Homework 9: Central Limit Theorem

Reading:

- [Why the mean matters (https://www.inferentialthinking.com/chapters/14/Why_the_Mean_Matters.html)](https://www.inferentialthinking.com/chapters/14/Why_the_Mean_Matters.html)

Please complete this notebook by filling in the cells provided. Before you begin, execute the following cell to load the provided tests. Each time you start your Python session, you will need to execute this cell again to load the tests.

Homework 9 is due **Thursday, 4/15 at 11:59pm**.

Start early so that you can come to office hours if you're stuck. Late work will not be accepted as per the course policies.

In [ ]:
```
# Don't change this cell; just run it.

import numpy as np
from datascience import *

# These lines do some fancy plotting magic.
import matplotlib
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
import warnings
warnings.simplefilter('ignore', FutureWarning)
```

1. Testing the Central Limit Theorem

The Central Limit Theorem tells us that the probability distribution of the **sum** or **average** of a large random sample drawn with replacement will be roughly normal, *regardless of the distribution of the population from which the sample is drawn*.

That's a pretty big claim, but the theorem doesn't stop there. It further states that the standard deviation of this normal distribution is given by

$$\frac{\texttt{sd of the original distribution}}{\sqrt{\texttt{sample size}}}$$

In other words, suppose we start with *any distribution* that has standard deviation x, take a sample of size n (where n is a large number) from that distribution with replacement, and compute the **mean** of that sample. If we repeat this procedure many times, then those sample means will have a normal distribution with standard deviation $\frac{x}{\sqrt{n}}$.

That's an even bigger claim than the first one! The proof of the theorem is beyond the scope of this class, but in this exercise, we will be exploring some data to see the CLT in action.

Question 1.1. The CLT only applies when sample sizes are "sufficiently large." This isn't a very precise statement. Is 10 large? How about 50? The truth is that it depends both on the original population distribution and just how "normal" you want the result to look. Let's use a simulation to get a feel for how the distribution of the sample mean changes as sample size goes up.

Consider a coin flip. If we say `Heads` is 1 and `Tails` is 0, then there's a 50% chance of getting a 1 and a 50% chance of getting a 0, which definitely doesn't match our definition of a normal distribution. The average of several coin tosses, where Heads is 1 and Tails is 0, is equal to the proportion of heads in those coin tosses (which is equivalent to the mean value of the coin tosses), so the CLT should hold **true** if we compute the sample proportion of heads many times.

Write a function called `sample_size_n` that takes in a sample size n. It should return an array that contains 5000 sample proportions of heads, each from n coin flips. We give you some starter code below.

In [ ]:
```
def sample_size_n(n):
    coin_proportions = make_array(.5, .5) # our coin is fair
    heads_proportions = make_array()
    for i in np.arange(5000):
        simulated_proportions = ...
        prop_heads = ...
        heads_proportions = ...
    return heads_proportions


sample_size_n(5)
```

The code below will use the function you just defined to plot the empirical distribution of the sample mean for various sample sizes. Drag the slider or click on the number to the right to type in a sample size of your choice. The x- and y-scales are kept the same to facilitate comparisons. Notice the shape of the graph as the sample size increases and decreases.

```
In [ ]: # Just run this cell
        from ipywidgets import interact

        def outer(f):
            def graph(x):
                bins = np.arange(-0.01,1.05,0.02)
                sample_props = f(x)
                Table().with_column('Sample Size: {}'.format(x), sample_props).h
        ist(bins=bins)
                plt.ylim(0, 30)
                print('Sample SD:', np.std(sample_props))
                plt.show()
            return graph

        interact(outer(sample_size_n), x=(0, 400, 1), continuous_update=False);

        # Min sample size is 0, max is 400
        # The graph will refresh a few times when you drag the slider around
```

You can see that even the means of samples of 10 items follow a roughly bell-shaped distribution. A sample of 50 items looks quite bell-shaped.

Question 1.2. In the plot for a sample size of 10, why are the bars spaced at intervals of .1, with gaps in between?

Write your answer here, replacing this text.

Now we will test the second claim of the CLT: The SD of the sample mean is the SD of the original distribution, divided by the square root of the sample size.

We have imported the flight delay data and computed its standard deviation for you.

```
In [ ]: united = Table.read_table('united_summer2015.csv')
        united_std = np.std(united.column('Delay'))
        united_std
```

Question 1.3. Write a function called `empirical_sample_mean_sd` that takes a sample size `n` as its argument. The function should simulate 500 samples with replacement of size `n` from the flight delays dataset, and it should return the standard deviation of the **means of those 500 samples**.

Hint: This function will be similar to the `sample_size_n` function you wrote earlier.

```
In [ ]: def empirical_sample_mean_sd(n):
            sample_means = make_array()
            for i in np.arange(500):
                sample = ...
                sample_mean = ...
                sample_means = ...
            return np.std(sample_means)

        empirical_sample_mean_sd(10)
```

Question 1.4. Now, write a function called `predict_sample_mean_sd` to find the predicted value of the standard deviation of means according to the relationship between the standard deviation of the sample mean and sample size that is discussed here (https://www.inferentialthinking.com/chapters/14/5/variability-of-the-sample-mean.html) in the textbook. It takes a sample size `n` (a number) as its argument. It returns the predicted value of the standard deviation of the mean delay time for samples of size `n` from the flight delays (represented in the table `united`).

```
In [ ]: def predict_sample_mean_sd(n):
            ...

        predict_sample_mean_sd(10)
```

The cell below will plot the predicted and empirical SDs for the delay data for various sample sizes.

```
In [ ]: sd_table = Table().with_column('Sample Size', np.arange(1,101))
        predicted = sd_table.apply(predict_sample_mean_sd, 'Sample Size')
        empirical = sd_table.apply(empirical_sample_mean_sd, 'Sample Size')
        sd_table = sd_table.with_columns('Predicted SD', predicted, 'Empirical S
        D', empirical)
        sd_table.scatter('Sample Size')
```

Question 1.5. Do our predicted and empirical values match? Why is this the case?

Hint: Are there any laws that we learned about in class that might help explain this?

Write your answer here, replacing this text.

2. Polling and the Normal Distribution

Question 2.1. Michelle is a statistical consultant, and she works for a group that supports Proposition 538 (which would mandate labeling of all horizontal or vertical axes), called Yes on 538. They want to know how many Californians will vote for the proposition.

Michelle polls a uniform random sample of all California voters, and she finds that 210 of the 400 sampled voters will vote in favor of the proposition. Fill in the code below to form a table with 3 columns: the first two columns should be identical to `sample`. The third column should be named `Proportion` and have the proportion of total voters that chose each option.

In [ ]:
```
sample = Table().with_columns(
    "Vote",  make_array("Yes", "No"),
    "Count", make_array(210,   190))
sample_size = ...
sample_with_proportions = ...
sample_with_proportions
```

Question 2.2. She then wants to use 10,000 bootstrap resamples to compute a confidence interval for the proportion of all California voters who will vote Yes. Fill in the next cell to simulate an empirical distribution of Yes proportions with 10,000 resamples. In other words, use bootstrap resampling to simulate 10,000 election outcomes, and populate `resample_yes_proportions` with the yes proportion of each bootstrap resample. Then, visualize `resample_yes_proportions` with a histogram. You should see a bell shaped curve centered near the proportion of Yes in the original sample.

In [ ]:
```
resample_yes_proportions = make_array()
for i in np.arange(10000):
    resample = ...
    resample_yes_proportions = ...
Table().with_column("Resample Yes proportion", resample_yes_proportions)
.hist(bins=np.arange(.2, .8, .01))
```

Question 2.3. Why does the Central Limit Theorem (CLT) apply in this situation, and how does it explain the distribution we see above?

Write your answer here, replacing this text.

In a population whose members are 0 and 1, there is a simple formula for the standard deviation of that population:

$$\texttt{standard deviation} = \sqrt{(\text{proportion of 0s}) \times (\text{proportion of 1s})}$$

(Figuring out this formula, starting from the definition of the standard deviation, is an fun exercise for those who enjoy algebra.)

Question 2.4. Using only the CLT and the numbers of Yes and No voters in our sample of 400, compute (*algebraically*) a number `approximate_sd` that's the predicted standard deviation of the array `resample_yes_proportions` according to the Central Limit Theorem. **Do not access the data in `resample_yes_proportions` in any way.** Remember that a predicted standard deviation of the sample means can be computed from the population SD and the size of the sample.

Also remember that if we do not know the population SD, we can use the sample SD as a reasonable approximation in its place.

In [ ]:
```
approximate_sd = ...
approximate_sd
```

Question 2.5. Compute the SD of the array `resample_yes_proportions` which will act as an approximation to the true SD of the possible sample proportions. This will help verify whether your answer to question 2.2 is approximately correct.

In [ ]:
```
exact_sd = ...
exact_sd
```

Question 2.6. Again, without accessing `resample_yes_proportions` in any way, compute an approximate 95% confidence interval for the proportion of Yes voters in California using the Central Limit Theorem and `approximate_sd`.

The cell below draws your interval as a red bar below the histogram of `resample_yes_proportions`; use that to verify that your answer looks right.

Hint: How many SDs corresponds to 95% of the distribution promised by the CLT? Recall the discussion in the textbook here. (https://www.inferentialthinking.com/chapters/14/3/sd-and-the-normal-curve.html) (https://www.inferentialthinking.com/chapters/14/3/sd-and-the-normal-curve.html)

(https://www.inferentialthinking.com/chapters/14/3/sd-and-the-normal-curve.html)
(https://www.inferentialthinking.com/chapters/14/3/sd-and-the-normal-curve.html)

In [ ]:
```
lower_limit = ...
upper_limit = ...
print('lower:', lower_limit, 'upper:', upper_limit)
```

In [ ]:
```
# Run this cell to plot your confidence interval.
Table().with_column("Resample Yes proportion", resample_yes_proportions)
.hist(bins=np.arange(.2, .8, .01))
plt.plot(make_array(lower_limit, upper_limit), make_array(0, 0), c='r',
lw=10);
```

Your confidence interval should overlap the number 0.5. That means we can't be very sure whether Proposition 538 is winning, even though the sample Yes proportion is a bit above 0.5.

The Yes on 538 campaign really needs to know whether they're winning. It's impossible to be absolutely sure without polling the whole population, but they'd be okay if the standard deviation of the sample mean were only 0.005. They ask Michelle to run a new poll with a sample size that's large enough to achieve that. (Polling is expensive, so the sample also shouldn't be bigger than necessary.)

Michelle consults Chapter 14 of your textbook. Instead of making the conservative assumption that the population standard deviation is 0.5 (coding Yes voters as 1 and No voters as 0), she decides to assume that it's equal to the standard deviation of the sample,

$$\sqrt{(\text{Yes proportion in the sample}) \times (\text{No proportion in the sample})}.$$

Under that assumption, Michelle decides that a sample of 9,975 would suffice.

Question 2.7. Does Michelle's sample size achieve the desired standard deviation of sample means? What SD would you achieve with a smaller sample size? A higher sample size? To explore this, first compute the SD of sample means obtained by using Michelle's sample size.

In [ ]:
```
estimated_population_sd = ...
michelle_sample_size = ...
michelle_sample_mean_sd = ...
print("With Michelle's sample size, you would predict a sample mean SD o
f %f." % michelle_sample_mean_sd)
```

Then, compute the SD of sample means that you would get from a smaller sample size. Ideally, you should pick a number that is significantly smaller, but any sample size smaller than Michelle's will do.

In [ ]:
```
smaller_sample_size = ...
smaller_sample_mean_sd = ...
print("With this smaller sample size, you would predict a sample mean SD
of %f" % smaller_sample_mean_sd)
```

Finally, compute the SD of sample means that you would get from a larger sample size. Here, a number that is significantly larger would make any difference more obvious, but any sample size larger than Michelle's will do.

In [ ]:
```
larger_sample_size = ...
larger_sample_mean_sd = ...
print("With this larger sample size, you would predict a sample mean SD
 of %f" % larger_sample_mean_sd)
```

(https://www.inferentialthinking.com/chapters/14/3/sd-and-the-normal-curve.html)

(https://www.inferentialthinking.com/chapters/14/3/sd-and-the-normal-curve.html)
(https://www.inferentialthinking.com/chapters/14/3/sd-and-the-normal-curve.html)

(https://www.inferentialthinking.com/chapters/14/3/sd-and-the-normal-curve.html)

3. Submission (https://www.inferentialthinking.com/chapters/14/3/sd-and-the-normal-curve.html)

Once you're finished, submit your assignment as a .pdf (download as .html, then print to save as a .pdf) on Gradescope.