In [1]:
```
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 [2]:
```
data = pd.read_csv('hw7t4v1.txt', header = None, sep='\s+')
data
```

Out[2]:

	0	1	2	3	4	5	6	7	8	9
0	350.679	365.072	369.438	341.794	370.960	348.542	354.961	352.684	356.232	346.113
1	349.056	366.898	371.662	344.392	371.601	347.953	355.961	354.277	355.804	344.766
2	351.519	367.114	371.047	344.434	372.191	351.316	355.416	352.506	357.331	345.261
3	350.140	366.307	370.995	344.264	371.204	351.015	357.414	352.651	356.881	346.014
4	352.484	367.037	370.171	343.372	370.292	350.836	355.497	352.758	355.163	343.838
5	348.116	366.893	370.529	342.394	370.388	348.641	352.330	352.329	354.097	344.098
6	352.257	365.078	370.915	343.558	371.975	349.513	356.010	351.969	356.427	345.943
7	350.030	365.458	368.594	342.754	375.320	347.830	355.199	352.579	356.520	345.614
8	350.367	366.951	371.182	343.720	371.201	350.269	356.227	352.045	357.075	345.023
9	349.746	368.216	370.529	344.384	371.705	350.829	355.869	352.008	356.787	347.020

In [6]:
```
samples = []
for i in range(10):
    sample = []
    for j in range(1,10):
        elem = data.values[i][j]
        if (elem is not None) and (str(elem) != 'nan'):
            sample.append(elem)
    samples.append(np.array(sample))
```

In [7]:
```
for i in range(10):
    print (samples[i])
```

```
[ 365.072  369.438  341.794  370.96   348.542  354.961  352.684  356.232
  346.113]
[ 366.898  371.662  344.392  371.601  347.953  355.961  354.277  355.804
  344.766]
[ 367.114  371.047  344.434  372.191  351.316  355.416  352.506  357.331
  345.261]
[ 366.307  370.995  344.264  371.204  351.015  357.414  352.651  356.881
  346.014]
[ 367.037  370.171  343.372  370.292  350.836  355.497  352.758  355.163
  343.838]
[ 366.893  370.529  342.394  370.388  348.641  352.33   352.329  354.097
  344.098]
[ 365.078  370.915  343.558  371.975  349.513  356.01   351.969  356.427
  345.943]
[ 365.458  368.594  342.754  375.32   347.83   355.199  352.579  356.52
  345.614]
[ 366.951  371.182  343.72   371.201  350.269  356.227  352.045  357.075
  345.023]
[ 368.216  370.529  344.384  371.705  350.829  355.869  352.008  356.787
  347.02 ]
```

In [8]:
```
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 [9]:
```
normality(samples)
```

Out[9]:
```
(array([False, False, False, False, False, False, False, False, False, False],
dtype=bool),
 array([ 0.87763166,  0.8435814 ,  0.87763166,  0.87763166,  0.84943005,
         0.81586869,  0.87763166,  0.87763166,  0.87763166,  0.87448077]),
 0.0051161968918237433,
 0.005)
```

In [10]:
```
sps.friedmanchisquare(*samples)
```

Out[10]:
```
FriedmanchisquareResult(statistic=23.530997304582197, pvalue=0.005107743893502
1304)
```

гипотеза $H_0 : \beta_0 = \beta_1 = \ldots = \beta_9$ не отвергается

Так как нормальность наших выборок не отвергается, проверим отсутствие влияния факторов критерием Фишера

In [53]:
```
def fisher(samples):
    res = len(samples) * (len(samples) - 1)
    res *= np.sum(samples.mean(axis=0) - np.mean(samples))
    s_1 = (samples - np.mean(samples))
    div = np.sum(s_1*s_1)
    div -= np.sum((samples.mean(axis=1)- np.mean(samples))**2) * len(samples[0]
    return res / div
```

In [54]:
```
fisher(data.values)
```

Out[54]: -1.656009164475756e-15

In [55]:
```
st = fisher(data.values)

p_value = sps.f.cdf(st, 9, 81)

print('Statistics is: %f, p_value: %f, result is: %s' %(st, p_value, str(p_valu
```

```
Statistics is: -0.000000, p_value: 0.000000, result is: False
```

Критерий Фишера отверг H_0