Data Viusalisation with Python Programming

- Presentation By Uplatz
- Contact us: https://training.uplatz.com
- Email: info@uplatz.com
- Phone: +44 7836 212635

Specialized data Visualisation tools (Part-II)

Learning outcomes:

Three-Dimensional Plotting in Matplotlib

- 3D Line Plot
- 3D Scatter Plot
- 3D Contour Plot
- 3D Wireframe Plot
- 3D Surface Plot

Three-Dimensional Plotting in Matplotlib

Matplotlib was initially designed with only two-dimensional plotting in mind. Around the time of the 1.0 release, some three-dimensional plotting utilities were built on top of Matplotlib's two-dimensional display, and the result is a convenient (if somewhat limited) set of tools for three-dimensional data visualization. Three-dimensional plots are enabled by importing the **mplot3d** toolkit, included with the main Matplotlib installation.

```
from mpl_toolkits import mplot3d
```

Uplatz

Three-Dimensional Plotting in Matplotlib

Once this submodule is imported, a three-dimensional axes can be created by passing the keyword projection='3d' to any of the normal axes creation routines:

```
import numpy as np
import matplotlib.pyplot as plt
fig = plt.figure()
ax = plt.axes(projection='3d')
```

Three-Dimensional Plotting in Matplotlib

With this three-dimensional axes enabled, we can now plot a variety of three-dimensional plot types. Three-dimensional plotting is one of the functionalities that benefits immensely from viewing figures interactively rather than statically in the notebook.

3D Line Plot

The most basic three-dimensional plot is a **3D line plot** created from sets of (x, y, z) triples. This can be created using the ax.plot3D function.
First, we have to install matplotlib to import the mplot3d toolkit. Mplot3d is a toolkit which will help for matplotlib to generate 3 axes on the graph. A three-dimensional axes can be created by passing the keyword projection='3d' to any of the normal axes creation routines.

3D Line Plot

Example_1:**Example 1:** Let’s create a basic 3D line plot using the ax.plot3D() function.

```
from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt
ax = plt.axes(projection='3d')
z = np.linspace(0, 1, 100)
x = z * np.sin(20 * z)
y = z * np.cos(20 * z)
ax.plot3D(x, y, z, 'gray')
ax.set_title('3D line plot')
plt.show()
```

3D Line Plot

Example_2:

```
from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt
ax = plt.axes(projection='3d')
z = np.linspace(1, 100, 100)
x = z * np.cos(200 * z)
y = z * np.sin(200 * z)
ax.plot3D(x, y, z, 'red')
ax.set_title('3D line plot')
plt.show()
```

3D Scatter Plot

A **3D Scatter Plot** is a mathematical diagram, the most basic version of three-dimensional plotting used to display the properties of data as three variables of a dataset using the Cartesian coordinates.

3D scatter plots are used to plot data points on three axes in the attempt to show the relationship between three variables. Each row in the data table is represented by a marker whose position depends on its values in the columns set on the X, Y, and Z axes.

3D Scatter Plot

Creating 3D Scatter Plot:

To create a 3D Scatter plot, Matplotlib's **mplot3d toolkit** is used to enable three dimensional plotting. Generally 3D scatter plot is created by using ax.scatter3D() the function of the matplotlib library which accepts a data sets of X, Y and Z to create the plot while the rest of the attributes of the function are the same as that of two dimensional scatter plot.

3D Scatter Plot

Example 1: Let’s create a basic 3D scatter plot using the ax.scatter3D() function.

```
from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt
ax = plt.axes(projection='3d')
x =[1,2,3,4,5,6,7,8,9,10]
y =[5,6,2,3,13,4,1,2,4,8]
z =[2,3,3,3,5,7,9,11,9,10]
ax.scatter3D(x, y, z, 'green')
ax.set_title('3D Scatter plot')
plt.show()
```

3D Scatter Plot

Example 2:

```
from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt
ax = plt.axes(projection='3d')
z = np.linspace(0, 1, 100)
x = z * np.sin(20 * z)
y = z * np.cos(20 * z)
ax.scatter3D(x, y, z, ‘pink')
ax.set_title('3D Scatter plot')
plt.show()
```

Uplatz

3D Contour Plot

Contour plots (sometimes called **Level Plots**) are a way to show a **three-dimensional** surface on a **two-dimensional plane**. It graphs two predictor variables X Y on the y-axis and a response variable Z as contours. These contours are sometimes called the z-slices.

A contour plot is appropriate if you want to see how value Z changes as a function of two inputs X and Y, such that Z = f(X,Y). A contour line of a function of two variables is a curve along which the function has a constant value.

3D Contour Plot

The **ax.contour3D()** function creates three-dimensional contour plot. It requires all the input data to be in the form of two-dimensional regular grids, with the Z-data evaluated at each point. The 3d plots are enabled by importing the **mplot3d** toolkit.

Here, we will show a three-dimensional contour diagram of a three-dimensional sinusoidal function.

3D Contour Plot

Creating 3D Contour Plot:

```
from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-6, 6, 30)
y = np.linspace(-6, 6, 30)
X, Y = np.meshgrid(x, y)
Z = np.sin(np.sqrt(X ** 2 + Y ** 2))
```

3D Contour Plot

Creating 3D Contour Plot:

```
ax = plt.axes(projection='3d')
ax.contour3D(X, Y, Z, 50, cmap='binary')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
ax.set_title('3D contour')
plt.show()
```

3D Wireframe Plot

Wireframe plot takes a grid of values and projects it onto the specified three-dimensional surface, and can make the resulting three-dimensional forms quite easy to visualize.
The **plot_wireframe()** function is used for this purpose.

3D Wireframe Plot

Creating 3D Wireframe Plot:

```
from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-6, 6, 30)
y = np.linspace(-6, 6, 30)
X, Y = np.meshgrid(x, y)
Z = np.sin(np.sqrt(X ** 2 + Y ** 2))
```

3D Wireframe Plot

Creating 3D Wireframe Plot:

```
ax = plt.axes(projection='3d')
ax. plot_wireframe(X, Y, Z, color='black')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
ax.set_title('3D Wireframe')
plt.show()
```

3D Surface Plot

A **Surface Plot** is a representation of three-dimensional dataset. It describes a functional relationship between two independent variables X and Z and a designated dependent variable Y, rather than showing the individual data points. It is a companion plot of the contour plot. It is similar to the wireframe plot, but each face of the wireframe is a filled polygon. This helps to create the topology of the surface which is being visualized.

3D Surface Plot

Creating 3D Surface Plot:

The axes3d present in Matplotlib's mpl_toolkits.mplot3d toolkit provides the necessary functions used to create 3D surface plots. Surface plots are created by using the ax.plot_surface() function.

3D Surface Plot

Creating 3D Surface Plot: Example_1

```
from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-6, 6, 30)
y = np.linspace(-6, 6, 30)
X, Y = np.meshgrid(x, y)
Z = np.sin(np.sqrt(X ** 2 + Y ** 2))
```

3D Surface Plot

Creating 3D Surface Plot: Example_1

```
ax = plt.axes(projection='3d')
ax. plot_surface(X, Y, Z, color=green')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
ax.set_title('3D Wireframe')
plt.show()
```

3D Surface Plot

Creating 3D Surface Plot: Example_2

Using **cmap,** we can add **colormap** for the surface.

We can also add the colorbar using **fig.colorbar()**

Let’s see the example.

Thank you

Uplatz