w2_assessment

August 9, 2020

In this notebook, we'll ask you to find numerical summaries for a certain set of data. You will use the values of what you find in this assignment to answer questions in the quiz that follows (we've noted where specific values will be requested in the quiz, so that you can record them.)

We'll also ask you to create some of the plots you have seen in previous lectures.

In [2]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import scipy.stats as stats
%matplotlib inline
import matplotlib.pyplot as plt
pd.set_option('display.max_columns', 100)

path = "nhanes_2015_2016.csv"
```

In [3]:
```
# First, you must import the data from the path given above
df = pd.read_csv(path)
df
# using pandas, read in the csv data found at the url defined by 'path'
```

Out[3]:

	SEQN	ALQ101	ALQ110	ALQ130	SMQ020	RIAGENDR	RIDAGEYR	RIDRETH1	\
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	
5	83737	2.0	2.0	NaN	2	2	72	1	
6	83741	1.0	NaN	8.0	1	1	22	4	
7	83742	1.0	NaN	1.0	2	2	32	1	
8	83743	NaN	NaN	NaN	2	1	18	5	
9	83744	1.0	NaN	NaN	2	1	56	4	
10	83747	1.0	NaN	1.0	1	1	46	3	
11	83750	1.0	NaN	3.0	1	1	45	5	
12	83752	1.0	NaN	2.0	1	2	30	2	
13	83754	2.0	1.0	1.0	2	2	67	2	
14	83755	1.0	NaN	3.0	2	1	67	4	
15	83757	1.0	NaN	1.0	2	2	57	2	
16	83759	2.0	2.0	NaN	2	2	19	1	

17	83761	1.0	NaN	1.0	2	2	24	5
18	83762	NaN	NaN	NaN	1	2	27	4
19	83767	2.0	2.0	NaN	2	2	54	5
20	83769	1.0	NaN	2.0	2	1	49	5
21	83773	2.0	2.0	NaN	2	2	80	3
22	83775	2.0	1.0	NaN	1	2	69	2
23	83776	NaN	NaN	NaN	2	2	58	1
24	83777	2.0	2.0	NaN	2	1	56	5
25	83781	1.0	NaN	3.0	2	2	27	4
26	83784	1.0	NaN	4.0	1	1	22	2
27	83785	2.0	1.0	1.0	1	2	60	2
28	83786	NaN	NaN	NaN	2	1	51	4
29	83787	2.0	2.0	NaN	2	2	68	1
...	...	...	...	...	...	...	...	...
5705	93645	2.0	1.0	NaN	1	1	80	3
5706	93652	1.0	NaN	NaN	1	1	72	2
5707	93653	1.0	NaN	3.0	2	2	25	3
5708	93654	2.0	2.0	NaN	2	2	29	5
5709	93655	1.0	NaN	NaN	1	1	38	3
5710	93656	2.0	2.0	NaN	2	2	75	1
5711	93659	1.0	NaN	1.0	1	1	62	1
5712	93661	1.0	NaN	2.0	2	2	27	2
5713	93663	1.0	NaN	4.0	2	1	43	1
5714	93664	2.0	1.0	2.0	2	1	39	1
5715	93665	NaN	NaN	NaN	2	2	34	3
5716	93668	1.0	NaN	NaN	1	2	73	1
5717	93670	1.0	NaN	3.0	1	1	32	3
5718	93671	1.0	NaN	3.0	2	1	45	4
5719	93672	2.0	2.0	NaN	1	2	63	2
5720	93675	1.0	NaN	1.0	2	1	38	5
5721	93676	1.0	NaN	2.0	2	2	35	4
5722	93677	1.0	NaN	1.0	2	2	34	3
5723	93679	2.0	1.0	NaN	1	2	72	4
5724	93682	NaN	NaN	NaN	2	2	41	5
5725	93684	2.0	2.0	NaN	2	1	34	4
5726	93685	1.0	NaN	2.0	1	1	53	1
5727	93689	2.0	1.0	NaN	2	2	69	1
5728	93690	1.0	NaN	3.0	2	1	32	2
5729	93691	2.0	2.0	NaN	2	1	25	5
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

	DMDCITZN	DMDEDUC2	DMDMARTL	DMDHHSIZ	WTINT2YR	SDMVPSU	SDMVSTRA	\
0	1.0	5.0	1.0	2	134671.37	1	125	
1	2.0	3.0	3.0	1	24328.56	1	125	

2	1.0	3.0	1.0	2	12400.01	1	131
3	1.0	5.0	6.0	1	102718.00	1	131
4	1.0	4.0	3.0	5	17627.67	2	126
5	2.0	2.0	4.0	5	11252.31	1	128
6	1.0	4.0	5.0	3	37043.09	2	128
7	2.0	4.0	1.0	4	22744.36	1	125
8	1.0	NaN	NaN	3	18526.16	2	122
9	1.0	3.0	3.0	1	20395.54	2	126
10	1.0	5.0	6.0	2	34513.08	1	121
11	1.0	2.0	5.0	5	96194.93	1	125
12	1.0	4.0	6.0	2	36978.42	1	124
13	1.0	5.0	1.0	7	10495.87	1	128
14	1.0	5.0	2.0	1	14080.10	1	126
15	1.0	1.0	4.0	5	11709.11	1	120
16	1.0	NaN	NaN	3	18928.63	2	121
17	2.0	5.0	5.0	1	15420.80	1	119
18	1.0	4.0	5.0	2	38387.34	1	132
19	1.0	4.0	3.0	6	21914.04	2	122
20	1.0	2.0	1.0	4	26837.74	1	132
21	1.0	3.0	2.0	1	37582.05	1	131
22	1.0	1.0	4.0	1	7250.10	2	119
23	1.0	5.0	1.0	3	21433.18	2	128
24	1.0	2.0	1.0	5	13892.04	2	133
25	1.0	5.0	5.0	2	34068.86	2	119
26	1.0	4.0	5.0	4	38148.53	2	121
27	1.0	5.0	3.0	4	10495.87	1	128
28	1.0	4.0	5.0	2	25558.45	2	129
29	1.0	1.0	3.0	1	8842.35	1	128
...	...	...	...	...	...	...	...
5705	1.0	5.0	3.0	1	19322.78	1	121
5706	1.0	2.0	1.0	2	10706.04	1	125
5707	1.0	5.0	6.0	2	110170.18	1	132
5708	2.0	5.0	1.0	2	23951.40	1	126
5709	1.0	2.0	3.0	4	39265.95	2	131
5710	1.0	3.0	2.0	1	11085.15	1	127
5711	2.0	1.0	1.0	7	8083.00	1	120
5712	1.0	4.0	6.0	4	24442.12	1	127
5713	2.0	1.0	1.0	6	24831.29	1	120
5714	2.0	3.0	1.0	5	25064.11	1	120
5715	1.0	5.0	1.0	4	115406.09	1	121
5716	1.0	1.0	2.0	1	12211.03	1	128
5717	2.0	5.0	1.0	4	120187.36	2	123
5718	1.0	5.0	1.0	3	21351.55	1	124
5719	1.0	1.0	1.0	3	8605.87	2	119
5720	2.0	5.0	4.0	1	26328.59	2	122
5721	1.0	5.0	3.0	2	32944.36	2	132
5722	1.0	5.0	5.0	3	136319.80	1	122
5723	1.0	2.0	2.0	3	21037.26	2	132

5724	1.0	5.0	1.0	5	23924.20	2	121
5725	1.0	5.0	1.0	5	29880.64	2	131
5726	2.0	2.0	1.0	5	22441.86	1	126
5727	1.0	1.0	1.0	1	9611.19	2	127
5728	1.0	2.0	1.0	4	43971.60	2	127
5729	2.0	5.0	5.0	7	13525.39	2	133
5730	1.0	3.0	2.0	1	58614.08	2	130
5731	1.0	5.0	1.0	3	122920.60	1	121
5732	1.0	4.0	2.0	1	49050.06	2	132
5733	2.0	1.0	1.0	5	42314.29	1	126
5734	1.0	5.0	5.0	3	107361.91	2	119

	INDFMPIR	BPXSY1	BPXDI1	BPXSY2	BPXDI2	BMXWT	BMXHT	BMXBMI	BMXLEG	\
0	4.39	128.0	70.0	124.0	64.0	94.8	184.5	27.8	43.3	
1	1.32	146.0	88.0	140.0	88.0	90.4	171.4	30.8	38.0	
2	1.51	138.0	46.0	132.0	44.0	83.4	170.1	28.8	35.6	
3	5.00	132.0	72.0	134.0	68.0	109.8	160.9	42.4	38.5	
4	1.23	100.0	70.0	114.0	54.0	55.2	164.9	20.3	37.4	
5	2.82	116.0	58.0	122.0	58.0	64.4	150.0	28.6	34.4	
6	2.08	110.0	70.0	112.0	74.0	76.6	165.4	28.0	38.8	
7	1.03	120.0	70.0	114.0	70.0	64.5	151.3	28.2	34.1	
8	5.00	NaN	NaN	NaN	NaN	72.4	166.1	26.2	NaN	
9	1.19	178.0	116.0	180.0	114.0	108.3	179.4	33.6	46.0	
10	0.75	144.0	94.0	150.0	90.0	86.2	176.7	27.6	41.0	
11	1.36	116.0	70.0	108.0	72.0	76.2	177.8	24.1	43.9	
12	5.00	104.0	50.0	104.0	50.0	71.2	163.6	26.6	37.3	
13	0.89	124.0	76.0	116.0	64.0	117.8	164.1	43.7	34.8	
14	2.04	132.0	84.0	136.0	82.0	97.4	183.8	28.8	42.5	
15	0.77	134.0	68.0	146.0	62.0	80.5	150.8	35.4	31.6	
16	1.74	102.0	68.0	102.0	66.0	100.8	175.4	32.8	40.7	
17	0.00	110.0	62.0	108.0	60.0	61.8	156.4	25.3	37.0	
18	2.12	138.0	76.0	144.0	84.0	107.9	168.5	38.0	40.1	
19	2.99	136.0	82.0	126.0	82.0	59.0	149.9	26.3	32.7	
20	2.97	110.0	70.0	106.0	68.0	72.8	170.7	25.0	42.8	
21	3.57	148.0	56.0	146.0	68.0	67.7	149.8	30.2	33.4	
22	0.55	140.0	56.0	132.0	48.0	77.7	160.2	30.3	32.7	
23	3.72	116.0	62.0	110.0	62.0	56.6	157.5	22.8	34.5	
24	1.35	136.0	86.0	132.0	84.0	69.0	166.1	25.0	40.7	
25	NaN	108.0	66.0	110.0	66.0	87.8	160.7	34.0	41.4	
26	NaN	122.0	78.0	124.0	84.0	73.7	170.7	25.3	41.2	
27	5.00	142.0	74.0	136.0	74.0	75.6	145.2	35.9	31.0	
28	NaN	132.0	80.0	140.0	80.0	102.1	182.2	30.8	41.8	
29	1.49	122.0	58.0	124.0	58.0	77.4	152.0	33.5	33.4	
...	...	...	...	...	...	...	...	...	...	
5705	1.57	174.0	0.0	168.0	0.0	NaN	165.9	NaN	37.0	
5706	0.83	124.0	36.0	132.0	30.0	88.9	165.9	32.3	36.7	
5707	2.97	130.0	84.0	132.0	84.0	76.9	169.3	26.8	41.8	
5708	NaN	102.0	82.0	100.0	72.0	54.5	152.6	23.4	36.3	

5709	1.15	132.0	60.0	126.0	60.0	75.9	177.8	24.0	40.5
5710	3.40	NaN	NaN	134.0	44.0	92.7	156.4	37.9	31.5
5711	3.27	144.0	94.0	140.0	88.0	67.9	164.6	25.1	33.7
5712	0.74	114.0	62.0	106.0	64.0	101.8	157.6	41.0	36.5
5713	1.07	116.0	82.0	114.0	74.0	75.7	177.1	24.1	43.0
5714	4.93	162.0	94.0	166.0	94.0	74.9	169.6	26.0	40.5
5715	4.12	124.0	76.0	126.0	84.0	69.0	169.6	24.0	38.9
5716	0.35	NaN	NaN	NaN	NaN	119.6	149.0	53.9	NaN
5717	NaN	112.0	70.0	112.0	72.0	81.4	170.1	28.1	41.7
5718	5.00	128.0	60.0	120.0	58.0	89.0	174.3	29.3	42.4
5719	0.20	NaN	NaN	NaN	NaN	81.9	147.6	37.6	35.0
5720	5.00	110.0	58.0	118.0	62.0	77.6	179.0	24.2	42.3
5721	2.68	118.0	78.0	114.0	76.0	92.2	161.7	35.3	41.5
5722	1.76	114.0	72.0	110.0	78.0	62.2	167.4	22.2	39.8
5723	2.98	142.0	64.0	136.0	78.0	57.8	157.0	23.4	32.0
5724	5.00	132.0	78.0	122.0	84.0	58.2	166.9	20.9	37.1
5725	2.81	110.0	72.0	112.0	72.0	101.2	180.9	30.9	43.7
5726	0.49	132.0	54.0	128.0	56.0	78.7	156.9	32.0	31.5
5727	0.97	164.0	62.0	166.0	64.0	64.8	151.9	28.1	32.2
5728	5.00	112.0	60.0	118.0	58.0	89.5	164.9	32.9	40.0
5729	1.59	112.0	80.0	112.0	76.0	39.2	136.5	21.0	33.6
5730	1.43	112.0	48.0	112.0	46.0	59.1	165.8	21.5	38.2
5731	2.99	118.0	68.0	116.0	76.0	112.1	182.2	33.8	43.4
5732	2.97	154.0	56.0	146.0	58.0	71.7	152.2	31.0	31.3
5733	0.00	104.0	62.0	106.0	66.0	78.2	173.3	26.0	40.3
5734	3.54	118.0	66.0	114.0	68.0	58.3	165.0	21.4	38.2

	BMXARML	BMXARMC	BMXWAIST	HIQ210
0	43.6	35.9	101.1	2.0
1	40.0	33.2	107.9	NaN
2	37.0	31.0	116.5	2.0
3	37.7	38.3	110.1	2.0
4	36.0	27.2	80.4	2.0
5	33.5	31.4	92.9	NaN
6	38.0	34.0	86.6	NaN
7	33.1	31.5	93.3	2.0
8	NaN	NaN	NaN	2.0
9	44.1	38.5	116.0	2.0
10	38.0	33.6	104.3	2.0
11	37.8	33.0	90.1	NaN
12	35.7	31.0	90.7	2.0
13	38.6	42.7	123.0	2.0
14	40.6	34.2	106.3	2.0
15	32.7	33.7	113.5	2.0
16	38.6	35.9	104.6	1.0
17	35.5	29.6	79.5	NaN
18	39.0	41.6	114.8	1.0
19	33.3	30.4	88.9	2.0

```
20       40.0     31.5      96.6     2.0
21       36.2     30.3     108.4     2.0
22       37.6     30.7     106.8     2.0
23       31.0     28.0      83.0     2.0
24       37.1     31.0      93.5     2.0
25       33.7     32.1     103.6     2.0
26       36.0     29.7      86.2     2.0
27       33.1     36.0     108.0     2.0
28       38.9     37.8     107.7     2.0
29       34.6     34.9     103.1     2.0
...       ...      ...       ...     ...
5705     38.0     30.0      95.0     2.0
5706     38.8     34.2     114.2     2.0
5707     35.9     29.8      79.0     1.0
5708     32.7     26.6      80.2     9.0
5709     39.8     30.2      92.0     2.0
5710     35.5     33.6     122.0     2.0
5711     37.0     31.5      90.7     2.0
5712     34.5     37.1     127.0     1.0
5713     39.7     31.3      94.6     1.0
5714     39.3     31.5      92.5     2.0
5715     37.5     27.6      91.5     2.0
5716     35.5     44.7       NaN     1.0
5717     38.2     33.9      93.6     1.0
5718     40.6     39.5      97.0     2.0
5719     33.0     38.2     122.5     2.0
5720     41.5     31.7      94.2     2.0
5721     37.5     38.9     110.9     2.0
5722     37.7     28.5      83.3     2.0
5723     35.5     26.8      90.4     2.0
5724     35.3     26.9      80.8     2.0
5725     43.0     41.3      99.0     2.0
5726     38.0     33.7     107.5     2.0
5727     32.6     28.7     101.1     2.0
5728     38.0     39.0     101.0     2.0
5729     29.7     23.8      75.4     2.0
5730     37.0     29.5      95.0     2.0
5731     41.8     42.3     110.2     2.0
5732     37.5     28.8       NaN     2.0
5733     37.5     30.6      98.9     2.0
5734     33.5     26.2      72.5     2.0

[5735 rows x 28 columns]
```

In [3]:
```
# Next, look at the 'head' of our DataFrame 'df'.
df.head()
```

```
        # If you can't remember a function, open a previous notebook or video as a reference
        # or use your favorite search engine to look for a solution
```

Out[3]:

	SEQN	ALQ101	ALQ110	ALQ130	SMQ020	RIAGENDR	RIDAGEYR	RIDRETH1	\
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	

	DMDCITZN	DMDEDUC2	DMDMARTL	DMDHHSIZ	WTINT2YR	SDMVPSU	SDMVSTRA	\
0	1.0	5.0	1.0	2	134671.37	1	125	
1	2.0	3.0	3.0	1	24328.56	1	125	
2	1.0	3.0	1.0	2	12400.01	1	131	
3	1.0	5.0	6.0	1	102718.00	1	131	
4	1.0	4.0	3.0	5	17627.67	2	126	

	INDFMPIR	BPXSY1	BPXDI1	BPXSY2	BPXDI2	BMXWT	BMXHT	BMXBMI	BMXLEG	\
0	4.39	128.0	70.0	124.0	64.0	94.8	184.5	27.8	43.3	
1	1.32	146.0	88.0	140.0	88.0	90.4	171.4	30.8	38.0	
2	1.51	138.0	46.0	132.0	44.0	83.4	170.1	28.8	35.6	
3	5.00	132.0	72.0	134.0	68.0	109.8	160.9	42.4	38.5	
4	1.23	100.0	70.0	114.0	54.0	55.2	164.9	20.3	37.4	

	BMXARML	BMXARMC	BMXWAIST	HIQ210
0	43.6	35.9	101.1	2.0
1	40.0	33.2	107.9	NaN
2	37.0	31.0	116.5	2.0
3	37.7	38.3	110.1	2.0
4	36.0	27.2	80.4	2.0

How many rows can you see when you don't put an argument into the previous method?
How many rows can you see if you use an int as an argument?
Can you use a float as an argument?

```
In [5]: # Lets only consider the feature (or variable) 'BPXSY2'
        bp = df['BPXSY2']
        print("sum:",np.sum(bp),", len :",len(df.index), "mean :- ", (np.sum(bp))/(len(df.inde
```

```
sum: 690674.0 , len : 5735 mean :-  120.43138622493461
```

0.1 Numerical Summaries

0.1.1 Find the mean (note this for the quiz that follows)

```
In [5]: # What is the mean of 'BPXSY2'?
        bp_mean = np.mean(bp)
        bp_mean
        # mean- 124.78301716350497
```

```
Out[5]: 124.78301716350497
```

In the method you used above, how are the rows of missing data treated? Are the excluded entirely? Are they counted as zeros? Something else? If you used a library function, try looking up the documentation using the code:

```
help(function_you_used)
```

For example:

```
help(np.sum)
```

.dropna() To make sure we know that we aren't treating missing data in ways we don't want, lets go ahead and drop all the nans from our Series 'bp'

```
In [6]: bp = bp.dropna()
```

0.1.2 Find the:

- Median
- Max
- Min
- Standard deviation
- Variance

You can implement any of these from base python (that is, without any of the imported packages), but there are simple and intuitively named functions in the numpy library for all of these. You could also use the fact that 'bp' is not just a list, but is a pandas.Series. You can find pandas.Series attributes and methods here

A large part of programming is being able to find the functions you need and to understand the documentation formatting so that you can implement the code yourself, so we highly encourage you to search the internet whenever you are unsure!

0.1.3 Example:

Find the difference of an element in 'bp' compared with the previous element in 'bp'.

```
In [7]: # Using the fact that 'bp' is a pd.Series object, can use the pd.Series method diff()
        # call this method by: pd.Series.diff()
        diff_by_series_method = bp.diff()
        # note that this returns a pd.Series object, that is, it had an index associated with
        diff_by_series_method.values # only want to see the values, not the index and values
```

```
Out[7]: array([ nan,  16.,  -8., ...,  30., -40.,   8.])
```

```
In [8]: # Now use the numpy library instead to find the same values
        # np.diff(array)
        diff_by_np_method = np.diff(bp)
        diff_by_np_method
        # note that this returns an 'numpy.ndarray', which has no index associated with it, an
        # the nan we get by the Series method
```

```
Out[8]: array([ 16.,  -8.,   2., ...,  30., -40.,   8.])
```

```
In [9]: # We could also implement this ourselves with some looping
        diff_by_me = [] # create an empty list
        for i in range(len(bp.values)-1): # iterate through the index values of bp
            diff = bp.values[i+1] - bp.values[i] # find the difference between an element and
            diff_by_me.append(diff) # append to out list
        np.array(diff_by_me) # format as an np.array
```

```
Out[9]: array([ 16.,  -8.,   2., ...,  30., -40.,   8.])
```

0.1.4 Your turn (note these values for the quiz that follows)

```
In [10]: bp_median = np.median(bp)
         bp_median
```

```
Out[10]: 122.0
```

```
In [11]: bp_max = np.max(bp)
         bp_max
```

```
Out[11]: 238.0
```

```
In [12]: bp_min = np.min(bp)
         bp_min
```

```
Out[12]: 84.0
```

```
In [13]: bp_std = np.std(bp)
         bp_std
```

```
Out[13]: 18.525338021233786
```

```
In [14]: bp_var = np.var(bp)
         bp_var
```

```
Out[14]: 343.1881488009701
```

0.1.5 How to find the interquartile range (note this value for the quiz that follows)

This time we need to use the scipy.stats library that we imported above under the name 'stats'

```
In [15]: bp_iqr = stats.iqr(bp)
         bp_iqr
```

```
Out[15]: 22.0
```

0.2 Visualizing the data

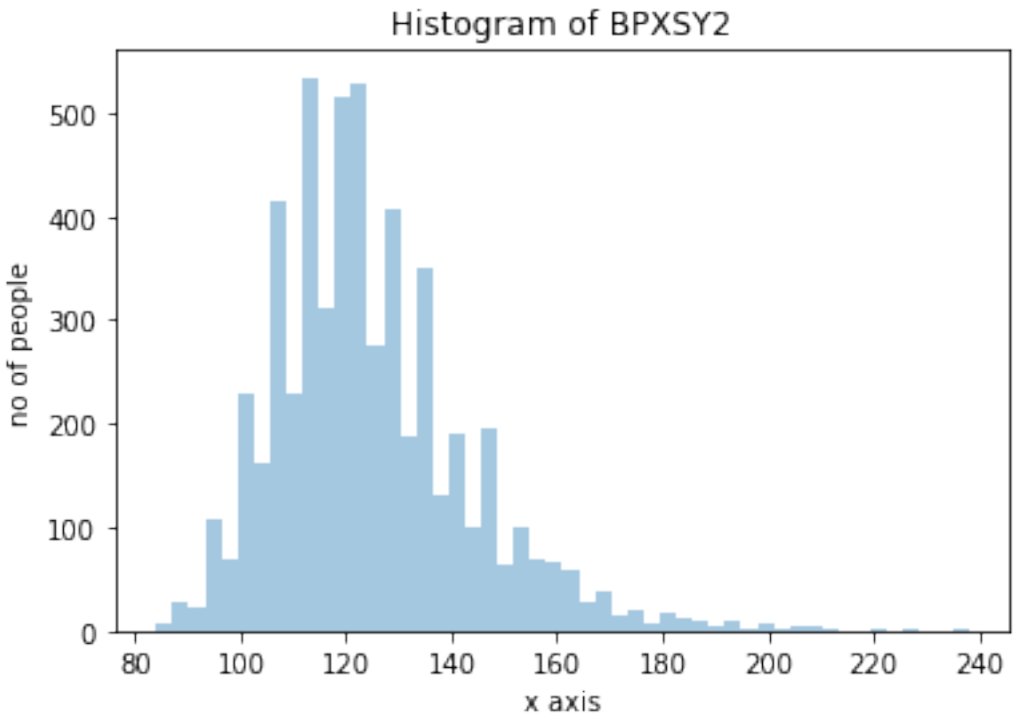
Next we'll use what you have learned from the *Tables, Histograms, Boxplots in Python* video

```
In [16]: # use the Series.describe() method to see some descriptive statistics of our Series '
         bp_descriptive_stats = bp.describe()
         bp_descriptive_stats
```

```
Out[16]: count    5535.000000
         mean      124.783017
         std        18.527012
         min        84.000000
         25%       112.000000
         50%       122.000000
         75%       134.000000
         max       238.000000
         Name: BPXSY2, dtype: float64
```

```
In [21]: # Make a histogram of our 'bp' data using the seaborn library we imported as 'sns'
         sns.distplot(bp,kde=False).set(title='Histogram of BPXSY2', xlabel='x axis', ylabel='
```

```
Out[21]: [Text(0,0.5,'no of people'),
          Text(0.5,0,'x axis'),
          Text(0.5,1,'Histogram of BPXSY2')]
```



Is your histogram labeled and does it have a title? If not, try appending

```
.set(title='your_title', xlabel='your_x_label', ylabel='your_y_label')
```

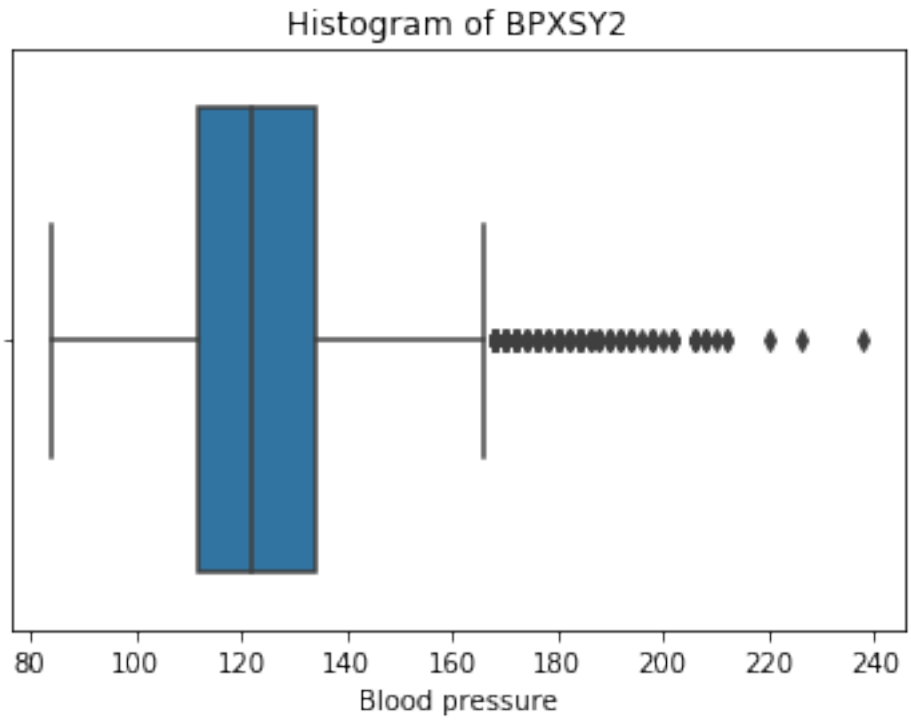
or just

```
.set(title='your_title')
```

to your graphing function

```
In [24]: # Make a boxplot of our 'bp' data using the seaborn library. Make sure it has a title
         sns.boxplot(bp).set(title='Histogram of BPXSY2', xlabel='Blood pressure')

Out[24]: [Text(0.5,0,'Blood pressure'), Text(0.5,1,'Histogram of BPXSY2')]
```




```
In [6]: sns.pairplot(bp)

        ---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call last)

        <ipython-input-6-2514819ba00c> in <module>()
    ----> 1 sns.pairplot(bp)
```

```
 /opt/conda/lib/python3.6/site-packages/seaborn/axisgrid.py in pairplot(data, hue, hue_
2068         raise TypeError(
2069             "'data' must be pandas DataFrame object, not: {typefound}".format(
-> 2070                 typefound=type(data)))
2071
2072     if plot_kws is None:


TypeError: 'data' must be pandas DataFrame object, not: <class 'pandas.core.series.Seri
```