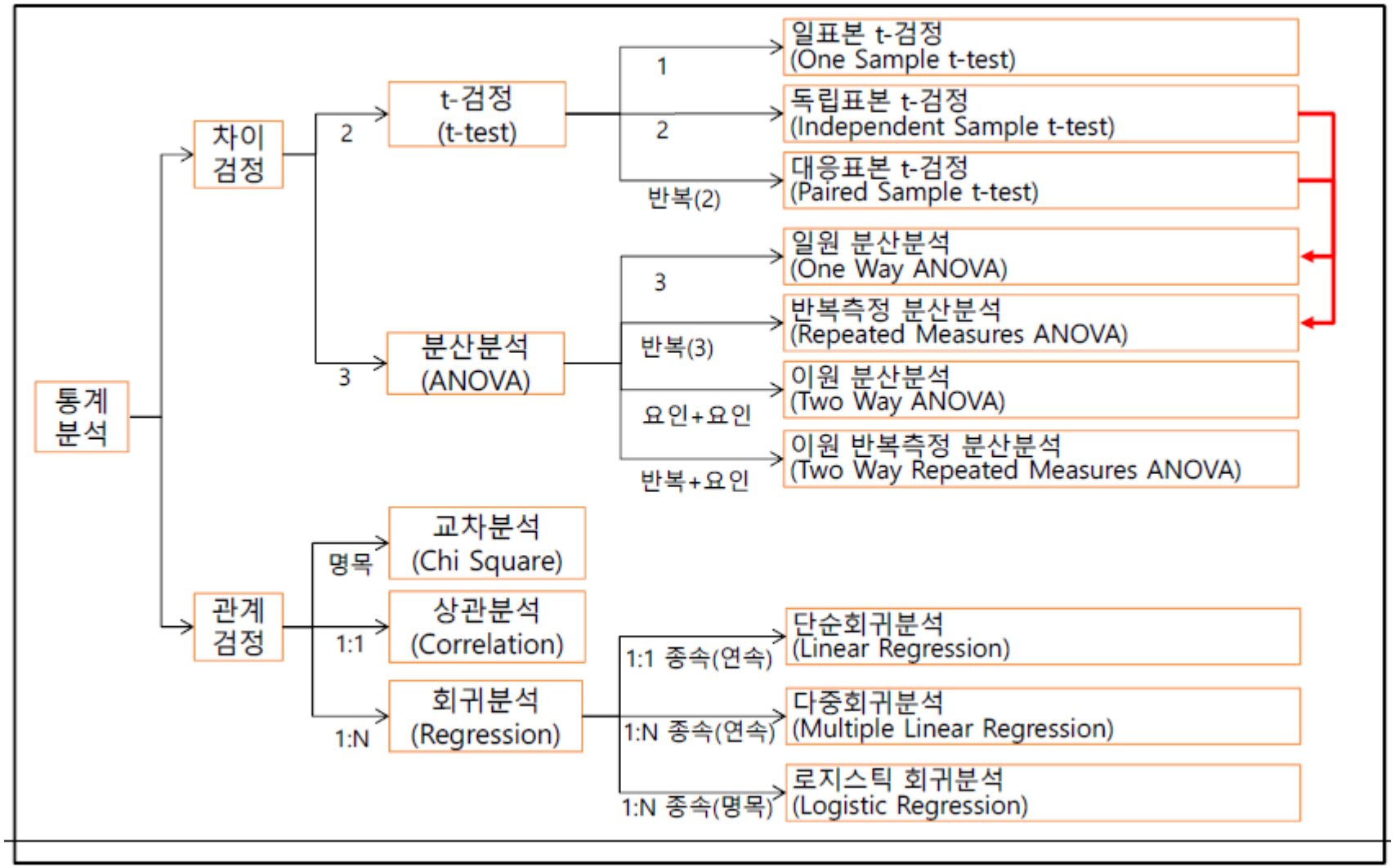
T검정(t-test)

보건빅데이터통계분석

이새봄
삼육대학교 SW융합교육원

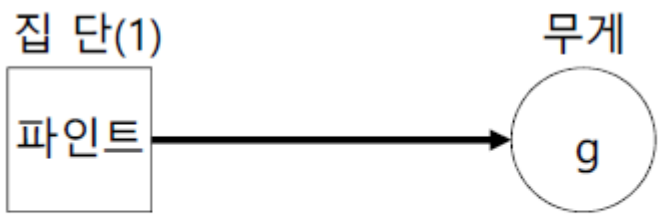
일표본 t-검정

통계분석방법


통계 분석
차이 검정
관계 검정
2
3
t-검정 (t-test)
분산분석 (ANOVA)
교차분석 (Chi Square)
상관분석 (Correlation)
회귀분석 (Regression)
명목
1:1
1:N
1
2
반복(2)
3
반복(3)
요인+요인
반복+요인
1:1 종속(연속)
1:N 종속(연속)
1:N 종속(명목)
일표본 t-검정 (One Sample t-test)
독립표본 t-검정 (Independent Sample t-test)
대응표본 t-검정 (Paired Sample t-test)
일원 분산분석 (One Way ANOVA)
반복측정 분산분석 (Repeated Measures ANOVA)
이원 분산분석 (Two Way ANOVA)
이원 반복측정 분산분석 (Two Way Repeated Measures ANOVA)
단순회귀분석 (Linear Regression)
다중회귀분석 (Multiple Linear Regression)
로지스틱 회귀분석 (Logistic Regression)

일표본 t-검정

- 문제의 정의
  - B아이스크림회사에서 판매하는 아이스크림 중 파인트의 무게는 320g이다.
  - 그러나 K대학 앞에 있는 점포에서 파는 아이스크림의 무게가 320g이 아니라는 소비자들의 불만이 있었다.
  - 이에 따라 소비자단체에서는 B아이스크림회사에서 만든 아이스크림이 320g인지를 검사하고자 한다



일표본 t-검정 (One Sample t-test)

	A
1	weight
2	319.3148
3	241.9693
4	290.9807
5	276.0801
6	347.5499
7	298.8435
8	292.8708
9	303.5787
10	296.1497
11	286.0313
12	301.4435
13	323.652
14	278.5676
15	322.7782
16	296.3468
17	270.8734

일표본 t-검정

- 문제의 정의
  - B아이스크림회사에서 판매하는 아이스크림 중 파인트의 무게는 320g이다.
  - 그러나 S대학 앞에 있는 점포에서 파는 아이스크림의 무게가 320g이 아니라는 소비자들의 불만이 있었다.
  - 이에 따라 소비자단체에서는 B아이스크림회사에서 만든 아이스크림이 320g인지를 검사하고자 한다.
  - (05.OST.csv)

    - 가설
      - 귀무가설(H_0): 파인트의 무게는 320g이다.

        $H_0: \mu = 320$

      - 연구가설(H_1): 파인트의 무게는 320g이 아니다.

        $H_1: \mu \neq 320$

일표본 t-검정

- 통계치
  - 표본 (n) : 100
  - 표본평균 ($\bar{X}$) : 295.44
  - 표본표준편차 (s): 20.04, 표준오차 ($\frac{s}{\sqrt{n}}$): 2.004
- 임계치

$$x_{critical} = \mu_0 \pm 1.984\frac{s}{\sqrt{n}} = 320 \pm 1.984\frac{20.04}{\sqrt{100}} = 320 \pm 3.97 = [316.02, 323.98]$$

- 검정통계량 (test statistics)

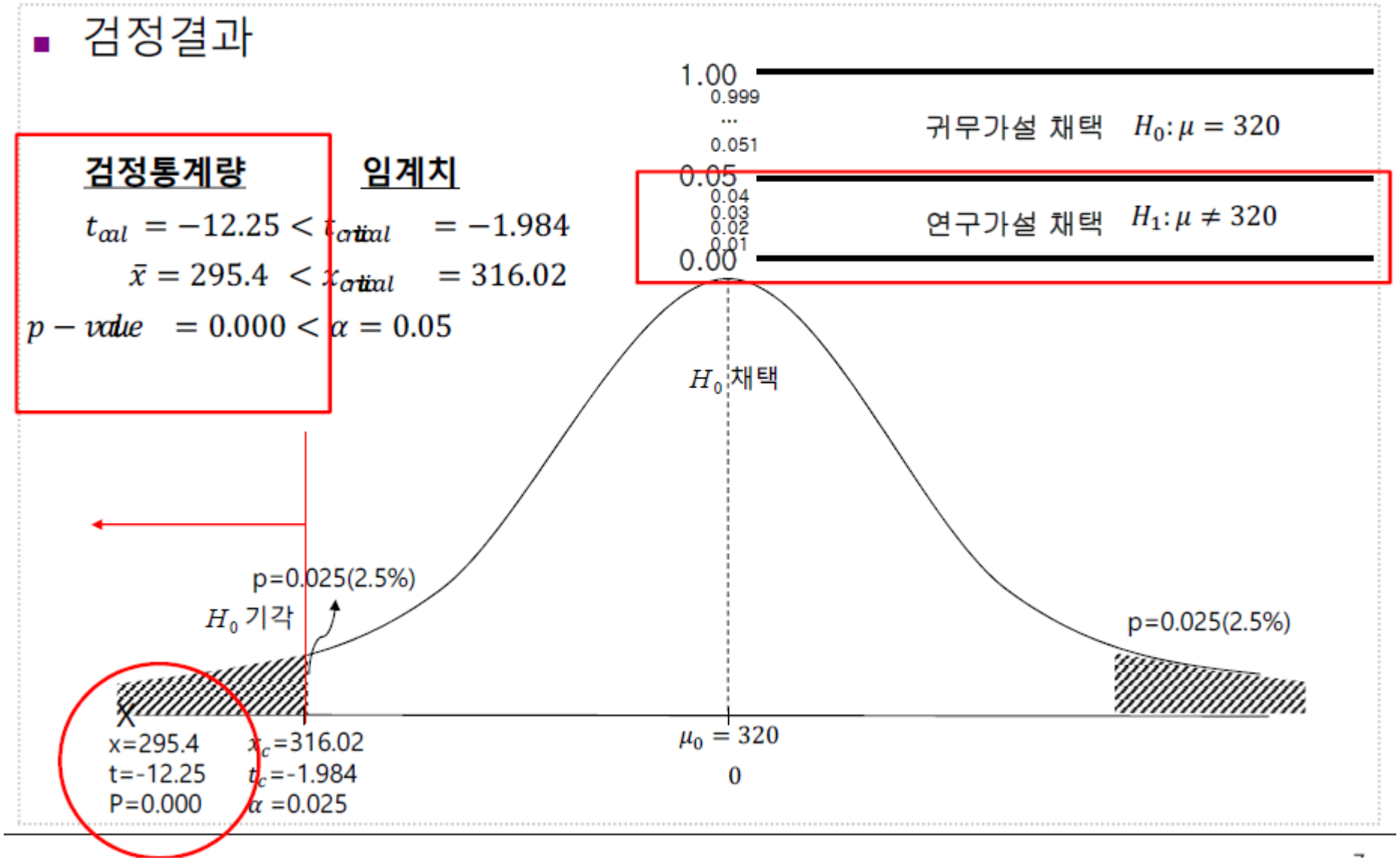
$$t_{cal} = \frac{\bar{x} - \mu_0}{\frac{s}{\sqrt{n}}} = \frac{295.4 - 320}{\frac{20.04}{\sqrt{100}}} = \frac{-24.6}{2.004} = -12.25$$

- 유의확률(p-value) 계산

$$p - value = 0.000$$

일표본 t-검정

- 검정결과


검정통계량
임계치
$t_{cal} = -12.25 < t_{critical} = -1.984$
$\bar{x} = 295.4 < x_{critical} = 316.02$
$p - value = 0.000 < \alpha = 0.05$
1.00
0.999
...
0.051
0.05
0.04
0.03
0.02
0.01
0.00
귀무가설 채택 $H_0: \mu = 320$
연구가설 채택 $H_1: \mu \neq 320$
H_0 채택
p=0.025(2.5%)
H_0 기각
p=0.025(2.5%)
x=295.4
t=-12.25
P=0.000
x_c=316.02
t_c=-1.984
α =0.025
$\mu_0 = 320$
0

일표본 t-검정

- 기본 패키지 설정

```python
# 그래프에서 한글 폰트 인식하기
!sudo apt-get install -y fonts-nanum
!sudo fc-cache -fv
!rm ~/.cache/matplotlib -rf
```

Python

```python
!pip install pingouin

#  *** 세션 다시 시작
```

Python

```python
# 1.기본
import numpy as np  # numpy 패키지 가져오기
import matplotlib.pyplot as plt # 시각화 패키지 가져오기
import seaborn as sns # 시각화

# 2.데이터 가져오기
import pandas as pd # csv -> dataframe으로 전환

# 3.통계분석 package
import pingouin as pg
from scipy import stats
import statsmodels.api as sm
```

Python

일표본 t-검정

```python
# 기본세팅
# 테마 설정
sns.set_theme(style = "darkgrid")

# 한글 인식
plt.rc('font', family='NanumBarunGothic')
plt.rcParams['axes.unicode_minus'] = False # -인식
```

Python

일표본 t-검정

- 데이터 불러오기 및 탐색

```python
ost_df = pd.read_csv('04_1.OST.csv', encoding="cp949")
ost_df.head()
```

Python

```python
ost_df.shape
```

Python

```
(102, 6)
```

```python
ost_df.info()
```

Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 102 entries, 0 to 101
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   무게1     102 non-null    float64
 1   무게2     102 non-null    float64
 2   무게3     102 non-null    float64
 3   무게4     102 non-null    float64
 4   무게5     102 non-null    float64
 5   무게6     102 non-null    float64
dtypes: float64(6)
memory usage: 4.9 KB
```

일표본 t-검정

- 기술통계분석
  - 수치형 기술통계량

각 열에 대한 count(개수), mean(평균), std(표준편차), min(최소값), 25%(1사분위수), 50%(중앙값), 75%(3사분위수), max(최대값)을 확인

```python
ost_df.columns
```

Python

```
Index(['무게1', '무게2', '무게3', '무게4', '무게5', '무게6'], dtype='object')
```

```python
# 수치형 변수
ost_df.describe().round(2).T
```

Python

```python
ost_df.agg({"무게1": ["count", "mean", "std", "min", "max", "median", "skew", "kurtosis"]}).T \
    .round(2)
```

Python

일표본 t-검정

T-검정 수행

- scipy.stats의 ttest_1samp 함수를 사용하여 단일표본 t-검정을 수행
  - ost_df["무게1"]: 검정할 데이터(표본)
  - popmean = 320: 비교할 모집단 평균(귀무가설에서의 값)
  - alternative = "two-sided": 양측검정 수행(귀무가설: 표본 평균 = 320)
- 결과로 검정통계량(t-value)과 p-값을 제공

```python
# scipy.stats.ttest_1samp
stats.ttest_1samp(ost_df["무게1"], popmean = 320, alternative = "two-sided")
```

Python

```
TtestResult(statistic=-2.899472691059131, pvalue=0.004586364436777763, df=101)
```

```python
# two-sided
pg.ttest(ost_df["무게1"], 320, alternative = "two-sided").round(3)

# 5.정규분포 가정 검정후 다시 분석
```

Python

- T: 검정통계량
- dof: 자유도
- alternative: 검정 유형(양측/단측)
- p-val: p-값
- CI95%: 95% 신뢰구간
- cohen-d: 효과크기(Cohen's d)
- BF10: 베이즈 팩터
- power: 검정력

일표본 t-검정

- 검정 유형 비교: 양측 vs 단측
  - •양측검정(two-sided): 표본 평균이 모집단 평균과 다른지 검정(μ ≠ 320)
  - •단측검정(less): 표본 평균이 모집단 평균보다 작은지 검정(μ < 320)

```python
# two-sided
pg.ttest(ost_df["무게2"], 320, alternative = "two-sided").round(4)
```

Python

```python
# less
pg.ttest(ost_df["무게2"], 320, alternative = "less").round(3)
```

Python

```python
# two-sided
pg.ttest(ost_df["무게3"], 320, alternative = "two-sided").round(4)
```

Python

```python
# less
pg.ttest(ost_df["무게3"], 320, alternative = "less").round(3)
```

Python

일표본 t-검정

```python
# two-sided
pg.ttest(ost_df["무게4"], 320, alternative = "two-sided").round(3)
```

Python

```python
# greater
pg.ttest(ost_df["무게4"], 320, alternative = "greater").round(3)
```

Python

일표본 t-검정

- 정규성 검정
  - Shapiro-Wilk 검정을 사용하여 데이터의 정규성을 검정

```python
pg.normality(ost_df["무게1"])
```

Python

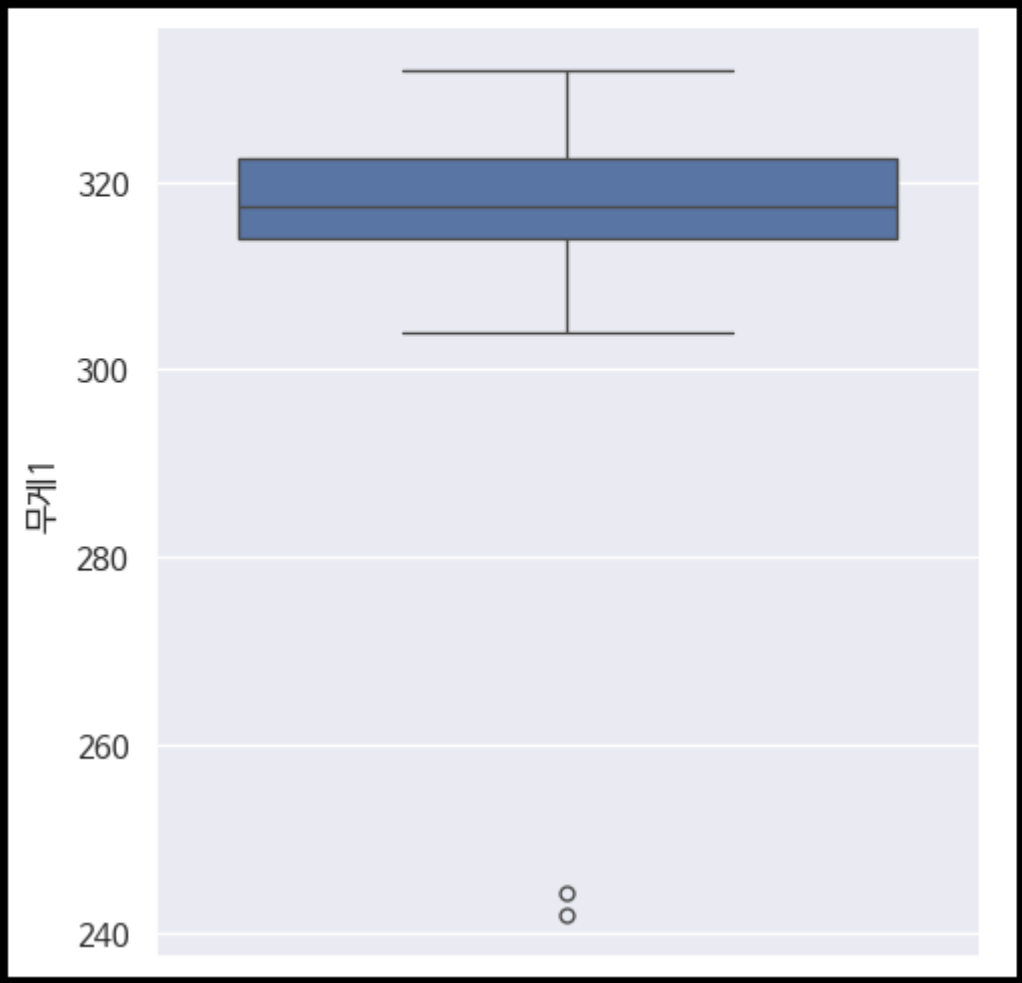
```python
pg.normality(ost_df).T.round(3)
```

Python

일표본 t-검정

- 상자도표로 이상치 확인

```
# 한글 폰트 인식
sns.catplot(data = ost_df,
            y = "무게1",
            kind = "box")
plt.show()
```



일표본 t-검정

- 이상치 식별 및 제거
  - 무게가 250 이하인 값을 이상치로 식별
  - 해당 이상치들을 확인
  - drop() 함수를 사용하여 이상치를 데이터셋에서 제거
- 이상치 제거 후 데이터 정규성 다시 확인

```
filter = (ost_df["무게1"] <= 250)
ost_df.loc[filter]
```

```
ost_df.drop(ost_df[filter].index, inplace = True)
```

```
pg.normality(ost_df).T.round(3)

# 4.1로 다시 분석
```

일표본 t-검정

- 비모수 통계 검정
  - 정규성 가정이 충족되지 않을 때 사용할 수 있는 비모수적 방법인 Wilcoxon 부호순위 검정을 수행
  - t-검정(모수적 방법)과 Wilcoxon 검정(비모수적 방법)의 결과를 비교.
  - 두 방법 모두 유의한 결과를 보이지만, 검정통계량과 p-값이 다름

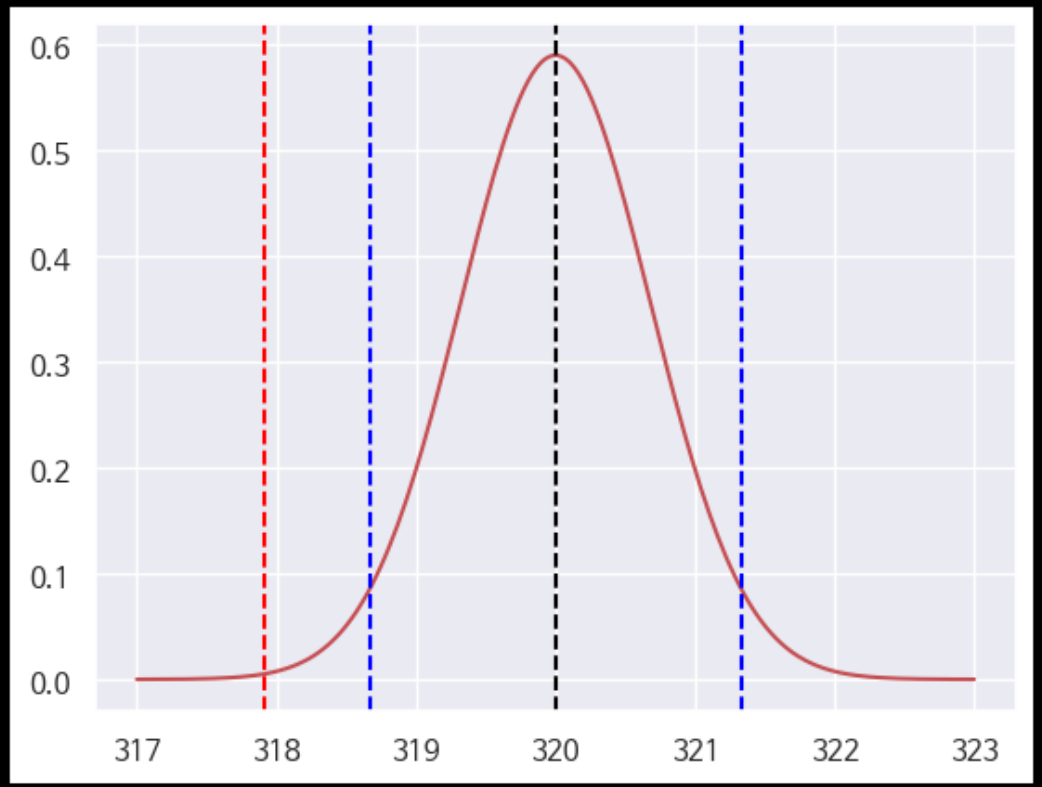
```
# 비모수통계
pg.wilcoxon(ost_df["무게6"] - 320, alternative = "two-sided").round(3)
```

```
# 모수통계 결과와 비교
pg.ttest(ost_df["무게6"], 320, alternative = "two-sided").round(3)
```

일표본 t-검정

- 검증결과 시각화
  - 모집단 평균(mu=320)을 중심으로 한 정규분포 곡선을 그림
  - 모집단 평균(검은색 점선)을 표시95% 신뢰구간의 경계(파란색 점선)를 표시
  - 표본 평균(빨간색 점선)을 표시

```
from scipy.stats import norm # 정규분포

x_data = np.linspace(317, 323, 200)

mu = 320   # 모집단 평균
x = 317.91 # 표본평균
se = 6.77/np.sqrt(100) # 표준오차(표준편차/sqrt(n)))

plt.plot(x_data, norm.pdf(x_data, loc = mu, scale = se), 'r-')
plt.axvline(x = mu, color='black', linestyle='--')
plt.axvline(x = mu - 1.96 * se, color='blue', linestyle='--')
plt.axvline(x = mu + 1.96 * se, color='blue', linestyle='--')
plt.axvline(x = x, color='red', linestyle='--')
plt.show()
```



일표본 t-검정

- 단일 모집단 비율검정
  - One sample t test of proportion

```python
# One Sample T Test of Proportion
from statsmodels.stats.proportion import proportions_ztest

z, p = proportions_ztest(count = 50,
                         nobs = 500,
                         value = 0.09)
print('z : {}, p : {}'.format(z, p))
```

Python

```
z : 0.7453559924999305, p : 0.45605654025025566
```

```python
# 이항분포로 검정 n*p < 5 일때
stats.binom_test([50, 450], p = 0.09, alternative="two-sided")
```

Python

```
<ipython-input-30-1e62f8be38b0>:2: DeprecationWarning: 'binom_test' is deprecated in favour of 'binomtest' from version 1.7.0
  stats.binom_test([50, 450], p = 0.09, alternative="two-sided")
```

일표본 t-검정

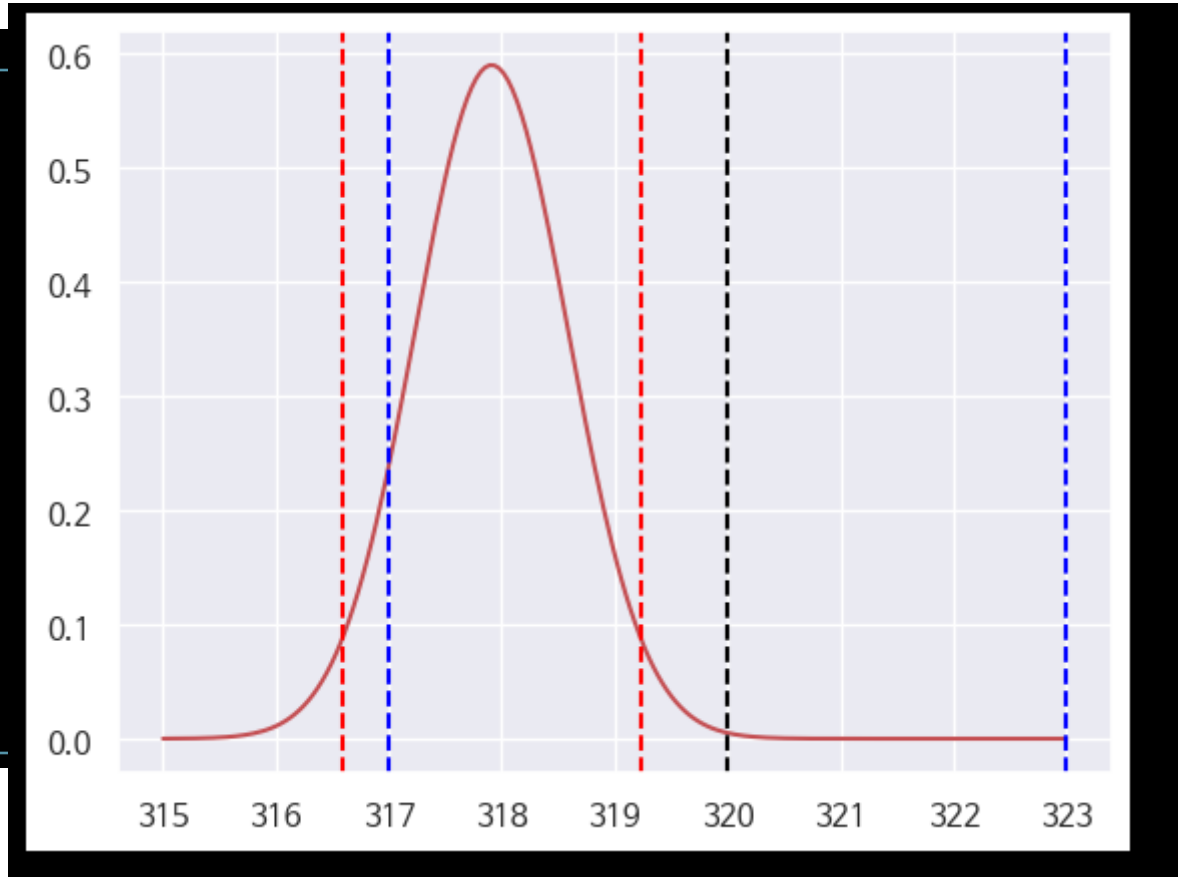
- 동등성 검정
  - 동등성 검정은 전통적인 가설검정과 다르게, 차이가 없다는 것을 증명
  - 전통적 검정: 차이가 있는지 검정 (귀무가설: 차이 없음)
  - 동등성 검정: 차이가 사전에 정의된 범위 내에 있는지 검정(귀무가설: 차이가 범위를 벗어남)
  - ound = 3은 동등성 범위를 ±3으로 설정
  - p-값이 0.05보다 작으면 두 집단이 동등하다고 결론

```
pg.tost(ost_df["무게1"],
        y = 320,
        bound = 3)
```

```
pg.tost(ost_df["무게4"],
        y = 320,
        bound = 3)
```

일표본 t-검정

```
from scipy.stats import norm # 정규분포

x_data = np.linspace(315, 323, 200)

mu = 320   # 모집단 평균
eb = 3
x = 317.91 # 표본평균
se = 6.77/np.sqrt(100) # 표준오차(표준편차/sqrt(n)))

plt.plot(x_data, norm.pdf(x_data, loc = x, scale = se), 'r-')
plt.axvline(x = mu, color='black', linestyle='--')
plt.axvline(x = x - 1.96 * se, color='red', linestyle='--')
plt.axvline(x = x + 1.96 * se, color='red', linestyle='--')
plt.axvline(x = mu - 3, color='blue', linestyle='--')
plt.axvline(x = mu + 3, color='blue', linestyle='--')
plt.show()
```



1.표본 평균(x=317.91)을 중심으로 한 정규분포 곡선
2.모집단 평균(검은색 점선)
3.표본 평균의 95% 신뢰구간(빨간색 점선)
4.동등성 범위 경계(파란색 점선, mu±3)
표본 평균의 신뢰구간이 동등성 범위 내에 완전히 포함되어 있다면, 두 집단은 통계적으로 동등하다고 결론

일표본 t-검정

- S대학 앞 점포에서 파는 아이스크림의 무게(316.4g)는 B아이스크림 회사에서 발표한 파인트의 무게(320g)보다 통계적으로 유의하게 적었다. (t=-3.087, p=0.003)

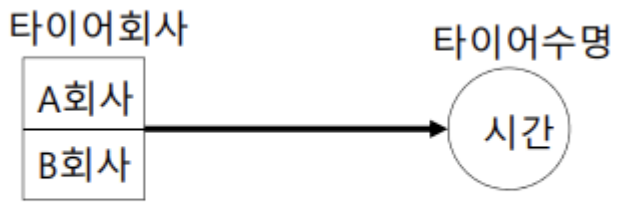
변수	M(SD)	t	p
무게1	317.91(12.39)	-3.087	0.003
무게2	317.13(12.58)	-1.993	0.049
무게3	317.14(12.58)	-	-
무게4	319.35(12.77)	1.286	0.202
무게5	321.35(12.98)	4.295	<0.001
무게6*	315.97(12.33)	W=1377.0	<0.001

*주: 무게6은 정규성 가정을 만족하지 않아 비모수적 방법(Wilcoxon 부호 순위 검정)을 적용함

독립표본 t-검정

독립표본 t-검정

- 문제의 정의
  - 이교수는 이번에 자동차 타이어를 교체하려고 하는데 수 명이 긴 타이어로 교체하려고 한다.
  - 시중에는 A회사의 타이어와 B회사의 타이어가 있는데, 이 교수는 이 중에서 어느 타이어를 골라야 하는가?



독립표본 t-검정(Independent Sample t-test)

	A	B
1	t_group	t_time
2	1	48187
3	2	47245
4	1	51020
5	1	50732
6	2	52416
7	2	49278
8	1	38214
9	1	46742
10	1	48706
11	2	54280
12	1	50635
13	1	51052
14	2	51568
15	1	51569
16	1	48825
17	1	49674

독립표본 t-검정

- 문제의 정의
  - 이교수는 이번에 자동차 타이어를 교체하려고 하는데 수명이 긴 타이어로 교체하려고 한다.
  - 시중에는 A회사의 타이어와 B회사의 타이어가 있는데, 이 교수는 이 중에서 어느 타이어를 골라야 하는가? (05_1.IST.csv).

- 가설
  - 귀무가설(H_0): A타이어회사와 B타이어회사의 타이어수명은 차이가 없다.

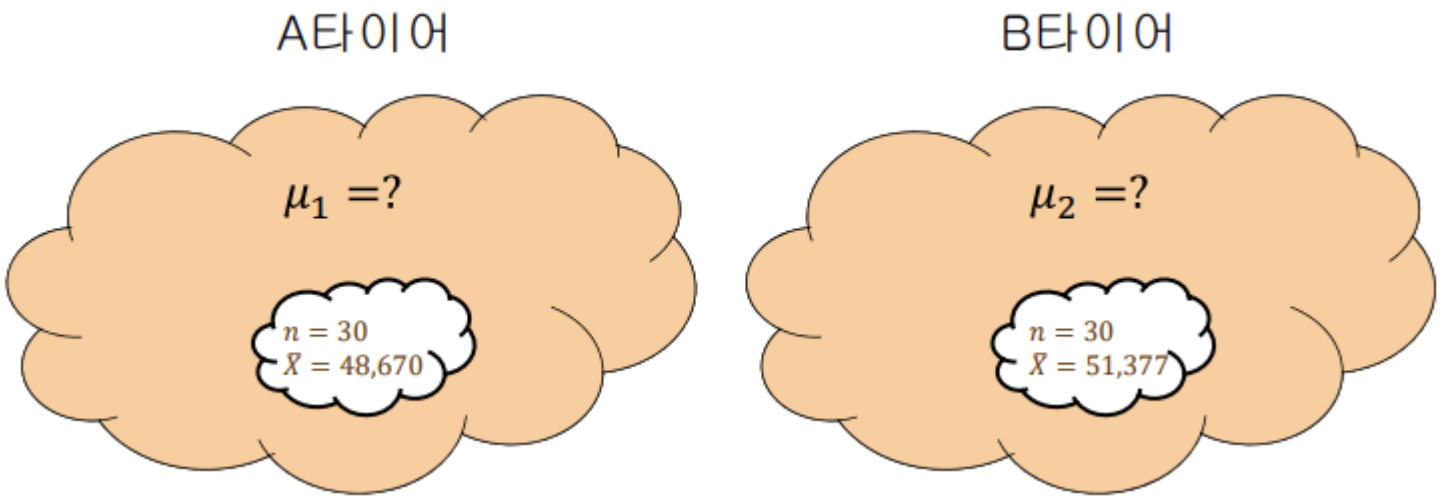
    $H_0: \mu_1 = \mu_2$ $\quad$ $H_0: \mu_1 - \mu_2 = 0$

  - 연구가설(H_1): A타이어회사와 B타이어회사의 타이어수명은 차이가 있다.

    $H_1: \mu_1 \neq \mu_2$ $\quad$ $H_1: \mu_1 - \mu_2 \neq 0$

독립표본 t-검정

- 모집단에서 표본추출



독립표본 t-검정

```python
# 그래프에서 한글 폰트 인식하기
!sudo apt-get install -y fonts-nanum
!sudo fc-cache -fv
!rm ~/.cache/matplotlib -rf
```

Python

```python
!pip install pingouin

#  *** 세션 다시 시작
```

Python

```python
# 1.기본
import numpy as np  # numpy 패키지 가져오기
import matplotlib.pyplot as plt # 시각화 패키지 가져오기
import seaborn as sns # 시각화

# 2.데이터 가져오기
import pandas as pd # csv -> dataframe으로 전환

# 3.통계분석 package
import pingouin as pg
from scipy import stats
import statsmodels.api as sm
```

Python

```python
# 기본세팅
# 테마 설정
sns.set_theme(style = "darkgrid")

# 한글 인식
plt.rc('font', family='NanumBarunGothic')
plt.rcParams['axes.unicode_minus'] = False # -인식
```

Python

독립표본 t-검정

```python
ist_df = pd.read_csv('05_1.IST.csv', encoding="cp949")
ist_df.head()
```

Python

```python
ist_df['회사'].replace({1:'A타이어', 2:'B타이어',}, inplace=True)
ist_df['회사'] = ist_df['회사'].astype('category')

ist_df.head()
```

Python

독립표본 t-검정

```python
ist_df.shape
```

```
(66, 6)
```

```python
ist_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 66 entries, 0 to 65
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   회사      66 non-null     category
 1   수명1     66 non-null     int64
 2   수명2     66 non-null     int64
 3   수명3     66 non-null     int64
 4   수명4     66 non-null     int64
 5   수명5     66 non-null     int64
dtypes: category(1), int64(5)
memory usage: 2.9 KB
```

```python
ist_df.columns
```

```
Index(['회사', '수명1', '수명2', '수명3', '수명4', '수명5'], dtype='object')
```

독립표본 t-검정

```python
# 그룹별 기술통계
ist_df.groupby('회사')["수명1"].describe().round(2)
```

```python
# 분석변수가 여러개 일 때
num_feature = ['수명1', '수명2', '수명3', '수명4', '수명5']
for num in num_feature:
    print("----", num, "----")
    results = ist_df.groupby('회사')[num].describe().round(2)
    print(results, "\n")
```

```
---- 수명1 ----
      count   mean   std   min   25%   50%   75%   max
회사
A타이어   31.0  48.94  3.33  42.0  47.0  49.0  51.0  56.0
B타이어   35.0  51.69  3.77  44.0  50.0  52.0  55.0  59.0

---- 수명2 ----
      count   mean   std   min   25%   50%   75%   max
회사
A타이어   31.0  50.94  3.33  44.0  49.0  51.0  53.0  58.0
B타이어   35.0  51.69  3.77  44.0  50.0  52.0  55.0  59.0

---- 수명3 ----
      count   mean   std   min   25%   50%   75%   max
회사
A타이어   31.0  50.03  3.19  42.0  48.0  50.0  52.0  57.0
B타이어   35.0  51.69  3.77  44.0  50.0  52.0  55.0  59.0

---- 수명4 ----
      count   mean   std   min   25%   50%   75%   max
회사
A타이어   31.0  48.71  1.97  44.0  47.0  49.0  50.0  52.0
B타이어   35.0  51.69  3.77  44.0  50.0  52.0  55.0  59.0

---- 수명5 ----
...
회사
A타이어   31.0  50.35  5.26  40.0  48.0  50.0  52.0  61.0
B타이어   35.0  51.69  5.78  40.0  50.0  52.0  55.0  63.0

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

독립표본 t-검정

```python
x = ist_df['수명1'][ist_df['회사'] == 'A타이어']
y = ist_df['수명1'][ist_df['회사'] == 'B타이어']
```

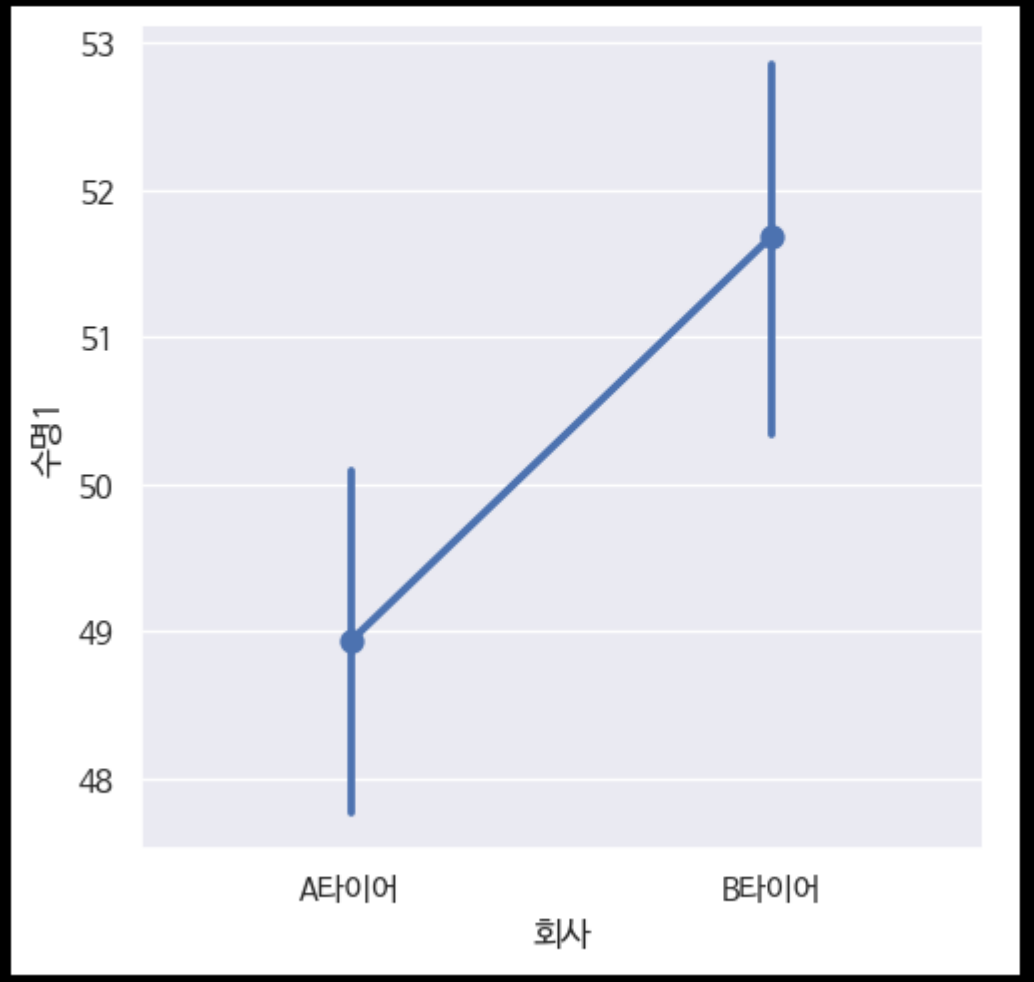
Python

```python
# paired = True : paired sample t-test
# correction = False : 등분산일때
pg.ttest(x, y,
         paired = False,
         alternative = "two-sided",
         correction = False).round(3)
```

Python

독립표본 t-검정

```
# 그래프
sns.catplot(x = "회사",
            y = "수명1",
            kind = "point",
            data = ist_df)
plt.show()
```



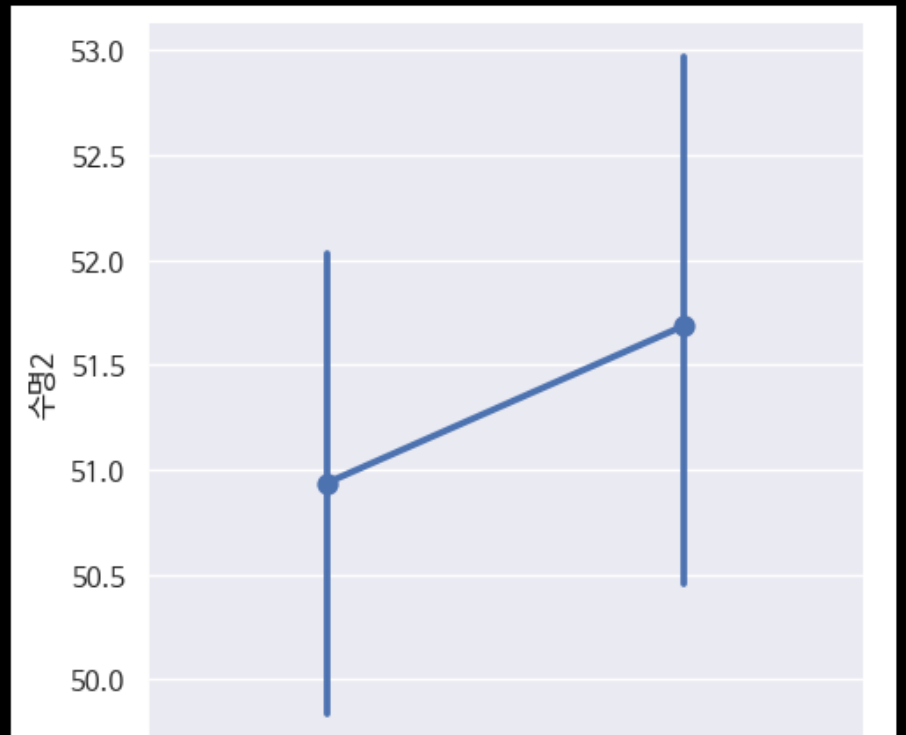
독립표본 t-검정

```python
x = ist_df['수명2'][ist_df['회사'] == 'A타이어']
y = ist_df['수명2'][ist_df['회사'] == 'B타이어']

pg.ttest(x, y,
         paired = False,
         alternative = "two-sided",
         correction = False).round(3)
```

Python

```python
# 그래프
sns.catplot(x = "회사",
            y = "수명2",
            kind = "point",
            data = ist_df)
plt.show()
```

Python



독립표본 t-검정

```python
# two-sided
x = ist_df['수명3'][ist_df['회사'] == 'A타이어']
y = ist_df['수명3'][ist_df['회사'] == 'B타이어']

pg.ttest(x, y,
         paired = False,
         alternative = "two-sided",
         correction = False).round(3)
```

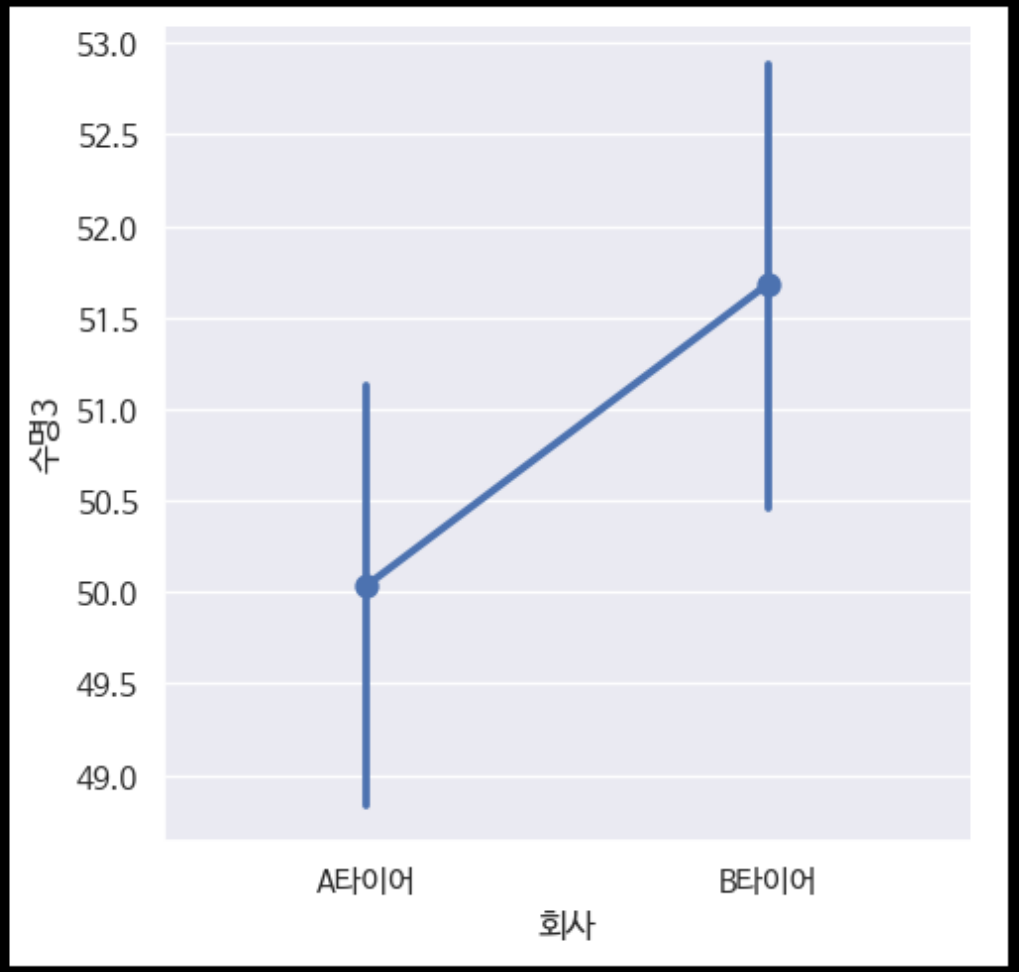
Python

```python
# less
pg.ttest(x, y,
         paired = False,
         alternative = "less",
         correction = False).round(3)
```

Python

독립표본 t-검정

```
# 그래프
sns.catplot(x = "회사",
            y = "수명3",
            kind = "point",
            data = ist_df)
plt.show()
```



독립표본 t-검정

```
# 등분산이면 지금까지 분석한 것이 문제 없음
pg.homoscedasticity(ist_df,
                    dv = "수명1",
                    group = "회사")
```

```
num_feature = ['수명1', '수명2', '수명3', '수명4', '수명5']
for num in num_feature:
    print("----", num, "----")
    results = pg.homoscedasticity(ist_df,
                                  dv = num,
                                  group = "회사")
    print(results, "\n")
```

```
---- 수명1 ----
               W      pval  equal_var
levene  0.195988  0.659471       True

---- 수명2 ----
               W      pval  equal_var
levene  0.195988  0.659471       True

---- 수명3 ----
               W      pval  equal_var
levene  0.400108  0.529287       True

---- 수명4 ----
              W      pval  equal_var
levene  7.02041  0.010141      False

---- 수명5 ----
               W      pval  equal_var
levene  0.072567  0.788502       True
```

독립표본 t-검정

```python
pg.homoscedasticity(ist_df,
                    dv = "수명4",
                    group = "회사")
```

Python

```python
x = ist_df['수명4'][ist_df['회사'] == 'A타이어']
y = ist_df['수명4'][ist_df['회사'] == 'B타이어']
```

Python

```python
# paired = True : paired sample t-test
# correction = True : 이분산일때
pg.ttest(x, y,
         paired = False,
         alternative = "two-sided",
         correction = True).round(3) # 이분산일때
```

Python

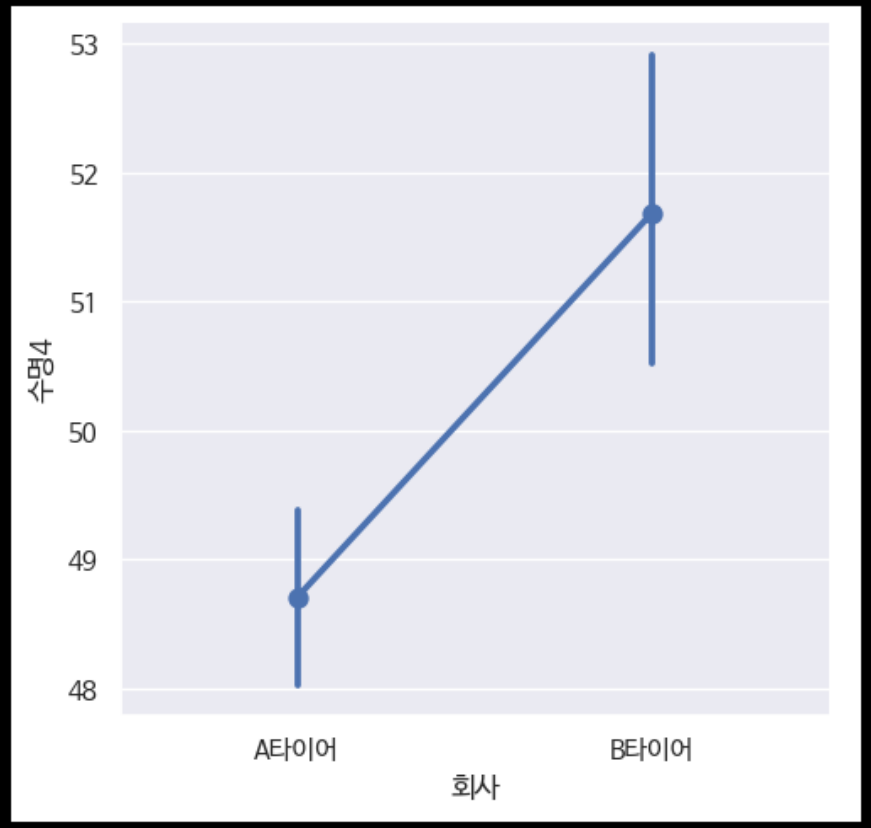
독립표본 t-검정

```
# correction = False : 등분산일때와 비교
pg.ttest(x, y,
         paired = False,
         alternative = "two-sided",
         correction = False).round(3)
```

Pytho

```
# 그래프
sns.catplot(x = "회사",
            y = "수명4",
            kind = "point",
            data = ist_df)
plt.show()
```

Pytho



독립표본 t-검정

```python
pg.normality(ist_df,
             dv = '수명1',
             group = '회사')
```

Python

```python
num_feature = ['수명1', '수명2', '수명3', '수명4', '수명5']
for num in num_feature:
    print("----", num, "----")
    results = pg.normality(ist_df,
                           dv = num,
                           group='회사')
    print(results, "\n")
```

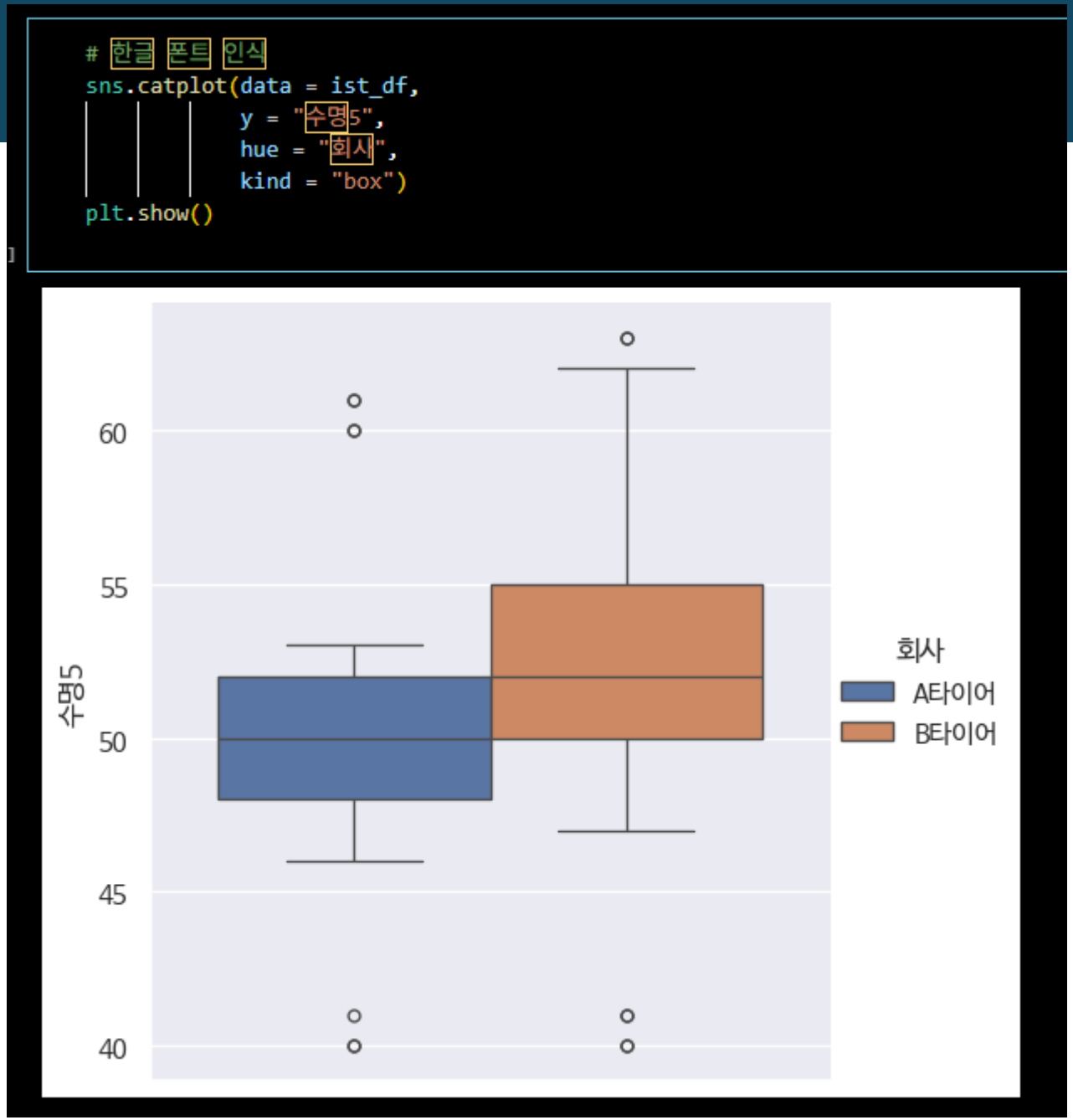
Python

독립표본 t-검정

```
---- 수명1 ----
              W      pval  normal
회사
A타이어  0.976345  0.705439    True
B타이어  0.959988  0.228475    True

---- 수명2 ----
              W      pval  normal
회사
A타이어  0.976345  0.705439    True
B타이어  0.959988  0.228475    True

---- 수명3 ----
              W      pval  normal
회사
A타이어  0.977802  0.749155    True
B타이어  0.959988  0.228475    True

---- 수명4 ----
              W      pval  normal
회사
A타이어  0.930326  0.044749   False
B타이어  0.959988  0.228475    True

---- 수명5 ----
...
회사
A타이어  0.917587  0.020380   False
B타이어  0.914812  0.010135   False

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

독립표본 t-검정

```
# 한글 폰트 인식
sns.catplot(data = ist_df,
            y = "수명5",
            hue = "회사",
            kind = "box")
plt.show()
```



독립표본 t-검정

```python
pg.normality(ist_df,
             dv = '수명5',
             group = '회사')
```

Python

```python
x = ist_df['수명5'][ist_df['회사'] == 'A타이어']
y = ist_df['수명5'][ist_df['회사'] == 'B타이어']
```

Python

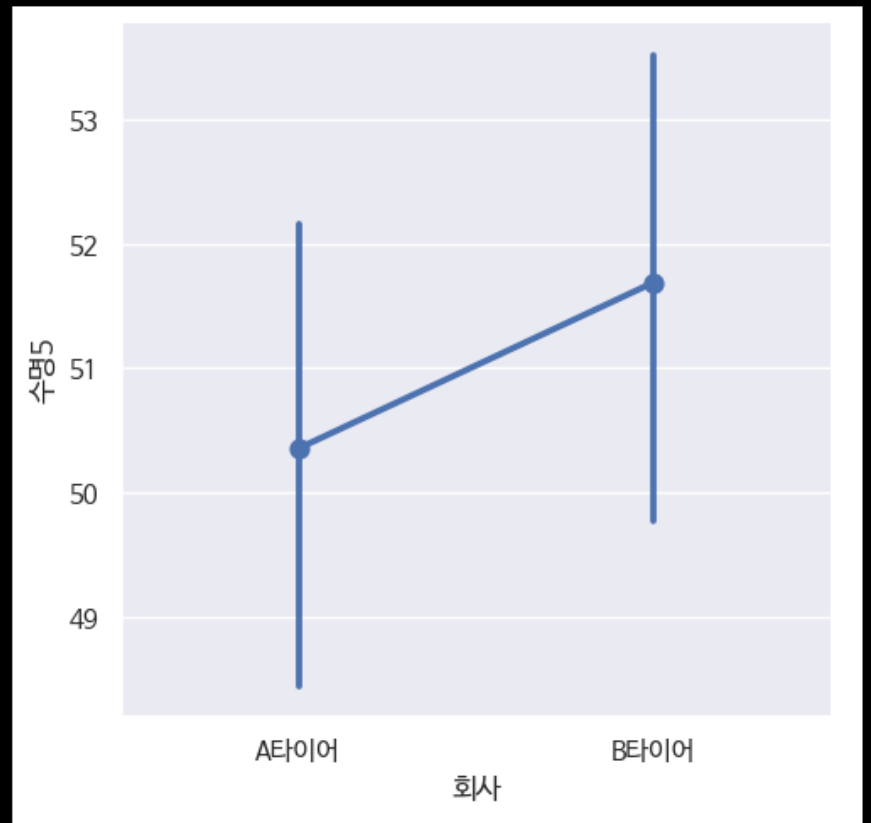
```python
# Mann-Whitney U test
pg.mwu(x, y,
       alternative='two-sided')
```

Python

독립표본 t-검정

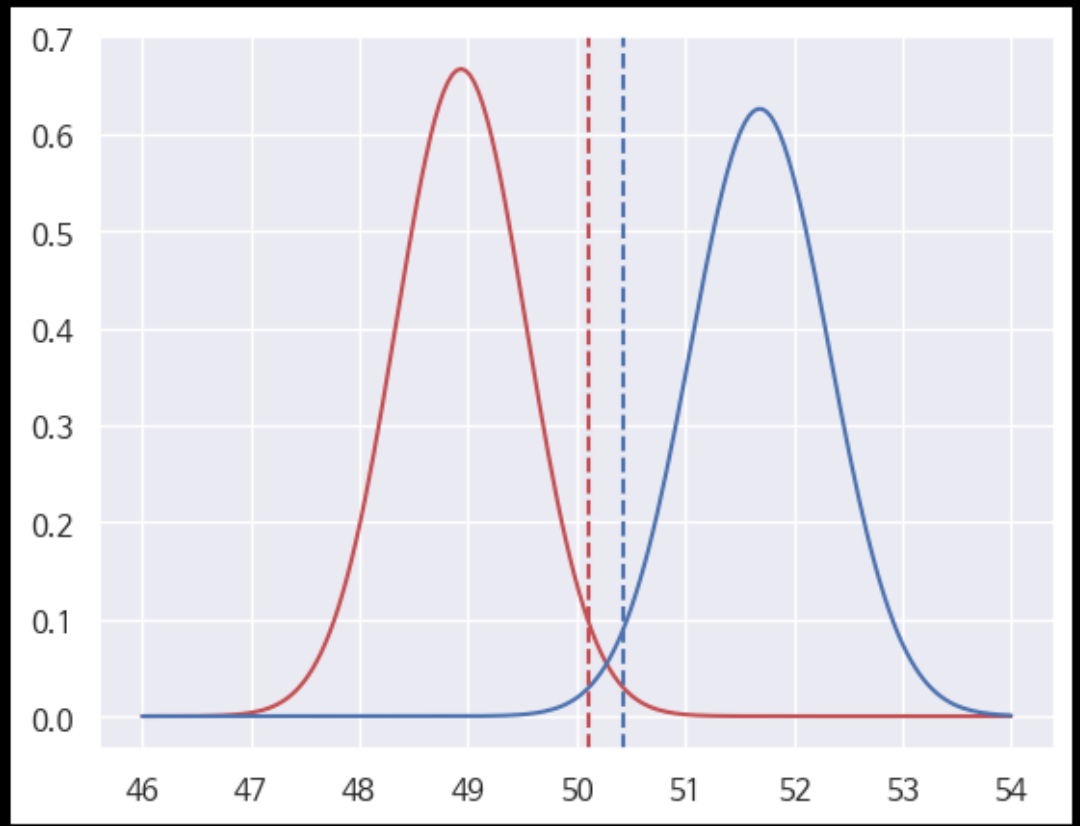
```
# 모수통계(t-test)와 비교
pg.ttest(x, y,
         paired = False,
         alternative = "two-sided",
         correction = True).round(3)
```

```
# 그래프
sns.catplot(x = "회사",
            y = "수명5",
            kind = "point",
            data = ist_df)
plt.show()
```



독립표본 t-검정

```
from scipy.stats import norm # 정규분포

x_data = np.linspace(46, 54, 200)

x1 = 48.935
x2 = 51.686
se1 = 3.33/np.sqrt(31) # 표준오차(표준편차/sqrt(n)))
se2 = 3.77/np.sqrt(35) # 표준오차(표준편차/sqrt(n)))

plt.plot(x_data, norm.pdf(x_data, loc = x1, scale = se1), 'r-')
plt.plot(x_data, norm.pdf(x_data, loc = x2, scale = se2), 'b-')
plt.axvline(x = x1+1.96 * se1, color='r', linestyle='--')
plt.axvline(x = x2-1.96 * se2, color='b', linestyle='--')
plt.show()
```



독립표본 t-검정

```
from statsmodels.stats.proportion import proportions_ztest

count = np.array([87, 671])      # x1, x2
nobs = np.array([2065, 27949])   # n1, n2

z, p = proportions_ztest(count = count,
                         nobs = nobs,
                         value = 0)
print('z : {}, p : {}'.format(z, p))
```

Python

```
z : 5.065085626514842, p : 4.0821681951628293e-07
```

```
# chi-square test로 분석한 결과
count = np.array([87, 2065-87])      # x1, x2
nobs = np.array([671, 27949-671])    # n1, n2

tab = [count, nobs]
result = sm.stats.Table(tab)
rslt = result.test_nominal_association()
print(rslt)
```

Python

```
df          1
pvalue      4.0821681956959566e-07
statistic   25.65509240392725
```

```
# z값과 비교
np.sqrt(rslt.statistic)
```

Python

```
5.065085626514842
```

독립표본 t-검정

```python
x = ist_df['수명1'][ist_df['회사'] == 'A타이어']
y = ist_df['수명1'][ist_df['회사'] == 'B타이어']
```

Python

```python
pg.tost(x, y,
        bound = 2,
        paired = False)
```

Python

```python
x = ist_df['수명2'][ist_df['회사'] == 'A타이어']
y = ist_df['수명2'][ist_df['회사'] == 'B타이어']
```

Python

```python
pg.tost(x, y,
        bound = 2,
        paired = False)
```

Python

독립표본 t-검정

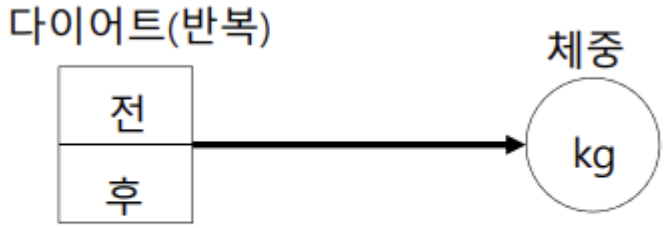
- A타이어회사의 타이어수명(M=48,671)과 B타이어회사의 타이어수명(M=51,378)간에는 통계적으로 유의한 차이가 있었으며, B타이어회사의 타이어수명이 더 높게 나타났다(t=-2.92, p=0.005).

	A타이어회사 (n=30)	A타이어회사 (n=30)	t	Sig
타이어수명	48,671	51,378	-2.92	0.005

대응표본 t-검정

대응표본 t-검정

- 문제의 정의
  - K제약회사의 신제품 개발부서에서는 3개월 안에 살이 빠 지는 다이어트 약을 개발하였다.
  - 회사 경영진에게 새롭게 개발한 다이어트약이 효과가 있는 지를 보고하기 위하여 약의 효능을 검증하였다.
  - 약을 먹기 전의 체중과 약을 먹은 후 3개월 후의 체중을 조 사하였다.
  - 과연 새로운 약은 다이어트에 효과가 있는가?

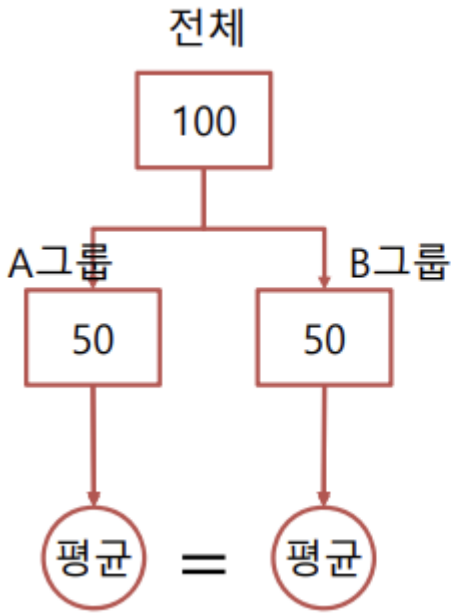


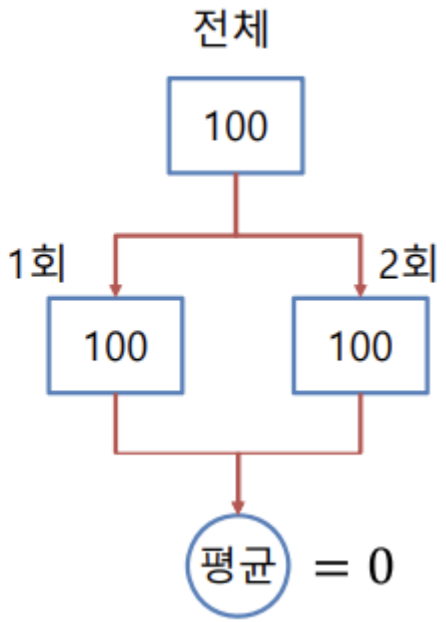
대응표본 t-검정 (Paired Sample t-test)

	A	B
1	pre	post
2	82	75
3	54	50
4	74	74
5	75	71
6	71	69
7	76	73
8	70	68
9	62	62
10	77	68
11	75	72
12	72	70
13	83	77
14	78	71
15	74	74
16	68	67
17	76	73

대응표본 t-검정

- 독립표본과 대응표본의 차이점
  - 독립표본 : 대상에서 1번만 측정
  - 대응표본 : 동일 대상에서 반복해서 측정





대응표본 t-검정

- 문제의 정의
  - K제약회사의 신제품 개발부서에서는 3개월 안에 살이 빠 지는 다이어트 약을 개발하였다.
  - 회사 경영진에게 새롭게 개발한 다이어트약이 효과가 있는 지를 보고하기 위하여 약의 효능을 검증하였다.
  - 약을 먹기 전의 체중과 약을 먹은 후 3개월 후의 체중을 조 사하였다.
  - 과연 새로운 약은 다이어트에 효과가 있는가? (07.PST.csv)

- 가설1
  - 귀무가설 (H_0): 다이어트약을 먹기 전과 후의 체중은 변화가 없다.

    $H_0: \mu_d = 0$

  - 연구가설 (H_1): 다이어트약을 먹기 전과 후의 체중은 변화가 있다.

    $H_1: \mu_d \neq 0$

대응표본 t-검정

반복 1(x_{i1})	반복 2(x_{i2})	$\bar{d} = x_{i1} - x_{i2}$
x_{11} x_{21} ⋮ x_{n1}	x_{12} x_{22} ⋮ x_{n2}	$\bar{d} = x_{11} - x_{12}$ $\bar{d} = x_{21} - x_{22}$ ⋮ $\bar{d} = x_{n1} - x_{n2}$

반복 1(x_{i1})	반복 2(x_{i2})	$\bar{d} = x_{i1} - x_{i2}$
82 54 ⋮ 71	75 50 ⋮ 74	−7 −4 ⋮ -3

$$\bar{d} = \frac{\sum_{i=1}^{n} d_i}{n}$$

$$s_d = \frac{\sum_{i=1}^{n} (d_i - \bar{d})^2}{n-1}$$

대응표본 t-검정

```
# 그래프에서 한글 폰트 인식하기
!sudo apt-get install -y fonts-nanum
!sudo fc-cache -fv
!rm ~/.cache/matplotlib -rf

#  *** 런타임 다시 시작
```

```
!pip install pingouin
```

```
# 1.기본
import numpy as np  # numpy 패키지 가져오기
import matplotlib.pyplot as plt # 시각화 패키지 가져오기
import seaborn as sns # 시각화

# 2.데이터 가져오기
import pandas as pd # csv -> dataframe으로 전환

# 3.통계분석 package
import pingouin as pg
from scipy import stats
import statsmodels.api as sm
```

```
# 기본세팅
# 테마 설정
sns.set_theme(style = "darkgrid")

# 한글 인식
plt.rc('font', family='NanumBarunGothic')
plt.rcParams['axes.unicode_minus'] = False # -인식
```

대응표본 t-검정

```python
pst_df = pd.read_csv('/06_1.PST.csv', encoding="cp949")
pst_df.head()
```

Python

```python
pst_df.shape
```

Python

```
(50, 4)
```

```python
pst_df.info()
```

Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   사전      50 non-null     float64
 1   사후1     50 non-null     float64
 2   사후2     50 non-null     float64
 3   사후3     50 non-null     float64
dtypes: float64(4)
memory usage: 1.7 KB
```

```python
pst_df.columns
```

Python

```
Index(['사전', '사후1', '사후2', '사후3'], dtype='object')
```

대응표본 t-검정

```python
# 그룹별 기술통계
pst_df.describe().round(2).T
```

Python

```python
# paired = True : paired sample t-test
pg.ttest(pst_df['사후1'], pst_df['사전'],
         paired = True,
         alternative = "two-sided").round(3)
```

Python

```python
# one sample로 분석할 때와 비교
pst_df['차이1'] = pst_df['사후1'] - pst_df['사전']
pg.ttest(pst_df['차이1'], 0, alternative = "two-sided").round(3)
```

Python

대응표본 t-검정

```python
# two-sided 차이가 없는 경우
pg.ttest(pst_df['사후2'], pst_df['사전'],
         paired = True,
         alternative = "two-sided").round(3)
```

Python

```python
# one-side로 바뀌면 차이가 있음
pg.ttest(pst_df['사후2'], pst_df['사전'],
         paired = True,
         alternative = "less").round(3)
```

Python

대응표본 t-검정

```python
# 정규분포일때
pg.normality(pst_df)
```

Python

```python
# 정규분포일때
pg.normality(pst_df['차이1'])
```

Python

대응표본 t-검정

```python
pst_df['차이2'] = pst_df['사후2'] - pst_df['사전']
pst_df['차이3'] = pst_df['사후3'] - pst_df['사전']
```

Python

```python
pg.normality(pst_df)
```

Python

```python
# Wilcoxon Rank test
pg.wilcoxon(pst_df['사후3'], pst_df['사전'],
            alternative='two-sided').round(3)
```

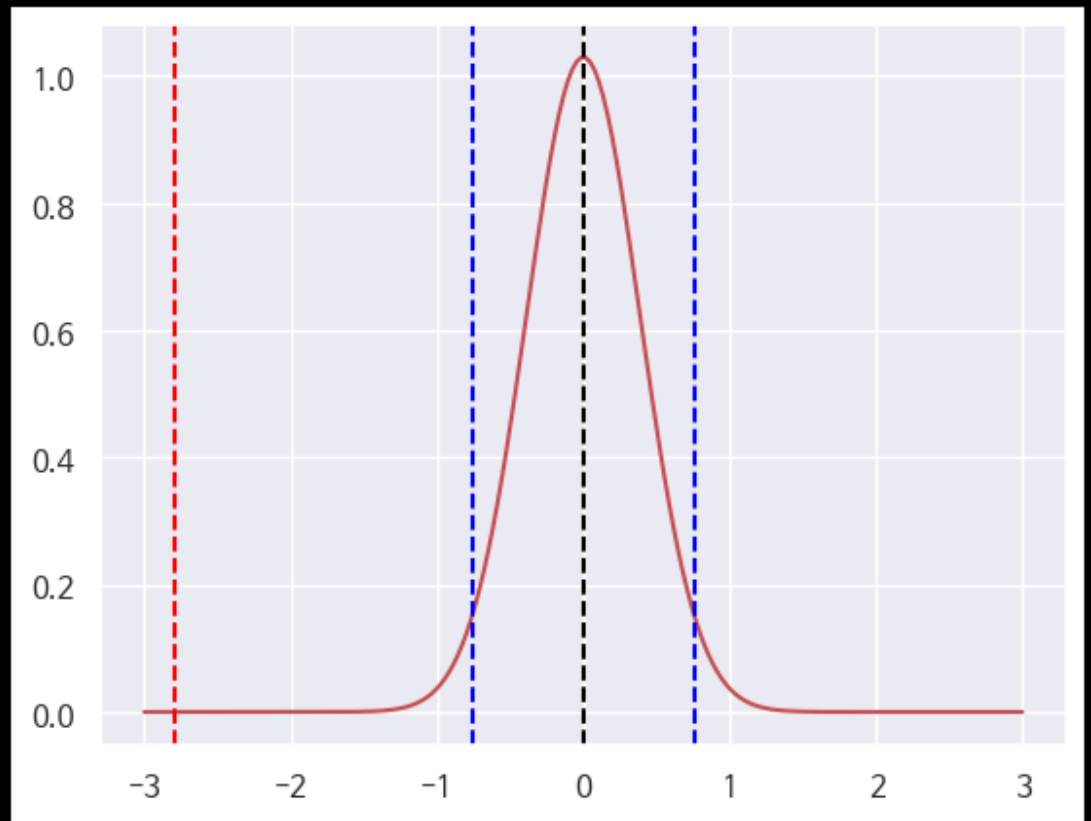
Python

```python
# 모수통계(t-test)와 비교
pg.ttest(pst_df['사후3'], pst_df['사전'],
         paired = True,
         alternative = "two-sided").round(3)
```

Python

대응표본 t-검정

```
from scipy.stats import norm # 정규분포

x_data = np.linspace(-3, 3, 200)

mu = 0 # 평균
x = -2.79 # 표본평균
se = 2.74/np.sqrt(50) # 표준편차(표준오차)

plt.plot(x_data, norm.pdf(x_data, loc = mu, scale = se), 'r-')
plt.axvline(x = mu, color='black', linestyle='--')
plt.axvline(x = mu + 1.96 * se, color='blue', linestyle='--')
plt.axvline(x = mu - 1.96 * se, color='blue', linestyle='--')
plt.axvline(x = x, color='red', linestyle='--')
plt.show()
```



대응표본 t-검정

```
pg.tost(x = pst_df['사후1'],
        y = pst_df['사전'],
        bound = 2,
        paired = True)
```

```
pg.tost(x = pst_df['사후2'],
        y = pst_df['사전'],
        bound = 2,
        paired = True)
```

대응표본 t-검정

- 섭취전(M=73.2)과 섭취후(M=70.6)는 통계적으로 유의한 차이가 있는 것으로 나타났으며, 다이어트약을 섭취한 후에 몸무게가 감 소한 것으로 나타났다 (t=3.63, p=0.002).

	섭취전(n=20)	섭취후(n=20)	t	p
몸무게	73.2	70.6	3.64	0.001

Q&A