1.5 자주 사용하는 통계량 : 기대값, 분산, 공분산, 상관계수

금융계산에서 자주 사용하는 통계가 기대값(평균), 변동성, 상관계수, 공분산, 정규분포이다. 이번 장에서는 이런 기본적인 통계량을 알아보도록 한다.

1.5.1 평균과 기대값

평균은 어느 데이터를 대표하는 가상의 숫자이다. 가상이라고 하는 이유는 평균값이 실제 데이터의 값이 아니라 이를 모아서 계산한 값이기 때문이다. 보통 평균이라고 하면 산술평균을 가리키고 모든 자료값을 더한 합계에 자료의 갯수로 나눈 값이다.

$$\text{average} = \frac{1}{n}\sum x_i$$

기대값은 확률이 더해진 평균이다. 데이터 원소와 이에 대한 확률값을 곱하여 더한 것이다.

$$\mathrm{E}(x) = \sum p_i x_i.$$

정리하면 평균은 확률을 특별히 고려하지 않은 것이고, 기대값은 확률적인 사건에 대해 이 사건이 일어날 것에 대해 기대되는 값이다. 주사위 값의 평균과 주사위를 한 번 던졌을 때 기대되는 값을 구해보자. 주사위에는 1~6 까지 수가 있고, 각 수가 나올 확률 1/6 로 동일하다.

평균은 산술평균이며 1 부터 6 까지의 합계를 숫자의 개수인 6 으로 나눈다.

평균 = (1+2+3+4+5+6)/6
= 21/6
= 3.5

기대값은 모든 값에 대해 나올 확률을 곱하면서 더하면 된다. 주사위를 던졌을 때 나올 수 있는 값은 1, 2, 3, 4, 5, 6 이다. 이렇게 6 가지의 수에 각각에 대한 확률은 1/6 을 곱하고 더하면 기대값이 된다.

기대값 = 1 x 1/6 + 2 x 1/6 + 3 x 1/6 + 4 x 1/6 + 5 x 1/6 + 6 x 1/6

= 1/6 + 2/6 + 3/6 + 4/6 + 5/6 + 6/6
= 21/6
= 3.5

모든 값이 동일한 확률을 가진 경우 일반적으로 '평균'과 '기대값'은 같은 값을 가진다. 정리하자면 '평균'은 확률적인 개념이 없을 때 쓰이고, '기대값'은 각 사건들이 일어날 확률이 있는 경우 그 기대되는 값을 표현할 때 쓰인다. 따라서 확률 개념이 들어가는 통계에서는 대부분 '평균'이라는 표현이 아닌 '기대값'이라는 표현으로 사용된다.

다음은 산술평균을 구하는 예이다.

<코드>

```
nums = [1, 2, 3, 4, 5, 6]                # nums 리스트에 값을 저장
print( sum(nums) / len(nums) )           # sum 함수로 합계를, len 함수로 데이터의 개수를 구한다
```

</코드>

파이썬의 표준함수말고 numpy 를 사용하여 다음과 같이 계산할 수도 있다.

<코드>

```
import numpy as np            # numpy 라이브러리를 임포트한다
a = np.array( [1,2,3,4,5,6] )    # numpy 의 array()함수를 사용하여 리스트를 array 개체로 저장한다
print( a.mean() )                        # 개체의 mean()함수를 사용하여 평균을 계산한다
```

</코드>

기대값을 계산하려면 사건과 사건이 일어날 확률을 가진 리스트가 2 개가 필요하다. 그리고 리스트를 반복하는 동안 두 개의 리스트에서 사건과 확률을 하나씩 꺼내 곱하고 더하면 된다.

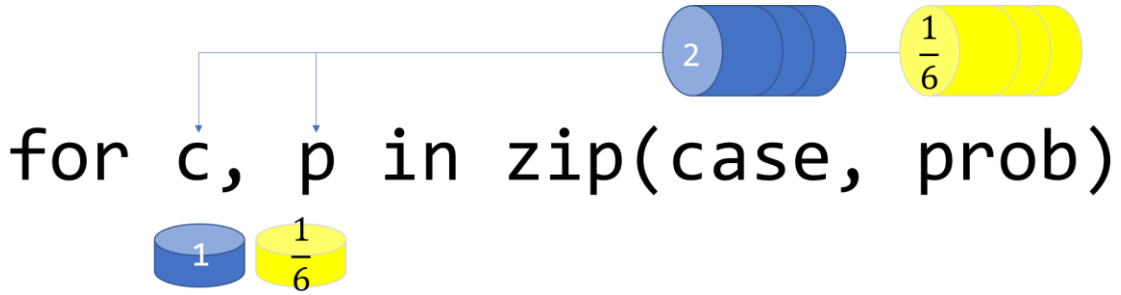
<코드>

```
# 사건과 확률을 case 와 prob 리스트에 저장한다
01: case = [ 1, 2, 3, 4, 5, 6 ]
```

02: prob = [1/6, 1/6, 1/6, 1/6, 1/6, 1/6]

사건과 확률 리스트를 zip 함수로 묶어 for 루프로 반복한다. 반복하는 동안 두 개의 리스트에서 값을 받아 변수 c 와 p 에 저장하고 곱한 결과를 ex 변수에 저장한다
03: ex = 0.0
04: for c, p in zip(case, prob):
05:     ex = ex + c*p
06: print(ex)
위의 3,4,5 번 행은 다음과 같은 인라인 for 루프로 대체가능하다.
07: ex = sum(c*p for c, p in zip(case, prob))
결과를 출력한다
08: print(ex)
</코드>

그림 25 zip 함수를 묶어 for 루프 실행



(그림 zip 함수는 두개의 리스크를 묶어 하나로 만든다. 그래서 루프에 사용하기 쉽다)

zip()함수는 여러 개의 리스트를 묶어 하나의 리스트처럼 사용하게 만들어 주는 함수이다. case 와 prob 리스트 두 개를 하나로 묶어 반복하면서 c 와 p 를 곱하여 더하면 기대값을 얻게 된다.

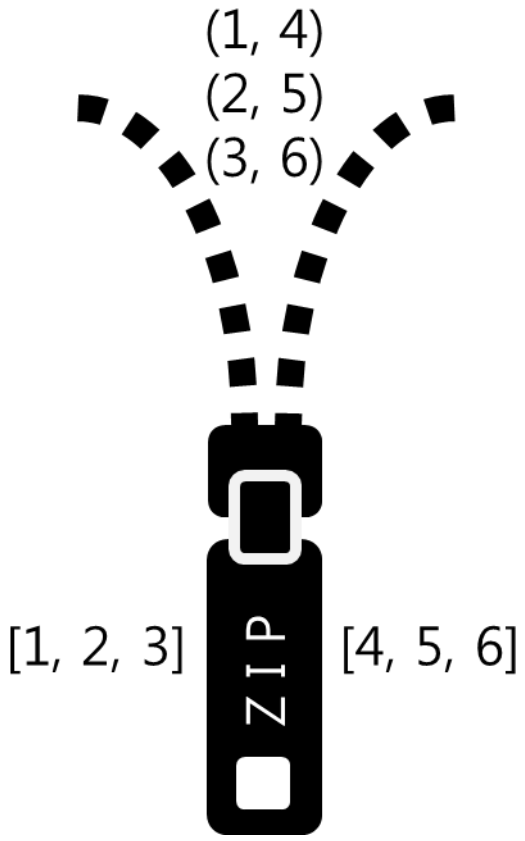
다음은 간단한 zip 함수 사용 예이다. 리스트 a 와 b 를 묶어 하나의 목록으로 만들어 출력을 하는 예이다.

<코드>

```
a = [1, 2, 3]
b = [4, 5, 6]
for ab in zip(a,b):
   print(ab)
```

</코드>

그림 26



1.5.2 이동평균

이동평균은 자주 사용하는 평균이다. 기술적 분석에서도 5 일 이동평균, 20 일 이동평균, MACD 등 여러 기술적 지표에서 자주 사용한다. 이동평균은 계산에 들어가는 값중에서 제일 오래된 값을 버리고 새로운 값을 추가하여 새로운 평균을 구한다. 이동평균은 정지된 값이 아니라 새로운 데이터를 받아 그 대표값을 업데이트한다.

그림 27 삼성전자 주식차트 – 막대봉과 이동평균

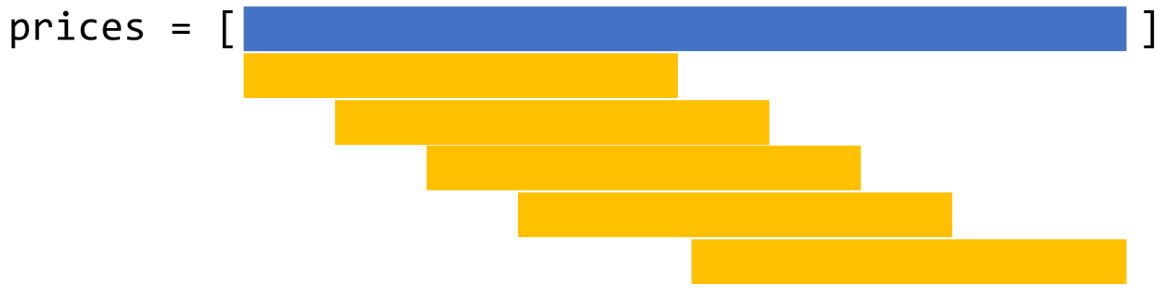
다음은 삼성전자의 2 주일 간의 시세표이다. 일자별 종가를 가지고 5 일 이동평균을 구하면 다음과 같다.

5 일 이동평균이므로 일별시세를 5 개 모아 산술평균을 만든다.

날짜	종가	이동평균
2019.06.10	44,800	
2019.06.11	44,850	
2019.06.12	44,600	
2019.06.13	43,750	
2019.06.14	44,000	44,400=(44800+44850+44600+43750+44000)/5
2019.06.17	43,900	44,220=(44850+44600+43750+44000+43900)/5
2019.06.18	44,350	44,120=(44600+43750+44000+43900+44350)/5
2019.06.19	45,350	44,270=(43750+44000+43900+44350+45350)/5
2019.06.20	45,500	44,620=(44000+43900+44350+45350+45500)/5
2019.06.21	45,700	44,960=(43900+44350+45350+45500+45700)/5

다음은 위의 삼성전자 2 주간의 종가를 가지고 이동평균을 계산하는 예이다. 아래의 소스코드에서는 전체 주가 리스트에서 5 개씩 잘라 이동평균을 계산할 것이다. 하나의 이동평균을 구하면 다시 5 개 리스트를 채워 이동평균을 구한다. 다음 그림과 같이 계산에 필요한 리스트를 만들게 된다.

그림 28prices 를 하나씩 움직여 평균을 계산



Prices 리스트에서 반복하는 동안 end_index 와 begin_index 를 이용하여 5 개의 항목을 가져와서 합계를 계산하고 평균을 낸다.

<코드>

```
01: # 이동평균
02: # 주가를 prices 리스트에 저장한다
03: prices =[ 44800, 44850, 44600, 43750, 44000, 43900, 44350, 45350, 45500,
45700 ]
04: n = 5  # 5 일 이동평균
05:
06: for p in prices[n:]: # prices 의 n 번째 항목부터 마지막 항목까지 반복한다
07:     end_index = prices.index(p) # 항목 p 의 index 를 마지막 인덱스로 정한다
08:     begin_index = end_index – n # 마지막 인덱스에서 n 만큼 앞에 있는
시작인덱스를 정한다
09:     print(begin_index, end_index) # end_index 와 begin_index 를 계산하여
가져올 위치를 확인

10: # 계산한 end_index 와 begin_index 를 가지고 prices 리스트에서 5 개의
항목을 확인
11: for p in prices[n:]:
12:     end_index = prices.index(p)
```

```
13:     begin_index = end_index - n
14:     print(prices[begin_index:end_index])

15: # 5 개씩 가져와서 sum()함수로 합계를 구하고 n 으로 나눠 이동평균을 계산
16: for p in prices[n:]:
17:     end_index = prices.index(p)
18:     begin_index = end_index - n
19:     print(sum(prices[begin_index:end_index])/n)
</코드>
```

1.5.3 가중(산술)평균

가중(산술)평균은 자료의 중요도나 영향 정도에 해당하는 가중치를 고려하여 구한 평균값이다. 주식의 평균매입단가도 가중평균이다. 지난 달에 A 종목 100 주를 주당 5,000 원에 매입하고, 다시 이번 달에 같은 종목 50 주를 주당 7,500 에 추가 매수하였다면 보유수량은 총 150 주이고 평균 매수단가는 6,250 원(5,000 원과 7,500 원의 단순산술평균)이 아니라 5000 원 매입단가의 비중과 7500 원 매입단가의 비중을 반영하여 가중평균을 계산하는 것이 타당하다. 그래서 주당 평균매수단가는 (100*5000+50*7500)/(100+50) = 5,833.33 원이다.

가중치는 시험에서도 볼 수 있는 데, 가령 모두 50 개의 문항이지만 쉬운 문제는 1 점짜리이고, 좀 더 어려운 문제는 3 점, 아주 어려운 문제는 5 점 등으로 각 문제에 대한 배점을 달리할 수 있다. 50 개 문항중 각각 하나의 문제이지만 그 가중치는 다른 것이다.

재무분야의 WACC(weighted average cost of capital) 역시 가중평균이다. WACC 는 회사의 자본구조에서 부채와 지분의 시장가치를 바탕으로 한 가중평균으로 현금흐름을 할인하는 데 사용한다.

다음은 한 학기 수업의 결과인데, 이 결과를 가지고 가중평균을 계산해보자.

평가	점수	비중	가중치	가중점수

퀴즈	82	20%	0.2	16.4
중간시험	90	35%	0.35	31.5
기말시험	76	45%	0.45	34.2

가중점수의 합은 = 82x20% + 90x35% + 76x45% = 16.4+31.5+34.2 = 82.1 이다. 가중점수의 합은 가중치의 합으로 나눠 가중평균을 나누는데, 가중치의 합은 1 이므로 가중평균은 82.1 / 1= 82.1 이 된다.

<코드>

```
# 평가점수와 평가비중을 scores 와 weight 리스트에 저장
01: scores = [ 82, 90, 76 ]
02: weight = [ 0.2, 0.35, 0.45 ]

# scores 와 weight 리스트를 zip 함수로 묶어 for 루프로 반복한다. 반복하는 동안 변수 s 와 w 에 저장하고 곱셈의 결과를 합한다
03: wgt_avg = 0.0 # 합계를 저장할 변수
04: for s, w in zip(scores, weight):
05:     wgt_avg = wgt_avg + s*w
# 결과를 출력한다
06: print (wgt_avg)

# 위의 for 루프는 인라인 for 루프로 바꾸어 간략하게 표현할 수 있다. 3, 4, 5 번 행을 하나의 행으로 줄일 수 있다.
07: wgt_avg = sum( s*w for s, w in zip(scores, weight) )

# 결과를 출력한다
08: print (wgt_avg)
```

</코드>

1.5.4 분산과 표준편차

금융에서 분산과 표준편차는 리스크를 가리키는 척도이다. 분산과 표준편차는 데이터의 흩어진 정도를 가리킨다. 평균과 기대값이 데이터의 중심을 가리킨다고

하면 분산과 표준편차는 데이터들이 데이터의 중심에서 얼마나 흩어져 있는 지를 설명하는 척도이다. 그래서 각 변수 값에서 평균을 빼서 제곱하여 합산한 것(이를 '편차 제곱합' 이라고 한다)에 대해서 평균을 내면 된다.

$$V = \frac{\sum |x - \bar{x}|^2}{n - 1}$$

예를 들어 1, 2, 3, 4, 5 다섯 개의 숫자를 있다면

평균 = (1+2+3+4+5) / 5 = 3 이고,

분산 = 편차제곱합 / 데이터의 개수-1

= ($(1-3)^2$ + $(2-3)^2$ + $(3-3)^2$ + $(4-3)^2$+ $(5-3)^2$) / (5-1) = 2.5 이다.

분산계산에서 제곱을 하는 이유는 각 데이터에서 평균을 빼다 보면 (+)와 (-) 부호 때문에 편차 합산이 0 이 되는 난감한 상황이 생기기 때문이다. 그래서 모두 같은 부호를 가지도록 제곱을 하는 것이다. 다만 제곱한 탓에 단위가 흐트러진다는 단점이 있다. 위의 분산 계산에서 제곱을 하지 않으면 (1-3)과 (5-3), (2-3)과 (4-3)을 합치면 0 이 된다.

<코드>

```
# 분산을 계산할 리스트 nums 를 준비한다
nums = [ 1, 2, 3, 4, 5 ]

# 리스트의 평균을 계산한다
avg = sum(nums) / len(nums)

# 리스트를 반복하여 편차제곱합을 계산한다. nums 리스트를 반복하면서 n 에 저장하고 여기서 평균(avg)을 뺀 결과를 제곱(**2)하여 합계(sumsquare = sumsquare +...)를 구한다
sumsquare = 0.0
for n in nums:
    sumsquare = sumsquare + (n - avg)**2
var = sumsquare / (len(nums)-1)
print(var)
```

```
# 위의 for 루프(3, 4, 5 번 )는 다음과 같이 인라인 for 루프로 바꿀 수 있다.
08: sumsquare = sum( (n-avg)**2 for n in nums )

# 편차제곱합에서 데이터 개수로 나누고 결과를 출력한다
09: var = sumsquare / (len(nums)-1)
10: print(var)

# 표준편차를 구하기 위해 sqrt() 함수가 필요한데, 이를 위해 math 모듈을
임포트한다
11: import math
12: stdev = math.sqrt(var)
13: print(stdev)
</코드>
```

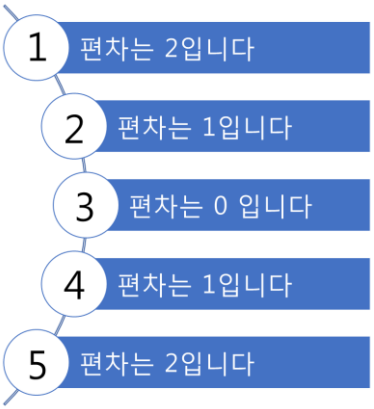
확률이 있을 때 분산은 각 값에서 기대값을 빼서 제곱을 한 후, 그 값이 나올 확률들을 곱하면서 전부 더한 값이다.

표준편차(standard deviation)는 분산에 제곱근을 취한 것이다. 분산계산에서 해준 제곱을 되돌리는 것이다.

$$\mathrm{SD} = \sqrt{\frac{\sum |x - \bar{x}|^2}{n-1}}$$

'표준'이라는 말의 의미를 생각해봐야 한다. 편차(deviation)는 평균과의 차이인데, 편차를 구해보면 크기가 다른 편차가 여러 개 존재한다. 그런데 '편차가 얼마냐?'라는 질문을 받는다면 모든 편차를 다 말해주어야 한다. 그래서 대표값(편차의 기대값)으로 편차중 대표가 되는 표준을 정한다.

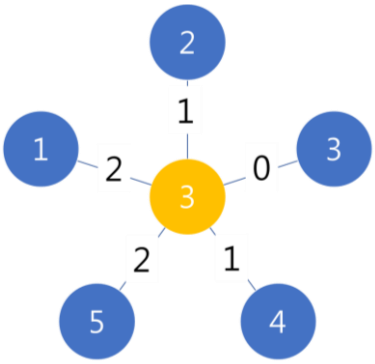
그림 29 각자의 입장에서 편차는 다르다



다음과 같은 5 개의 값을 가지고 표준편차를 계산해보자.

1, 2, 3, 4, 5

그림 30 표준편차의 의미



표준편차를 구하기 앞서 위의 그림의 5 개의 편차중 대표편차를 계산한 후 표준편차를 구해보자. 중앙에 평균인 3 이 있고 그 주변에는 1,2,3,4,5 가 있다. 그리고 중앙까지의 거리는 편차인데, 각각 2,1,0,1,2 이다. 그래서 5 개의 편차중 대표를 구하기 위해 편차들의 평균을 계산하면 (2+1+0+1+2) / 5 = 1.2 이다. 이제 우리는 하나의 편차로 1.2 를 생각해두면 될 것이다.

이번에는 정식으로 평균, 분산, 표준편차를 구하면 그 과정은 다음과 같다.

평균 = (1+2+3+4+5) / 5 = 3

분산 = ((1-3)2 + (2-3)2 + (3-3)2 + (4-3)2+ (5-3)2) / 5 = 2

분산에 제곱근을 취하여 표준편차를 계산하면

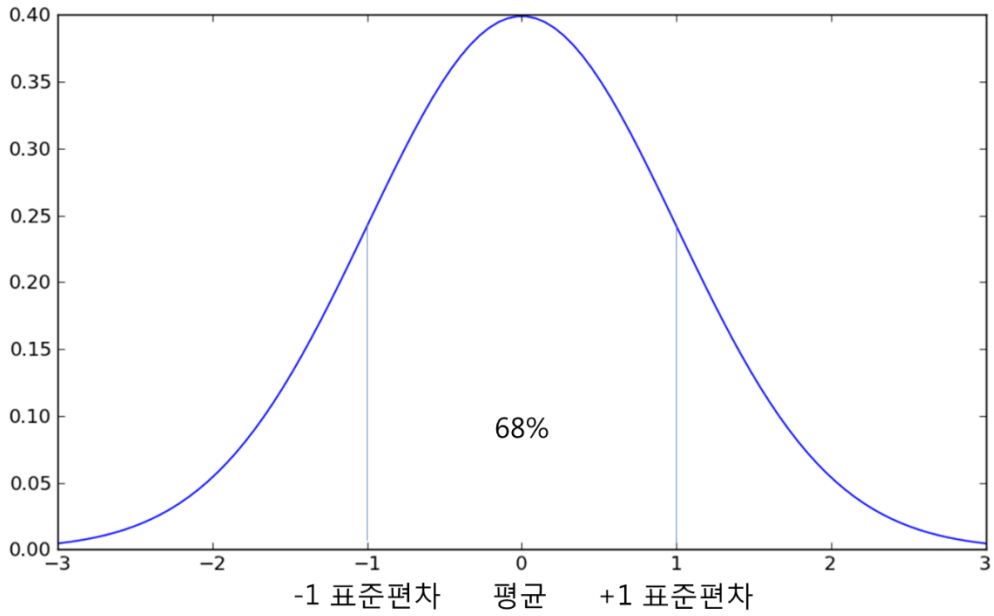
표준편차 = √분산 = √2 = 1.414...

이제 위의 데이터는 평균이 3 이고 각 값들은 평균적으로 약 1.4 정도 떨어져 있다고 생각하면 된다. 즉 평균과 표준편차 두 개로 다섯 개의 숫자로 이루어진 데이터의 특성을 압축하는 셈이다. 이제 데이터가 5 개, 5000 개, 500,000 개라도 두 개의 숫자만 계산하면 그 데이터를 손에 쥐는 셈이다.

1.5.5 정규분포에서 표준편차와 평균

표준편차와 평균과 밀접한 관계를 가진 것이 정규분포인데, 경험상 종모양의 정규분포에서 평균을 중심으로 평균±1 표준편차내 범위는 정규분포 면적의 대략 68%(면적이 확률이므로 68%의 확률)를 차지한다.

그림 31 표준편차 1 단위는 정규분포면적(확률)의 약 68%를 차지한다



1. 모든 관측치의 약 68%는 평균으로부터 1 표준편차 이내에 속한다.
2. 모든 관측치의 약 95%는 평균으로부터 2 표준편차 이내에 속한다.
3. 모든 관측치의 약 99.7%는 평균으로부터 3 표준편차 이내에 속한다.

정규분포는

<참고>matplotlib

Matplotlib 는 파이썬에서 자료를 차트(chart)나 플롯(plot)으로 시각화(visulaization)하는 패키지이다. Matplotlib 는 라인 플롯, 바 차트, 파이차트, 히스토그램, Box Plot, Scatter Plot 등을 비롯하여 다양한 차트와 플롯 스타일을 제공한다.

Matplotlib 갤러리 웹사이트(https://matplotlib.org/gallery/index.html)에는 Matplotlib 를 사용한 여러 시각화 예제들을 볼 수 있다.

그림 32 Matplotlib 갤러리

</참고>

다음은 matplotlib 를 이용하여 -10~10 사이에서 평균이 0, 표준편차가 2 인 정규분포를 그린 것이다.

<코드>

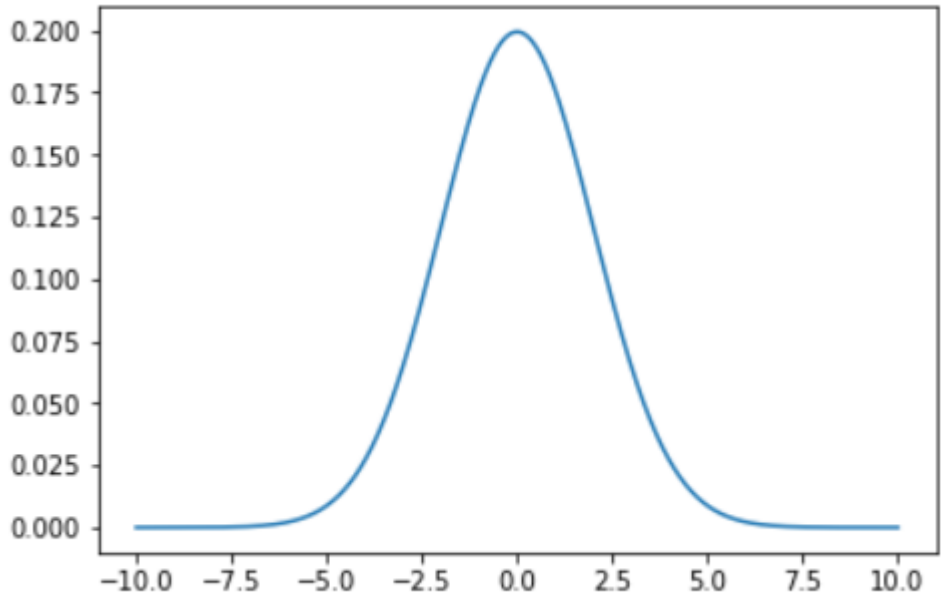
```
# numpy, matplotlib,scipy 라이브러리를 임포트한다
01: import numpy as np
01: import matplotlib.pyplot as plt
01: from scipy.stats import norm
```

```
# x 축은 -10 ~ 10 사이에서 0.001 간격으로 정한다
02: x_axis = np.arange(-10, 10, 0.001)

# 평균= 0, 표준편차 = 2.0 인 정규분포를 만든다
02: plt.plot(x_axis, norm.pdf(x_axis,0,2))

# 정규분포를 화면에 출력한다
03: plt.show()
</코드>
```

그림 33 평균이 0, 표준편차이 2 인 정규분포



(그림 코드 실행결과)

1.5.6 자유도

분산(variance) 또는 표준편차(standard deviation)를 계산할 때는 데이터가 모집단(population)인지 또는 표본(sample)인지에 따라 계산이 약간 달라진다. 분산과 표준편차의 식-사실상 하나이지만-을 보면 분자와 분모로 나누어진 나눗셈이다.

분산 = 편차 제곱합 / 데이터의 개수(N)
편차 제곱합 = Σ(데이터 – 평균)^2

편차 제곱합인 분자부분의 식을 그대로인데, 데이터의 갯수인 분모는 데이터가 모집단인지 표본집단인지에 따라 달라진다. 즉 데이터의 개수가 N 이라고 할 때 모집단이면 분모는 N 이고 표본집단인 경우 N-1 이다.

표본분산 = 편차 제곱합 / 표본데이터의 개수(N-1)
편차 제곱합 = Σ(표본데이터 – 표본평균)^2

정확한 통계량을 구하기 위한 가장 좋은 방법은 모든 대상을 상대로 데이터를 구하는 것이다. 그런 모든 대상을 모집단이라고 한다. 그러나 각종 설문조사에서 모집단을 상대로 데이터를 구하려면 현실적인 문제가 있다. 비용문제도 있지만 아예 불가능한 경우도 있기 때문이다. 그래서 모집단의 일부만을 대상으로 데이터를 구한다. 모집단의 일부를 표본집단이라고 한다.

그런데 전체가 아닌 일부이기 때문에 정확한 통계량이 정확하진 않다.
표본집단의 분산도 마찬가지이다. 특히 표본집단의 분산계산에서 사용한 평균은 모집단의 평균이 아니다. 이러한 오차를 일부 보정하기 위해 N-1 로 나누어 준다. 수학적으로 말하면 데이터가 표본집단일때 N-1 로 편차제곱합을 나누어야 모집단 분산에 대한 불편추정량(unbiased estimator)이 되기 때문이다.

N-1 을 자유도(degree of freedom)라고 한다. 자유도는 자유롭게 움직이는 정도라는 의미인데 가령 합계가 알려져 있을 때 5 개의 숫자중 4 개를 자유롭게 고르고 나면 나머지 하나는 자유롭게 고르지 못하고 합계와 일치하기 위해 고정된다. 그래서 자유도가 4 이다.

그림 34 수도쿠에서 자유롭게 숫자를 채우다 보면 어쩔 수 없이 맘대로 채울 수 없는 칸이 생긴다

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

아래와 같은 표가 있다. 맨 오른쪽 열은 왼쪽 열들의 합이고, 매 아래쪽 행 역시 위의 두 개의 행의 합이다. 그리고 맨 오른쪽 하단의 셀은 전체의 합이다.

만일 a=2 라고 정하면 b, c, d 는 자동으로 정해진다.
B 는 a=2 이고 같은 행의 합계가 6 이므로 선택의 여지없이 4 가 된다.
C 는 a=2 이고 그 아래의 합계가 5 이므로 자동으로 3 이 된다.
D 의 경우 c=3 이고 같은 행의 합계가 4 이므로 어쩔 수 없이 1 이 된다.
그래서 a=2, b=4, c=3, d=1 이 되어 전체 합계 10 과 일치한다.

만일 a 말고 b, c, d 중 하나를 먼저 선택해도 마찬가지이다. 즉 자유롭게 선택할 수 있는 것은 a, b, c, d 중 하나이다. 그래서 자유도는 1 이 된다.

그림 35 채울 수 있는 자유가 제한적이다. 임의로 값을 선택가능한 칸은 하나이다

a	b	6
c	d	4
5	5	10

1.5.7 공분산과 상관계수

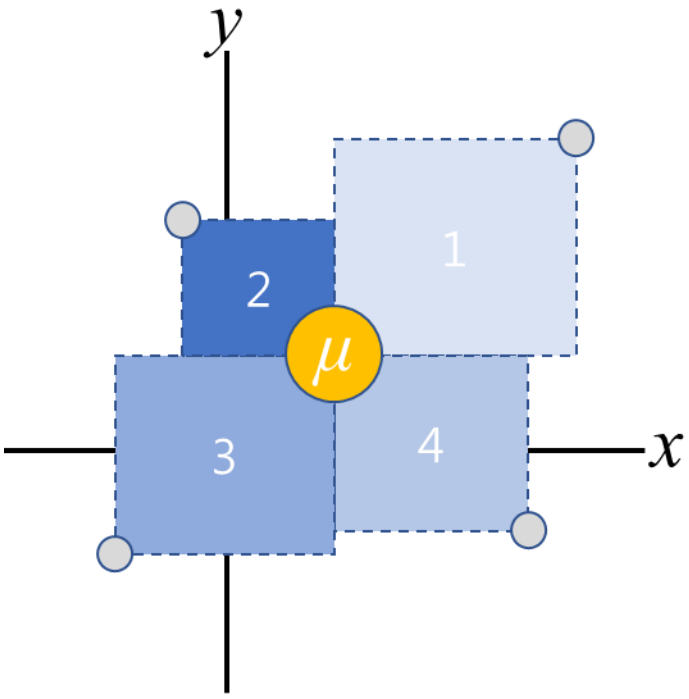
두 변수의 상관관계를 나타내는 척도에는 공분산과 상관계수가 있다. 공분산의 수식(표본의 공분산)은 다음과 같다.

$$\sigma_{xy} = \frac{\sum(x-\mu_x)(y-\mu_y)}{n-1}$$

여기서 x 와 y 는 각각의 변수를 의미하고 μ는 변수의 평균이다. 그리고 n 은 데이터의 개수를 의미한다. 수식을 분석하면 분모는 데이터의 개수로 나눈 것이다. 그리고 분자에 해당하는 부분 $\sum(x-\mu_x)(y-\mu_y)$을 보면 가로와 세로를 곱한 것(사각형의 면적)을 합쳐 놓은 듯한 모습이다.

합계를 개수(n-1)로 나누었다는 점에서 위의 공분산 수식은 평균의 일종임을 알 수 있다. 그래서 종합하면 다음의 그림과 같이 평균 μ을 영점으로 하는 1, 2, 3, 4 로 번호를 매겨둔 네 개 사각형의 평균면적이다.

그림 36 공분산의 1,2,3,4 평균을 중심으로 모인 네개의 사각형의 평균면적으로 생각할 수 있다



사각형 1 의 경우 ○에서 x 와 y 값이 평균 μ보다 크므로 $(x-\mu_x)(y-\mu_y)>0$이고 사각형 2 의 경우 ○에서 x 는 평균보다 작고 y 는 평균보다 크므로 $(x-\mu_x)(y-\mu_y)<0$이다. 사각형 3 의 경우 ○에서 x 는 평균보다 작고 y 도 평균보다 작으므로 $(x-\mu_x)(y-\mu_y)>0$이다. 마지막으로 사각형 4 의 경우 ○에서 x 는 평균보다 크고 y 는 평균보다 작으므로 $(x-\mu_x)(y-\mu_y)<0$이다.

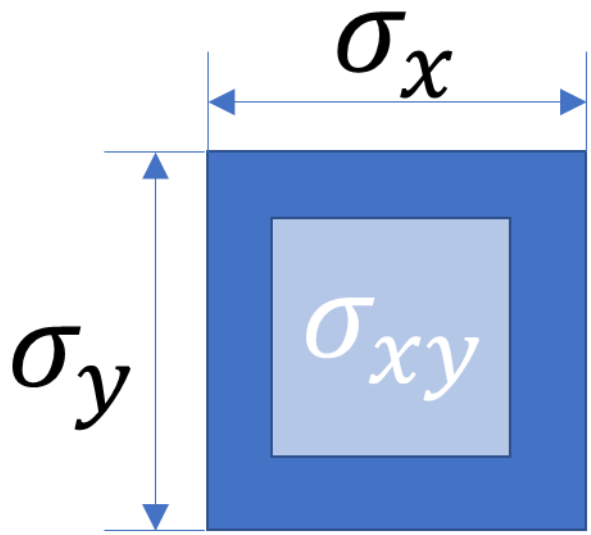
공분산은 부호가 중요할 뿐이다. 공분산이 (+)이면 X 와 Y 는 양의 상관관계이다. X 가 증가하면 Y 도 증가한다는 의미이다. 반대로 공분산이 (-)이면 X 와 Y 는 음의 상관관계이다. X 가 증가하면 Y 도 감소한다는 의미이다. 공분산은 X 와 Y 변수간의 방향성을 알려주는 것일 뿐 상관관계의 정도를 알려주진 못한다.

공분산은 상관관계의 정도를 구체적으로 표현하지 못하는 단점이 있는데, 상관관계를 표준화한 값이 상관계수이다. 표준화되었기 때문에 상관계수는 (-1)~(+1) 사이이므로 상관관계 비교가 가능하다. 상관계수는 ρ(rho)라고 표시하는데, 공식은 다음과 같다.

$$\rho_{xy} \equiv \frac{\sigma_{xy}}{\sigma_x \sigma_y}$$

상관계수식은 분수의 형태인데, 즉 공분산과 두 변수의 표준편차곱 간의 비율이다. 두 변수 표준편차를 곱하는 것이므로 사각형의 면적위에 공분산 사각형을 올려두어 비율로 나타낸 것으로 이해할 수 있다.

그림 37 상관계수는 공분산과 두 변수의 표준편차곱 간의 비율



상관계수는 두 변수간의 관계를 의미한다. 그러나 두 변수간의 연관된 정도를 나타낼 뿐 인과(원인과 결과)관계를 설명하는 것은 아니다. 두 변수간 인과관계는 회귀분석을 통해 확인해 볼 수 있다. 보편적으로 많이 사용하는 상관계수는 피어슨 상관계수인데, X 와 Y 의 피어슨 상관계수 수식은 다음과 같다.

$$r = \frac{\sum XY - \frac{\sum X \sum Y}{n}}{\sqrt{\left[\sum X^2 - \frac{(\sum X)^2}{n}\right]\left[\sum Y^2 - \frac{(\sum Y)^2}{n}\right]}}$$

X 와 Y 가 완전히 동일하면 상관계수 r 값은 +1, 전혀 다르면 0, 반대방향으로 완전히 동일 하면 –1 을 가진다. 결정계수 (coefficient of determination) 는 r^2 인데 이것은 X 로부터 Y 를 예측할 수 있는 정도를 의미한다.

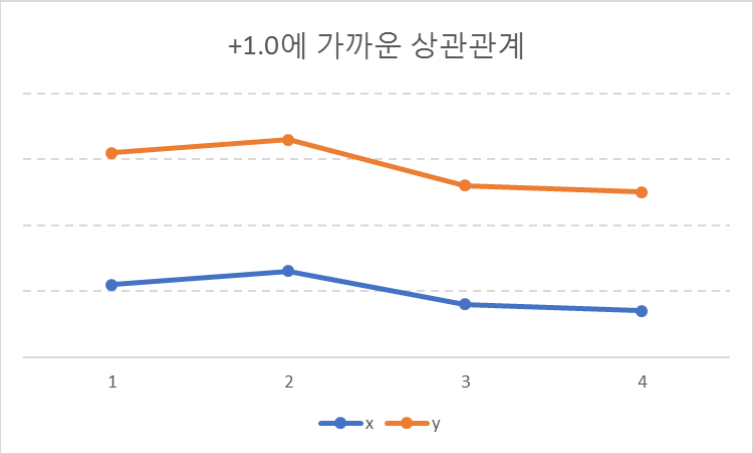
가령 다음과 같이 두 개의 변수가 있다.

X	y
41	61
43	63
38	56
37	55

숫자만 봐서는 두 변수의 관계를 금방 알 수 없다. 그러나 두 변수를 차트로 그리면 두 개의 변수 움직임이 비슷하다. 오를 때는 같이 오르고, 내릴 때도 같이 내린다. 또 상승폭과 하락폭도 비슷하다.

상관계수는 이러한 변수들의 관계를 숫자로 표현하는 방법이다. 상관계수는 -1.0 ~ +1.0 사이의 값으로 완벽하게 두 변수가 같이 움직이면 +1.0, 정반대로 움직이면 -1.0 이다. 아래와 같이 두 개의 변수가 같이 움직이는 경우 상관계수는 +0.99507659 인데, 완전 양의 상관를 나타대는 계수 1.0 과 가까운 모습을 보이고 있다.

그림 38+1 의 상관관계

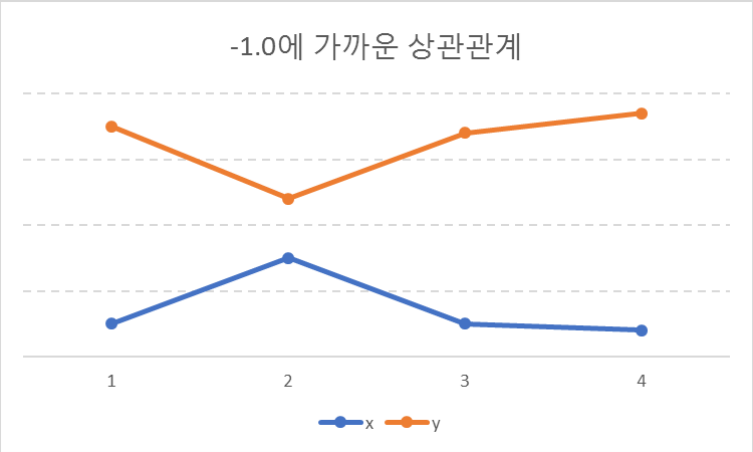


이번에는 두 개의 변수가 다음과 같다.

X	y
35	65
45	54
35	64
34	67

위의 변수들을 차트로 다시 그리면 다음과 같다. 변수 x 가 상승하면 변수 y 는 하락하고, 다시 x 가 하락하면 y 는 상승한다. 즉 두 변수는 서로 반대방향으로 움직이고 있다. 두 변수의 상관계수는 -0.99103977 인데, 음의 상관관계 -1.0 에 가깝다.

그림 39-1 의 상관관계

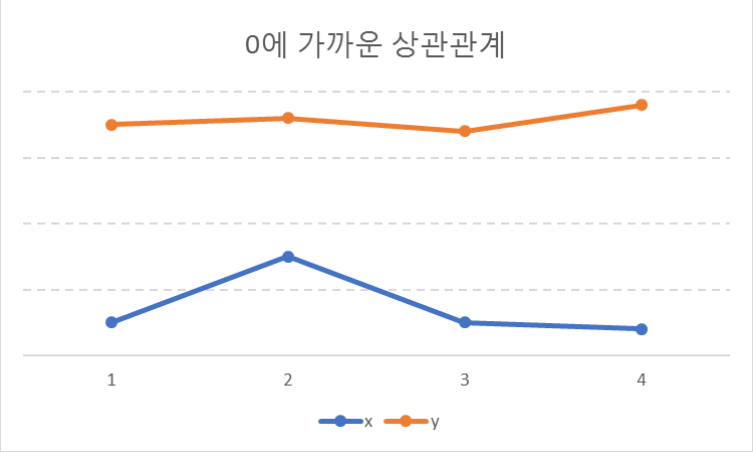


만일 두 개의 변수가 전혀 상관관계가 없다면 상관계수는 0.0 에 가까운 값이다. 다음의 경우 상관계수는 0.009405128 이다.

X	y
35	65
45	66
35	64
34	68

차트를 그려보면 다음과 같다. 두 변수의 움직임은 크게 관계가 있어 보이지 않는다. 각각의 변수는 상대방의 움직임과 상관없이 제 갈 길을 가는 듯 보인다.

그림 400 의 상관관계



다음은 앞서의 상관계수 계산을 프로그래밍한 한 것이다.

<코드>

```
01: # 상관계수를 계산하기 위해 필요한 함수들
02: import math

# 평균을 계산하는 함수
03: def mean(x):
04:     return sum(x) / len(x)

# 두 리스트의 곱의 합계 즉 엑셀의 SUMPRODUCT()함수와 같다.
05: def sum_of_product(xs, ys):
06:     return sum( x * y for x, y in zip(xs, ys) )

# 제곱합을 계산하는 함수
07: def sum_of_squares(v):
08:     return sum_of_product(v, v)

# 편차를 계산하는 함수
09: def deviation(xs):
10:     x_mean = mean(xs)
11:     return [ x - x_meanr for x in xs ]

# 분산을 계산하는 함수
12: def variance(x):
13:     n = len(x)
14:     deviations = deviation(x)
15:     return sum_of_squares(deviations) / (n-1)

# 공분산을 계산하는 함수
16: def covariance(x, y):
17:     n = len(x)
18:     return sum_of_product( deviation(x), deviation(y) ) / (n-1)
```

```
# 표준편차를 계산하는 함수
19: def standard_deviation(x):
20:     return math.sqrt( variance(x) ) # math 모듈의 제곱근 함수 sqrt()를 사용

# 상관계수를 계산하는 함수
21: def correlation(xs, ys):
22:     stdev_x = standard_deviation(xs)
23:     stdev_y = standard_deviation(ys)
24:     if stdev_x > 0 and stdev_y > 0:
25:         return covariance(xs, ys) / (stdev_x * stdev_y)
26:     else :
27:         return 0 # 편차가 존재하지 않는다면 상관관계는 0

# x 와 y 리스트와 correlation()함수를 사용하여 상관관계를 계산
28: x=[ 41, 43, 38, 37 ]
29: y=[ 61, 63, 56, 55 ]
30: print( correlation(x, y) )

31: x=[ 35, 45, 35, 34 ]
32: y=[ 65, 54, 64, 67 ]
33: print( correlation(x, y) )

34: x=[ 35, 45, 35, 34 ]
35: y=[ 65, 66, 64, 68 ]
36: print( correlation(x, y) )
</코드>
```