Open in Colab

Definitions

[ ] ↳ *2 cells hidden*

Correcting for multiple comparisons

↳ *5 cells hidden*

Exercise

In this exercise we will run through an example of correcting for multiple comparisons with both the Benjamini-Hochberg procedure and the more conservative Bonferroni correction.

First, simulate multiple (say, 1000) t-tests comparing two samples with equal means and standard deviations, and save the p-values. Obviously, at p<0.05 we expect that ~5% of the simulations to yield a "statistically significant" result (of rejecting the NULL hypothesis that the samples come from distributions with equal means).

Second, once you have the simulated p-values, apply both methods to address the multiple comparisons problem.

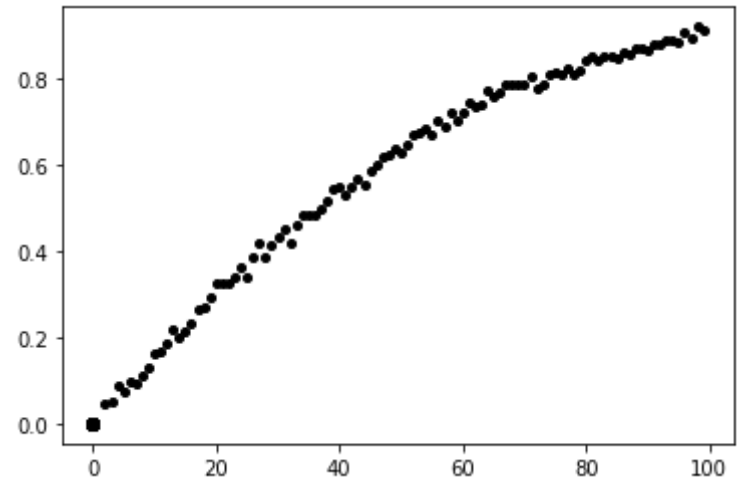
Third, set the sample 1 and sample 2 means to be 1 and 2 respectively, and re-run the exercise. What do you notice? What if you make the difference between means even greater?

```
import numpy as np
import scipy.stats as st
import matplotlib.pyplot as plt
from IPython.display import display, clear_output
```

```
from statsmodels.stats.multitest import multipletests


# Define a test distribution with a population mean different than 0 and a std of >1
test_mu  = 1
test_std = 3

# Null distribution
null_mu  = 1
null_std = test_std

# Max number of samples
max_n = 100

# for histograms
data_bin_size = 0.1
data_bins = np.arange(-10-data_bin_size/2, 10+1.5*data_bin_size, data_bin_size)
dax = (data_bins[1:] + data_bins[:-1])/2

# for simulations
num_experiments = 1000

pvaluelist = []; #List of all p-values

for n in np.arange(2, max_n):
  # Simulate multiple experiments
  samples = np.random.normal(test_mu, test_std, (num_experiments, n))

  # Compute the t-statistic from each experient
  t_stats = samples.mean(axis=1)/samples.std(axis=1,ddof=1)*np.sqrt(n)

  # Compute the p-value
  pvalue = np.count_nonzero(t_stats>st.t.ppf(0.975, n-1))/num_experiments
  pvaluelist.append(pvalue)

  plt.plot(0, 0, 'ko')
  plt.plot(n, np.count_nonzero(t_stats>st.t.ppf(0.975, n-1))/num_experiments, 'k.', markersize=8)

x = multipletests(pvaluelist, alpha=0.05, method='bonferroni') #Bonferroni correction
```

```
print(len(x[1][np.where(x[1]<0.05)]))
y = multipletests(pvaluelist, alpha=0.05, method='fdr_bh') # Benjamini/Hochberg correction
print(len(y[1][np.where(y[1]<0.05)]))
```

```
0
0
```



Additional Resources

How to correct for multiple comparisons in Matlab, R, and Python

Credits

✓ 0s completed at 7:06 AM