In [24]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as sps
import pandas as pd
from statsmodels.sandbox.stats.multicomp import multipletests


%matplotlib inline
```

In [3]:
```
data = pd.read_csv('hw7t3v1.txt', header = None, sep='\s+')
data
```

Out[3]:

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	sample0	2.754	2.040	0.759	-0.974	0.040	0.334	-0.205	0.904	0.708	-0.159	0.586	0.960
1	sample1	0.506	0.424	2.434	0.554	3.560	0.092	1.214	-0.483	NaN	NaN	NaN	NaN
2	sample2	-0.102	0.609	-0.209	-0.777	-0.356	0.954	-0.590	0.426	-0.934	-0.204	NaN	NaN
3	sample3	-3.638	0.263	-0.336	0.445	-1.346	1.100	-0.659	1.462	0.659	NaN	NaN	NaN
4	sample4	1.379	-0.400	-3.323	-1.824	0.192	-1.063	-0.602	-0.009	NaN	NaN	NaN	NaN
5	sample5	-0.638	0.296	-2.296	-1.521	0.354	0.030	-1.219	-1.408	1.248	-0.579	0.516	1.858
6	sample6	-0.535	0.166	0.459	0.205	-0.482	-0.546	0.632	1.664	0.343	-0.191	-0.589	-0.299
7	sample7	1.326	1.035	1.082	-0.041	-1.384	0.019	-0.688	1.836	NaN	NaN	NaN	NaN
8	sample8	0.028	-0.954	0.229	-0.676	-0.604	-0.722	-3.432	0.114	0.676	-0.993	-0.119	0.003
9	sample9	0.965	0.690	0.823	0.605	-0.225	2.806	1.212	0.663	-1.813	0.967	-1.688	-0.276

In [36]:
```
N = 0
samples = []
for i in range(10):
    sample = []
    for j in range(1,16):
        elem = data.values[i][j]
        if (elem is not None) and (str(elem) != 'nan'):
            sample.append(elem)
            ++N
    samples.append(np.array(sample))
```

In [23]:
```
for i in range(10):
    print (samples[i])
```

```
[ 2.754  2.04   0.759 -0.974  0.04   0.334 -0.205  0.904  0.708 -0.159
  0.586  0.96   1.758 -0.241  0.779]
[ 0.506  0.424  2.434  0.554  3.56   0.092  1.214 -0.483]
[-0.102  0.609 -0.209 -0.777 -0.356  0.954 -0.59   0.426 -0.934 -0.204]
[-3.638  0.263 -0.336  0.445 -1.346  1.1   -0.659  1.462  0.659]
[ 1.379 -0.4   -3.323 -1.824  0.192 -1.063 -0.602 -0.009]
[-0.638  0.296 -2.296 -1.521  0.354  0.03  -1.219 -1.408  1.248 -0.579
  0.516  1.858]
[-0.535  0.166  0.459  0.205 -0.482 -0.546  0.632  1.664  0.343 -0.191
 -0.589 -0.299]
[ 1.326  1.035  1.082 -0.041 -1.384  0.019 -0.688  1.836]
[  2.80000000e-02  -9.54000000e-01   2.29000000e-01  -6.76000000e-01
  -6.04000000e-01  -7.22000000e-01  -3.43200000e+00   1.14000000e-01
   6.76000000e-01  -9.93000000e-01  -1.19000000e-01   3.00000000e-03]
[ 0.965  0.69   0.823  0.605 -0.225  2.806  1.212  0.663 -1.813  0.967
 -1.688 -0.276 -0.782  0.009]
```

In [29]:
```
def normality(samples):
    p_val = np.zeros(len(samples))
    for i in range(len(samples)):
        p_val[i] = sps.shapiro(samples[i])[1]
    return multipletests(p_val, alpha=0.05)
```

In [30]:
```
normality(samples)
```

Out[30]:
```
(array([False, False, False, False, False, False, False, False, False, False],
dtype=bool),
 array([ 0.99489923,  0.84643734,  0.99489923,  0.83032883,  0.99489923,
         0.99489923,  0.48617296,  0.99489923,  0.08167792,  0.97083459]),
 0.0051161968918237433,
 0.005)
```

Нормальность выборок не отвергается

In [33]:
```
def variance_test(samples):
    return sps.levene(*samples)
```

In [34]:
```
variance_test(samples)
```

Out[34]:
```
LeveneResult(statistic=0.80438359574895302, pvalue=0.61316086453348229)
```

Гипотеза H' не отвергается

In [38]:
```
sps.f_oneway(*samples)
```

Out[38]:
```
F_onewayResult(statistic=2.3297365017058489, pvalue=0.020115921363988196)
```

Условия выполнены, но гипотеза H'' : $\mu_0 = \mu_1 = \ldots = \mu_9$ при уровне доверия 0.05 отвергается