Confidence_Intervals_Differences_Population_Parameters

February 10, 2022

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

```
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
```

```
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
```

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

```
Out[5]:      SMQ020x RIAGENDRx
        0        Yes      Male
        1        Yes      Male
        2        Yes      Male
        3         No    Female
        4         No    Female
        5         No    Female
        6        Yes      Male
        7         No    Female
        8         No      Male
        9         No      Male
        10       Yes      Male
        11       Yes      Male
        12       Yes    Female
        13        No    Female
```

```
14         No      Male
15         No    Female
16         No    Female
17         No    Female
18        Yes    Female
19         No    Female
20         No      Male
21         No    Female
22        Yes    Female
23         No    Female
24         No      Male
25         No    Female
26        Yes      Male
27        Yes    Female
28         No      Male
29         No    Female
...       ...       ...
5705      Yes      Male
5706      Yes      Male
5707       No    Female
5708       No    Female
5709      Yes      Male
5710       No    Female
5711      Yes      Male
5712       No    Female
5713       No      Male
5714       No      Male
5715       No    Female
5716      Yes    Female
5717      Yes      Male
5718       No      Male
5719      Yes    Female
5720       No      Male
5721       No    Female
5722       No    Female
5723      Yes    Female
5724       No    Female
5725       No      Male
5726      Yes      Male
5727       No    Female
5728       No      Male
5729       No      Male
5730      Yes    Female
5731       No      Male
5732      Yes    Female
5733      Yes      Male
5734       No    Female
```

```
        [5725 rows x 2 columns]

In [6]: pd.crosstab(dx.SMQ020x, dx.RIAGENDRx)

Out[6]: RIAGENDRx  Female  Male
        SMQ020x
        No           2066  1340
        Yes           906  1413

In [9]: # Recode SMQ020x from Yes/No to 1/0 into existing variable SMQ020x
        dx["SMQ020x"] = dx.SMQ020x.replace({"Yes": 1, "No": 0})


        ---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call last)

        <ipython-input-9-0a796a728ba0> in <module>()
          1 # Recode SMQ020x from Yes/No to 1/0 into existing variable SMQ020x
    ----> 2 dx["SMQ020x"] = dx.SMQ020x.replace({"Yes": 1, "No": 0})


        /opt/conda/lib/python3.6/site-packages/pandas/core/series.py in replace(self, to_replac
       3839         return super(Series, self).replace(to_replace=to_replace, value=value,
       3840                                            inplace=inplace, limit=limit,
    -> 3841                                            regex=regex, method=method)
       3842
       3843     @Appender(generic._shared_docs['shift'] % _shared_doc_kwargs)


        /opt/conda/lib/python3.6/site-packages/pandas/core/generic.py in replace(self, to_repla
       6496
       6497             return self.replace(to_replace, value, inplace=inplace,
    -> 6498                                 limit=limit, regex=regex)
       6499         else:
       6500


        /opt/conda/lib/python3.6/site-packages/pandas/core/series.py in replace(self, to_replac
       3839         return super(Series, self).replace(to_replace=to_replace, value=value,
       3840                                            inplace=inplace, limit=limit,
    -> 3841                                            regex=regex, method=method)
       3842
       3843     @Appender(generic._shared_docs['shift'] % _shared_doc_kwargs)


        /opt/conda/lib/python3.6/site-packages/pandas/core/generic.py in replace(self, to_repla
       6545                                                          dest_list=value,
       6546                                                          inplace=inplace,
    -> 6547                                                           regex=regex)
       6548
       6549                     else:  # [NA, ''] -> 0


        /opt/conda/lib/python3.6/site-packages/pandas/core/internals/managers.py in replace_li
        557                 return _compare_or_regex_match(values, s, regex)
        558
    --> 559             masks = [comp(s, regex) for i, s in enumerate(src_list)]
        560
        561             result_blocks = []


        /opt/conda/lib/python3.6/site-packages/pandas/core/internals/managers.py in <listcomp>
        557                 return _compare_or_regex_match(values, s, regex)
        558
    --> 559             masks = [comp(s, regex) for i, s in enumerate(src_list)]
        560
        561             result_blocks = []


        /opt/conda/lib/python3.6/site-packages/pandas/core/internals/managers.py in comp(s, re
        555                     return _compare_or_regex_match(maybe_convert_objects(values),
        556                                                    getattr(s, 'asm8'), regex)
    --> 557                 return _compare_or_regex_match(values, s, regex)
        558
        559             masks = [comp(s, regex) for i, s in enumerate(src_list)]


        /opt/conda/lib/python3.6/site-packages/pandas/core/internals/managers.py in _compare_o
       1949             raise TypeError(
       1950                 "Cannot compare types {a!r} and {b!r}".format(a=type_names[0],
    -> 1951                                                                b=type_names[1]))
       1952         return result
       1953


        TypeError: Cannot compare types 'ndarray(dtype=int64)' and 'str'
```

```
In [10]: dx
```

```
Out[10]:       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 [11]: dz = dx.groupby("RIAGENDRx").agg({"SMQ020x": [np.mean, np.size]})
         dz
```

```
Out[11]:             SMQ020x
                        mean  size
         RIAGENDRx
         Female     0.304845  2972
         Male       0.513258  2753
```

```
In [12]: dz.columns = ["Proportion", "Total n"]
         dz
```

```
Out[12]:            Proportion  Total n
         RIAGENDRx
         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 [13]: p = .304845
         n = 2972
         se_female = np.sqrt(p * (1 - p)/n)
         se_female

Out[13]: 0.00844415041930423
```

```
In [14]: p = .513258
         n = 2753
         se_male = np.sqrt(p * (1 - p)/ n)
         se_male

Out[14]: 0.009526078787008965
```

```
In [15]: se_diff = np.sqrt(se_female**2 + se_male**2)
         se_diff

Out[15]: 0.012729880335656654
```

```
In [16]: # no calculo do intervalo de confianca, usar o standard erro da diferenca (se_diff) e
         d = .304845 - .513258
         lcb = d - 1.96 * se_diff
         ucb = d + 1.96 * se_diff
         (lcb, ucb)

Out[16]: (-0.23336356545788706, -0.18346243454211297)
```

Difference of Two Population Means

```
In [18]: da["BMXBMI"].head()

Out[18]: 0    27.8
         1    30.8
         2    28.8
         3    42.4
         4    20.3
         Name: BMXBMI, dtype: float64
```

```
In [19]: da.groupby("RIAGENDRx").agg({"BMXBMI": [np.mean, np.std, np.size]})

Out[19]:               BMXBMI
                         mean       std    size
         RIAGENDRx
         Female     29.939946  7.753319  2976.0
         Male       28.778072  6.252568  2759.0
```

```
In [20]: sem_female = 7.753319 / np.sqrt(2976)
         sem_male = 6.252568 / np.sqrt(2759)
         (sem_female, sem_male)
```

```
Out[20]: (0.14212523289878048, 0.11903716451870151)

In [21]: sem_diff = np.sqrt(sem_female**2 + sem_male**2)
         sem_diff

Out[21]: 0.18538993598139303

In [24]: d = 29.939946 - 28.778072
         d

Out[24]: 1.1618739999999974

In [25]: lcb = d - 1.96 * sem_diff
         ucb = d + 1.96 * sem_diff
         (lcb, ucb)

Out[25]: (0.798509725476467, 1.5252382745235278)
```