Annotated follow-along guide_ Confidence intervals in Python

June 13, 2023

1 Confidence intervals

Throughout the following exercises, you will learn to use Python to construct a confidence interval for a point estimate. Before starting on this programming exercise, we strongly recommend watching the video lecture and completing the IVQ for the associated topics.

All the information you need for solving this assignment is in this notebook, and all the code you will be implementing will take place within this notebook.

As we move forward, you can find instructions on how to install required libraries as they arise in this notebook. Before we begin with the exercises and analyzing the data, we need to import all libraries and extensions required for this programming exercise. Throughout the course, we will be using numpy, pandas, and scipy stats for operations.

[1]:
```
import numpy as np
import pandas as pd
from scipy import stats
```

[2]:
```
education_districtwise = pd.read_csv("education_districtwise.csv")
education_districtwise = education_districtwise.dropna()
```

We'll continue with our previous scenario, in which you're a data professional working for the Department of Education of a large nation. Earlier, we imagined that the Department of Education asked you to collect the data on district literacy rates. You were only able to survey 50 randomly chosen districts, instead of all 634 districts included in your original dataset. You used Python to simulate taking a random sample of 50 districts, and make a point estimate of the population mean, or literacy rate for *all* districts.

Now imagine that the department asks you to construct a 95% confidence interval for your estimate of the mean district literacy rate. You can use Python to construct the confidence interval.

You can also use the same sample data that you worked with earlier. Write the code to have Python simulate the same random sample of district literacy rate data. First, name your variable `sampled_data`. Then, enter the arguments of the `sample()` function.

- `n`: Your sample size is `50`.
- `replace`: Choose `True` because you are sampling with replacement.
- `random_state`: Choose the same random number to generate the same results–previously, you used `31,208`.

[3]:
```
sampled_data = education_districtwise.sample(n=50, replace=True,␣
↪random_state=31208)
sampled_data
```

[3]:

	DISTNAME	STATNAME	BLOCKS	VILLAGES	CLUSTERS	TOTPOPULAT	OVERALL_LI
661	DISTRICT528	STATE6	9	112	89	1863174.0	92.14
216	DISTRICT291	STATE28	14	1188	165	3273127.0	52.49
367	DISTRICT66	STATE23	12	1169	116	1042304.0	62.14
254	DISTRICT458	STATE3	3	157	19	82839.0	76.33
286	DISTRICT636	STATE35	3	187	44	514683.0	86.70
369	DISTRICT512	STATE23	6	589	30	717169.0	68.35
258	DISTRICT156	STATE3	6	80	9	35289.0	59.94
10	DISTRICT412	STATE1	11	187	95	476820.0	68.69
512	DISTRICT277	STATE9	10	558	179	2298934.0	84.31
144	DISTRICT133	STATE21	14	1672	136	3673849.0	69.61
325	DISTRICT1	STATE33	4	534	98	957853.0	69.37
227	DISTRICT159	STATE28	18	870	134	2954367.0	66.23
86	DISTRICT667	STATE25	5	396	75	896129.0	82.23
425	DISTRICT144	STATE31	7	1064	108	2662077.0	71.59
260	DISTRICT305	STATE3	2	62	6	145538.0	69.88
281	DISTRICT385	STATE35	6	531	30	354972.0	75.00
262	DISTRICT552	STATE3	3	103	4	111997.0	52.23
253	DISTRICT168	STATE3	5	312	16	176385.0	82.14
301	DISTRICT551	STATE14	9	103	63	693281.0	88.29
356	DISTRICT494	STATE34	25	2179	223	3596292.0	70.95
165	DISTRICT196	STATE21	10	1354	119	1795092.0	77.52
565	DISTRICT308	STATE17	8	721	144	848868.0	86.54
388	DISTRICT281	STATE23	6	392	58	949159.0	73.92
461	DISTRICT619	STATE22	5	859	57	1064989.0	68.36
384	DISTRICT455	STATE23	9	1217	55	1063458.0	68.85
590	DISTRICT70	STATE20	7	427	84	1846993.0	80.30
343	DISTRICT354	STATE33	2	192	46	1260419.0	88.66
539	DISTRICT440	STATE17	15	1465	167	2887826.0	88.23
459	DISTRICT431	STATE22	9	1778	143	2363744.0	73.42
667	DISTRICT123	STATE11	3	80	16	237586.0	88.49
387	DISTRICT231	STATE23	6	657	63	530299.0	64.51
306	DISTRICT37	STATE4	7	1083	92	642923.0	68.38
213	DISTRICT347	STATE28	11	623	94	2228397.0	59.65
97	DISTRICT22	STATE2	7	182	7	2531583.0	87.12
78	DISTRICT247	STATE25	7	314	60	1332042.0	72.73
394	DISTRICT640	STATE24	17	1857	191	1802777.0	69.00
184	DISTRICT596	STATE21	11	1281	108	2149066.0	51.76
147	DISTRICT335	STATE21	17	1945	138	4380793.0	69.44
542	DISTRICT489	STATE17	7	749	63	1198810.0	85.14
105	DISTRICT157	STATE13	14	1994	508	3671999.0	71.68
254	DISTRICT458	STATE3	3	157	19	82839.0	76.33
109	DISTRICT158	STATE13	6	769	211	1338114.0	66.19

609	DISTRICT17	STATE20	4	359	59	9588910.0	88.48
53	DISTRICT126	STATE26	3	197	21	596294.0	68.90
81	DISTRICT45	STATE25	9	351	130	1742815.0	73.24
516	DISTRICT300	STATE9	5	651	84	590379.0	73.29
641	DISTRICT484	STATE6	15	333	83	1721179.0	74.92
650	DISTRICT145	STATE6	11	489	100	1614069.0	84.09
70	DISTRICT99	STATE25	4	279	43	558890.0	83.44
163	DISTRICT366	STATE21	9	1330	86	1579160.0	79.99

The output shows 50 districts selected randomly from your dataset. Each has a different literacy rate.

1.1 Construct a 95% confidence interval

Now, construct a 95% confidence interval of the mean district literacy rate based on your sample data. Recall the four steps for constructing a confidence interval:

1. Identify a sample statistic
2. Choose a confidence level
3. Find the margin of error
4. Calculate the interval

1.1.1 `scipy.stats.norm.interval()`

Earlier, you worked through these steps one by one to construct a confidence interval. With Python, you can construct a confidence interval with just a single line of code–and get your results faster!

If you're working with a large sample size, say larger than 30, you can construct a confidence interval for the mean using `scipy.stats.norm.interval()`. This function includes the following arguments:

- `alpha`: The confidence level
- `loc`: The sample mean
- `scale`: The sample standard error

Reference: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.norm.html.

Let's explore each argument in more detail.

`alpha`: The confidence level The Department of Education requests a confidence level of 95%, which is the accepted standard for government funded research.

`loc`: The sample mean This is the mean literacy rate of your sample of 50 districts. Name a new variable `sample_mean`. Then, compute the mean district literacy rate for your sample data.

[4]:
```
sample_mean = sampled_data['OVERALL_LI'].mean()
```

`scale`: The sample standard error Recall that **standard error** measures the variability of your sample data. You may remember that the formula for the sample standard error is the sample standard deviation divided by the square root of the sample size.

Note: In practice, we typically don't know the true standard error, so we replace it with the estimated standard error.

You can write code to express the formula and have Python do the calculation for you:

1. Name a new variable `estimated_standard_error`.
2. Take the standard deviation of your sample data, and divide by the square root of your sample.
3. In parentheses, write the name of your data frame followed by the shape function and zero in brackets. Recall that the shape function returns the number of rows and columns in a data frame. `shape[0]` returns only the number of rows, which is the same number as your sample size.

[5]:
```
estimated_standard_error = sampled_data['OVERALL_LI'].std() / np.
↪sqrt(sampled_data.shape[0])
```

Now you're ready to put all this together to construct your confidence interval for the mean using `scipy.stats.norm.interval()`. First, write out the function and set the arguments:

- `alpha`: Enter `0.95` because you want to use a 95% confidence level
- `loc`: Enter the variable `sample_mean`
- `scale`: Enter the variable `estimated_standard_error`

[6]:
```
stats.norm.interval(alpha=0.95, loc=sample_mean, scale=estimated_standard_error)
```

[6]:
```
(71.42241096968617, 77.02478903031381)
```

You have a 95% confidence interval for the mean district literacy rate that stretches from about 71.4% to 77.0%.

95% CI: (71.42, 77.02)

The Department of Education will use your estimate of the mean district literacy rate to help make decisions about distributing funds to different states.

1.2 Construct a 99% confidence interval

Now imagine that a senior director in the department wants to be even *more* confident about your results. The director wants to make sure you have a reliable estimate, and suggests that you recalculate your interval with a 99% confidence level.

To compute a 99% confidence interval based on the same sample data, just change `alpha` to `0.99`.

[7]:
```
stats.norm.interval(alpha=0.99, loc=sample_mean, scale=estimated_standard_error)
```

[7]:
```
(70.54221358373107, 77.90498641626891)
```

You have a 99% confidence interval for the mean district literacy rate that stretches from about 70.5% to 77.9%.

99% CI: (70.54, 77.90)

1.2.1 Relationship between confidence level and confidence interval

You may notice that as the confidence *level* gets higher, the confidence *interval* gets wider.

- With a confidence level of 95%, the interval covers 5.6 percentage points (71.4% - 77.0%)
- With a confidence level of 99%, the interval covers 7.4 percentage points (70.5% - 77.9%)

This is because a wider confidence interval is more likely to include the actual population parameter.

Your results will help the Department of Education decide how to distribute government resources to improve literacy.

If you have successfully completed the material above, congratulations! You now understand how to use Python to construct a confidence interval for a point estimate. Going forward, you can start using Python to construct confidence intervals for your own data.