DATA VISUALIZATIONS IN PYTHON WITH MATPLOTLIB

Sidita Duli[1], PhD
[1]University "Luigj Gurakuqi", Shkodra, Albania, sidita.duli@unishk.edu.al

Abstract: *The amount and the complexity of information processed in science, business, engineering is increasing at the highest rates. Nowadays, scientists rely on graphs, charts, scanner plots, curve plots, and image annotations. Data visualization is a crucial part of all analytical projects. Python programming language is very popular among scientists because of the great tools it offers, with lots of features to build graphics. One of the most popular Python libraries used in data visualization is matplotlib.*
This article discusses different ways of data visualization in Python. It highlights the ease of use of this programming language in data science.

Keywords: *data visualization, Python, matplotlib*

INTRODUCTION

Python is becoming very popular in the field of data sciences. One of the reasons for this popularity is the great tools and libraries that Python offers to analyze and visualize data, such as Scientific Python (SciPy) and its component, matplotlib. Matplotlib is a multiplatform data visualization library and designed to work with the broader SciPy stack. It was conceived by John Hunter in 2002, originally as a patch to IPython for enabling interactive MATLAB-style plotting via gnuplot from the IPython command line. [1] Its syntax can be a bit confusing at first, but once one masters its main concepts, it is easy to draw pretty much any graph. [2]
SciPy extends the functionality of NumPy with a considerable collection of valuable algorithms, like minimization, Fourier transformation, regression, and other applied mathematical techniques. [3]
In the related study [4], an experiment includes some numerical examples by using a recent implementation of matplotlib. As the module improves, it would be possible to apply various types of numerical calculations.
This study consists in building a set of examples in Python and showing how easy it is to build the main charts in matplotlib.

METHODS

The choice of matplotlib was not random. There are some advantages that it offers, which makes it the primary method in this study. [5]
One of Matplotlib's most important features is its ability to play well with many operating systems and graphics backends. Matplotlib supports dozens of backends and output types, which means you can count on it to work regardless of which operating system you are using or which output format you wish. This cross-platform, everything-to-everyone approach has been one of the great strengths of Matplotlib. [1]
Another feature of Matplotlib is the ability to save figures in a wide variety of formats. You can save a figure using the savefig( ) command.[1]
Besides matplotlib, this study uses the libraries NumPy and pandas.

- NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more. [6]
- Python with pandas is in use in a wide variety of academic and commercial domains, including Finance, Neuroscience, Economics, Statistics, Advertising, Web Analytics, and more. [7] Pandas provides a host of useful tools, methods, and functionality on top of the basic data structures, but nearly everything that follows will require an understanding of these structures. Pandas provides a solid foundation upon which a very powerful data analysis ecosystem can be established.[8]

Table 1 shows a list of functions in matplotlib, which implements different types of charts.

Chart type	**matplotlib function**
Bar Chart	`bar(x, height , width, bar)`
Line Chart	`plot (x, y)`
Pie Chart	`pie(data, explode, labels, colors, autopct, shadow)`
Histogram	`hist(x, bins, facecolor, alpha)`
Scatter Plot	`scatter(x, y, s, c, marker, alpha, linewidths)`
Boxplot	`boxplot(data, notch, vert, widths)`
Density Chart	`plot(kind='density') in pandas.DataFrame.plot()`
Donut Chart	`pie(data) and gca().add_artist(circle)`
Bubble Chart	`scatter(x, y, size, color, alpha=0.5)`
Stacked area Chart	`stackplot(x,y1, y2, y3, labels)`
Filled polygons	`fill(x, y, color)`
Contures	`contour(x, y, z, levels)`

Spectrogram	`specgram(x, NFFT, Fs, Fc, sides, scale_by_freq, mode, scale, data)`
Violin Plot	`violinplot(dataset,p,v,w,showmeans,showextrema,medians, data)`
Polar Plot	`polar(theta, r)`
3D Chart	`axes(projection='3d')`

Table 1: List of functions for building charts with matplotlib

This study is focused on building a chart for each of these popular types:

- Line Chart,
- Scatter Plot,
- Histogram,
- Pie Chart,
- Bubble Chart,
- Stacked area Chart.

For each chart build in this study, in focus is the data visualizations and the level of the difficulty in building the chart using matplotlib.

RESULTS AND DISCUSSIONS

The study consists in building different graphical data visualizations using matplotlib. Data consists of the sales in one week of an item.

Line Chart

The simplest of all graphical representations of some data is the line chart. It displays the evolution of one or several numeric variables. In this example, the variables are the number of items sold on each of the days of the week. The Python code below builds a line chart using matplotlib.

```
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
item_sold=[2, 14, 50, 25, 10, 55, 68]
days=["Mon","Tue","Wed","Thu","Fri","Sat","Sun"]
plt.plot( days,item_sold, marker='*')
plt.xlabel("Days")
plt.ylabel("Items sold")
plt.show()
```

Figure 1 shows the data visualization of the items sold in one week. It represents the curve of sales during the week.

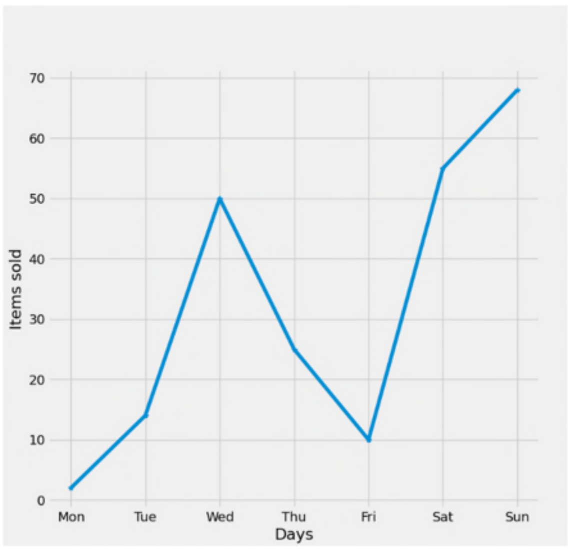

Figure 1: Line Chart

The Line Chart is built using the function `plot (x, y, marker)`. The difficulty level in building this chart is very low.

Scatter Plot

Another popular chart type is Scatter Plot. It displays the relationship between two variables, where each data is represented by a circle. In this example, one variable is the number of days we collected data, and the second variable is the number of items sold in a store. Figure 2 shows the data visualization in a Scatter Plot graphic.

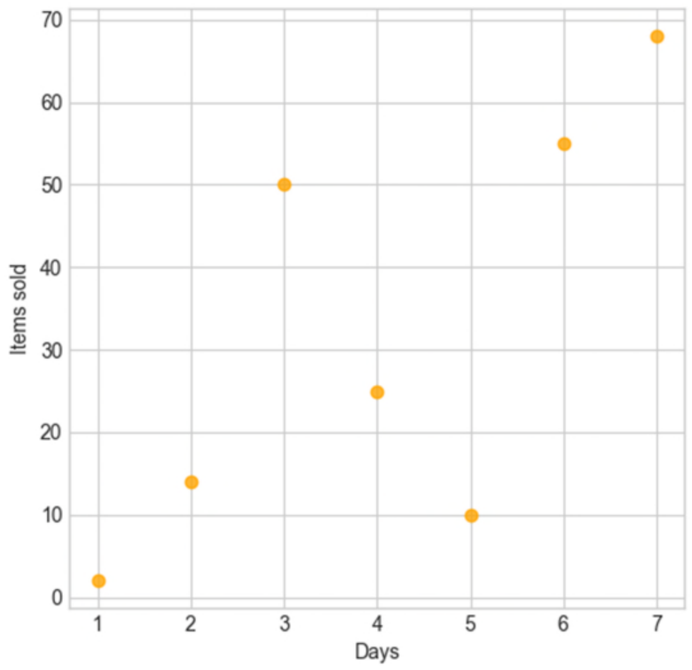

Figure 2: Scatter Plot example

The matplotlib function is scatter ( ) and the graphic is implemented in this line of code:

```
plt.scatter(range(1, 8), item_sold, c="orange", alpha=0.8)
```

The difficulty level in building this chart is very low.

Histogram

A simple histogram can be the first step in understanding a dataset. In this example, the Histogram shows the frequency of sales. The function hist ( ) requires the values of bins in which the frequency is calculated.

```
item_sold=[2,8,50,25,88,57,91]
bins=[0,10,20,30,40,50,60,70,80,90,99]
plt.hist(item_sold, bins )
```

Figure 3 shows the data visualization in Histogram. Items sold are grouped in the frequency of sales per one day.

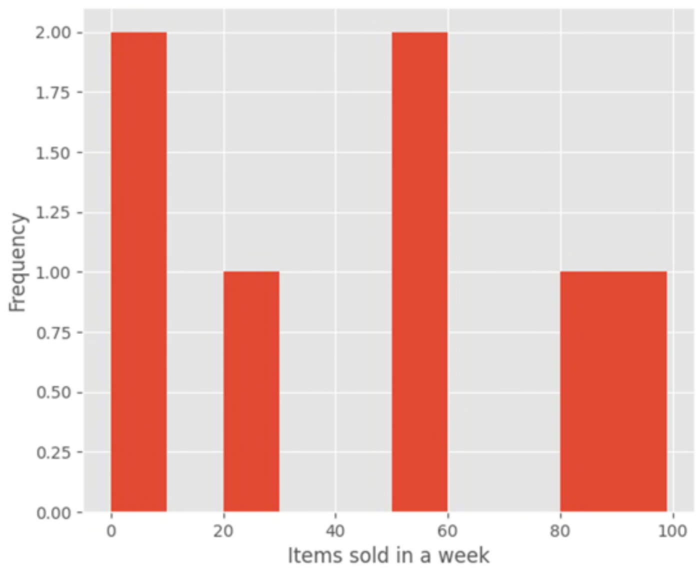

Figure 3 Histogram example

The difficulty level in building this chart is low. matplotlib offers many variables in the customization of the layout, such as colors, grids, and so on.

The Pie chart

A Pie Chart is a circular statistical plot that displays only a variable. The area of the chart is the total percentage of the given data. Building the Pie Chart in maltplotlib, the function pie ( ) requires these variables:

- labels, in this example, the days of the week,
- auto-labeling the percentage,
- offsetting a slice with "explode", in this example, the sales of Sunday,
- drop-shadow, not applied in this example,
- the explode effect, represented by an array of settings,
- custom start angle, in this example 90.

The code in Python that will build a Pie Chart is :

```
plt.pie(item_sold, explode=explode, labels = days, autopct='%1.1f%%', startangle
= 90)
```

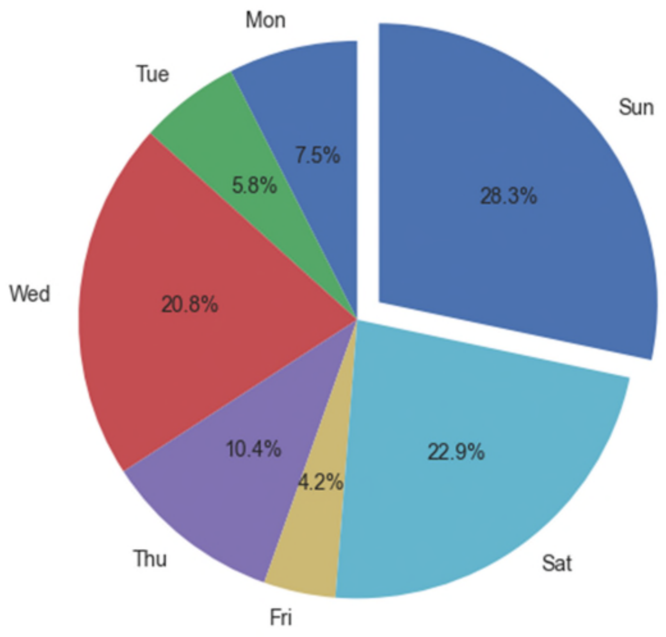

Figure 4: Pie Chart example

The difficulty level in building this chart is medium. The parameters include the labels and the explode array; in this case, the Sunday sales are with the "explode" effect and the start angle.

Bubble Chart

A Bubble Chart is similar to the Scatter Plot, but with the difference of extra variables, such as the size of the plot and color of the bubble. In this example, the bubble plot displays data in four variables:

- Item ID, represented in the x-axis.
- Total of items sold in one week, represented in the y-axis.
- The price for each item is visualized as the size of the circle.
- Category, encoded as the color of the circle.

The code implemented in this example is:

```
item_id=range(1, 8)
item_sold=[72,18,50,25,88,57,91]
prices=[500.99, 299.99, 500.99, 100.50, 100.50, 399.99, 99.99]
category=["blue", "pink", "yellow","green","red", "orange", "cyan"]
plt.scatter(item_id, item_sold, s=prices, c=category, alpha=0.8)
```

Figure 5 is the data visualization in Bubble Chart of the item id, item sold, category, and prices for each item.

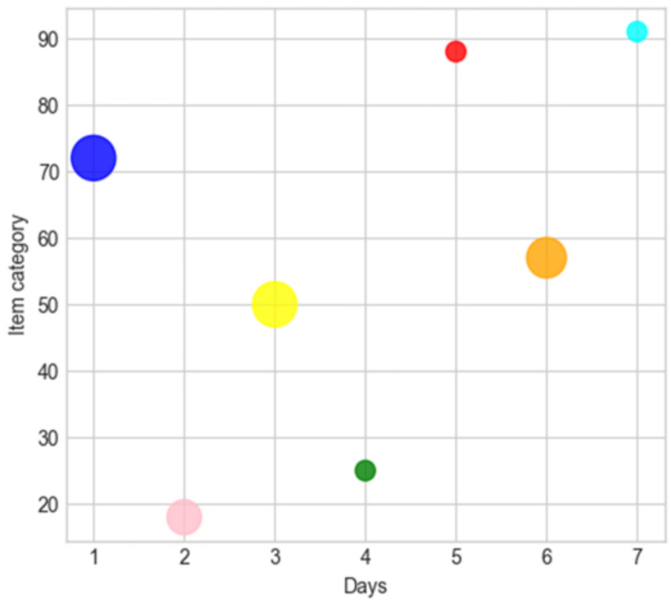

Figure 5: Bubble Plot example

The difficulty level is medium. The parameters of the function scatter ( ) show a different variable in the chart.

Stacked area chart
The Stacked area chart shows the complete datasets in one visualization. It shows each part stacked onto one another and how each dataset makes the complete figure.
In this example, the x variable is the item id; the y variable is consists of the sales in three months.
The code that implements this data visualization is :

```
itemsid=['S233', 'D333', 'E414', 'F124']
itemSoldJan=[14, 27, 50, 25]
itemSoldFeb=[13, 11, 40, 55]
itemSoldMar=[16, 29, 36, 17]
plt.stackplot(itemsid, itemSoldJan, itemSoldFeb, itemSoldMar,
labels=['January','February','March'])
```

Figure 6 shows the data visualization example in Scatter plot Area type.

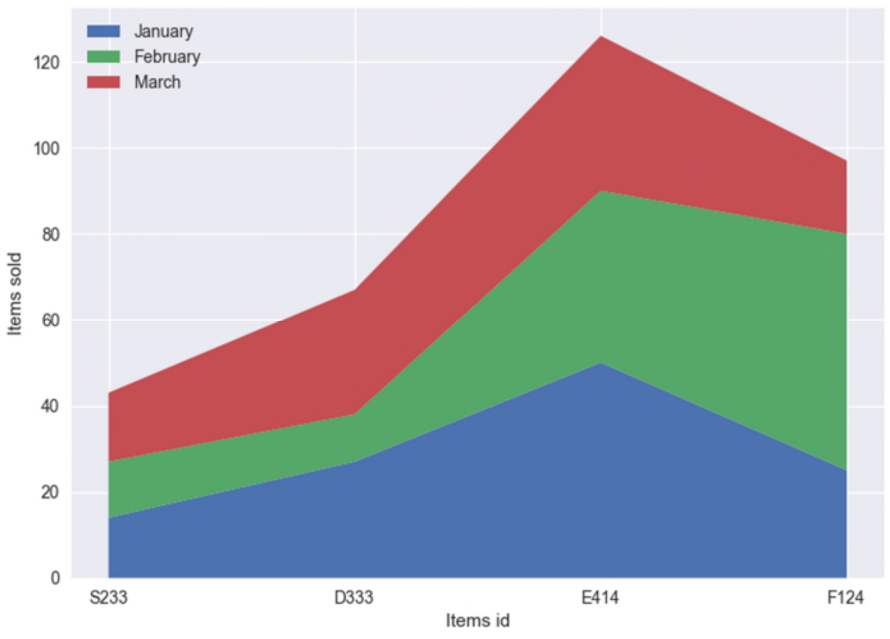

Figure 6: Scatter Plot Area example

The difficulty level in building this chart type is medium. In the case of the Scatter plot Area, the chart should also include the legend, which shows the labels for each area.

CONCLUSIONS

This study consists of building data visualization with matplotlib. We can conclude that:

- matplotlib supports various types of graphical representations like Line Chart, Scatter Plot, Histogram, Pie Chart, and other more complex data visualizations such as 3D, Polar plot, and Spectrogram.
- Building the charts in matplotlib is quite easy. Generally, the parameters of each function are the dataset and the specific labels of the chart.
- matplotlib comes with lots of different functions, which allows the user to build highly customized, elegant, and interactive plots.

REFERENCES

1) Vander P. J. 2016. Python Data Science Handbook - Essential Tools for Working with Data, O'Reilly.
2) I. Stančin and A. Jović, "An overview and comparison of free Python libraries for data mining and big data analysis," 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2019, pp. 977-982, DOI: 10.23919/MIPRO.2019.8757088.
3) J. Ranjani, A. Sheela and K. P. Meena, "Combination of NumPy, SciPy, and Matplotlib/Pylab -a good alternative methodology to MATLAB - A Comparative analysis," 2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT), 2019, pp. 1-5, DOI: 10.1109/ICIICT1.2019.8741475.
4) N. Ari and M. Ustazhanov, "Matplotlib in python," 2014 11th International Conference on Electronics, Computer and Computation (ICECCO), 2014, pp. 1-6, DOI: 10.1109/ICECCO.2014.6997585.
5) Matplotlib official website, last accessed on 16 August 2021. https://matplotlib.org/
6) NumPy official website, last accessed on 16 August 2021. https://numpy.org/
7) Pandas official website, last accessed on 20 August 2021. https://pandas.pydata.org/
8) W. Mckinney, pandas: a Foundational Python Library for Data Analysis and Statistics. Python High-Performance Science Computer, 2011.