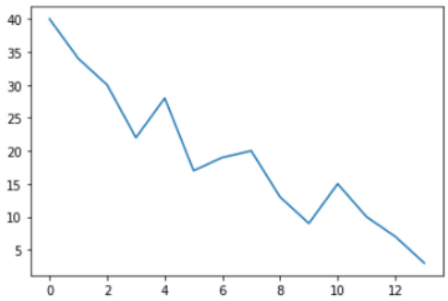
Data Visualization with Pandas

Line Plot

First import pandas. Then, let's just make a basic Series in pandas and make a line plot.

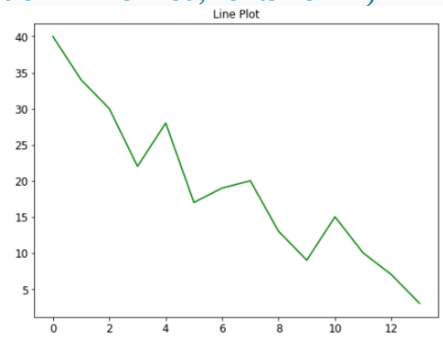
```
import pandas as pd
a = pd.Series([40, 34, 30, 22, 28, 17, 19, 20, 13, 9, 15, 10, 7, 3])
a.plot()
```



Improve plot by adding:

a figure size to make the size of the plot bigger, color to change the default blue color, title on top that shows what this plot is about and font size to change the default font size of those numbers on the axis

```
a.plot(figsize=(8, 6), color='green', title = 'Line Plot', fontsize=12)
```



Area Plot

Use the same series 'a' and make an area plot here,

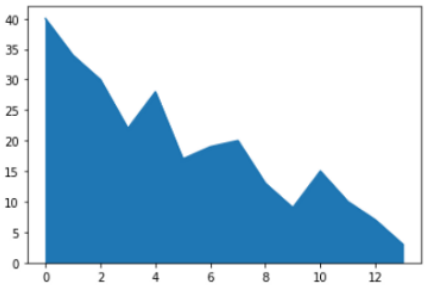
Use the .plot() method and pass a parameter kind to specify the kind of plot I want like:

```
a.plot(kind='area')
```

or

```
a.plot.area()
```

Both of the methods I mentioned above will create this plot:



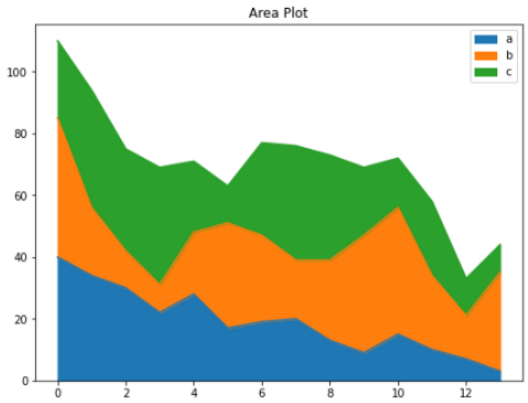
Make a couple more Series, then
Make a DataFrme, and make an area plot from it.

```
b = pd.Series([45, 22, 12, 9, 20, 34, 28, 19, 26, 38, 41, 24, 14, 32])
c = pd.Series([25, 38, 33, 38, 23, 12, 30, 37, 34, 22, 16, 24, 12, 9])
d = pd.DataFrame({'a':a, 'b': b, 'c': c})
```

	a	b	c
0	40	45	25
1	34	22	38
2	30	12	33
3	22	9	38
4	28	20	23
5	17	34	12
6	19	28	30
7	20	19	37
8	13	26	34
9	9	38	22
10	15	41	16
11	10	24	24
12	7	14	12
13	3	32	9

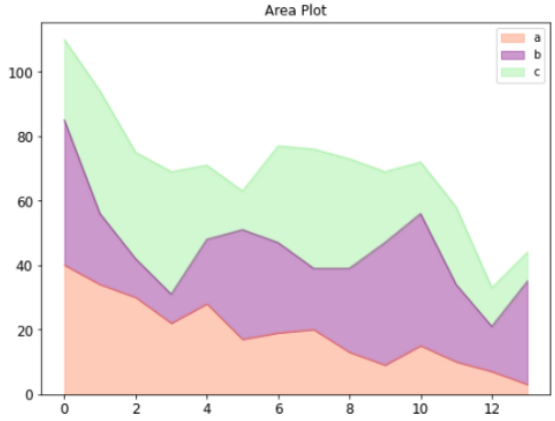
plot this DataFrame ‘d’ as an area plot now,

```
d.plot.area(figsize=(8, 6), title='Area Plot')
```



Change default colors

```
d.plot.area(alpha=0.4, color=['coral', 'purple', 'lightgreen'],figsize=(8, 6), title='Area Plot', fontsize=12)
```



The ‘alpha’ parameter adds some translucent looks to the plot.

It appears to be very useful at times when we have overlapping area plots or histograms or dense scatter plots.

This .plot() can male eleven types of plots:

1. line
2. area
3. bar
4. barh
5. pie
6. box
7. hexbin
8. hist
9. kde
10. density
11. scatter

Use the ***NHANES dataset*** by the Centers for Disease Control and Prevention.
keep it in the same folder as this Jupyter notebook.

download the dataset here

Import the dataset:

```
df = pd.read_csv('nhanes_2015_2016.csv')
df.head()
```

	SEQN	ALQ101	ALQ110	ALQ130	SMQ020	RIAGENDR	RIDAGEYR	RIDRETH1	DMDCITZN	DMDEDUC2	...	BPXSY2	BPXDI2	BMXWT	BMXHT	BMI
0	83732	1.0	NaN	1.0	1	1	62	3	1.0	5.0	...	124.0	64.0	94.8	184.5	
1	83733	1.0	NaN	6.0	1	1	53	3	2.0	3.0	...	140.0	88.0	90.4	171.4	
2	83734	1.0	NaN	NaN	1	1	78	3	1.0	3.0	...	132.0	44.0	83.4	170.1	
3	83735	2.0	1.0	1.0	2	2	56	3	1.0	5.0	...	134.0	68.0	109.8	160.9	
4	83736	2.0	1.0	1.0	2	2	42	4	1.0	4.0	...	114.0	54.0	55.2	164.9	

This dataset has 30 columns and 5735 rows.

check the columns of the dataset:

```
df.columns
```

output:

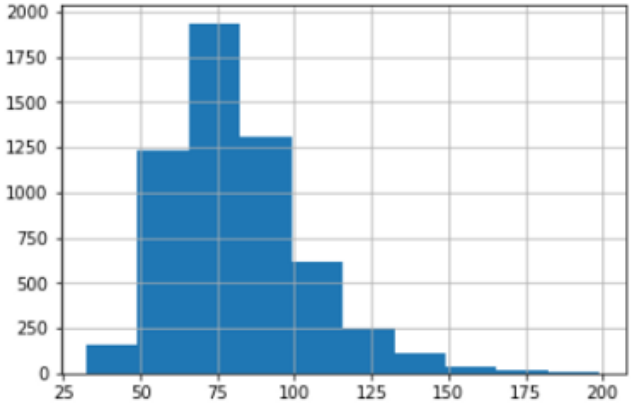
```
Index(['SEQN', 'ALQ101', 'ALQ110', 'ALQ130', 'SMQ020', 'RIAGENDR', 'RIDAGEYR', 'RIDRETH1',
'DMDCITZN', 'DMDEDUC2', 'DMDMARTL', 'DMDHHSIZ', 'WTINT2YR', 'SDMVPSU', 'SDMVSTRA',
'INDFMPIR', 'BPXSY1', 'BPXDI1', 'BPXSY2', 'BPXDI2', 'BMXWT', 'BMXHT', 'BMXBMI', 'BMXLEG',
'BMXARML', 'BMXARMC', 'BMXWAIST', 'HIQ210', 'DMDEDUC2x', 'DMDMARTLx'], dtype='object')
```

we will not use all the columns.
We will use some of them to **practice** these plots.

Histogram

I will use the weight of the population to make a basic histogram.

```
df['BMXWT'].hist()
```



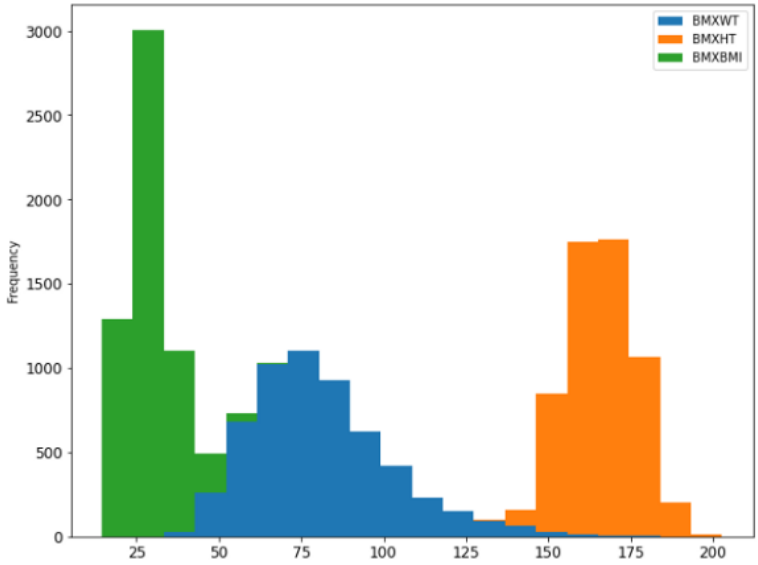
the histogram provides the distribution of frequency.

about 1825 people have a weight of 75. The maximum people have the weight in the range of 49 to 99.

Now put several histograms in one plot

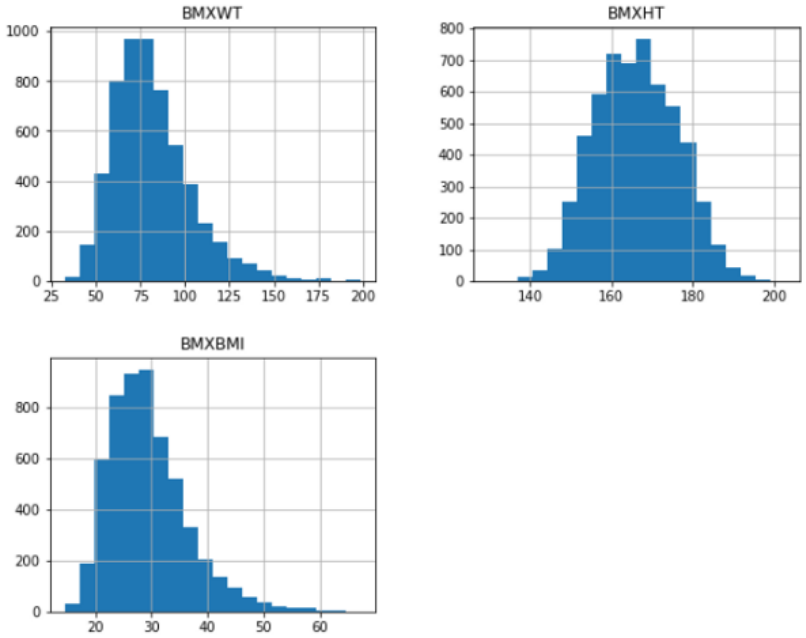
Make three histograms in one plot using weight, height, and body mass index (BMI).

```
df[['BMXWT', 'BMXHT', 'BMXBMI']].plot.hist(stacked=True, bins=20, fontsize=12, figsize=(10, 8))
```



three different histograms:

```
df[['BMXWT', 'BMXHT', 'BMXBMI']].hist(bins=20,figsize=(10, 8))
```



It can be even more dynamic!

We have systolic blood pressure data in the ‘BPXSY1’ column and the level of education in the ‘DMDEDUC2’ column.

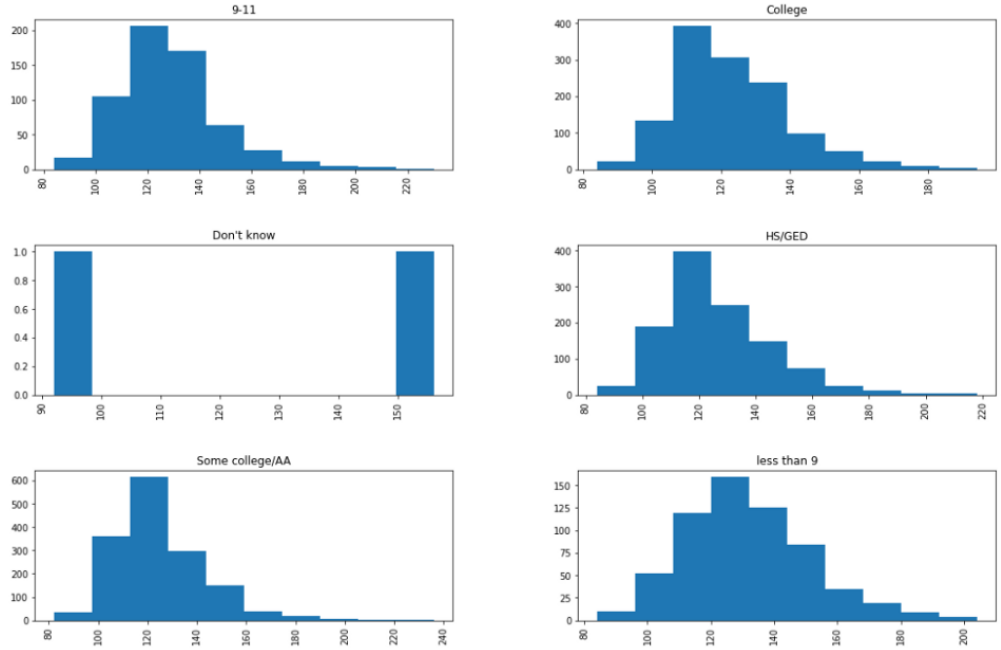
Examine the distribution of the systolic blood pressure in the population of each education level

Replace the numeric value of the ‘DMDEDUC2’ column with more meaningful string values:

```
df["DMDEDUC2x"] = df.DMDEDUC2.replace({1: "less than 9", 2: "9-11", 3: "HS/GED", 4: "Some college/AA", 5: "College", 7: "Refused", 9: "Don't know"})
```

Make the histograms now,

```
df[['DMDEDUC2x', 'BPXSY1']].hist(by='DMDEDUC2x', figsize=(18, 12))
```



Now we have the distribution of systolic blood pressure levels for each education level

Bar Plot

How the systolic blood pressure changes with marital status.

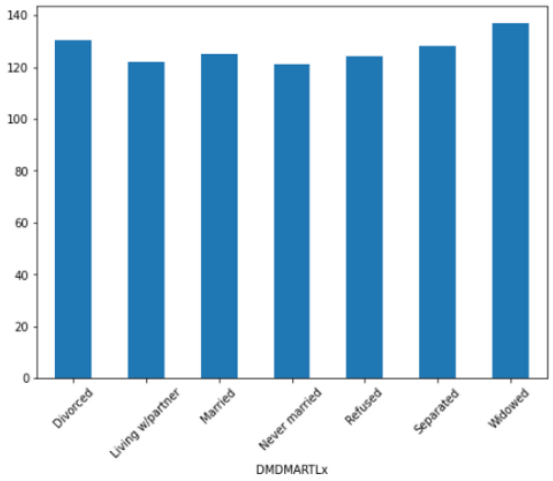
Replace the numeric values of the ‘DMDMARTL’ column with more meaningful strings.

```
df["DMDMARTLx"] = df.DMDMARTL.replace({1: "Married", 2: "Widowed", 3: "Divorced", 4: "Separated", 5: "Never married", 6: "Living w/partner", 77: "Refused"})
```

To make the bar plot we need to preprocess the data.

Group the data by different marital statuses and take the mean of each group and plot:

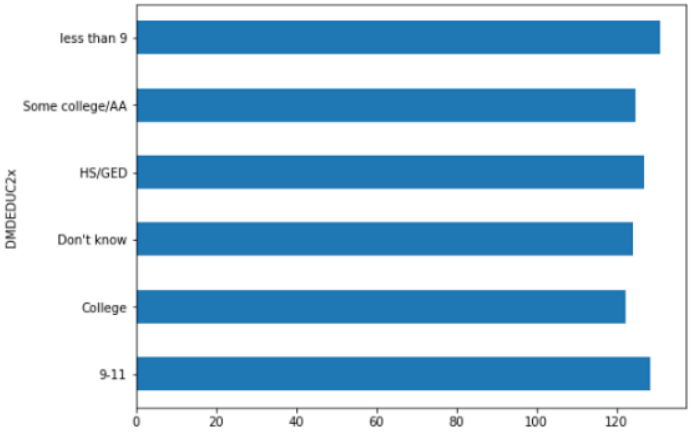
```
df.groupby('DMDMARTLx')['BPXSY1'].mean().plot(kind='bar', rot=45, fontsize=10, figsize=(8, 6))
```



Used the ‘rot’ parameter to rotate the x ticks 45 degrees. Otherwise, they will be too cluttered.

Make it horizontal as well,

```
df.groupby('DMDEDUC2x')['BPXSY1'].mean().plot(kind='barh', rot=45, fontsize=10, figsize=(8, 6))
```



Bar plot with multiple variables.

We have a column that contains the ethnic origin of the population. It will be interesting to see if people’s weight, height, and body mass index change with ethnic origin.

Group those three columns (weight, height, and body mass index) by ethnic origin and take the mean.

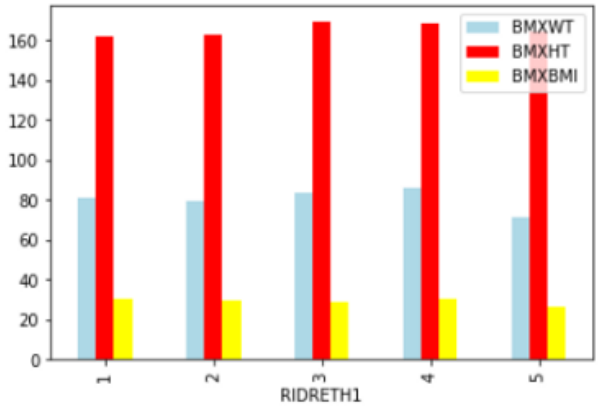
```
df_bmx = df.groupby('RIDRETH1')['BMXWT', 'BMXHT', 'BMXBMI'].mean().reset_index()
```

	RIDRETH1	BMXWT	BMXHT	BMXBMI
0	1	81.400300	161.916650	30.965900
1	2	79.388338	162.695442	29.928150
2	3	83.703412	169.103730	29.199559
3	4	86.330687	168.457415	30.399751
4	5	71.365436	164.598322	26.152125

This time I did not change the ethnic origin data. I kept the numeric values as it is.

Make the bar plot now,

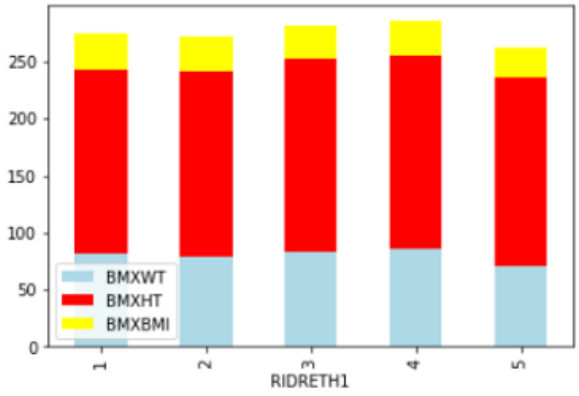
```
df_bmx.plot(x = 'RIDRETH1',
        y=['BMXWT', 'BMXHT', 'BMXBMI'],
        kind = 'bar',
        color = ['lightblue', 'red', 'yellow'],
        fontsize=10)
```



See that ethnic group 4 is a little higher than the rest of them. But they are all very close. No significant difference.

We can stack different parameters (weight, height, and body mass index) on top of each other as well.

```
df_bmx.plot(x = 'RIDRETH1',
        y=['BMXWT', 'BMXHT', 'BMXBMI'],
        kind = 'bar', stacked=True,
        color = ['lightblue', 'red', 'yellow'],
        fontsize=10)
```



Pie Plot

Check if marital status and education have any relation.

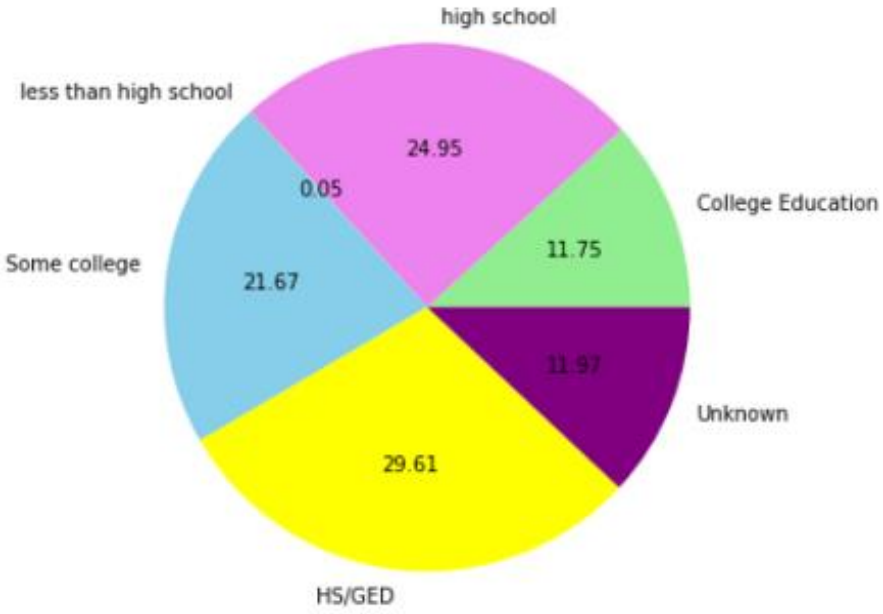
Group the marital status by education level and count the population in each marital status group by education level.

```
df_edu_marit = df.groupby('DMDEDUC2x')['DMDMARTL'].count()
pd.Series(df_edu_marit)
```

```
DMDEDUC2x
9-11                643
College            1366
Don't know            3
HS/GED             1186
Some college/AA    1621
less than 9         655
```

Using this Series make a pie plot very easily:

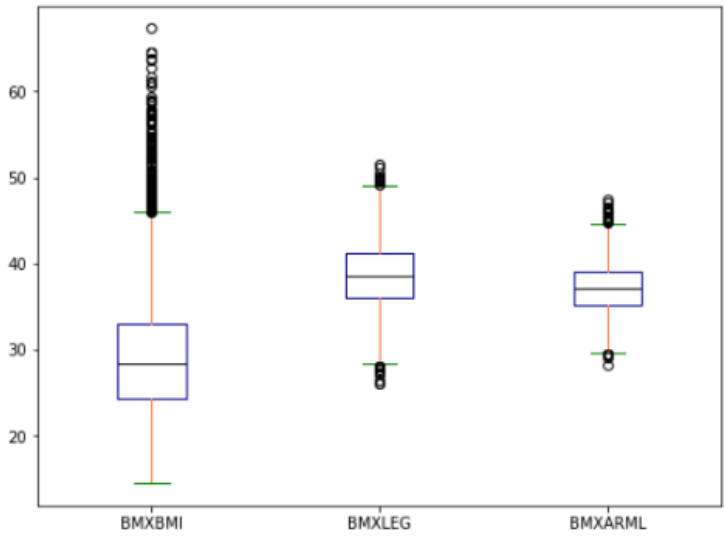
```
ax = pd.Series(df_edu_marit).plot.pie(subplots=True, label='',
    labels = ['College Education', 'high school',
    'less than high school', 'Some college',
    'HS/GED', 'Unknown'],
    figsize = (8, 6),
    colors = ['lightgreen', 'violet', 'coral', 'skyblue', 'yellow', 'purple'], autopct = '%.2f')
```



Boxplot

Make a box plot using body mass index, leg, and arm length data.

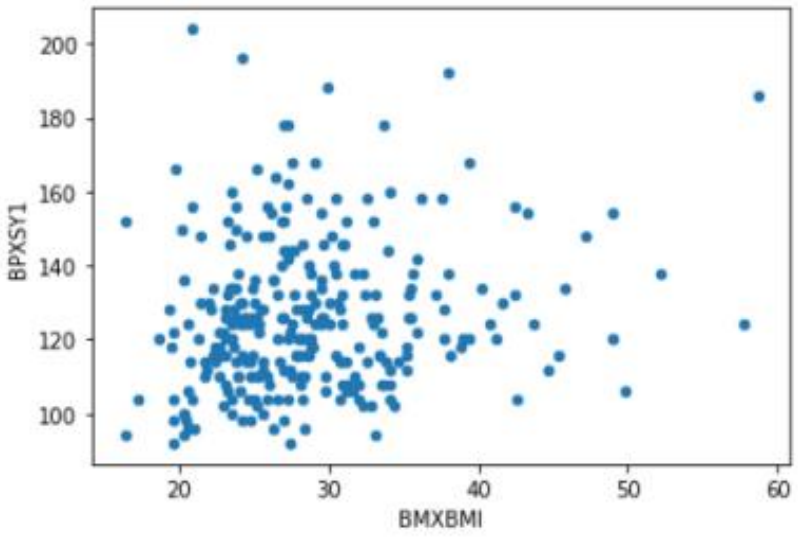
```
color = {'boxes': 'DarkBlue', 'whiskers': 'coral',
        'medians': 'Black', 'caps': 'Green'}
df[['BMXBMI', 'BMXLEG', 'BMXARML']].plot.box(figsize=(8, 6),color=color)
```



Scatter Plot

Simple scatter plot to see if there is any relationship between body mass index(‘BMXBMI’) and systolic blood pressure(‘BPXSY1’).

```
df.head(300).plot(x='BMXBMI', y= 'BPXSY1', kind = 'scatter')
```



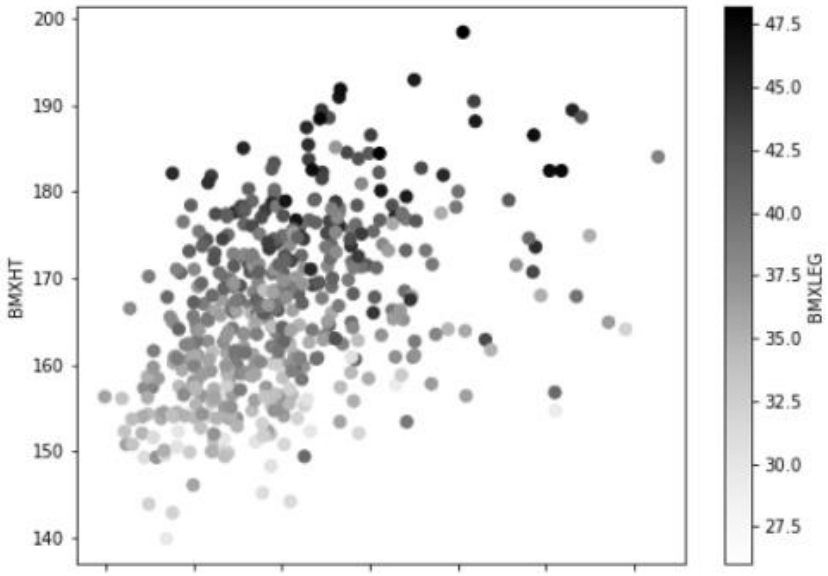
Used only 300 data because if I use all the data, the scatter plot becomes too dense to understand.

Now, check a little advanced scatter plot

add some color shades. male a scatter plot, putting weight in the x-axis and height in the y-axis.

Also add the length of the leg. But the length of the legs will show in shades. If the length of the leg is longer the shade will be darker else the shade will be lighter.

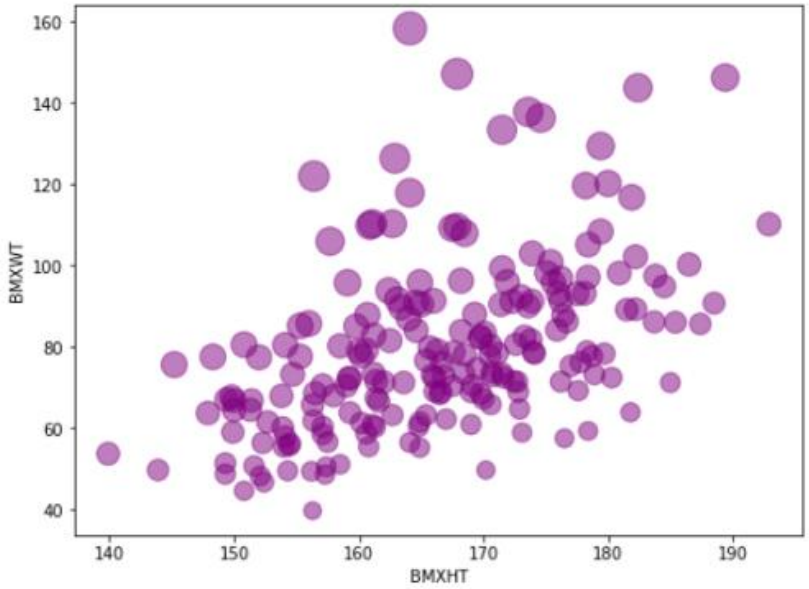
```
df.head(500).plot.scatter(x= 'BMXWT', y = 'BMXHT', c ='BMXLEG', s=50, figsize=(8, 6))
```



It shows the relationship between weight and height.

Another way of adding a third parameter like that is to add size in the particles. Here, I am putting the height in the x-axis, weight in the y-axis, and body mass index as an indicator of the bubble size.

```
df.head(200).plot.scatter(x= 'BMXHT', y = 'BMXWT',
                          s =df['BMXBMI'][:200] * 7,
                          alpha=0.5, color='purple',
                          figsize=(8, 6))
```



Here the smaller dots means lower BMI and the bigger dots mean the higher BMI.

Hexbin
Another beautiful type of visualization where the dots are hexagonal.

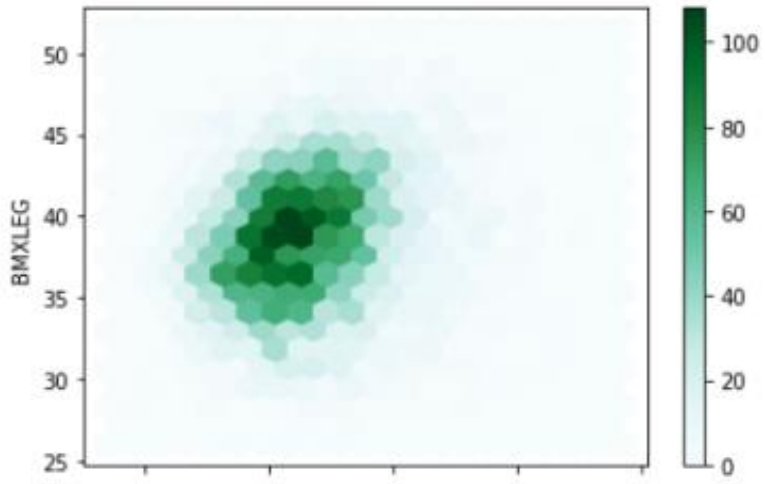
When the data is too dense, it is useful to put them in bins.

As you can see, in the previous two plots I used only 500 and 200 data because if I put all the data in the dataset the plot becomes too dense to understand or daw any information from it.

In this case, using spatial distribution can be very useful.

Using hexbin where data will be represented in hexagons. Each hexagon is a bin representing the density of that bin. Example of basic hexbin.

```
df.plot.hexbin(x='BMXARMC', y='BMXLEG', gridsize= 20)
```



Here the darker color represents the higher density of data and the lighter color represents the lower density of data.

Sound like a histogram (Instead of bars, it is represented by colors).

If we add an extra parameter 'C', the distribution changes. It will not be like a histogram anymore.

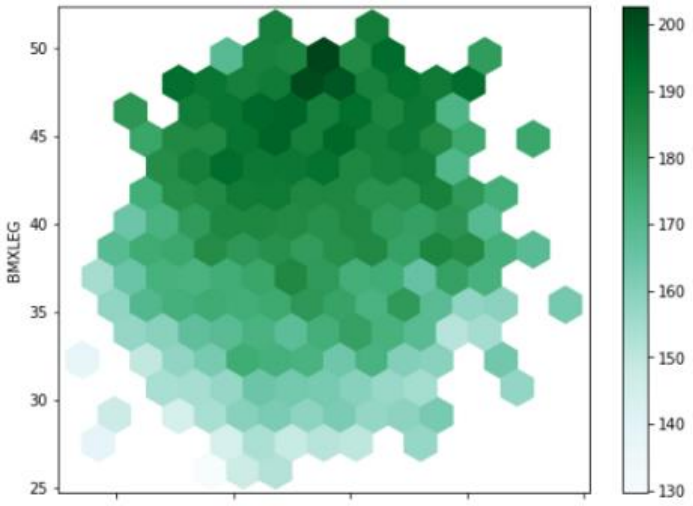
The parameter 'C' specifies the position of each (x, y) coordinate, accumulates for each hexagonal bins, and then reduced by using *reduce_C_function.*

If the reduce_C_ function is not specified, by default it uses np.mean.

You can specify it the way you want as np.mean, np.max, np.sum, np.std, etc.

Example:

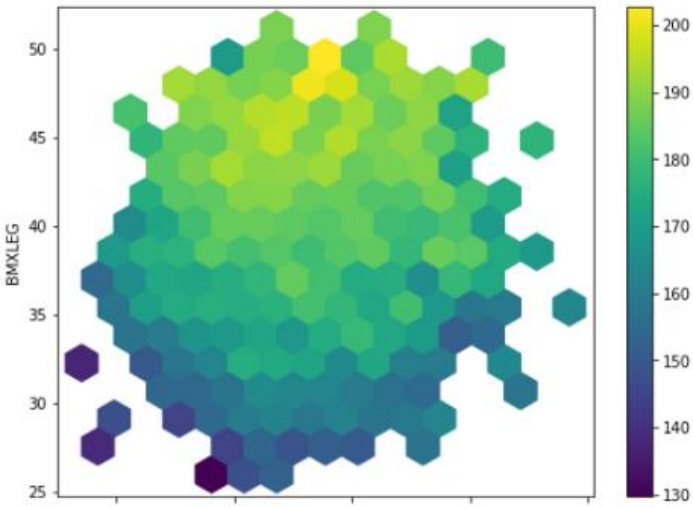
```
df.plot.hexbin(x='BMXARMC', y='BMXLEG', C = 'BMXHT',
                reduce_C_function=np.max,
                gridsize=15,
               figsize=(8,6))
```



Here the darker color of a hexagon means, np.max has a higher value for the population height('BMXHT') for that bin as you can see that I used np.max as a reduce_C_function.

Use a colormap instead of shades of color:

```
df.plot.hexbin(x='BMXARMC', y='BMXLEG', C = 'BMXHT',
                reduce_C_function=np.max,
                gridsize=15,
               figsize=(8,6),
               cmap = 'viridis')
```



Some Advanced Visualization

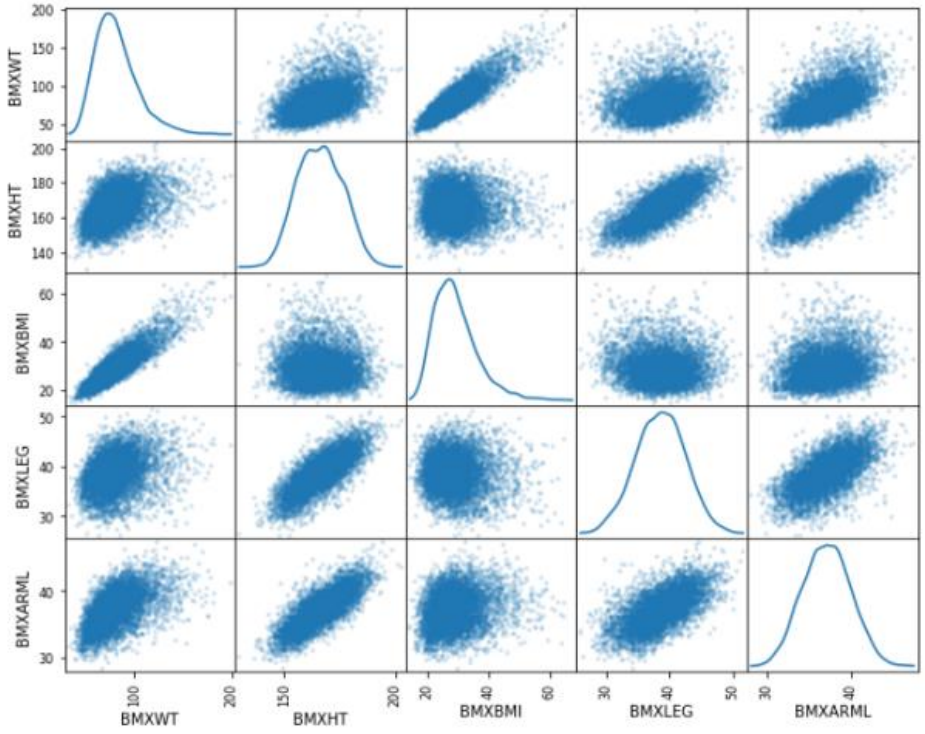
Scatter_matrix

Provides a huge amount of information packed in one plot.

It can be used for general data analysis or feature engineering on machine learning.

Example:

```
from pandas.plotting import scatter_matrixscatter_matrix(df[['BMXWT', 'BMXHT', 'BMXBMI', 'BMXLEG', 'BMXARML']], alpha = 0.2, figsize=(10, 8), diagonal = 'kde')
```

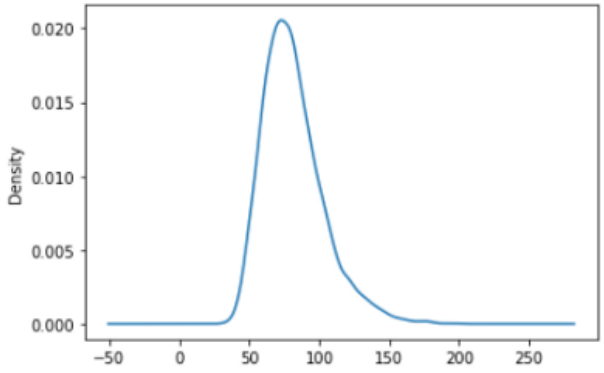


Five features here. We got the relationship between all five variables with each other. In the diagonals, it gives you the density plot of each individual feature.

KDE or density plots

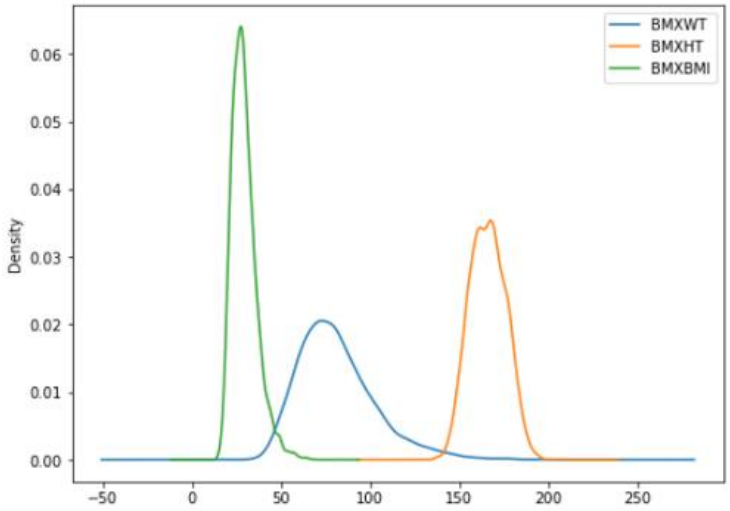
KDE plots or Kernel Density Plots are built to provide the probability distribution of a series or a column in a DataFrame. Let's see the probability distribution of the weight variable ('BMXWT').

```
df['BMXWT'].plot.kde()
```



Probability distribution of height, weight, and BMI in the same plot:

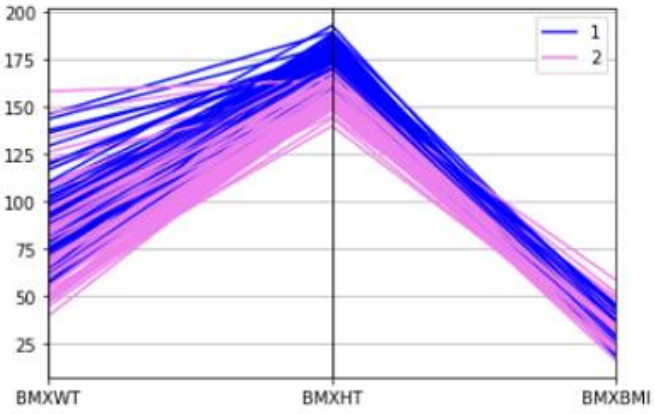
```
df[['BMXWT', 'BMXHT', 'BMXBMI']].plot.kde(figsize = (8, 6))
```



Parallel_coordinates
This is a good way of showing multi-dimensional data.

It clearly shows the clusters if there is any.

For example, see if there is any difference in height, weight, and BMI between men and women.
from pandas.plotting import parallel_coordinatesparallel_coordinates(df[['BMXWT', 'BMXHT', 'BMXBMI', 'RIAGENDR']].dropna().head(200), 'RIAGENDR', color=['blue', 'violet'])



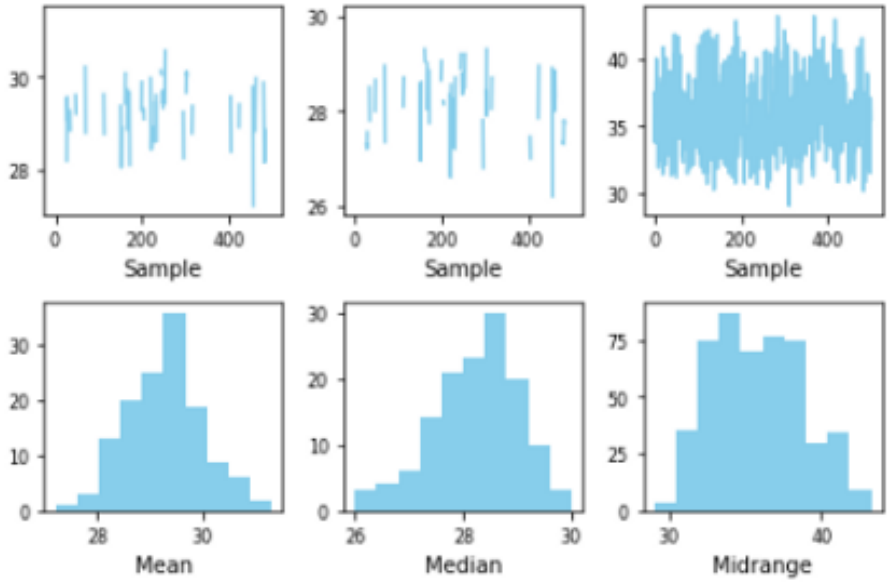
We can see the clear difference in body weight, height, and BMI between men and women. Here, 1 is men and 2 is women.

Bootstrap_plot
Used to assess the uncertainty of a given dataset. This is an extremely important process and a time saver for statisticians and researchers.

This function takes a random sample of a specified size. Then mean, median, and midrange is calculated for that sample. This process is repeated a specified number of times.

Bootstrap plot using the BMI data:

```
from pandas.plotting import bootstrap_plotbootstrap_plot(df['BMXBMI'], size=100, samples=1000, color='skyblue')
```



Here, the sample size is 100 and the number of samples is 1000. So, it took a random sample of 100 data to calculate mean, median, and midrange. The process is repeated 1000 times.