intro_confidenceintervals

October 12, 2020

0.1 Statistical Inference with Confidence Intervals

Throughout week 2, we have explored the concept of confidence intervals, how to calculate them, interpret them, and what confidence really means.

In this tutorial, we're going to review how to calculate confidence intervals of population proportions and means.

To begin, let's go over some of the material from this week and why confidence intervals are useful tools when deriving insights from data.

0.1.1 Why Confidence Intervals?

Confidence intervals are a calculated range or boundary around a parameter or a statistic that is supported mathematically with a certain level of confidence. For example, in the lecture, we estimated, with 95% confidence, that the population proportion of parents with a toddler that use a car seat for all travel with their toddler was somewhere between 82.2% and 87.7%.

This is ***different*** than having a 95% probability that the true population proportion is within our confidence interval.

Essentially, if we were to repeat this process, 95% of our calculated confidence intervals would contain the true proportion.

0.1.2 How are Confidence Intervals Calculated?

Our equation for calculating confidence intervals is as follows:

$$Best\ Estimate \pm Margin\ of\ Error$$

Where the *Best Estimate* is the **observed population proportion or mean** and the *Margin of Error* is the **t-multiplier**.

The t-multiplier is calculated based on the degrees of freedom and desired confidence level. For samples with more than 30 observations and a confidence level of 95%, the t-multiplier is 1.96

The equation to create a 95% confidence interval can also be shown as:

$$Population\ Proportion\ or\ Mean\ \pm (t - multiplier *\ Standard\ Error)$$

Lastly, the Standard Error is calculated differenly for population proportion and mean:

$$Standard\ Error\ for\ Population\ Proportion = \sqrt{\frac{Population\ Proportion * (1 - Population\ Proportion)}{Number\ Of\ Observations}}$$

$$Standard\ Error\ for\ Mean = \frac{Standard\ Deviation}{\sqrt{Number\ Of\ Observations}}$$

Let's replicate the car seat example from lecture:

```
In [2]: import numpy as np
```

```
In [3]: #variables to calculate confidence interval
        tstar = 1.96 #95% confidence interval
        p = .85  #population proportion
        n = 659  #sample size

        se = np.sqrt((p * (1 - p))/n)  #standard error for popn proportion
        se
```

```
Out[3]: 0.01390952774409444
```

0.1.3 calculate lower and upper bounds of confidence interval

```
In [4]: lcb = p - tstar * se  #lower confidence bound
        ucb = p + tstar * se  #upper confidence bound
        (lcb, ucb)
```

```
Out[4]: (0.8227373256215749, 0.8772626743784251)
```

lower bound: 82.2%, upper bound: 87.7%

1 Easier way to do this? Use statsmodels library

```
In [5]: import statsmodels.api as sm
```

```
In [6]: sm.stats.proportion_confint(n * p, n)
```

```
Out[6]: (0.8227378265796143, 0.8772621734203857)
```

We can see the confidence interval is the same as above with less code.

Now, lets take our Cartwheel dataset introduced in lecture and calculate a **confidence interval for our mean cartwheel distance:**

```
In [7]: import pandas as pd

        df = pd.read_csv("Cartwheeldata.csv")
```

```
In [8]: df.head()
```

```
Out[8]:    ID  Age Gender  GenderGroup Glasses  GlassesGroup  Height  Wingspan  \
        0   1   56      F            1       Y             1    62.0      61.0
        1   2   26      F            1       Y             1    62.0      60.0
        2   3   33      F            1       Y             1    66.0      64.0
```

```
3   4   39      F            1        N             0    64.0       63.0
4   5   27      M            2        N             0    73.0       75.0

   CWDistance Complete  CompleteGroup  Score
0          79        Y              1      7
1          70        Y              1      8
2          85        Y              1      7
3          87        Y              1     10
4          72        N              0      4
```

Next we will calculate: 1. mean 2. standard deviation 3. n

```
In [10]: mean = df["CWDistance"].mean()
         sd = df["CWDistance"].std()
         n = len(df)

In [11]: #mean
         mean

Out[11]: 82.48

In [12]: #standard deviation
         sd

Out[12]: 15.058552387264852

In [13]: #n of dataset
         n

Out[13]: 25
```

t multiplier is not going to be 1.96 because number of observations in dataset is less than 30 (it is 24), so the t table value is 2.064.

Can do this using scipy library.

```
In [15]: from scipy.stats import t

         alpha = 0.025 #95% confidence

         t.ppf(1 - alpha, df=24)

Out[15]: 2.0638985616280205
```

Can also do it this way:

```
In [17]: from scipy.special import stdtrit # stdtrit, it is the Student T DisTRibution functio

         alpha = 0.025 #95% confidence

         stdtrit(24, 1 - alpha)
```

```
Out[17]: 2.0638985616280205
```

calculate standard error for sample mean

```
In [18]: tstar = 2.064

         se = sd/np.sqrt(n)

         se
```

```
Out[18]: 3.0117104774529704
```

calculate lower and upper bounds.

```
In [19]: lcb = mean - tstar * se
         ucb = mean + tstar * se
         (lcb, ucb)
```

```
Out[19]: (76.26382957453707, 88.69617042546294)
```

1.0.1 Another way to calculate this using statsmodels library

```
In [20]: sm.stats.DescrStatsW(df["CWDistance"]).zconfint_mean()
```

```
Out[20]: (76.57715593233024, 88.38284406766977)
```

2 Now let's calculate this for wingspan variable

```
In [21]: df.head()
```

```
Out[21]:    ID  Age Gender  GenderGroup Glasses  GlassesGroup  Height  Wingspan  \
         0   1   56      F            1       Y             1    62.0      61.0
         1   2   26      F            1       Y             1    62.0      60.0
         2   3   33      F            1       Y             1    66.0      64.0
         3   4   39      F            1       N             0    64.0      63.0
         4   5   27      M            2       N             0    73.0      75.0

            CWDistance Complete  CompleteGroup  Score
         0          79        Y              1      7
         1          70        Y              1      8
         2          85        Y              1      7
         3          87        Y              1     10
         4          72        N              0      4
```

```
In [23]: df.Wingspan.value_counts()
```

```
Out[23]: 66.0    3
         60.0    2
         63.0    2
```

```
71.0    2
70.0    1
64.0    1
75.0    1
76.0    1
62.0    1
73.0    1
68.0    1
58.0    1
64.5    1
57.5    1
74.0    1
72.0    1
59.5    1
69.0    1
67.0    1
61.0    1
Name: Wingspan, dtype: int64
```

We can see there is a broad range of values so we will go ahead and perform confidence interval testing.

First we need the mean, sd, and n.

```
In [26]: mean2 = df["Wingspan"].mean()
         sd2 = df["Wingspan"].std()
         n2 = len(df)
```

```
In [27]: mean2
```

```
Out[27]: 66.26
```

```
In [28]: sd2
```

```
Out[28]: 5.492646599469755
```

```
In [29]: n2
```

```
Out[29]: 25
```

```
In [30]: from scipy.stats import t

         alpha = 0.025 #95% confidence

         t.ppf(1 - alpha, df=24)
```

```
Out[30]: 2.0638985616280205
```

calculate standard error

```
In [31]: tstar = 2.064

         se2 = sd2/np.sqrt(n2)

         se2

Out[31]: 1.098529319893951
```

calculate upper and lower bounds

```
In [43]: lcb = mean2 - tstar * se2
         ucb = mean2 + tstar * se2
         (lcb, ucb)

Out[43]: (63.99263548373889, 68.52736451626112)
```

Another way to calculate upper and lower bounds

```
In [44]: sm.stats.DescrStatsW(df["Wingspan"]).zconfint_mean()

Out[44]: (64.10692209704658, 68.41307790295343)
```

So we can say for the wingspan with 95% confidence the average winspance is 66.26 +/- 1.09 (64.11 to 68.41)

2.0.1 Let's calculate this for height

```
In [34]: df.Height.value_counts()

Out[34]: 65.00    3
         69.00    2
         70.00    2
         71.00    2
         61.50    2
         75.00    2
         73.00    2
         66.00    2
         62.00    2
         68.00    1
         62.75    1
         69.50    1
         63.00    1
         74.00    1
         64.00    1
         Name: Height, dtype: int64
```

Calculate mean, sd, n:

```
In [36]: mean3 = df['Height'].mean()
         sd3 = df['Height'].std()
         n = len(df)
```

```
In [37]: mean3
```

```
Out[37]: 67.65
```

```
In [38]: sd3
```

```
Out[38]: 4.43118682371514
```

```
In [39]: n
```

```
Out[39]: 25
```

```
In [40]: #t star calculation
         from scipy.stats import t

         alpha = 0.025 #95% confidence

         t.ppf(1 - alpha, df=24)
```

```
Out[40]: 2.0638985616280205
```

standard error calculation

```
In [42]: tstar = 2.064

         se3 = sd3/np.sqrt(n)

         se3
```

```
Out[42]: 0.8862373647430279
```

Upper and Lower Bounds

```
In [45]: lcb = mean3 - tstar * se3
         ucb = mean3 + tstar * se3
         (lcb, ucb)
```

```
Out[45]: (65.8208060791704, 69.47919392082962)
```

Another way to calculate upper and lower bounds

```
In [46]: sm.stats.DescrStatsW(df["Height"]).zconfint_mean()
```

```
Out[46]: (65.91300668334998, 69.38699331665003)
```

So we can say with 95% confidence the average height is 67.65cm +/- 0.89 (65.82 to 69.47)