Module Four Discussion: Hypothesis Testing for the Difference in Two Population Proportions

This notebook contains the step-by-step directions for your Module Four discussion. It is very important to run through the steps in order. Some steps depend on the outputs of earlier steps. Once you have completed the steps in this notebook, be sure to answer the questions about this activity in the discussion for this module.

Reminder: If you have not already reviewed the discussion prompt, please do so before beginning this activity. That will give you an idea of the questions you will need to answer with the outputs of this script.

Initial post (due Thursday)

Step 1: Generating sample data

This block of Python code will generate two samples, both of size 50, that you will use in this discussion. The datasets will be unique to you and therefore your answers will be unique as well. The numpy module in Python allows you to create a data set using a Normal distribution. The data sets will be saved in Python dataframes and will be used in later calculations.

Click the block of code below and hit the **Run** button above.

In [1]:
```
import pandas as pd
import numpy as np

# create 50 randomly chosen values from a normal distribution. (arbitrarily us
ing mean=2.48 and standard deviation=0.500)
diameters_sample1 = np.random.normal(2.48,0.500,50)

# convert the array into a dataframe with the column name "diameters" using pa
ndas library
diameters_sample1_df = pd.DataFrame(diameters_sample1, columns=['diameters'])
diameters_sample1_df = diameters_sample1_df.round(2)

# create 50 randomly chosen values from a normal distribution. (arbitrarily us
ing mean=2.50 and standard deviation=0.750)
diameters_sample2 = np.random.normal(2.50,0.750,50)

# convert the array into a dataframe with the column name "diameters" using pa
ndas library
diameters_sample2_df = pd.DataFrame(diameters_sample2, columns=['diameters'])
diameters_sample2_df = diameters_sample2_df.round(2)

# print the dataframe to see the first 5 observations (note that the index of
dataframe starts at 0)
print("Diameters data frame of the first sample (showing only the first five o
bservations)")
print(diameters_sample1_df.head())
print()
print("Diameters data frame of the second sample (showing only the first five
observations)")
print(diameters_sample2_df.head())
```

```
Diameters data frame of the first sample (showing only the first five observa
tions)
   diameters
0       2.49
1       2.96
2       2.06
3       3.67
4       2.60

Diameters data frame of the second sample (showing only the first five observ
ations)
   diameters
0       1.08
1       2.18
2       3.64
3       2.99
4       3.58
```

Step 2: Performing hypothesis test for the difference in population proportions

The z-test for proportions can be used to test for the difference in proportions. The **proportions_ztest** method in statsmodels.stats.proportion submodule runs this test. The input to this method is a list of counts meeting a certain condition (given in the problem statement) and a list of sample sizes for the two samples.

Counts Python list that is assigned the number of observations in each sample with diameter values less than 2.20.

n Python list that is assigned the total number of observations in each sample.

Click the block of code below and hit the **Run** button above.

In [2]:
```
from statsmodels.stats.proportion import proportions_ztest

# number of observations in the first sample with diameter values less than 2.
20.
count1 = len(diameters_sample1_df[diameters_sample1_df['diameters']<2.20])

# number of observations in the second sample with diameter values less than
2.20.
count2 = len(diameters_sample2_df[diameters_sample2_df['diameters']<2.20])

# counts Python list
counts = [count1, count2]


# number of observations in the first sample
n1 = len(diameters_sample1_df)

# number of observations in the second sample
n2 = len(diameters_sample2_df)

# n Python list
n = [n1, n2]

# perform the hypothesis test. output is a Python tuple that contains test_sta
tistic and the two-sided P_value.
test_statistic, p_value = proportions_ztest(counts, n)

print("test-statistic =", round(test_statistic,2))
print("two tailed p-value =", round(p_value,4))
```

```
test-statistic = -1.98
two tailed p-value = 0.0473
```