04 One Sample Test

November 20, 2022

1 One Sample Test

[1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
```

1.1 Hypothesis

$$H_0 : \mu = \mu_0$$
$$H_1 : \mu > \mu_0 \cap H_1 : \mu < \mu_0 \cap H_1 : \mu \neq \mu_0$$

1.2 z-Test

Test statistic

$$T = \frac{\bar{X} - \mu_0}{\frac{\sigma}{\sqrt{n}}}$$

[2]:
```python
mu0 = 1  # Hypothesis: μ0 = 1
sigma = 0.1  # Assume to be known
n = 100  # Number of measurements

alpha = 0.05  # Significance level

c = stats.norm().ppf(1 - alpha)  # rejection area - right-sided

values = stats.norm(1.02, 0.1).rvs(n)
x_mean = np.mean(values)

t = (x_mean - mu0) / (sigma / np.sqrt(n))

p_value = 1 - stats.norm().cdf(t)

print(f"Test statatistic t: {t}")
print(f"Rejection area:     [{c}; inf]")
print(f"Reject H0:          {t > c}")
print(f"P-Value:            {p_value}")
```

```
Test statatistic t: 2.1460272545454684
Rejection area:     [1.6448536269514722; inf]
Reject H0:          True
P-Value:            0.015935402159858558
```

1.3 t-Test

Test statistic

$$T = \frac{\bar{X} - \mu_0}{\frac{s_x}{\sqrt{n}}}$$

The statistic T has a student-t distribution with $n - 1$ degrees of freedom

[7]:

```
mu0 = 1  # Hypothesis: μ0 = 1
n = 100  # Number of measurements
alpha = 0.05  # Significance level

# Generate random values
values = stats.norm(mu0 + 0.02, 0.1).rvs(n)
x_mean = np.mean(values)

# Rejection area
c = stats.t(n - 1).ppf(1 - alpha)
s = np.std(values, ddof=1)

t = (x_mean - mu0) / (s / np.sqrt(n))

p_value = 1 - stats.norm().cdf(t)

print(f"Test statistic t: {t}")
print(f"Rejection area:   {c}")
print(f"Reject H0:        {t > c}")
print(f"P-Value:          {p_value}")
```

```
Test statistic t: 2.6019082042417536
Rejection area:   1.6603911559963895
Reject H0:        True
P-Value:          0.004635333150248888
```

1.4 scipy.stats.ttest_1samp

Build in function in scipy.stats package for calculating the t-test

[10]:

```
n = 100  # Number of measurements
mu0 = 1  # Hypothesis: μ0 = 1
alpha = 0.05  # Significance level
```

```python
values = stats.norm(1.02, 0.1).rvs(n)

# Calculate rejection area
c = stats.t(n - 1).ppf(1 - alpha)

# Execute the t-test
res = stats.ttest_1samp(values, popmean=mu0, alternative='greater')

print(f"Rejection area: {c}")
print(f"Statistic:      {res[0]}")
print(f"P-Value:        {res[1]}")
print(f"Reject H0:      {res[0] > c}")
```

```
Rejection area: 1.6603911559963895
Statistic:      2.40829931379035
P-Value:        0.008938768515980647
Reject H0:      True
```

[ ]:

```
```