nhanes_univariate_practice

October 18, 2022

1 Practice notebook for univariate analysis using NHANES data

This notebook will give you the opportunity to perform some univariate analyses on your own using the NHANES. These analyses are similar to what was done in the week 2 NHANES case study notebook.

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

        da = pd.read_csv("nhanes_2015_2016.csv")
```

```
In [4]: print(da.info())
        data_size = da.shape
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5735 entries, 0 to 5734
Data columns (total 28 columns):
SEQN        5735 non-null int64
ALQ101      5208 non-null float64
ALQ110      1731 non-null float64
ALQ130      3379 non-null float64
SMQ020      5735 non-null int64
RIAGENDR    5735 non-null int64
RIDAGEYR    5735 non-null int64
RIDRETH1    5735 non-null int64
DMDCITZN    5734 non-null float64
```

```
DMDEDUC2    5474 non-null float64
DMDMARTL    5474 non-null float64
DMDHHSIZ    5735 non-null int64
WTINT2YR    5735 non-null float64
SDMVPSU     5735 non-null int64
SDMVSTRA    5735 non-null int64
INDFMPIR    5134 non-null float64
BPXSY1      5401 non-null float64
BPXDI1      5401 non-null float64
BPXSY2      5535 non-null float64
BPXDI2      5535 non-null float64
BMXWT       5666 non-null float64
BMXHT       5673 non-null float64
BMXBMI      5662 non-null float64
BMXLEG      5345 non-null float64
BMXARML     5427 non-null float64
BMXARMC     5427 non-null float64
BMXWAIST    5368 non-null float64
HIQ210      4732 non-null float64
dtypes: float64(20), int64(8)
memory usage: 1.2 MB
None
```

In [2]:
```
# Check the class distribution of DMDMARTL using groupby and size
print(da.groupby('DMDMARTL').size())
```

```
DMDMARTL
1.0     2780
2.0      396
3.0      579
4.0      186
5.0     1004
6.0      527
77.0       2
dtype: int64
```

1.1 Question 1

Relabel the marital status variable DMDMARTL to have brief but informative character labels. Then construct a frequency table of these values for all people, then for women only, and for men only. Then construct these three frequency tables using only people whose age is between 30 and 40.

In [5]:
```
# insert your code here
da["DMDMARTLx"] = da.DMDMARTL.replace({1: "Married", 2: "Widowed", 3: "Divorced", 4: "S
da.DMDMARTLx.value_counts()
```

```
Out[5]: Married                 2780
        Never Married           1004
        Divorced                 579
        Living with Partner      527
        Widowed                  396
        Missing                  261
        Separated                186
        Refused                    2
        Name: DMDMARTLx, dtype: int64
```

```
In [6]: # Relabel the gender Status variable
        da["RIAGENDRx"] = da.RIAGENDR.replace({1: "Male", 2: "Female"})
        # Check the class distribution of RIAGENDR using groupby and size
        print(da.groupby('RIAGENDRx').size())
```

```
RIAGENDRx
Female    2976
Male      2759
dtype: int64
```

Q1a. Briefly comment on some of the differences that you observe between the distribution of marital status between women and men, for people of all ages.

```
In [7]: dx = da.groupby(["DMDMARTLx"])["RIAGENDRx"].value_counts().unstack()
        dx = dx.apply(lambda x: x/x.sum(), axis = 0)
        print(dx.to_string(float_format = "%.4f"))
```

```
RIAGENDRx            Female   Male
DMDMARTLx
Divorced             0.1176 0.0830
Living with Partner  0.0880 0.0960
Married              0.4378 0.5353
Missing              0.0423 0.0489
Never Married        0.1747 0.1754
Refused              0.0003 0.0004
Separated            0.0397 0.0246
Widowed              0.0995 0.0362
```

Q1b. Briefly comment on the differences that you observe between the distribution of marital status states for women between the overall population, and for women between the ages of 30 and 40.

```
In [8]: # between the distribution of marital status status for women between the overall popu
        da["agegap"] = pd.cut(da.RIDAGEYR,[30, 40])
        da[(da.RIAGENDRx == "Female") & (da.agegap == pd.Interval(30, 40))].DMDMARTLx.value_cou
```

```
Out[8]: Married                 0.044987
        Never Married           0.016914
        Living with Partner     0.009939
        Divorced                0.007498
        Separated               0.002964
        Widowed                 0.000349
        Name: DMDMARTLx, dtype: float64
```

```
In [9]: # between the distribution of marital status states for women between the ages of 30 a
        da["agegap"] = pd.cut(da.RIDAGEYR,[30, 40])
        da[(da.RIAGENDRx == "Female") & (da.agegap == pd.Interval(30, 40))].DMDMARTLx.value_cou
```

```
Out[9]: Married                 0.044987
        Never Married           0.016914
        Living with Partner     0.009939
        Divorced                0.007498
        Separated               0.002964
        Widowed                 0.000349
        Name: DMDMARTLx, dtype: float64
```

Q1c. Repeat part b for the men.

```
In [10]: # between the distribution of marital status states for men between the overall popul
         da[(da.RIAGENDRx == "Male")].DMDMARTLx.value_counts()/da["DMDMARTLx"].shape[0]
```

```
Out[10]: Married                 0.257541
         Never Married           0.084394
         Living with Partner     0.046207
         Divorced                0.039930
         Missing                 0.023540
         Widowed                 0.017437
         Separated               0.011857
         Refused                 0.000174
         Name: DMDMARTLx, dtype: float64
```

```
In [11]: # between the distribution of marital status states for men between the ages of 30 an
         da["agegap"] = pd.cut(da.RIDAGEYR,[30, 40])
         da[(da.RIAGENDRx == "Male") & (da.agegap == pd.Interval(30, 40))].DMDMARTLx.value_cour
```

```
Out[11]: Married                 0.044987
         Never Married           0.015519
         Living with Partner     0.012554
         Divorced                0.004185
         Separated               0.002092
         Widowed                 0.000349
         Refused                 0.000174
         Name: DMDMARTLx, dtype: float64
```

1.2 Question 2

Restricting to the female population, stratify the subjects into age bands no wider than ten years, and construct the distribution of marital status within each age band. Within each age band, present the distribution in terms of proportions that must sum to 1.

```
In [16]: # insert your code here
         x = da[da.RIAGENDRx == "Female"]
         x["agegap2"] = pd.cut(da.RIDAGEYR, [10, 20, 30, 40, 50, 60, 70, 80])
         dx = x.groupby(["agegap2"])["DMDMARTLx"].value_counts().unstack()
         dx = dx.apply(lambda y: y/y.sum(), axis = 0)
         print(dx.to_string(float_format = "%.2f"))
```

```
DMDMARTLx  Divorced  Living with Partner  Married  Missing  Never Married  Refused  Separated
agegap2
(10, 20]        NaN                 0.03     0.00     1.00           0.06      NaN        NaN
(20, 30]       0.03                 0.40     0.12      NaN           0.44      NaN       0.09
(30, 40]       0.12                 0.22     0.20      NaN           0.19      NaN       0.14
(40, 50]       0.20                 0.14     0.22      NaN           0.12      NaN       0.28
(50, 60]       0.24                 0.12     0.20      NaN           0.08     1.00       0.23
(60, 70]       0.24                 0.07     0.16      NaN           0.07      NaN       0.19
(70, 80]       0.17                 0.01     0.10      NaN           0.04      NaN       0.07
```

```
In [17]: da["agegap"] = pd.cut(da.RIDAGEYR, [10, 20, 30, 40, 50, 60, 70, 80])
         (da[da["RIAGENDRx"] == "Female"].groupby(["agegap", "DMDMARTLx"]).size() / da[da["RIA
```

```
Out[17]: DMDMARTLx  Divorced  Living with Partner   Married   Missing  Never Married  \
         agegap
         (10, 20]        NaN             0.048485  0.006061  0.763636       0.181818
         (20, 30]   0.021401             0.206226  0.305447       NaN       0.445525
         (30, 40]   0.090717             0.120253  0.544304       NaN       0.204641
         (40, 50]   0.137450             0.073705  0.573705       NaN       0.125498
         (50, 60]   0.176596             0.068085  0.546809       NaN       0.089362
         (60, 70]   0.192744             0.043084  0.480726       NaN       0.086168
         (70, 80]   0.143902             0.007317  0.317073       NaN       0.051220

         DMDMARTLx   Refused  Separated   Widowed
         agegap
         (10, 20]        NaN        NaN       NaN
         (20, 30]        NaN   0.021401       NaN
         (30, 40]        NaN   0.035865  0.004219
         (40, 50]        NaN   0.065737  0.023904
         (50, 60]   0.002128   0.057447  0.059574
         (60, 70]        NaN   0.049887  0.147392
         (70, 80]        NaN   0.019512  0.460976
```

Q2a. Comment on the trends that you see in this series of marginal distributions.
Q2b. Repeat the construction for males.

```
In [18]: # insert your code here
         x = da[da.RIAGENDRx == "Male"]
         x["agegap2"] = pd.cut(da.RIDAGEYR, [10, 20, 30, 40, 50, 60, 70, 80])
         dx = x.groupby(["agegap2"])["DMDMARTLx"].value_counts().unstack()
         dx = dx.apply(lambda y: y/y.sum(), axis = 0)
         print(dx.to_string(float_format = "%.2f"))
```

```
DMDMARTLx  Divorced  Living with Partner  Married  Missing  Never Married  Refused  Separated
agegap2
(10, 20]        NaN                 0.01     0.00     1.00           0.07      NaN        NaN
(20, 30]       0.01                 0.35     0.07      NaN           0.47      NaN       0.10
(30, 40]       0.10                 0.27     0.17      NaN           0.18     1.00       0.18
(40, 50]       0.15                 0.12     0.19      NaN           0.08      NaN       0.16
(50, 60]       0.25                 0.13     0.20      NaN           0.10      NaN       0.15
(60, 70]       0.24                 0.08     0.20      NaN           0.08      NaN       0.21
(70, 80]       0.25                 0.03     0.17      NaN           0.02      NaN       0.21
```

```
In [19]: da["agegap"] = pd.cut(da.RIDAGEYR, [10, 20, 30, 40, 50, 60, 70, 80])
         (da[da["RIAGENDRx"] == "Male"].groupby(["agegap", "DMDMARTLx"]).size() / da[da["RIAGEI
```

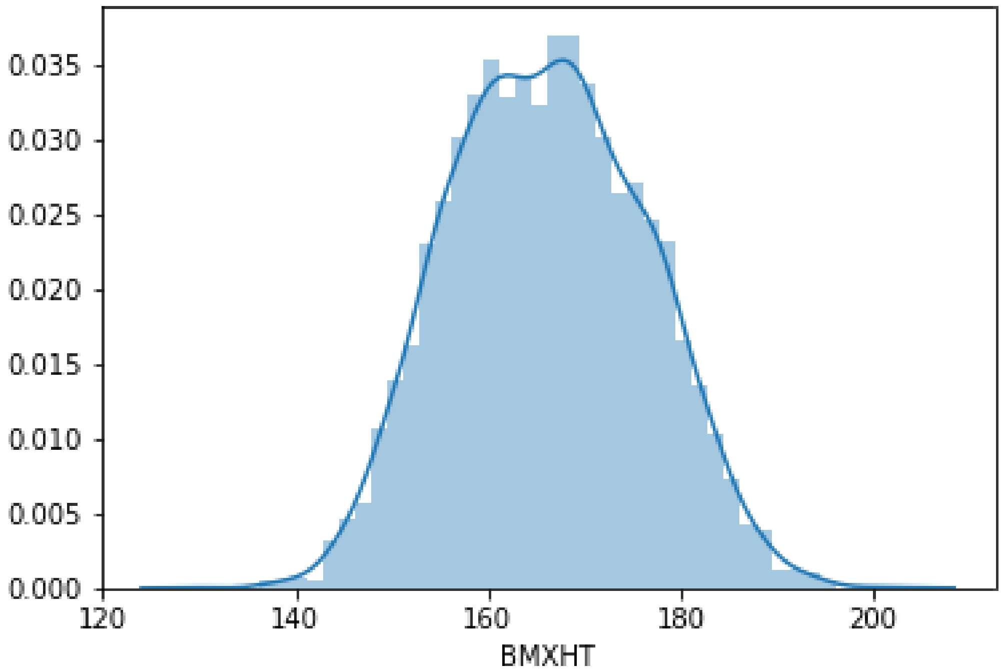
```
Out[19]: DMDMARTLx  Divorced  Living with Partner   Married   Missing  Never Married  \
         agegap
         (10, 20]        NaN             0.018182  0.006061  0.818182       0.218182
         (20, 30]   0.003891             0.178988  0.200389       NaN       0.439689
         (30, 40]   0.050633             0.151899  0.544304       NaN       0.187764
         (40, 50]   0.067729             0.065737  0.561753       NaN       0.077689
         (50, 60]   0.121277             0.072340  0.629787       NaN       0.100000
         (60, 70]   0.124717             0.049887  0.659864       NaN       0.086168
         (70, 80]   0.139024             0.021951  0.600000       NaN       0.021951

         DMDMARTLx  Refused  Separated   Widowed
         agegap
         (10, 20]       NaN        NaN       NaN
         (20, 30]       NaN   0.013619  0.003891
         (30, 40]   0.00211   0.025316  0.004219
         (40, 50]       NaN   0.021912  0.003984
         (50, 60]       NaN   0.021277  0.021277
         (60, 70]       NaN   0.031746  0.038549
         (70, 80]       NaN   0.034146  0.163415
```

Q2c. Comment on any notable differences that you see when comparing these results for females and for males.

1.3 Question 3

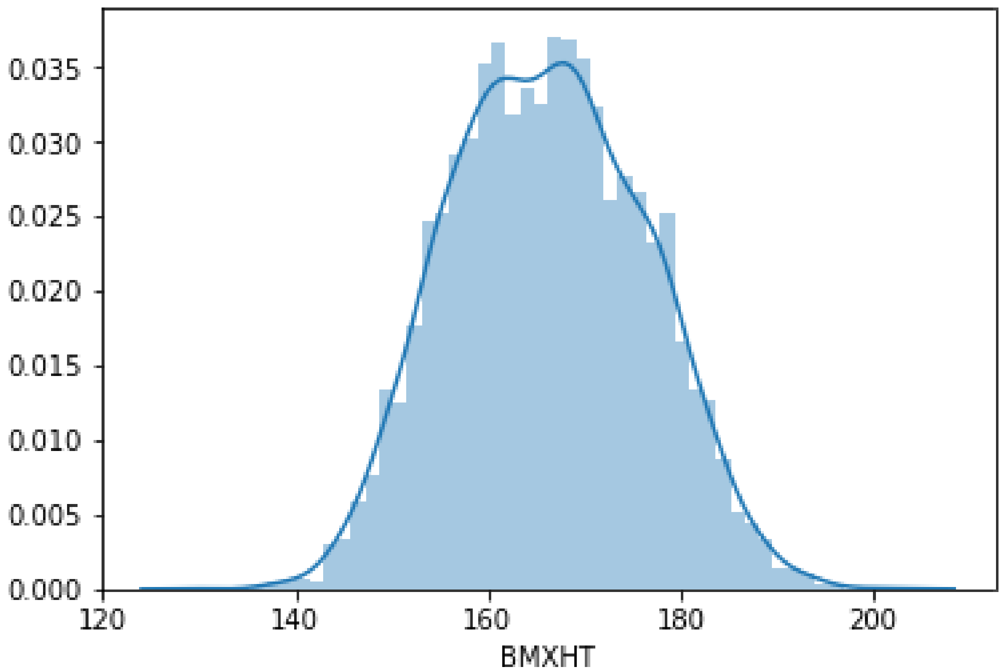
Construct a histogram of the distribution of heights using the BMXHT variable in the NHANES sample.

```
In [23]: # insert your code here
         sns.distplot(da.BMXHT.dropna())
         plt.show()
```



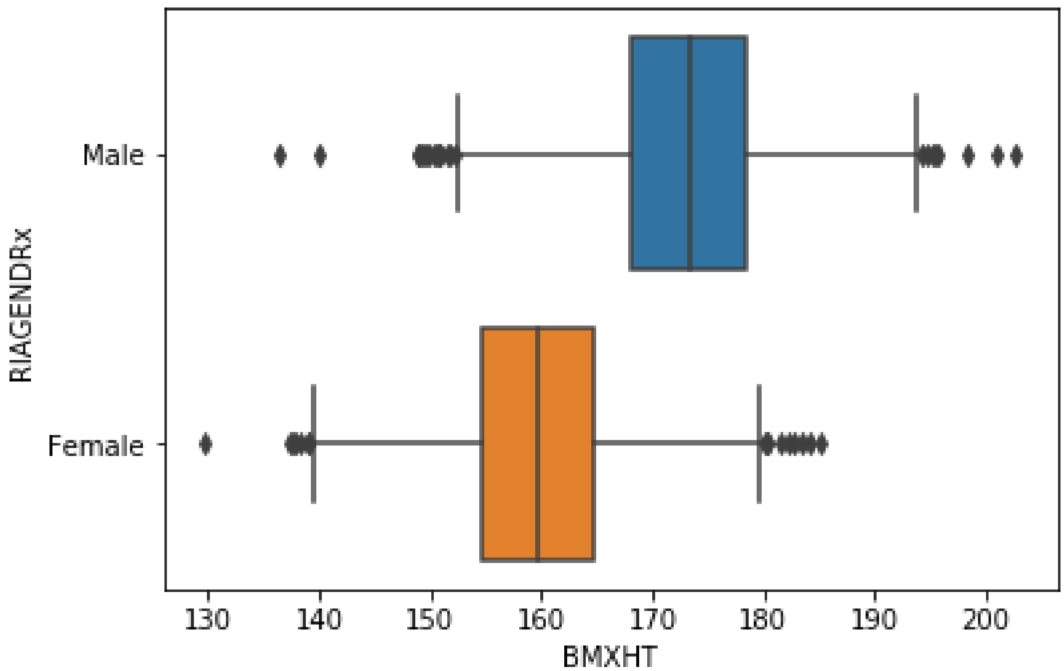
Q3a. Use the `bins` argument to distplot to produce histograms with different numbers of bins. Assess whether the default value for this argument gives a meaningful result, and comment on what happens as the number of bins grows excessively large or excessively small.

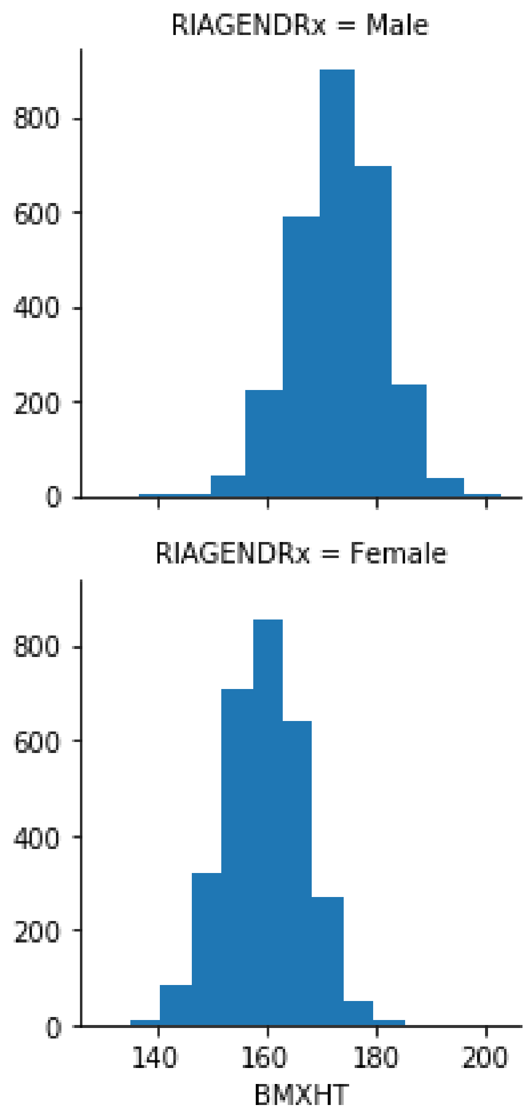
```
In [24]: sns.distplot(da.BMXHT.dropna(), bins = 50)
         plt.show()
```



Q3b. Make separate histograms for the heights of women and men, then make a side-by-side boxplot showing the heights of women and men.

```
In [25]: # insert your code here
         sns.boxplot(x= da["BMXHT"], y = da["RIAGENDRx"])
         g = sns.FacetGrid(da, row ="RIAGENDRx")
         g = g.map(plt.hist, "BMXHT")
         plt.show()
```


Male
Female
RIAGENDRx
130
140
150
160
170
180
190
200
BMXHT

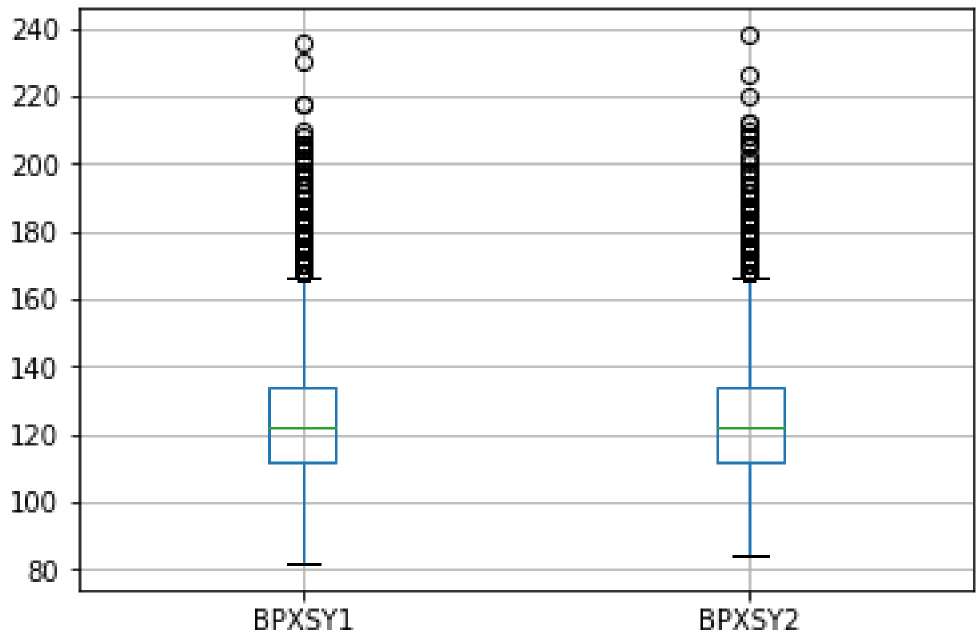


Q3c. Comment on what features, if any are not represented clearly in the boxplots, and what features, if any, are easier to see in the boxplots than in the histograms.

1.4 Question 4

Make a boxplot showing the distribution of within-subject differences between the first and second systolic blood pressure measurements (BPXSY1 and BPXSY2).

```
In [26]: # insert your code here
         x = da[["BPXSY1", "BPXSY2"]]
         x.boxplot()
         plt.show()
```



Q4a. What proportion of the subjects have a lower SBP on the second reading compared to the first?

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

Q4b. Make side-by-side boxplots of the two systolic blood pressure variables.

```
In [4]: # insert your code here
```

Q4c. Comment on the variation within either the first or second systolic blood pressure measurements, and the variation in the within-subject differences between the first and second systolic blood pressure measurements.

1.5 Question 5

Construct a frequency table of household sizes for people within each educational attainment category (the relevant variable is DMDEDUC2). Convert the frequencies to proportions.

```
In [28]: # insert your code here
         dx = da.groupby(["DMDEDUC2"])["DMDHHSIZ"].value_counts().unstack()
         dx = dx.apply(lambda x: x/x.sum(), axis = 1)
         print(dx.to_string(float_format="%.2f"))
```

```
DMDHHSIZ    1    2    3    4    5    6    7
DMDEDUC2
1.0      0.11 0.22 0.15 0.13 0.15 0.11 0.13
2.0      0.12 0.22 0.16 0.15 0.15 0.11 0.09
```

```
3.0       0.15 0.27 0.17 0.16 0.11 0.07 0.07
4.0       0.15 0.27 0.19 0.17 0.12 0.05 0.05
5.0       0.14 0.35 0.19 0.17 0.10 0.03 0.03
9.0        NaN 0.67  NaN  NaN 0.33  NaN  NaN
```

Q5a. Comment on any major differences among the distributions.

Q5b. Restrict the sample to people between 30 and 40 years of age. Then calculate the median household size for women and men within each level of educational attainment.

```
In [29]: # insert your code here
         da[(da.RIDAGEYR >= 30) & (da.RIDAGEYR <= 40)].groupby(["DMDEDUC2", "RIAGENDR"])["DMDHH
```

```
Out[29]: DMDEDUC2  RIAGENDR
         1.0       1           5.0
                   2           5.0
         2.0       1           4.5
                   2           5.0
         3.0       1           4.0
                   2           5.0
         4.0       1           4.0
                   2           4.0
         5.0       1           3.0
                   2           3.0
         Name: DMDHHSIZ, dtype: float64
```

1.6 Question 6

The participants can be clustered into "maked variance units" (MVU) based on every combination of the variables SDMVSTRA and SDMVPSU. Calculate the mean age (RIDAGEYR), height (BMXHT), and BMI (BMXBMI) for each gender (RIAGENDR), within each MVU, and report the ratio between the largest and smallest mean (e.g. for height) across the MVUs.

```
In [36]: # insert your code here
         (
             (
                 da.groupby(['SDMVSTRA', 'SDMVPSU', 'RIAGENDR'])
                 [['RIDAGEYR', 'BMXHT', 'BMXBMI']]
                 .mean()
             )

         ).unstack()
```

```
Out[36]:                     RIDAGEYR                  BMXHT                     BMXBMI  \
         RIAGENDR                   1          2          1           2          1
         SDMVSTRA SDMVPSU
         119      1         47.861111  47.663265  172.741667  159.570408  26.958333
                  2         54.363636  52.987952  172.906818  159.244578  27.160465
         120      1         43.130000  43.636364  169.537755  155.402041  30.939175
```

	2	45.219178	43.736111	173.075342	159.218056	27.727397
121	1	46.750000	44.397959	172.177885	158.871579	29.416505
	2	42.063158	44.376344	174.764516	160.229032	26.273118
122	1	44.653061	42.897436	173.998969	161.315385	28.528866
	2	44.320000	47.333333	170.332323	157.231111	25.744444
123	1	47.829787	44.841121	174.315217	162.059615	29.231522
	2	52.126582	46.457447	174.454430	160.476596	28.811392
124	1	50.750000	51.664000	172.109009	158.788710	28.614414
	2	48.245614	42.541667	174.291228	162.853521	27.714035
125	1	55.165289	50.900901	173.631092	160.762385	29.727731
	2	49.705882	51.660000	174.456863	160.021429	29.143564
126	1	48.416667	46.229167	175.149398	160.387500	29.033333
	2	48.666667	47.205882	174.713043	160.892000	29.039130
127	1	53.137931	49.694444	171.545349	157.422430	31.062353
	2	54.070588	51.486239	173.366667	159.022936	30.557831
128	1	53.673267	55.638462	169.325000	156.339062	31.749000
	2	45.822785	45.589744	172.400000	160.437179	26.835443
129	1	43.922222	45.329787	171.094318	156.900000	26.493182
	2	45.775510	43.500000	173.138298	161.034259	28.961702
130	1	50.516854	47.810526	176.974157	161.977895	30.337079
	2	50.535354	50.833333	175.061224	160.060577	29.237755
131	1	53.140187	54.893617	175.610476	161.989362	28.259615
	2	46.778846	45.000000	175.091346	161.673810	30.077885
132	1	42.380435	43.210526	172.534066	161.508421	28.546154
	2	49.038760	51.700000	172.809524	159.138281	28.966667
133	1	44.054795	45.105882	171.509722	158.295122	27.495833
	2	47.489796	47.063158	171.179167	158.627368	27.966667

RIAGENDR		2
SDMVSTRA	SDMVPSU	
119	1	30.052041
	2	27.849398
120	1	32.419388
	2	27.400000
121	1	30.856842
	2	26.470968
122	1	29.447436
	2	26.611111
123	1	29.905769
	2	30.641489
124	1	29.533065
	2	28.640845
125	1	30.385321
	2	28.564286
126	1	31.262500
	2	29.612121
127	1	32.189720

```
              2          30.770642
128           1          32.303125
              2          27.491026
129           1          29.019149
              2          29.429630
130           1          30.700000
              2          31.490385
131           1          30.061702
              2          32.984127
132           1          29.848421
              2          30.540625
133           1          27.959259
              2          29.000000
```

In [38]:
```
(
    (
        da.groupby(['SDMVSTRA', 'SDMVPSU', 'RIAGENDR'])
        [['RIDAGEYR', 'BMXHT', 'BMXBMI']]
        .max()
    )
    /
    (
        da.groupby(['SDMVSTRA', 'SDMVPSU', 'RIAGENDR'])
        [['RIDAGEYR', 'BMXHT', 'BMXBMI']]
        .min()
    )
).unstack()
```

Out[38]:

		RIDAGEYR		BMXHT		BMXBMI	
RIAGENDR		1	2	1	2	1	2
SDMVSTRA	SDMVPSU						
119	1	4.444444	4.444444	1.231270	1.221838	2.994413	3.256684
	2	4.000000	4.444444	1.269385	1.267041	2.224390	4.045161
120	1	4.444444	4.444444	1.297715	1.254360	3.042105	3.666667
	2	4.444444	4.444444	1.271065	1.296011	2.875000	2.443182
121	1	4.444444	4.444444	1.252324	1.308458	3.104938	2.538462
	2	4.444444	4.444444	1.296036	1.249319	2.542373	3.349112
122	1	4.444444	4.444444	1.244646	1.232877	3.284916	3.066298
	2	4.444444	4.444444	1.272849	1.214035	1.902703	2.730539
123	1	4.444444	4.444444	1.210127	1.262143	3.039548	2.483333
	2	4.210526	4.444444	1.290116	1.237474	3.275000	2.947917
124	1	4.210526	4.444444	1.214700	1.259393	2.212871	3.656627
	2	4.210526	4.444444	1.204545	1.216480	2.436464	3.530387
125	1	4.444444	4.444444	1.266364	1.230070	3.271605	2.948571
	2	4.444444	4.444444	1.229072	1.248281	2.420765	3.437126
126	1	4.210526	4.210526	1.224347	1.223684	2.769231	3.395210
	2	4.444444	4.444444	1.244715	1.217753	2.355670	3.226994
127	1	4.444444	4.444444	1.219481	1.226573	3.220930	3.195531

	2	4.444444	4.444444	1.223082	1.267626	2.704142	3.184971
128	1	4.444444	4.000000	1.359029	1.277698	2.577114	3.160428
	2	4.444444	4.444444	1.271883	1.223135	2.734940	3.434483
129	1	4.444444	4.444444	1.209941	1.350810	2.124324	2.806630
	2	4.444444	4.444444	1.250484	1.245775	3.039326	3.413408
130	1	4.444444	4.444444	1.224691	1.227972	3.457831	3.261538
	2	4.444444	4.000000	1.255995	1.288210	2.994475	3.875862
131	1	4.210526	4.444444	1.203232	1.220938	3.357616	3.444444
	2	4.444444	4.444444	1.204759	1.203528	2.713568	3.822485
132	1	4.111111	4.210526	1.268568	1.198903	2.432161	3.121212
	2	4.444444	4.444444	1.292895	1.275562	3.512195	4.078788
133	1	4.444444	4.444444	1.265293	1.273934	2.502762	2.969512
	2	4.444444	4.444444	1.372161	1.202364	3.222222	3.706897

Q6a. Comment on the extent to which mean age, height, and BMI vary among the MVUs.

Q6b. Calculate the inter-quartile range (IQR) for age, height, and BMI for each gender and each MVU. Report the ratio between the largest and smalles IQR across the MVUs.

In [39]:
```
# insert your code here
(
    (
        da.groupby(['SDMVSTRA', 'SDMVPSU', 'RIAGENDR'])
        [['RIDAGEYR', 'BMXHT', 'BMXBMI']]
        .quantile(0.75)
    )
    -
    (
        da.groupby(['SDMVSTRA', 'SDMVPSU', 'RIAGENDR'])
        [['RIDAGEYR', 'BMXHT', 'BMXBMI']]
        .quantile(0.25)
    )
).unstack()
```

Out[39]:

0.75		RIDAGEYR		BMXHT		BMXBMI	
RIAGENDR		1	2	1	2	1	2
SDMVSTRA	SDMVPSU						
119	1	29.75	31.25	9.000	9.325	5.350	9.750
	2	29.00	33.50	11.225	9.950	5.300	9.350
120	1	23.75	26.50	12.125	8.750	9.400	8.775
	2	26.00	25.75	10.500	10.550	7.100	7.750
121	1	34.50	26.25	10.725	9.150	7.500	9.000
	2	25.50	26.00	8.600	9.600	5.700	8.100
122	1	29.50	24.00	9.400	10.400	7.700	9.875
	2	30.00	25.00	10.150	7.575	4.100	8.475
123	1	28.25	30.50	9.350	9.675	8.050	10.450
	2	31.50	34.50	9.900	11.200	8.100	9.975
124	1	32.00	27.00	9.800	8.375	6.100	8.950
	2	31.00	23.50	11.600	8.650	8.700	9.000

125	1	29.00	31.00	10.350	9.100	8.300	8.000
	2	33.50	32.25	7.925	10.675	7.900	10.325
126	1	36.25	30.25	10.450	8.500	8.000	10.675
	2	34.00	31.75	8.125	12.025	6.850	10.350
127	1	30.00	27.25	9.025	7.700	8.200	11.750
	2	28.00	30.00	10.750	11.600	5.950	9.200
128	1	33.00	28.00	9.950	9.125	6.675	8.500
	2	25.50	22.00	9.850	10.650	5.800	9.375
129	1	20.75	24.75	12.300	10.375	6.025	9.500
	2	30.75	26.25	10.700	8.900	5.800	9.725
130	1	36.00	35.50	9.900	8.650	6.700	11.200
	2	28.50	30.25	8.625	10.225	8.375	8.050
131	1	36.00	35.75	10.500	10.025	7.525	11.075
	2	28.00	24.00	7.750	7.575	7.850	10.625
132	1	21.25	30.00	10.600	10.950	6.600	10.700
	2	38.00	33.00	10.550	10.100	9.600	11.750
133	1	33.00	34.00	8.925	10.300	6.425	8.300
	2	32.25	28.50	8.850	9.550	5.900	9.650

Q6c. Comment on the extent to which the IQR for age, height, and BMI vary among the MVUs.