Python Practice 5

Probability and Statistics Programming (Sejong University)

Date: 2019.06.05 (By: S.M. Riazul Islam)

Multiple Comparison in ANOVA (Tukey's Method) and Simple Linear Regression

In [3]:
```
# Example from Book- Probability & Statistics for Enginering and Sciences
# Example 10.1: Comparison of Box Types Strength

import scipy.stats as st
sample1=[655.5, 788.3, 734.3, 721.4, 679.1, 699.4];
sample2=[789.2, 772.5, 786.9, 686.1, 732.1, 774.8];
sample3=[737.1, 639, 696.3, 671.7, 717.2, 727.1];
sample4=[535.1, 628.7, 542.4, 559, 586.9, 520];

st.f_oneway(sample1, sample2, sample3, sample4)

# Result will show that true average compression strength does appear to depend
```

Out[3]:
```
F_onewayResult(statistic=25.094289367676065, pvalue=5.525449877817013e-07)
```

Tukey's Method

In [6]:
```
# Create four groups of dummy data

import numpy as np
import scipy.stats as st

mu, sigma = 10, 3
group1 = np.random.normal(mu, sigma, 50)

mu, sigma = 11, 3
group2 = np.random.normal(mu, sigma, 50)

mu, sigma = 12, 3
group3 = np.random.normal(mu, sigma, 50)

mu, sigma = 13, 3
group4 = np.random.normal(mu, sigma, 50)

F_statistic, pvalue = st.f_oneway(group1, group2, group3, group4)

print ('F: ', F_statistic)
print ('P-value: ', pvalue)
```

```
F:  9.10909548174399
P-value:  1.1306023915968505e-05
```

In [13]:
```
# Creaate a dataframe using above dummy data

import pandas as pd

df = pd.DataFrame()
df['sample1'] = group1
df['sample2'] = group2
df['sample3'] = group3
df['sample4'] = group4
df.head()
```

Out[13]:

	sample1	sample2	sample3	sample4
0	13.525817	12.416787	10.931684	14.692670
1	6.398919	7.945670	10.659838	11.324119
2	12.212202	5.999966	10.956939	14.212653
3	9.821616	11.268730	21.201404	13.064506
4	10.269342	9.471676	11.765585	11.769238

In [24]:
```
# Reorganize data for MultiComparison

mydata=df.stack()
mydata.head()
```

Out[24]:
```
0  sample1    13.525817
   sample2    12.416787
   sample3    10.931684
   sample4    14.692670
1  sample1     6.398919
dtype: float64
```

In [25]:
```
mydata=mydata.reset_index()
mydata.head()
```

Out[25]:

	level_0	level_1	0
0	0	sample1	13.525817
1	0	sample2	12.416787
2	0	sample3	10.931684
3	0	sample4	14.692670
4	1	sample1	6.398919

In [26]:
```
mydata=mydata.rename(columns={'level_0': 'id','level_1': 'sample', 0:'datavalue'}
mydata.head()
```

Out[26]:

	id	sample	datavalue
0	0	sample1	13.525817
1	0	sample2	12.416787
2	0	sample3	10.931684
3	0	sample4	14.692670
4	1	sample1	6.398919

In [27]:
```
import statsmodels.stats.multicomp as mycomp
mc=mycomp.MultiComparison(mydata['datavalue'], mydata['sample'])
mc.tukeyhsd().summary()
```

Out[27]:

Multiple Comparison of Means - Tukey HSD,FWER=0.05

group1	group2	meandiff	lower	upper	reject
sample1	sample2	0.7765	-0.7325	2.2856	False
sample1	sample3	2.3003	0.7912	3.8093	True
sample1	sample4	2.6152	1.1061	4.1242	True
sample2	sample3	1.5237	0.0147	3.0328	True
sample2	sample4	1.8386	0.3296	3.3477	True
sample3	sample4	0.3149	-1.1942	1.8239	False

Simple Linear Regression

In [30]:
```
import pandas as pd
import numpy as np
gpadata=pd.read_csv("study_gpa.csv",delimiter=',')
gpadata.head()
```

Out[30]:

	study_hours	gpa
0	2	69
1	9	98
2	5	82
3	5	77
4	3	71

In [35]:
```
# We will use LinearRegression Class from sklearn.linear_model
# To use that, independent data (here study_hours) should be of 2D


study_hours=gpadata.study_hours.values  # series data to array
gpa=gpadata.gpa.values                  # series data to array

x=study_hours.reshape(-1,1)             # x has 1 colum and as many as rows requ
y=gpa                                   # y data can be of either 1D or 2D form
```

In [36]:
```
# Ready to apply Regression Model on x and y data

from sklearn.linear_model import LinearRegression    #LinearRegression is a class
model=LinearRegression()                # instance of LinearRegression with defau
model.fit(x,y)
```

Out[36]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [38]:
```
model.fit(x,gpa)
b0=model.intercept_
b1=model.coef_
print('Intercept, b0: ',b0)
print('Slope, b1: ', b1)
```

```
Intercept, b0:  55.03550295857988
Slope, b1:  [4.74260355]
```