01-Stats-1T-test

In [1]:

```
import importlib as imp
import sys, scipy.stats as stats
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
```

T-test, F-test, Hotelling's T-squared distribution¶

The t-Test is used to test the null hypothesis that the means of two populations are equal.

$H_0: \mu_1 - \mu_2 = 0$

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

1. First perform F-test to check if the variances are equal

In this page we don't talk about the Hotelling's T-squared test which is a generalization of t-test for multivariate case.

F-test¶

Given two series x, y - test if the variances of two population are equal. F-test is easy to conduct. $F-value$ of two series x and y is just the ratio $\frac{Var(x)}{Var(y)}$

H_0 **Null-hypothesis**: Two variances are same

H_a **Research hypothesis**: Two variances are different

F-test is done before t-test to determine if the variance of the two population are different.

Conditions:¶

- Both random variables are normally distributed
- The samples are independent

If X and Y have a normal distribution, the F-statistic will have F-distribution with $N__x -1\ and\ N__y -1$ degrees of freedom. To define the significance level which corresponds to the value of F-statistic high-precision, F-distribution approximation is used.

Example:¶

```
Data: is the number of study hours between male and female
Female: 26, 25, 43,34,18,52
Male:   23,30,18,25,28

</pre>
$H_0$: Two variances are same
$H_a$: Two variances differ
As you can see from the calculations below: $F$ is 7.373 ( $F_{critical}$ = 6.25 , P = 0.03
$F$ >> $F_{critical}$, therefore we reject null hypothesis and say "two vairances are diffe
t- test¶From t-test we get t=1.47260514049 p=0.187258096865.
$p >> p_{critical} (0.05)$ therefore we cannot reject the null and state that there is no di
```

In [2]:

```
# -*- coding: utf-8 -*-
# t- test
def ComputeDegreesOfFreedomFor_t_test(x,y, equal_variance = True):
    n1 = len(x)
    n2 = len(y)
    if equal_variance:
        return n1 + n2 -2;
    s1 = x.var()
    s2 = y.var()
    num = pow((s1/n1 + s2/n2),2)
    den = (pow(s1/n1,2) /(n1 -1)) + (pow(s2/n2,2) /(n2 -1))
    ret = num/den;
    ret = round(ret)
    return ret;
```

```
def t_critical(alpha, df, tails=1):
    t_critical_one_tailed=stats.t.ppf(1-alpha, df);
    t_critical_two_tailed=stats.t.ppf(1-alpha/2, df);

    if ( tails == 1):
        return t_critical_one_tailed;
    else:
        return t_critical_two_tailed;
#
# Returns equal Variance = true if there is no difference in variances
#
#
def Ftest(x,y, alpha = 0.05):
    F = x.var()/ y.var();
    n1 = len(x);  # degrees of freedom
    n2 = len(y);  # degrees of freedom
    cl = 1 - alpha; # Confidence Level
    F_c = stats.f.ppf(cl, n1 -1 , n2 - 1);  # n1-1, n2-1 are degrees of freedom
    p = 1- stats.f.cdf(F, n1 - 1, n2 - 1)

    equal_variance = True;
    if ( F > F_c ):
        equal_variance = False;

    ostr =  "Equal Variance: " + str(equal_variance) + \
            ", F= "+ str(F) + ", F_critical="+ str(F_c) + ",P=" + str(p);
    return (equal_variance , F, F_c, p, ostr);

#
# Chi square Critical value
def Chi2Critical(df, alpha = 0.05):
    cl = 1 - alpha; # Confidence Level
    Chi2_c = stats.chi2.ppf(cl, df);
    return Chi2_c;


a1 = np.array ( "26, 25, 43,34,18,52,23,30,18,25,28".split(",")).astype(int)
a2 = ["f"]*6 + ["m"]*5;
dfL = pd.DataFrame( {"a1":a1, "a2":a2})

#displayDFs(dfL)
d1 = dfL.loc[dfL['a2'] == 'f']['a1']
d2 = dfL.loc[dfL['a2'] == 'm']['a1']
F = d1.var()/ d2.var();
F_critical = stats.f.ppf(.95,len(d1) - 1, len(d2) - 1);  # n1 - 1, n2-1 are degrees of freed
```

```
p = 1- stats.f.cdf(F,5,4)  # 5 and 4 or len(d1)- 1 and len(d2) -1

print (np.array(d1), np.array(d2), "F =", F, "($F_critical,p)=", F_critical, p)
#
# F is >> F_critical and a case for rejecting Null hypothesis
#
equal_variance = True;  # Null Hypothesis
if ( F > F_critical):
    equal_variance = False; # Reject the Null Hypothesis

print ("Equal Variance: ", equal_variance)
```

```
[26 25 43 34 18 52] [23 30 18 25 28] F = 7.373271889400921 ($F_critical,p)= 6.25605650216088
Equal Variance:  False
```

In [3]:

```
# Try other tests for the heck of it
#print stats.bartlett(d2, d1)
#print stats.levene(d2, d1)
#stats.f.pdf(F, d1, d2)
```

In [4]:

```
t_stat, p = stats.ttest_ind(d1, d2, equal_var=equal_variance)

alpha = 0.05;
df = ComputeDegreesOfFreedomFor_t_test(d1,d2,equal_variance);
t_critical_one_tailed=stats.t.ppf(1-alpha, df);
t_critical_two_tailed=stats.t.ppf(1-alpha/2, df);

print (t_critical_one_tailed, t_critical_two_tailed, df)
print( t_stat, p )

print ('''
As p > p_critical of 0.05, we fail to reject the.
The observed difference between the sample means (33 - 24.8)
is not convincing enough to say that the average number of study hours

between female and male students differ significantly.
''')
```

```
1.894578605061305 2.3646242510102993 7
1.4726051404930287 0.18725809686523553

As p > p_critical of 0.05, we fail to reject the.
The observed difference between the sample means (33 - 24.8)
is not convincing enough to say that the average number of study hours
```

between female and male students differ significantly.

In [5]:

```
# Another set of examples I found at: https://gist.github.com/mblondel/1761714
# from scipy.stats import ttest_1samp, wilcoxon, ttest_ind, mannwhitneyu
#======================================================================
# EXAMPLE 1.
# one sample t-test
# null hypothesis: expected value = 7725

# daily intake of energy in kJ for 11 women
daily_intake = np.array([5260,5470,5640,6180,6390,6515,6805,7515,7515,8230,8770])

t_statistic, p_value = stats.ttest_1samp(daily_intake, 7725)


print ("one-sample t-test: p_value = ", p_value , '''
# daily intake of energy in kJ for 11 women
daily_intake = np.array([5260,5470,5640,6180,6390,6515,6805,7515,7515,8230,8770])

# Conducting: one sample t-test
# null hypothesis: expected value = 7725

# p_value < 0.05 => alternative hypothesis:
# data deviate significantly from the hypothesis that the mean
# is 7725 at the 5% level of significance

''')
# one sample wilcoxon-test
wz_statistic, wp_value = stats.wilcoxon(daily_intake - 7725)
print ("one-sample wilcoxon-test", wp_value)
#======================================================================
# EXAMPLE 2.

# two-sample t-test
# null hypothesis: the two groups have the same mean
# this test calls F test to check if equal variance can be assumed
# independent groups: e.g., how boys and girls fare at an exam
# dependent groups: e.g., how the same class fare at 2 different exams

energ = np.array([
# energy expenditure in mJ and stature (0=obese, 1=lean)
[9.21, 0],[7.53, 1],[7.48, 1],[8.08, 1],[8.09, 1],[10.15,1],[8.40, 1],[10.88, 1],
[6.13, 1],[7.90, 1],[11.51,0],[12.79,0],[7.05, 1],[11.85,0],[9.97, 0],[7.48,  1],
[8.79, 0],[9.69, 0],[9.68, 0],[7.58, 1],[9.19, 0],[8.11, 1]]
)
```

```
# similar to expend ~ stature in R
group1 = energ[:, 1] == 0
group1 = energ[group1][:, 0]
group2 = energ[:, 1] == 1
group2 = energ[group2][:, 0]
(equal_variance, F , F_c , p,d) = Ftest(group1, group2)
t_statistic, p_value = stats.ttest_ind(group1, group2,  equal_var=equal_variance)

# p_value (0.00079) < 0.05 => alt hypothesis: Mean value differ 5% significance level
print ("======================\nExample 2: two-sample t-test, p_value=", p_value)

# two-sample wilcoxon test
# a.k.a Mann Whitney U
u, p_value = stats.mannwhitneyu(group1, group2)
print ("two-sample wilcoxon-test p_value=", p_value)

#=====================================================================
# EXAMPLE 3
# pre and post-menstrual energy intake
intake = np.array([
[5260, 3910],[5470, 4220],[5640, 3885],[6180, 5160],[6390, 5645],[6515, 4680],
[6805, 5265],[7515, 5975],[7515, 6790],[8230, 6900],[8770, 7335],
])

pre = intake[:, 0]
post= intake[:, 1]

# paired t-test: doing two measurments on the same experimental unit
# e.g., before and after a treatment
t_statistic, p_value = stats.ttest_1samp(post - pre, 0)

# p < 0.05 => alternative hypothesis:
# the difference in mean is not equal to 0
print ("======================\nExample 3: paired t-test p_value=", p_value)

# alternative to paired t-test when data has an ordinary scale or when not
# normally distributed
z_statistic, p_value = stats.wilcoxon(post - pre)

print ("paired wilcoxon-test p_value=", p_value)
```

```
one-sample t-test: p_value =  0.018137235176105812
# daily intake of energy in kJ for 11 women
daily_intake = np.array([5260,5470,5640,6180,6390,6515,6805,7515,7515,8230,8770])
```

```
# Conducting: one sample t-test
# null hypothesis: expected value = 7725

# p_value < 0.05 => alternative hypothesis:
# data deviate significantly from the hypothesis that the mean
# is 7725 at the 5% level of significance


one-sample wilcoxon-test 0.0244140625
========================
Example 2: two-sample t-test, p_value= 0.0007989982111700593
two-sample wilcoxon-test p_value= 0.0010608066929400244
========================
Example 3: paired t-test p_value= 3.059020942934875e-07
paired wilcoxon-test p_value= 0.0009765625
```

In [6]:

```
#%run  "../StatUtils.py"
(eq,a,b,c,d) = Ftest(group1, group2)
(eq,a,b,c,d)
```

Out[6]:

```
(True,
 1.2275707804649485,
 2.848565142067682,
 0.3612456947013797,
 'Equal Variance: True, F= 1.2275707804649485, F_critical=2.848565142067682,P=0.361245694701
```

In [ ]:

In [ ]: