Forward School

Program Code: J620-002-4:2020

Program Name: FRONT-END SOFTWARE DEVELOPMENT

Title : Hypothesis Testing with NHANES Dataset

Name: Chuay Xiang Ze

IC Number: 021224070255

Date : 14/7/2023

Introduction : Learning how to do hypothesis testing with a NHANES dataset.

Conclusion : Managed to complete tasks relating to the topic.

Case Study - Hypothesis testing with NHANES Data

In this notebook we will explore formal hypothesis testing using the NHANES (https://www.cdc.gov/nchs/nhanes/index.htm) 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.

First we import the libraries that we will need.

In [121]:

```
%matplotlib inline
import pandas as pd
import numpy as np
import seaborn as sns
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. The NHANES codebooks for SMQ020 (https://wwwn.cdc.gov/Nchs/Nhanes/2015-2016/SMQ_I.htm#SMQ020), RIAGENDR (https://wwwn.cdc.gov/Nchs/Nhanes/2015-2016/DEMO_I.htm#RIAGENDR), and DMDCITZN (https://wwwn.cdc.gov/Nchs/Nhanes/2015-2016/DEMO_I.htm#DMDCITZN) describe the meanings of the numerical codes.

In [122]:

```
# read data
da = pd.read_csv("./nhanes_2015_2016.csv")
da.head()
```

Out[122]:

	SEQN	ALQ101	ALQ110	ALQ130	SMQ020	RIAGENDR	RIDAGEYR	RIDRETH1	DMDCITZ
0	83732	1.0	NaN	1.0	1	1	62	3	1
1	83733	1.0	NaN	6.0	1	1	53	3	2
2	83734	1.0	NaN	NaN	1	1	78	3	1
3	83735	2.0	1.0	1.0	2	2	56	3	1
4	83736	2.0	1.0	1.0	2	2	42	4	1

5 rows × 28 columns

In [123]:

```
## for 'SMQ020x', replace 1: Yes, 2: No, 7: Nan, 9: Nan
da['SMQ020x'] = da['SMQ020']
reset = {1: "Yes", 2: "No", 7: None, 9: None}
da = da.replace({"SMQ020x": reset})
da['SMQ020x']
da
```

Out[123]:

	SEQN	ALQ101	ALQ110	ALQ130	SMQ020	RIAGENDR	RIDAGEYR	RIDRETH1	DMDC
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	
3	83735	2.0	1.0	1.0	2	2	56	3	
4	83736	2.0	1.0	1.0	2	2	42	4	
...	...	...	...	...	...	...	...	...	
5730	93695	2.0	2.0	NaN	1	2	76	3	
5731	93696	2.0	2.0	NaN	2	1	26	3	
5732	93697	1.0	NaN	1.0	1	2	80	3	
5733	93700	NaN	NaN	NaN	1	1	35	3	
5734	93702	1.0	NaN	2.0	2	2	24	3	

5735 rows × 29 columns

In [124]:

```
## for 'RIAGENDRx' replace 1: Male, and 2: Female
da['RIAGENDRx'] = da['RIAGENDR']
reset = {1: "Male", 2: "Female"}
da = da.replace({"RIAGENDRx": reset})
da['RIAGENDRx']
```

Out[124]:

```
0         Male
1         Male
2         Male
3       Female
4       Female
         ...
5730    Female
5731      Male
5732    Female
5733      Male
5734    Female
Name: RIAGENDRx, Length: 5735, dtype: object
```

In [125]:

```
## for 'DMDCITZNx' replace 1: Yes, 2: No, 7: Nan, 9: Nan
da['DMDCITZNx'] = da['DMDCITZN']
reset = {1: "Yes", 2: "No", 7: None, 9: None}
da = da.replace({"DMDCITZNx": reset})
da['DMDCITZNx']
da
```

Out[125]:

	SEQN	ALQ101	ALQ110	ALQ130	SMQ020	RIAGENDR	RIDAGEYR	RIDRETH1	DMDC
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	
3	83735	2.0	1.0	1.0	2	2	56	3	
4	83736	2.0	1.0	1.0	2	2	42	4	
...	...	...	...	...	...	...	...	...	
5730	93695	2.0	2.0	NaN	1	2	76	3	
5731	93696	2.0	2.0	NaN	2	1	26	3	
5732	93697	1.0	NaN	1.0	1	2	80	3	
5733	93700	NaN	NaN	NaN	1	1	35	3	
5734	93702	1.0	NaN	2.0	2	2	24	3	

5735 rows × 31 columns

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 [126]:

```
# Drop missing values from SMQ020x, RIDAGEYR and RIAGENDRx
new_da = da.copy()
new_da.dropna(subset = ['SMQ020x', 'RIDAGEYR', 'RIAGENDRx'], inplace=True)
```

In [127]:

```
# Select only people between 20 and 25 years old for RIDAGEYR
# and store in a new dataframe 'dx'
dx = new_da[(new_da['RIDAGEYR'] >= 20) & (new_da['RIDAGEYR'] <= 25)].copy()
dx
```

Out[127]:

	SEQN	ALQ101	ALQ110	ALQ130	SMQ020	RIAGENDR	RIDAGEYR	RIDRETH1	DMDC
6	83741	1.0	NaN	8.0	1	1	22	4	
17	83761	1.0	NaN	1.0	2	2	24	5	
26	83784	1.0	NaN	4.0	1	1	22	2	
38	83809	2.0	2.0	NaN	2	2	20	4	
40	83813	1.0	NaN	2.0	1	1	24	3	
...	...	...	...	...	...	...	...	...	
5688	93618	2.0	2.0	NaN	2	1	25	1	
5701	93639	1.0	NaN	2.0	2	1	25	3	
5707	93653	1.0	NaN	3.0	2	2	25	3	
5729	93691	2.0	2.0	NaN	2	1	25	5	
5734	93702	1.0	NaN	2.0	2	2	24	3	

524 rows × 31 columns

In [ ]:

```
```

In [128]:

```
# Summarize the data by calculating the proportion of yes responses and the sample size
reset = {"Yes": 1, "No": 0}
dx = dx.replace({"SMQ020x": reset})
dx = dx[['SMQ020x', 'RIAGENDRx']]
dz = dx[['SMQ020x', 'RIAGENDRx']].groupby('RIAGENDRx').agg(['mean', 'size'])
print(dz)
```

```
            SMQ020x
               mean size
RIAGENDRx
Female     0.238971  272
Male       0.341270  252
```

Test statistic = $\frac{Best\,Estimate - Hypothesized\,Mean}{Standard\,Error\,Estimate}$

$= \frac{\hat{p}_1 - \hat{p}_2}{se(\hat{p})}$

$se(\hat{p}) = \sqrt{\hat{p}(1-\hat{p})(\frac{1}{n_1} + \frac{1}{n_2})}$

where $\hat{p}$ is the population proportion mean for smokers, `SMQ020x == "Yes"` .

In [198]:

```
# The pooled rate of yes responses, and the standard error of the
# estimated difference of proportions
```

Out[198]:

```
0.005027468075988649
```

In [143]:

```

```

```
---------------------------------------------------------------------------
-
AttributeError                            Traceback (most recent call las
t)
Cell In[143], line 1
----> 1 n1_mean = n1.sum() / len(n1)
      2 n2_mean = n2.sum() / len(n2)
      3 t_val = (n1_sum - n2_sum) / pooled_se

AttributeError: 'int' object has no attribute 'sum'
```

In [11]:

```
# Calculate the test statistic and its p-value
```

```
-2.5833303066279414 0.009785159057508375
```

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 [203]:

```
dx_females = dx.loc[dx.RIAGENDRx=="Female", "SMQ020x"].replace({"Yes": 1, "No": 0})
dx_males = dx.loc[dx.RIAGENDRx=="Male", "SMQ020x"].replace({"Yes": 1, "No": 0})
sm.stats.ttest_ind(dx_females, dx_males) # prints test statistic, p-value, degrees of fr
```

Out[203]:

```
(-2.5949731446269344, 0.00972590232121254, 522.0)
```

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 [86]:

```
# Drop missing values from BPXSY1, RIDAGEYR and RIAGENDRx
alt_da = da.copy()
alt_da.dropna(subset = ['BPXSY1', 'RIDAGEYR', 'RIAGENDRx'], inplace=True)
alt_da
```

Out[86]:

	SEQN	ALQ101	ALQ110	ALQ130	SMQ020	RIAGENDR	RIDAGEYR	RIDRETH1	DMDC
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	
3	83735	2.0	1.0	1.0	2	2	56	3	
4	83736	2.0	1.0	1.0	2	2	42	4	
...	...	...	...	...	...	...	...	...	
5730	93695	2.0	2.0	NaN	1	2	76	3	
5731	93696	2.0	2.0	NaN	2	1	26	3	
5732	93697	1.0	NaN	1.0	1	2	80	3	
5733	93700	NaN	NaN	NaN	1	1	35	3	
5734	93702	1.0	NaN	2.0	2	2	24	3	

5401 rows × 31 columns

In [62]:

```
# Select only "Male" between the ages >= 40 and <= 50
filtered = alt_da[(alt_da['RIAGENDRx'] == 'Male') & (alt_da['RIDAGEYR'] >= 40) & (alt_da
new_filtered = filtered[['BPXSY1', 'RIDAGEYR', 'RIAGENDRx']].head(5)
new_filtered
```

In [88]:

```
## calculate the mean blood pressure
filtered['BPXSY1'].mean()
```

Out[88]:

```
125.86698337292161
```

In [89]:

```
# using stats.ztest library print the test statistic, p-value where the average is 120
from statsmodels.stats.weightstats import ztest

sample = filtered['BPXSY1']
mean_value = 120

ztest(sample, value=mean_value)
```

Out[89]:

```
(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 [90]:

```
# Drop missing values from BPXSY1, RIDAGEYR and RIAGENDRx
bp_da = da.copy()
bp_da.dropna(subset = ['BPXSY1', 'RIDAGEYR', 'RIAGENDRx'], inplace=True)
bp_da
```

Out[90]:

	SEQN	ALQ101	ALQ110	ALQ130	SMQ020	RIAGENDR	RIDAGEYR	RIDRETH1	DMDC
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	
3	83735	2.0	1.0	1.0	2	2	56	3	
4	83736	2.0	1.0	1.0	2	2	42	4	
...	...	...	...	...	...	...	...	...	
5730	93695	2.0	2.0	NaN	1	2	76	3	
5731	93696	2.0	2.0	NaN	2	1	26	3	
5732	93697	1.0	NaN	1.0	1	2	80	3	
5733	93700	NaN	NaN	NaN	1	1	35	3	
5734	93702	1.0	NaN	2.0	2	2	24	3	

5401 rows × 31 columns

In [91]:

```
# Select only people from the ages >= 50 and <= 60
filtered_50 = bp_da[(bp_da['RIDAGEYR'] >= 50) & (bp_da['RIDAGEYR'] <= 60)]
new_50 = filtered_50[['BPXSY1', 'RIDAGEYR', 'RIAGENDRx']].head(5)
new_50
```

Out[91]:

	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 [92]:

```
## Select only female
female_filter = filtered_50[filtered_50['RIAGENDRx'] == "Female"]
female_filter[['BPXSY1', 'RIDAGEYR', 'RIAGENDRx']]
```

Out[92]:

	BPXSY1	RIDAGEYR	RIAGENDRx
3	132.0	56	Female
15	134.0	57	Female
19	136.0	54	Female
23	116.0	58	Female
27	142.0	60	Female
...	...	...	...
5648	146.0	57	Female
5661	114.0	55	Female
5675	136.0	55	Female
5681	124.0	57	Female
5686	132.0	60	Female

484 rows × 3 columns

In [93]:

```
## Select only male
male_filter = filtered_50[filtered_50['RIAGENDRx'] == "Male"]
male_filter[['BPXSY1', 'RIDAGEYR', 'RIAGENDRx']]
```

Out[93]:

	BPXSY1	RIDAGEYR	RIAGENDRx
1	146.0	53	Male
9	178.0	56	Male
24	136.0	56	Male
28	132.0	51	Male
32	114.0	56	Male
...	...	...	...
5659	116.0	53	Male
5663	108.0	52	Male
5666	106.0	50	Male
5680	134.0	50	Male
5726	132.0	53	Male

470 rows × 3 columns

In [94]:

```
## prints female mean, male mean
# 127.92561983471074 129.23829787234044
print(female_filter['BPXSY1'].mean(), male_filter['BPXSY1'].mean())
```

```
127.92561983471074 129.23829787234044
```

In [95]:

```
## use the sm.stats.ztest, print the test statistic, p-value
# (-1.105435895556249, 0.2689707570859362)
sample1 = female_filter['BPXSY1']
sample2 = male_filter['BPXSY1']
ztest(sample1, sample2)
```

Out[95]:

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

In [96]:

```
#use the sm.stats.ttest_ind, print the test statistic, p-value
import statsmodels.api as sm
sm.stats.ttest_ind(sample1, sample2)
```

Out[96]:

```
(-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 statistically 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 [152]:

```
## Select BMXBMI, RIDAGEYR, RIAGENDRx and drop all NAs
da.dropna(subset = ['BMXBMI', 'RIDAGEYR', 'RIAGENDRx'], inplace=True)
da[['BMXBMI', 'RIDAGEYR', 'RIAGENDRx']].head(5)
```

Out[152]:

	BMXBMI	RIDAGEYR	RIAGENDRx
0	27.8	62	Male
1	30.8	53	Male
2	28.8	78	Male
3	42.4	56	Female
4	20.3	42	Female

In [97]:

```
import math
age_groups = [18, 30, 40, 50, 60, 70, 80]
da['agegrp'] = pd.cut(da['RIDAGEYR'], bins=age_groups, right=True)
std_deviations_female = da[da['RIAGENDRx'] == 'Female'].groupby('agegrp')['BMXBMI'].std(
std_deviations_male = da[da['RIAGENDRx'] == 'Male'].groupby('agegrp')['BMXBMI'].std()
result = pd.DataFrame({'Female': std_deviations_female, 'Male': std_deviations_male})
result
```

Out[97]:

agegrp	Female	Male
(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

In [26]:

```
## cut to [18, 30, 40, 50, 60, 70, 80] age group, and calculate the standard deviation
```

Out[26]:

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 male 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 methods 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 [27]:

```
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='pool
    print("unequal:", sm.stats.CompareMeans(bmi_female, bmi_male).ztest_ind(usevar='uneq
    print()
```

```
(18, 30]
pooled:  (1.7026932933643306, 0.08862548061449803)
unequal: (1.7174610823927183, 0.08589495934713169)

(30, 40]
pooled:  (1.4378280405644919, 0.15048285114648174)
unequal: (1.4437869620833497, 0.1487989105789246)

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

(50, 60]
pooled:  (3.362108779981383, 0.0007734964571391287)
unequal: (3.3754943901739387, 0.0007368319423226156)

(60, 70]
pooled:  (3.617240144243268, 0.00029776102103194453)
unequal: (3.628483094544553, 0.00028509141471492935)

(70, 80]
pooled:  (2.926729252512241, 0.003425469414486057)
unequal: (2.9377798867692064, 0.0033057163315194853)
```

Paired tests

A common situation in applied research is to measure the same quantity multiple times on each unit of analysis. For example, in NHANES, systolic blood pressure is measured at least two times (sometimes there is a third measurement) on each subject. Although the measurements are repeated, there is no guarantee that the mean is the same each time, i.e. the mean blood pressure may be slightly lower on the second measurement compared to the first, since people are a bit more nervous the first time they are measured. A paired test is a modified form of mean test that can be used when we are comparing two repeated measurements on the same unit.

A paired t-test for means is equivalent to taking the difference between the first and second measurement, and using a one-sample test to compare the mean of these differences to zero. Below we see that in the entire NHANES sample, the first measurement of systolic blood pressure is on average 0.67 mm/Hg greater than the second measurement. While this difference is not large, it is strongly statistically significant. That is, there is strong evidence that the mean values for the first and second blood pressure measurement differ.

In [99]:

```
## select only BPXSY1 and BPXSY2, drop all NAs_
paired_da = da.copy()
paired_da.dropna(subset = ['BPXSY1', 'BPXSY2'], inplace = True)
paired_da = paired_da[['BPXSY1', 'BPXSY2']]
paired_da
```

Out[99]:

	BPXSY1	BPXSY2
0	128.0	124.0
1	146.0	140.0
2	138.0	132.0
3	132.0	134.0
4	100.0	114.0
...	...	...
5730	112.0	112.0
5731	118.0	116.0
5732	154.0	146.0
5733	104.0	106.0
5734	118.0	114.0

5369 rows × 2 columns

In [110]:

```
##find the difference between BPXSY1 and BPXSY2
from statsmodels.stats.weightstats import ztest
paired_da['diff'] = paired_da['BPXSY1'] - paired_da['BPXSY2']
paired_da['diff']
```

Out[110]:

```
0        4.0
1        6.0
2        6.0
3       -2.0
4      -14.0
        ...
5730     0.0
5731     2.0
5732     8.0
5733    -2.0
5734     4.0
Name: diff, Length: 5369, dtype: float64
```

In [104]:

```
paired_da['diff'].mean()
```

Out[104]:

```
0.6749860309182343
```

In [30]:

```
## calculate the mean difference
```

0.6749860309182343

In [112]:

```
sample = paired_da['diff']
ztest(sample, value = 0)
```

Out[112]:

(9.800634425497911, 1.1188070930963587e-22)

In [31]:

```
## use the sm.stats.ztest library to print test statistic and p-value
```

Out[31]:

(9.800634425497911, 1.1188070930963587e-22)

To probe this effect further, we can divide the population into 10 year wide age bands and also stratify by gender, then carry out the paired t-test within each of the resulting 12 strata. We see that the second systolic blood pressure measurement is always lower on average than the first. The difference is larger for older people and for males. The difference is statistically significant for females over 30, and for males over 60.

Conducting many hypothesis tests and "cherry picking" the interesting results is usually a bad practice. Here we are doing such "multiple testing" for illustration, and acknowledge that the strongest differences may be over-stated. Nevertheless, there is a clear and consistent trend with age -- older people tend to have greater differences between their first and second blood pressure measurements than younger people. There is also a difference between the genders, with older men having a stronger difference between the first and second blood pressure measurements than older women. The gender difference for younger peple is less clear.

In [113]:

```
dx = da[["RIAGENDRx", "BPXSY1", "BPXSY2", "RIDAGEYR"]].dropna()
dx["agegrp"] = pd.cut(dx.RIDAGEYR, [18, 30, 40, 50, 60, 70, 80])
for k, g in dx.groupby(["RIAGENDRx", "agegrp"]):
    db = g.BPXSY1 - g.BPXSY2
    # print stratum definition, mean difference, sample size, test statistic, p-value
    print(k, db.mean(), db.size, sm.stats.ztest(db.values, value=0))
```

```
('Female', Interval(18, 30, closed='right')) 0.13708260105448156 569 (0.76
12107360791227, 0.4465312067051751)
('Female', Interval(30, 40, closed='right')) 0.6713615023474179 426 (3.307
398751951031, 0.0009416674523368051)
('Female', Interval(40, 50, closed='right')) 0.5970149253731343 469 (2.604
0611621024654, 0.009212631487347644)
('Female', Interval(50, 60, closed='right')) 0.7685393258426966 445 (3.102
3718750881724, 0.0019197663012041 96)
('Female', Interval(60, 70, closed='right')) 0.8787878787878788 396 (3.102
4528501809625, 0.0019192411825181255)
('Female', Interval(70, 80, closed='right')) 1.4512820512820512 390 (5.141
706875154317, 2.722536503552981e-07)
('Male', Interval(18, 30, closed='right')) 0.00390625 512 (0.0195962284164
7691, 0.9843654725443948)
('Male', Interval(30, 40, closed='right')) 0.46296296296296297 432 (1.9451
535788714596, 0.05175649697939119)
('Male', Interval(40, 50, closed='right')) 0.17894736842105263 380 (0.7201
800810138878, 0.47141412641258706)
('Male', Interval(50, 60, closed='right')) 0.3691588785046729 428 (1.43911
15097646396, 0.1501189315054144)
('Male', Interval(60, 70, closed='right')) 1.2736077481840193 413 (4.78194
0964515296, 1.7361067031915549e-06)
('Male', Interval(70, 80, closed='right')) 2.031413612565445 382 (6.801341
4549535005, 1.036494265013724e-11)
```

In [ ]:

```
```