```
import numpy as np
from scipy.stats import *
import pandas as pd
```

분산 계산

```
x = [1, 2, 3, 4, 5]
print(np.var(x, ddof = 1)) # 분모 = n - 1 ( 5 - 1), 표본
print(np.array(x).var()) # 분모 = n
print(pd.Series(x).var(ddof = 0)) # 분모 = n , 자유도 0(모분산)
```

```
2.5
2.0
2.0
```

표준편차 계산

```
x = [1, 2, 3, 4, 5]
print(np.std(x, ddof = 1))
print(np.array(x).std(ddof = 0))
print(pd.Series(x).std(ddof = 1))
```

```
1.5811388300841898
1.4142135623730951
1.5811388300841898
```

변동계수의 필요성

- 분산과 표준편차 모두 값의 스케일에 크게 영향을 받아 상대적인 산포를 보여주는데 부작합함.
- 변동 계수 = 표준편차 / 평균

```
x1 = np.array([1, 2, 3, 4, 5])
x2 = x1 * 10

print(np.std(x1, ddof = 1))
print(np.std(x2, ddof = 1))
```

```
1.5811388300841898
15.811388300841896
```

```
print(variation(x1)) # 변동 계수
print(variation(x2))
```

```
0.47140452079103173
0.4714045207910317
```

```
print(np.std(x1, ddof = 1) / np.mean(x1))
print(np.std(x2, ddof = 1) / np.mean(x2))
```

```
print(np.std(x2, ddof=1) / np.mean(x2))
```

```
0.5270462766947299
0.5270462766947299
```

스케일링

- 둘 이상의 변수의 값을 상대적으로 비교할 때 사용

```
x1
```

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

```
x2
```

```
array([10, 20, 30, 40, 50])
```

```
# Standard Scaling
z1 = (x1 - x1.mean()) / x1.std()
z2 = (x2 - x2.mean()) / x2.std()

print(z1)
print(z2)
```

```
[-1.41421356 -0.70710678  0.          0.70710678  1.41421356]
[-1.41421356 -0.70710678  0.          0.70710678  1.41421356]
```

```
# Min-max Scaling
z1 = (x1 - x1.min()) / (x1.max() - x1.min())
z2 = (x2 - x2.min()) / (x2.max() - x2.min())

print(z1)
print(z2)
```

```
[0.   0.25 0.5  0.75 1.  ]
[0.   0.25 0.5  0.75 1.  ]
```

```
# sklearn을 이용한 스케일링
X = pd.DataFrame({"X1":[1, 2, 3, 4, 5],
    "X2": [10, 20, 30, 40, 50]})

X
```

```
        X1  X2
```

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler() # 인스턴스화
Z = scaler.fit_transform(X) # fit_transform => ndarray
pd.DataFrame(Z, columns=['X1', 'X2'])
```

	X1	X2
0	0.00	0.00
1	0.25	0.25
2	0.50	0.50
3	0.75	0.75
4	1.00	1.00

범위와 사분위 범위 계산

- 정규분포(np.random.normal)로 부터 무작위 샘플 만들기
- np.random.normal(0,1,1000) # [정규분포] 평균 0, 표준편차 1, 개수 1000개

```
x = np.random.normal(100, 20, size = 1000)
x
```

```
 69.12051027,  89.52676197, 124.85166247,  79.50172772,
 90.10410686,  84.72279952,  94.00238999,  67.74197613,
100.91029351, 144.58023952, 118.34668442, 141.24787059,
 80.75372172, 103.39085227,  96.43614563, 110.80333485,
100.16511177,  67.51592247, 108.35383708,  98.5992919 ,
128.93336842, 116.11236682, 104.80065116,  98.69828206,
115.03038117, 111.9725086 , 116.41361567,  83.3740884 ,
 98.91493186, 131.09172749, 107.35284219, 125.08445267,
103.70660151,  90.21731521, 119.84075833,  85.97570043,
 97.85155299, 101.61674854, 114.26152319,  68.8400246 ,
 93.75085309, 123.43197004, 123.04810499,  84.74282541,
 92.18188159,  52.05665578, 114.19193742,  89.52297494,
 99.83815756,  97.86002283, 137.11250888, 109.96072373,
 77.79579933, 108.13441904, 113.0038473 , 105.27000745,
 84.85527901, 131.89312407, 110.09155188, 114.30334976,
 96.48738123,  84.11751878,  73.09000053,  89.21459244,
102.26646725, 104.80194725,  85.09154069, 129.59876778,
 27.30623172, 117.08147022,  75.58929782,  44.70821672,
161.02003234, 122.86116241, 100.47173781,  82.05475897,
137.96936576,  91.5591604 ,  97.50574418, 109.58099458,
106.01484841, 117.50531958, 125.49005486, 116.90053989,
126.99652924, 112.09395364, 112.311422  , 126.98488346,
 90.74886935, 110.24631512,  76.4341392 ,  55.11232153,
 97.64767522,  69.73001067, 111.3088174 ,  93.78965238,
107.52902319, 122.07270392,  88.89072309,  72.81304459,
 97.11062657,  88.56437269,  87.87150449,  36.4187843 ,
108.88100375, 132.18748428,  68.32067714,  73.60590794,
 64.18404124,  49.79176235, 133.81473913,  90.17649025,
100.53220308,  77.92245128, 122.39123189, 123.69898795,
 98.56022745,  80.34602067, 108.04648553, 103.08615456,
121.42443927, 108.21241993, 102.94616841, 102.06886079,
109.07296413, 115.08234491, 124.47468401,  59.00065336,
 56.95443407, 111.95563469,  89.06153324,  71.19137763,
```

```
 90.33087199,  84.89322244,  77.75175505, 112.49006861,
 74.89204176, 124.951337  , 116.53585733, 123.10123273,
121.31096232, 120.03094915,  82.99208972,  72.66876232,
 55.11821139,  76.80776945,  81.34310711, 127.93327408,
134.79464761, 120.75393651,  66.82342245,  89.72253357,
 90.86745391,  66.72124853, 108.98011266, 102.92479991,
 94.93804526,  83.29724383,  96.09866338, 110.41214422,
100.66629101,  97.3701801 ,  91.5605207 ,  88.98406117,
 99.12228211, 115.31841011, 111.11388513, 140.02066692,
101.66189633,  75.38601192, 126.56335145, 108.04150555,
107.06250968,  96.49224435,  89.21584333, 102.80282439,
106.93843163, 108.75666146,  59.4603225 , 126.00089296,
108.54999398, 106.11533968, 122.38653304, 122.17344376,
 72.50461173,  87.58242629, 114.20359264,  90.7058169 ,
 91.59169571,  76.92208568, 117.50875317, 114.0357505 ,
 96.12980321, 107.07432406,  95.13712379, 130.68853393,
 99.29994201,  87.54781428,  96.99238955, 101.87524233,
 84.36127348,  97.02929292,  90.7494916 , 102.51170629,
105.68038318, 113.38340743,  90.7566578 , 128.6716967 ,
113.32925912,  52.85414168, 112.72333736, 113.22815292,
 92.55674795, 111.69561368,  70.87675321, 102.6211937 ,
 87.77556229, 104.60347675, 101.38100896,  93.60473704,
 86.79848533, 121.25972446,  81.80284206,  80.55158002,
122.77193416,  92.8496363 , 117.83045296,  91.66959286,
 84.35961598, 107.67865374, 110.11353737,  61.14222388,
 81.89788924, 138.12772891, 106.94588573, 123.23249727,
 88.64498665, 140.01264509,  84.93437422, 120.04252499
```

```
print(np.ptp(x))
print(np.max(x) - np.min(x))
```

```
125.58707261610778
125.58707261610778
```

```
print(np.quantile(x, 0.75) - np.quantile(x, 0.25))
print(iqr(x))
```

```
25.991697468850703
25.991697468850703
```