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 [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
```

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 [5]:

```
x = da.SMQ020x.dropna() == "Yes"
```

In [6]:

```
p = x.mean()
```

In [7]:

```
p
```

Out[7]:

```
0.4050655021834061
```

In [8]:

```
se = np.sqrt(.4 * (1 - .4)/ len(x))
se
```

Out[8]:

```
0.00647467353462031
```

In [9]:

```
test_stat = (p - 0.4) / se
test_stat
```

Out[9]:

```
0.7823563854332805
```

In [10]:

```
pvalue = 2 * dist.norm.cdf(-np.abs(test_stat))
print(test_stat, pvalue)
```

```
0.7823563854332805 0.4340051581348052
```

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.

In [11]:

```
sm.stats.proportions_ztest(x.sum(), len(x), 0.4)
```

Out[11]:

```
(0.7807518954896244, 0.43494843171868214)
```

In [12]:

```
sm.stats.binom_test(x.sum(), len(x), 0.4)
```

Out[12]:

```
0.4340360854459431
```

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 [13]:

```
dx = da[["SMQ020x", "RIDAGEYR", "RIAGENDRx"]].dropna()

dx.head()
```

Out[13]:

	SMQ020x	RIDAGEYR	RIAGENDRx
0	Yes	62	Male
1	Yes	53	Male
2	Yes	78	Male
3	No	56	Female
4	No	42	Female

In [14]:

```
p = dx.groupby("RIAGENDRx")["SMQ020x"].agg([lambda z: np.mean(z == "Yes"), "size"])
p.columns = ["Smoke", "N"]
print(p)
```

```
              Smoke     N
RIAGENDRx
Female     0.304845  2972
Male       0.513258  2753
```

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:

In [15]:

```
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 [16]:

```
(p_comb, va, se)
```

Out[16]:

```
(0.4050655021834061, 0.2409874411243111, 0.01298546309757376)
```

In [17]:

```
test_stat = (p.Smoke.Female - p.Smoke.Male) / se
p_value = 2 * dist.norm.cdf(-np.abs(test_stat))
(test_stat, p_value)
```

Out[17]:

```
(-16.049719603652488, 5.742288777302776e-58)
```

In [18]:

```
dx_females = dx.loc[dx.RIAGENDRx == "Female", "SMQ020x"].replace({"Yes": 1, "No": 0})
dx_females
```

Out[18]:

```
3       0
4       0
5       0
7       0
12      1
13      0
15      0
16      0
17      0
18      1
19      0
21      0
22      1
23      0
25      0
27      1
29      0
30      1
33      0
34      0
35      1
36      0
38      0
39      0
43      0
46      0
47      0
50      0
52      0
54      0
       ..
5678    1
5679    0
5681    0
5682    1
5683    0
5684    0
5685    0
5686    0
5689    0
5692    0
5696    1
5697    0
5699    0
5703    1
5704    0
5707    0
5708    0
5710    0
5712    0
5715    0
```

```
5716    1
5719    1
5721    0
5722    0
5723    1
5724    0
5727    0
5730    1
5732    1
5734    0
Name: SMQ020x, Length: 2972, dtype: int64
```

In [19]:

```
dx_males = dx.loc[dx.RIAGENDRx == "Male", "SMQ020x"].replace({"Yes": 1, "No": 0})
dx_males
```

Out[19]:

```
0       1
1       1
2       1
6       1
8       0
9       0
10      1
11      1
14      0
20      0
24      0
26      1
28      0
31      0
32      1
37      0
40      1
41      0
42      0
44      1
45      1
48      0
49      1
51      0
53      1
56      1
57      0
59      0
60      1
64      1
       ..
5672    0
5673    1
5677    1
5680    0
5687    1
5688    0
5690    1
5691    0
5693    0
5694    0
5695    0
5698    1
5700    1
5701    0
5702    0
5705    1
5706    1
5709    1
5711    1
5713    0
5714    0
```

```
5717    1
5718    0
5720    0
5725    0
5726    1
5728    0
5729    0
5731    0
5733    1
Name: SMQ020x, Length: 2753, dtype: int64
```

In [20]:

```
sm.stats.ttest_ind(dx_females, dx_males)
```

Out[20]:

```
(-16.42058555898443, 3.032088786691117e-59, 5723.0)
```

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 [21]:

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

Out[21]:

	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

	BPXSY1	RIDAGEYR	RIAGENDRx
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 × 3 columns

In [22]:

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

Out[22]:

	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
...	...	...	...

	BPXSY1	RIDAGEYR	RIAGENDRx
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 × 3 columns

In [23]:

```
print(dx.BPXSY1.mean())
```

```
125.86698337292161
```

In [24]:

```
sm.stats.ztest(dx.BPXSY1, value=120)
```

Out[24]:

```
(7.469764137102597, 8.033869113167905e-14)
```

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 [25]:

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

Out[25]:

	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 [26]:

```
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 [27]:

```
print(sm.stats.ztest(bpx_female, bpx_male))
```

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

In [28]:

```
print(sm.stats.ttest_ind(bpx_female, bpx_male))
```

```
(-1.105435895556249, 0.26925004137768577, 952.0)
```

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 [29]:

```
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[29]:

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

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 [30]:

```
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='pooled'
    print("unequal: ", sm.stats.CompareMeans(bmi_female, bmi_male).ztest_ind(usevar='unequa
    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, 0.00029761021031936)
unequal:  (3.62848309454456, 0.0002850914147149227)
```

In [ ]: