Module Three Discussion: Confidence Intervals and Hypothesis Testing

This notebook contains the step-by-step directions for your Module Three 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 a unique sample of size 50 that you will use in this discussion. Note that your sample will be unique 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. Note that the mean and standard deviation were chosen for you. The data set will be saved in a Python dataframe that 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
import math
import scipy.stats as st

# create 50 randomly chosen values from a Normal distribution. (arbitrarily us
ing mean=2.48 and standard deviation=0.50).
diameters = np.random.normal(2.4800,0.500,50)

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

# print the dataframe (note that the index of dataframe starts at 0).
print("Diameters data frame\n")
print(diameters_df)
```

```
Diameters data frame

    diameters
0        2.64
1        1.99
2        3.00
3        1.86
4        2.20
5        2.26
6        2.79
7        1.94
8        2.70
9        2.70
10       2.83
11       2.11
12       1.85
13       2.61
14       2.49
15       2.18
16       1.89
17       2.65
18       2.72
19       2.74
20       3.41
21       3.37
22       3.40
23       2.62
24       1.87
25       2.64
26       2.35
27       2.36
28       2.52
29       1.96
30       2.11
31       2.49
32       2.08
33       2.51
34       2.77
35       2.53
36       2.58
37       1.52
38       1.90
39       2.53
40       3.10
41       2.96
42       2.17
43       2.13
44       2.55
45       2.35
46       3.02
47       2.44
48       3.07
49       2.74
```

Step 2: Constructing confidence intervals

You will assume that the population standard deviation is known and that the sample size is sufficiently large. Then you will use the Normal distribution to construct these confidence intervals. You will use the submodule scipy.stats to construct confidence intervals using your sample data.

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

In [2]:
```
# Python methods that calculate confidence intervals require the sample mean a
nd the standard error as inputs.

# calculate the sample mean
mean = diameters_df['diameters'].mean()

# input the population standard deviation, which was given in Step 1.
std_deviation = 0.5000

# calculate standard error = standard deviation / sqrt(n)   where n is the sam
ple size.
stderr = std_deviation/math.sqrt(len(diameters_df['diameters']))

# construct a 90% confidence interval.
conf_int_90 = st.norm.interval(0.90, mean, stderr)
print("90% confidence interval (unrounded) =", conf_int_90)
print("90% confidence interval (rounded) = (", round(conf_int_90[0], 2), ",",
round(conf_int_90[1], 2), ")")
print("")

# construct a 99% confidence interval.
conf_int_99 = st.norm.interval(0.99, mean, stderr)
print("99% confidence interval (unrounded) =", conf_int_99)
print("99% confidence interval (rounded) = (", round(conf_int_99[0], 2), ",",
round(conf_int_99[1], 2), ")")
```

```
90% confidence interval (unrounded) = (2.367691284632332, 2.600308715367667)
90% confidence interval (rounded) = ( 2.37 , 2.6 )

99% confidence interval (unrounded) = (2.301861363228155, 2.6661386367718443)
99% confidence interval (rounded) = ( 2.3 , 2.67 )
```

Step 3: Performing hypothesis testing for the population mean

Since you were given the population standard deviation in Step 1 and the sample size is sufficiently large, you can use the z-test for population means. The z-test method in statsmodels.stats.weightstats submodule runs the z-test. The input to this method is the sample dataframe and the value under the null hypothesis. The output is the test-statistic and the two-tailed P-value.

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

In [3]:
```
from statsmodels.stats.weightstats import ztest

# run z-test hypothesis test for population mean. The value under the null hyp
othesis is 2.30.
test_statistic, p_value = ztest(x1 = diameters_df['diameters'],  value = 2.30)

print("z-test hypothesis test for population mean")
print("test-statistic =", round(test_statistic,2))
print("two tailed p-value =",round(p_value,4))
```

```
z-test hypothesis test for population mean
test-statistic = 3.0
two tailed p-value = 0.0027
```

End of initial post

Attach the HTML output to your initial post in the Module Three discussion. The HTML output can be downloaded by clicking **File**, then **Download as**, then **HTML**. Be sure to answer all questions about this activity in the Module Three discussion.

Follow-up posts (due Sunday)

Return to the Module Three discussion to answer the follow-up questions in your response posts to other students. There are no Python scripts to run for your follow-up posts.