.02이고 pvalue=0.000122<0.05이므로 유의수준 0.05에서 귀무가설을 기각한다.
# 따라서 유의수준 0.05에서 4가지 필터 종류에 따라 여과 능력에 차이가 있다고 할 수 있다.
```

[8]:

	df	sum_sq	mean_sq	F	PR(>F)
group	3.0	265.345238	88.448413	21.017445	0.000122
Residual	10.0	42.083333	4.208333	NaN	NaN

4번

[9]:

```python
A=[7.3,10,8.5,6.3,7.9,7.5,8.2]
B=[10.2,11.2,10.8,12.8]
c=[9.1,9.5,9.7,10.2,9.4]

data3=pd.DataFrame({'group':np.hstack([np.tile("A",7),np.tile("B",4),np.
 ↪tile("C",5)]),'freq': np.hstack([A,B,c])})
data3.head()
```

[9]:

	group	freq
0	A	7.3
1	A	10.0
2	A	8.5
3	A	6.3
4	A	7.9

[10]:

```python
aov3=smf.ols("freq~group",data=data3).fit()
sm.stats.anova_lm(aov3)

# H0: tauA=tauB=tauC=0 (세 가지 종류의 모기약에 따라 살충효과가 같다.)
# H1: not H0 (세 가지 종류의 모기약에 따라 살충효과가 다르다.)

# 검정통계량 F=14.92이고 pvalue=0.00043<0.05이므로 유의수준 0.05에서 귀무가설을 기각한다,
# 따라서 유의수준 0.05에서 세 가지 종류의 모기약에 따라 살충 효과가 다르다고 할 수 있다.
```

[10]:

	df	sum_sq	mean_sq	F	PR(>F)
group	2.0	28.222357	14.111179	14.920146	0.00043

```
Residual   13.0   12.295143    0.945780         NaN       NaN
```

5번

[11]:
```python
A=[20,22,24]
B=[12,14,16]
c=[13,16,19]

data4=pd.DataFrame({'group':np.hstack([np.tile("A",3),np.tile("B",3),np..
↪tile("C",3)]),'freq': np.hstack([A,B,c])})
data4.head()
```

[11]:

	group	freq
0	A	20
1	A	22
2	A	24
3	B	12
4	B	14

[12]:
```python
aov4=smf.ols("freq~group",data=data4).fit()
sm.stats.anova_lm(aov4)

# H0: tauA=tauB=tauC=0 (세탁 세제에 따라 효과가 같다.)
 # H1: not H0 (세탁 세제에 따라 효과가 다르다.)

# 검정통계량 F=9.18이고 pvalue=0.015<0.05이므로 유의수준 0.05에서 귀무가설을 기각한다,
# 따라서 유의수준 0.05에서 세탁 세제에 따라 효과가 다르다고 할 수 있다.
```

[12]:

	df	sum_sq	mean_sq	F	PR(>F)
group	2.0	104.0	52.000000	9.176471	0.014955
Residual	6.0	34.0	5.666667	NaN	NaN

6번

[13]:
```python
A=[28,28,29,27]
B=[31,29,30,30]
c=[30,30,31,31]
D=[27,26,26,27]

data5=pd.DataFrame({'group':np.hstack([np.tile("A",4),np.tile("B",4),np.
↪tile("C",4),np.tile("D",4)]),'freq': np.hstack([A,B,c,D])})
data5.head()
```

[13]:

	group	freq
0	A	28
1	A	28
2	A	29
3	A	27
4	B	31

[14]:

```
aov5=smf.ols("freq~group",data=data5).fit()
sm.stats.anova_lm(aov5)

# H0: tauA=tauB=tauC=tauD=0 (4가지 여과지 종류에 따라 불순물 제거 능력에 차이가 없다.)
# H1: not H0 (4가지 여과지 종류에 따라 불순물 제거 능력에 차이가 있다.)

# 검정통계량 F=27.33이고 pvalue=0.000012<0.05이므로 유의수준 0.05에서 귀무가설을 기각한다,
# 따라서 유의수준 0.05에서 4가지 여과지 종류에 따라
    불순물 제거 능력에 차이가 있다고 할 수 있다.
```

[14]:

	df	sum_sq	mean_sq	F	PR(>F)
group	3.0	41.0	13.666667	27.333333	0.000012
Residual	12.0	6.0	0.500000	NaN	NaN

7번

[15]:

```
pressure=[100,100,120,120,150,150]
temp=[50,100,50,100,50,100]
freq=[9,13,12,18,14,20]

data6=pd.DataFrame({'pressure':pressure, 'temp':temp, 'freq':freq})
data6.head()
```

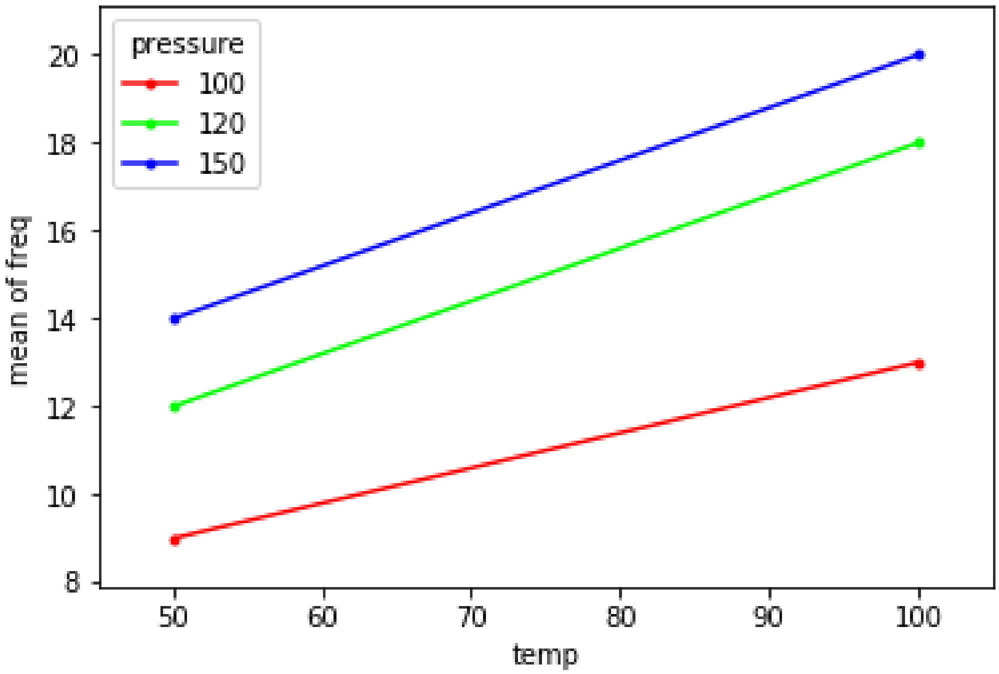
[15]:

	pressure	temp	freq
0	100	50	9
1	100	100	13
2	120	50	12
3	120	100	18
4	150	50	14

[16]:

```
#(a)
plt.figure(figsize=(8,3))
interaction_plot(x=data6['temp'],trace=data6['pressure'],response=data6['freq'],xlabel='temp')
plt.show()
```

<Figure size 576x216 with 0 Axes>



[18]:
```
#(b)
formula='freq ~ C(temp) + C(pressure)'
aov6=smf.ols(formula,data=data6).fit()
sm.stats.anova_lm(aov6)
```

[18]:

	df	sum_sq	mean_sq	F	PR(>F)
C(temp)	1.0	42.666667	42.666667	64.0	0.015268
C(pressure)	2.0	37.333333	18.666667	28.0	0.034483
Residual	2.0	1.333333	0.666667	NaN	NaN

[19]:
```
#(c)
sm.stats.anova_lm(aov6)

# H0: 온도에 따라 접착제 강도에 차이가 없다.
 # H1: 온도에 따라 접착제 강도에 차이가 있다.

# 검정통계량 F=64이고 pvalue=0.015<0.05이므로 유의수준 0.05에서 귀무가설을 기각한다,
# 따라서 유의수준 0.05에서 온도에 따라 접착제 강도에 차이가 있다고 할 수 있다.
```

[19]:

	df	sum_sq	mean_sq	F	PR(>F)
C(temp)	1.0	42.666667	42.666667	64.0	0.015268
C(pressure)	2.0	37.333333	18.666667	28.0	0.034483
Residual	2.0	1.333333	0.666667	NaN	NaN

[20]:
```
#(d)
sm.stats.anova_lm(aov6)

 # H0: 압력에 따라 접착제 강도에 차이가 없다.
  # H1:  압력에 따라 접착제 강도에 차이가 있다.

# 검정통계량 F=28이고 pvalue=0.034<0.05이므로 유의수준 0.05에서 귀무가설을 기각한다,
# 따라서 유의수준 0.05에서 압력에 따라 접착제 강도에 차이가 있다고 할 수 있다.
```

[20]:

	df	sum_sq	mean_sq	F	PR(>F)
C(temp)	1.0	42.666667	42.666667	64.0	0.015268
C(pressure)	2.0	37.333333	18.666667	28.0	0.034483
Residual	2.0	1.333333	0.666667	NaN	NaN

8번

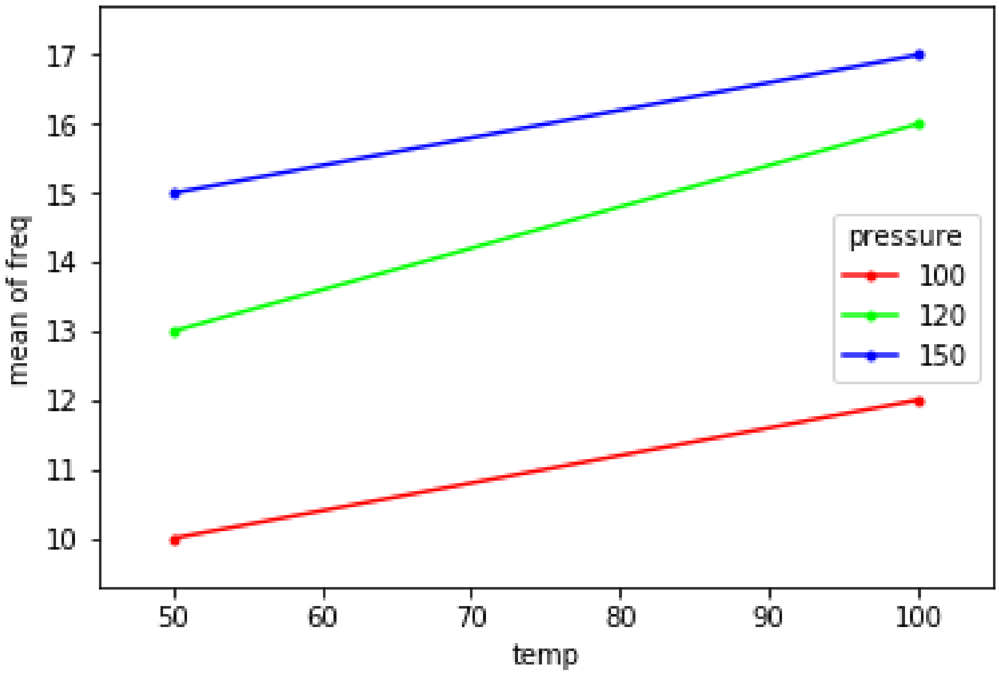
[3]:
```
pressure=[100,100,100,100,120,120,120,120,150,150,150,150]
temp=[50,50,100,100,50,50,100,100,50,50,100,100]
freq=[9,11,11,13,12,14,14,18,14,16,14,20]

data7=pd.DataFrame({'pressure':pressure, 'temp':temp, 'freq':freq})
data7.head()
```

[3]:

	pressure	temp	freq
0	100	50	9
1	100	50	11
2	100	100	11
3	100	100	13
4	120	50	12

[4]:
```
#(a)
plt.figure(figsize=(8,3))
interaction_plot(x=data7['temp'],trace=data7['pressure'],response=data7['freq'],xlabel='temp')
plt.show()
```

```
<Figure size 576x216 with 0 Axes>
```



[5]:
```
#(b)
formula='freq ~ C(temp) + C(pressure) + C(temp):C(pressure)'
aov7=smf.ols(formula,data=data7).fit()
sm.stats.anova_lm(aov7)
```

[5]:

	df	sum_sq	mean_sq	F	PR(>F)
C(temp)	1.0	16.333333	16.333333	2.882353	0.140475
C(pressure)	2.0	52.666667	26.333333	4.647059	0.060378
C(temp):C(pressure)	2.0	0.666667	0.333333	0.058824	0.943410
Residual	6.0	34.000000	5.666667	NaN	NaN

[6]:
```
#(c)
sm.stats.anova_lm(aov7)

 # H0: 온도에 따라 접착제 강도에 차이가 없다.
 # H1: 온도에 따라 접착제 강도에 차이가 있다.

# 검정통계량 F=2.88이고 pvalue=0.14>0.05이므로 유의수준 0.05에서 귀무가설을 기각하지 못한다,
# 따라서 유의수준 0.05에서 온도에 따라 접착제 강도에 차이가 있다고 할 수 없다.
```

[6]:

	df	sum_sq	mean_sq	F	PR(>F)
C(temp)	1.0	16.333333	16.333333	2.882353	0.140475
C(pressure)	2.0	52.666667	26.333333	4.647059	0.060378
C(temp):C(pressure)	2.0	0.666667	0.333333	0.058824	0.943410

```
Residual              6.0  34.000000   5.666667        NaN        NaN
```

[8]:
```
#(d)
sm.stats.anova_lm(aov7)

 # H0: 압력에 따라 접착제 강도에 차이가 없다.
 # H1: 압력에 따라 접착제 강도에 차이가 있다.

# 검정통계량 F=4.65이고 pvalue=0.06>0.05이므로 유의수준 0.05에서 귀무가설을 기각하지 못한다,
# 따라서 유의수준 0.05에서 압력에 따라 접착제 강도에 차이가 있다고 할 수 없다.
```

[8]:

	df	sum_sq	mean_sq	F	PR(>F)
C(temp)	1.0	16.333333	16.333333	2.882353	0.140475
C(pressure)	2.0	52.666667	26.333333	4.647059	0.060378
C(temp):C(pressure)	2.0	0.666667	0.333333	0.058824	0.943410
Residual	6.0	34.000000	5.666667	NaN	NaN

[10]:
```
#(e)
sm.stats.anova_lm(aov7)

 # H0: 온도와 압력에 따른 상호작용이 없다.
 # H1: 온도와 압력에 따른 상호작용이 있다.

# 검정통계량 F=0.06이고 pvalue=0.94>0.05이므로 유의수준 0.05에서 귀무가설을 기각하지 못한다,
# 따라서 유의수준 0.05에서 온도와 압력에 따른 상호작용이 있다고 할 수 없다.
```

[10]:

	df	sum_sq	mean_sq	F	PR(>F)
C(temp)	1.0	16.333333	16.333333	2.882353	0.140475
C(pressure)	2.0	52.666667	26.333333	4.647059	0.060378
C(temp):C(pressure)	2.0	0.666667	0.333333	0.058824	0.943410
Residual	6.0	34.000000	5.666667	NaN	NaN

9번

[11]:
```
A=[20,20,30,30,10,10,20,20,30,30,10,10,20,20,30,30,10,10]
B=[20,20,20,20,20,20,40,40,40,40,40,40,60,60,60,60,60,60]
freq=[3.9,3.7,6.5,6.1,7.7,8.7,4.2,4.4,9.8,9.2,14.3,13.7,6.9,7.3,12.7,13.1,16.
↪8,17.2]

data8=pd.DataFrame({'A':A, 'B':B, 'freq':freq})
data8.head()
```

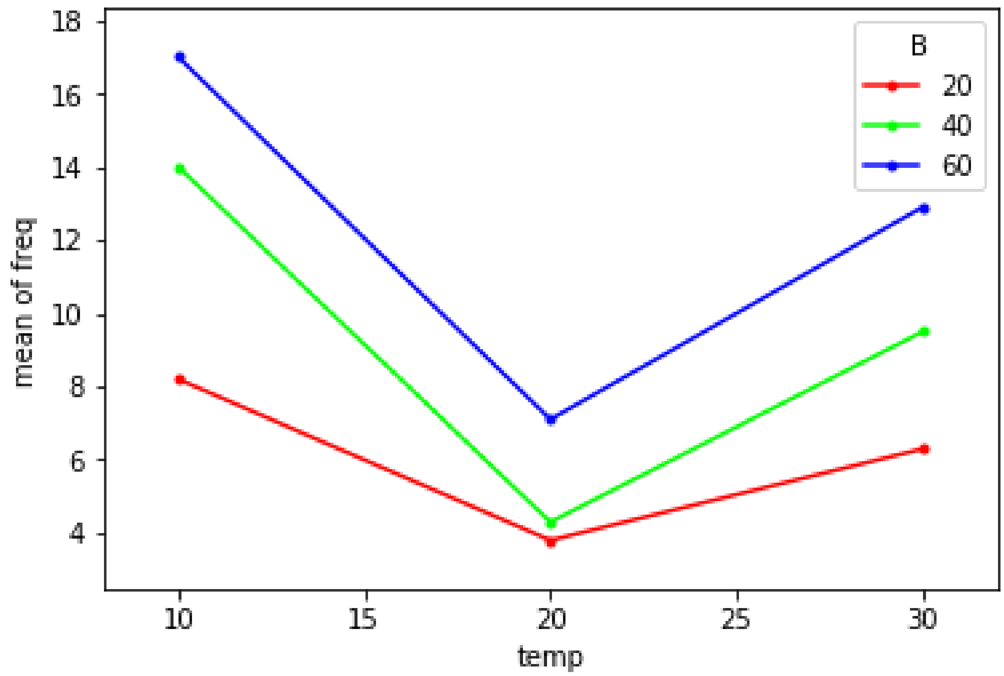
[11]:

	A	B	freq
0	20	20	3.9
1	20	20	3.7
2	30	20	6.5
3	30	20	6.1

```
4  10  20   7.7
```

[12]:
```
#(a)
plt.figure(figsize=(8,3))
interaction_plot(x=data8['A'],trace=data8['B'],response=data8['freq'],xlabel='temp')
plt.show()
```

```
<Figure size 576x216 with 0 Axes>
```



[13]:
```
#(b)
formula='freq ~ C(A) + C(B) + C(A):C(B)'
aov8=smf.ols(formula,data=data8).fit()
sm.stats.anova_lm(aov8)
```

[13]:

	df	sum_sq	mean_sq	F	PR(>F)
C(A)	2.0	193.000000	96.500000	711.885246	1.233947e-10
C(B)	2.0	116.573333	58.286667	429.983607	1.171052e-09
C(A):C(B)	4.0	19.706667	4.926667	36.344262	1.460851e-05
Residual	9.0	1.220000	0.135556	NaN	NaN

[14]:
```
#(c)
sm.stats.anova_lm(aov8)

# H0: 온도에 따라 에너지 소비량에 차이가 없다.
# H1: 온도에 따라 에너지 소비량에 차이가 있다.
```

```
# 검정통계량 F=711.89이고 pvalue=1.23e-10<0.05이므로 유의수준 0.05에서 귀무가설을 기각한다,
# 따라서 유의수준 0.05에서 온도에 따라 에너지 소비량에 차이가 있다고 할 수 있다.
```

[14]:

	df	sum_sq	mean_sq	F	PR(>F)
C(A)	2.0	193.000000	96.500000	711.885246	1.233947e-10
C(B)	2.0	116.573333	58.286667	429.983607	1.171052e-09
C(A):C(B)	4.0	19.706667	4.926667	36.344262	1.460851e-05
Residual	9.0	1.220000	0.135556	NaN	NaN

[15]:

```
#(d)
sm.stats.anova_lm(aov8)

# H0: 점성에 따라 에너지 소비량에 차이가 없다.
# H1: 점성에 따라 에너지 소비량에 차이가 있다.

# 검정통계량 F=429.98이고 pvalue=1.17e-09<0.05이므로 유의수준 0.05에서 귀무가설을 기각한다,
# 따라서 유의수준 0.05에서 점성에 따라 에너지 소비량에 차이가 있다고 할 수 있다.
```

[15]:

	df	sum_sq	mean_sq	F	PR(>F)
C(A)	2.0	193.000000	96.500000	711.885246	1.233947e-10
C(B)	2.0	116.573333	58.286667	429.983607	1.171052e-09
C(A):C(B)	4.0	19.706667	4.926667	36.344262	1.460851e-05
Residual	9.0	1.220000	0.135556	NaN	NaN

[16]:

```
#(e)
sm.stats.anova_lm(aov8)

# H0: 온도와 점성에 따른 상호작용이 없다.
  # H1: 온도와 점성에 따른 상호작용이 있다.

# 검정통계량 F=36.34이고 pvalue=1.46e-05<0.05이므로 유의수준 0.05에서 귀무가설을 기각한다,
# 따라서 유의수준 0.05에서 온도와 점성에 따른 상호작용이 있다고 할 수 있다.
```

[16]:

	df	sum_sq	mean_sq	F	PR(>F)
C(A)	2.0	193.000000	96.500000	711.885246	1.233947e-10
C(B)	2.0	116.573333	58.286667	429.983607	1.171052e-09
C(A):C(B)	4.0	19.706667	4.926667	36.344262	1.460851e-05
Residual	9.0	1.220000	0.135556	NaN	NaN

10번

[21]:

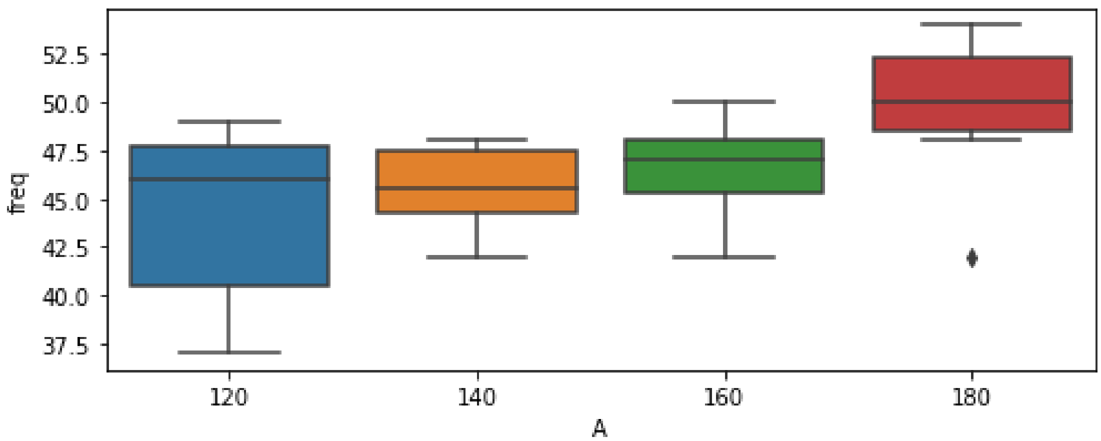
```
A=np.hstack([np.tile(120,6),np.tile(140,6),np.tile(160,6),np.tile(180,6)])
B=[15,15,20,20,25,25,15,15,20,20,25,25,15,15,20,20,25,25,15,15,20,20,25,25]
freq=[39,37,45,49,47,48,42,44,48,46,45,48,42,48,50,46,45,48,48,50,42,54,50,53]

data9=pd.DataFrame({'A':A, 'B':B, 'freq':freq})
```

```
data9.head()
```

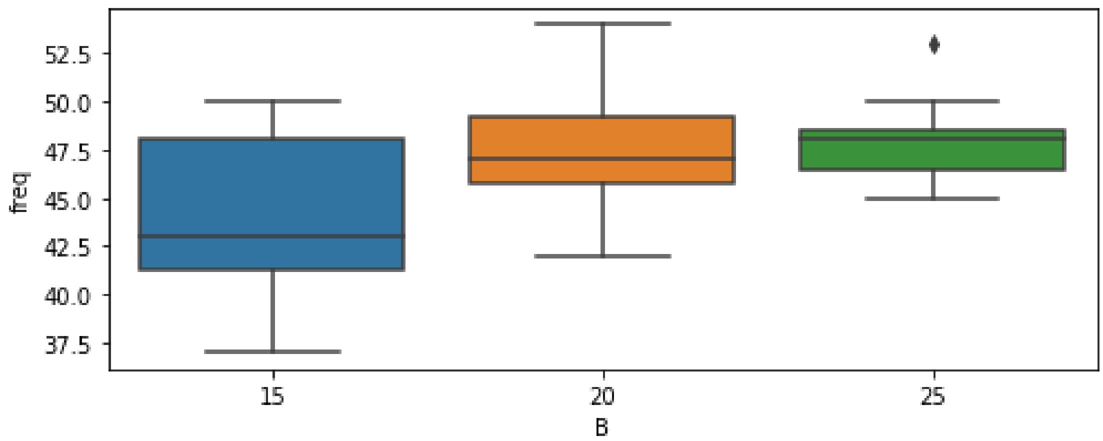
[21]:

	A	B	freq
0	120	15	39
1	120	15	37
2	120	20	45
3	120	20	49
4	120	25	47

[25]:
```
#(a)
plt.figure(figsize=(8,3))
sns.boxplot(x='A',y='freq',data=data9)
plt.show()
```



[26]:
```
#(b)
plt.figure(figsize=(8,3))
sns.boxplot(x='B',y='freq',data=data9)
plt.show()
```



[29]:
```
#(c)
formula='freq ~ C(A) + C(B) + C(A):C(B)'
aov9=smf.ols(formula,data=data9).fit()
sm.stats.anova_lm(aov9)
```

[29]:

	df	sum_sq	mean_sq	F	PR(>F)
C(A)	3.0	92.500000	30.833333	2.890625	0.079354
C(B)	2.0	86.333333	43.166667	4.046875	0.045365
C(A):C(B)	6.0	69.000000	11.500000	1.078125	0.427208
Residual	12.0	128.000000	10.666667	NaN	NaN

[31]:
```
#(d)
sm.stats.anova_lm(aov9)

# H0: 절단속도에 따라 금속절단기 수명에 차이가 없다.
# H1: 절단속도에 따라 금속절단기 수명에 차이가 있다.

# 검정통계량 F=2.89이고 pvalue=0.079>0.05이므로 유의수준 0.05에서 귀무가설을 기각하지 못한다,
# 따라서 유의수준 0.05에서 절단 속도에 따라 금속절단기 수명에 차이가 있다고 할 수 없다.
```

[31]:

	df	sum_sq	mean_sq	F	PR(>F)
C(A)	3.0	92.500000	30.833333	2.890625	0.079354
C(B)	2.0	86.333333	43.166667	4.046875	0.045365
C(A):C(B)	6.0	69.000000	11.500000	1.078125	0.427208
Residual	12.0	128.000000	10.666667	NaN	NaN

[32]:
```
#(e)
sm.stats.anova_lm(aov9)

 # H0: 기계의 각도에 따라 금속절단기 수명에 차이가 없다.
 # H1: 기계의 각도에 따라 금속절단기 수명에 차이가 있다.
```

```
# 검정통계량 F=4.05이고 pvalue=0.045<0.05이므로 유의수준 0.05에서 귀무가설을 기각한다,
# 따라서 유의수준 0.05에서 기계의 각도에 따라 금속절단기 수명에 차이가 있다고 할 수 있다.
```

[32]:

	df	sum_sq	mean_sq	F	PR(>F)
C(A)	3.0	92.500000	30.833333	2.890625	0.079354
C(B)	2.0	86.333333	43.166667	4.046875	0.045365
C(A):C(B)	6.0	69.000000	11.500000	1.078125	0.427208
Residual	12.0	128.000000	10.666667	NaN	NaN

[33]:
```
#(f)
sm.stats.anova_lm(aov9)

# H0: 절단 속도와 기계의 각도에 따른 상호작용이 없다.
# H1: 절단 속도와 기계의 각도에 따른 상호작용이 있다.

# 검정통계량 F=1.08이고 pvalue=0.427>0.05이므로 유의수준 0.05에서 귀무가설을 기각하지 못한다,
# 따라서 유의수준 0.05에서 절단 속도와 기계의 각도에 따른 상호작용이 있다고 할 수 없다.
```

[33]:

	df	sum_sq	mean_sq	F	PR(>F)
C(A)	3.0	92.500000	30.833333	2.890625	0.079354
C(B)	2.0	86.333333	43.166667	4.046875	0.045365
C(A):C(B)	6.0	69.000000	11.500000	1.078125	0.427208
Residual	12.0	128.000000	10.666667	NaN	NaN

11번

[34]:
```
cement=pd.read_csv("C:/PythonbookData/problem10.11_cement.csv")
cement.head()
```

[34]:

	a	b	y
0	1	1	305
1	1	1	302
2	1	2	335
3	1	2	337
4	1	3	366

[37]:
```
formula='y ~ C(a) + C(b) + C(a):C(b)'
aov10=smf.ols(formula,data=cement).fit()
sm.stats.anova_lm(aov10)

 # H0: 석고 종류에 따라 시멘트 강도에 차이가 없다.
 # H1: 석고 종류에 따라 시멘트 강도에 차이가 있다.

# 검정통계량 F=551.06이고 pvalue=7.15e-17<0.05로 유의수준 0.05에서 귀무가설을 기각한다.
# 따라서 유의수준 0.05에서 석고 종류에 따라 시멘트 강도에 차이가 있다고 할 수 있다.
```

```
# H0: 석고 첨가량에 따라 시멘트 강도에 차이가 없다.
# H1: 석고 첨가량에 따라 시멘트 강도에 차이가 있다.

# 검정통계량 F=339.74이고 pvalue=8.76e-18<0.05로 유의수준 0.05에서 귀무가설을 기각한다.
# 따라서 유의수준 0.05에서 석고 첨가량에 따라 시멘트 강도에 차이가 있다고 할 수 있다.

# H0: 석고 종류와 석고 첨가량에 따른 상호작용이 없다.
# H1: 석고 종류와 석고 첨가량에 따른 상호작용이 있다.

# 검정통계량 F=171.98이고 pvalue=9.45e-16<0.05로 유의수준 0.05에서 귀무가설을 기각한다.
# 따라서 유의수준 0.05에서 석고 종류와 석고 첨가량에 따른 교호작용이 있다고 할 수 있다.
```

[37]:

	df	sum_sq	mean_sq	F	PR(>F)
C(a)	2.0	3092.055556	1546.027778	551.059406	7.146209e-17
C(b)	5.0	5607.472222	1121.494444	399.740594	8.760881e-18
C(a):C(b)	10.0	4824.944444	482.494444	171.978218	9.452192e-16
Residual	18.0	50.500000	2.805556	NaN	NaN

11장 연습문제

[1]:
```
import numpy as np
import math as m
import pandas as pd
import statsmodels.formula.api as smf
import statsmodels.api as sm
import scipy as sp
```

1번

[3]:
```
def triangle_area(base,height):
    return((base*height)/2)

print('base: 10, height: 8 => area',triangle_area(10,8))
```

```
base: 10, height: 8 => area 40.0
```

2번

[5]:
```
def circle_area(radius):
    return (np.pi*radius*radius)

print('radius: 5 => area', circle_area(5))
```

```
radius: 5 => area 78.53981633974483
```

3번

[30]:
```
def pearson(x,y):
    xsd=np.std(x,ddof=1)
    ysd=np.std(y,ddof=1)
    cov=np.cov(x,y)[0,1]
    return((cov/(xsd*ysd)))

df=pd.DataFrame({"x":[100,200,300,400],
                 "y":[400,300,100,250]})
corr=df.corr(method='pearson')
```

```
print(corr)
print('pearson:',pearson(df['x'],df['y']))
```

```
          x         y
x  1.000000 -0.671317
y -0.671317  1.000000
pearson: -0.6713171133426188
```

4번

[2]:
```
def my_summary(dist,n):
    array=[]
    if(dist==1):
        array=sp.stats.norm.rvs(loc=0,scale=1,size=n)
    if(dist==2):
        array=sp.stats.expon.rvs(scale=1,size=n)

    print("mean:",np.mean(array))
    print("sd:",np.std(array,ddof=1))
    print("var:",np.var(array,ddof=1))
    print("range:",(np.max(array)-np.min(array)))
    print("max:",np.max(array))
    print("min:",np.min(array))
    print("percentile:",np.percentile(array,[10,20,25,50,75,80,90]))

print("#dist=1")
my_summary(1,50)
print("\n")

print("#dist=2")
my_summary(2,50)
```

```
#dist=1
mean: -0.03200014509670611
sd: 1.0913106248850861
var: 1.190958879987077
range: 5.268787945514457
max: 1.8611779942075795
min: -3.407609951306877
percentile: [-1.22491078 -0.80334456 -0.67140629  0.17076385  0.67811021
0.82816875
  1.1154983 ]


#dist=2
mean: 0.8744985632804719
sd: 0.9506827967075749
var: 0.9037977799557362
```

```
range: 4.602447018625659
max: 4.650221023548259
min: 0.047774004922599644
percentile: [0.09229173 0.15187083 0.18672318 0.58240459 1.27507368 1.34500304
 1.91651456]
```

5번

[65]:
```python
def newton(a):
    x=a
    for i in range(100):
        x1=x-(x**2-np.sin(x))/(2*x-np.cos(x))
        if(np.abs(x1-x)<0.0001):
            break
        x=x1
    return x

newton(4)
```

[65]:
```
0.8767263000340024
```

6번

[72]:
```python
def my_moment(x):
    n=len(x)
    m1=np.mean(x)
    s=np.sum((x-m1)**2)/(n-1)
    g1=(np.sum((x-m1)**3)/n)/(np.sum((x-m1)**2)/n)**(3/2)
    g2=(np.sum((x-m1)**4)/n)/(np.sum((x-m1)**2)/n)**2-3

    return m1,s,g1,g2

#
t=[159, 280, 101, 121, 224, 222, 379, 179, 250, 170]
my_moment(t)
```

[72]:
```
(208.5, 6704.722222222223, 0.7004722929143328, -0.046606318224597993)
```