Untitled7

June 24, 2020

In [1]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

In [2]:
```
#reading csv file using pandas
da = pd.read_csv("nhanes_2015_2016.csv")
```

In [3]:
```
#value_counts is used to determine the number of time each dat has appeared distinctly
da.DMDEDUC2.value_counts()
```

Out[3]:
```
4.0    1621
5.0    1366
3.0    1186
1.0     655
2.0     643
9.0       3
Name: DMDEDUC2, dtype: int64
```

In [5]:
```
#use of sum function, manually summing and determining the shape
print(da.DMDEDUC2.value_counts().sum())
print(1621+ 1366+ 1186+ 655+ 643+3)
print(da.shape)
```

```
5474
5474
(5735, 28)
```

In [6]:
```
#isnull function used to locate all the null values and later determine how many null
pd.isnull(da.DMDEDUC2).sum()
```

Out[6]:
```
261
```

In [7]:
```
# replacing the values and then countiing the results using replace
da["DMDEDUC2x"] = da.DMDEDUC2.replace({1: "<9", 2: "9-11", 3: "HS/GED", 4: "Some colleg
                                       7: "Refused", 9: "Don't know"})
da.DMDEDUC2x.value_counts()
```

Out[7]:
```
Some college/AA    1621
College            1366
HS/GED             1186
<9                  655
9-11                643
Don't know            3
Name: DMDEDUC2x, dtype: int64
```

In [9]:
```
da["RIAGENDRx"] = da.RIAGENDR.replace({1:"Male", 2:"Female" })
da.RIAGENDRx.value_counts()
```

Out[9]:
```
Female    2976
Male      2759
Name: RIAGENDRx, dtype: int64
```

In [13]:
```
# using proportions in x
x = da.DMDEDUC2x.value_counts()
(x / x.sum())
```

Out[13]:
```
Some college/AA    0.296127
College            0.249543
HS/GED             0.216661
<9                 0.119657
9-11               0.117464
Don't know         0.000548
Name: DMDEDUC2x, dtype: float64
```

In [14]:
```
# using percentage in x
x = da.DMDEDUC2x.value_counts()
(x / x.sum())*100
```

Out[14]:
```
Some college/AA    29.612715
College            24.954330
HS/GED             21.666058
<9                 11.965656
9-11               11.746438
Don't know          0.054805
Name: DMDEDUC2x, dtype: float64
```

In [15]:
```
# miising is npow created as another category and is renamed using "fillna"
# the result shows that "missing" is 4.6%
da["DMDEDUC2x"] = da.DMDEDUC2.fillna("Missing")
x = da.DMDEDUC2x.value_counts()
(x / x.sum())*100
```

Out[15]:
```
4.0        28.265039
5.0        23.818657
3.0        20.680035
1.0        11.421099
```

```
2.0        11.211857
Missing     4.551003
9.0         0.052310
Name: DMDEDUC2x, dtype: float64
```

```
In [17]: #quick way of getting numerical summaries in quantitative data using describe an drop
         #you can interchange describe and dropna they will yield similar results
         da.BMXWT.describe().dropna()
```

```
Out[17]: count    5666.000000
         mean       81.342676
         std        21.764409
         min        32.400000
         25%        65.900000
         50%        78.200000
         75%        92.700000
         max       198.900000
         Name: BMXWT, dtype: float64
```

```
In [25]: #individual summary statistics for one dataset using pandas and numpy
         x = da.BMXWT.dropna() # extract all the missing data in dropna
         print(x.mean()) #pandas method
         print(np.mean(x))  # using numpy(put the 'x' in bracket as it can lead to error)

         print(x.median()) #pandas method to get median
         print(np.median(x)) #numpy method to get median
         print(np.percentile(x, 50)) # same as median(numpy)
         print(np.percentile(x, 75)) # to get 75 percentile(numpy)
         print(x.quantile(0.75)) #here quantile is used to get 75 percentile(pandas)
```

```
81.34267560889516
81.34267560889516
78.2
78.2
78.2
92.7
92.7
```

```
In [26]: #frequencies for a systolic blood pressure measurement (BPXSY1).
         #"BPX" here is the NHANES prefix for blood pressure measurements.
         #"SY" stands for "systolic" blood pressure (blood pressure at the peak of a heartbeat
         #"1" indicates that this is the first of three systolic blood presure measurements ta
         #A person is generally considered to have pre-hypertension when their systolic blood 
         #Considering only the systolic condition, we can calculate the proprotion of the NHAN

         np.mean((da.BPXSY1 >= 120) & (da.BPXSY2 <= 139))
```

```
Out[26]: 0.3741935483870968
```

In [27]:
```
np.mean((da.BPXSY1 >= 80) & (da.BPXSY2 <= 89))
```

Out[27]:
```
0.002789886660854403
```

In [29]:
```
a = (da.BPXSY1 >= 120) & (da.BPXSY2 <= 139)
b = (da.BPXSY1 >= 80) & (da.BPXSY2 <= 89)
print(np.mean(a | b)) # "|" means "or"
```

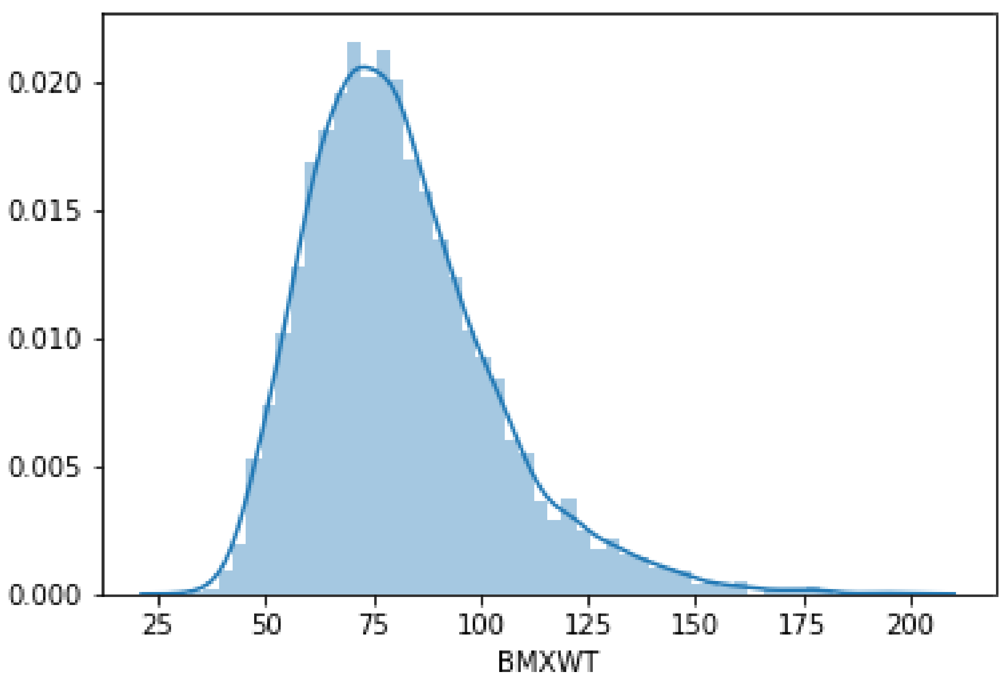
```
0.3769834350479512
```

In [30]:
```
print(np.mean(da.BPXSY1 - da.BPXSY2))
print(np.mean(da.BPXDI1 - da.BPXDI2))
```

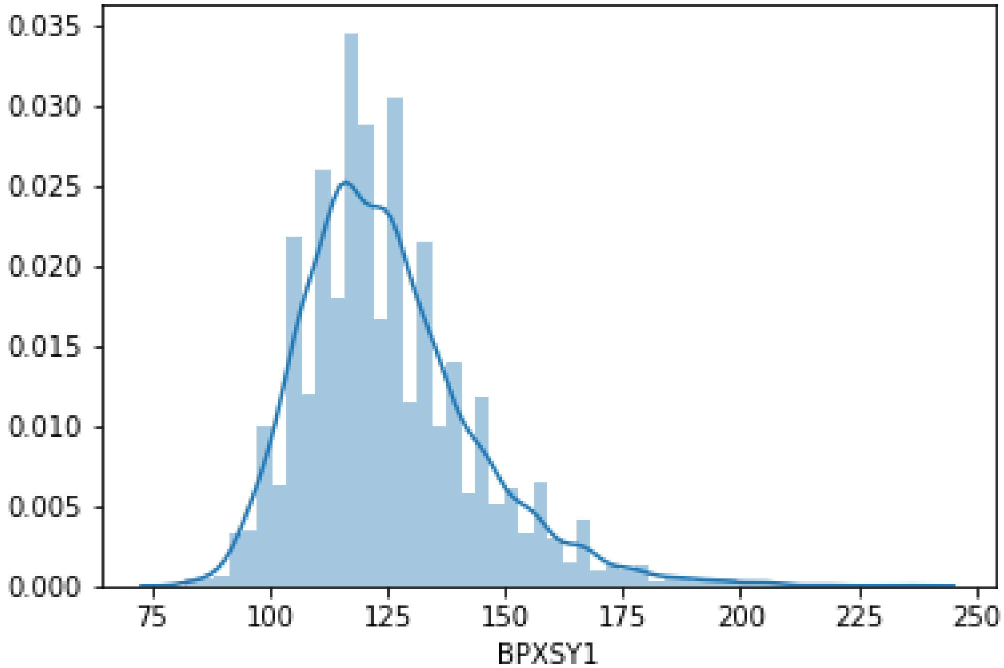
```
0.6749860309182343
0.3490407897187558
```

In [31]:
```
#Graphical summaries
#distribution of body weight (in kg)
sns.distplot(da.BMXWT.dropna())
```

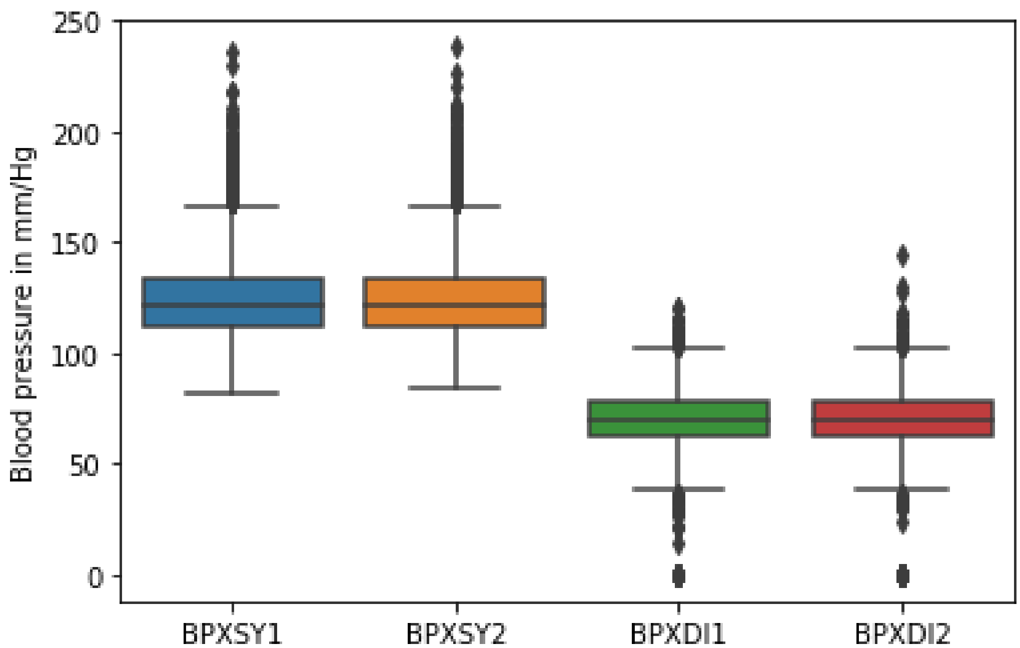
Out[31]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f32952a3da0>
```



In [36]:
```
#histogram of systolic blood pressure measurements
sns.distplot(da.BPXSY1.dropna())
plt.show()
```




```
In [39]: # use of boxplots to measure systloic1, 2 and diastolic 1, 2
         bp = sns.boxplot(data = da.loc[:, ["BPXSY1", "BPXSY2", "BPXDI1", "BPXDI2"]])
         bp = bp.set_ylabel("Blood pressure in mm/Hg")
```

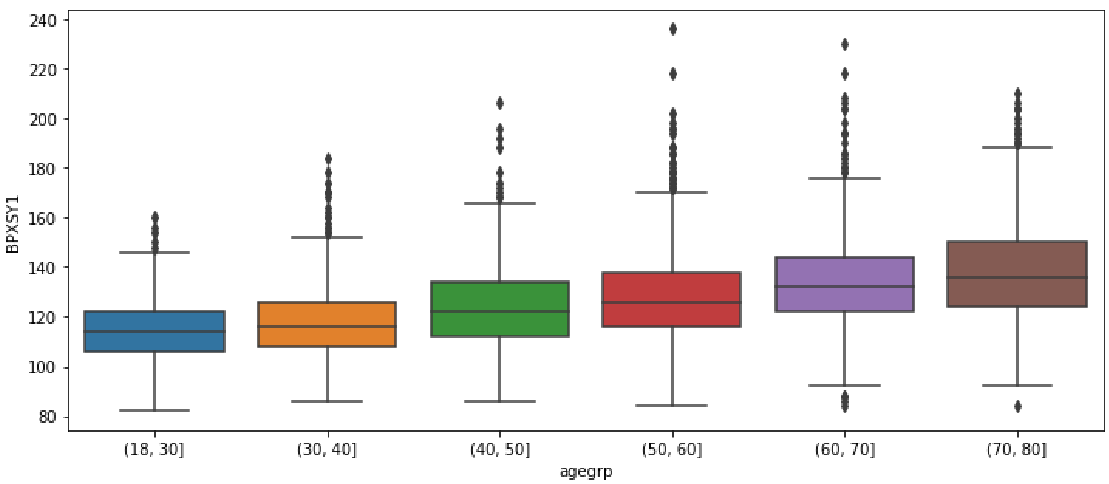


0.0.1 Stratification

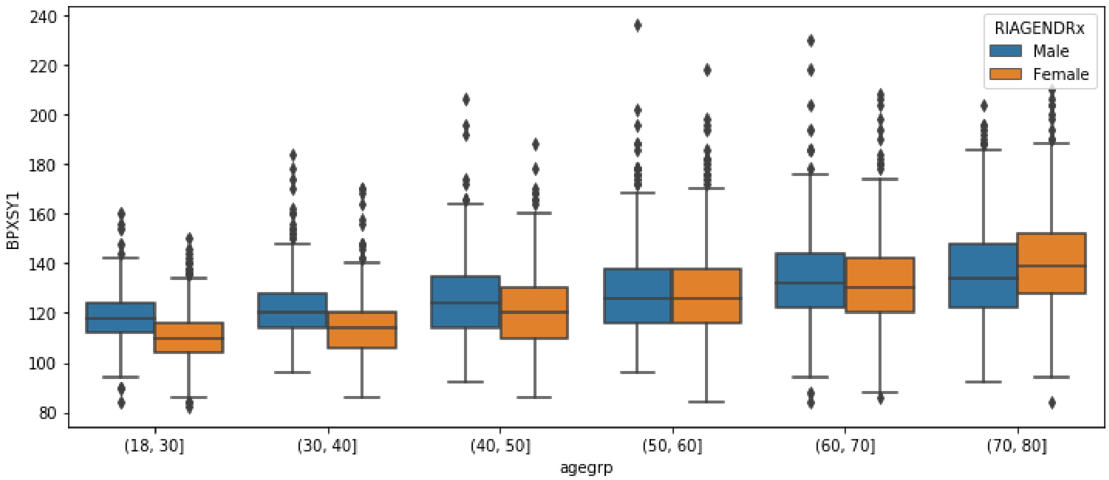
One of the most effective ways to get more information out of a dataset is to divide it into smaller, more uniform subsets, and analyze each of these "strata" on its own. We can then formally or informally compare the findings in the different strata. When working with human subjects, it is very common to stratify on demographic factors such as age, sex, and race.

To illustrate this technique, consider blood pressure, which is a value that tends to increase with age. To see this trend in the NHANES data, we can partition the data into age strata, and construct side-by-side boxplots of the systolic blood pressure (SBP) distribution within each stratum. Since age is a quantitative variable, we need to create a series of "bins" of similar SBP values in order to stratify the data. Each box in the figure below is a summary of univariate data within a specific population stratum (here defined by age).

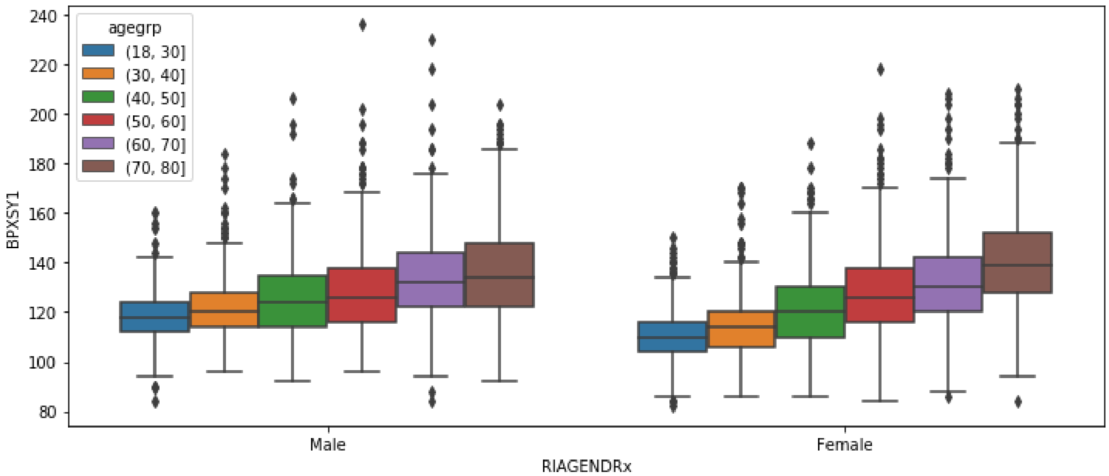
```
In [41]: da["agegrp"] = pd.cut(da.RIDAGEYR, [18, 30, 40, 50, 60, 70, 80]) # Create age strata 
         plt.figure(figsize=(12, 5))  # Make the figure wider than default (12cm wide by 5cm t
         sns.boxplot(x="agegrp", y="BPXSY1", data=da)  # Make boxplot of BPXSY1 stratified by 
         plt.show()
```




```
In [43]: a["agegrp"] = pd.cut(da.RIDAGEYR, [18, 30, 40, 50, 60, 70, 80]) # Create age strata b
         plt.figure(figsize=(12, 5))  # Make the figure wider than default (12cm wide by 5cm t
         sns.boxplot(x="agegrp", y="BPXSY1",hue = "RIAGENDRx", data=da)  # Make boxplot of BPX
         plt.show()
```




```
In [44]: a["agegrp"] = pd.cut(da.RIDAGEYR, [18, 30, 40, 50, 60, 70, 80]) # Create age strata b
         plt.figure(figsize=(12, 5))  # Make the figure wider than default (12cm wide by 5cm t
         sns.boxplot(x="RIAGENDRx", y="BPXSY1",hue = "agegrp", data=da)  # Make boxplot of BPX
         plt.show()
```




```
In [48]: da["DMDEDUC2x"] = da.DMDEDUC2.replace({1: "<9", 2: "9-11", 3: "HS/GED", 4: "Some colle
                                               7: "Refused", 9: "Don't know"})
         da.groupby("agegrp")["DMDEDUC2x"].value_counts()

Out[48]: agegrp    DMDEDUC2x
         (18, 30]  Some college/AA     364
                   College             278
                   HS/GED              237
```

```
             9-11                  99
             <9                    47
(30, 40]     Some college/AA      282
             College              264
             HS/GED               182
             9-11                 111
             <9                    93
(40, 50]     Some college/AA      262
             College              260
             HS/GED               171
             9-11                 112
             <9                    98
(50, 60]     Some college/AA      258
             College              220
             HS/GED               220
             9-11                 122
             <9                   104
(60, 70]     Some college/AA      238
             HS/GED               192
             College              188
             <9                   149
             9-11                 111
(70, 80]     Some college/AA      217
             HS/GED               184
             <9                   164
             College              156
             9-11                  88
             Don't know             3
Name: DMDEDUC2x, dtype: int64
```

In [49]:
```
dx = da.loc[~da.DMDEDUC2x.isin(["Don't know", "Missing"]), :]  # Eliminate rare/missi
dx = dx.groupby(["agegrp", "RIAGENDRx"])["DMDEDUC2x"]
dx = dx.value_counts()
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 pro
print(dx.to_string(float_format="%.3f"))  # Limit display to 3 decimal places
```

```
DMDEDUC2x           9-11    <9  College  HS/GED  Some college/AA
agegrp   RIAGENDRx
(18, 30] Female    0.080 0.049    0.282   0.215            0.374
         Male      0.117 0.042    0.258   0.250            0.333
(30, 40] Female    0.089 0.097    0.314   0.165            0.335
         Male      0.151 0.103    0.251   0.227            0.269
(40, 50] Female    0.110 0.106    0.299   0.173            0.313
         Male      0.142 0.112    0.274   0.209            0.262
(50, 60] Female    0.117 0.102    0.245   0.234            0.302
         Male      0.148 0.123    0.231   0.242            0.256
(60, 70] Female    0.118 0.188    0.195   0.206            0.293
```

	Male	0.135	0.151	0.233	0.231	0.249
(70, 80]	Female	0.105	0.225	0.149	0.240	0.281
	Male	0.113	0.180	0.237	0.215	0.255