Confidence_Intervals_Differences_Population_Parameters

October 12, 2020

1 Confidence Intervals

This tutorial is going to demonstrate how to load data, clean/manipulate a dataset, and construct a confidence interval for the difference between two population proportions and means.

We will use the 2015-2016 wave of the NHANES data for our analysis.

*Note: We have provided a notebook that includes more analysis, with examples of confidence intervals for one population proportions and means, in addition to the analysis I will show you in this tutorial. I highly recommend checking it out!

For our population proportions, we will analyze the difference of proportion between female and male smokers. The column that specifies smoker and non-smoker is "SMQ020" in our dataset.

For our population means, we will analyze the difference of mean of body mass index within our female and male populations. The column that includes the body mass index value is "BMXBMI".

Additionally, the gender is specified in the column "RIAGENDR".

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib
        matplotlib.use('Agg')
        import seaborn as sns
        %matplotlib inline
        import matplotlib.pyplot as plt
        import statsmodels.api as sm
```

```
In [2]: url = "nhanes_2015_2016.csv"
        da = pd.read_csv(url)
```

1.0.1 Investigating and Cleaning Data

SMQ020x describes smokers vs. non smokers. - 1 = Yes smoker - 2 = Non smoking - 7 = occasionally smoke - 9 = mostly smoke We will change these values an create a new column. We will omit the 7 and 9 and make then nan values.

```
In [3]: # Recode SMQ020 from 1/2 to Yes/No into new variable SMQ020x
        da["SMQ020x"] = da.SMQ020.replace({1: "Yes", 2: "No", 7: np.nan, 9: np.nan})
        da["SMQ020x"]
```

Out[3]:
```
0       Yes
1       Yes
2       Yes
3        No
4        No
5        No
6       Yes
7        No
8        No
9        No
10      Yes
11      Yes
12      Yes
13       No
14       No
15       No
16       No
17       No
18      Yes
19       No
20       No
21       No
22      Yes
23       No
24       No
25       No
26      Yes
27      Yes
28       No
29       No
        ...
5705    Yes
5706    Yes
5707     No
5708     No
5709    Yes
5710     No
5711    Yes
5712     No
5713     No
5714     No
5715     No
5716    Yes
5717    Yes
5718     No
5719    Yes
5720     No
5721     No
```

```
5722     No
5723    Yes
5724     No
5725     No
5726    Yes
5727     No
5728     No
5729     No
5730    Yes
5731     No
5732    Yes
5733    Yes
5734     No
Name: SMQ020x, Length: 5735, dtype: object
```

We will also change the RIAGENDR variable from 1,2 to Male, Female.

```
In [4]: # Recode RIAGENDR from 1/2 to Male/Female into new variable RIAGENDRx
        da["RIAGENDRx"] = da.RIAGENDR.replace({1: "Male", 2: "Female"})
        da["RIAGENDRx"]
```

```
Out[4]: 0         Male
        1         Male
        2         Male
        3       Female
        4       Female
        5       Female
        6         Male
        7       Female
        8         Male
        9         Male
        10        Male
        11        Male
        12      Female
        13      Female
        14        Male
        15      Female
        16      Female
        17      Female
        18      Female
        19      Female
        20        Male
        21      Female
        22      Female
        23      Female
        24        Male
        25      Female
        26        Male
```

```
27      Female
28        Male
29      Female
         ...
5705      Male
5706      Male
5707    Female
5708    Female
5709      Male
5710    Female
5711      Male
5712    Female
5713      Male
5714      Male
5715    Female
5716    Female
5717      Male
5718      Male
5719    Female
5720      Male
5721    Female
5722    Female
5723    Female
5724    Female
5725      Male
5726      Male
5727    Female
5728      Male
5729      Male
5730    Female
5731      Male
5732    Female
5733      Male
5734    Female
Name: RIAGENDRx, Length: 5735, dtype: object
```

Now we will make a crosstab table to compute a frequency table of the smoking vs. non smoking for males vs. females.

```
In [5]: dx = da[['SMQ020x', 'RIAGENDRx']].dropna()
        pd.crosstab(dx.SMQ020x, dx.RIAGENDRx)
```

```
Out[5]: RIAGENDRx  Female  Male
        SMQ020x
        No           2066  1340
        Yes           906  1413
```

We will now typecast the smoking column back to its respective 1 and 0 so we can perform statistical analysis.

In [16]:
```
# Recode SMQ020x from Yes/No to 1/0 into existing variable SMQ020x
#dx["SMQ020x"] = dx.SMQ020x.replace({"Yes": 1, "No": 0})
dx
```

Out[16]:

	SMQ020x	RIAGENDRx
0	1	Male
1	1	Male
2	1	Male
3	0	Female
4	0	Female
5	0	Female
6	1	Male
7	0	Female
8	0	Male
9	0	Male
10	1	Male
11	1	Male
12	1	Female
13	0	Female
14	0	Male
15	0	Female
16	0	Female
17	0	Female
18	1	Female
19	0	Female
20	0	Male
21	0	Female
22	1	Female
23	0	Female
24	0	Male
25	0	Female
26	1	Male
27	1	Female
28	0	Male
29	0	Female
...	...	...
5705	1	Male
5706	1	Male
5707	0	Female
5708	0	Female
5709	1	Male
5710	0	Female
5711	1	Male
5712	0	Female
5713	0	Male
5714	0	Male
5715	0	Female
5716	1	Female

```
5717         1      Male
5718         0      Male
5719         1    Female
5720         0      Male
5721         0    Female
5722         0    Female
5723         1    Female
5724         0    Female
5725         0      Male
5726         1      Male
5727         0    Female
5728         0      Male
5729         0      Male
5730         1    Female
5731         0      Male
5732         1    Female
5733         1      Male
5734         0    Female

[5725 rows x 2 columns]
```

```
In [17]: dz = dx.groupby("RIAGENDRx").agg({"SMQ020x": [np.mean, np.size]})
         dz.columns = ["Proportion", "Total n"]
         dz
```

Out[17]:

RIAGENDRx	Proportion	Total n
Female	0.304845	2972
Male	0.513258	2753

1.0.2 Constructing Confidence Intervals

Now that we have the population proportions of male and female smokers, we can begin to calculate confidence intervals. From lecture, we know that the equation is as follows:

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

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

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

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

The Standard Error (SE) is calculated differenly for population proportion and mean:

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

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

Lastly, the standard error for difference of population proportions and means is:

$$Standard\ Error\ for\ Difference\ of\ Two\ Population\ Proportions\ Or\ Means = \sqrt{(SE_1)^2 + (SE_2)^2}$$

Difference of Two Population Proportions

```
In [18]: #female calculations
         p = .304845 #percent that are female and smokers
         n = 2972    #number of females
         se_female = np.sqrt(p * (1 - p)/n)
         se_female

Out[18]: 0.00844415041930423

In [19]: #male calculations
         p = .513258
         n = 2753
         se_male = np.sqrt(p * (1 - p)/ n)
         se_male

Out[19]: 0.009526078787008965

In [21]: #standard error difference
         se_diff = np.sqrt(se_female**2 + se_male**2)
         se_diff

Out[21]: 0.012729880335656654

In [22]: #confidence bounds
         d = .304845 - .513258 #difference of population proportions
         lcb = d - 1.96 * se_diff
         ucb = d + 1.96 * se_diff
         (lcb, ucb)

Out[22]: (-0.23336356545788706, -0.18346243454211297)
```

Difference of Two Population Means

```
In [23]: da["BMXBMI"].head()

Out[23]: 0    27.8
         1    30.8
         2    28.8
         3    42.4
         4    20.3
         Name: BMXBMI, dtype: float64
```

```
In [24]: da.groupby("RIAGENDRx").agg({"BMXBMI": [np.mean, np.std, np.size]})
```

```
Out[24]:              BMXBMI
                        mean       std    size
         RIAGENDRx
         Female     29.939946  7.753319  2976.0
         Male       28.778072  6.252568  2759.0
```

```
In [25]: sem_female = 7.753319 / np.sqrt(2976)
         sem_male = 6.252568 / np.sqrt(2759)
         (sem_female, sem_male)
```

```
Out[25]: (0.14212523289878048, 0.11903716451870151)
```

```
In [26]: sem_diff = np.sqrt(sem_female**2 + sem_male**2)
         sem_diff
```

```
Out[26]: 0.18538993598139303
```

```
In [29]: d = 29.939946 - 28.778072
         d
```

```
Out[29]: 1.1618739999999974
```

```
In [28]: lcb = d - 1.96 * sem_diff
         ucb = d + 1.96 * sem_diff
         (lcb, ucb)
```

```
Out[28]: (0.798509725476467, 1.5252382745235278)
```

95% confident that the BMI index for males and females is between .79 and 1.52