

Assignment T-test

Date: 05/05/2020 Name: D.Saravanan

TTest test the null hypothesis H_0 against the alternative hypothesis H_1 .

For univariate samples, TTest performs a Student t test. The test statistic is assumed to follow a StudentTDistribution [df].

The degrees of freedom df, used to specify the distribution of the test statistic, depend on the sample size, number of samples, and in the case of two univariate samples, the results of a test for equal variances.

For the TTest, a cutoff α is chosen such that H_0 is rejected only if $p < \alpha$. The value of α used for the "TestConclusion" and "ShortTestConclusion" properties is controlled by the SignificanceLevel option. This value α is also used in diagnostic tests of assumptions, including tests for normality, equal variance, and symmetry. By default, α is set to 0.05.

1. Certain refined edible oil is packed in tins holding 16 kg each. The filling machine can maintain this but with a standard deviation of 0.5 kg. Samples of 25 are taken from the production line. If a sample means are (i) 16.35 kg, (ii) 15.8 kg. Can we be 95 per cent sure that the sample has come from a population of 16 kg tins?

Null hypothesis: $H_0 : \mu = 16$

Alternative hypothesis: $H_1 : \mu \neq 16$

- (i) $\bar{x} = 16.35, s = 0.5$ and $n = 25$

$$t = \frac{\bar{x} - \mu}{s/\sqrt{n}} = \frac{16.35 - 16}{0.5/\sqrt{25}} = 3.5$$

- (ii) $\bar{x} = 15.8, s = 0.5$ and $n = 25$

$$t = \frac{\bar{x} - \mu}{s/\sqrt{n}} = \frac{15.8 - 16}{0.5/\sqrt{25}} = -2.0$$

Program:

```
#!/usr/bin/env python3
import numpy as np
from scipy.stats import t
import matplotlib.pyplot as plt
plt.style.use('seaborn-darkgrid')

print("_____T-test_____")

p_mean = float(input("\nPopulation mean: "))
s_mean = float(input("Sample mean: "))
s_stdv = float(input("Sample stand_dev: "))
n = float(input("Sample length: "))

# degrees of freedom, level of significance, confidence level
dof = n - 1; los = 0.05; cnl = 1 - los

# calculation of T-statistics
tstatistics = (s_mean - p_mean)/(s_stdv/np.sqrt(n))
tstatistics = -tstatistics if tstatistics > 0 else tstatistics
print("\nT-statistics: {:.5f}".format(tstatistics))
```

```

# calculation of Critical values
tcritical_l = t.ppf(q = los/2, df = dof)
tcritical_u = -tcritical_l
print("\nCritical values are {:.5f}, {:.5f}".format(tcritical_l, tcritical_u))

# decision making - using T-statistics and Critical values
if tstatistics < tcritical_l or tstatistics > tcritical_u:
    print("Reject the Null hypothesis.")
else: print("Fail to reject the Null hypothesis.")

# calculation of p-value
pvalue = 2*t.cdf(tstatistics, df = dof)
print("\np-value: {:.5f}".format(pvalue))

# decision making - using p-value
if pvalue < los: print("Reject the Null hypothesis.")
else: print("Fail to reject the Null hypothesis.")

# standard error
std_err = s_stdv/np.sqrt(n)
print("\nStandard Error: {:.5f}".format(std_err))

# confidence interval
cnf_int = s_mean + std_err * np.array([tcritical_l, tcritical_u])
print("Confidence Interval: {}".format(cnf_int))

# plot script
x1=np.linspace(-10,tcritical_l,1000); y1=t.pdf(x1,df=dof)
x2=np.linspace(tcritical_u,10,1000); y2=t.pdf(x2,df=dof)
x3=np.linspace(-4,4,1000); y3=t.pdf(x3,df=dof)
ax=plt.figure().add_subplot(111); ax.plot(x3,y3,color='brown')
ax.fill_between(x1,y1,0,alpha=0.5,color='brown',label='Rejection Region')
ax.fill_between(x2,y2,0,alpha=0.5,color='brown'); ax.set(xlabel='X',ylabel='P(X)')
ax.axvline(x=tstatistics,ls='--',c='b',label='T statistics = {:.5f}'.format(tstatistics))
ax.set(xlim=[-4,4],title="T Distribution (degrees of freedom = {:.0f})".format(dof))
plt.legend(); plt.show()

```

Output:

_____T-test_____

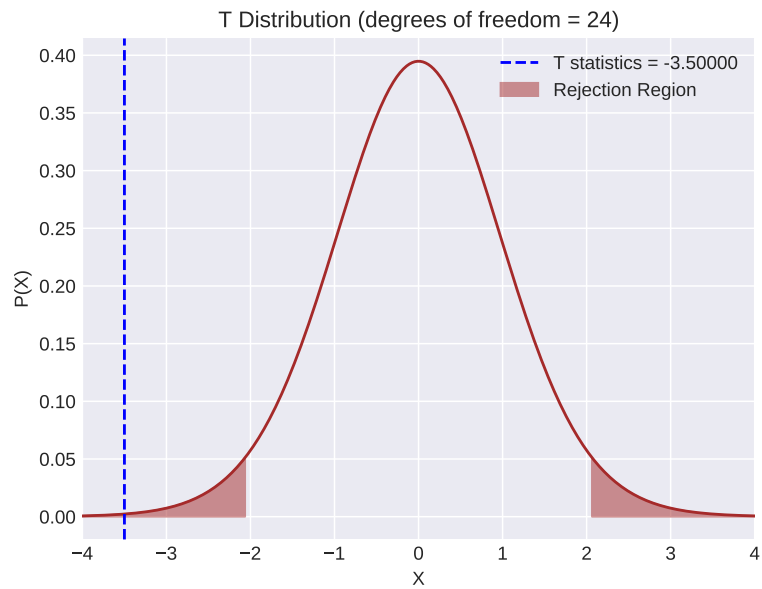
Population mean: 16
Sample mean: 16.35
Sample stand_dev: 0.5
Sample length: 25

T-statistics: -3.50000

Critical values are -2.06390, 2.06390
Reject the Null hypothesis.

p-value: 0.00184
Reject the Null hypothesis.

Standard Error: 0.10000
Confidence Interval: [16.14361014 16.55638986]



T-test

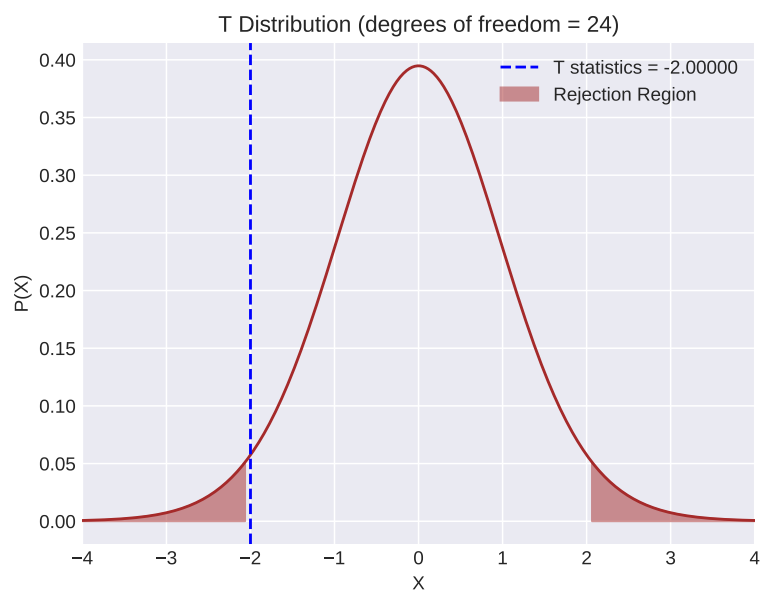
Population mean: 16
 Sample mean: 15.8
 Sample stand_dev: 0.5
 Sample length: 25

T-statistics: -2.00000

Critical values are -2.06390, 2.06390
 Fail to reject the Null hypothesis.

p-value: 0.05694
 Fail to reject the Null hypothesis.

Standard Error: 0.10000
 Confidence Interval: [15.59361014 16.00638986]



2. A filling machine is expected to fill 5 Kg of powder into bags. A sample of 10 bags gave the weights 4.7, 4.9, 5.0, 5.1, 5.4, 5.2, 4.6, 5.1, 4.6 and 4.7 Test whether the machine is working properly.

Null hypothesis: $H_0 : \mu = 5$

Alternative hypothesis: $H_1 : \mu \neq 5$

Sample mean:

$$\bar{x} = \frac{\sum_{i=1}^n x_i}{n} = \frac{4.7 + 4.9 + 5.0 + 5.1 + 5.4 + 5.2 + 4.6 + 5.1 + 4.6 + 4.7}{10} = 4.93$$

Sample variance:

$$s^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1} = \frac{\sum_{i=1}^{10} (x_i - 4.93)^2}{10 - 1} = 0.07567$$

Sample standard deviation:

$$s = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1}} = \sqrt{\frac{\sum_{i=1}^{10} (x_i - 4.93)^2}{10 - 1}} = \sqrt{0.07567} = 0.275076$$

T-statistics:

$$t = \frac{\bar{x} - \mu}{s/\sqrt{n}} = \frac{4.93 - 5}{0.275076/\sqrt{10}} = -0.80472$$

Standard Error:

$$S.E = \frac{s}{\sqrt{n}} = \frac{0.275076}{\sqrt{10}} = 0.08699$$

Program:

```
#!/usr/bin/env python3
import numpy as np
from scipy.stats import t, ttest_1samp
import matplotlib.pyplot as plt
plt.style.use('seaborn-darkgrid')

print("_____T-test (One Sample)_____")

p_mean = float(input("\nPopulation mean: "))
n = int(input("Sample length: "))
sample = [float(value) for value in input("Sample values: ").split()]
s_mean = np.mean(sample); s_stdv = np.std(sample, ddof=1)

# degrees of freedom, level of significance, confidence level
dof = n - 1; los = 0.05; cnl = 1 - los

# calculation of T-statistics and p-value
tstatistics, pvalue = ttest_1samp(sample, p_mean)
print("\nT statistics: {:.5f}".format(tstatistics))
```

```

# calculation of Critical values
tcritical_l = t.ppf(q = los/2, df = dof)
tcritical_u = -tcritical_l
print("\nCritical values are {:.5f}, {:.5f}".format(tcritical_l, tcritical_u))

# decision making - using T-statistics and Critical values
if tstatistics < tcritical_l or tstatistics > tcritical_u:
    print("Reject the Null hypothesis.")
else: print("Fail to reject the Null hypothesis.")

print("\np-value: {:.5f}".format(pvalue))

# decision making - using p-value
if pvalue < los: print("Reject the Null hypothesis.")
else: print("Fail to reject the Null hypothesis.")

# standard error
std_err = s_stdv/np.sqrt(n)
print("\nStandard Error: {:.5f}".format(std_err))

# confidence interval
cnf_int = s_mean + std_err * np.array([tcritical_l, tcritical_u])
print("Confidence Interval: {}".format(cnf_int))

# plot script
x1=np.linspace(-10,tcritical_l,1000); y1=t.pdf(x1,df=dof)
x2=np.linspace(tcritical_u,10,1000); y2=t.pdf(x2,df=dof)
x3=np.linspace(-4,4,1000); y3=t.pdf(x3,df=dof)
ax=plt.figure().add_subplot(111); ax.plot(x3,y3,color='brown')
ax.fill_between(x1,y1,0,alpha=0.5,color='brown',label='Rejection Region')
ax.fill_between(x2,y2,0,alpha=0.5,color='brown'); ax.set(xlabel='X',ylabel='P(X)')
ax.axvline(x=tstatistics,ls='--',c='b',label='T statistics = {:.5f}'.format(tstatistics))
ax.set(xlim=[-4,4],title="T Distribution (degrees of freedom = {:.0f})".format(dof))
plt.legend(); plt.savefig('tscript2.pdf',dpi=72,bbox_inches='tight'); plt.show()

```

Output:

_____T-test (One Sample)_____

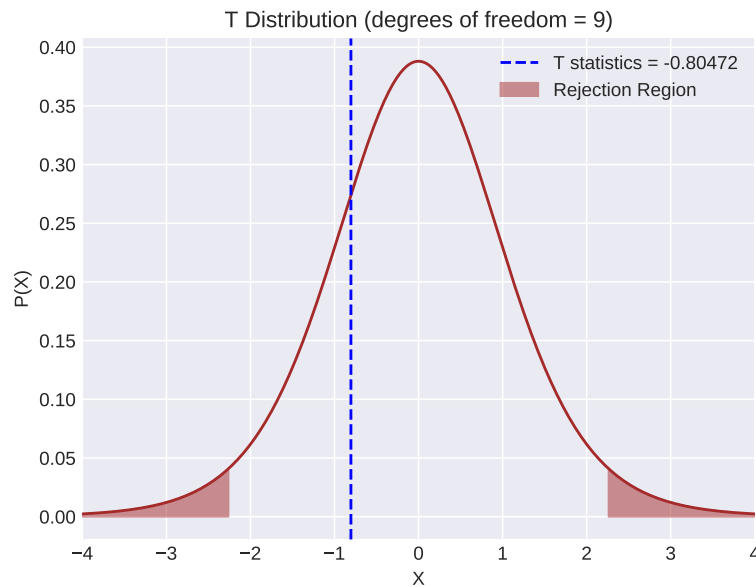
Population mean: 5
Sample length: 10
Sample values: 4.7 4.9 5.0 5.1 5.4 5.2 4.6 5.1 4.6 4.7

T statistics: -0.80472

Critical values are -2.26216, 2.26216
Fail to reject the Null hypothesis.

p-value: 0.44172
Fail to reject the Null hypothesis.

Standard Error: 0.08699
Confidence Interval: [4.73322266 5.12677734]



3. Two sets of ten students selected at random from a college were taken. One set was given memory test as they were and the other was given the memory test after two weeks of training and the scores are given below.

Set A: 10 8 7 9 8 10 9 6 7 8

Set B: 12 8 8 10 8 11 9 8 9 9

Do you think there is a significant effect due to training?

Null hypothesis: $H_0 : \mu_1 = \mu_2$

Alternative hypothesis: $H_1 : \mu_1 \neq \mu_2$

Set A:

Mean:

$$\bar{x}_1 = \frac{\sum_{i=1}^{n1} x_{1i}}{n1} = \frac{10 + 8 + 7 + 9 + 8 + 10 + 9 + 6 + 7 + 8}{10} = 8.2$$

Variance:

$$s_1^2 = \frac{\sum_{i=1}^{n1} (x_{1i} - \bar{x}_1)^2}{n1 - 1} = \frac{\sum_{i=1}^{10} (x_{1i} - 8.2)^2}{10 - 1} = 1.73333$$

Standard Deviation:

$$s_1 = \sqrt{\frac{\sum_{i=1}^{n1} (x_{1i} - \bar{x}_1)^2}{n1 - 1}} = \sqrt{\frac{\sum_{i=1}^{10} (x_{1i} - 8.2)^2}{10 - 1}} = \sqrt{1.73333} = 1.31656$$

Set B:

Mean:

$$\bar{x}_2 = \frac{\sum_{i=1}^{n2} x_{2i}}{n2} = \frac{12 + 8 + 8 + 10 + 8 + 11 + 9 + 8 + 9 + 9}{10} = 9.2$$

Variance:

$$s_2^2 = \frac{\sum_{i=1}^{n2} (x_{2i} - \bar{x}_2)^2}{n2 - 1} = \frac{\sum_{i=1}^{10} (x_{2i} - 9.2)^2}{10 - 1} = 1.95556$$

Standard Deviation:

$$s_2 = \sqrt{\frac{\sum_{i=1}^{n2} (x_{2i} - \bar{x}_2)^2}{n2 - 1}} = \sqrt{\frac{\sum_{i=1}^{10} (x_{2i} - 9.2)^2}{10 - 1}} = \sqrt{1.95556} = 1.39841$$

Calculation of s_1/s_2 :

$$\frac{s_1}{s_2} = \frac{1.31656}{1.39841} = 0.94147$$

Test statistic:

$$T = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{s_1^2/n1 + s_2^2/n2}}$$

where $n1$ and $n2$ are the sample sizes, \bar{x}_1 and \bar{x}_2 are the sample means, and s_1^2 and s_2^2 are the sample variances.

If equal variances are assumed ($0.5 < s_1/s_2 < 2$), then the formula reduces to:

$$T = \frac{\bar{x}_1 - \bar{x}_2}{s_p \sqrt{1/n1 + 1/n2}}$$

where

$$s_p^2 = \frac{(n1 - 1)s_1^2 + (n2 - 1)s_2^2}{n1 + n2 - 2}$$

Calculation of s_p :

$$s_p = \sqrt{\frac{(10 - 1) \times 1.31656^2 + (10 - 1) \times 1.39841^2}{10 + 10 - 2}} = 1.35810$$

Calculation of T statistic:

$$T = \frac{8.2 - 9.2}{1.35810 \times \sqrt{1/10 + 1/10}} = -1.64646$$

Calculation of Standard Error:

$$S.E = s_p \sqrt{1/n1 + 1/n2} = 1.35810 \times \sqrt{1/10 + 1/10} = 0.60736$$

Program:

```
#!/usr/bin/env python3
import numpy as np
from scipy.stats import t
import matplotlib.pyplot as plt
plt.style.use('seaborn-darkgrid')
```

```

def similar_variance(n1,s1_mean,n2,s2_mean):
    dof = n1+n2-2
    sp = np.sqrt(((n1-1)*s1_stdv**2+(n2-1)*s2_stdv**2)/(dof))
    return (s1_mean-s2_mean)/(sp * np.sqrt(1/n1 + 1/n2)), dof, sp

def non_similar_variance(n1,s1_mean,n2,s2_mean):
    sd = np.sqrt(s1_stdv**2/n1 + s2_stdv**2/n2)
    dof = (sd**4)/(((s1_stdv**2/n1)**2/(n1-1)) + ((s2_stdv**2/n2)**2/(n2-1)))
    return (s1_mean-s2_mean)/sd, dof, sd

def ttest_and_variance(s1_stdv, s2_stdv):
    if 0.5 < s1_stdv/s2_stdv < 2:
        return similar_variance(n1,s1_mean,n2,s2_mean)
    else:
        return non_similar_variance(n1,s1_mean,n2,s2_mean)

print("_____Two-sample T-test for unpaired data_____\\n")

sample1 = [float(value) for value in input("Sample_1 values: ").split()]
sample2 = [float(value) for value in input("Sample_2 values: ").split()]

# Number of Observations
n1 = len(sample1); n2 = len(sample2)

# Mean and Standard Deviation
s1_mean = np.mean(sample1); s1_stdv = np.std(sample1, ddof=1)
s2_mean = np.mean(sample2); s2_stdv = np.std(sample2, ddof=1)

# level of significance, confidence level
los = 0.05; cnl = 1 - los

# standard error (SE) of mean of sample1
std_err1 = s1_stdv/np.sqrt(n1)

print("\\nSample 1: \\n\\t Number of Observations = {} \\n\\t Mean = {:.5f}".format(n1,s1_mean))
print("\\t Standard Deviation = {:.5f}".format(s1_stdv))
print("\\t Standard Error of the Mean = {:.5f}".format(std_err1))

# standard error (SE) of mean of sample2
std_err2 = s2_stdv/np.sqrt(n2)

print("\\nSample 2: \\n\\t Number of Observations = {} \\n\\t Mean = {:.5f}".format(n2,s2_mean))
print("\\t Standard Deviation = {:.5f}".format(s2_stdv))
print("\\t Standard Error of the Mean = {:.5f}".format(std_err2))

# calculation of T-statistics and degrees of freedom
tstatistics, dof, sp = ttest_and_variance(s1_stdv, s2_stdv)
print("\\nT statistics: {:.5f}".format(tstatistics))

# calculation of Critical values
tcritical_l = t.ppf(q = los/2, df = dof)
tcritical_u = -tcritical_l
print("\\nCritical values are {:.5f}, {:.5f}".format(tcritical_l, tcritical_u))

# decision making - using T-statistics and Critical values
if tstatistics < tcritical_l or tstatistics > tcritical_u:
    print("Reject the Null hypothesis.")
else: print("Fail to reject the Null hypothesis.")

# calculation of p-value
pvalue = 2*t.cdf(tstatistics, df = dof)
print("\\np-value: {:.5f}".format(pvalue))

```



```

# decision making - using p-value and level of significance
if pvalue < los: print("Reject the Null hypothesis.")
else: print("Fail to reject the Null hypothesis.")

# standard error
std_error = sp * np.sqrt(1/n1 + 1/n2)
print("\nStandard Error: {:.5f}".format(std_error))

# confidence interval
cnf_int = (s1_mean - s2_mean) + std_error * np.array([tcritical_l, tcritical_u])
print("Confidence Interval: {}".format(cnf_int))

# plot script
x1=np.linspace(-10,tcritical_l,1000); y1=t.pdf(x1,df=dof)
x2=np.linspace(tcritical_u,10,1000); y2=t.pdf(x2,df=dof)
x3=np.linspace(-4,4,1000); y3=t.pdf(x3,df=dof)
ax=plt.figure().add_subplot(111); ax.plot(x3,y3,color='brown')
ax.fill_between(x1,y1,0,alpha=0.5,color='brown',label='Rejection Region')
ax.fill_between(x2,y2,0,alpha=0.5,color='brown'); ax.set(xlabel='X',ylabel='P(X)')
ax.axvline(x=tstatistics,ls='--',c='b',label='T statistics = {:.5f}'.format(tstatistics))
ax.set(xlim=[-4,4],title="T Distribution (degrees of freedom = {:.0f}".format(dof))
plt.legend(); plt.savefig('tscript3.pdf',dpi=72,bbox_inches='tight'); plt.show()

```

Output:

_____Two-sample T-test for unpaired data_____

Sample_1 values: 10 8 7 9 8 10 9 6 7 8
Sample_2 values: 12 8 8 10 8 11 9 8 9 9

Sample 1:
Number of Observations = 10
Mean = 8.20000
Standard Deviation = 1.31656
Standard Error of the Mean = 0.41633

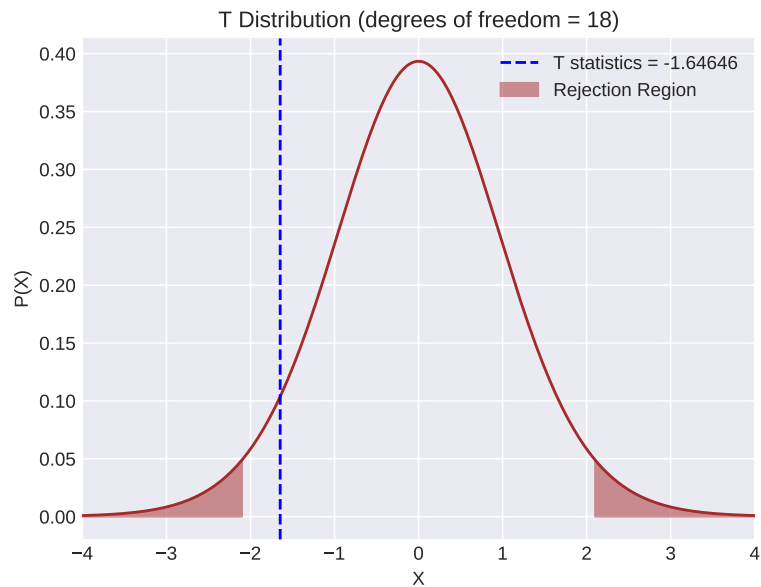
Sample 2:
Number of Observations = 10
Mean = 9.20000
Standard Deviation = 1.39841
Standard Error of the Mean = 0.44222

T statistics: -1.64646

Critical values are -2.10092, 2.10092
Fail to reject the Null hypothesis.

p-value: 0.11702
Fail to reject the Null hypothesis.

Standard Error: 0.60736
Confidence Interval: [-2.27602071 0.27602071]



Program: T-test with in-built scipy.stats.ttest

```
#!/usr/bin/env python3
import numpy as np
from scipy.stats import t, ttest_ind
import matplotlib.pyplot as plt
plt.style.use('seaborn-darkgrid')

print("_____Two-sample T-test for unpaired data_____\\n")

sample1 = [float(value) for value in input("Sample_1 values: ").split()]
sample2 = [float(value) for value in input("Sample_2 values: ").split()]

# Number of Observations
n1 = len(sample1); n2 = len(sample2)

# Mean and Standard Deviation
s1_mean = np.mean(sample1); s1_stdv = np.std(sample1, ddof=1)
s2_mean = np.mean(sample2); s2_stdv = np.std(sample2, ddof=1)

# calculation of degrees of freedom and pooled variance
if 0.5 < s1_stdv/s2_stdv < 2:
    dof = n1+n2-2
    sp = np.sqrt(((n1-1)*s1_stdv**2+(n2-1)*s2_stdv**2)/(dof))
else:
    sp = np.sqrt(s1_stdv**2/n1 + s2_stdv**2/n2)
    dof = (sp**4)/(((s1_stdv**2/n1)**2/(n1-1)) + ((s2_stdv**2/n2)**2/(n2-1)))

# level of significance, confidence level
los = 0.05; cnl = 1 - los

# standard error (SE) of mean of sample1
std_err1 = s1_stdv/np.sqrt(n1)

print("\\nSample 1: \\n\\t Number of Observations = {} \\n\\t Mean = {:.5f}".format(n1,s1_mean))
print("\\t Standard Deviation = {:.5f}".format(s1_stdv))
print("\\t Standard Error of the Mean = {:.5f}".format(std_err1))

# standard error (SE) of mean of sample2
std_err2 = s2_stdv/np.sqrt(n2)
```

```

print("\nSample 2: \n\t Number of Observations = {} \n\t Mean = {:.5f}".format(n2,s2_mean))
print("\t Standard Deviation = {:.5f}".format(s2_stdv))
print("\t Standard Error of the Mean = {:.5f}".format(std_err2))

# calculation of T-statistics and p-value
tstatistics, pvalue = ttest_ind(sample1, sample2)
print("\nT statistics: {:.5f}".format(tstatistics))

# calculation of Critical values
tcritical_l = t.ppf(q = los/2, df = dof)
tcritical_u = -tcritical_l
print("\nCritical values are {:.5f}, {:.5f}".format(tcritical_l, tcritical_u))

# decision making - using T-statistics and Critical values
if tstatistics < tcritical_l or tstatistics > tcritical_u:
    print("Reject the Null hypothesis.")
else: print("Fail to reject the Null hypothesis.")

print("\np-value: {:.5f}".format(pvalue))

# decision making - using p-value
if pvalue < los: print("Reject the Null hypothesis.")
else: print("Fail to reject the Null hypothesis.")

# standard error
std_error = sp * np.sqrt(1/n1 + 1/n2)
print("\nStandard Error: {:.5f}".format(std_error))

# confidence interval
cnf_int = (s1_mean - s2_mean) + std_error * np.array([tcritical_l, tcritical_u])
print("Confidence Interval: {}".format(cnf_int))

# plot script
x1=np.linspace(-10,tcritical_l,1000); y1=t.pdf(x1,df=dof)
x2=np.linspace(tcritical_u,10,1000); y2=t.pdf(x2,df=dof)
x3=np.linspace(-4,4,1000); y3=t.pdf(x3,df=dof)
ax=plt.figure().add_subplot(111); ax.plot(x3,y3,color='brown')
ax.fill_between(x1,y1,0,alpha=0.5,color='brown',label='Rejection Region')
ax.fill_between(x2,y2,0,alpha=0.5,color='brown'); ax.set(xlabel='X',ylabel='P(X)')
ax.axvline(x=tstatistics,ls='--',c='b',label='T statistics = {:.5f}'.format(tstatistics))
ax.set(xlim=[-4,4],title="T Distribution (degrees of freedom = {:.0f})".format(dof))
plt.legend(); plt.savefig('ttest2.pdf',dpi=72,bbox_inches='tight'); plt.show()

```

Output:

_____Two-sample T-test for unpaired data_____

Sample_1 values: 10 8 7 9 8 10 9 6 7 8
Sample_2 values: 12 8 8 10 8 11 9 8 9 9

Sample 1:
Number of Observations = 10
Mean = 8.20000
Standard Deviation = 1.31656
Standard Error of the Mean = 0.41633

Sample 2:

Number of Observations = 10
Mean = 9.20000
Standard Deviation = 1.39841
Standard Error of the Mean = 0.44222

T statistics: -1.64646

Critical values are -2.10092, 2.10092

Fail to reject the Null hypothesis.

p-value: 0.11702

Fail to reject the Null hypothesis.

Standard Error: 0.60736

Confidence Interval: [-2.27602071 0.27602071]

