



## AI4D LAB TTRAINING

---

[Zephania Reuben \(https://nsoma.me\)](https://nsoma.me)

July 17, 2023

---

## DATA SCIENCE | SEABORN

---

# Introduction

To analyse a set of data using Python, we make use of Matplotlib, a widely implemented 2D plotting library. Likewise, Seaborn is a visualization library in Python. It is built on top of Matplotlib.

## Seaborn Vs Matplotlib

It is summarized that if Matplotlib “tries to make easy things easy and hard things possible”, Seaborn tries to make a well-defined set of hard things easy too.” Seaborn helps resolve the two major problems faced by Matplotlib; the problems are:

- Default Matplotlib parameters
- Working with data frames

As Seaborn compliments and extends Matplotlib, the learning curve is quite gradual. If you know Matplotlib, you are already half way through Seaborn.

## Important Features of Seaborn

Seaborn is built on top of Python's core visualization library Matplotlib. It is meant to serve as a complement, and not a replacement. However, Seaborn comes with some very important features. Let us see a few of them here. The features help in -

- Built in themes for styling matplotlib graphics
- Visualizing univariate and bivariate data
- Fitting in and visualizing linear regression models
- Plotting statistical time series data
- Seaborn works well with NumPy and Pandas data structures
- It comes with built in themes for styling Matplotlib graphics In most cases, you will still use Matplotlib for simple plotting. The knowledge of Matplotlib is recommended to tweak Seaborn's default plots.

# Environment Setup

Let us begin with the installation and understand how to get started as we move ahead.

## Installing Seaborn and getting started

### *Using Pip Installer*

To install the latest release of Seaborn, you can use pip:

```
pip install seaborn
```

It is also possible to install the released version using conda:

```
conda install seaborn
```

## Dependencies

Consider the following dependencies of Seaborn:

- Python 3.6+
- numpy
- scipy
- pandas
- matplotlib

# Importing Datasets and Libraries

Seaborn comes handy when dealing with DataFrames, which is most widely used data structure for data analysis.

```
In [1]: # import libraries
import matplotlib.pyplot as plt
import seaborn as sb
```

Seaborn comes with a few important datasets in the library. When Seaborn is installed, the datasets download automatically.

```
In [2]: sb.load_dataset('titanic')
```

```
Out[2]:
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	ad
0	0	3	male	22.0	1	0	7.2500	S	Third	man	
1	1	1	female	38.0	1	0	71.2833	C	First	woman	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	
3	1	1	female	35.0	1	0	53.1000	S	First	woman	
4	0	3	male	35.0	0	0	8.0500	S	Third	man	
...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	
887	1	1	female	19.0	0	0	30.0000	S	First	woman	
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	
889	1	1	male	26.0	0	0	30.0000	C	First	man	
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	

891 rows × 15 columns

```
In [3]: sb.load_dataset('iris')
```

```
Out[3]:
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

```
In [4]: #import datasets

df = sb.load_dataset('tips')
df.head()
```

```
Out[4]:
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

```
In [5]: sb.get_dataset_names()
```

```
Out[5]: ['anagrams',
'anscombe',
'attention',
'brain_networks',
'car_crashes',
'diamonds',
'dots',
'dowjones',
'exercise',
'flights',
'fmri',
'geyser',
'glue',
'healthexp',
'iris',
'mpg',
'penguins',
'planets',
'seaice',
'taxis',
'tips',
'titanic']
```

To view all the available data sets in the Seaborn library, you can use the following command and with the `get_dataset_names()` function as shown below:

```
In [ ]: print(sb.get_dataset_names())
```

```
In [ ]: print(sb.load_dataset('titanic').head())
```

## Figure Aesthetic

**Aesthetics** means a set of principles concerned with the nature and appreciation of beauty, especially in art. Visualization is an art of representing data in effective and easiest possible way.

Matplotlib library highly supports customization, but knowing what settings to tweak to achieve an attractive and anticipated plot is what one should be aware of to make use of it. Unlike Matplotlib, Seaborn comes packed with customized themes and a high -level interface for customizing and controlling the look of Matplotlib figures.

```
In [7]: import numpy as np  
s = np.linspace(0,3,5)
```

```
In [8]: s.shape
```

```
Out[8]: (5,)
```

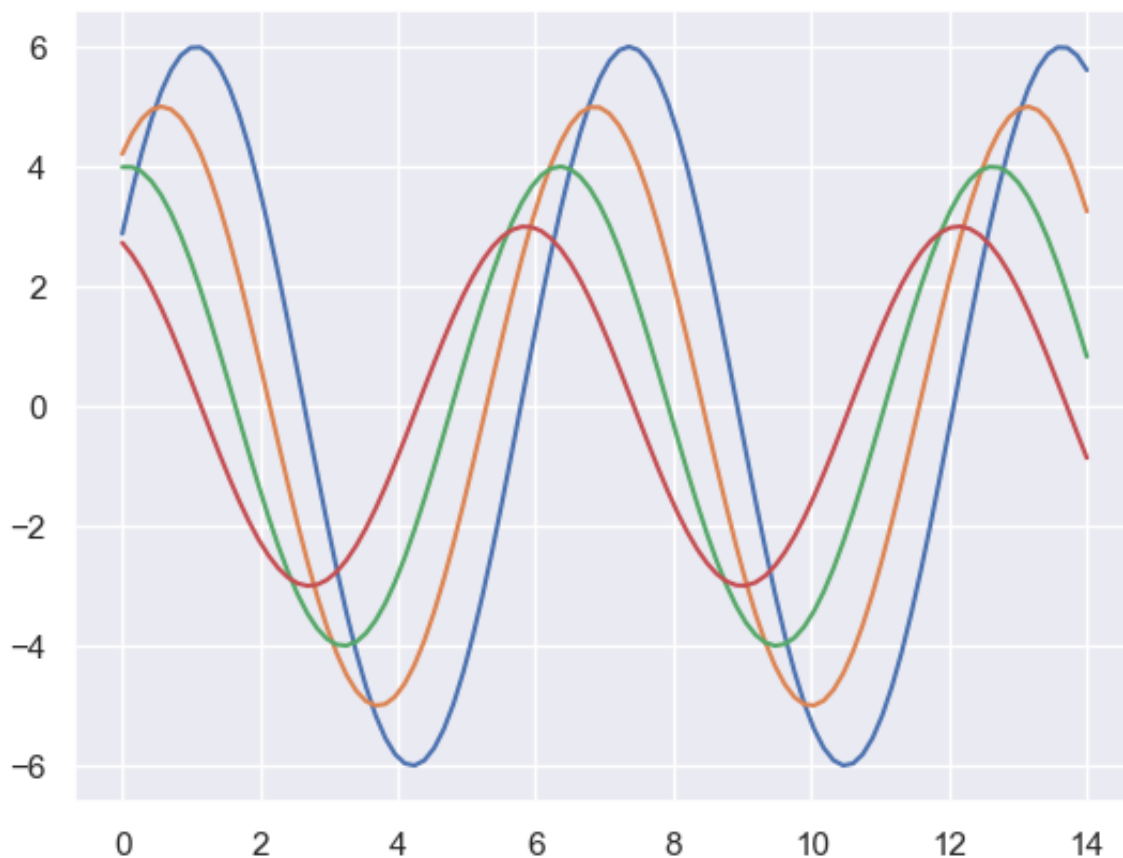
```
In [9]: s
```

```
Out[9]: array([0. , 0.75, 1.5 , 2.25, 3.  ])
```

```
In [10]: from matplotlib import pyplot as plt
```

```
In [11]: def sinplot(flip=1):  
    x = np.linspace(0, 14, 100)  
    for i in range(1, 5):  
        plt.plot(x, np.sin(x + i * .5) * (7 - i) * flip)
```

```
In [12]: import seaborn as sb
sb.set()
sinplot()
plt.show()
```



To change the same plot to Seaborn defaults, use the `set()` function:

```
In [ ]: import numpy as np
from matplotlib import pyplot as plt
def sinplot():
    x = np.linspace(0, 14, 100)
    for i in range(1, 5):
        plt.plot(x, np.tanh(x + i * .5))
sinplot()
plt.show()
```

$$y = \sin\left(\left(x + \frac{i}{2}\right)(7 - i)\right)$$

The above two figures show the difference in the default Matplotlib and Seaborn plots. The representation of data is same, but the representation style varies in both. Basically, Seaborn splits the Matplotlib parameters into two groups :

- Plot styles
- Plot scale

## Seaborn Figure Styles

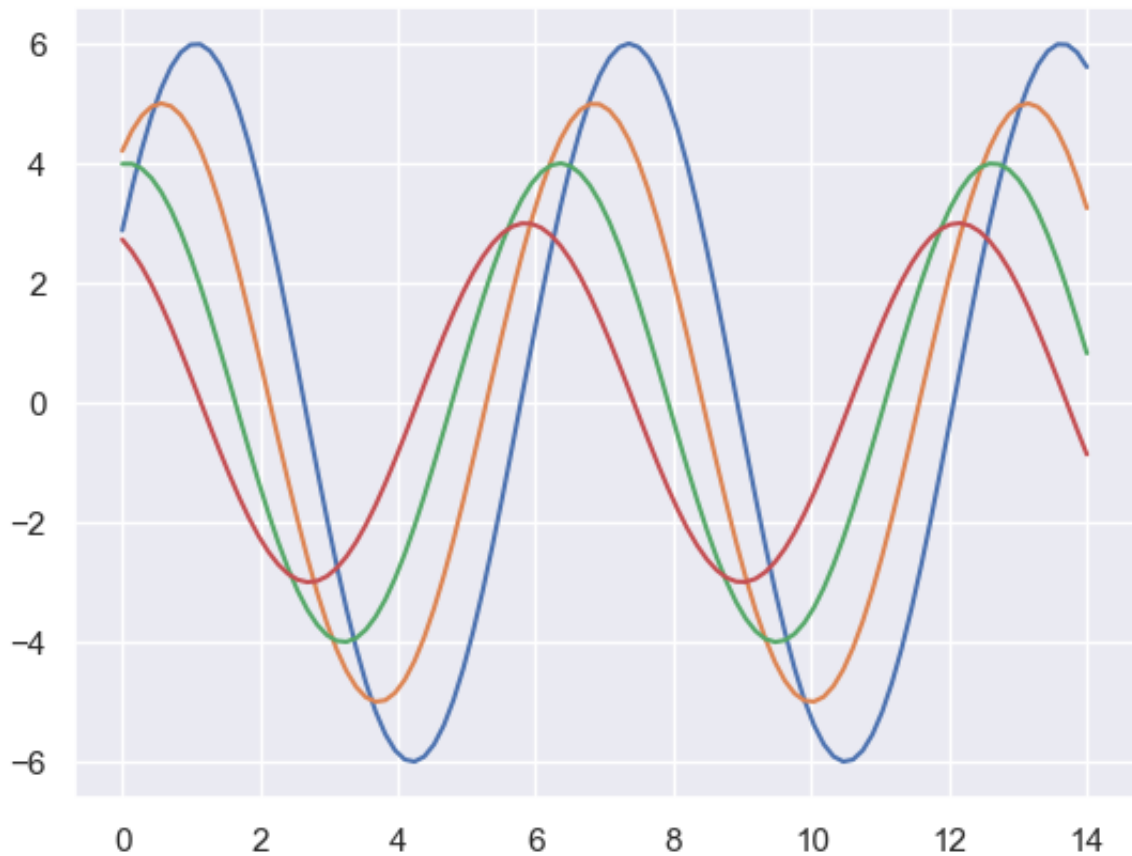
The interface for manipulating the styles is `set_style()` . Using this function you can set the theme of the plot. As per the latest updated version, below are the five themes available.

- Darkgrid
- Whitegrid
- Dark
- White
- Ticks

In [ ]:



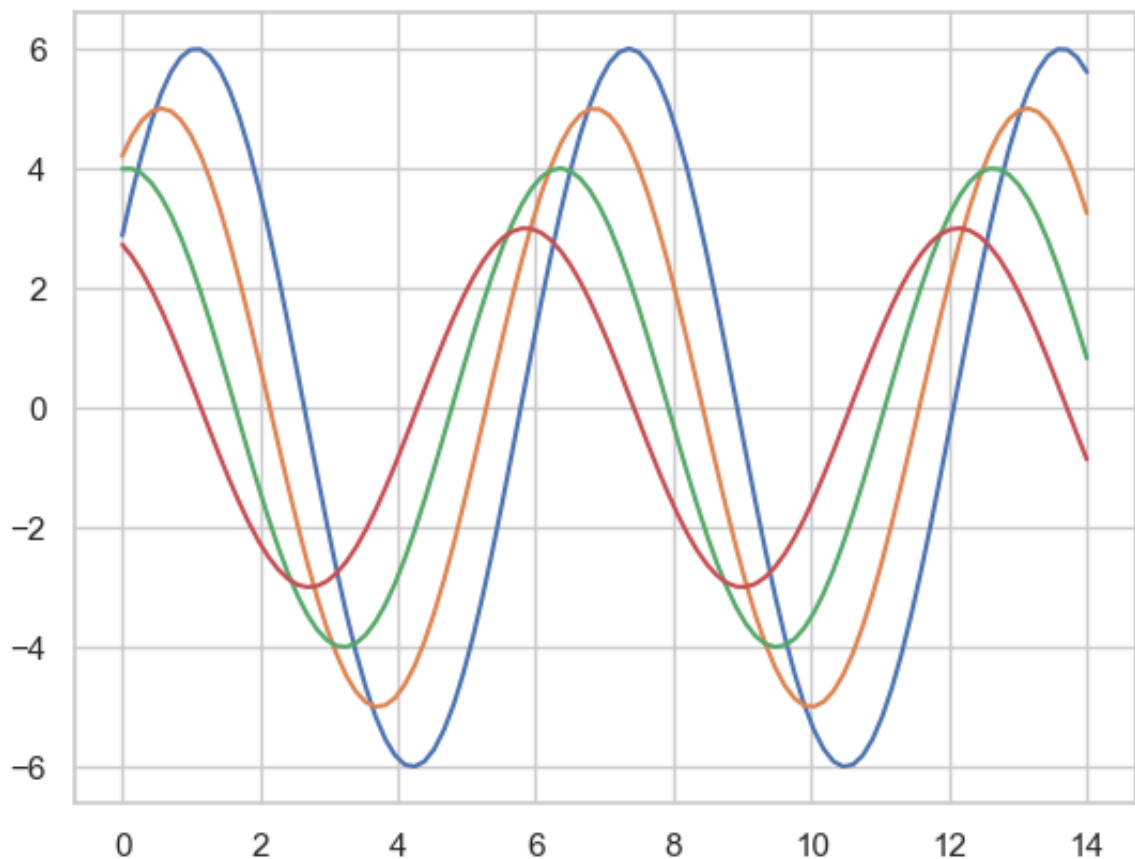
```
In [16]: import numpy as np
from matplotlib import pyplot as plt
def sinplot(flip=1):
    x = np.linspace(0, 14, 100)
    for i in range(1, 5):
        plt.plot(x, np.sin(x + i * .5) * (7 - i) * flip)
import seaborn as sb
sb.set_style("darkgrid")
sinplot()
plt.show()
```



## Removing Axes Spines

In the white and ticks themes, we can remove the top and right axis spines using the `despine()` function.

```
In [28]: import numpy as np
from matplotlib import pyplot as plt
def sinplot(flip=1):
    x = np.linspace(0, 14, 100)
    for i in range(1, 5):
        plt.plot(x, np.sin(x + i * .5) * (7 - i) * flip)
import seaborn as sb
sb.set_style("whitegrid", {'axes.axisbelow': True})
sinplot()
plt.show()
```



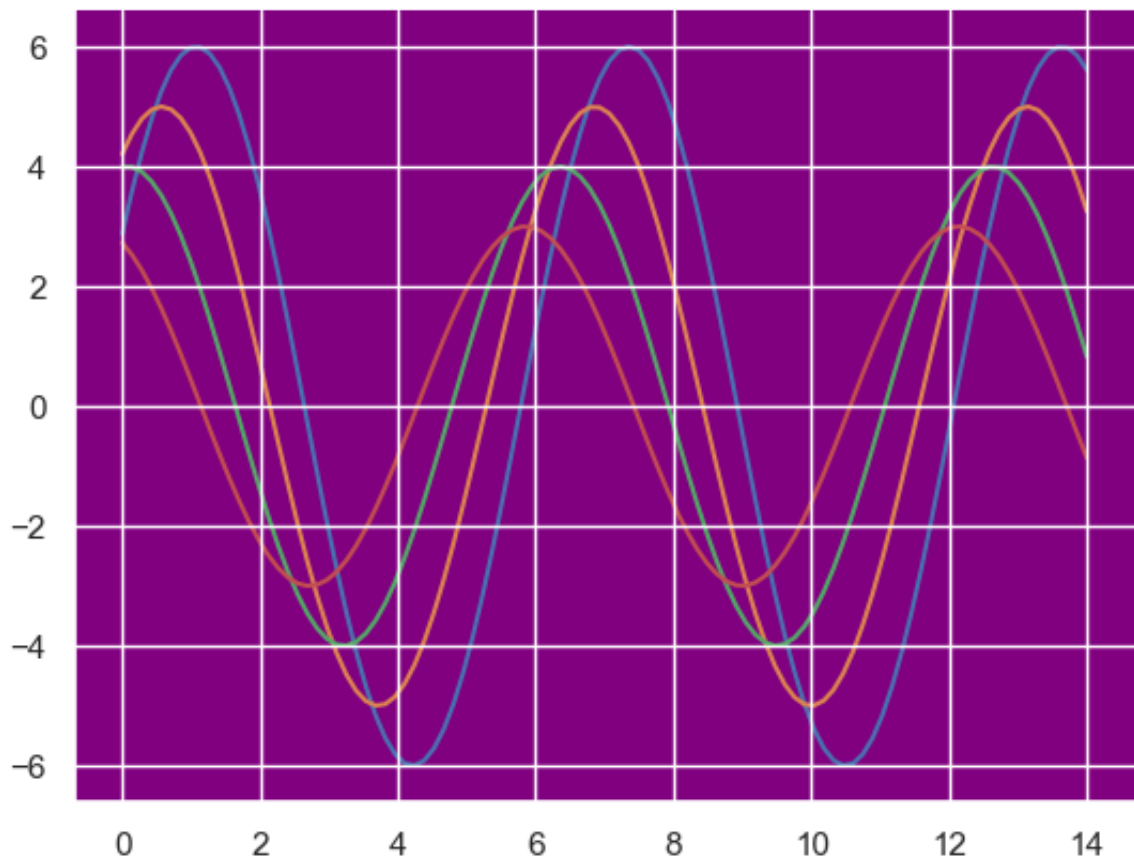
In the regular plots, we use left and bottom axes only. Using the `despine()` function, we can avoid the unnecessary right and top axes spines, which is not supported in Matplotlib.

## Overriding the Elements

If you want to customize the Seaborn styles, you can pass a dictionary of parameters to the `set_style()` function. Parameters available are viewed using `axes_style()` function.

```
In [ ]: import seaborn as sb
print(sb.axes_style())
```

```
In [29]: import numpy as np
from matplotlib import pyplot as plt
def sinplot(flip=1):
    x = np.linspace(0, 14, 100)
    for i in range(1, 5):
        plt.plot(x, np.sin(x + i * .5) * (7 - i) * flip)
import seaborn as sb
sb.set_style("darkgrid", {'axes.axisbelow': False, 'axes.facecolor':
'purple'})
sinplot()
sb.despine()
plt.show()
```



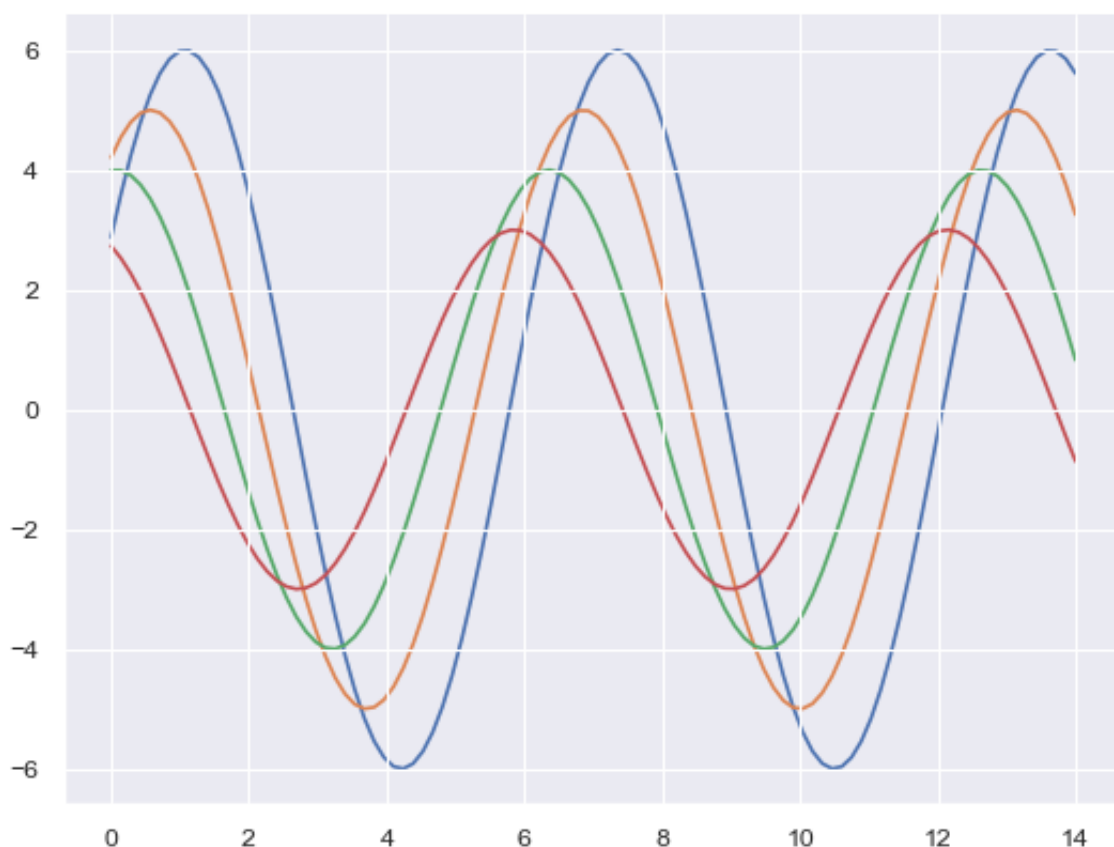
## Scaling Plot Elements

We also have control on the plot elements and can control the scale of plot using the `set_context()` function. We have four preset templates for contexts, based on relative size, the contexts are named as follows :

- Paper
- Notebook
- Talk
- Poster

By default, context is set to `notebook` ; and was used in the plots above.

```
In [32]: import numpy as np
from matplotlib import pyplot as plt
def sinplot(flip=1):
    x = np.linspace(0, 14, 100)
    for i in range(1, 5):
        plt.plot(x, np.sin(x + i * .5) * (7 - i) * flip)
import seaborn as sb
sb.set_style("darkgrid", {'axes.axisbelow': False})
sinplot()
sb.despine()
sb.set_context('notebook')
plt.show()
```



## Color Palette

Color plays an important role than any other aspect in the visualizations. When used effectively, color adds more value to the plot. A palette means a flat surface on which a painter arranges and mixes paints.

## Building Color Palette

Seaborn provides a function called `color_palette()`, which can be used to give colors to plots and adding more aesthetic value to it.

```
seaborn.color_palette(palette=None, n_colors=None, desat=None)
```

Following are the readily available Seaborn palettes:

- Deep
- Muted
- Bright
- Pastel
- Dark
- Colorblind

It is hard to decide which palette should be used for a given data set without knowing the characteristics of data. Being aware of it, we will classify the different ways for using `color_palette()` types:

- qualitative
- sequential
- diverging

We have another function `seaborn.palplot()` which deals with color palettes. This function plots the color palette as horizontal array. We will know more regarding `seaborn.palplot()` in the coming examples.

## Qualitative Color Palettes

Qualitative or categorical palettes are best suitable to plot the categorical data.

```
In [41]: from matplotlib import pyplot as plt
import seaborn as sb
current_palette = sb.color_palette(palette='bright', n_colors=8)
sb.palplot(current_palette)
plt.show()
```



We haven't passed any parameters in `color_palette()` ; by default, we are seeing 10 colors. You can see the desired number of colors by passing a value to the `n_colors` parameter. Here, the `palplot()` is used to plot the array of colors horizontally.

## Sequential Color Palettes

Sequential plots are suitable to express the distribution of data ranging from relative lower values to higher values within a range.

Appending an additional character 's' to the color passed to the color parameter will plot the Sequential plot.

```
In [43]: from matplotlib import pyplot as plt
import seaborn as sb
current_palette = sb.color_palette(palette='Greens', n_colors=9)
sb.palplot(current_palette)
plt.show()
```



## Diverging Color Palette

Diverging palettes use two different colors. Each color represents variation in the value ranging from a common point in either direction.

Assume plotting the data ranging from -1 to 1. The values from -1 to 0 takes one color and 0 to +1 takes another color.

By default, the values are centered from zero. You can control it with parameter `center` by passing a value.

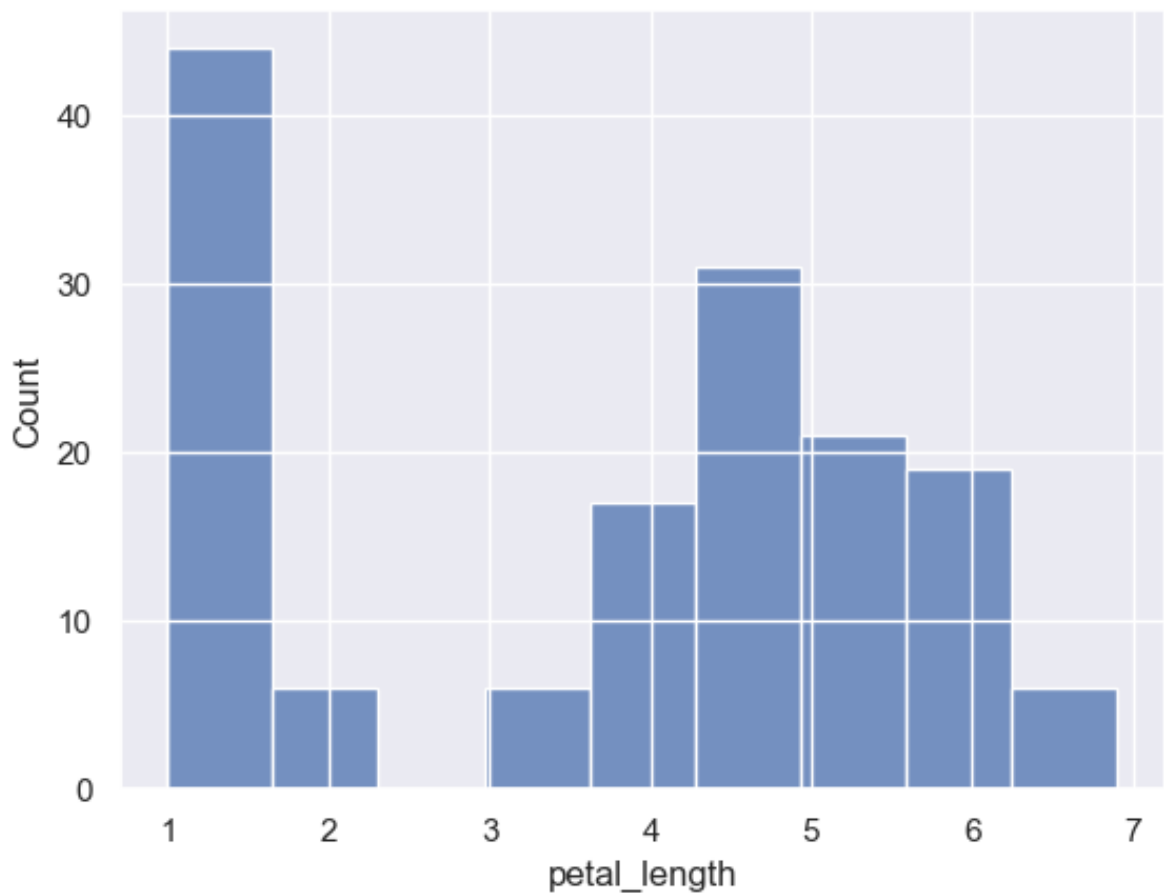
```
In [37]: from matplotlib import pyplot as plt
import seaborn as sb
current_palette = sb.color_palette()
sb.palplot(sb.color_palette(palette="bwr", n_colors=7))
plt.show()
```



# Histogram

Histograms represent the data distribution by forming bins along the range of the data and then drawing bars to show the number of observations that fall in each bin.

```
In [44]: import seaborn as sb
from matplotlib import pyplot as plt
df = sb.load_dataset('iris')
sb.histplot(df['petal_length'])
plt.show()
```



```
In [ ]: ## Pie Chart
```



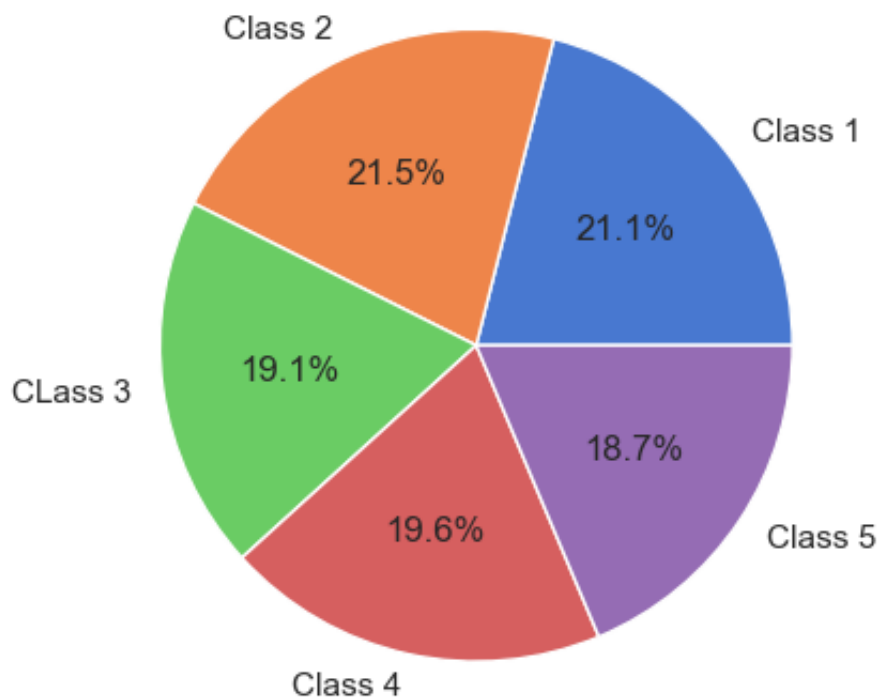
```
In [46]: # importing libraries
import matplotlib.pyplot as plt
import seaborn

# declaring data
data = [44, 45, 40, 41, 39]
keys = ['Class 1', 'Class 2', 'Class 3', 'Class 4', 'Class 5']

# define Seaborn color palette to use
palette_color = seaborn.color_palette('muted')

# plotting data on chart
plt.pie(data, labels=keys, colors=palette_color, autopct='%.1f%%')

# displaying chart
plt.show()
```



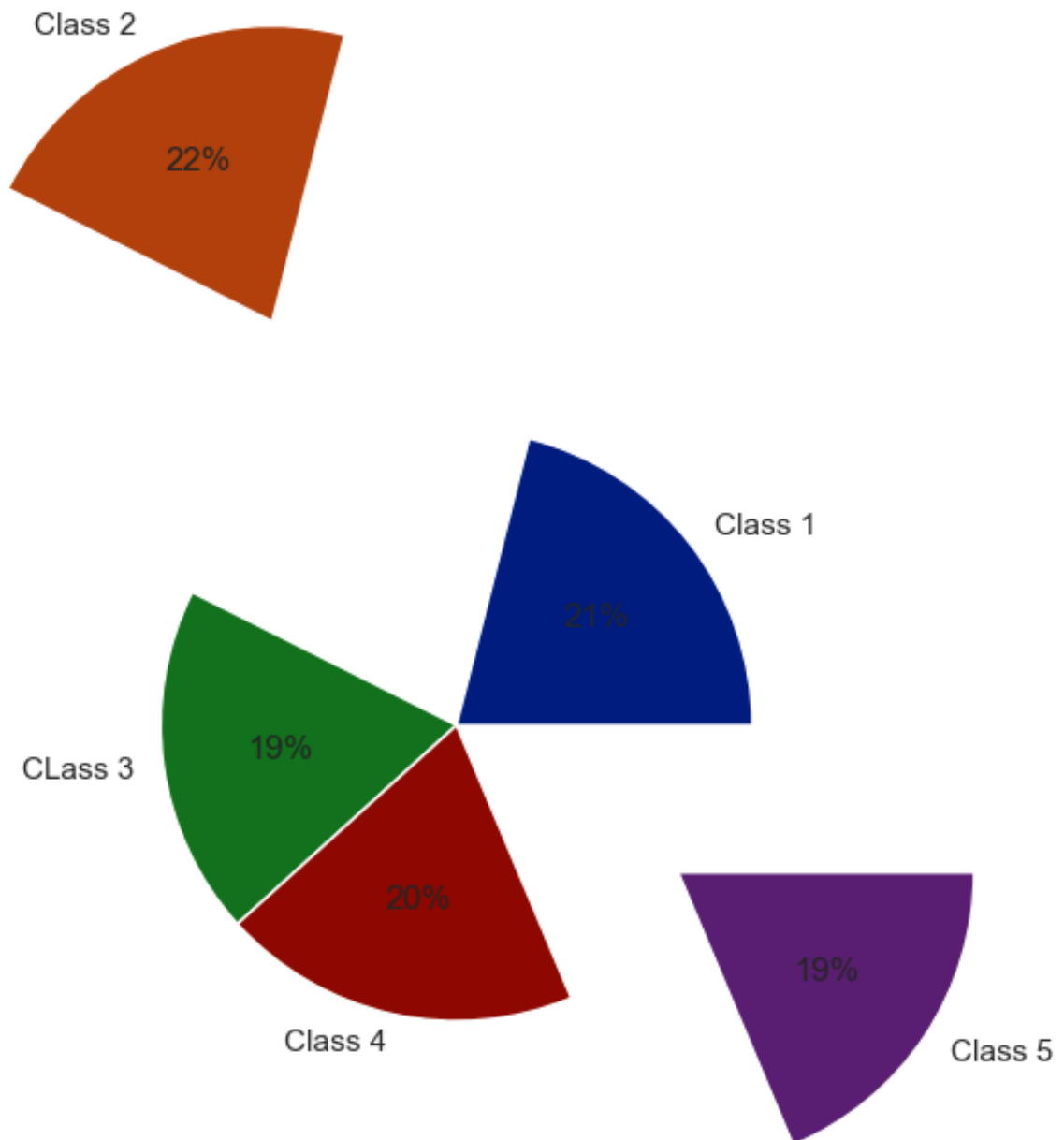
```
In [50]: # importing libraries
import matplotlib.pyplot as plt
import seaborn

# declaring data
data = [44, 45, 40, 41, 39]
keys = ['Class 1', 'Class 2', 'Class 3', 'Class 4', 'Class 5']

# declaring exploding pie
explode = [0, 1.5, 0, 0, 0.9]
# define Seaborn color palette to use
palette_color = seaborn.color_palette('dark')

# plotting data on chart
plt.pie(data, labels=keys, colors=palette_color,
        explode=explode, autopct='%0f%%')

# displaying chart
plt.show()
```



In [ ]: