FILTROS

December 3, 2021

In [2]:
```
%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")
da["DMDMARTLx"]=da.DMDMARTL.replace({1:"Married", 2:"Widowed",3:"Divorced",4:"Separate
da["RIAGENDRx"] = da.RIAGENDR.replace({1: 'Male', 2: 'Female'})

#MINIMO E MAXIMO
da["agegroup"]=pd.cut(da.RIDAGEYR, bins=[31, 40], right=False)
print(da.groupby(["agegroup"])["RIDAGEYR"].min())
print(da.groupby(["agegroup"])["RIDAGEYR"].max())
print("")

# IDADE ENTRE 31 E 40
selections = (da.RIDAGEYR < 40) & (da.RIDAGEYR > 30)

# ESTADO CIVIL AGRUPADO POR GENERO
print('-----------ESTADO CIVIL AGRUPADO POR  GENERO-----------')
print(da.groupby(["RIAGENDRx"])["DMDMARTLx"].value_counts())

# ESTADO CIVIL AGRUPADO POR IDADE
print('-----------ESTADO CIVIL AGRUPADO POR IDADE-----------')
print(da.groupby(["agegroup"])["DMDMARTLx"].value_counts())

# ESTADO CIVIL AGRUPADO POR IDADE, GENERO
print('-----------ESTADO CIVIL AGRUPADO POR IDADE, GENERO-----------')
print(da.groupby(["agegroup", "RIAGENDRx"])["DMDMARTLx"].value_counts())
print("")

# FILTRO MALE
print('----------ESTADO CIVIL HOMENS-----------')
selections = (da.RIAGENDRx == "Male")
print(da.where(selections).DMDMARTLx.value_counts())
print("")
```

```
        # FILTRO FEMALE
        print('-----------ESTADO CIVIL MULHRES------------')
        selections = (da.RIAGENDRx == "Female")
        print(da.where(selections).DMDMARTLx.value_counts())
        print("")

        # FILTRO IDADE >30 <40
        print("--------------ESTADO CIVIL TODO MUNDO ENTRE 30 E 40-------------")
        selections = (da.RIDAGEYR < 40) & (da.RIDAGEYR > 30)
        print(da.where(selections).DMDMARTLx.value_counts())
        print("")

        # FILTRO HOMENS COM IDADE >30 <40
        print("--------------ESTADO CIVIL HOMENS ENTRE 30 E 40----------------")
        selections = (da.RIAGENDRx == "Male") & (da.RIDAGEYR < 40) & (da.RIDAGEYR > 30)
        print(da.where(selections).DMDMARTLx.value_counts())
        print("")

        # FILTRO MULHRRER COM IDADE >30 <40
        print("-------------------ESTADO CIVIL MULHERES ENTRE 30 E 40--------------------")
        selections = (da.RIAGENDRx == "Female") & (da.RIDAGEYR < 40) & (da.RIDAGEYR > 30)
        print(da.where(selections).DMDMARTLx.value_counts())
```

```
agegroup
[31, 40)    31
Name: RIDAGEYR, dtype: int64
agegroup
[31, 40)    39
Name: RIDAGEYR, dtype: int64

-----------ESTADO CIVIL AGRUPADO POR  GENERO------------
RIAGENDRx  DMDMARTLx
Female     Married                1303
           Never Married           520
           Divorced                350
           Widowed                 296
           Living with Partner     262
           Separated               118
           Refused                   1
Male       Married                1477
           Never Married           484
           Living with Partner     265
           Divorced                229
           Widowed                 100
           Separated                68
           Refused                   1
Name: DMDMARTLx, dtype: int64
```

```
----------ESTADO CIVIL AGRUPADO POR IDADE------------
agegroup  DMDMARTLx
[31, 40)  Married               462
          Never Married         169
          Living with Partner   122
          Divorced               55
          Separated              26
          Widowed                 4
          Refused                 1
Name: DMDMARTLx, dtype: int64
----------ESTADO CIVIL AGRUPADO POR IDADE, GENERO------------
agegroup  RIAGENDRx  DMDMARTLx
[31, 40)  Female     Married               228
                     Never Married          88
                     Living with Partner    55
                     Divorced               35
                     Separated              15
                     Widowed                 2
          Male       Married               234
                     Never Married          81
                     Living with Partner    67
                     Divorced               20
                     Separated              11
                     Widowed                 2
                     Refused                 1
Name: DMDMARTLx, dtype: int64

----------ESTADO CIVIL HOMENS------------
Married               1477
Never Married          484
Living with Partner    265
Divorced               229
Widowed                100
Separated               68
Refused                  1
Name: DMDMARTLx, dtype: int64

----------ESTADO CIVIL MULHRES------------
Married               1303
Never Married          520
Divorced               350
Widowed                296
Living with Partner    262
Separated              118
Refused                  1
Name: DMDMARTLx, dtype: int64

--------------ESTADO CIVIL TODO MUNDO ENTRE 30 E 40-------------
```

```
Married                462
Never Married          169
Living with Partner    122
Divorced                55
Separated               26
Widowed                  4
Refused                  1
Name: DMDMARTLx, dtype: int64

--------------ESTADO CIVIL HOMENS ENTRE 30 E 40----------------
Married                234
Never Married           81
Living with Partner     67
Divorced                20
Separated               11
Widowed                  2
Refused                  1
Name: DMDMARTLx, dtype: int64

------------------ESTADO CIVIL MULHERES ENTRE 30 E 40------------------
Married                228
Never Married           88
Living with Partner     55
Divorced                35
Separated               15
Widowed                  2
Name: DMDMARTLx, dtype: int64
```

```
In [5]: # subset the data to include only females
        da = da.where(da.RIAGENDRx == 'Female')

        # cut age into bands no wider than 10 years
        da['agegrp'] = pd.cut(da.RIDAGEYR, [20, 30, 40, 50, 60, 70, 80])

        # Eliminate rare/missing values
        dx = da.loc[~da.DMDMARTLx.isin(["Don't know", "Missing"]), :]

        # group marital status by age group band
        dx = dx.groupby(["agegrp"])["DMDMARTLx"]

        # obtain the counts for marital status within each age group band
        dx = dx.value_counts()
        dx

Out[5]: agegrp    DMDMARTLx
        (20, 30]  Never Married          229
                  Married                157
```

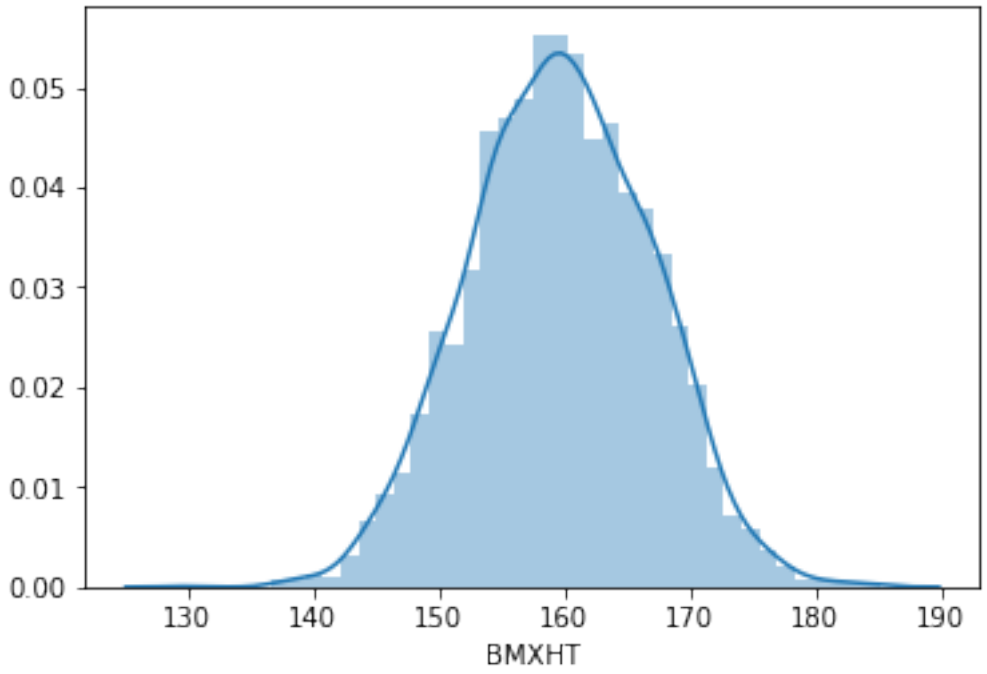
```
                Living with Partner    106
                Divorced                11
                Separated               11
        (30, 40]  Married              258
                Never Married           97
                Living with Partner     57
                Divorced                43
                Separated               17
                Widowed                  2
        (40, 50]  Married              288
                Divorced                69
                Never Married           63
                Living with Partner     37
                Separated               33
                Widowed                 12
        (50, 60]  Married              257
                Divorced                83
                Never Married           42
                Living with Partner     32
                Widowed                 28
                Separated               27
                Refused                  1
        (60, 70]  Married              212
                Divorced                85
                Widowed                 65
                Never Married           38
                Separated               22
                Living with Partner     19
        (70, 80]  Widowed              189
                Married                130
                Divorced                59
                Never Married           21
                Separated                8
                Living with Partner      3
        Name: DMDMARTLx, dtype: int64
```

```
In [4]: dx = dx.unstack() # Restructure the results from 'long' to 'wide'
        dx = dx.apply(lambda x: x/x.sum(), axis=1) # Normalize within each stratum to get prop
        print(dx.to_string(float_format="%.3f"))  # Limit display to 3 decimal places
```

```
DMDMARTLx  Divorced  Living with Partner  Married  Never Married  Refused  Separated  Widowed
agegrp
(20, 30]      0.021                0.206    0.305          0.446      NaN      0.021      NaN
(30, 40]      0.091                0.120    0.544          0.205      NaN      0.036    0.004
(40, 50]      0.137                0.074    0.574          0.125      NaN      0.066    0.024
(50, 60]      0.177                0.068    0.547          0.089    0.002      0.057    0.060
(60, 70]      0.193                0.043    0.481          0.086      NaN      0.050    0.147
(70, 80]      0.144                0.007    0.317          0.051      NaN      0.020    0.461
```

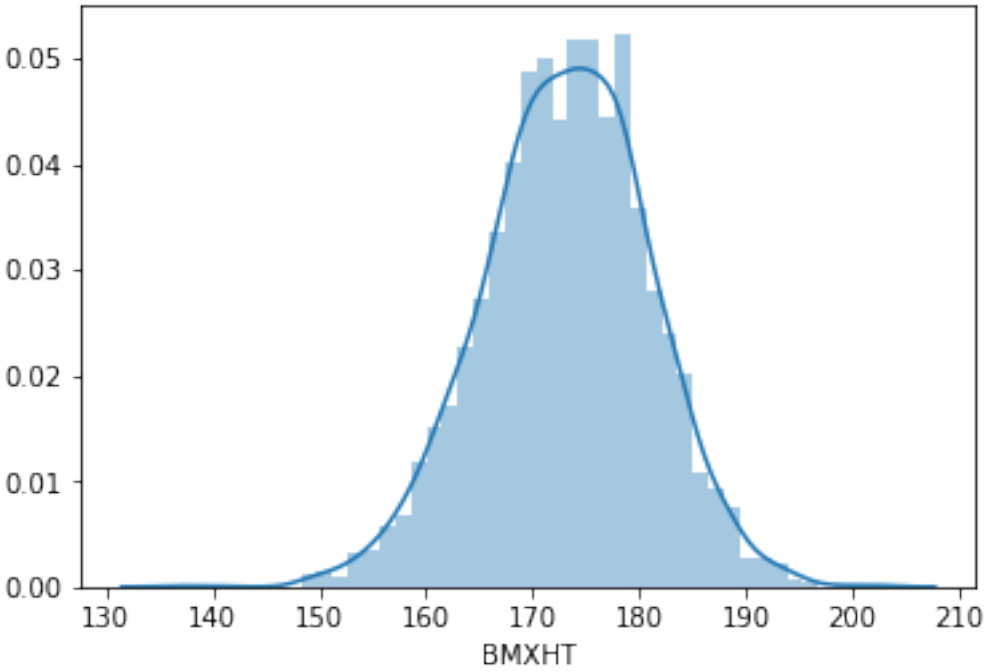
In [9]:
```
# histograms for males and females

%matplotlib inline
import matplotlib.pyplot as plt

import seaborn as sns
import pandas as pd

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

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

df = da.loc[da.RIAGENDRx.isin(["Female"]), :]
dm = da.loc[da.RIAGENDRx.isin(["Male"]), :]

sns.distplot(df.BMXHT.dropna())
```

Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f538cf6c4a8>
```



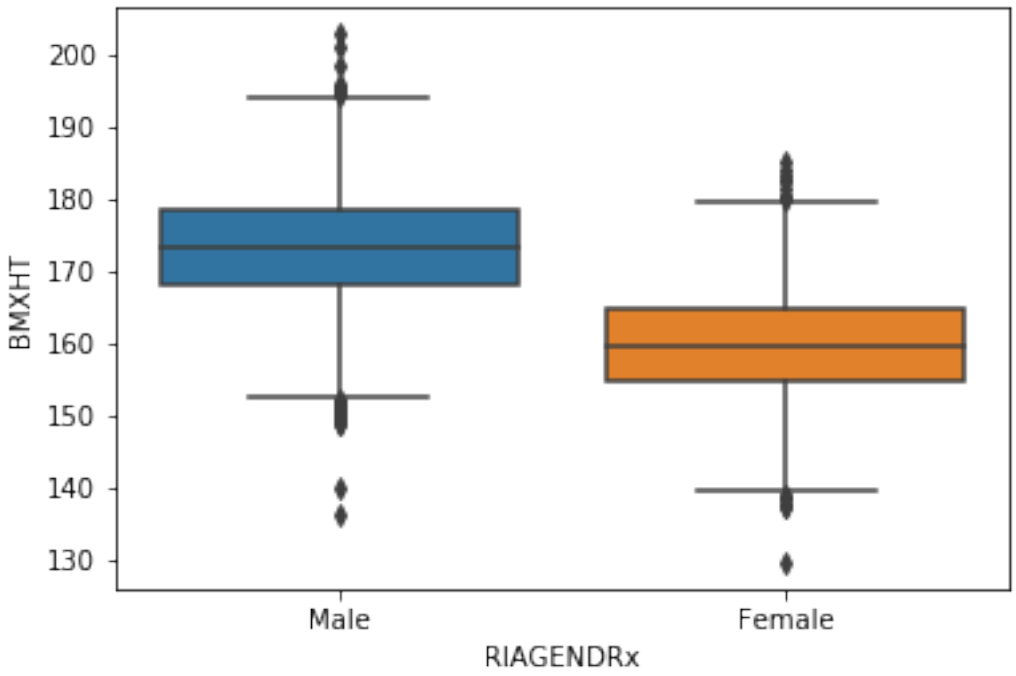
In [10]:
```
sns.distplot(dm.BMXHT.dropna())
```

Out[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f538cf6c5f8>
```

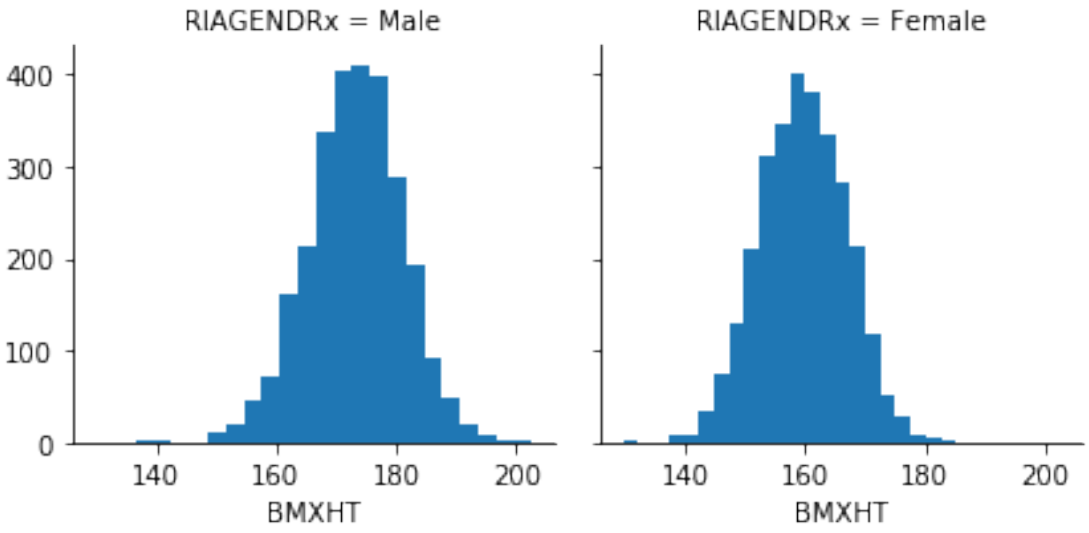


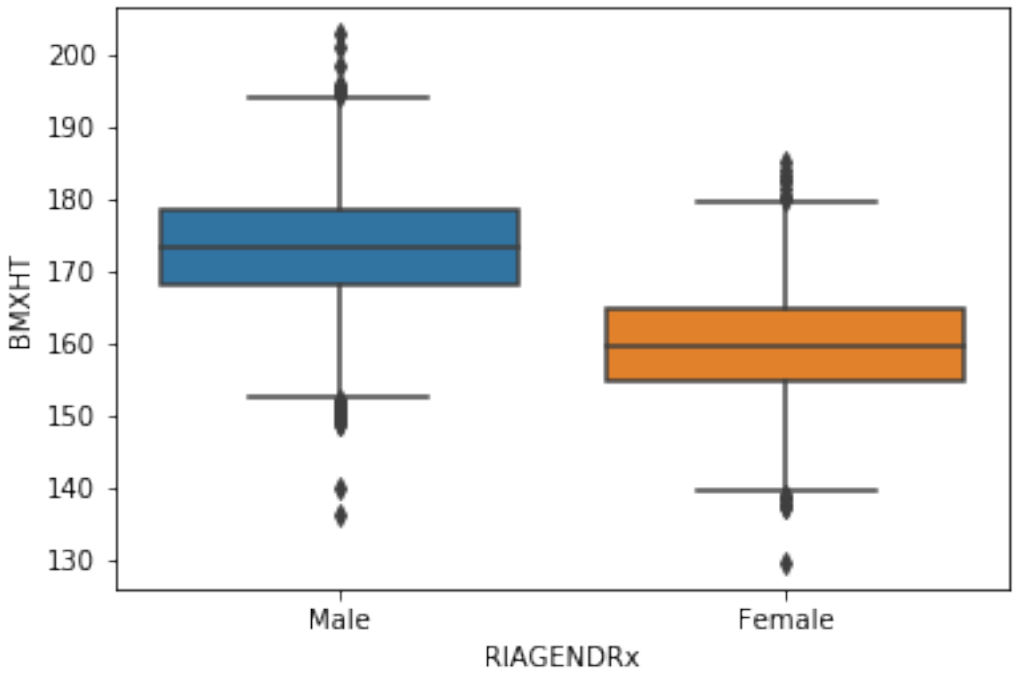
In [11]:
```
# boxplots for males and females

%matplotlib inline
import matplotlib.pyplot as plt

import seaborn as sns
import pandas as pd

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

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

sns.boxplot(x="RIAGENDRx", y="BMXHT", data=da)
```

Out[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f538cde82b0>
```



In [12]:
```
#Code for Q3b, all charts in same 'box'
da["RIAGENDRx"] = da.RIAGENDR.replace({1: "Male", 2: "Female"})
da['BMXHT'].dropna(inplace=True)

g = sns.FacetGrid(da, col = "RIAGENDRx")
g.map(plt.hist, "BMXHT", bins=22) #set number of bins uing the rule: #_bins = (2n)^(1
plt.show()

sns.boxplot(x = 'RIAGENDRx', y = 'BMXHT', data=da)
plt.show()
```






```
In [13]: # DEMO CODE

         # imports
         import pandas as pd
```

```
        # read csv dataset into dataframe
        da = pd.read_csv("nhanes_2015_2016.csv")

        # create new dataframes for Males and Females
        dam = da.where(da.RIAGENDR == 1)
        daf = da.where(da.RIAGENDR == 2)

        # get max mean for age across every combo of SDMVPSU and SDMVSTRA
        agemax = dam.groupby(['SDMVPSU', 'SDMVSTRA'])["RIDAGEYR"].mean().max()

        # get min mean for age across every combo of SDMVPSU and SDMVSTRA
        agemin = dam.groupby(['SDMVPSU', 'SDMVSTRA'])["RIDAGEYR"].mean().min()

        # print max, min and ratio
        print("Male agemax:", agemax)

        print("Male agemin:", agemin)

        print("Male ageratio:", agemax/agemin)
```

```
Male agemax: 55.16528925619835
Male agemin: 42.06315789473684
Male ageratio: 1.3114871069416525
```

```
In [15]: # Univariate Practice Q5a

        # imports
        import pandas as pd

        # read the data
        da = pd.read_csv("nhanes_2015_2016.csv")

        # recode the educational variable
        da["DMDEDUC2x"] = da.DMDEDUC2.replace({1: "A:<9", 2: "B:9-11", 3: "C:HS/GED", 4: "D:S
                                               7: "F:Refused", 9: "G:Don't know"})
        # obtain counts for each level of household size by the grouping variable
        dx = da.groupby(["DMDEDUC2x"])["DMDHHSIZ"].value_counts()

        # restructure the results from 'long' to 'wide'
        dx = dx.unstack()

        # normalize within each stratum to get proportions that sum to 1
        dx = dx.apply(lambda x: x/x.sum(), axis=1)

        # print the results and format to three decimal points
        print(dx.to_string(float_format="%.3f"))
```

```
DMDHHSIZ              1     2     3     4     5     6     7
```

```
DMDEDUC2x
A:<9               0.110 0.224 0.147 0.133 0.148 0.108 0.130
B:9-11             0.117 0.222 0.163 0.152 0.146 0.114 0.086
C:HS/GED           0.153 0.271 0.171 0.162 0.110 0.066 0.068
D:Some college/AA  0.151 0.269 0.193 0.169 0.122 0.051 0.045
E:College          0.143 0.348 0.194 0.165 0.095 0.029 0.026
G:Don't know         NaN 0.667   NaN   NaN 0.333   NaN   NaN
```

In [23]:
```
# DEMO CODE FOR Q5b

# imports
import pandas as pd

# read the csv file
da = pd.read_csv("nhanes_2015_2016.csv")

# restrict ages 30 to 40
da["agegrp"]=pd.cut(da.RIDAGEYR,[30,40])

# recode gender variable
da["RIAGENDRx"] = da.RIAGENDR.replace({1: "Male", 2: "Female"})

# recode educational variable
da["DMDEDUC2x"] = da.DMDEDUC2.replace({1: "A:<9", 2: "B:9-11", 3: "C:HS/GED", 4: "D:S
                                       7: "F:Refused", 9: "G:Don't know"})

# obtain descriptives for household size by grouping variables
dx=da.groupby(["agegrp","DMDEDUC2x","RIAGENDRx"])["DMDHHSIZ"].value_counts()
dx
# find the column names so that we can print median only
#print(dx.columns)
```

Out[23]:
```
agegrp    DMDEDUC2x          RIAGENDRx  DMDHHSIZ
(30, 40]  A:<9               Female     6           13
                                        5           10
                                        7            9
                                        4            7
                                        3            4
                                        2            2
                                        1            1
                             Male       7           14
                                        5           12
                                        3            7
                                        4            7
                                        6            5
                                        2            2
                  B:9-11     Female     5           11
```

```
                                    6            9
                                    2            7
                                    4            7
                                    7            4
                                    1            2
                                    3            2
                          Male      4           16
                                    5           16
                                    6           13
                                    3           10
                                    2            6
                                    7            6
                                    1            2
          C:HS/GED        Female    3           16
                                    4           15
                                    6           15
                                                ..
                          Male      6           11
                                    1           10
          D:Some college/AA Female  4           43
                                    5           42
                                    3           29
                                    2           16
                                    7           15
                                    6           12
                                    1            2
                          Male      4           27
                                    5           25
                                    3           18
                                    1           17
                                    2           15
                                    6           13
                                    7            8
          E:College       Female    4           49
                                    3           34
                                    2           23
                                    5           18
                                    1           14
                                    6            7
                                    7            4
                          Male      4           33
                                    3           28
                                    2           24
                                    1           12
                                    5           12
                                    6            4
                                    7            2
Name: DMDHHSIZ, Length: 69, dtype: int64
```

```
In [24]: dx.describe()

Out[24]: count    69.000000
         mean     13.507246
         std      10.290806
         min       1.000000
         25%       7.000000
         50%      12.000000
         75%      16.000000
         max      49.000000
         Name: DMDHHSIZ, dtype: float64

In [25]: dx.median()

Out[25]: 12.0
```