Hyperiondev

TASK

Data Visualisation V

Visit our website

Introduction

WELCOME TO THE DATA VISUALISATION V TASK!

Now that you are familiar with some of the basic concepts and techniques associated with creating data visualisation with Python, we will begin to explore creating some more advanced visualisations. In this task, you will create 3D visualisations with Matplotlib and be exposed to some other libraries that can be used for data visualisation.

Get in touch

Connect for support

Remember that with our courses, you're not alone! You can contact your mentor to get support on any aspect of your course.

The best way to get help is to login to **www.hyperiondev.com/portal** to start a chat with your mentor. You can also schedule a call or get support via email.

Your mentor is happy to offer you support that is tailored to your individual career or education needs. Do not hesitate to ask a question or for additional support!

STACKED AREA CHART EXAMPLE

Below is an example of a stacked area chart. These graphs are often used to compare multiple variables that change over time.

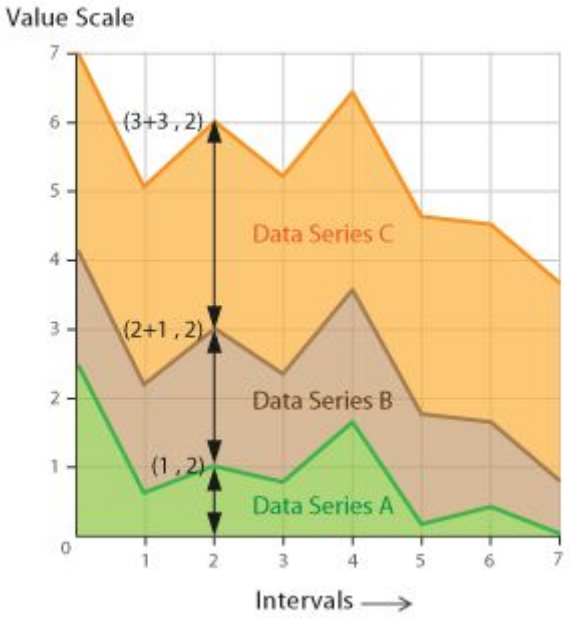


Image source: **https://datavizcatalogue.com/methods/stacked_area_graph.html**

Stacked area graphs can be easily created with Matplotlib as shown in the example below:

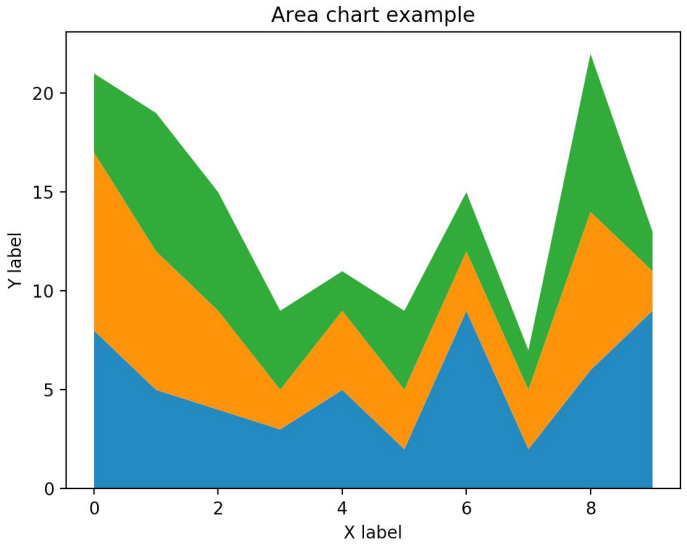
```
import numpy as np
import matplotlib.pyplot as plt

#Generate random data
groupOne = np.random.randint(1,10,10)
groupTwo = np.random.randint(1,10,10)
groupThree = np.random.randint(1,10,10)

#Creating the stacked area
y = np.row_stack((groupOne,groupTwo,groupThree))
x = np.arange(10)

y1, y2, y3=(groupOne,groupTwo,groupThree)

fig,ax = plt.subplots()
ax.stackplot(x,y)

#Labels
```

```
plt.title("Stacked area chart example")
plt.xlabel("X label")
plt.ylabel("Y label")
plt.show()
```

In the code above we have created three arrays (groupOne, groupTwo and groupThree) that contain the data that we want to represent on our stacked area graph. The instruction, `y = np.row_stack((groupOne,groupTwo,groupThree))`, takes a sequence of arrays and stack them vertically to make a single array. The code, `x = np.arange(10)`, returns an array with evenly spaced elements as per the interval.

With a stacked area graph we want to be able to view different data together in the same graph. To do this we use subplots. A subplot is defined as, "groups of smaller axes that can exist together within a single figure. These subplots might be insets, grids of plots, or other more complicated layouts (VanderPlas, 2016)." Notice that we use the code, `plt.subplots()`. This function returns a figure, which is the top level container for all the plot elements, and an array of Axes objects. The code, `ax.stackplot(x,y)`, creates a stackplot. Stackplots are generated by plotting different datasets vertically on top of one another rather than overlapping with one another.

When you run the program you will get something like this:



3D SCATTERPLOT EXAMPLE

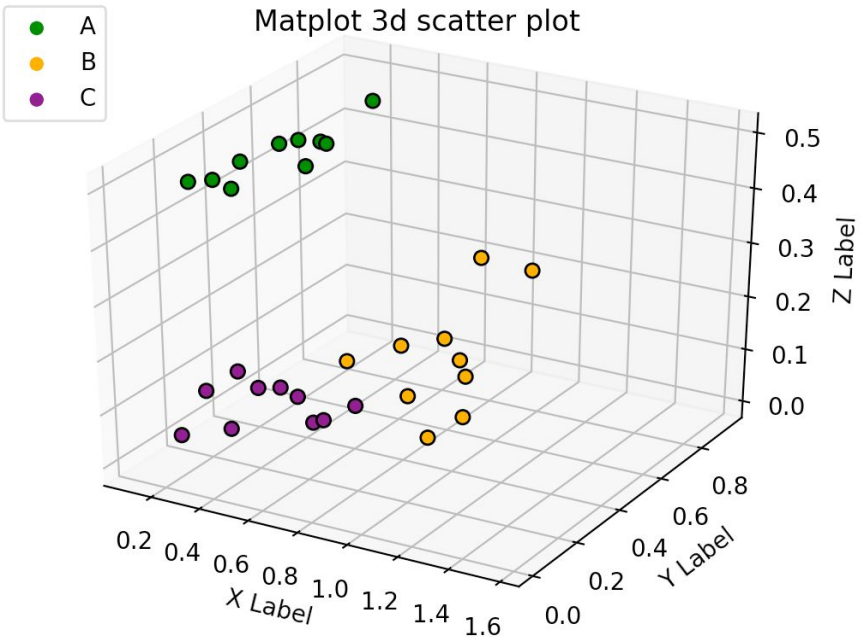
3D scatterplots are useful when one wants to compare 3 characteristics of a data set instead of two. Again, we can use Matplotlib to create a scatterplot as shown below:

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

#Generate random data
n = 10
g1 = (0.1+0.2 * np.random.rand(n),
np.random.rand(n),0.4+0.1*np.random.rand(n))
g2=(0.7+0.9*np.random.rand(n),0.2*np.random.rand(n),0.51*np.random.rand(n))
g3=(0.6 * np.random.rand(n),
0.7*np.random.rand(n),0.4+0.1*np.random.rand(n))

#Create visualisation
data= (g1,g2,g3)
colours = ("green", "orange", "purple")
groups = ("A", "B", "C")

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax = fig.gca(projection="3d")

for data, colour, group in zip(data, colours, groups):
     x, y, z = data
     ax.scatter(x, y, z, alpha = 1, c = colour, edgecolors = "none", s = 35,
label = group)

ax.set_xlabel("X label")
ax.set_ylabel("Y label")
ax.set_zlabel("Z label")

plt.title("3D scatter plot example")
plt.legend(loc = 2)
plt.show()
```

From the code above, notice the following:

- To create 3D visualisations with Matplotlib add the following import statement: `from mpl_toolkits.mplot3d import Axes3D`
- Once you have added this import statement, the option 'projection' becomes available. By default, Matplotlib produces 2D visualisations. Once you have imported mplot3D, you can specify that `projection="3d"`.
- The code, `ax = fig.gca(projection="3d")` gets the current 3D axes on the figure or creates and returns the appropriate axes if they don't already exist.
- `ax.scatter()` creates a scatterplot.

When you run the program you will get something like this:



DATA VISUALISATION USING OTHER LIBRARIES

There are many other libraries out there that can help you create all kinds of data visualisation.

For example, if you want to create a *word cloud* visualisation you can install the **wordcloud** library (`pip3 install wordcloud`). The code below demonstrates how to use this library:

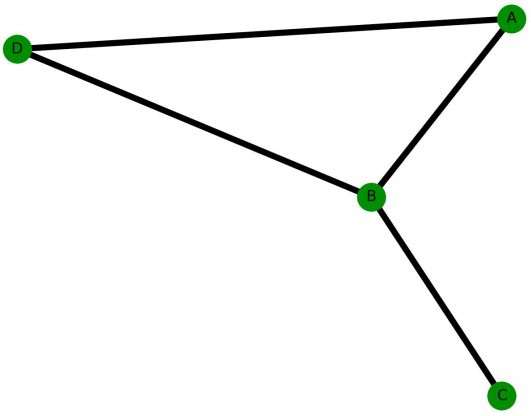
```
import matplotlib.pyplot as plt
from wordcloud import WordCloud

#Create text
text = ("Python Data Visualisation HyperionDev Learning Data Science Mentors
Careers Analytics")

#Create visualisation
wordcloud = wordCloud(width = 500, height = 500, margin = 10).generate(text)
plt.imshow(wordcloud, interpolation = "bilinear")
plt.axis("off")
plt.margins(x = 0, y = 0)
plt.show()
```

When you run the program you will get something like this:

A library called **networkx** allows one to create network graphs. Download and install this library here: **https://networkx.github.io/**. The code below demonstrates how to use this library:

```
import pandas as pd
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt

#Create a dataframe
df = pd.DataFrame({'from': ['C', 'A', 'B', 'D'], 'to': ['B', 'D', 'A',
'B']})

#Create the network graph
G = nx.from_pandas_edgelist(df, 'from', 'to')

#Customize the nodes
nx.draw(G, with_labels=True, node_color='green', node_size=500, width=5)

plt.show()
```

When you run the program you will get something like this:



A note from our coding mentor

Sarah

*You can find plenty of python packages related to data visualisation with a quick search. A common package that I like to use is **ggplot**. For geographic or map visualisations I use **geoplotlib**.*

There are always new packages that are created and old ones being updated, make sure you keep track of the latest versions! Sometimes old tutorials may use syntax that does not work for the new package versions.

Instructions

Before you get started we strongly suggest you start using Notepad++ or IDLE to open all text files (.txt) and python files (.py). Do not use the normal Windows notepad as it will be much harder to read.

Compulsory Task 1

Follow these steps:

- Create a new jupyter notebook in this folder named **wine.ipynb**.
- Read in the **wine.csv** file.
- Do a short Exploratory Data Analysis (EDA) on the dataset to give an idea of the kind of data we are dealing with.
- There are some missing values in the data. Would you undertake imputation? If yes How? Please outline this in the notebook.
- Do you think the missing values greatly affect your EDA?
- Create Word Clouds of the Province and the variety of the wine.

NOTE: Be Creative and Imaginative.

: Your Notebook should be neat and easy to follow

Completed the task(s)?

Ask your mentor to review your work!

Review work

Things to look out for:

1. Make sure that you have correctly installed any visualisation libraries that you want to use before using them in your code.

Rate us
Share your thoughts

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved or think we've done a good job?

Click here to share your thoughts anonymously.

References:

VanderPlas, J. (2016, November). *Multiple Subplots.* Retrieved from Python Data Science Handbook: **https://jakevdp.github.io/PythonDataScienceHandbook/04.08-multiple-subplots.html**