# W06P1 Multi-dimensional data: In-class assignments for Section 1

Subhasis Ray*

*<2022-10-06 Thu>*

- Save your work in a file named `W06P2_classwork.py` and submit it to codePost.
- The first deadline for submitting your work on codePost is 1:30 PM. Do this to get attendance.
- The submission will open again in the afternoon, and you have until midnight to submit an updated version. Also submit the figures (saved files) on LMS.
- Do not worry about making your code look clean. Leave any comments/docstring you wrote intact when submitting.
- Refer to Pandas guide here: `https://pandas.pydata.org/pandas-docs/stable/user_guide/index.html`
- Look at matplotlib tutorial here: `https://matplotlib.org/stable/tutorials/introductory/pyplot.html`
- You can see a gallery of plots here: `https://matplotlib.org/stable/gallery/index.html`, they have links to the Python code for reproducing the plots, too.

## 1 Startup

In this assignment we shall explore more complex plotting

- Import numpy, pandas, and matplotlib.

*subhasis.ray@plaksha.edu.in

- In addition, we shall use the iris dataset from `sklearn` package.

```
#%% imports
import numpy as np
import pandas as pd
from sklearn import datasets
import matplotlib.pyplot as plt
```

## 2 Load data

We shall use the iris dataset for the initial part of this.

- Read the description of the data here `https://scikit-learn.org/stable/datasets/toy_dataset.html#iris-plants-dataset`
- Load the data as a pandas dataframe called 'df'. Note that `sklearn.datasets.load_iris(...)` and other data fetching/loading functions in sklearn take the keyword argument `as_frame`, which, if `True`, returns the data as a Pandas dataframe.

```
#%% Load the iris dataset
data = datasets.load_iris(as_frame=True)
df = data.data.copy()
# Print the column names
print('Original columns', df.columns.values)
# Rename columns (creating a new df)
df = df.rename(columns={'sepal length (cm)': 'slength',
                        'sepal width (cm)': 'swidth',
                        'petal length (cm)': 'plength',
                        'petal width (cm)': 'pwidth'})
print('After renaming', df.columns.values)
```

```
Original columns ['sepal length (cm)' 'sepal width (cm)' 'petal length (cm)'
 'petal width (cm)']
After renaming ['slength' 'swidth' 'plength' 'pwidth']
```

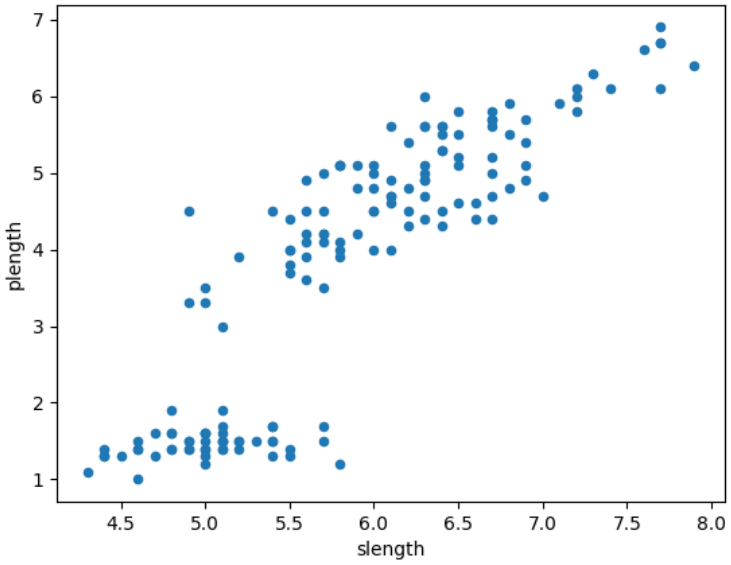
## 3 Scatter plot: 10 points

Scatter plots are useful for visualizing the relationship between two variables. Pandas gives you a quick and dirty way create scatter plots.

### 3.1 With pandas

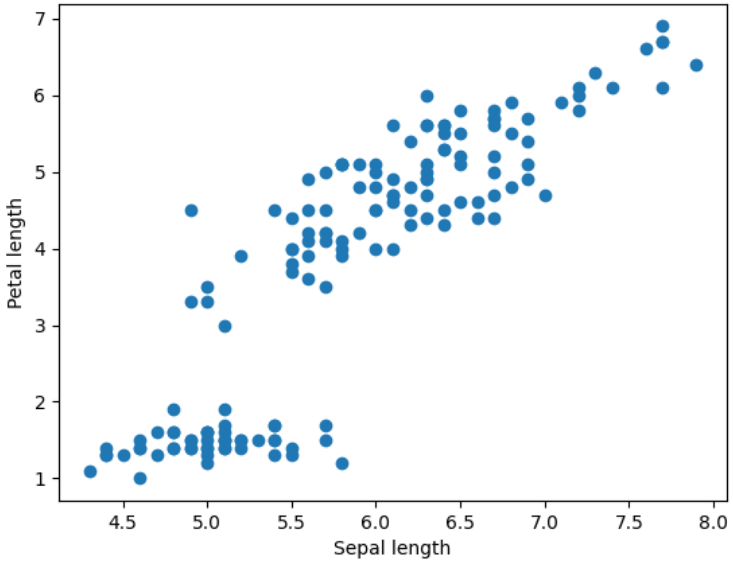
First see how we can plot one column (variable) against another:

```
#%% Scatter plot with pandas
df.plot.scatter(x='slength', y='plength')
fname = 'slen_plen_pandas.png'
plt.savefig(fname)
fname
```




### 3.2 With matplotlib:

```
#%% Scatter plot with matplotlib
fig, ax = plt.subplots()
ax.scatter(df.slength, df.plength)
ax.set_xlabel('Sepal length')
ax.set_ylabel('Petal length')
fname = 'slen_plen_mpl.png'
plt.savefig(fname)
fname
```

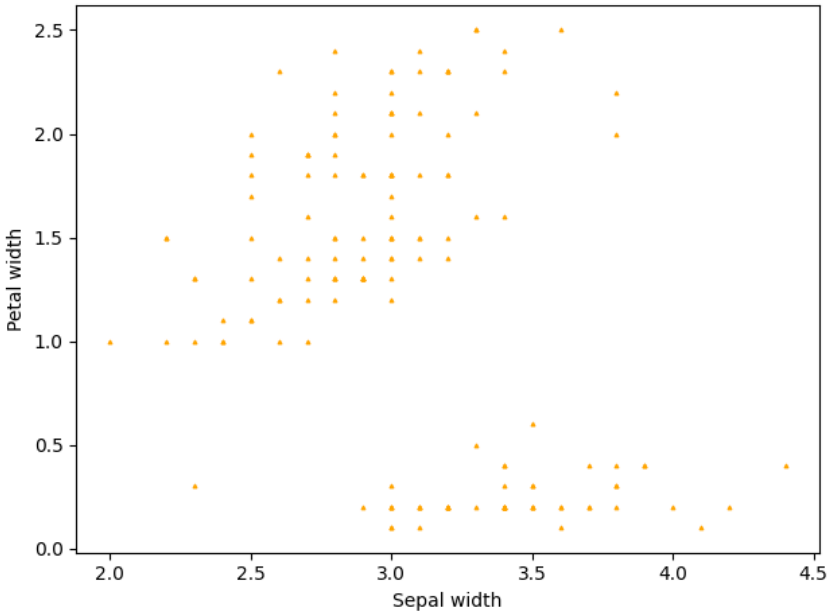



This looks denser because the default marker size is bigger.

- Read the matplotlib documentation for `scatter` to figure out how to specify marker style and marker size.

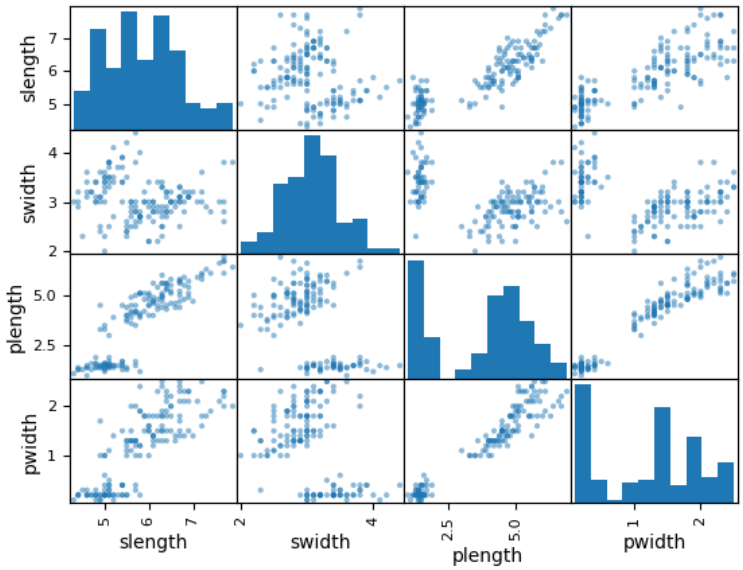
### 3.3 Plot and submit: 5 points

- Now plot petal width against sepal width using upright triangles as markers, setting marker size to 3.0, and color to orange. Save the figure as `swidth_pwidth_plot.png`. Upload this on LMS.




### 3.4 Scatterplot matrix: 5 points

For multivariate data like the iris plant data, it is convenient to plot each pair of variables in a matrix to get an idea of their relationship. Pandas has a convenience function for this: pandas.plotting.scatter$_{\text{matrix}}$. Create the scatterplot matrix for the iris dataset and save it as `iris_scatter_matrix.png`. When you do not have a figure handle (as in Pandas plot), you can save the current figure using `plt.savefig(filename)`. What do the diagonal entries in the plot indicate? Write it as a comment after your code.




## 4 3D plotting: 10 points

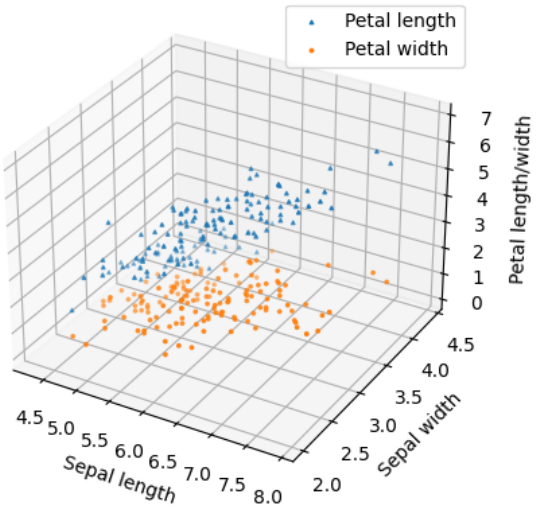
Things get tricky in 3D. Since our visualization is limited to 2D surfaces, all we can achieve is a 2D projection, with the possibility of moving the axes around to get a sense of 3D. The mplot3d toolkit allows some simple 3D plotting.

Although so far we have been using `plt.subplots(...)` function to create figure and axes(-grid) in one go, the traditional way of creating axes in matplotlib is `fig.add_subplot(nrows, ncols, axis_no, **kwargs)`. To use `mplot3d` for 3D axes you have to use this approach.

The developers of `mplot3d` toolkit have kept most things compatible with rest of the matplotlib API. From matplotlib v3.2.0 you can simply specify the `projection` kwarg to create 3D axes.

Now look into the `mplot3d` tutorial to figure out how to create 3D axes. Then plot petal length and petal width on z-axis, keeping sepal length and sepal width on x and y axes. Take a screenshot and save as `iris_3d.png`. Upload on LMS.

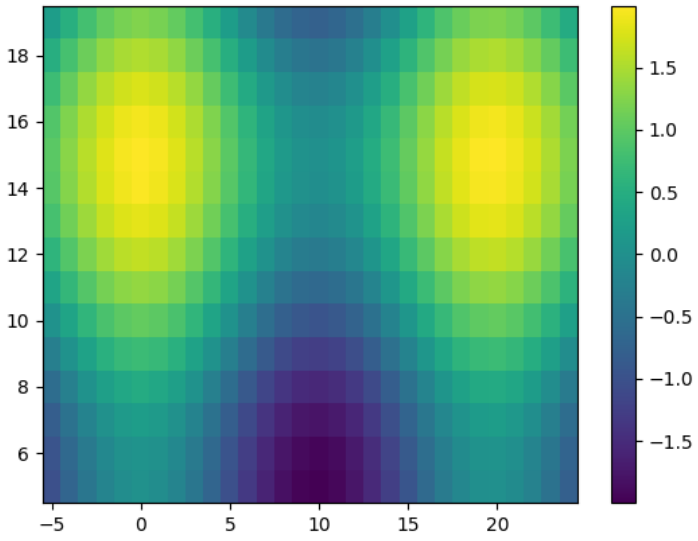
As you can see from the picture below, a static 3D plot is not very informative. Therefore avoid using them unless users can interact with the plot. Even then, consider using multiple 2D plots instead.




## 5 Visualizing with color maps: 10 points

What is a more informative way of visualizing higher dimensional data? We can exploit another aspect of our visual perception: color. It is useful when z is a function of x and y. Look up the documentation for `pcolormesh` and `colorbar` functions in matplotlib, and adapt the example to plot the data below where color indicates value. Add a colorbar to your figure and save as `colormesh_plot.png`. Upload it on LMS.

```
#%% data
x = np.arange(-5, 25)
y = np.arange(5, 20)
X, Y = np.meshgrid(x, y)
Z = np.cos(X / np.pi) - np.sin(Y / np.pi)
```




NOTE: You can also make histograms for 3D data using color. Instead of using rectangular bins, it is better to do that with hexagonal bins. The function is called `hexbin` in matplotlib.

## 6 Submit the figures on the LMS and your code on codePost.