In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
from itertools import combinations
import scipy
import re
```

Problem 1

In [ ]:
```
df = pd.DataFrame(data=[[0.43,0.415],[0.266,0.238],[0.567,0.39],\
    [0.531,0.41],[0.707,0.605],[0.716,0.609],[0.651,0.632],[0.589,0.523]\
    ,[0.469,0.411],[0.723,0.612]],columns=['bottom','surface']).astype(float)
df # create date set
```

Out[ ]:

	bottom	surface
0	0.430	0.415
1	0.266	0.238
2	0.567	0.390
3	0.531	0.410
4	0.707	0.605
5	0.716	0.609
6	0.651	0.632
7	0.589	0.523
8	0.469	0.411
9	0.723	0.612

In [ ]:
```
diff = df['bottom']-df['surface']
diff_mean = diff.mean()
diff_std = diff.std()
n=10
m=1
T2 = n*(diff_mean**2)/diff_std  #Calculate T^2
T2
```

Out[ ]:
```
1.2366104015295314
```

In [ ]:
```
F = ( (n-m)/m*(n-1) ) * T2  # Transformaing T^2 statistic into F for small sample
F
```

Out[ ]:
```
100.16544252389204
```

In [ ]:
```
F_mnma = scipy.stats.f.ppf(0.95,n-m,m) #F_m,n-m,a
F_mnma
```

Out[ ]:
```
240.54325471326283
```

In [ ]:
```
F > F_mnma # false means do not reject
```

Out[ ]:
```
False
```

F statistic < F_m,n-m,a We do not reject H0.

Problem 2

read and clean data

In [ ]:
```
df= pd.read_csv('pottery.csv')
df = df.drop(columns=['Unnamed: 0'])
df = df[df['kiln'] != 3]  # Reading and cleaning data
grand_mean = df.loc[:, df.columns != 'kiln'].mean() # grand mean for each variable
df.head() #display first five observations
```

Out[ ]:

	Al2O3	Fe2O3	MgO	CaO	Na2O	K2O	TiO2	MnO	BaO	kiln
0	18.8	9.52	2.00	0.79	0.40	3.20	1.01	0.077	0.015	1
1	16.9	7.33	1.65	0.84	0.40	3.05	0.99	0.067	0.018	1
2	18.2	7.64	1.82	0.77	0.40	3.07	0.98	0.087	0.014	1
3	16.9	7.29	1.56	0.76	0.40	3.05	1.00	0.063	0.019	1
4	17.8	7.24	1.83	0.92	0.43	3.12	0.93	0.061	0.019	1

In [ ]:
```
SS_error_vector = df.groupby('kiln').agg(lambda x: (len(x)-1 )* x.var()).sum() # Calculate SS error for each variable
print(SS_error_vector)
```

```
Al2O3    95.937548
Fe2O3    19.767617
MgO      15.211988
CaO       1.755473
Na2O      0.731214
K2O       3.845135
TiO2      0.578913
MnO       0.018690
BaO       0.000363
dtype: float64
```

In [ ]:
```
SS_treat_vector = (df.groupby('kiln').transform('mean') - grand_mean)**2
SS_treat_vector['kiln'] = df['kiln']
SS_treat_vector = SS_treat_vector.groupby('kiln').sum().sum()   #calculate SS treat for each variable
print(SS_treat_vector)
```

```
Al2O3    191.881522
Fe2O3    234.656294
MgO      114.402687
CaO        7.224732
Na2O       0.588637
K2O       24.183493
TiO2       0.652766
MnO        0.075848
BaO        0.000013
dtype: float64
```

In [ ]:
```
N = df.shape[0] # total sample size
g = 4 # number of samples(sites)

MS_error_vector = SS_error_vector / (N-g)  # Transform SS error into MS error
print(MS_error_vector)
```

```
Al2O3    2.459937
Fe2O3    0.506862
MgO      0.390051
CaO      0.045012
Na2O     0.018749
K2O      0.098593
TiO2     0.014844
MnO      0.000479
BaO      0.000009
dtype: float64
```

In [ ]:
```
MS_treat_vector = SS_treat_vector / (g - 1) #Transform SS treat into MS treat
print(MS_treat_vector)
```

```
Al2O3     63.960507
Fe2O3     78.218765
MgO       38.134229
CaO        2.408244
Na2O       0.196212
K2O        8.061164
TiO2       0.217589
MnO        0.025283
BaO        0.000004
dtype: float64
```

In [ ]:
```
F = MS_treat_vector/MS_error_vector #Calculate F statistic
print(F)
```

```
Al2O3      26.000871
Fe2O3     154.319654
MgO        97.767298
CaO        53.502126
Na2O       10.465167
K2O        81.761859
TiO2       14.658454
MnO        52.756295
BaO         0.459020
dtype: float64
```

In [ ]:
```
p_value = 1-scipy.stats.f.cdf(F, g-1, N-g) # Transform F Statistic into p-value base on F distribution
p_value
```

Out[ ]:
```
array([2.08349515e-09, 1.11022302e-16, 1.11022302e-16, 6.88338275e-14,
       3.48017068e-05, 1.11022302e-16, 1.52468510e-06, 8.55981952e-14,
       7.12481811e-01])
```

q2.2

In [ ]:
```
alpha = 0.05             #Apply Bonferroni orrection
print(p_value < alpha/9) #True means reject
```

```
[ True  True  True  True  True  True  True  True False]
```

We reject the null for first eights elements, meaning at least one site has a different population mean with the other sites for each element. Since there is at least one reject, so we reject H0:µ1=µ2=µ4=µ5

Q2.3

In [ ]:
```
#Apply BH procudure
sorted_p = p_value.copy().tolist()
```

```
sorted_p.sort() # sort the list to increasing order     (note:order of original p-value list stay the same)
k=0
for i in range(9):
    if sorted_p[i] <= (i+1)*0.1/50: # find the largest k such that p(k) <= k/m * a ; i+1 since i starts with 1
        k=i

print(k) # the largest k that we find, our k start at 0
```

```
7
```

In [ ]:

```
bh_result = (p_value<=sorted_p[k]) # find H0 the p(j) <= p(k*)
print(bh_result)

#If True, it means reject; If False, it means we can't reject H0
#result with bh procedure
```

```
[ True  True  True  True  True  True  True  True False]
```

After applying the Benjamini-Hochberg method, we reject the hypothesis for the first eight elements and pass the hypothesis for the last element. The results show that for the first eight elements, there is at least one site that has different different population mean with the others; for the last element, we accept all sites have same population mean. However, since there is at least one reject, so we reject H0:μ1=μ2=μ4=μ5

Everyone in this group contributed equally.