Homework 08: Hypothesis Testing, P-values, Bootstrapping

Name: Julia Troni

This assignment is due on Canvas by **6:00PM on Friday November 4**. Your solutions to theoretical questions should be done in Markdown directly below the associated question. Your solutions to computational questions should include any specified Python code and results as well as written commentary on your conclusions. Remember that you are encouraged to discuss the problems with your classmates, but **you must write all code and solutions on your own**.

NOTES:

- Any relevant data sets should be available in the Homework 01 assignment write-up on Canvas. To make life easier on the grader if they need to run your code, do not change the relative path names here. Instead, move the files around on your computer.
- If you're not familiar with typesetting math directly into Markdown then by all means, do your work on paper first and then typeset it later. Remember that there is a reference guide linked on Canvas on writing math in Markdown. **All** of your written commentary, justifications and mathematical work should be in Markdown.
- Because you can technically evaluate notebook cells is a non-linear order, it's a good idea to do Kernel → Restart & Run All as a check before submitting your solutions. That way if we need to run your code you will know that it will work as expected.
- It is **bad form** to make your reader interpret numerical output from your code. If a question asks you to compute some value from the data you should show your code output **AND** write a summary of the results in Markdown directly below your code.
- This probably goes without saying, but... For any question that asks you to calculate something, you **must show all work and justify your answers to receive credit**. Sparse or nonexistent work will receive sparse or nonexistent credit.

The standard imports for this HW:

In [1]:
```
# Per the standard import pandas as 'pd' and numpy as 'np'
import pandas as pd
import numpy as np
import matplotlib.pylab as plt
%matplotlib inline
import scipy.stats as stats
from scipy.stats import norm
```

Problem 1

In this HW you will need to use `.std()` when you are finding the test statistic. However, there are two kinds of standard deviations: those for a sample and those for a population.

Consider the python list below:

In [2]:
```
py_list = [4,2,3,4,2,3]
py_list
```

Out[2]:
```
[4, 2, 3, 4, 2, 3]
```

(3 points) Find both the sample standard deviation and the popluation standard deviation by hand.

$N = 6$

Sample mean $= \bar{X} = \frac{4+2+3+4+2+3}{6} = \frac{18}{6} = 3$

Sample Variance

$$= \frac{\sum_{i=1}^{N}(X_i - \bar{X})^2}{N-1} = \frac{\sum_{i=1}^{6}(X_i - 3)^2}{5} = \frac{(4-3)^2+(2-3)^2+(3-3)^2+(4-3)^2+(2-3)^2+(3-3)^2}{5} = \frac{4}{5} = 0.8$$

Sample Standard Deviation = $\sqrt{\frac{4}{5}} = 0.894427$

Population mean $= \bar{X} = \frac{4+2+3+4+2+3}{6} = \frac{18}{6} = 3$

Population Variance

$$= \frac{\sum_{i=1}^{N}(X_i - \bar{X})^2}{N} = \frac{\sum_{i=1}^{6}(X_i - 3)^2}{6} = \frac{(4-3)^2+(2-3)^2+(3-3)^2+(4-3)^2+(2-3)^2+(3-3)^2}{6} = \frac{4}{6} = 0.6666$$

Population Standard Deviation = $\sqrt{\frac{4}{6}} = 0.81649$

(2 points) Make an array from `py_list` and call it `np_array`.

In [3]:
```
# Using numpy.array()
np_array= np.array(py_list)
np_array
```

Out[3]:
```
array([4, 2, 3, 4, 2, 3])
```

(2 points) Make a Pandas Series from the list and call it `dfSeries`.

In [4]:
```
#Using pd.Series
dfSeries=pd.Series(py_list)
dfSeries
```

Out[4]:
```
0    4
1    2
2    3
3    4
4    2
```

```
5    3
dtype: int64
```

(2 points) Find `np_array.std()` and `dfSeries.std()`.

What type of standard deviation does `np_array.std()` return?

What type of standard deviation does `dfSeries.std()` return?

In [5]:
```
print("np_array.std()= {:.4f}, which is population standard deviation".format(np_array.
print("dfSeries.std()= {:.4f}, which is sample standard deviation".format(dfSeries.std(
```

```
np_array.std()= 0.8165, which is population standard deviation
dfSeries.std()= 0.8944, which is sample standard deviation
```

(2 points) Now find `np_array.std(ddof=0)` and `dfSeries.std(ddof=0)`, and `np_array.std(ddof=1)` and `dfSeries.std(ddof=1)`.

What do these return?

In [6]:
```
print("np_array.std(ddof=0) returns population standard deviation = {:.4f}".format(np_a
print("dfSeries.std(ddof=0) returns population standard deviation = {:.4f}".format(dfSe


print("np_array.std(ddof=1) returns sample standard deviation = {:.4f}".format(np_array
print("dfSeries.std(ddof=1) returns sample standard deviation = {:.4f}".format(dfSeries
```

```
np_array.std(ddof=0) returns population standard deviation = 0.8165
dfSeries.std(ddof=0) returns population standard deviation = 0.8165
np_array.std(ddof=1) returns sample standard deviation = 0.8944
dfSeries.std(ddof=1) returns sample standard deviation = 0.8944
```

Problem 2

A nematologist is interested in determining whether a new worm food (wood bark treated with peanut butter) results in shorter worm length than the standard length of 15.7 cm.

Shorter worms are more desirable as they tend to be stronger and live longer.

The nematologist feeds a random sample of worms with the new food and subsequently obtained the worm lengths found in the `csv` file `worm.csv`.

If the nematologist has in fact discovered a healthy new worm food then this food formula can be patented and sold world wide!

Therefore, the nematologist has hired you to explain whether or not this new food outperforms (with respect to worm length) the old worm food.

(read-in) Read in the csv file here:

In [7]:
```
# read in worm.csv
# Path to the data
file_path = 'worm.csv'
```

```
# Load the data into a DataFrame
dfWorm = pd.read_csv(file_path)
```

(1 point) Take a look at the first 5 rows of data.

In [8]:
```
# code here for looking at data:
dfWorm.head()
```

Out[8]:

	length
0	11.5
1	15.2
2	16.5
3	15.1
4	11.8

In order to determine whether or not this new worm food outperforms the standard food, you and the nematologist decide on a hypthesis test run at the 5% significance level.

(2 points) What does a 5% significance level mean?

Solution:

The significance level is the probability of rejecting the null hypothesis when it is true. So a 5% significant level means there is a 5% chance of rejecting the null hypothesis given it is true. In other words there is a confidence level of 95% that the evience shows that the null hypothesis is true when it actually is true.

Numerically this is $z_{-0.05} = \texttt{-stats.norm.ppf(0.95)} = -1.644$

(2 points) What is the null hypothesis and alternate hypothesis for this test?

Solution:

Null hypothesis: $H_0 : \mu = 15.7cm$

Alternate hypothesis: $H_1 : \mu < 15.7cm$

(1 point) How many worms were in this sample?

In [9]:
```
n= len(dfWorm)

print("n= ", n)
```

```
n=  33
```

(1 point) What is the mean of the sample?

In [10]:
```
xbar= dfWorm['length'].mean()

print("The sample mean Xbar= ", xbar)
```

```
The sample mean Xbar=  13.663636363636364
```

(1 point) What is the standard deviation of the sample?

In [11]:
```
s= dfWorm['length'].std(ddof=1)


print("The sample standard deviation s= ", s)
```

```
The sample standard deviation s=  2.5435971307650833
```

(2 points) What is the critical value?

In [12]:
```
zcrit= -norm.ppf(0.95)
print("The critical value is ", zcrit)
```

```
The critical value is  -1.6448536269514722
```

(2 points) What is the test statistic?

The test statistic is the sample mean which is, $\bar{X} = 13.6636$

(3 points) What is the conclusion?

In [13]:
```
zval=(xbar-15.7)/(s/np.sqrt(33))
zval
```

Out[13]:
```
-4.599006005602886
```

Solution:

By CLT $\bar{X} \approx N(\mu = 15.7, \frac{s^2}{n} = \frac{2.543^2}{33})$

$z = \frac{13.663 - 15.7}{\frac{2.543}{\sqrt{33}}} = -4.599$

$z_{-0.05} = -\texttt{stats.norm.ppf(0.95)} = -1.644$

Then since $-4.599 < -1.644$ we reject the null hypothesis and conclude that there is sufficient evidence to believe, at the 5% significance level, that the nematologists new food outperforms (with respect to worm length) the old worm food.

Instead of using a critical value to determine the above answer about the worm food, suppose you decide to base your decision on the p-value for this same data.

(2 points) In general, what is it that a p-value measures?

Solution:

The p value is the probability of seeing evidence of being in the rejection region, so it is the probability of finding evidence against the null hypothesis, and hence would causes you to reject the null hypothesis.

(2 points) What is the p-value for this experiment?

In [14]:
```
# Code your answer here:
p= stats.norm.cdf(zval)

print("The pvalue is ", p)
```

```
The pvalue is  2.1225577307875395e-06
```

(2 points) According to the p-value, should we reject the null or fail to reject the null?

Solution:

Since the p-value$= 2.122e-06 < \alpha = 0.05$, then we reject the null hypothesis

(2 points) Will the decisions concerning rejecting the null ever be different with respect to using a critical number versus a p-value?

Solution:

No. It does not matter if you are using the critical number approach or p-value approach, your decision to reject the null will always be the same. In other words, the two approaches yield the same result, but just use a different method to get there.

(2 points) According to the acquired p-value what is the largest confidence interval we could have used to reject the null hypothesis?

In [15]:
```
#to reject the null hypothesis p<alpha
#the highest confidence is 100% confidence, which is 1
#so subtract p from 1
1-p
print("The  largest confidence interval we could have used to reject the null hypothesi
```

```
The  largest confidence interval we could have used to reject the null hypothesis is  0.
9999978774422692
```

Problem 3

Widg's are fairly rare and difficult to come by.

In order to determine the density of a widg, one must destroy them with a crushing mechanism. A sample of of $n = 200$ widg densities has been determined.

From this sample, we would like to determine the probable density of other widgs' in the population. Of course we don't want to crush anymore widgs and they are hard to come by, so we will have to make due with this one sample.

The csv file `strap.csv` is the sample ($n = 200$) obtained from a distribution of widg densities.

(read-in) load the csv into a dataframe called dfWidg.

In [16]:
```
# read in strap.csv
# Path to the data
strap_file_path = 'strap.csv'

# Load the data into a DataFrame
dfWidg = pd.read_csv(strap_file_path)
```

(4 points) Write a function to draw 10000 bootstrapped resamples (with replacement) from this sample of 200 widg densities and compute a bootstrapped confidence interval for the mean at the 90% confidence level.

In [17]:
```
#get bootstrap mean of a sample with total of 'num_boots' samples
def bootstrap_widg_mean(sample, num_boots=10000):

    #empty li
    list_of_means = []
```

```
    for i in range(0,num_boots):
        #grab 1 random sample
        resample = np.random.choice(sample,size = len(sample))
        #take mean of that one random sample
        mean_resample = np.mean(resample)
        list_of_means.append(mean_resample)
    #90% confidence interval so cut off 5% off of each tail
    L = np.percentile(list_of_means,5)
    U = np.percentile(list_of_means,95)

    CI = [L, U]
    return CI

bootstrap_widg_mean(dfWidg["widg"])
```

Out[17]: [2.39, 2.705]

(2 points) What is the meaning of this 90% CI?

Solution:

This 90% CI means that we can be 90% confident that the population mean for widgs lies within the range [2.385, 2.7]

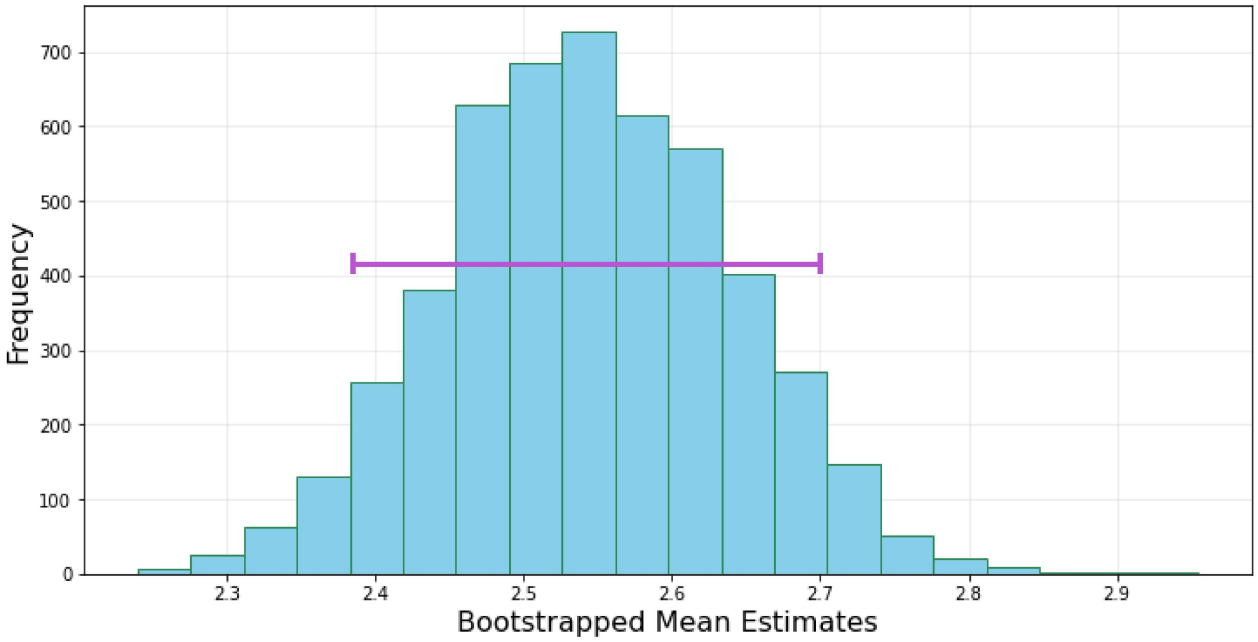
(4 points) write a function that will graph a histogram of our 10000 bootstrap samples with the confidence interval superimposed on the histogram.

BTW, choose your own colors: https://datascientyst.com/full-list-named-colors-pandas-python-matplotlib/

In [18]:
```
# Code your answer here:
def hist_bootstrap_widg_mean(sample, num_boots=5000):

     #empty list
    list_of_means = []

    for i in range(0,num_boots):
        #grab 1 random sample
        resample = np.random.choice(sample,size = len(sample))
        #take mean of that one random sample
        mean_resample = np.mean(resample)
        list_of_means.append(mean_resample)
    #90% confidence interval so cut off 5% off of each tail
    L = np.percentile(list_of_means,5)
    U = np.percentile(list_of_means,95)

    CI = [L, U]

    #plot the list of bootstraped sample means with labels
    fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(12,6))
    pd.Series(list_of_means).hist(ax=ax, color="skyblue", edgecolor="seagreen", bins=20
    ax.grid(alpha=0.25)
    ax.set_axisbelow(True)
    ax.set_xlabel("Bootstrapped Mean Estimates", fontsize=16)
    ax.set_ylabel("Frequency", fontsize=16)

    #plot the line to span the confidence interval
```

```
    ax.plot([CI[0], CI[1]], [num_boots/12, num_boots/12], color="mediumorchid", lw=3)
    ax.plot([CI[0], CI[0]], [num_boots/12-10, num_boots/12+10], color="mediumorchid", l
    ax.plot([CI[1], CI[1]], [num_boots/12-10, num_boots/12+10], color="mediumorchid", l
    return CI
hist_bootstrap_widg_mean(dfWidg["widg"])
```

Out[18]: [2.385, 2.7]



(2 points) What is the mean of the $n = 200$ data, and is this mean found inside the bootstrap CI?

In [19]:
```
sampmean=dfWidg["widg"].mean()
print("The mean of the  n=200  data is {} so it is found inside the CI found above".for
```

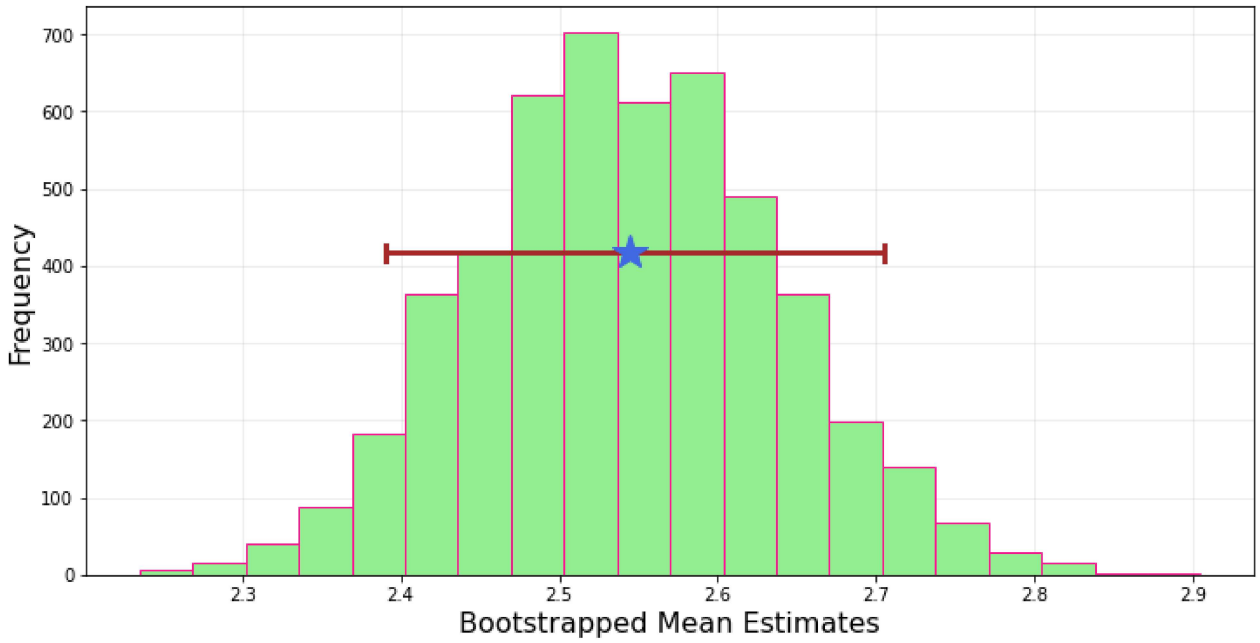
```
The mean of the  n=200  data is 2.545 so it is found inside the CI found above
```

(2 points) Graph a histogram of our 10000 bootstrap samples with the confidence interval superimposed on the histogram, AND the sample mean as a dot on the CI. BTW, the actual population mean is 2.5. You likely arrived at a mean that is very close to this.

In [20]:
```
def hist_bootstrap_widg_mean_dot(sample, num_boots=5000):

     #empty li
    list_of_means = []

    for i in range(0,num_boots):
        #grab 1 random sample
        resample = np.random.choice(sample,size = len(sample))
        #take mean of that one random sample
        mean_resample = np.mean(resample)
        list_of_means.append(mean_resample)
    #90% confidence interval so cut off 5% off of each tail
    L = np.percentile(list_of_means,5)
    U = np.percentile(list_of_means,95)

    CI = [L, U]

    #plot the list of bootstraped sample means with labels
```

```
    fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(12,6))
    pd.Series(list_of_means).hist(ax=ax, color="lightgreen", edgecolor="deeppink", bins
    ax.grid(alpha=0.25)
    ax.set_axisbelow(True)
    ax.set_xlabel("Bootstrapped Mean Estimates", fontsize=16)
    ax.set_ylabel("Frequency", fontsize=16)

    #plot the line to span the confidence interval
    ax.plot([CI[0], CI[1]], [num_boots/12, num_boots/12], color="brown", lw=3)
    ax.plot([CI[0], CI[0]], [num_boots/12-10, num_boots/12+10], color="brown", lw=3)
    ax.plot([CI[1], CI[1]], [num_boots/12-10, num_boots/12+10], color="brown", lw=3)

    #plot the point at the sample mean
    #I did a star instead of a dot to make it more fun :)
    ax.plot([sampmean, sampmean], [num_boots/12, num_boots/12], marker = '*', ms = 20,

    return CI
hist_bootstrap_widg_mean_dot(dfWidg["widg"])
```

Out[20]: [2.39, 2.705]



In [ ]:

In [ ]: