hw08_master

November 8, 2019

1 Homework 8: Confidence Intervals

Reading: * Estimation

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 server, you will need to execute this cell again to load the tests.

Homework 8 is due **Thursday, 10/31 at 11:59pm**. You will receive an early submission bonus point if you turn in your final submission by Wednesday, 10/30 at 11:59pm. Start early so that you can come to office hours if you're stuck. Check the website for the office hours schedule. Late work will not be accepted as per the policies of this course.

Directly sharing answers is not okay, but discussing problems with the course staff or with other students is encouraged. Refer to the policies page to learn more about how to learn cooperatively.

For all problems that you must write our explanations and sentences for, you **must** provide your answer in the designated space. Moreover, throughout this homework and all future ones, please be sure to not re-assign variables throughout the notebook! For example, if you use `max_temperature` in your answer to one question, do not reassign it later on.

[ ]:
```
# 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)

from client.api.notebook import Notebook
ok = Notebook('hw08.ok')
```

1.1 1. Plot the Vote

Four candidates are running for President of Dataland. A polling company surveys 1000 people selected uniformly at random from among voters in Dataland, and it asks each one who they are planning on voting for. After compiling the results, the polling company releases the following proportions from their sample:

Candidate	Proportion
Candidate C	0.47
Candidate T	0.38
Candidate J	0.08
Candidate S	0.03
Undecided	0.04

These proportions represent a uniform random sample of the population of Dataland. We will attempt to estimate the corresponding *parameters*, or the proportion of the votes that each candidate will receive from the entire population. We will use confidence intervals to compute a range of values that reflects the uncertainty of our estimates.

The table `votes` contains the results of the survey. Candidates are represented by their initials. Undecided voters are denoted by `U`.

[14]:
```
votes = Table.read_table('votes.csv')
num_votes = votes.num_rows
votes
```

[14]:
```
vote
C
J
C
S
J
J
T
C
T
C
… (990 rows omitted)
```

Question 1. Complete the function `one_resampled_proportion` below. It should return Candidate C's proportion of votes after simulating one bootstrap sample of `tbl`.

Note: `tbl` will always be in the same format as `votes`.

```
BEGIN QUESTION
name: q1_1
manual: false
```

[15]:
```
def one_resampled_proportion(tbl):
    # BEGIN SOLUTION
    bootstrap = tbl.sample()
    single_proportion = np.count_nonzero(bootstrap.column('vote') == 'C') /␣
 ↪num_votes
    return single_proportion
    # END SOLUTION
```

Question 2. Complete the `proportions_in_resamples` function such that it returns an array of 5,000 bootstrapped estimates of the proportion of voters who will vote for Candidate C. You should use the `one_resampled_proportion` function you wrote above.

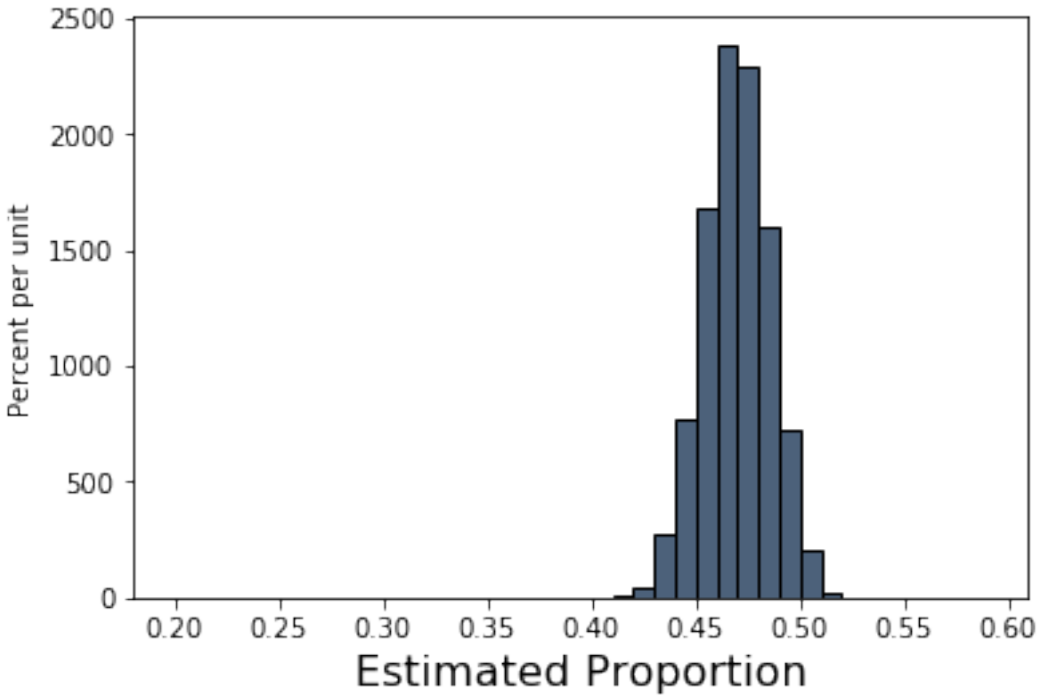
Note: There are no public tests for this question, the autograder cell below will return 0.0% passed.

```
BEGIN QUESTION
name: q1_2
manual: false
```

[18]:
```
def proportions_in_resamples():
    prop_c = make_array()
    # BEGIN SOLUTION
    for i in np.arange(5000):
        single_proportion = one_resampled_proportion(votes)
        prop_c = np.append(prop_c, single_proportion)
    return prop_c
    # END SOLUTION
```

In the following cell, we run the function you just defined, `proportions_in_resamples`, and create a histogram of the calculated statistic for the 5,000 bootstrap estimates of the proportion of voters who voted for Candidate C. Based on what the original polling proportions were, does the graph seem reasonable? Talk to a friend or ask a TA if you are unsure!

[15]:
```
resampled_proportions = proportions_in_resamples()
Table().with_column('Estimated Proportion', resampled_proportions).hist(bins=np.
 ↪arange(0.2,0.6,0.01))
```



Question 3. Using the array `resampled_proportions`, find the values at the two edges of the middle 95% of the values in the data. (Compute the lower and upper ends of the interval, named `c_lower_bound` and `c_upper_bound`, respectively.)

```
BEGIN QUESTION
name: q1_3
manual: false
```

[16]:
```
c_lower_bound = percentile(2.5, resampled_proportions) # SOLUTION
c_upper_bound = percentile(97.5, resampled_proportions) # SOLUTION
print("Bootstrapped 95% confidence interval for the proportion of C voters in␣
 ↪the population: [{:f}, {:f}]".format(c_lower_bound, c_upper_bound))
```

```
Bootstrapped 95% confidence interval for the proportion of C voters in the
population: [0.439000, 0.500000]
```

Question 4. The survey results seem to indicate that Candidate C is beating Candidate T among voters. We would like to use confidence intervals to determine a range of likely values for her true *lead*. Candidate C's lead over Candidate T is:

Candidate C's proportion of the vote − Candidate T's proportion of the vote.

Define the function `one_resampled_difference` that returns **exactly one value** of Candidate C's lead over Candidate T from one bootstrap sample of `tbl`.

```
BEGIN QUESTION
```

```
name: q1_4
manual: false
```

[9]:
```
def one_resampled_difference(tbl):
    bootstrap = tbl.sample() #SOLUTION
    c_proportion = np.count_nonzero(bootstrap.column('vote') == 'C') / tbl.
 ↪num_rows #SOLUTION
    t_proportion = np.count_nonzero(bootstrap.column('vote') == 'T') / tbl.
 ↪num_rows #SOLUTION
    return c_proportion - t_proportion #SOLUTION
```

Question 5. Write a function called `leads_in_resamples` that finds 5,000 bootstrapped estimates (the result of calling `one_resampled_difference`) of Candidate C's lead over Candidate T. Plot a histogram of the resulting samples.

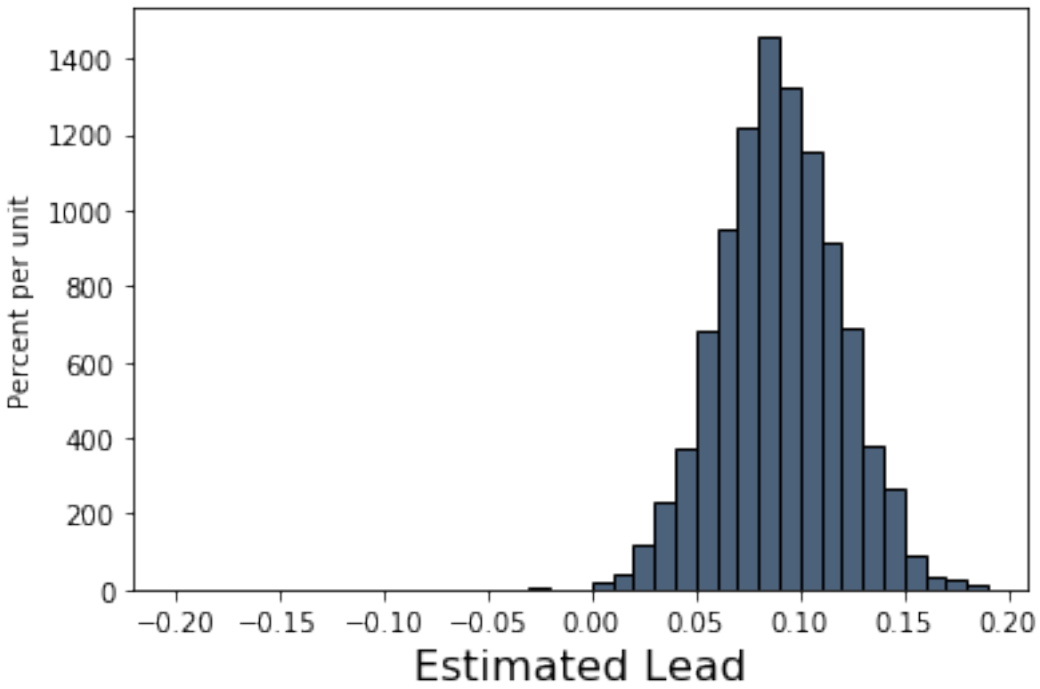
Note: Candidate C's lead can be negative.

```
BEGIN QUESTION
name: q1_5
manual: false
```

[28]:
```
bins = np.arange(-0.2,0.2,0.01)

def leads_in_resamples():
    # BEGIN SOLUTION
    leads = make_array()
    for i in np.arange(5000):
        bootstrap_lead = one_resampled_difference(votes)
        leads = np.append(leads, bootstrap_lead)
    return leads
    # END SOLUTION

sampled_leads = leads_in_resamples()
Table().with_column('Estimated Lead', sampled_leads).hist(bins=bins)
```



Question 6. Use the simulated data from Question 5 to compute an approximate 95% confidence interval for Candidate C's true lead over Candidate T.

```
BEGIN QUESTION
name: q1_6
manual: false
```

[29]:
```
diff_lower_bound = percentile(2.5, sampled_leads) # SOLUTION
diff_upper_bound = percentile(97.5, sampled_leads) # SOLUTION
print("Bootstrapped 95% confidence interval for Candidate C's true lead over␣
 ↪Candidate T: [{:f}, {:f}]".format(diff_lower_bound, diff_upper_bound))
```

```
Bootstrapped 95% confidence interval for Candidate C's true lead over Candidate
T: [0.033000, 0.146000]
```

1.2 2. Interpreting Confidence Intervals

The staff computed the following 95% confidence interval for the proportion of Candidate C voters:

$$[.439, .5]$$

(Your answer may have been a bit different; that doesn't mean it was wrong!)

Question 1 Can we say that there is a 95% probability that the interval [.439, .5] contains the true proportion of the population who is voting for Candidate C? Answer "yes" or "no" and explain your reasoning.

Note: ambiguous answers using language like "sometimes" or "maybe" will not receive credit.

```
BEGIN QUESTION
name: q2_1
manual: true
```

SOLUTION: No, the true proportion is some value x. Our observed interval [.439, .5] has already been fixed, so x is either in the interval or it is not.

Question 2

The staff also created 80%, 90%, and 99% confidence intervals from the same sample, but we forgot to label which confidence interval represented which percentages! Identify which confidence levels correspond to which confidence intervals, and match each pair by writing the interval (e.g. [.444,.495]) and what percentage corresponds with each interval in the cell below. **Then**, explain your thought process.

The intervals are below:

- [.444,.495]
- [.450,.490]
- [.430,.511]

```
BEGIN QUESTION
name: q2_2
manual: true
```

SOLUTION:

80% CI: [.450,.490]

90% CI: [.444,.495]

99% CI: [.430,.511]

We compute these intervals by taking the middle X% of a bunch of bootstrap statistics. As the confidence level increases, we are including more and more of the statistics, so the interval widens. Intuitively, we might be very confident that the population parameter is within in some giant interval, but only moderately confident that it's within some smaller interval.

Question 3 Suppose we produced 10,000 new samples (each one a uniform random sample of 1,000 voters) from the population and created a 95% confidence interval from each one. Roughly how many of those 10,000 intervals do you expect will actually contain the true proportion of the population?

Assign your answer to `true_proportion_intervals`.

```
BEGIN QUESTION
name: q2_3
manual: false
```

[2]:
```
true_proportion_intervals = 9500 # SOLUTION
```

Recall the second bootstrap confidence interval you created, which estimated Candidate C's lead over Candidate T. Among voters in the sample, her lead was .09. The staff's 95% confidence interval for her true lead (in the population of all voters) was

$$[.032, .15].$$

Suppose we are interested in testing a simple yes-or-no question:

"Are the candidates tied?"

Our null hypothesis is that the proportions are equal, or, equivalently, that Candidate C's lead is exactly 0. Our alternative hypothesis is that her lead is not equal to 0. In the questions below, don't compute any confidence interval yourself - use only the staff's 95% confidence interval.

Question 4

Say we use a 5% P-value cutoff. Do we reject the null, fail to reject the null, or are we unable to tell using our staff confidence interval?

Assign `candidates_tied` to the number corresponding to the correct answer.

1. Reject the null
2. Data is consistent with the null hypothesis
3. Unable to tell using our staff confidence interval

Hint: If you're confused, take a look at this chapter of the textbook.

```
BEGIN QUESTION
name: q2_4
manual: false
```

[5]:
```
candidates_tied = 1 # SOLUTION
```

Question 5 What if, instead, we use a P-value cutoff of 1%? Do we reject the null, fail to reject the null, or are we unable to tell using our staff confidence interval?

Assign `cutoff_one_percent` to the number corresponding to the correct answer.

1. Reject the null
2. Data is consistent with the null hypothesis
3. Unable to tell using our staff confidence interval

```
BEGIN QUESTION
name: q2_5
manual: false
```

[8]:
```
cutoff_one_percent = 3 # SOLUTION
```

Question 6 What if we use a P-value cutoff of 10%? Do we reject, fail to reject, or are we unable to tell using our confidence interval?

Assign `cutoff_ten_percent` to the number corresponding to the correct answer.

1. Reject the null
2. Data is consistent with the null hypothesis
3. Unable to tell using our staff confidence interval

```
BEGIN QUESTION
name: q2_6
manual: false
```

[11]:
```
cutoff_ten_percent = 1 # SOLUTION
```

1.3 3. Submission

Once you're finished, select "Save and Checkpoint" in the File menu and then execute the `submit` cell below. The result will contain a link that you can use to check that your assignment has been submitted successfully. If you submit more than once before the deadline, we will only grade your final submission. If you mistakenly submit the wrong one, you can head to okpy.org and flag the correct version. To do so, go to the website, click on this assignment, and find the version you would like to have graded. There should be an option to flag that submission for grading!

[ ]:
```
_ = ok.submit()
```

[ ]:
```
# For your convenience, you can run this cell to run all the tests at once!
import os
print("Running all tests...")
_ = [ok.grade(q[:-3]) for q in os.listdir("tests") if q.startswith('q') and␣
 ↪len(q) <= 10]
print("Finished running all tests.")
```