



Matplotlib and Seaborn



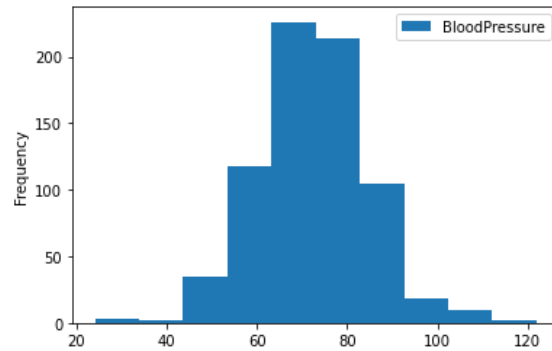
Library for data analysis

- **NumPy** (Numerical Python)
 - Large multidimensional array operations
- **SciPy** (Scientific Python)
 - Many efficient numerical routines such as routines for numerical integration and optimization
- **Pandas**
 - Data manipulation and data visualization
- **Matplotlib**
 - Data exploration and data visualization
- **Seaborn**
 - High-level data visualization library based on Matplotlib
- **Scikit-learn**
 - Machine learning and statistical modeling

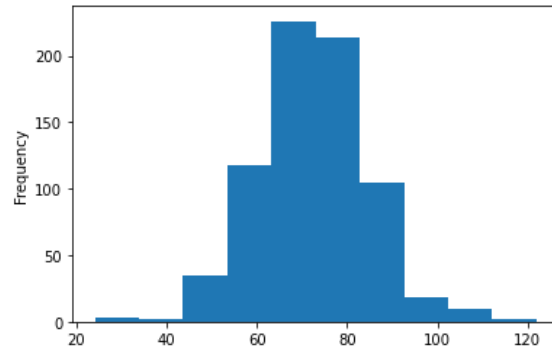
Different plotting methods

Pandas

```
diabetes_df.plot(kind = "hist", y = "BloodPressure");
```



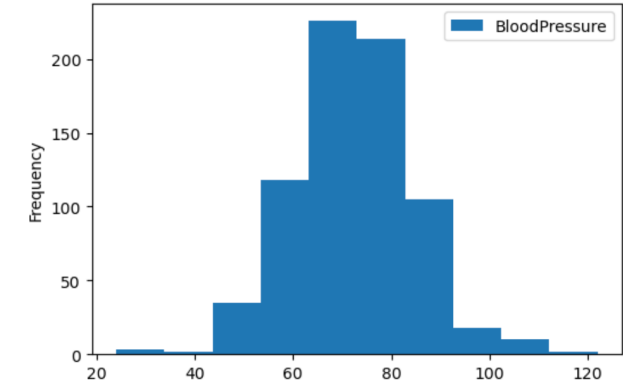
```
plt.hist(diabetes_df.BloodPressure)  
plt.ylabel('Frequency')  
plt.show()
```



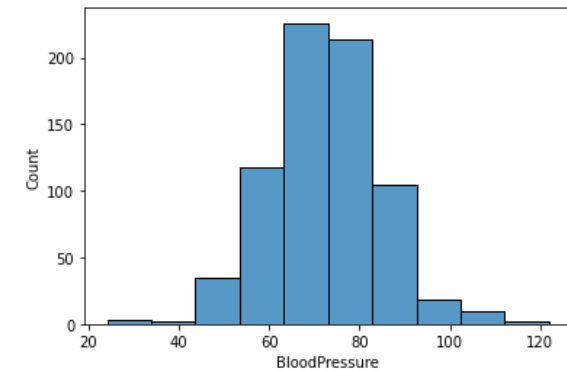
Matplotlib

Matplotlib (Create a container) Pandas (plot)

```
fig, ax = plt.subplots(figsize = (6,4))  
diabetes_df.plot(kind = "hist", y = "BloodPressure", ax = ax);
```



```
sns.histplot(data = diabetes_df, x = "BloodPressure", bins = 10);
```

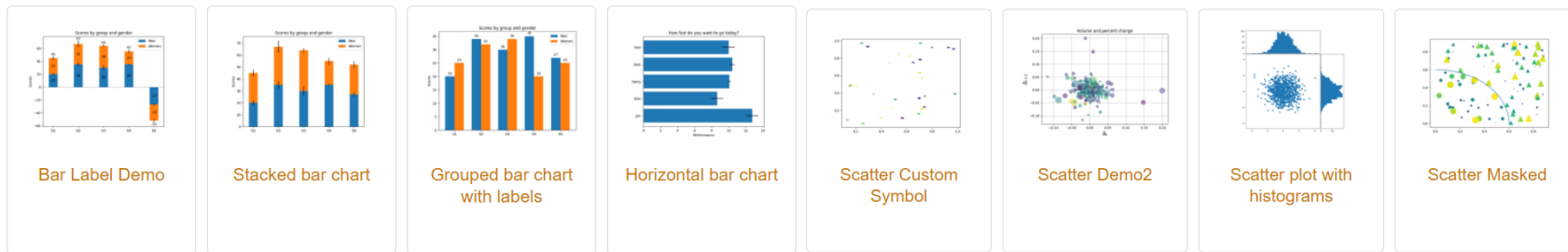


Outline

- Matplotlib
 - Single plot
 - Multiple plot
 - Secondary y-axis
- Seaborn
 - X-axis with categorical data
 - (Lineplot), countplot, barplot, heatmap
 - Numerical data
 - Histogram, scatterplot
 - Multiple plots
 - Jointplot, pairplot, FacetGrid

Matplotlib

- **matplotlib**
 - Matplotlib is a **low-level** data visualization library in python.

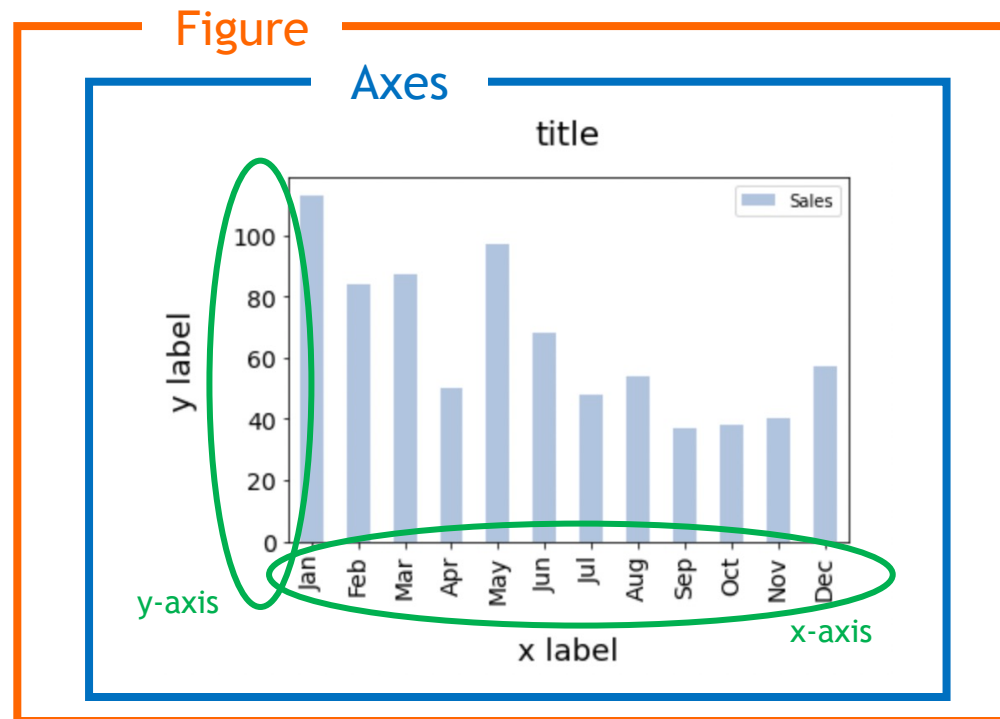


- **matplotlib.pyplot**
 - Pyplot is a subset of matplotlib, which is a collection of the most commonly used plotting functions.

```
import matplotlib.pyplot as plt
```

Matplotlib - figure and axes

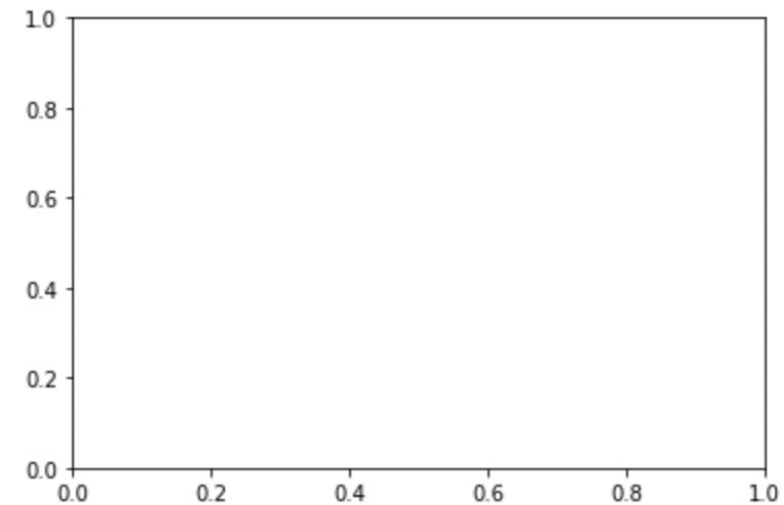
- **Figure** object: The outermost container for matplotlib plots, which can contain multiple Axes objects.
- **Axes** object: The axes is the area your plot appears in. (In matplotlib, axes is not the plural form of axis)



Matplotlib - single plot (1/3)

- Step1: Create figure and axes
- Step2: Plot a chart in axes
- Step3: Format the style

```
#step1: Create a figure and axes  
fig, ax = plt.subplots()
```



Matplotlib - single plot (2/3)

- Step1: Create figure and axes
- Step2: Plot a chart in axes
- Step3: Format the style

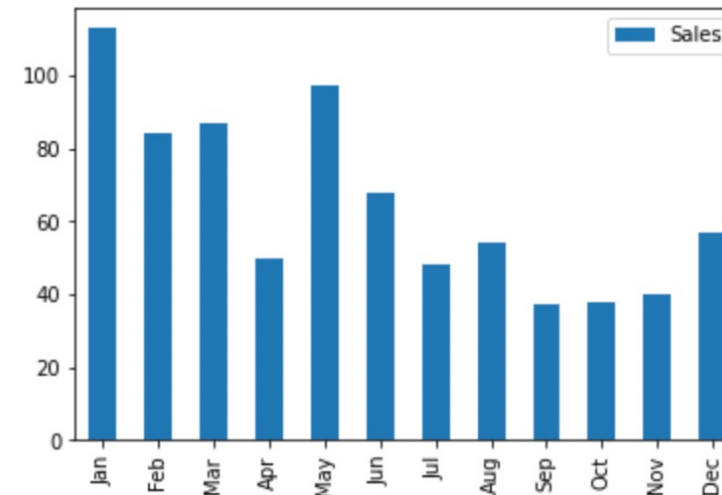
	Sales	Cumulative_sales
Jan	113	113
Feb	84	197
Mar	87	284
Apr	50	334
May	97	431
Jun	68	499
Jul	48	547
Aug	54	601
Sep	37	638
Oct	38	676
Nov	40	716
Dec	57	773

#step1: Create a figure and axes

```
fig, ax = plt.subplots()
```

#step2: Plot a chart in axes

```
sales_df.plot(kind = "bar", y = "Sales", ax = ax);
```



Matplotlib - single plot (3/3)

- Step1: Create figure and axes
- Step2: Plot a chart in axes
- Step3: Format the style

```
#step1: Create a figure and axes
fig, ax = plt.subplots()

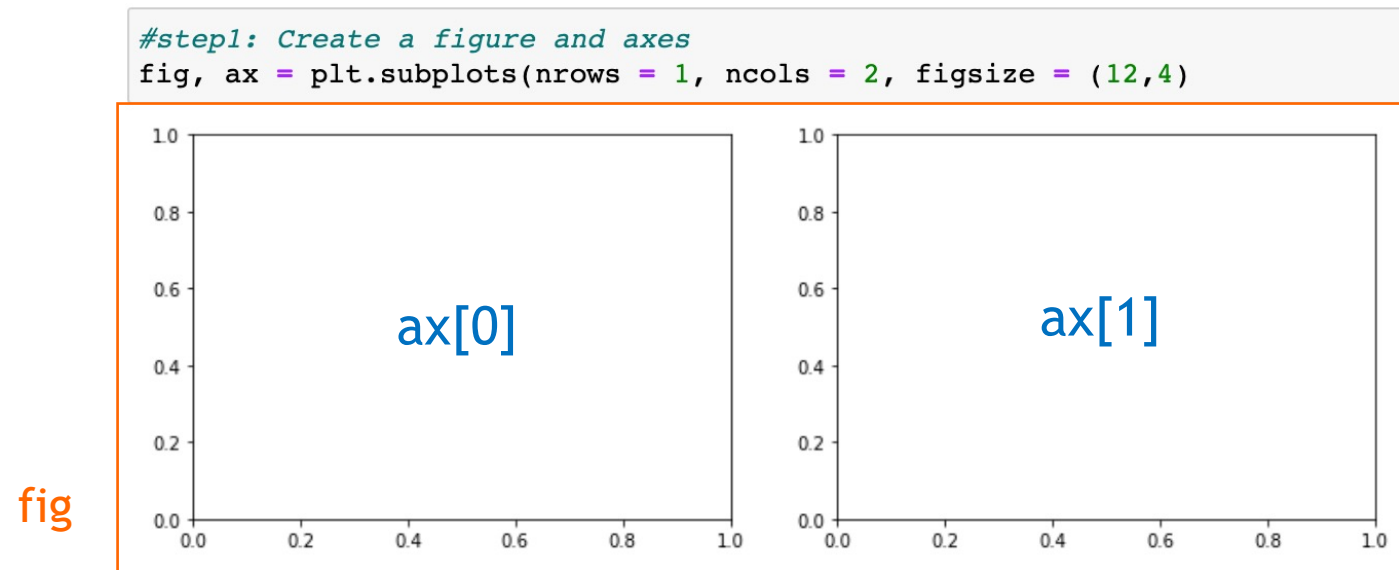
#step2: Plot a chart in axes
sales_df.plot(kind = "bar", y = "Sales", ax = ax)

#step3: Format the style
ax.set_title("Monthly sales report", fontsize=18)
ax.set_xlabel("Month", fontsize=12)
ax.set_ylabel("Sales", fontsize=12);
```



Matplotlib - multiple plots

- Pass two arguments to determine the number of subplots.
 - `subplot(1,2)` → 2 subplots
 - `subplot(2,3)` → 6 subplots



`subplot(1,2)`

ax[0]	ax[1]
-------	-------

ax: 1 x 2 array

`subplot(2,3)`

ax[0,0]	ax[0,1]	ax[0,2]
ax[1,0]	ax[1,1]	ax[1,2]

ax: 2 x 3 array

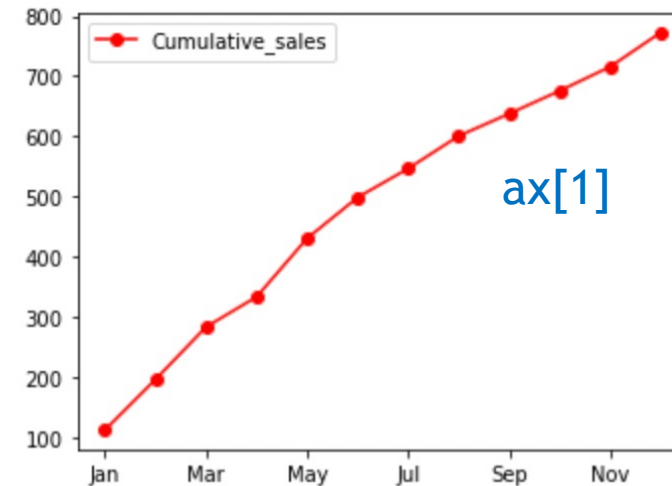
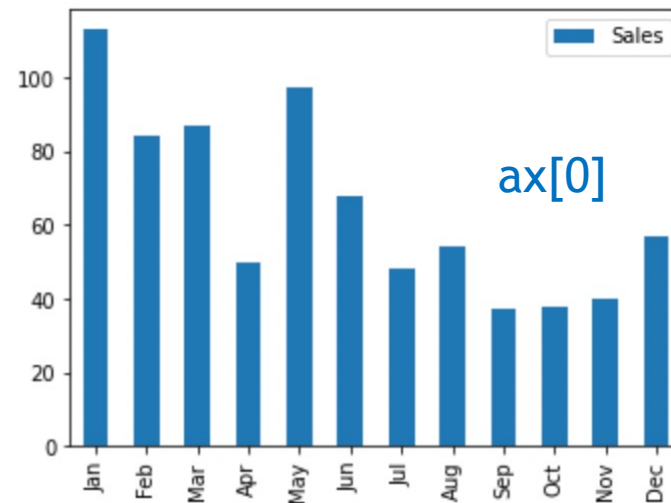
Matplotlib - multiple plots

- Assign axes by the argument `ax`.

	Sales	Cumulative_sales
Jan	113	113
Feb	84	197
Mar	87	284
Apr	50	334
May	97	431
Jun	68	499
Jul	48	547
Aug	54	601
Sep	37	638
Oct	38	676
Nov	40	716
Dec	57	773

```
#step1: Create a figure and axes
fig, ax = plt.subplots(nrows = 1, ncols = 2, figsize = (12,4))
#step2: Plot a chart in axes
sales_df.plot(kind = "bar", y = "Sales", ax=ax[0])
sales_df.plot(kind = "line", y = "Cumulative_sales", ax=ax[1], color = "red", marker = "o")
```

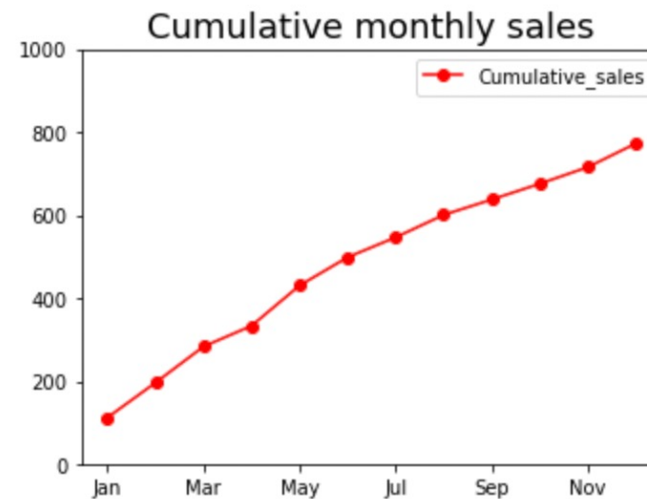
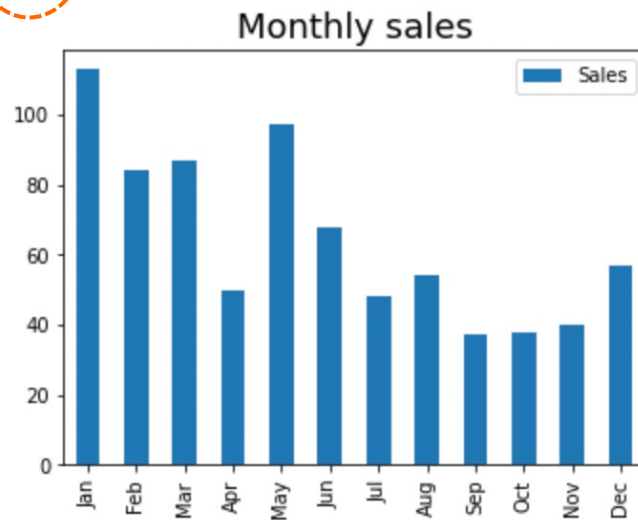
<AxesSubplot:>



Matplotlib - multiple plots

- Plot charts in axes and format the charts

```
#step1: Create a figure and axes
fig, ax = plt.subplots(nrows = 1, ncols = 2, figsize = (12,4))
#step2: Plot a chart in axes
sales_df.plot(kind = "bar", y = "Sales", ax=ax[0])
sales_df.plot(kind = "line", y = "Cumulative_sales", ax=ax[1], color = "red", marker = "o")
#step3: Format the style
ax[0].set_title("Monthly sales", fontsize = 18)
ax[1].set_title("Cumulative monthly sales", fontsize = 18)
ax[1].set_ylim([0,1000]);
```



Exercise

(A.1) Given the synthetic dataset above, each column represents the quarterly sales of products A, B, and C. Create a figure with three bar charts to show the sales data of each product.

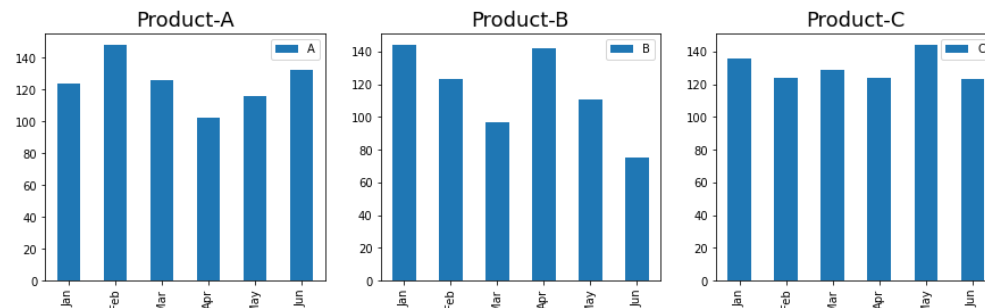
Setting: `figsize = (15,4)`

#step1: Create a figure and axes

#step2: Plot a chart in axes

#step3: Format the style

	A	B	C
Jan	124	144	136
Feb	148	123	124
Mar	126	97	129
Apr	102	142	124
May	116	111	144
Jun	132	75	123

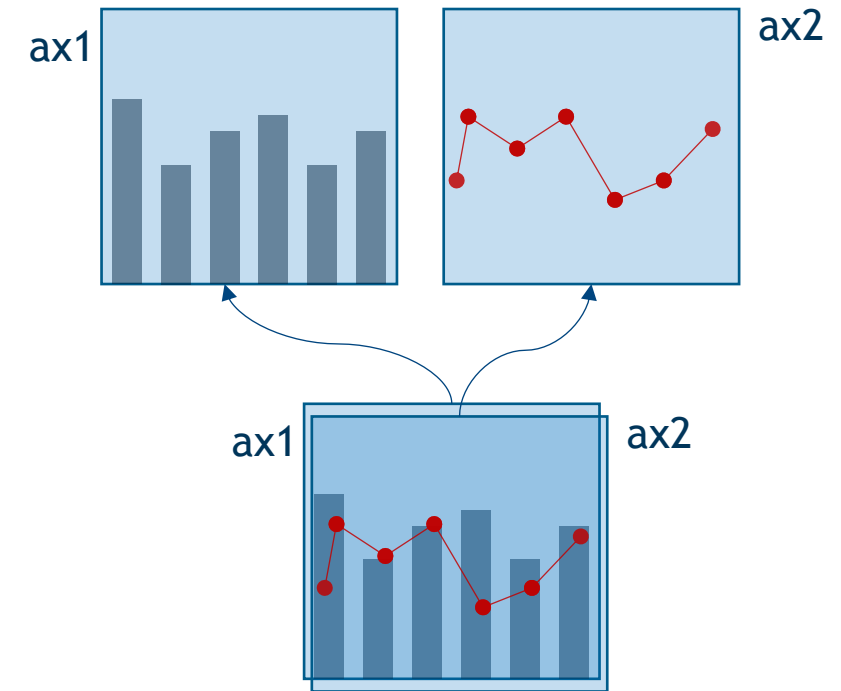
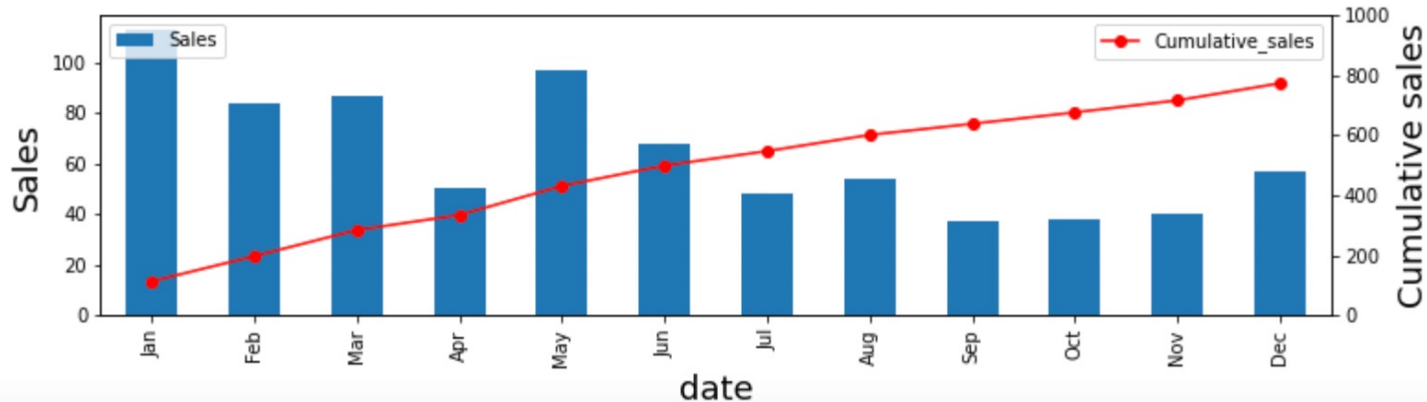


ax[0]	ax[1]	ax[2]
-------	-------	-------

Matplotlib - Single plot with secondary y-axis



```
#step1: Create a figure and axes
fig, ax1 = plt.subplots(figsize=(12,3))
ax2 = ax1.twinx()
#step2: Plot a chart in axes
sales_df.plot(kind = "bar", y = "Sales", ax=ax1)
sales_df.plot(kind = "line", y = "Cumulative_sales", ax=ax2, color = "red", marker = "o")
#step3: Format the style
ax1.set_xlabel("date", fontsize = 18)
ax1.set_ylabel("Sales", fontsize = 18)
ax1.legend(loc="upper left")
ax2.set_ylabel("Cumulative sales", fontsize = 18)
ax2.set_ylim([0,1000])
ax2.legend(loc="upper right");
```

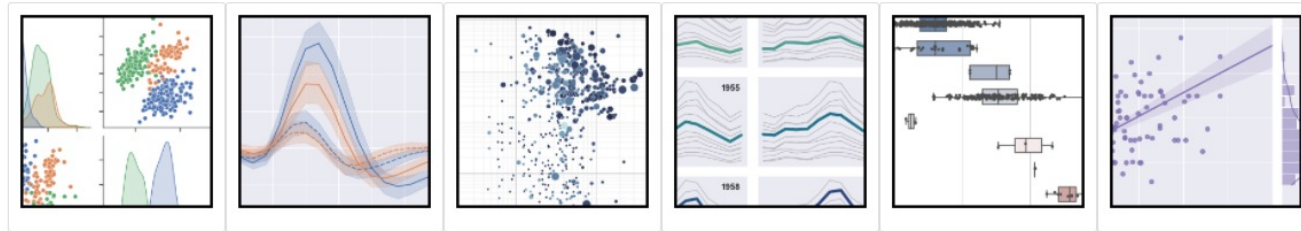


Outline

- Matplotlib
 - Single plot
 - Multiple plot
 - Secondary y-axis
- Seaborn
 - X-axis with categorical data
 - (Lineplot), countplot, barplot, heatmap
 - Numerical data
 - Histogram, scatterplot
 - Multiple plots
 - Jointplot, pairplot, FacetGrid

Seaborn

- Seaborn is a package that provides many types of insightful plots. You can write simple code to visualize complex graphics.



- Data structures accepted by seaborn
 - Objects from pandas or numpy
 - Long-form and wide-form
- Installation

```
!pip install seaborn
```

```
import seaborn as sns
```


Lineplot

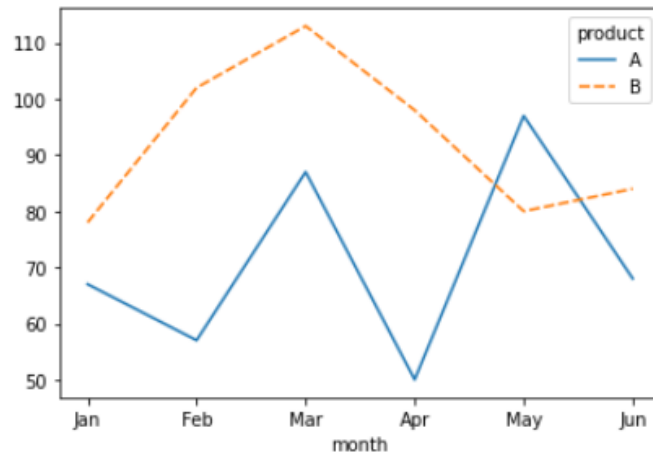
- Use the function `lineplot()` to draw a line chart.
- Use `hue` to specify which categorical column should be used to define the subsets.

Wide-form

```
sns.lineplot(data = product_df_wide)
```

```
<AxesSubplot:xlabel='month'>
```

	A	B
Jan	67	78
Feb	57	102
Mar	87	113
Apr	50	98
May	97	80
Jun	68	84

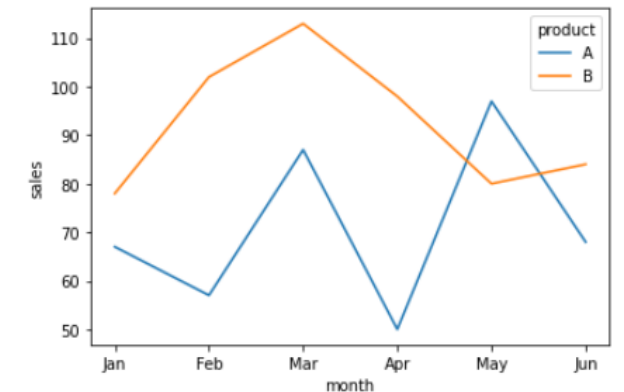


Long-form

```
sns.lineplot(data = product_df_long,  
             x = "month",  
             y = "sales",  
             hue = "product")
```

```
<AxesSubplot:xlabel='month', ylabel='sales'>
```

	month	product	sales
0	Jan	A	67
1	Feb	A	57
2	Mar	A	87
3	Apr	A	50
4	May	A	97
5	Jun	A	68
6	Jan	B	78
7	Feb	B	102
8	Mar	B	113
9	Apr	B	98
10	May	B	80
11	Jun	B	84



No need to reshape the dataframe

Countplot

- Use `countplot()` to show the counts of observations in each bar.
- Pass arguments: `data`, `x`

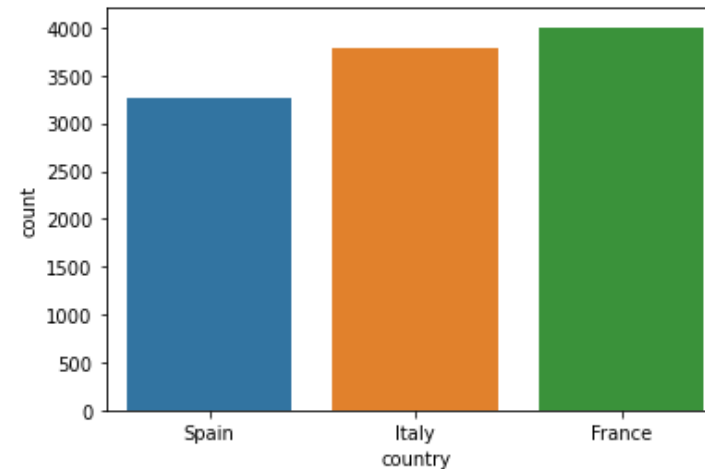
	country	price	variety
17	Spain	80.0	Tempranillo
39	Italy	29.0	Red Blend
43	Italy	39.0	Red Blend
45	Italy	30.0	Red Blend
51	France	68.0	Chardonnay
...
150908	France	65.0	Pinot Noir
150909	France	52.0	Pinot Noir
150910	France	38.0	Pinot Noir
150911	France	37.0	Pinot Noir
150912	France	65.0	Pinot Noir

11045 rows × 3 columns

Long-form

```
sns.countplot(data = wine_df, x = "country")
```

```
<AxesSubplot:xlabel='country', ylabel='count'>
```



The bar height is the number of wines of each country.

No need to group data in advance

```
wine_df.groupby("country").size()
```

```
country
France    4004
Italy     3783
Spain     3258
dtype: int64
```

Understand the data behind the chart

Target: Country



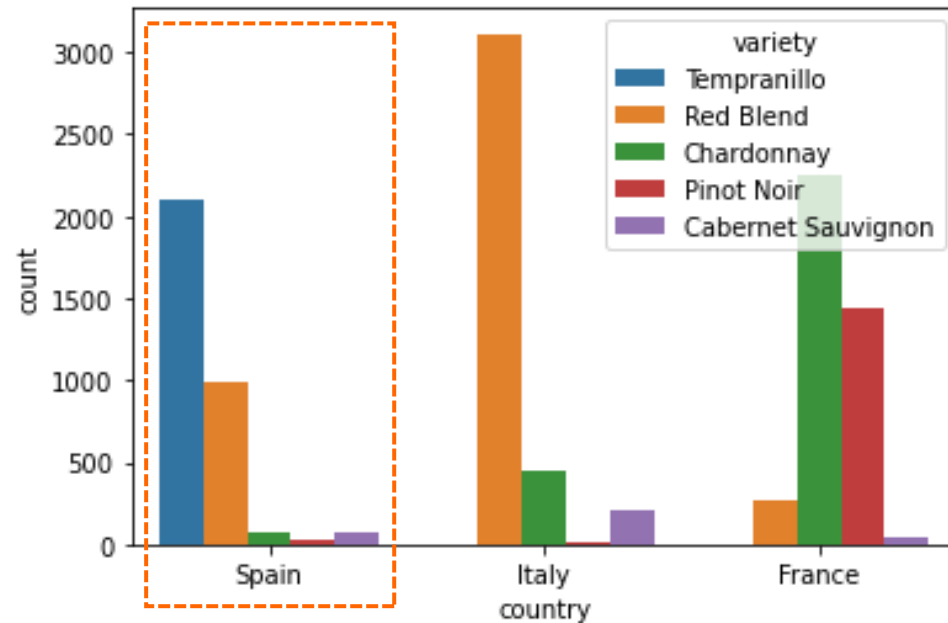
<https://seaborn.pydata.org/generated/seaborn.countplot.html>

Countplot

- Use the parameter `hue` to divide each bar into sub-bars.

```
sns.countplot(data = wine_df, x = "country", hue = "variety")
```

```
<AxesSubplot:xlabel='country', ylabel='count'>
```



```
pd.crosstab(wine_df.country, wine_df.variety)
```

variety	Cabernet Sauvignon	Chardonnay	Pinot Noir	Red Blend	Tempranillo
country					
France	44	2247	1446	267	0
Italy	214	450	7	3111	1
Spain	74	67	31	987	2099

Understand the data
behind the chart

Barplot

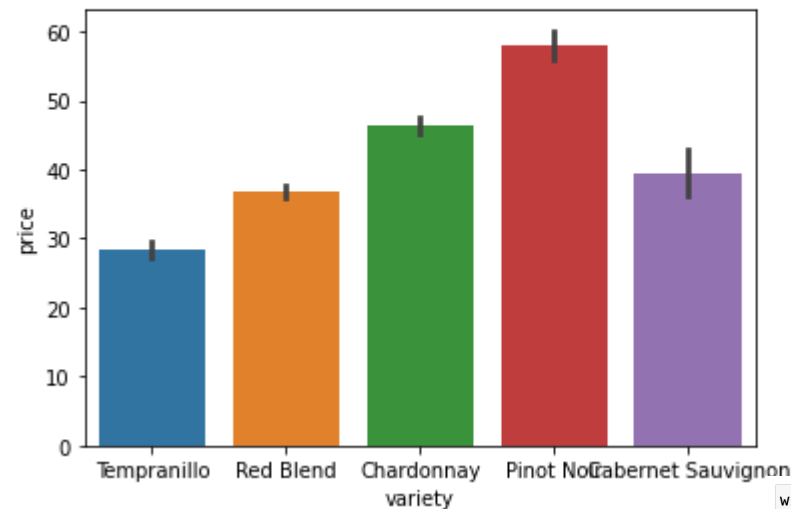
- Use `barplot()` to draw a bar chart grouped by a categorical variable.
- A bar plot shows the **mean** (or other estimator) value.

	country	price	variety
17	Spain	80.0	Tempranillo
39	Italy	29.0	Red Blend
43	Italy	39.0	Red Blend
45	Italy	30.0	Red Blend
51	France	68.0	Chardonnay
...
150908	France	65.0	Pinot Noir
150909	France	52.0	Pinot Noir
150910	France	38.0	Pinot Noir
150911	France	37.0	Pinot Noir
150912	France	65.0	Pinot Noir

11045 rows × 3 columns

```
sns.barplot(data = wine_df, x = "variety", y = "price")
```

```
<AxesSubplot:xlabel='variety', ylabel='price'>
```



The bar height is the average price of each grape variety.

```
wine_df.groupby("variety").price.mean()
variety
Cabernet Sauvignon    39.490964
Chardonnay            46.447540
Pinot Noir            58.053908
Red Blend             36.891409
Tempranillo           28.294762
Name: price, dtype: float64
```

Heatmap

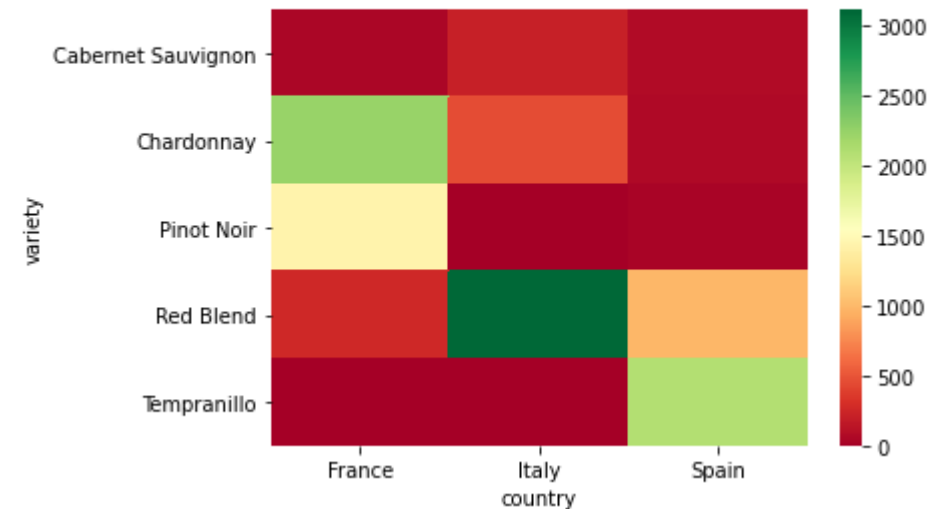
- A heat map is a two-dimensional visual representation of data, where colors represent different values.

```
# data preparation
heatmap_data = pd.crosstab(wine_df["variety"], wine_df["country"])
heatmap_data
```

	country	France	Italy	Spain
variety				
Cabernet Sauvignon		44	214	74
Chardonnay		2247	450	67
Pinot Noir		1446	7	31
Red Blend		267	3111	987
Tempranillo		0	1	2099

Without specifying parameter “values” and “aggfunc”,
crosstab() will calculate the frequency.

```
sns.heatmap(data = heatmap_data, cmap = "RdYlGn")
<AxesSubplot:xlabel='country', ylabel='variety'>
```



Heatmap

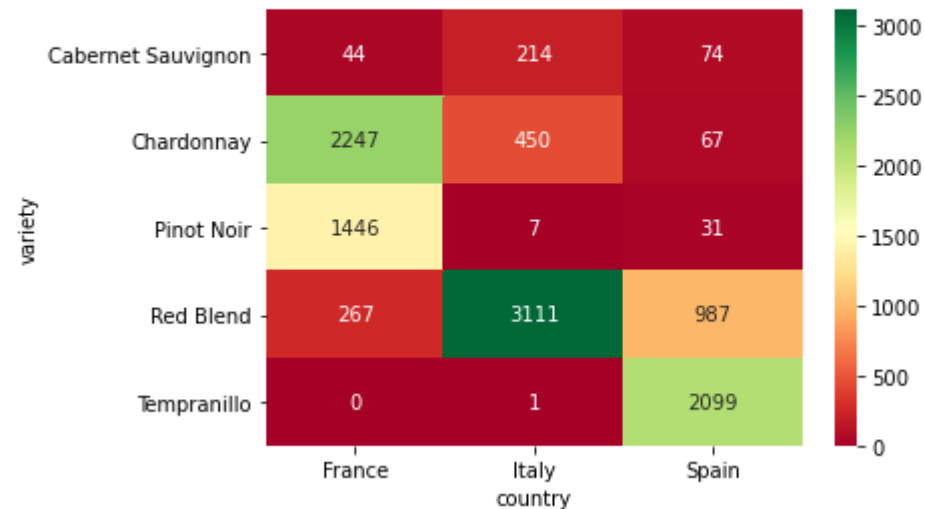
- Custom style

- `annot`: If True, show the data value in each cell.
- `fmt`: String formatting code to use when adding annotations.

```
sns.heatmap(data = heatmap_data, annot = True, fmt = "d", cmap = "RdYlGn")
```

```
<AxesSubplot:xlabel='country', ylabel='variety'>
```

country	France	Italy	Spain
variety			
Cabernet Sauvignon	44	214	74
Chardonnay	2247	450	67
Pinot Noir	1446	7	31
Red Blend	267	3111	987
Tempranillo	0	1	2099



Exercise

```
titanic_df = sns.load_dataset("titanic", dtype = {"survived": object, "pclass":object})
```

(B.1) Show the first 10 rows of the dataset.

(B.2) Use a count plot to display the number of male and female passengers.

(B.3) Use a count plot to display the number of male and female passengers, and use the column `survived` to divide each bar into two subbars.

Hint: `hue`

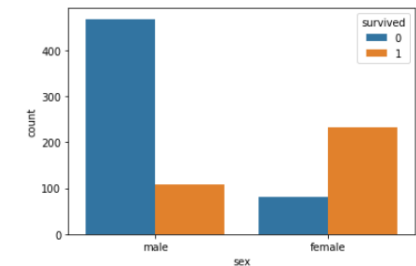
(B.4) Count the number of surviving and non-surviving passengers in each class.

Hint: Use `crosstab()` to create a contingency table based on "pclass" and "survived" columns.

(B.5) Use the result obtained in (B.4) to draw a heatmap.

Setting: `cmap = "coolwarm"`

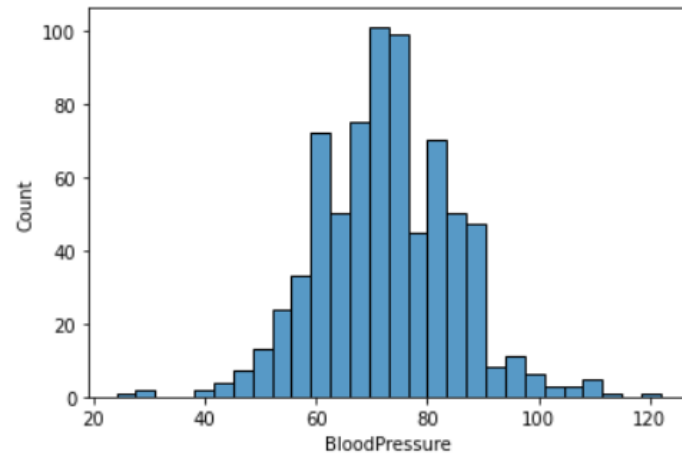
(B.3)



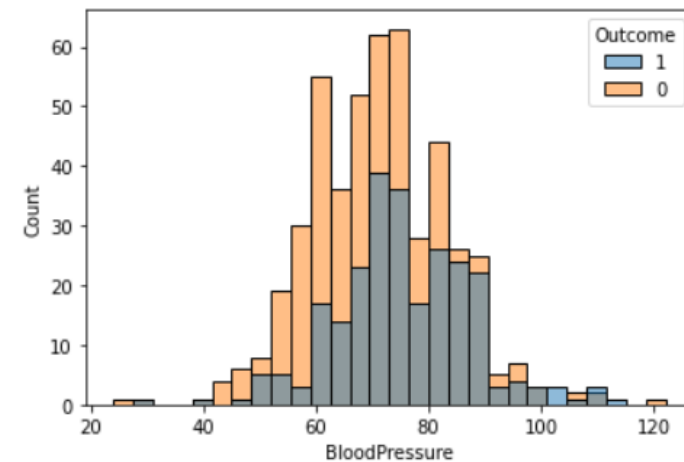
Histogram

- Use `histplot()` to get the histogram.
- Use `hue` to show the distribution of each category.

```
sns.histplot(data = diabetes_df, x = "BloodPressure")  
<AxesSubplot:xlabel='BloodPressure', ylabel='Count'>
```



```
sns.histplot(data = diabetes_df, x = "BloodPressure", hue = "Outcome")  
<AxesSubplot:xlabel='BloodPressure', ylabel='Count'>
```

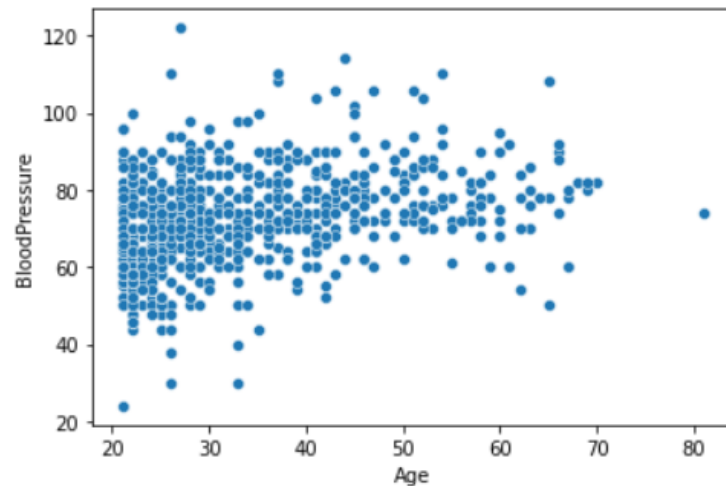


Scatter plot

- Use `scatterplot()` to show the relationship between two variables.
- Use `hue` to show the scatter plot for each category.

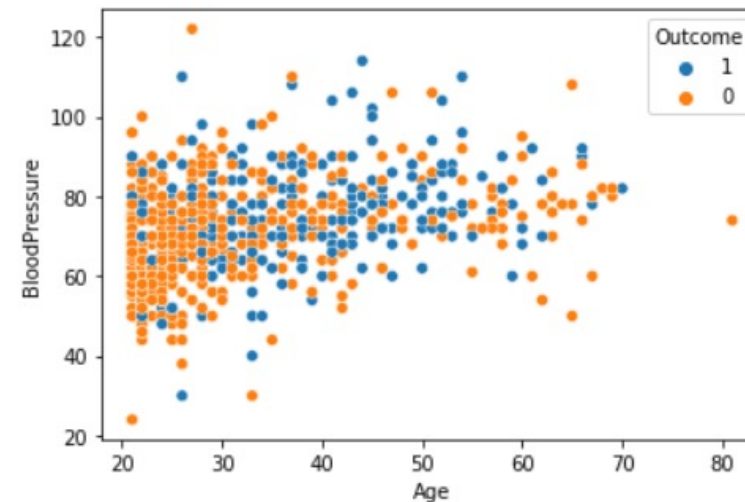
```
sns.scatterplot(data = diabetes_df, x = "Age", y = "BloodPressure")
```

```
<AxesSubplot:xlabel='Age', ylabel='BloodPressure'>
```



```
sns.scatterplot(data = diabetes_df, x = "Age", y = "BloodPressure",  
               hue = "Outcome")
```

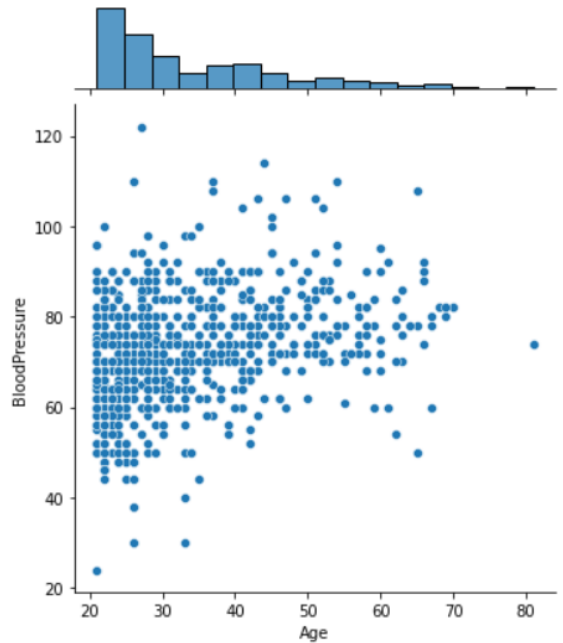
```
<AxesSubplot:xlabel='Age', ylabel='BloodPressure'>
```



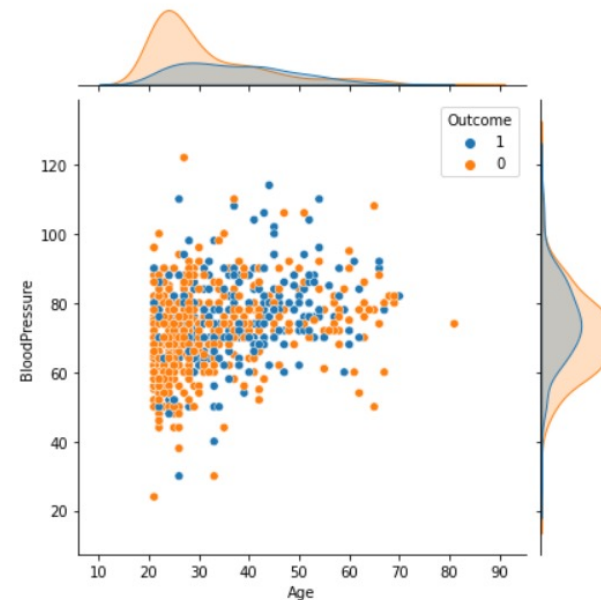
Jointplot

- Use `jointplot()` to show the relationship between two variables and their respective distributions.

```
sns.jointplot(data = diabetes_df, x = "Age", y = "BloodPressure")  
<seaborn.axisgrid.JointGrid at 0x19793e56640>
```



```
sns.jointplot(data = diabetes_df, x = "Age", y = "BloodPressure", hue = "Outcome")  
<seaborn.axisgrid.JointGrid at 0x19793df2e20>
```



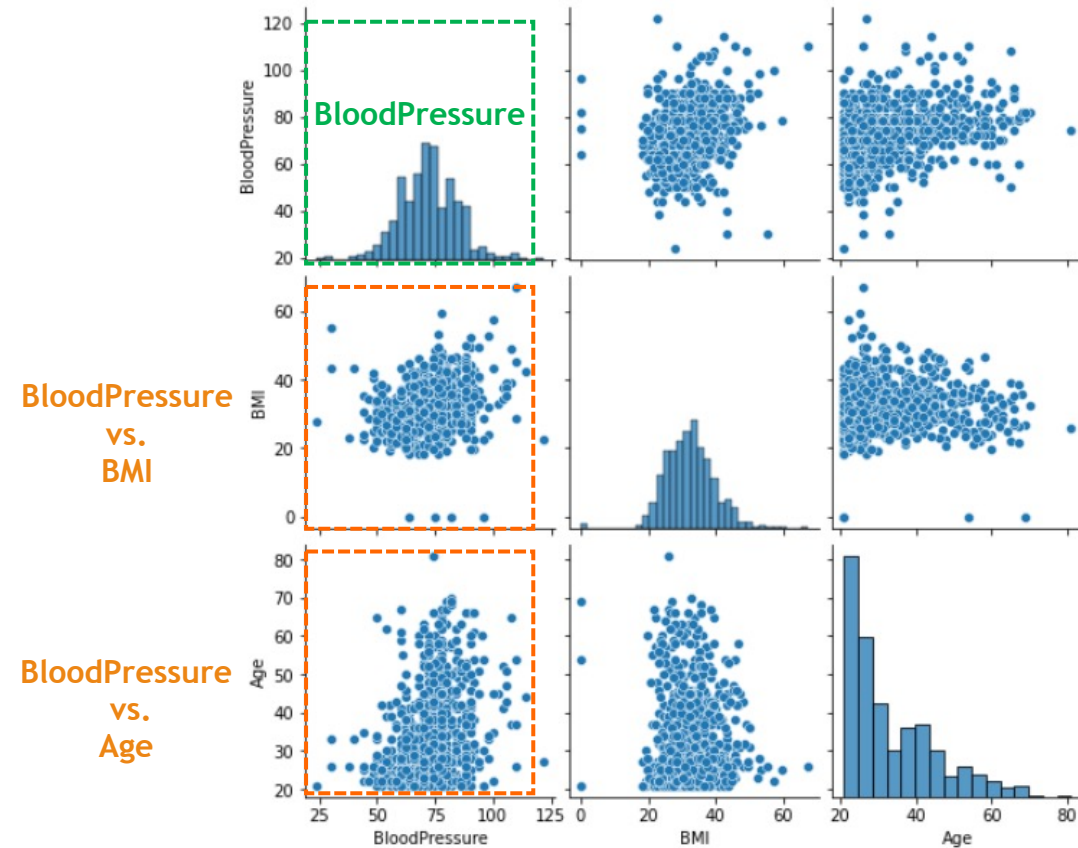
Pair plot



- Use `Pairplot()` to plot pairwise relationships in a dataset.
- The **diagonal plots** are univariate distribution plot (histogram) of the data in each column.

```
sns.pairplot(data = diabetes_df.loc[:,["BloodPressure", "BMI", "Age"]])
```

<seaborn.axisgrid.PairGrid at 0x1de936fda30>



Exercise

Exercise.C

The "diamonds" dataset contains information about diamond characteristics, including price, carat (weight), cut, color, clarity, depth, table (width), and dimensions such as width, length, and depth. Below are descriptions of some of the variables used in this exercise.

- **price:** Price (USD)
- **carat:** Weight of the diamond
- **cut:** Quality of the cut (Fair, Good, Very Good, Premium, Ideal)
- **clarity:** A measurement of how clear the diamond is (I1 (worst), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (best))

(C.1) Load the dataset "diamonds" from searbon, and display the first 5 rows of the dataset.

Hint: `sns.load_dataset()`

```
:
```

(C.2) Show the distribution of the price.

Hint: `histplot()`

```
:
```

(C.3) Use a scatter plot to show the relationship between `carat` and `price` of the best clarity diamonds.

Hint: Select a subset by using `diamond_df.clarity == "IF"`

```
:
```

(C.4) Use a join plot to show the relationship between `carat` and `price` of the best clarity diamonds and their individual distributions.

```
:
```

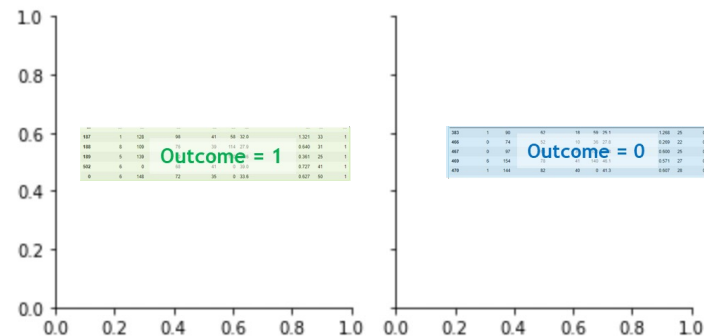
FacetGrid

- FacetGrid is basically a grid of subplots.
- The function `FacetGrid()` return a `FacetGrid` object which stores some information on how you want to break down your data visualization.

Step1: Initialize the grid

```
g = sns.FacetGrid(data = diabetes_df, col = "Outcome")
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
383	1	90	62	18	59	25.1	1.268	25	0
466	0	74	52	10	36	27.8	0.269	22	0
467	0	97					0.600	25	0
469	6	154	70	41	160	35.1	0.571	27	0
470	1	144	82	40	0	41.3	0.607	28	0
...
187	1	128	98	41	58	32.0	1.321	33	1
188	8	109	75	39	114	27.9	0.640	31	1
189	5	139					0.361	25	1
502	6	0					0.727	41	1
0	6	148	72	35	0	33.6	0.627	50	1

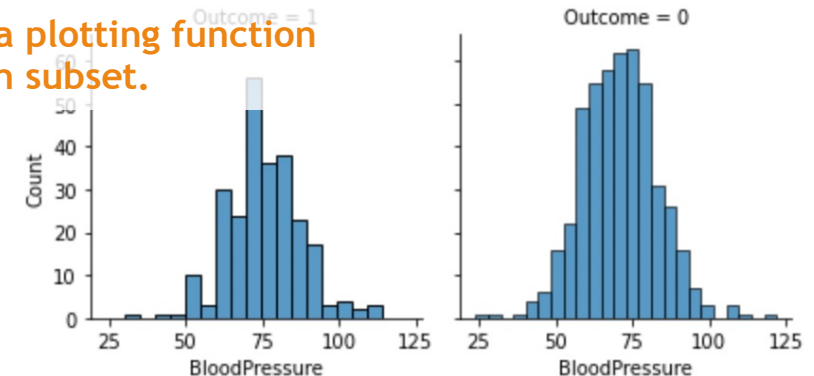


Step2: Draw a plot for each facet

```
g = sns.FacetGrid(data =diabetes_df, col="Outcome")  
g.map(sns.histplot, "BloodPressure")
```

<seaborn.axisgrid.FacetGrid at 0x7f9025b415b0>

Apply a plotting function to each subset.

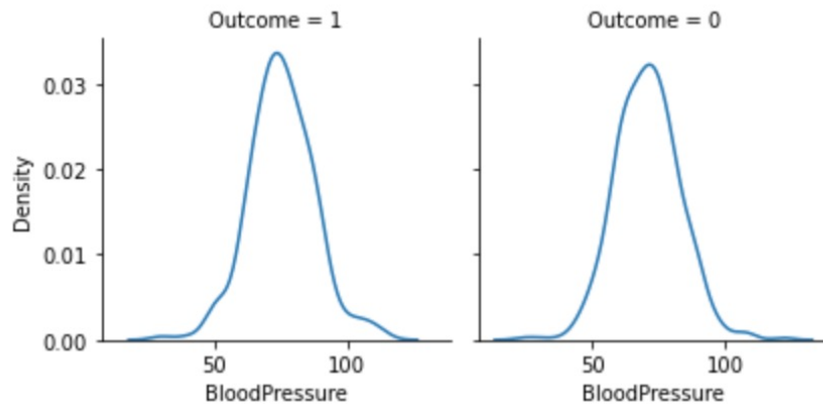


FacetGrid

- Change chart type

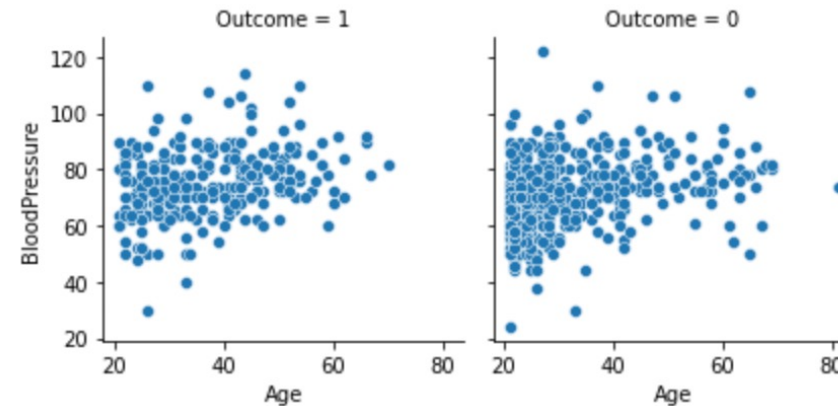
```
g = sns.FacetGrid(data = diabetes_df, col = "Outcome")  
g.map(sns.kdeplot, "BloodPressure")
```

<seaborn.axisgrid.FacetGrid at 0x7f9026123610>



```
g = sns.FacetGrid(data = diabetes_df, col = "Outcome")  
g.map(sns.scatterplot, "Age", "BloodPressure")
```

<seaborn.axisgrid.FacetGrid at 0x7f90120f7a30>



Exercise

Exercise.D

(D.1) Use the dataframe `diamond_df` in (C.1). Draw a price histogram for each `cut` category.

Hint: `col="cut"`

