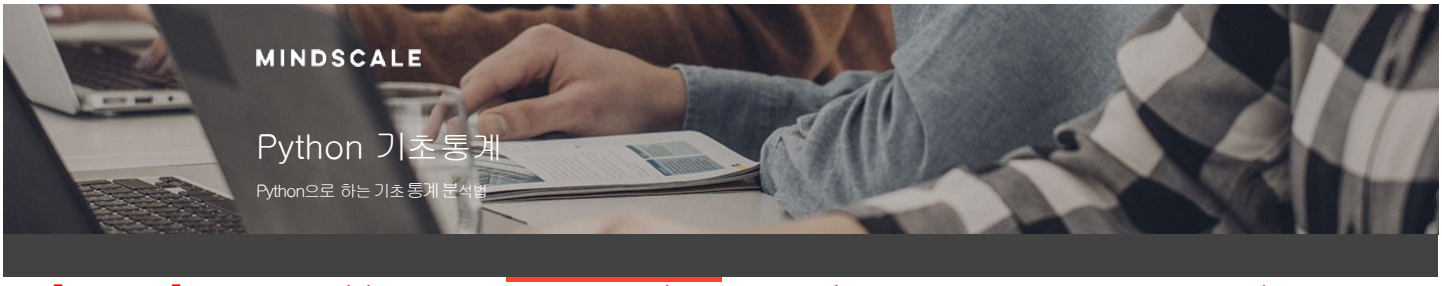


[참고] https://mindscale.kr/course/basic-stat-python/

제목

1. 변수와 척도

변수의 종류

연속변수

- 연속적인 값을 가지는 변수
  예) 나이, 점수, 무게, 가격 등

범주변수(이산변수)

- 서로 다른 것으로 구분되는 변수
  예) 성별(남자, 여자), 애완동물(강아지, 고양이)

척도 (명->순->구->비)

척도 (scale) : 측정된 변수의 값을 표현하는 수준 (levels of measurement)을 의미

명목척도 (nominal scale)

- 측정값이 같고 다름을 말할 수 있음
- 측정값들 사이에 순서가 없음
- 사칙연산이 불가능
- 종류에 따른 빈도만 계산
  예) 성별, 혈액형, 거주지역, 인종

순위척도 (ordinal scale), 서열척도

- 측정값들 사이에 순서가 있음
- 측정값들의 간격이 동일하지 않음
- 사칙연산은 불가능
  예) 직급(부장 과장 대리)...) 대리보다 높음
  부장이 과장보다 과장이 대리보다 높음
  부장과 과장의 차이가 과장과 대리의 차이와 같지 않음

구간척도 (interval scale), 등간척도

- 측정값들 사이에 순서가 있고 간격이 일정
- 영점(0)의 의미가 임의적(영점을 옮겨도 무방함)
- 덧셈 뺄셈이 가능
- 예) 섭씨온도
  섭씨 20도는 섭씨 10도보다 수치로는 2배
  그러나 2배 따뜻한 것이 아님
  화씨로 바꾸면 각각 50도와 68도가 되어 1.36배에 불과
  영점의 기준이 임의적이기 때문(섭씨 0도 = 화씨 32도)

비율척도 (ratio scale)

- 구간척도 + 절대영점(순서, 간격, 영점)
- 사칙연산 모두 가능
- 절대영점이란 영점의 의미가 아무 것도 존재하지 않는 상태를 말함
- 예) 길이
  20미터는 10미터보다 수치로도 2배이고, 실제로도 2배 김
  미터를 피트로 바꿔도 328피트와 656피트로 2배
  영점의 기준이 절대적 (0미터 = 0피트)

척도의 중요성

- 척도에 따라 적용가능한 통계 분석방법이 다름
- 숫자로 표현된 경우라 하더라도 무조건 사칙연산이가능하지는 않음, 그 숫자의 의미(=척도) 를 이해해야
  예) 남자 = 1, 여자 = 2로표현하는 경우
  수로 표현되었지만, 1 + 1 = 2와 같이계산하면 남자 2명이여자 1명과 같다는 이상한 해석이됨

- 가능하면 비율척도나 등간척도의 형태로 자료 수집을 하는 것이분석에 용이
  예) 연령을 조사하는경우
  서열척도로 조사 : "어린이 청장년 노인"과 같이 나누어 조사
  비율척도로 조사 : 만 나이로조사
  비율척도는 다양한 계산과 분석이가능하지만, 서열척도는 어린이몇 명, 노인몇 명 등의 분석만 가능

2. 도수분포표와 히스토그램

- 도수분포표 (frequency table) : 데이터를 구간으로 나누어 각 구간의 빈도를 나타낸 표
- 히스토그램 (histogram) : 도수분포표를 그래프로 그린것

범주 변수

다음과 같이 혈액형이 있다고 하자

```
blood = ['A', 'A', 'A', 'B', 'B', 'AB', 'O']
```

numpy 를 이용한 도수 분포표

```
import numpy as np
np.unique(blood, return_counts=True)
```

```
(array(['A', 'AB', 'B', 'O'], dtype='<U2'), array([3, 1, 2, 1], dtype=int64))
```

pandas 를 이용하는 방법

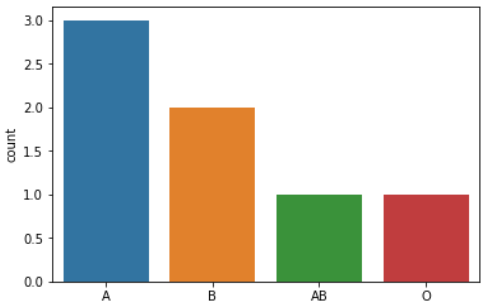
```
import pandas as pd
pd.Series(blood).value_counts()
```

```
A     3
B     2
O     1
AB    1
dtype: int64
```

시각화

```
import seaborn as sns
sns.countplot(x='b', data=pd.DataFrame({'b':blood}))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x151634bb0b8>
```



연속변수

```
x = [1, 1, 1, 2, 3, 5, 5, 7, 8, 9]
```

데이터의 범위를 4구간으로 나눔

```
hist, edges = np.histogram(x, 4)
```

- 먼저 각 구간의 경계는 1,3,5,7,9임
- 1이상 3미만, 3이상 5미만, 등등으로 구간이 정해짐

```
edges
```

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

첫번째 구간 1~ 3의 빈도는 4, 두번째 구간 3~5의 빈도

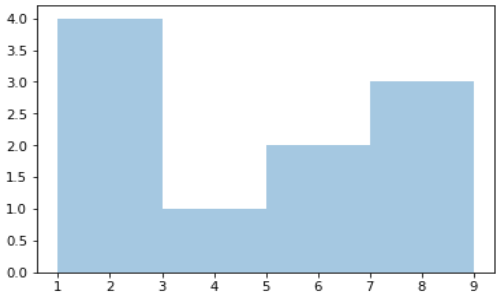
```
hist
```

```
array([4, 1, 2, 3], dtype=int64)
```

시각화

```
sns.distplot(x, bins=4, kde=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x15164622198>
```



3. 중심경향치(평균, 최빈치, 중앙값)

중심경향치 (central tendency measures)란

- 자료의 중심을 나타내는 숫자
- 자료 전체를 대표
- 평균, 중간값, 최빈값 등이 있다

평균

- 평균 (mean) 자료의 합을 자료의 갯수로 나눈 값

다음과 같은 리스트가있을 때

```
x = [100, 100, 200, 400, 500]
```

numpy 를 이용해 다음과 같이 평균을 구할 수 있다

```
import numpy
numpy.mean(x)
```

```
260.0
```

평균은 극단적인 값 (outliers) 의 영향을 잘 받는다. 위의 x 에서 500 하나를 1700으로 바꾸었을 뿐이지만 전체 평균은 크게 변한다

```
y = [100, 100, 200, 400, 1700]
numpy.mean(y)
```

```
500.0
```

소득처럼 분포가 비대칭적인 경우에도 평균이 자료를 잘 대표하기 어렵다
평균은 사칙연산 중 덧셈이 가능해야 하므로, 등간척도/비율척도에서 쓸 수 있다.
서열척도와 명목척도에서는 쓸 수 없다

중간값

- 중간값 (median) 자료를 크기 순으로 정렬했을 때 정 가운데에 있는 값
- 자료의 상위 50%와 하위 50%를 가르는 지점
  중앙값 또는 중위수 라고도 한다

```
numpy.median(x)
```

```
200.0
```

극단적인 값에 영향을 받지 않는다

```
numpy.median(y)
```

```
200.0
```

데이터가 짝수 개일 경우에는 가운데 두 값의 평균

```
numpy.median([100, 200, 300, 400])
```

```
250.0
```

자료를 크기 순으로 정렬할 수만 있으면 되므로 서열척도/등간척도/비율척도에서 쓸 수 있다.
명목척도에서는 쓸 수 없다

최빈값

- 최빈값 (mode) 가장 빈번하게 관찰 측정되는 값

```
from scipy.stats import mode
```

```
mode(x)
```

```
ModeResult(mode=array([100]), count=array([2]))
```

모든 척도에 가능하나 주로 범주변수(명목척도/순위척도) 에 사용

4. 변산성 측정치(분산, 표준편차, 범위, 사분위간 범위)

- 변산성 (variability) : 자료가 흩어져 있는 정도 혹은 개체에 따라 변할 수 있는 정도
- 중심경향치가 자료가 **무엇을 중심으로** 모여있는가 혹은 흩어져 있는가를 나타내는 것이라면,
- 변산성 측정치는 그 **모여있는 정도** 혹은 흩어져 있는 정도 를 의미함

실습을 위한 준비

```
import numpy
x = [1, 1, 2, 3, 3, 3, 4, 5, 5, 7]
```

범위 (range)

- 자료가갖는 최대값과 최소값 사이의 거리 즉 자료가얼마나 퍼져있는가를 나타냄
  범위 = 최댓값 - 최솟값

최솟값

```
numpy.min(x)
```

```
1
```

최댓값

```
numpy.max(x)
```

```
7
```

범위

```
numpy.max(x) -  numpy.min(x)
```

```
6
```

분산

- 평균에서 데이터가 벗어난 정도를 수치화한 값
- 각각의 데이터에서 평균값을 빼고 그것을 제곱하여 평균을 구함
- 분산이 크면 : 데이터가 평균에서 많이 벗어나 있다
- 분산이 작으면 : 데이터가 평균 주변에 모여 있다

```
numpy.var(x)
```

```
3.2399999999999998
```

표준편차

- 평균에서 데이터가벗어난 정도를 수치화한 값

```
numpy.std(x)
```

```
1.8
```

표준편차는 분산의 양의 제곱근

아래와 같이 계산하면 numpy.std 한 것과 같음

```
numpy.sqrt(numpy.var(x))
```

```
1.8
```

numpy.sqrt 로제곱근을 구할 수 있음 sqrt(4) = 2.

```
numpy.sqrt(4)
```

```
2.0
```

사분위간 범위

- 사분위간 범위 (IQR InterQuartile Range) 는 제3사분위수에서 제1사분위수 간의 범위
- 사분위수란 전체 데이터를 작은 값부터 큰 값까지순서대로나열한 후 4등분 하였을 때, 각 지점에 해당하는 값
  - 제1사분위수 (Q1) : 25% 지점
  - 제2사분위수 (Q2) : 50% 지점 = 중간값
  - 제3사분위수 (Q3) : 75%
- 제1사분위수와 제3사분위수 사이의구간에는 항상 전체 데이터의 50%가포함 됨

사분위는 임의로정하는 기준이므로필요에 따라 십분위 등으로변경가능

제1사분위수

```
numpy.quantile(x, .25)
```

```
2.25
```

제2사분위수 (50% 지점) = 중간값

```
numpy.quantile(x, .5)
```

```
3.0
```

```
numpy.median(x)
```

```
3.0
```

제3사분위수 (75% 지점)

```
numpy.quantile(x, .75)
```

```
4.75
```

사분위간 범위 (Q3 – Q1)

```
numpy.quantile(x, .75) - numpy.quantile(x, .25)
```

```
2.5
```

변산성 측정치를 이용한 이상점 진단

- 대부분의 자료는 중심경향치 주변에 몰려있음
- 변산성 측정치를 기준으로벗어난 정도를 파악할 수 있음
- 평균에서 벗어난 정도를 판단할 때는 표준편차를 사용
  - 평균에서 표준편차의 n배 떨어져 있으면 "n표준편차"와 같이 표현
- 중간값에서 벗어난 정도를 판단할 때는 IQR을 사용
- 중심경향치에서 크게 벗어났다면 이상점으로 의심할 수 있음

[참고 : 기초 통계 기본 개념 – 분산구하는 식]
https://gooopy.tistory.com/category/Python%EC%9C%BC%EB%A1%9C%20%ED%95%98%EB%8A%94%20%EA%B8%B0%EC%B4%88%ED%86%B5%EA%B3%84%ED%95%99/%EA%B8%B0%EB%B3%B8%20%EA%B0%9C%EB%85%90?page=3

- **모집단의 분산과 표준편차**

모집단의 분산: $\sigma^2 = \frac{\sum(X_i-\mu)^2}{N}$

모집단의 표준편차: $\sigma = \sqrt{\frac{\sum(X_i-\mu)^2}{N}}$

- **표본집단의 분산과 표준편차**

표본집단의 분산: $S^2 = \frac{\sum(X_i-\bar{X})^2}{n-1}$

표본집단의 표준편차: $S = \sqrt{\frac{\sum(X_i-\bar{X})^2}{n-1}}$

5. 공분산

공분산 (covariance) 두 변수가 함께 변화하는 정도를 나타내는 지표

공분산

두 개의 데이터 x 와 y 가있을 때

```
x = [8, 3, 6, 6, 9, 4, 3, 9, 3, 4]
y = [6, 2, 4, 6, 10, 5, 1, 8, 4, 5]
```

산점도로 나타내기 가로축은 x , 세로축은 y 를 나타냄

```
import matplotlib.pyplot as plt
plt.plot(x, y, 'o')
```

```
[<matplotlib.lines.Line2D at 0x20e0fb52898>]
```

np.cov 를 사용하면 x의 분산 y의 분산과 함께 x와 y의 공분산을 계산

```
import numpy as np

np.cov(x, y)
```

```
array([[6.05555556, 5.61111111],
       [5.61111111, 6.98888889]])
```

공분산만 보고자 한다면 아래와 같이 함

```
np.cov(x, y)[0, 1]
```

```
5.61111111111111
```

공분산의 해석

부호:

- 공분산이 +인 경우 : 두 변수가 같은 방향으로 변화 (하나가 증가하면 다른 하나도 증가)
- 공분산이 -인 경우 : 두 변수가 반대 방향으로 변화 (하나가 증가하면 다른 하나는 감소)

크기:

- 공분산 = 0이면 두 변수가 독립, 즉, 한 변수의 변화로 다른 변수의 변화를 예측하지 못함
- 공분산의 크기가 클 수록 두 변수는 함께 많이 변화
  - 단위에 따라 공분산의 크기가 달라지므로 절대적 크기로 판단이 어려움
  - 공분산을 -1 ~ 1 범위로 표준화 시킨것이 상관계수

주의: 공분산은 선형적인 관계를 측정하기 때문에 두 변수가 비선형적으로 함께 변하는 경우는 잘 측정하지 못함

아래 z와 w는 $w = z^2$ 의 관계가 있지만 공분산은 0

```
z = [-3, -2, -1, 0, 1, 2, 3]
w= [9, 4, 1, 0, 1, 4, 9]


np.cov(z, w)[0, 1]
```

```
0.0
```

6. 상관계수

상관계수 (correlation coefficient) 두 변수가 함께 변하는 정도를 -1~1 범위의 수로 나타낸 것

피어슨 상관계수

- 칼 피어슨 (Karl Pearson) 이 개발한 상관계수
- 적률상관계수 (product moment correlation coefficient) 라고도 함
- 일반적으로 상관계수라고 하면 피어슨 상관계수를 말함

np.corrcoef 함수로 계산

```
import numpy as np

np.corrcoef(x, y)
```

$$r = \frac{S_{xy}}{\sqrt{S_{xx}} \cdot \sqrt{S_{yy}}}$$

두 변수의 상관계수만 확인하려면

```
np.corrcoef(x, y)[0, 1]
```

$$\bar{x} = \frac{1}{n}\sum x_i, \bar{y} = \frac{1}{n}\sum y_i$$

$$S_{xx} = \sum (x_i - \bar{x})^2$$

$$S_{yy} = \sum (y_i - \bar{y})^2$$

$$S_{xy} = \sum (x_i - \bar{x})(y_i - \bar{y})$$

아래와 같은 원리로 계산

```
cov = np.cov(x, y)[0, 1] #공분산
xsd = np.std(x, ddof=1)    #x의 표본표준편차
ysd = np.std(y, ddof=1)    #y의 표본표준편차

cov / ( xsd * ysd )
```

상관계수의 해석

공분산과 비슷하게 해석

부호 :
- 상관이 +인 경우 : 두 변수가 같은 방향으로 변화 하나가 증가하면 다른 하나도 증가
- 상관이 - 인 경우 : 두 변수가 반대 방향으로 변화 하나가 증가하면 다른 하나는 감소

크기 :
- 상관 = 0이면 두 변수가 독립 즉 한 변수의 변화로 다른 변수의 변화를 예측하지 못함
- 상관이 클수록 두 변수는 함께 많이 변화
- pearson 상관계수를 제곱하면 분산=변화량 에서 공유하는 비율
  - 예를 들어 x와 y의 상관계수가 0.4이면 그 제곱은 0.16(=16%)
  - x의 분산 중 16%를 y와 공유 또는 x의 분산 중 16%가 y로 설명됨

주의: 상관계수는 선형적인 관계를 측정하기 때문에 두 변수가 비선형적으로 함께 변하는 경우는 잘 측정하지 못함

아래 z와 w는 $\mathbf{w = z^2}$ 의 관계가 있지만 상관계수는 0

```
z = [-3, -2, -1, 0, 1, 2, 3]
w= [9, 4, 1, 0, 1, 4, 9]

np.corrcoef(z, w)[0, 1]
```

- pearson 상관계수는 공분산을 그 기반으로 하기 때문에 기본적으로 **구간척도/비율척도** 를 사용한 변수에만 적용가능함

spearman 상관계수

- 상관분석을 실시함에 있어 **순위척도(서열척도)**를 사용한 변수가 포함되어 있거나
- 등간 비율척도를 사용한 변수들이라 하더라도 두 변수 간의 관계가 **비선형적** 일 때 구하는 상관계수

```
import scipy.stats

scipy.stats.spearmanr(x, y).correlation
```

kendall의 tau

- spearman의 상관계수와 마찬가지로 비선형적 관계이거나 서열변수일 때 사용
- spearman의 상관계수보다 믿을만 한 것으로 알려짐 (특히 표본이 작을 때)

```
scipy.stats.kendalltau(x, y).correlation
```

7. 상관계수의 통계적 검증

상관분석

- 우리가구한 상관계수는 **표본**에서 구한 것
- 동일한 모집단에서도 표본에 따라 상관계수가달라질 수 있음

```
x = [8, 3, 6, 6, 9, 4, 3, 9, 3, 4]
y = [6, 2, 4, 6, 10, 5, 1, 8, 4, 5]
```

두 변수의 피어슨 상관계수와 p값을 계산

```
import scipy.stats

scipy.stats.pearsonr(x, y)
```

```
(0.8625172792135779, 0.0013196539142000057)
```

상관계수는 0.86이고, p값은 0.001.

p값은 모집단에서 상관계수가0일 때 현재와 같은 크기의 표본에서 관찰된 상관계수 여기서는 0.86보다더 극단적인상관계수가 관찰될 확률

보통 0.05 (5%)와 같은 임계치를 정하고, p값이 그보다 작을 경우 "통계적으로 유의미하다" 라고 함.

8. 상관분석 유의할 점

상관분석 실시 전 유의할 점

- **두 변수의 관계가 선형적 (= 직선)인지 확인할 것!**
- **산점도을 그려서 확인**
- **명확하게 두 변수의 관계가 곡선 형태라면 spearman이나 kendall의 방법을 사용**

상관분석 결과 해석 시 유의할 점

상관분석 결과 해석 시 유의할 점

- 두 변수의 상관관계는 인과관계를 담보하지 않음
- 상관관계가 있다고 반드시 인과관계가 있는 것은 아님
- 제3 변인의 문제
- 도시 내 범죄 발생 건수와 종교 시설의 수는 양의 상관 관계가 있음
- 범죄가 많아서 종교에 의존하는가? 또는 종교가 범죄를 부추기는가?
- 사실은 인구가 많아지면 범죄도 늘고, 종교 시설도 많아짐.
- 이질적인 집단들의 합 (심슨의 역설)
- 각 집단별 상관관계와 전체 총합의 상관관계는 다를 수 있음
- 상관분석 결과가 예상과 다를 경우, 이질적인 하위집단들이 존재하는지 살펴봐야 할 수도 있음
- 극단치(outliers)에 의한 인위적 상관 존재 가능성
- 자료 내에 극단치가 있을 때, 존재하지 않는 상관관계가 포착되거나, 존재하는 상관관계가 포착되지 못하는 경우가 생기기도 함

[상관관계와 인과관계]
https://dbr.donga.com/article/view/1303/article_no/6894/ac/magazine

9. 회귀분석이란 무엇인가

회귀분석 (regression)은 가장 넓은 의미로는 독립변수 x로종속변수 y를 예측하는 것을 의미한다

독립변수와 종속변수

체중과 식사량의 관계에 대한 모형을 만든다고 해보자. 이모형에서 체중이 식사량에 따라 달라진다고 가정하면 식사량이 많아지면 체중도 증가하고,식사량이 감소하면 체중도 감소할 것이다.그러면 체중은 식사량에 종속되었다고 할 수 있다.그래서 체중은 종속변수가 된다

그런데 이모형은 체중이 식사량이 왜 변하는지는 말하고있지 않다.

식사량은 모형에 독립적으로 변하므로 독립변수가 된다

- 독립변수 : 변수의 변화 원인이 모형 **밖**에 있는 변수
- 종속변수 : 변수의 변화 원인이 모형 **안**에 있을 변수

혼입변수

혼입변수 (confounding variable) 는 모형에 포함되지 않았지만, 종속 변수에 영향을 미치는 변수이다

예를 들어 건강이 좋지 않아 식사량이 줄고, 체중도 줄어든 사람들이 있다면 마치 식사량이 줄면 체중이 줄어드는 것처럼 보일 수도 있다

선형회귀분석

회귀분석을 좀 더 좁은 의미로 말할 때는 종속변수가 연속인 경우를 말한다. 종속변수가 범주형인 경우에는 '분류' 라고 한다

더 좁은 의미로는 선형 회귀 분석을 의미한다.
선형 회귀 분석은 독립변수와 종속변수 사이에 직선적인 형태의 관계가 있다고 가정을 한다

직선적인 형태란 독립변수가 일정하게 증가하면, 종속변수도 그에 비례해서 증가하거나 또는 감소하는 형태를 말한다

예를 들어 밥을 한 그릇 더 먹으면 체중이 정확히 100g 증가한다면 직선적인 형태라고 할 수 있다.
밥을 한 그릇, 두 그릇, 세 그릇 .. 먹으면 체중이 100g, 200g, 300g으로 증가하기 때문에 이것을 그림으로 그려보면 직선이 그려진다

회귀계수

선형 모형에서 x와 y의 관계를 수식으로 나타내면 아래와 같다

$$y = wx + b$$

위의 식에서 w를 회귀계수 (coefficient) 라고 한다 독립변수 x가 1 증가할 때마다 종속변수 y는 w만큼 증가
밥을 1그릇 먹을 때마다 체중이 100g씩 증가하면 밥 그릇 수에 100을 곱하면 체중의 변화량이 된다
따라서 이때의 회귀계수는 100이라고 할 수 있다

그림으로 그리면 회귀계수는 직선의 기울기(slope) 가된다
위의 식에서 독립변수 x = 0이면 y = b가 된다. 이를 y 절편 또는 간단히 절편 (intercept) 이라고 한다

- 절편 (intercept) : 독립변수가 모두 0일때 종속변수 y의 값

회귀분석의 결과로 알 수 있는 것

- **모형적합도**: 모형이 데이터에 얼마나 잘 맞는가
  - 예) 식사량과 체중의 관계가 데이터에 잘 맞는지 검증해볼 수 있다
- **회귀계수**: 독립변수의 변화가 종속변수를 얼마나 변화시키는가
  - 예) 식사량이 증가하면 체중이 얼마나 증가하는지 알 수 있다

10. 회귀분석의 사전 진단

판다스를 불러들인다

```
import pandas as pd
```

실습을 위해 cars.csv를 다운로드 받아 연다

```
df = pd.read_csv('cars.csv')
```

cars 데이터는 speed 와 dist 두 개의 변수로 이뤄져 있다. 여기서는 speed 를 독립변수 dist 를 종속변수로 사용한다

```
df.head()
```

	speed	dist
0	4	2
1	4	10
2	7	4
3	7	22
4	8	16

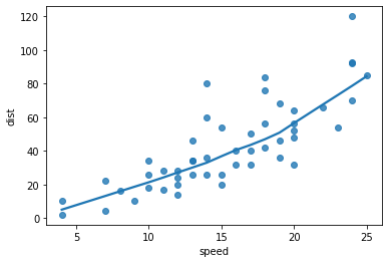
사전작업

산점도에 추세선을 넣어 데이터에 선형적인 패턴이 있는지 확인한다. 아래와 같이 그려보면 대체로 speed 가 증가할 수록 dist도 증가하는 관계가 있는 것을 볼 수 있다

```
import seaborn as sns
```

```
sns.regplot('speed', 'dist', lowess=True, data = df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x250129a7cc8>
```



극단값이 있을 경우 회귀분석의 결과가 왜곡될 수 있다 상자. 그림을 그려서 극단값이 있는지 확인해본다. 아래 그림을 보면 dist 에서 값 하나가 크게 위에 있는 것을 볼 수 있다

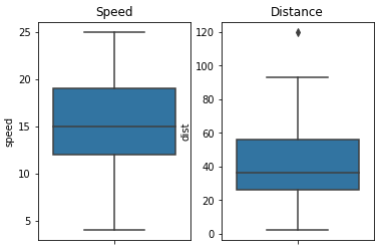
```
import matplotlib.pyplot as plt
```

```
# 1행 2열 형태로 2개의 그래프를 그린다
fig, (ax1, ax2) = plt.subplots(1, 2)

# speed의 상자 그림을 첫번째(ax1)로 그린다. 방향은 수직(orient='v')
sns.boxplot('speed', data=df, ax=ax1, orient='v')
ax1.set_title('Speed')

# dist의 상자 그림을 두번째(ax2)로 그린다.
sns.boxplot('dist', data=df, ax=ax2, orient='v')  ax2.set_title('Distance')
```

```
Text(0.5, 1.0, 'Distance')
```



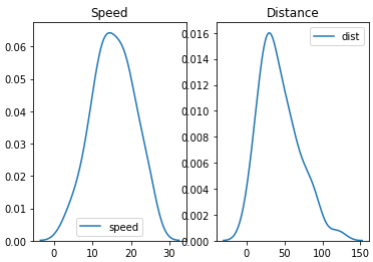
선형회귀분석은 독립변수와 종속변수가 정규분포를 따를 때 잘 작동한다.
밀도 플롯 (density plot) 을 그려서 정규분포의 형태인지 확인해본다

```
# 1행 2열 형태로 2개의 그래프를 그린다
fig, (ax1, ax2) = plt.subplots(1, 2)

# speed의 밀도 플롯
sns.kdeplot(df['speed'], ax=ax1)
ax1.set_title('Speed')

# dist의 밀도 플롯
sns.kdeplot(df['dist'], ax=ax2)  ax2.set_title('Distance')
```

```
Text(0.5, 1.0, 'Distance')
```



대체로 중심부에 데이터가 몰려있고 좌우로 갈수록 줄어드는 정규분포와 비슷한 형태를 보인다.
앞에서 봤듯이 dist 에 위로 치우친 값이 있기 하나 있기 때문에 밀도 플롯도 오른쪽으로 약간 늘어진 모습을 보인다

데이터가 치우친 정도를 나타내는 왜도 (skewness) 를 구해본다.
e1071 라이브러리가 없을 경우에는 install.packages('e1071') 을 실행하여 설치한다

```
import scipy.stats

scipy.stats.skew(df['speed'])
```

```
-0.11395477012828319
```

```
scipy.stats.skew(df['dist'])
```

```
0.7824835173114966
```

speed 는 왜도가 -0.11이다. 마이너스 쪽으로 약간 치우쳤다는 것을 뜻한다.
dist 의 왜도는 0.76으로 플러스 쪽으로 어느 정도 치우쳤다. 위에서 그래프로 본 것과 비슷한 결과이다

11. 회귀분석 실시하기

예제 데이터 준비

판다스를 불러들인다

```
import pandas as pd
```

실습을 위해 cars.csv를 다운로드 받아 연다

```
df = pd.read_csv('cars.csv')
```

회귀분석 실시

statsmodels 를 불러들인다

```
from statsmodels.formula.api import ols
```

ols 함수로 회귀분석을 실시한다. 종속변수 ~ 독립변수 의 형태로 모형식을 쓴다.
(수학에서는 y = f(x)처럼 종속변수를 왼쪽에 독립변수를 오른쪽에 쓰는 것이 관습)

```
res = ols('dist ~ speed', data=df).fit()
```

결과는 .summary() 메소드로 확인할 수 있다

```
res.summary()
```

OLS Regression Results

Dep. Variable:	dist	R-squared:	0.651
Model:	OLS	Adj. R-squared:	0.644
Method:	Least Squares	F-statistic:	89.57
Date:	Thu, 23 Jan 2020	Prob (F-statistic):	1.49e-12
Time:	13:58:33	Log-Likelihood:	-206.58
No. Observations:	50	AIC:	417.2
Df Residuals:	48	BIC:	421.0
Df Model:	1		
Covariance Type:	nonrobust		
coef	std err	t	P>|t|
	[0.025	0.975]	
Intercept	-17.5791	6.758	-2.601
	0.012	-31.168	-3.990
speed	3.9324	0.416	9.464
	0.000	3.097	4.768
Omnibus:	8.975	Durbin-Watson:	1.676
Prob(Omnibus):	0.011	Jarque-Bera (JB):	8.189
Skew:	0.885	Prob(JB):	0.0167
Kurtosis:	3.893	Cond. No.	50.7

Warnings

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

모형적합도

```
R-squared: 0.651
Adj. R-squared: 0.644
F-statistic: 89.57
Prob(F-statistic):
Log-Likelihood: -206.58
AIC: 417.2
BIC: 421.0
```

모형이 데이터에 잘 맞는 정도를 보여주는 지표들이다

- R-squared 0.6511
  - R 제곱
  - 모형 적합도 혹은 설명력
  - dist 의 분산을 speed 가약 65%를 설명한다
  - 각 사례마다 dist 에 차이가있다
- Adj. R-squared 0.6438
  - 독립변수가 여러개인 다중회귀분석에서 사용
  - 독립변수의 개수와 표본의 크기를 고려하여 R-squared를 보정
  - 서로다른 모형을 비교할 때는 이지표가 높은 쪽은 선택한다
- F-statistic 89.57, Prob(F-statistic) : 1.49e-12
  - 회귀모형에 대한 통계적 유의미성 검증 결과 유의미함 ($p < 0.05$)
  - 즉 이모형은 주어진 표본 뿐 아니라 모집단에서도 의미있는 모형이라 할 수 있음
- 로그 우도 : 종속변수가 정규분포라 가정했을 때 그 우도
  - 로그우도도 R제곱과 마찬가지로 독립변수가 많아지면 증가한다
  - AIC BIC 로그우도를 독립변수의 수로 보정한 값 (작을 수록 좋다)

회귀계수 (Coefficients)

Coefficients:	Estimate	P(>|t|)
(Intercept)	-17.5791	0.012
speed	3.9324	0.000

Coef는 데이터로부터 얻은 계수의 추정치를 말한다

절편 Intercept 의 추정치는 -17.5791로, speed 가0일 때 dist 의 값이다

speed 의 계수 추정치는 3.9324로 speed 가1 증가할 때마다 dist 가3.9324 증가

한다는 것을 의미한다 이를 수식으로 정리하면 아래와 같다

$$\text{dist} = -17.5791 + 3.9324 \times \text{speed}$$

추정치의 표 중간의 P(>|t|) 는 모집단에서 계수가0일 때 현재와 같은 크기의 표본에서 이러한 계수가 추정될 확률인p값을 나타낸다. 이확률이 매우 작다는 것은 모집단에서 speed 의 계수가 정확히 3.9324는 아니더라도 현재의 표본과 비슷하게 0보다큰 어떤 범위에 있을 가능성이높다는 것을 의미한다. 보통 5%와 같은 유의수준을 정하여 p값이 그보다작으면 ($p < 0.05$), "통계적으로 유의미하다" 라고 한다.

즉 speed 가증가할 때 기대되는 dist 의 변화는 유의수준 5%에서 통계적으로 유의미하다

결과 보고

논문 등에서 회귀분석의 결과는 다음 순서대로 보고한다

먼저 모형적합도를 보고한다. F 분포의 파라미터 2개와 그 때의 F 값 p-value와 유의수준의 비교를 적시한다

dist에 대하여 speed로 예측하는 회귀분석을 실시한 결과, 이 회귀모형은 통계적으로 유의미하였다
(F (1, 48) = 89.57, p < 0.05).

다음으로 독립변수에 대해 보고한다
speed의 회귀계수는 3.9324로, dist에 대하여 유의미한 예측변인인 것으로 나타났다
(t (48) = 9.464, p < 0.05).

12. 절편의 고정

cars 데이터에서 dist 는 거리를 뜻한다. 앞선 분석 결과에서 speed 가 0일 때 dist 가 -17.5791이 되는데 거리가 마이너스가 될 수는 없으므로 해석이 어색하다. 이럴 경우 모형에서 절편을 제거한다 절편을 제거하면 절편을 0으로 고정시킨 것과 같아진다

예제 데이터 준비

판다스를 불러들인다

```
import pandas as pd
```

실습을 위해 cars.csv를 다운로드 받아 연다

```
cars = pd.read_csv('cars.csv')
```

절편의 제거

모형식에 0 + 를 추가하여 모형에서 절편을 제거한다

```
from statsmodels.formula.api import ols
res = ols('dist ~ 0 + speed', cars).fit()
res.summary()
```

Dep. Variable:	dist	R-squared (uncentered):	0.896
Model:	OLS	Adj. R-squared (uncentered):	0.894
Method:	Least Squares	F-statistic:	423.5
Date:	Thu, 23 Jan 2020	Prob (F-statistic):	9.23e-26
Time:	14:11:20	Log-Likelihood:	-209.87
No. Observations:	50	AIC:	421.7
Df Residuals:	49	BIC:	423.7
Df Model:	1		
Covariance Type:	nonrobust		

OLS Regression Results

	coef	std err	t	P>|t|	[0.025	0.975]
speed	2.9091	0.141	20.578	0.000	2.625	3.193

Omnibus:	14.345	Durbin-Watson:	1.409
Prob(Omnibus):	0.001	Jarque-Bera (JB):	15.573
Skew:	1.202	Prob(JB):	0.000415
Kurtosis:	4.302	Cond. No.	1.00

Warnings

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

R제곱

모형에서 절편을 제거한 경우 R제곱을 해석할 때 주의가 필요하다.
절편이 있는 경우와 없는 경우 계산 방식이 다르기 때문이다.
절편이 있는 경우에 R제곱은 **종속변수의 분산**에서 설명하는 비율을 나타내지만,
절편이 없는 경우는 **종속변수의 제곱의 평균**에서 설명하는 비율을 나타낸다.

13. 다중공선성

공선성 (collinearity) 하나의 독립변수가 다른 하나의 독립변수로 잘 예측되는 경우 또는 서로 상관이 높은 경우

다중공선성 (multicollinearity) 하나의 독립변수가 다른 여러 개의 독립변수들로 잘 예측되는 경우

(다중)공선성이 있으면

- 계수 추정이 잘 되지 않거나 불안정해져서 데이터가 약간만 바뀌어도 추정치가 크게 달라질 수 있다
- 계수가 통계적으로 유의미하지 않은 것처럼 나올 수 있다

다중공선성의 진단

- 분산팽창계수 (VIF, Variance Inflation Factor)를 구하여 판단
- 엄밀한 기준은 없으나 보통 10보다 크면 다중공선성이 있다고 판단 5를 기준으로 하기도 함

예제 데이터

crab.csv를 다운로드 받아 연다 이 데이터는 게의 크기와 무게 등을 측정한 데이터이다

```
import pandas as pd
df = pd.read_csv('crab.csv')
df.head()
```

	crab	sat	y	weight	width	color	spine
0	1	8	1	3.05	28.3	2	3
1	2	0	0	1.55	22.5	3	3
2	3	9	1	2.30	26.0	1	1
3	4	0	0	2.10	24.8	3	3
4	5	4	1	2.60	26.0	3	3

회귀분석

```
from statsmodels.formula.api import ols
```

모형을 만든다

```
model = ols('y ~ sat + weight + width', df)
```

모형을 적합시킨다

```
res = model.fit()
```

분석 결과를 확인한다

```
res.summary()
```

Dep. Variable:	y	R-squared:	0.514
Model:	OLS	Adj. R-squared:	0.506
Method:	Least Squares	F-statistic:	59.69
Date:	Thu, 23 Jan 2020	Prob (F-statistic):	2.30e-26
Time:	16:11:28	Log-Likelihood:	-55.831
No. Observations:	173	AIC:	119.7
Df Residuals:	169	BIC:	132.3
Df Model:	3		
Covariance Type:	nonrobust		

OLS Regression Results

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-0.9366	0.500	-1.872	0.063	-1.924	0.051
sat	0.0971	0.009	11.018	0.000	0.080	0.115
weight	-0.0465	0.098	-0.475	0.635	-0.240	0.147
width	0.0535	0.026	2.023	0.045	0.001	0.106

Omnibus:	29.724	Durbin-Watson:	1.475
Prob(Omnibus):	0.000	Jarque-Bera (JB):	7.545
Skew:	0.086	Prob(JB):	0.0230
Kurtosis:	1.992	Cond. No.	526.

Warnings:

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

결과를 보면 유의수준 5%에서 `sat` 와 `width` 는 통계적으로 유의미하고, `weight` 는 유의미하지 않게 나왔다

VIF 계산

`statsmodels` 에서 `variance_inflation_factor` 함수를 불러들인다

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

모형식에서 독립변수는 절편 `sat` , `weight` , `width` 순이다

```
model.exog_names
```

```
['Intercept', 'sat', 'weight', 'width']
```

X의 1번째 독립변수 `sat` 의 VIF를 계산한다

```
variance_inflation_factor(model.exog, 1)
```

```
1.15883687808578
```

X의 2번째 독립변수 `weight` 의 VIF를 계산한다

```
variance_inflation_factor(model.exog, 2)
```

```
4.8016794240392375
```

한 번에 모든 컬럼의 VIF를 계산한다

```
pd.DataFrame({'컬럼': column, 'VIF': variance_inflation_factor(model.exog, i)}
              for i, column in enumerate(model.exog_names)
              if column != 'Intercept') # 절편의 VIF는 구하지 않는다.
```

	컬럼	VIF
0	sat	1.158837
1	weight	4.801679
2	width	4.688660

weight 와 width 의 VIF가 각각 4.8과 4.6이다.
게의 무게 weight 와 너비 width 는 서로 상관이 높기 때문에 VIF가 약간 높게 나타나는 것이다

대처

- 계수가 통계적으로 유의미하지 않다면 대처
  - 계수가 통계적으로 유의미하다면 VIF가 크더라도 특별히 대처할 필요없음
- 변수들을 더하거나 빼서 새로운 변수를 만든다
  - (개념적으로나 이론적으로) 두 예측변수를 더하거나 빼더라도 문제가 없는 경우
    - 예) 남편의 수입과 아내의 수입이 서로 상관이 높다면 두 개를 더해 가족 수입이라는 하나의 변수로 투입한다
  - 더하거나 빼기 어려운 경우는 변수를 모형에서 제거한다
    - 단 변수를 제거하는 것은 자료의 다양성을 해치고 분석하려던 가설이나 이론에 영향을 미칠 수 있기 때문에 가급적 자제

weight 와 width 가 VIF 기준을 넘는 것은 아니지만 실험삼아 width 를 제거해보자

```
model = ols('y ~ sat + weight', df)
```

```
model.fit().summary()
```

Dep. Variable:	y	R-squared:	0.503
Model:	OLS	Adj. R-squared:	0.497
Method:	Least Squares	F-statistic:	85.93
Date:	Thu, 23 Jan 2020	Prob (F-statistic):	1.63e-26
Time:	16:18:20	Log-Likelihood:	-57.901
No. Observations:	173	AIC:	121.8
Df Residuals:	170	BIC:	131.3
Df Model:	2		
Covariance Type:	nonrobust		

OLS Regression Results

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.0495	0.114	0.433	0.665	-0.176	0.275
sat	0.0976	0.009	10.982	0.000	0.080	0.115
weight	0.1260	0.049	2.598	0.010	0.030	0.222

Omnibus:	40.033	Durbin-Watson:	1.433
Prob(Omnibus):	0.000	Jarque-Bera (JB):	8.709
Skew:	0.121	Prob(JB):	0.0128
Kurtosis:	1.928	Cond. No.	22.7

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

이전의 분석에서는 weight 가 유의미하지 않게 나왔지만 width 를 제거한 후에는 유의미하게 나왔다.
weight 와 width 가 공선성이 있기 때문에 width 를 제거하자. weight 가 유의미해진 것으로 볼 수 있다

14. 잔차분석

회귀분석 결과를 바탕으로 다양한 잔차 분석을 실시한다

실습 준비

실습을 위해 cars.csv를 다운로드 받아 열고 회귀분석을 한다

```
import pandas as pd
from statsmodels.formula.api import ols

df = pd.read_csv('cars.csv')
res = ols('dist ~ speed', data=df).fit()
```

모형의 선형성

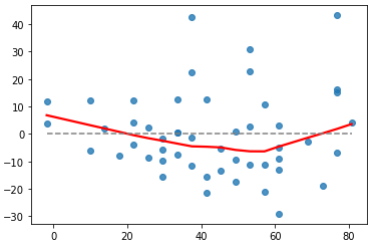
- 예측값 (fitted) 과 잔차 (residual) 의 비교
- 모든 예측값에서 잔차가 비슷하게 있어야 함 (가운데 점선)
- 빨간 실선은 잔차의 추세를 나타냄
- 빨간 실선이 점선에서 크게 벗어난다면 예측값에 따라 잔차가 크게 달라진다는 것

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
fitted = res.predict(df)
residual = df['dist'] - fitted
```

```
sns.regplot(fitted, residual, lowess=True, line_kws={'color': 'red'})
plt.plot([fitted.min(), fitted.max()], [0, 0], '--', color='grey')
```

```
[<matplotlib.lines.Line2D at 0x2b23d82bc88>]
```



잔차의 정규성

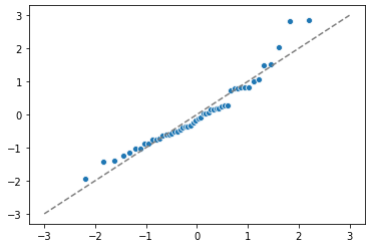
- 잔차가 정규분포를 따른다는 가정
- Q-Q 플롯으로 확인할 수 있음
- 잔차가 정규분포를 띄면 Q-Q플롯에서 점들이 점선을 따라 배치되어 있어야 함

```
import scipy.stats
```

```
sr = scipy.stats.zscore(residual)
(x, y), _ = scipy.stats.probplot(sr)
```

```
sns.scatterplot(x, y)
plt.plot([-3, 3], [-3, 3], '--', color='grey')
```

```
[<matplotlib.lines.Line2D at 0x2b23d9e2bc8>]
```



잔차의 정규성은 샤피로 검정으로 확인할 수 있다.
아래 분석에서 두 번째 값이 p값이다 p값이 0.020이므로 유의수준 5%에서 잔차의 정규성이 위반되었다고 판단한다

```
scipy.stats.shapiro(residual)
```

```
(0.9450905919075012, 0.02152460627257824)
```

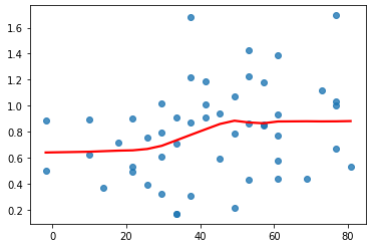
잔차의 등분산성

- 회귀모형을 통해 예측된 값이 크던 작던 모든 값들에 대하여 잔차의 분산이 동일하다는 가정
- 아래 그래프는 예측값 (가로축)에 따라 잔차가 어떻게 달라지는지 보여줌
- 빨간색 실선이 수평선을 그리는 것이 이상적

```
import numpy as np
```

```
sns.regplot(fitted, np.sqrt(np.abs(sr)), lowess=True, line_kws={'color': 'red'})
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2b23dc1f6c8>
```



극단값

- Cook's distance는 극단값을 나타내는 지표
- 48번 22번 38번 자료가 특히 예측에서 많이 벗어남을 알 수 있음

```
from statsmodels.stats.outliers_influence import OLSInfluence
```

```
cd, _ = OLSInfluence(res).cooks_distance
```

```
cd.sort_values(ascending=False).head()
```

```
48    0.340396
22    0.085552
38    0.068053
44    0.053176
34    0.052576
dtype: float64
```

잔차의 독립성

- 회귀분석에서 잔차는 정규성 등분상성 그리고 독립성을 가지는것으로 가정
- 자료 수집 과정에서 무작위 표집 (random sampling) 을 하였다면, 잔차의 독립성은 만족하는 것으로 봄
- 시계열 자료나 종단연구 자료처럼 연구 설계 자체가 독립성을 담보할 수 없는 경우에는
  더빈 왓슨 검정 (Durbin Watson test) 등을 실시

잔차 분석 결과를 바탕으로 대응

- 잔차 분석 결과에 따라 다양한 방식의 대응이 가능
  - 극단값을 제거
  - 독립변수를 추가
  - 종속변수를 수학적으로 변환

위의 예에서는 48번 자료가 극단값으로 보이고 이 때문에 잔차의 정규성이 위배되는 것으로 추측된다.
따라서 48번 자료를 제거하고 다시분석을 시도해볼 수 있다

15. t 검증

표준화 standardization

- 표준화란 어떤 변수의 값 들 에서 그 변수의 평균을 빼고 표준편차로 나누어주는 것

$$z = \frac{x - \mu}{\sigma}$$

- 표준화하는 이유 : 변수마다 각 변수의 단위와 변산성 =퍼져있는 정도 이 다르기 때문에 직접적인 비교를 할 수 없는데, 표준화를 하게 되면 이러한 정보들이 제거되므로 변수들 간에 비교가 가능해짐

t 검증

- 정규분포를 따르는 무선변수를 표준화하면 표준정규분포를 따르는 것으로 변환됨
  - 표준정규분포 평균이 0이고, 표준편차가 1인 정규분포
- 보통 모집단의 표준편차 (σ) 를 모름
  - 모집단의 표준편차 대신 표본의 표준편차를 이용하면 됨
  - 표본의 표준편차로 표준화를 하면, 변환된 무선변수는 표준정규분포를 따르지 않고 t 분포를 따름
- **t분포** : 표준정규분포와 매우 유사하게 생긴 분포 똑같지는 않음
- 따라서 표본에서 구한 분산 표준편차로 어떤 변수를 표준화를 하면 z값이 아닌 t값이 되고 이를 이용하여 검증하기 때문에 **t-test** 라고 함

16. 독립표본 t 검증

독립표본 t-test

- 두 독립표본의 평균을 통계적으로 비교하는 기법

독립표본?

- 두 표본집단을 구성 표집 수집할 때 독립이면 독립표본
- 대응표본 : 부부 100쌍을 뽑아 남편 100명과 아내 100명으로 두 집단을 비교하는 경우
- 독립표본 : 무작위로 남자 100명과 여자 100명을 뽑아 두 집단을 비교하는 경우

독립표본 t-test의 논리

- 두 집단의 모평균 모집단의 평균 은 동일하다는 가설 두
  - 집단의 모평균 차이는 0
- 표본평균들은 모집단 평균과 다를수 있지만 차이가 크지 않을 것
- 따라서 두 집단의 표본평균의 차이가 0은 아닐 수 있지만, 그렇게 큰 차이를 보이지는 않을 것임
- 만약. 두 표본평균이 심각하게 차이가 난다면. 두 집단의 모평균이 동일하다는, 즉 두 집단의 모평균 차이가 0이라는 가설이 맞지 않을 가능성이 높음
- 따라서, 두 표본평균이 차이나는 정도에 대한 가능성 확률 을 계산하고 이를 바탕으로 두 집단의 모평균 (모집단의 평균)은 동일하다는 가설에 대해서 판단함

두 집단 `dat_M` 와 `dat_F` 가있음

```
dat_M = [117, 108, 105, 89, 101, 93, 96, 108, 108, 94, 93, 112, 92, 91, 100, 96, 120, 86, 96, 95]
dat_F = [121, 101, 102, 114, 103, 105, 101, 131, 96, 109, 109, 113, 115, 94, 108, 96, 110, 112, 120, 100]
```

`dat_M` 의 평균은 100

```
import numpy as np

np.mean(dat_M)
```

```
100.0
```

`dat_F` 의 평균은 108

```
np.mean(dat_F)
```

```
108.0
```

두 집단의 평균 차이가 통계적으로 유의미한지 t-검증

```
import scipy.stats

scipy.stats.ttest_ind(dat_M, dat_F, equal_var=False)
```

```
Ttest_indResult(statistic=-2.670573872669349, pvalue=0.01108318824471652)
```

- 검증 결과 두 집단의 평균 차이는 통계적으로 유의미함 (p<0 05)

통계적 유의미성

- 검증 결과 통계적으로 유의미한 결과를 얻었다면 ?
- 귀무가설 (null hypothesis) 을 전제로 했을 때 어떤 통계량의 값을 얻을 확률이,
  연구자가 정한 유의수준보다 작을 경우 "통계적으로 유의미하다" 라고 표현함

유의수준

- 어떤 사건이 일어날 확률이 희박한지 판단하기 위해 연구자가 **주관적으로** 정하는 기준
- 어떤 사건이 일어날 확률이 유의수준보다 작으면 그 사건은 유의미한 사건 희귀한 사건 으로 여기겠다는 것임
- 통상적으로 학계에서는 0.05나 0.01을 관례적으로 사용

17. t 검증 결과 보고

효과크기

- t-test에서 효과 크기 (effect size) 란 두 집단의 평균 차이를 일정한 기준으로 표현한 것
- 효과 크기의 표현 방법에는 Cohen's d Pearson s r 등 여러 가지가있음
- 다른 실험의 효과 크기와 비교하여 상대적으로 판단한다.
  - 절대적인 것은 아니나 보통 0.2 정도면 작은 편 0.5 정도면 중간 0 80이면 큰 편

먼저 t test를 수행한다

```
dat_M = [117, 108, 105, 89, 101, 93, 96, 108, 108, 94, 93, 112, 92, 91, 100, 96, 120, 86, 96, 95]
dat_F = [121, 101, 102, 114, 103, 105, 101, 131, 96, 109, 109, 113, 115, 94, 108, 96, 110, 112, 120, 100]


import scipy.stats

m = scipy.stats.ttest_ind(dat_M, dat_F, equal_var=False)
```

Cohen's d

```
import numpy as np

t = m.statistic
df = len(dat_M) + len(dat_F) - 2

abs(t) / np.sqrt(df)
```

```
0.4332242888591059
```

Pearson's r

```
t2 = t ** 2

np.sqrt(t2 / (t2 + df))
```

```
0.39752319599996255
```

독립표본 t 검증 결과 보고

- 각 표본에 대한 기술통계(평균 표준편차 표본크기 등)
- t값과 그 때의 자유도 df
- p-value와 유의수준의 비교 (p value를 적시하지는않음)
- 효과크기 (주로 Cohen's d나 Pearson s'r)와 그에 대한 해석

> 집단 M과 F에 대하여 독립표본 t 검증을 실시한 결과 집단 M의 측정값 (M = 100, SD = 9.515) 은 집단 F의 측정값 (M = 108, SD = 9.431) 보다 통계적으로 유의미하게 낮았으며 (t (37 997) = −2.671, p < 0.05),
> 효과 크기는 중간 수준이었다(Cohen's d = 0.43, r = 0.39).

18. 대응표본 t 검증

대응표본이란 ?

- 두 집단의 자료를 쌍으로 묶을 수 있을 때
  - 예) 남편과 아내, 쌍둥이, before & after
- 두 집단의 자료를 쌍으로 묶어야 하기때문에, 독립표본과는 달리 두 집단의 자료 갯수가 동일해야 함

대응표본 t-test의 논리

- 쌍을 이루고있는 두 값의 차이를 구함
- 모집단에서 차이의 평균은 0이라고 귀무가설을 세움
- 그렇다면 표본에서도 차이의 평균은 0에 가까울 것
- 표본에서 구한 차이의 평균이 0보다 심각하게 크다면 귀무가설이 옳을 확률은 희박
- 그렇다면 귀무 가설을 기각하게 되고 두 집단 간 차이가 유의미하다고 말하는 것

대응표본 t-test 실시하기

`dat_M` 과 `dat_F` 가 순서대로 짝지어져 있다고 가정(예 117과 121, 108과 101 등등)

```
dat_M = [117, 108, 105, 89, 101, 93, 96, 108, 108, 94, 93, 112, 92, 91, 100, 96, 120, 86, 96, 95]
dat_F = [121, 101, 102, 114, 103, 105, 101, 131, 96, 109, 109, 113, 115, 94, 108, 96, 110, 112, 120, 100]
```

`paired=T` 를 하여 대응표본 t-test

```
import scipy.stats

scipy.stats.ttest_rel(dat_M, dat_F)
```

```
Ttest_relResult(statistic=-2.9868874599588247, pvalue=0.007578486289181322)
```

통계적으로 유의미한 차이가있음 ($p < 0.05$)

효과 크기의 계산과 결과 보고는 독립표본 t-검증과 동일