NHANES Hypothesis Testing Walkthrough

October 14, 2020

1 Hypothesis Testing

In this notebook we demonstrate formal hypothesis testing using the NHANES data.

It is important to note that the NHANES data are a "complex survey". The data are not an independent and representative sample from the target population. Proper analysis of complex survey data should make use of additional information about how the data were collected. Since complex survey analysis is a somewhat specialized topic, we ignore this aspect of the data here, and analyze the NHANES data as if it were an independent and identically distributed sample from a population.

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib
        matplotlib.use('Agg') # workaround, there may be a better way
        import seaborn as sns
        %matplotlib inline
        import matplotlib.pyplot as plt
        import statsmodels.api as sm
        import scipy.stats.distributions as dist
```

Below we read the data, and convert some of the integer codes to text values.

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

        da["SMQ020x"] = da.SMQ020.replace({1: "Yes", 2: "No", 7: np.nan, 9: np.nan})
```

```
In [5]: #head of data
        da.head(3)
```

```
Out[5]:     SEQN  ALQ101  ALQ110  ALQ130  SMQ020  RIAGENDR  RIDAGEYR  RIDRETH1  \
        0  83732     1.0     NaN     1.0       1         1        62         3
        1  83733     1.0     NaN     6.0       1         1        53         3
        2  83734     1.0     NaN     NaN       1         1        78         3

           DMDCITZN  DMDEDUC2  ...  BMXWT  BMXHT  BMXBMI  BMXLEG  BMXARML  BMXARMC  \
        0       1.0       5.0  ...   94.8  184.5    27.8    43.3     43.6     35.9
```

```
1       2.0       3.0  ...   90.4  171.4    30.8    38.0     40.0     33.2
2       1.0       3.0  ...   83.4  170.1    28.8    35.6     37.0     31.0

   BMXWAIST  HIQ210  SMQ020x  RIAGENDRx
0     101.1     2.0      Yes       Male
1     107.9     NaN      Yes       Male
2     116.5     2.0      Yes       Male

[3 rows x 30 columns]
```

```
In [3]: da["SMQ020x"].head()
```

```
Out[3]: 0    Yes
        1    Yes
        2    Yes
        3     No
        4     No
        Name: SMQ020x, dtype: object
```

```
In [4]: da["RIAGENDRx"] = da.RIAGENDR.replace({1: "Male", 2: "Female"})

        da["RIAGENDRx"].head()
```

```
Out[4]: 0      Male
        1      Male
        2      Male
        3    Female
        4    Female
        Name: RIAGENDRx, dtype: object
```

```
In [6]: da['DMDCITZN'] = da.DMDCITZN.replace({1: "Yes",2:"No",7:np.nan,9:np.nan})

        da['DMDCITZN'].head()
```

```
Out[6]: 0    Yes
        1     No
        2    Yes
        3    Yes
        4    Yes
        Name: DMDCITZN, dtype: object
```

1.0.1 Hypothesis Tests for One Proportion

The most basic hypothesis test may be the one-sample test for a proportion. This test is used if we have specified a particular value as the null value for the proportion, and we wish to assess if the data are compatible with the true parameter value being equal to this specified value. One-sample tests are not used very often in practice, because it is not very common that we have a specific fixed value to use for comparison. For illustration, imagine that the rate of lifetime smoking in another country was known to be 40%, and we wished to assess whether the rate of lifetime smoking in

the US were different from 40%. In the following notebook cell, we carry out the (two-sided) one-sample test that the population proportion of smokers is 0.4, and obtain a p-value of 0.43. This indicates that the NHANES data are compatible with the proportion of (ever) smokers in the US being 40%.

```
In [7]: x = da.SMQ020x.dropna() == "Yes"  #popn of smokers
        p = x.mean() #popn proportion
        se = np.sqrt(0.4*0.6/len(x)) #standard error calculation
        test_stat = (p-0.4) / se #test statistic
        pvalue = 2 * dist.norm.cdf(-np.abs(test_stat)) #p-value calculation, #cdf = cumulative
        print(test_stat,pvalue)
```

```
0.7823563854332805 0.4340051581348052
```

We can conclude: 1. Test stat of 0.78 is small 2. p-value of 0.43 is very high Therefore we do not reject the null hypothesis here. We continue to believe the population of smokers is .4 or 40%.

The following cell carries out the same test as performed above using the Statsmodels library. The results in the first (default) case below are slightly different from the results obtained above because Statsmodels by default uses the sample proportion instead of the null proportion when computing the standard error. This distinction is rarely consequential, but we can specify that the null proportion should be used to calculate the standard error, and the results agree exactly with what we calculated above. The first two lines below carry out tests using the normal approximation to the sampling distribution of the test statistic, and the third line below carries uses the exact binomial sampling distribution. We can see here that the p-values are nearly identical in all three cases. This is expected when the sample size is large, and the proportion is not close to either 0 or 1.

Statsmodels can do this easily for you!

```
In [8]: sm.stats.proportions_ztest(x.sum(), len(x), 0.4)
```

```
Out[8]: (0.7807518954896244, 0.43494843171868214)
```

```
In [9]: sm.stats.proportions_ztest(x.sum(), len(x), 0.4, prop_var = 0.4)
```

```
Out[9]: (0.7823563854332805, 0.4340051581348052)
```

```
In [10]: sm.stats.binom_test(x.sum(), len(x), 0.4) #exact binomial p-value
```

```
Out[10]: 0.4340360854459431
```

Conclusion: we can again see that we obtained the same values as above. The test statistic is small and the p value is very high therefore we accept the null hypothesis and continue to believe the proportion of smokers is 0.4 or 40%.

1.0.2 Hypothesis Tests for Two Proportions

Comparative tests tend to be used much more frequently than tests comparing one population to a fixed value. A two-sample test of proportions is used to assess whether the proportion of individuals with some trait differs between two sub-populations. For example, we can compare

the smoking rates between females and males. Since smoking rates vary strongly with age, we do this in the subpopulation of people between 20 and 25 years of age. In the cell below, we carry out this test without using any libraries, implementing all the test procedures covered elsewhere in the course using Python code. We find that the smoking rate for men is around 10 percentage points greater than the smoking rate for females, and this difference is statistically significant (the p-value is around 0.01).

```
In [14]: dx = da[["SMQ020x", "RIDAGEYR", "RIAGENDRx"]].dropna()
         dx = dx.loc[(dx.RIDAGEYR >=20) & (dx.RIDAGEYR <=25), :] #Restrict to people between a

         dx.head()
```

Out[14]:

	SMQ020x	RIDAGEYR	RIAGENDRx
6	Yes	22	Male
17	No	24	Female
26	Yes	22	Male
38	No	20	Female
40	Yes	24	Male

```
In [15]: #calculate the proportion of YES responses and sample size
         p = dx.groupby("RIAGENDRx")["SMQ020x"].agg([lambda z: np.mean(z == "Yes"), "size"])
         p.columns = ["Smoke", "N"]
         print(p)
```

```
               Smoke    N
RIAGENDRx
Female      0.238971  272
Male        0.341270  252
```

Essentially the same test as above can be conducted by converting the "Yes"/"No" responses to numbers (Yes=1, No=0) and conducting a two-sample t-test, as below: The pooled approach will be used.

```
In [16]: p_comb = (dx.SMQ020x == "Yes").mean()
         va = p_comb * (1 - p_comb)

         se = np.sqrt(va * (1 / p.N.Female + 1 / p.N.Male))

In [17]: (p_comb, va, se)

Out[17]: (0.2881679389312977, 0.2051271779033856, 0.039599757248262944)

In [19]: #test statistic and p value
         test_stat = (p.Smoke.Female - p.Smoke.Male) / se
         p_value = 2 * dist.norm.cdf(-np.abs(test_stat))
         print(test_stat, p_value)

-2.5833303066279414 0.009785159057508375
```

Findings: 1. Test statistic is very small (-2.58) 2. P value is small (0.009) Now we can reject the null hypothesis and say that the population proportion of female and male smokers is different.

```
In [20]: #dataframes for females
         dx_females = dx.loc[dx.RIAGENDRx == "Female", "SMQ020x"].replace({"Yes": 1, "No": 0})
         dx_females
```

```
Out[20]: 17      0
         38      0
         46      0
         69      1
         102     1
         128     0
         136     0
         179     1
         182     0
         191     0
         203     0
         239     0
         264     0
         272     0
         297     0
         328     0
         340     0
         379     0
         391     1
         430     1
         436     0
         437     0
         442     0
         450     1
         458     0
         460     0
         463     0
         480     0
         494     0
         514     0
                ..
         5136    0
         5149    0
         5158    0
         5170    0
         5184    0
         5187    0
         5208    0
         5224    0
         5245    0
         5258    0
```

```
5313    0
5320    1
5328    0
5372    0
5437    1
5440    0
5463    0
5466    0
5472    0
5485    0
5494    1
5507    0
5513    0
5523    1
5597    0
5622    0
5649    0
5678    1
5707    0
5734    0
Name: SMQ020x, Length: 272, dtype: int64
```

```
In [21]: #dataframe for males
         dx_males = dx.loc[dx.RIAGENDRx == "Male", "SMQ020x"].replace({"Yes": 1, "No": 0})
         dx_males
```

```
Out[21]: 6       1
         26      1
         40      1
         48      0
         96      0
         123     1
         142     0
         146     1
         163     0
         240     0
         254     0
         275     0
         304     0
         324     1
         351     0
         390     1
         428     0
         431     1
         492     1
         493     0
         554     0
         589     0
```

```
603     0
620     0
641     0
642     0
648     0
651     0
652     1
667     0
       ..
4968    1
4980    1
4989    1
4991    1
4992    0
4994    0
5091    0
5107    1
5109    0
5173    1
5209    0
5268    0
5329    0
5334    1
5345    1
5351    1
5394    0
5420    0
5454    0
5474    1
5520    1
5522    1
5532    1
5535    0
5573    0
5602    0
5667    1
5688    0
5701    0
5729    0
Name: SMQ020x, Length: 252, dtype: int64
```

Now we can simply use the statsmodels library to calculate this using both females and males dataframes.

```
In [24]: sm.stats.ttest_ind(dx_females, dx_males)
```

```
Out[24]: (-2.5949731446269344, 0.00972590232121254, 522.0)
```

Same outcome is achieved. We can reject the null hypothesis.

1.0.3 Hypothesis Tests Comparing Means

Tests of means are similar in many ways to tests of proportions. Just as with proportions, for comparing means there are one and two-sample tests, z-tests and t-tests, and one-sided and two-sided tests. As with tests of proportions, one-sample tests of means are not very common, but we illustrate a one sample test in the cell below. We compare systolic blood pressure to the fixed value 120 (which is the lower threshold for "pre-hypertension"), and find that the mean is significantly different from 120 (the point estimate of the mean is 126).

```
In [25]: dx = da[["BPXSY1", "RIDAGEYR", "RIAGENDRx"]].dropna()
         dx
```

Out[25]:

	BPXSY1	RIDAGEYR	RIAGENDRx
0	128.0	62	Male
1	146.0	53	Male
2	138.0	78	Male
3	132.0	56	Female
4	100.0	42	Female
5	116.0	72	Female
6	110.0	22	Male
7	120.0	32	Female
9	178.0	56	Male
10	144.0	46	Male
11	116.0	45	Male
12	104.0	30	Female
13	124.0	67	Female
14	132.0	67	Male
15	134.0	57	Female
16	102.0	19	Female
17	110.0	24	Female
18	138.0	27	Female
19	136.0	54	Female
20	110.0	49	Male
21	148.0	80	Female
22	140.0	69	Female
23	116.0	58	Female
24	136.0	56	Male
25	108.0	27	Female
26	122.0	22	Male
27	142.0	60	Female
28	132.0	51	Male
29	122.0	68	Female
30	146.0	69	Female
...	...	...	...
5702	116.0	38	Male
5703	178.0	64	Female
5704	134.0	75	Female
5705	174.0	80	Male
5706	124.0	72	Male

```
5707   130.0        25    Female
5708   102.0        29    Female
5709   132.0        38      Male
5711   144.0        62      Male
5712   114.0        27    Female
5713   116.0        43      Male
5714   162.0        39      Male
5715   124.0        34    Female
5717   112.0        32      Male
5718   128.0        45      Male
5720   110.0        38      Male
5721   118.0        35    Female
5722   114.0        34    Female
5723   142.0        72    Female
5724   132.0        41    Female
5725   110.0        34      Male
5726   132.0        53      Male
5727   164.0        69    Female
5728   112.0        32      Male
5729   112.0        25      Male
5730   112.0        76    Female
5731   118.0        26      Male
5732   154.0        80    Female
5733   104.0        35      Male
5734   118.0        24    Female

[5401 rows x 3 columns]
```

```
In [26]: dx = dx.loc[(dx.RIDAGEYR >= 40) & (dx.RIDAGEYR <= 50) & (dx.RIAGENDRx == "Male"), :]
         dx
```

Out[26]:

	BPXSY1	RIDAGEYR	RIAGENDRx
10	144.0	46	Male
11	116.0	45	Male
20	110.0	49	Male
42	128.0	42	Male
51	118.0	50	Male
66	124.0	41	Male
70	104.0	40	Male
72	140.0	48	Male
94	112.0	49	Male
101	104.0	43	Male
116	124.0	45	Male
119	132.0	43	Male
133	134.0	49	Male
135	120.0	40	Male
144	130.0	40	Male
152	154.0	43	Male

```
173    112.0        44      Male
176    102.0        46      Male
197    136.0        40      Male
204    120.0        45      Male
224    104.0        46      Male
246    192.0        45      Male
249    152.0        46      Male
251    156.0        43      Male
252    152.0        46      Male
269    106.0        45      Male
299    148.0        50      Male
323    116.0        41      Male
339    114.0        40      Male
358     98.0        42      Male
...      ...       ...       ...
5309   144.0        44      Male
5317   124.0        46      Male
5330   118.0        40      Male
5358   114.0        49      Male
5369   114.0        41      Male
5370   136.0        46      Male
5376   142.0        49      Male
5378   110.0        43      Male
5379   138.0        42      Male
5388   128.0        50      Male
5421   116.0        46      Male
5448   162.0        48      Male
5486   116.0        40      Male
5501   132.0        47      Male
5555   124.0        44      Male
5593   126.0        48      Male
5596   146.0        50      Male
5601   114.0        50      Male
5610   106.0        47      Male
5612   124.0        46      Male
5625   114.0        47      Male
5628   104.0        41      Male
5644   134.0        48      Male
5662   146.0        47      Male
5666   106.0        50      Male
5680   134.0        50      Male
5690   138.0        48      Male
5693    96.0        41      Male
5713   116.0        43      Male
5718   128.0        45      Male

[421 rows x 3 columns]
```

```
In [27]: print(dx.BPXSY1.mean())
```

```
125.86698337292161
```

We can see the mean blood pressure is 125.86

```
In [28]: sm.stats.ztest(dx.BPXSY1, value=120) #value is null hypothesis that mean bp for males
```

```
Out[28]: (7.469764137102597, 8.033869113167905e-14)
```

Conclusion: We can reject the null hypothesis as the p-value is very low and this thus proves that the mean blood pressure for males between 40 and 50 is not 120.

1.0.4 Two populations

In the cell below, we carry out a formal test of the null hypothesis that the mean blood pressure for women between the ages of 50 and 60 is equal to the mean blood pressure of men between the ages of 50 and 60. The results indicate that while the mean systolic blood pressure for men is slightly greater than that for women (129 mm/Hg versus 128 mm/Hg), this difference is not statistically significant.

There are a number of different variants on the two-sample t-test. Two often-encountered variants are the t-test carried out using the t-distribution, and the t-test carried out using the normal approximation to the reference distribution of the test statistic, often called a z-test. Below we display results from both these testing approaches. When the sample size is large, the difference between the t-test and z-test is very small.

```
In [29]: dx = da[["BPXSY1", "RIDAGEYR", "RIAGENDRx"]].dropna()
         dx = dx.loc[(dx.RIDAGEYR >= 50) & (dx.RIDAGEYR <= 60), :]
         dx.head()
```

```
Out[29]:     BPXSY1  RIDAGEYR RIAGENDRx
         1    146.0        53      Male
         3    132.0        56    Female
         9    178.0        56      Male
         15   134.0        57    Female
         19   136.0        54    Female
```

```
In [30]: bpx_female = dx.loc[dx.RIAGENDRx=="Female", "BPXSY1"]
         bpx_male = dx.loc[dx.RIAGENDRx=="Male", "BPXSY1"]
         print(bpx_female.mean(), bpx_male.mean())
```

```
127.92561983471074 129.23829787234044
```

```
In [31]: print(sm.stats.ztest(bpx_female, bpx_male))
```

```
(-1.105435895556249, 0.2689707570859362)
```

```
In [32]: print(sm.stats.ttest_ind(bpx_female, bpx_male))

(-1.105435895556249, 0.26925004137768577, 952.0)
```

We ran 2 different t tests. The p value is very high so we can't reject the null hypothesis in this case.

Another important aspect of two-sample mean testing is **"heteroscedasticity", meaning that the variances within the two groups being compared may be different.** While the goal of the test is to compare the means, the variances play an important role in calibrating the statistics (deciding how big the mean difference needs to be to be declared statisitically significant). In the NHANES data, we see that there are moderate differences between the amount of variation in BMI for females and for males, looking within 10-year age bands. In every age band, females having greater variation than males.

```
In [33]: dx = da[["BMXBMI", "RIDAGEYR", "RIAGENDRx"]].dropna()
         da["agegrp"] = pd.cut(da.RIDAGEYR, [18, 30, 40, 50, 60, 70, 80])
         da.groupby(["agegrp", "RIAGENDRx"])["BMXBMI"].agg(np.std).unstack()
```

Out[33]:

RIAGENDRx	Female	Male
agegrp		
(18, 30]	7.745893	6.649440
(30, 40]	8.315608	6.622412
(40, 50]	8.076195	6.407076
(50, 60]	7.575848	5.914373
(60, 70]	7.604514	5.933307
(70, 80]	6.284968	4.974855

Therefore we can say that heteroscedasticity is present as the variances between the 2 groups are different.

The standard error of the mean difference (e.g. mean female blood pressure minus mean mal blood pressure) can be estimated in at least two different ways. In the statsmodels library, these approaches are referred to as the "pooled" and the "unequal" approach to estimating the variance. If the variances are equal (i.e. there is no heteroscedasticity), then there should be little difference between the two approaches. Even in the presence of moderate heteroscedasticity, as we have here, we can see that the results for the two differences are quite similar. Below we have a loop that considers each 10-year age band and assesses the evidence for a difference in mean BMI for women and for men. The results printed in each row of output are the test-statistic and p-value.

```
In [34]: for k, v in da.groupby("agegrp"):
             bmi_female = v.loc[v.RIAGENDRx=="Female", "BMXBMI"].dropna()
             bmi_female = sm.stats.DescrStatsW(bmi_female)
             bmi_male = v.loc[v.RIAGENDRx=="Male", "BMXBMI"].dropna()
             bmi_male = sm.stats.DescrStatsW(bmi_male)
             print(k)
             print("pooled: ", sm.stats.CompareMeans(bmi_female, bmi_male).ztest_ind(usevar='po
             print("unequal: ", sm.stats.CompareMeans(bmi_female, bmi_male).ztest_ind(usevar='u
             print()
```

```
(18, 30]
pooled:  (1.7026932933643388, 0.08862548061449649)
unequal:  (1.7174610823927268, 0.08589495934713022)

(30, 40]
pooled:  (1.4378280405644916, 0.1504828511464818)
unequal:  (1.4437869620833494, 0.14879891057892475)

(40, 50]
pooled:  (2.8933761158070186, 0.003811246059501354)
unequal:  (2.9678691663536725, 0.0029987194174035366)

(50, 60]
pooled:  (3.362108779981367, 0.0007734964571391746)
unequal:  (3.375494390173923, 0.0007368319423226574)

(60, 70]
pooled:  (3.6172401442432753, 0.000297761021031936)
unequal:  (3.62848309454456, 0.0002850914147149227)

(70, 80]
pooled:  (2.926729252512258, 0.0034254694144858636)
unequal:  (2.937779886769224, 0.003305716331519299)
```

We can see the p values are small which verifies that the variances are different between each group.