nhanes_confidence_intervals_practice

February 20, 2022

1 Practice notebook for confidence intervals using NHANES data

This notebook will give you the opportunity to practice working with confidence intervals using the NHANES data.

You can enter your code into the cells that say "enter your code here", and you can type responses to the questions into the cells that say "Type Markdown and Latex".

Note that most of the code that you will need to write below is very similar to code that appears in the case study notebook. You will need to edit code from that notebook in small ways to adapt it to the prompts below.

To get started, we will use the same module imports and read the data in the same way as we did in the case study:

In [2]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
import statsmodels.api as sm


da = pd.read_csv("nhanes_2015_2016.csv")
#pd.set_option('display.max_rows', da.shape[0]+1)
pd.set_option('display.max_rows', 100)
```

In [3]:
```
# woman
df_woman = da[da["RIAGENDR"] == 2]
df_woman.count()
```

Out[3]:
```
SEQN        2976
ALQ101      2660
ALQ110      1203
ALQ130      1577
SMQ020      2976
RIAGENDR    2976
RIDAGEYR    2976
RIDRETH1    2976
DMDCITZN    2975
```

```
DMDEDUC2    2850
DMDMARTL    2850
DMDHHSIZ    2976
WTINT2YR    2976
SDMVPSU     2976
SDMVSTRA    2976
INDFMPIR    2651
BPXSY1      2780
BPXDI1      2780
BPXSY2      2857
BPXDI2      2857
BMXWT       2947
BMXHT       2946
BMXBMI      2944
BMXLEG      2753
BMXARML     2805
BMXARMC     2805
BMXWAIST    2762
HIQ210      2504
dtype: int64
```

```
In [4]: # woman
        df_woman = da[da["RIAGENDR"] == 2]
        df_woman.count()
```

```
Out[4]: SEQN        2976
        ALQ101      2660
        ALQ110      1203
        ALQ130      1577
        SMQ020      2976
        RIAGENDR    2976
        RIDAGEYR    2976
        RIDRETH1    2976
        DMDCITZN    2975
        DMDEDUC2    2850
        DMDMARTL    2850
        DMDHHSIZ    2976
        WTINT2YR    2976
        SDMVPSU     2976
        SDMVSTRA    2976
        INDFMPIR    2651
        BPXSY1      2780
        BPXDI1      2780
        BPXSY2      2857
        BPXDI2      2857
        BMXWT       2947
        BMXHT       2946
        BMXBMI      2944
```

```
BMXLEG      2753
BMXARML     2805
BMXARMC     2805
BMXWAIST    2762
HIQ210      2504
dtype: int64
```

```
In [5]: # checks
        higher_eq_35_query = df_woman[df_woman['RIDAGEYR'] >= 35]
        higher_eq_35_query["RIDAGEYR"].count()
```

```
Out[5]: 2108
```

```
In [6]: higher_eq_35_slice = df_woman.query('RIDAGEYR >= 35')
        higher_eq_35_slice["RIDAGEYR"].count()
```

```
Out[6]: 2108
```

```
In [7]: higher_50_slice = df_woman.query('RIDAGEYR > 50')
        higher_50_slice["RIDAGEYR"].count()
```

```
Out[7]: 1321
```

```
In [8]: range_35_50 = higher_eq_35_slice["RIDAGEYR"].count() - higher_50_slice["RIDAGEYR"].cou
        range_35_50
```

```
Out[8]: 787
```

```
In [9]: range_35_50_query = df_woman.query('RIDAGEYR >=35 and RIDAGEYR <= 50')
        range_35_50_query = range_35_50_query [["RIDAGEYR", "DMDMARTL", "DMDEDUC2"]].dropna()
        range_35_50_query["RIDAGEYR"].count()
```

```
Out[9]: 787
```

1.1 Question 1

Restrict the sample to women between 35 and 50, then use the marital status variable DMDMARTL to partition this sample into two groups - women who are currently married, and women who are not currently married (married = 1, Missing = ., Dont know = 99, Refused = 77). Within each of these groups, calculate the proportion of women who have completed college (variable DMDEDUC2). Calculate 95% confidence intervals for each of these proportions. ich.edu/web/DSDR/studies/25504/datasets/0232/variables/DMDEDUC2?archive=DSDR

DMDEDUC2: 1 Less Than 9th Grade, 2 9-11th Grade (Includes 12th grade with no diploma), 3 High School Grad/GED or Equivalent, 4 Some College or AA degree, 5 College Graduate or above, 7 Refused, 9 Don't know, . Missing values https://www.icpsr.um

```
In [10]: # selecting women in the age range 35-50
         df_women_35_50 = df_woman.query('RIDAGEYR >=35 and RIDAGEYR <= 50')
         df_women_35_50 = df_women_35_50 [["RIDAGEYR", "DMDMARTL", "DMDEDUC2"]].dropna()
         df_women_35_50.count()
```

```
Out[10]: RIDAGEYR    787
         DMDMARTL    787
         DMDEDUC2    787
         dtype: int64
```

```
In [11]: # added colum EDULEVEL to store if has college educational level
         df_women_35_50 = df_women_35_50.query('DMDEDUC2 != 7 and DMDEDUC2 != 9 and DMDEDUC2 !=
         df_women_35_50["HAS_COLLEGE"]=df_women_35_50["DMDEDUC2"] == 5
         df_women_35_50
```

Out[11]:

	RIDAGEYR	DMDMARTL	DMDEDUC2	HAS_COLLEGE
4	42	3.0	4.0	False
34	37	1.0	4.0	False
50	39	1.0	3.0	False
52	50	4.0	1.0	False
55	45	1.0	2.0	False
58	44	5.0	1.0	False
61	37	1.0	3.0	False
62	49	1.0	3.0	False
63	46	1.0	5.0	True
76	42	1.0	5.0	True
82	38	1.0	4.0	False
95	47	1.0	4.0	False
98	49	1.0	4.0	False
100	43	6.0	4.0	False
106	48	1.0	1.0	False
114	44	1.0	5.0	True
124	46	1.0	5.0	True
127	47	3.0	3.0	False
129	44	1.0	4.0	False
131	44	3.0	4.0	False
147	37	5.0	2.0	False
150	41	6.0	3.0	False
159	45	6.0	2.0	False
166	41	1.0	5.0	True
178	41	3.0	4.0	False
186	45	1.0	5.0	True
192	50	1.0	4.0	False
193	45	1.0	4.0	False
194	48	4.0	4.0	False
199	39	5.0	3.0	False
202	46	5.0	3.0	False
206	45	1.0	4.0	False
234	46	4.0	4.0	False
241	35	5.0	1.0	False
244	46	1.0	1.0	False
290	43	5.0	1.0	False
294	40	5.0	2.0	False

295	47	5.0	4.0	False
300	42	1.0	5.0	True
301	43	1.0	3.0	False
318	44	1.0	4.0	False
321	36	1.0	3.0	False
326	36	1.0	5.0	True
331	36	5.0	3.0	False
332	41	1.0	5.0	True
335	45	1.0	5.0	True
346	50	1.0	4.0	False
348	39	3.0	4.0	False
355	36	1.0	1.0	False
357	48	4.0	2.0	False
...	...	...	...	...
5256	36	3.0	4.0	False
5272	49	1.0	5.0	True
5299	50	5.0	5.0	True
5324	38	5.0	4.0	False
5338	39	1.0	4.0	False
5359	40	1.0	5.0	True
5363	42	1.0	5.0	True
5364	36	1.0	3.0	False
5366	49	4.0	1.0	False
5385	44	2.0	5.0	True
5386	49	3.0	5.0	True
5392	48	1.0	5.0	True
5396	45	1.0	4.0	False
5404	40	3.0	3.0	False
5410	47	6.0	4.0	False
5442	46	5.0	3.0	False
5444	44	1.0	4.0	False
5445	38	6.0	4.0	False
5446	47	1.0	2.0	False
5455	44	6.0	4.0	False
5475	46	3.0	3.0	False
5476	37	5.0	5.0	True
5481	48	1.0	3.0	False
5488	46	1.0	1.0	False
5489	35	1.0	5.0	True
5495	40	5.0	5.0	True
5497	40	1.0	2.0	False
5499	46	5.0	4.0	False
5509	39	1.0	4.0	False
5514	43	1.0	4.0	False
5530	43	1.0	3.0	False
5536	36	5.0	5.0	True
5540	40	1.0	5.0	True
5552	35	5.0	4.0	False

```
5556        50       1.0       5.0         True
5557        36       4.0       2.0        False
5558        47       1.0       2.0        False
5559        50       3.0       4.0        False
5575        43       1.0       5.0         True
5582        36       5.0       3.0        False
5598        44       1.0       1.0        False
5606        42       1.0       4.0        False
5609        47       1.0       4.0        False
5623        40       5.0       3.0        False
5627        43       1.0       4.0        False
5658        46       3.0       2.0        False
5685        36       1.0       5.0         True
5689        44       1.0       2.0        False
5721        35       3.0       5.0         True
5724        41       1.0       5.0         True

[787 rows x 4 columns]
```

In [12]:
```
# added colum MARITAL_STATUS to store if is married or not
df_women_35_50= df_women_35_50.query('DMDMARTL != 77 and DMDMARTL != 99 and DMDMARTL
df_women_35_50["IS_MARRIED"]= df_women_35_50["DMDMARTL"] == 1
df_women_35_50.head()
```

Out[12]:

	RIDAGEYR	DMDMARTL	DMDEDUC2	HAS_COLLEGE	IS_MARRIED
4	42	3.0	4.0	False	False
34	37	1.0	4.0	False	True
50	39	1.0	3.0	False	True
52	50	4.0	1.0	False	False
55	45	1.0	2.0	False	True

In [13]:
```
df_women_35_50["MARITAL_STATUS"] = df_women_35_50.IS_MARRIED.replace({True: "Married"
df_women_35_50["EDUC_LEVEL"] = df_women_35_50.HAS_COLLEGE.replace({True: "College", Fa
```

In [14]:
```
df_women_35_50.count()
```

Out[14]:
```
RIDAGEYR          787
DMDMARTL          787
DMDEDUC2          787
HAS_COLLEGE       787
IS_MARRIED        787
MARITAL_STATUS    787
EDUC_LEVEL        787
dtype: int64
```

In [15]:
```
counts = pd.crosstab(df_women_35_50.EDUC_LEVEL, df_women_35_50.MARITAL_STATUS)
counts
```

Out[15]:
```
MARITAL_STATUS  Married  Single
EDUC_LEVEL
```

```
College             162      72
Other               287     266
```

In [16]:
```
df_women_35_50.head(10)
```

Out[16]:
```
     RIDAGEYR  DMDMARTL  DMDEDUC2  HAS_COLLEGE  IS_MARRIED MARITAL_STATUS  \
4          42       3.0       4.0        False       False         Single
34         37       1.0       4.0        False        True        Married
50         39       1.0       3.0        False        True        Married
52         50       4.0       1.0        False       False         Single
55         45       1.0       2.0        False        True        Married
58         44       5.0       1.0        False       False         Single
61         37       1.0       3.0        False        True        Married
62         49       1.0       3.0        False        True        Married
63         46       1.0       5.0         True        True        Married
76         42       1.0       5.0         True        True        Married

   EDUC_LEVEL
4       Other
34      Other
50      Other
52      Other
55      Other
58      Other
61      Other
62      Other
63    College
76    College
```

In [17]:
```
# Proportion married and not married
#           total, college, not college
#married     34,   17,    17
#not married 27,   9,     18

proportions = df_women_35_50.groupby(df_women_35_50.MARITAL_STATUS).agg({"EDUC_LEVEL"
proportions.columns = ["With_College", "Total_n"]
proportions
```

Out[17]:
```
                With_College  Total_n
MARITAL_STATUS
Married             0.360802      449
Single              0.213018      338
```

In [18]:
```
# Women single

# standard error
p_single = proportions.With_College.Single # Women Single proportion
n_single = proportions.Total_n.Single # Total number of females
se_single = np.sqrt(p_single * (1 - p_single) / n_single)
```

```
        # confidence interval
        lcb = p_single - 1.96 * np.sqrt(p_single * (1 - p_single) / n_single)
        ucb = p_single + 1.96 * np.sqrt(p_single * (1 - p_single) / n_single)
        ci_range = ucb - lcb

        print('single:', se_single, ' - confidence interval (', lcb, ',', ucb, ') - range:', c

        # Women married

        # standard error
        p_married = proportions.With_College.Married
        n_married = proportions.Total_n.Married # Total number of females
        se_married = np.sqrt(p_married * (1 - p_married) / n_married)

        # confidence interval
        lcb = p_married - 1.96 * np.sqrt(p_married * (1 - p_married) / n_married)
        ucb = p_married + 1.96 * np.sqrt(p_married * (1 - p_married) / n_married)
        ci_range = ucb - lcb
        print('married:', se_married, ' - confidence interval(', lcb, ',',  ucb, ') - range:'
```

```
single: 0.022270605048202215  - confidence interval ( 0.1693673655848136 , 0.25666813737376626
married: 0.02266360248455356  - confidence interval( 0.3163811208674688 , 0.4052224426069187 )
```

```
In [19]: # same calcs using the lib statsmodels

         # Women single
         sm.stats.proportion_confint(counts.Single.College , n_single)
```

```
Out[19]: (0.16936816767089768, 0.2566673352876822)
```

```
In [20]: # Women married
         sm.stats.proportion_confint(counts.Married.College , n_married)
```

```
Out[20]: (0.31638193710753626, 0.4052216263668512)
```

Q1a. Identify which of the two confidence intervals is wider, and explain why this is the case.
The married woman confidence interval is slightly wider.

Q1b. Write 1-2 sentences summarizing these findings for an audience that does not know what a confidence interval is (the goal here is to report the substance of what you learned about how marital status and educational attainment are related, not to teach a person what a confidence interval is).

With 95% confident that, approximatelly, between 32% and 41% of married woman completed a college degree Whereas between 17% and 26% of unmarried woman completed a colledge degree

1.2 Question 2

Construct a 95% confidence interval for the proportion of smokers who are female. Construct a 95% confidence interval for the proportion of smokers who are male. Construct a 95% confidence interval for the **difference** between those two gender proportions.

```
In [44]: # woman
         df = da[["RIAGENDR","SMQ020" ]]
         df["GENDER"] = df.RIAGENDR.replace({1: "Male", 2: "Female"})
         df["SMOKER"] = df.SMQ020.replace({1: "Yes", 2: "No", 7: np.nan, 9: np.nan})
         df = df[["SMOKER", "GENDER"]].dropna()
         df.head()
```

```
Out[44]:   SMOKER  GENDER
         0    Yes    Male
         1    Yes    Male
         2    Yes    Male
         3     No  Female
         4     No  Female
```

```
In [45]: df.count()
```

```
Out[45]: SMOKER    5725
         GENDER    5725
         dtype: int64
```

```
In [47]: proportions = df.groupby(df.GENDER).agg({"SMOKER": [lambda x: np.mean(x=="Yes"), np.s
         proportions.columns = ["Proportion", "Total_n"]
         proportions
```

```
Out[47]:         Proportion  Total_n
         GENDER
         Female    0.304845     2972
         Male      0.513258     2753
```

```
In [51]: # Construct a 95% confidence interval for the proportion of smokers who are female
         # standard error
         p_women = proportions.Proportion.Female # Women smokers proportion
         n_women = proportions.Total_n.Female # Total number of females
         se_women = np.sqrt(p_women * (1 - p_women) / n_women)

         # confidence interval
         lcb = p_women - 1.96 * np.sqrt(p_women * (1 - p_women) / n_women)
         ucb = p_women + 1.96 * np.sqrt(p_women * (1 - p_women) / n_women)
         ci_range = ucb - lcb

         print('women:', se_women, ' - confidence interval (', lcb, ',', ucb, ') - range:', ci

         # Construct a 95% confidence interval for the proportion of smokers who are male
```

```
        # standard error
        p_men = proportions.Proportion.Male # Males smokers proportion
        n_men = proportions.Total_n.Male # Total number of males
        se_men = np.sqrt(p_men * (1 - p_men) / n_men)

        # confidence interval
        lcb = p_men - 1.96 * np.sqrt(p_men * (1 - p_men) / n_men)
        ucb = p_men + 1.96 * np.sqrt(p_men * (1 - p_men) / n_men)
        ci_range = ucb - lcb
        print('males:', se_men, ' - confidence interval(', lcb, ',',  ucb, ') - range:', ci_ra
```

```
women: 0.008444152146214435  - confidence interval ( 0.288294683866098 , 0.32139576027925865 )
males: 0.009526078653689868  - confidence interval( 0.49458714955108174 , 0.531929377873546 ) -
```

```
In [59]: # Construct a 95% confidence interval for the difference between those two gender pro
         # standard error of the difference
         # sqrt(SE1^2 + SE2^2) is the standard error for the difference of these proportions
         se_diff = np.sqrt(se_women**2 + se_men**2)
         se_diff

         # confidence interval
         diff = proportions.Proportion.Male -  proportions.Proportion.Female  # difference bet
         lcb = diff - 2*se_diff
         ucb = diff + 2*se_diff
         ci_range = ucb - lcb

         print('diff:', 'std error ', se_diff, ' - confidence interval(', lcb, ',',  ucb, ') -
```

```
diff: std error  0.012729881381407434  - confidence interval( 0.18295327887682067 , 0.233872804
```

The 95% confidence interval above shows us that any value for the difference of population proportions (between females and males) lying between 0,1829 and 0.2338.

Q2a. Why might it be relevant to report the separate gender proportions **and** the difference between the gender proportions?

A confidence interval (C.I.) for a difference in proportions is a range of values that is likely to contain the true difference between two population proportions with a certain level of confidence. The confidence intervals for the proportions of female and male smokers shown above are quite narrow and do not overlap. This suggests that there is a substantial difference between the lifetime smoking rates for women and men. However there is no explicit information here about how different the two population proportions might be. To address this question, we can form a confidence interval for the difference between the proportion of females who smoke and the proportion of males who smoke.

Q2b. How does the **width** of the confidence interval for the difference between the gender proportions compare to the widths of the confidence intervals for the separate gender proportions?

It is wider and contains zero. Since the interval does not contain 0, we see that the difference seen in this study was "significant."

1.3 Question 3

Construct a 95% interval for height (BMXHT) in centimeters. Then convert height from centimeters to inches by dividing by 2.54, and construct a 95% confidence interval for height in inches. Finally, convert the endpoints (the lower and upper confidence limits) of the confidence interval from inches to back to centimeters

```
In [ ]: # centimeters
        ddof=1mean_cm = da_bmxht_cm.mean()
        std_cm = da_bmxht_cm.std(ddof=1)

        # inches
        da_bmxht_inches = da_bmxht_cm.apply(lambda x: x/2.54)
        mean_inches = da_bmxht_inches.mean()
        std_inches = da_bmxht_inches.std(ddof=1)

        # common variables
        n = da_bmxht_inches.count()
        z = 1.96
```

```
In [ ]: # centimeters
        std_error_inches = std_inches / np.sqrt(n)

        # inches
        std_error_cm = std_cm / np.sqrt(n)
```

```
In [ ]: # inches
        lcb_inches = mean_inches - z * std_error_inches
        ucb_inches = mean_inches + z * std_error_inches
        print(lcb_inches, ucb_inches)
        print('With 95% confidence, the population mean is estimaded to be between',lcb_inches
```

```
In [ ]: # inches to centimeters
        lcb_inches_to_cm = lcb_inches * 2.54
        ucb_inches_to_cm = ucb_inches * 2.54
        print('With 95% confidence, the population mean is estimaded to be between',lcb_inches_
```

```
In [ ]: # centimeters
        lcb_cm = mean_cm - z * std_error_cm
        ucb_cm = mean_cm + z * std_error_cm
        print('With 95% confidence, the population mean is estimaded to be between',lcb_cm, 'a
```

```
In [ ]: sm.stats.DescrStatsW(da_bmxht_inches).zconfint_mean()
```

```
In [ ]: sm.stats.DescrStatsW(da_bmxht_cm).zconfint_mean()
```

Q3a. Describe how the confidence interval constructed in centimeters relates to the confidence interval constructed in inches.

1.4 Question 4

Partition the sample based on 10-year age bands, i.e. the resulting groups will consist of people with ages from 18-28, 29-38, etc. Construct 95% confidence intervals for the difference between the mean BMI for females and for males within each age band.

```
In [ ]: da_bmi = da[["RIAGENDR", "RIDAGEYR", "BMXBMI"]].dropna()
        da_bmi["RIAGENDR"] = da_bmi.RIAGENDR.replace({1: "Male", 2: "Female"})
        da_bmi.head()
```

```
In [ ]: da_bmi["agegrp"] = pd.cut(da_bmi.RIDAGEYR, [18, 28, 38, 48, 58, 68, 78, 88, 98])
        da_bmi.head()
```

```
In [ ]: da_bmi_agg_by_year_gender = da_bmi.groupby(["agegrp", "RIAGENDR"]).agg({"BMXBMI": [np.r
        da_bmi_agg_by_year_gender
```

Q4a. How do the widths of these confidence intervals differ? Provide an explanation for any substantial diferences in the confidence interval widths that you see.

```
In [ ]: # Calculate the SEM (standard error) for females and for males within each age band
        da_bmi_agg_by_year_gender["BMXBMI", "sem", "Female"] = da_bmi_agg_by_year_gender["BMXBM
        da_bmi_agg_by_year_gender["BMXBMI", "sem", "Male"] = da_bmi_agg_by_year_gender["BMXBMI"
        da_bmi_agg_by_year_gender
```

```
In [ ]: # Calculate the mean difference of BMI between females and males within each age band
        da_bmi_agg_by_year_gender["BMXBMI", "mean_diff", ""] = da_bmi_agg_by_year_gender["BMXBM
        da_bmi_agg_by_year_gender["BMXBMI", "sem_diff", ""] = np.sqrt(da_bmi_agg_by_year_gender
        da_bmi_agg_by_year_gender
```

```
In [ ]: # calculates the lower and upper limits of its 95% CI.
        da_bmi_agg_by_year_gender["BMXBMI", "lcb_diff", ""] = da_bmi_agg_by_year_gender["BMXBMI
        da_bmi_agg_by_year_gender["BMXBMI", "ucb_diff", ""] = da_bmi_agg_by_year_gender["BMXBMI
        da_bmi_agg_by_year_gender
```

1.5 Question 5

Construct a 95% confidence interval for the first and second systolic blood pressure measures, and for the difference between the first and second systolic blood pressure measurements within a subject.

```
In [ ]: # enter code here
```

Q5a. Based on these confidence intervals, would you say that a difference of zero between the population mean values of the first and second systolic blood pressure measures is consistent with the data?

Q5b. Discuss how the width of the confidence interval for the within-subject difference compares to the widths of the confidence intervals for the first and second measures.

1.6 Question 6

Construct a 95% confidence interval for the mean difference between the average age of a smoker, and the average age of a non-smoker.

```
In [ ]: # insert your code here
```

Q6a. Use graphical and numerical techniques to compare the variation in the ages of smokers to the variation in the ages of non-smokers.

```
In [1]: # insert your code here
```

Q6b. Does it appear that uncertainty about the mean age of smokers, or uncertainty about the mean age of non-smokers contributed more to the uncertainty for the mean difference that we are focusing on here?