Data Viusalisation with Python Programming

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

Uplatz

Specialized data Visualisation tools (Part-I)

Uplatz

Learning outcomes:

- Bubble charts
- Contour plots
- Quiver Plot

Bubble charts

Bubble charts display data as a cluster of circles. A bubble plot is a scatterplot where a third dimension is added: the value of an additional variable is represented through the **size of the dots**. You need 3 numerical variables as input: one is represented by the X axis, one by the Y axis, and one by the size.

Bubble charts

Note that too many bubble make the chart hard to read, so this type of representation is usually not recommended for big amount of data. Last but not least, note that the area of the circles must be proportional to the **value**, not to the **radius**, to avoid exaggerate the variation in your data.

Bubble charts

Creating Bubble Chart/Plot:

A bubble plot is very close to a scatterplot.
With Matplotlib, we will construct them using the same **scatter** function. However, we will use the ‘**s**‘ argument to map a **third numerical variable** to the **size** of the marker. This will give us an additive information on the same graphic.

Bubble charts

Creating Bubble Chart/Plot:

Example_1:

```
import matplotlib.pyplot as plt
import numpy as np
# create data
x = np.random.rand(40)
y = np.random.rand(40)
z = np.random.rand(40)
# use the scatter function
plt.scatter(x, y, s=z*1000)
plt.show()
```

Uplatz

Bubble charts

Creating Bubble Chart/Plot:
Example_2: Customizing your bubble plot

We can change the colour of the bubble in our plot with c ='red'.(Specify any colour)
We can also change the shape with marker.
We can also add line around dot using linewidth.
We can also add the title and labels on x-axis and y-axis.
Let's see the example.

Contour plots

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.

Contour plots

The independent variables x and y are usually restricted to a regular grid called **meshgrid**. The **numpy.meshgrid** creates a rectangular grid out of an **array of x values** and an **array of y values**. Matplotlib API contains **contour()** and **contourf()** functions that draw contour lines and filled contours, respectively. Both functions need three parameters **x,y and z.**

Contour plots

Creating Contour Plot:

The **matplotlib.pyplot.contour()** are usually useful when **Z = f(X, Y)** i.e Z changes as a function of input X and Y. A **contourf()** is also available which allows us to draw filled contours.

Contour plots

Creating Contour Plot:

Example_1:

```
import numpy as np
import matplotlib.pyplot as plt
#Generating 100 numbers from -3.0 to 3.0
xlist = np.linspace(-3.0, 3.0, 100)
ylist = np.linspace(-3.0, 3.0, 100)
X, Y = np.meshgrid(xlist, ylist)
Z = np.sqrt(X**2 + Y**2)
cp = plt.contourf(X, Y, Z)
plt.colorbar(cp) # Add a colorbar to a plot
plt.show()
```

Contour plots

Creating Contour Plot:

Example_2: Creating shape like ellipse

```
import numpy as np
import matplotlib.pyplot as plt
xlist = np.linspace(-4, 4, 9)
ylist = np.linspace(-4, 4, 9)
X, Y = np.meshgrid(xlist, ylist)
Z = np.sqrt(X**2 + 4*(Y**2))
cp = plt.contourf(X, Y, Z)
plt.colorbar(cp) # Add a colorbar to a plot
plt.show()
```

Quiver Plot

Quiver plot is basically a type of 2D plot which shows vector lines as arrows. This type of plots are useful in Electrical engineers to visualize electrical potential and show stress gradients in Mechanical engineering.
A quiver plot displays the velocity vectors as arrows with components (u,v) at the points (x,y).
quiver(x,y,u,v)
The above command plots vectors as arrows at the coordinates specified in each corresponding pair of elements in x and y.

Quiver Plot

We need to use the quiver() function that takes four arguments for creating quiver plot:

Syntax:

matplotlib.pyplot.quiver(x_pos, y_pos, x_dir, y_dir, color)

Here x_pos and y_pos are the starting positions of the arrow while x_dir and y_dir are the directions of the arrow.

Quiver Plot

Creating Quiver Plot: Example_1

Let’s start creating a simple quiver plot containing one arrow.

```
import numpy as np
import matplotlib.pyplot as plt
# Creating arrow
x_pos = 0
y_pos = 0
x_direct = 1
y_direct = 1
plt.quiver(x_pos, y_pos, x_direct, y_direct)
plt.show()
```

Quiver Plot

Creating Quiver Plot: Example_2
Let’s add another arrow to the plot passing through two starting points and two directions. By keeping the original arrow starting at origin(0, 0) and pointing towards up and to the right direction(1, 1), and create the second arrow starting at (0, 0) pointing down in direction(0, -1).

Let’s see the example.

Quiver Plot

Creating Quiver Plot: Quiver Plot using meshgrid Example_3:

A quiver plot containing two arrows is a good start, but it is too slow and too long to add arrows to the quiver plot one by one. So to create a fully 2D surface of arrows we will use meshgrid() method of Numpy. First, create a set of arrays named X and Y which represents the starting positions of x and y respectively of each arrow on the quiver plot. The starting positions of x, y arrows can also be used to define the x and y components of each arrow direction.

Quiver Plot

Creating Quiver Plot: Quiver Plot using meshgrid Example_3:

In the following plot u and v denote the array of directions of the quiver arrows and we will define the arrow direction based on the arrow starting point by using the equations below:

x_{direction} = cos(x_{starting \ point})
y_{direction} = sin(y_{starting \ point})

Let’s see the example.

Thank you

Uplatz