TASK

Data Visualisation - Python Libraries

Visit our website

Introduction

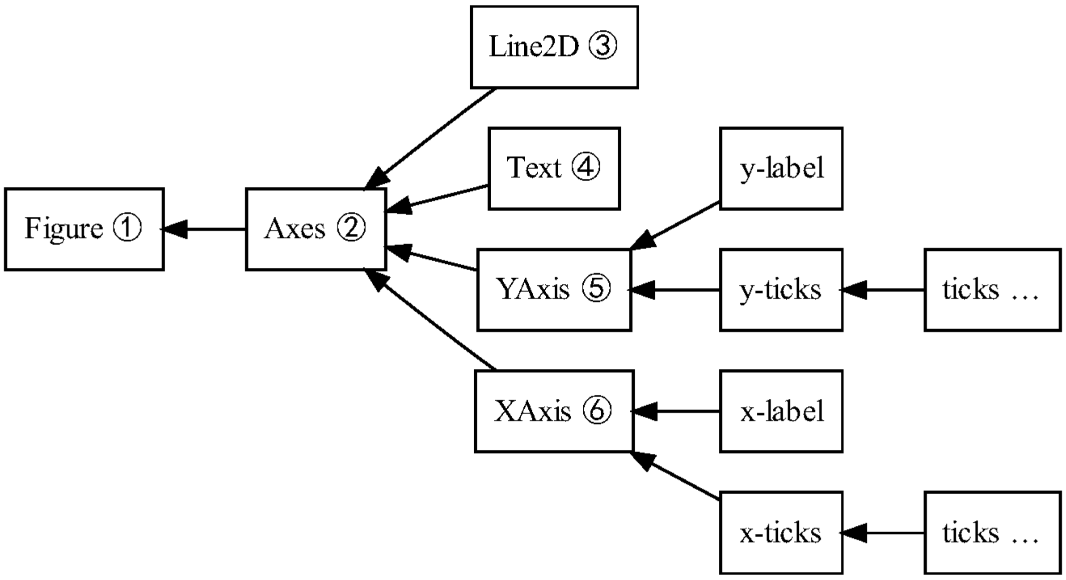
WELCOME TO THE DATA VISUALISATION - PYTHON LIBRARIES TASK!

In this task, you will learn more about matplotlib and seaborn, and practice using them to visualise data.

MATPLOTLIB

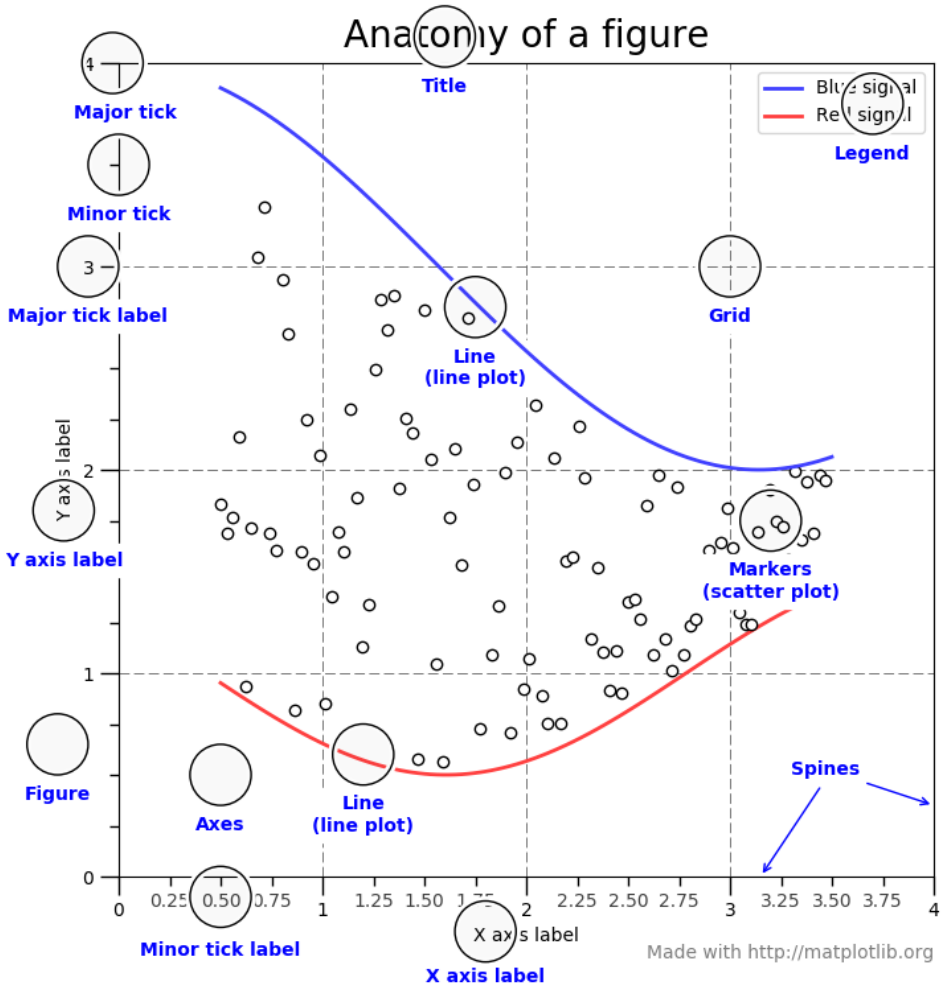
Recall **matplotlib** is a powerful visualisation tool to create both exploratory and deployment visualisations. In this task, you will learn about how matplotlib is structured and how to create a variety of plots.

Matplotlib is organised into a hierarchy. At the top of the hierarchy, there is a library called pyplot. We create a pyplot to create a figure. The figure keeps track of all the child axes, artists (titles, figure legends, etc.), and the canvas.



(Hunter, 2007)

Artists are defined as follows: “Basically everything you can see on the figure is an artist (even the Figure, Axes, and Axis objects). This includes Text objects, Line2D objects, collection objects, Patch objects … (you get the idea). When the figure is rendered, all of the artists are drawn to the canvas. Most Artists are tied to an Axes; such an Artist cannot be shared by multiple Axes or moved from one to another” (Hunter, 2007).



(Hunter, 2007)

All of the plotting functions expect `np.array` or `np.ma.masked_array` as input, so it is best to convert any pandas (or similar array-like data structures) to arrays before using them with matplotlib.

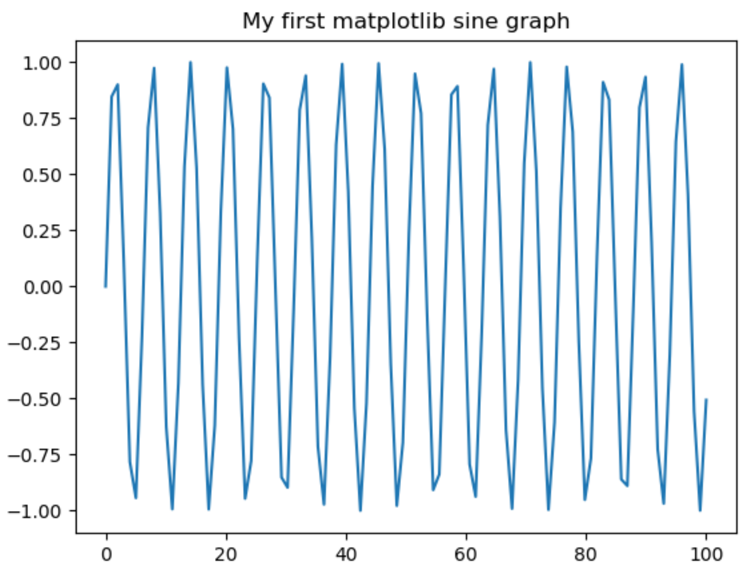
Remember to import packages or libraries at the top of your Python file:

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

We can create a quick dataset using Numpy and visualise the data using matplotlib:

```
# Prepare the data
x = np.linspace(0, 100, 100) # x axis
y = np.sin(x) # y values

# Plot the data
plt.plot(x, y)

# Create a title
plt.title("My first matplotlib sine graph")

# Show the plot
plt.show()
```

And if you run the program, you will get something like this:



Try to play around with the sine graph code provided in an example file to get a better understanding.

Extra resource

For more information about working with matplotlib, please consult the fourth chapter ("**Visualization with Matplotlib**") in the book entitled, "**Python Data Science Handbook**" by Jake VanderPlas. You can also explore the **matplotlib webpage**.

Now that you are familiar with some of the basic concepts and techniques associated with creating data visualisations with matplotlib, we will begin to explore creating some more advanced visualisations.

Stacked Area Chart

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

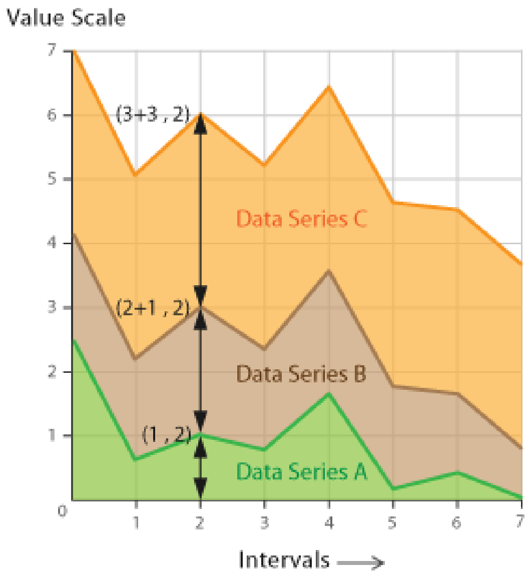


Image source: ***https://datavizcatalogue.com/methods/stacked_area_graph.html***

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

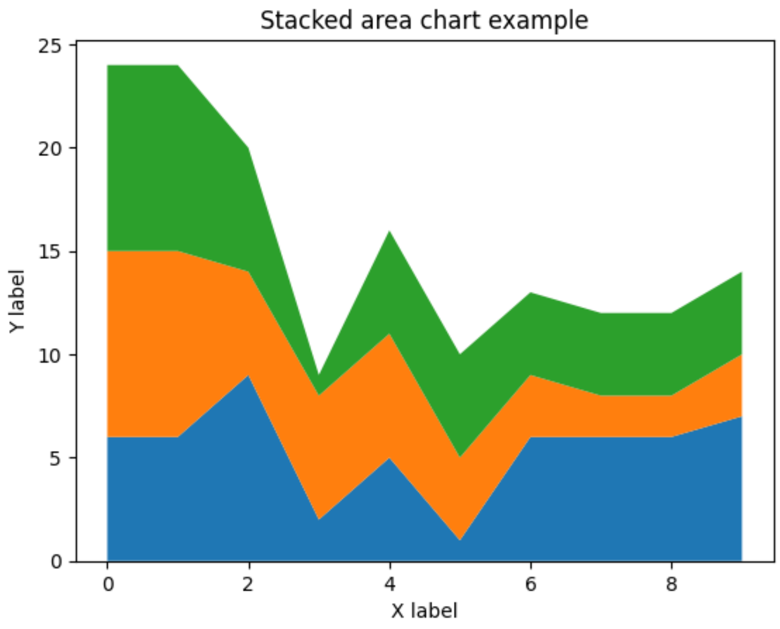
```
import numpy as np
import matplotlib.pyplot as plt

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

```
groupTwo = np.random.randint(1,10,10)
groupThree = np.random.randint(1,10,10)

# Create 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 stacks 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 follows: "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:



SEABORN

Recall seaborn is a more advanced visualisation library than matplotlib. It is well suited for statistical visualisations and is designed to work with pandas data structures. It also allows you to switch easily between different types of graphs to look at the data from different angles.

Some commonly-used Seaborn plots include:

- `histplot()`
- `barplot()`
- `boxplot()`

And there are many others! You will get accustomed to many of these methods during the course of this bootcamp.

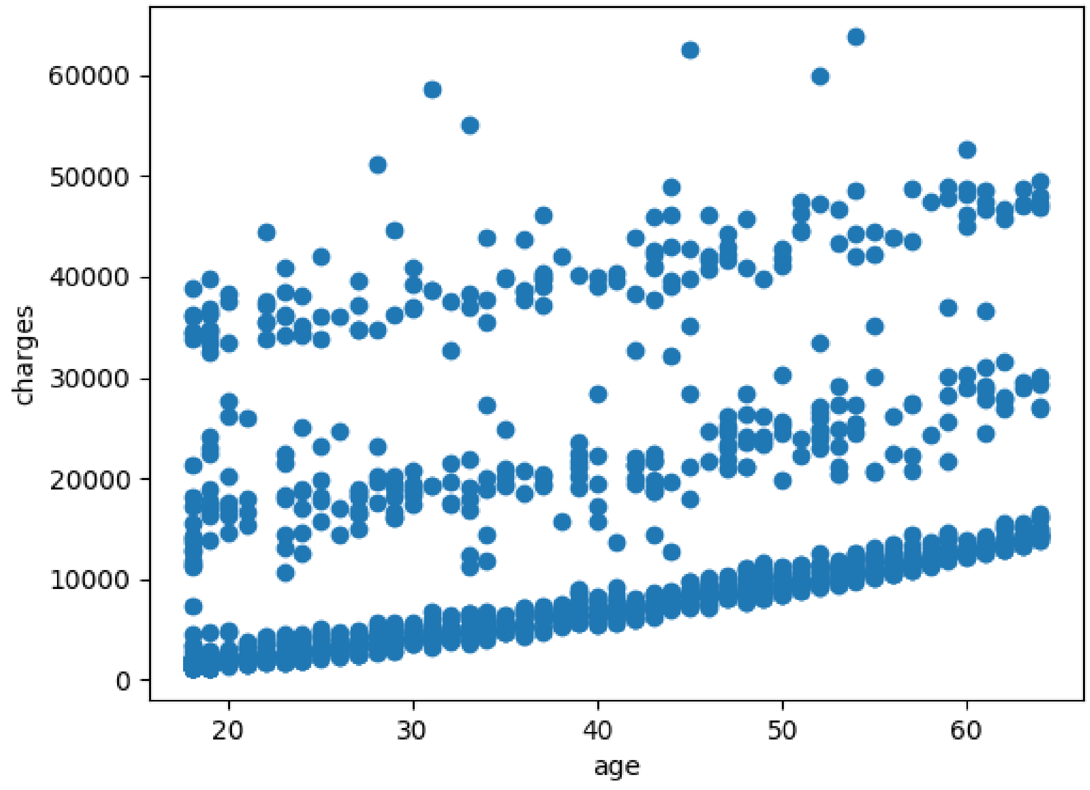
Let's say that we are reading insurance data that contains a column for age and a column for the insurance charge. We would like to understand the relationship existing between these two columns.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load data
ins_df = pd.read_csv('insurance.csv')
```

In matplotlib, the **`scatter()`** method would be most appropriate. This can be achieved as follows:

```
# Plot scatterplot
plt.figure()
plt.scatter(ins_df['age'], ins_df['charges'])
plt.xlabel("age")
plt.ylabel('charges')
plt.show()
plt.close()
```

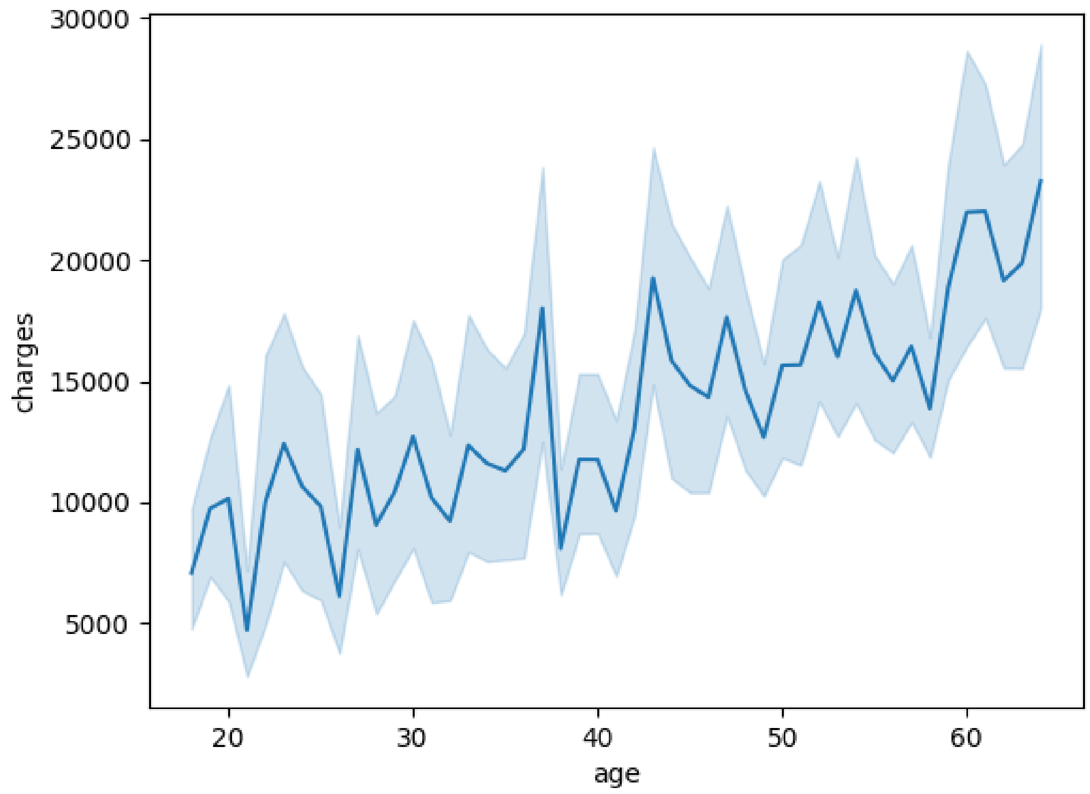
And would look something like this:



You can get a general sense of this data, but it may take time to understand the overall trend. In Seaborn, there is a **`lineplot()`** method that automatically plots averages and standard deviations for ease of reading. To do this, see the following:

```
# Plot lineplot
plt.figure()
sns.lineplot(x='age', y='charges', data=ins_df)
plt.show()
plt.savefig('sns_lineplot.png') # save as a png image file
plt.close()
```

You will end up with something that looks like this:



This makes it a lot easier to see the overall trend existing in the data: the higher your age, the larger your insurance charges.

Multi-Plot Grids

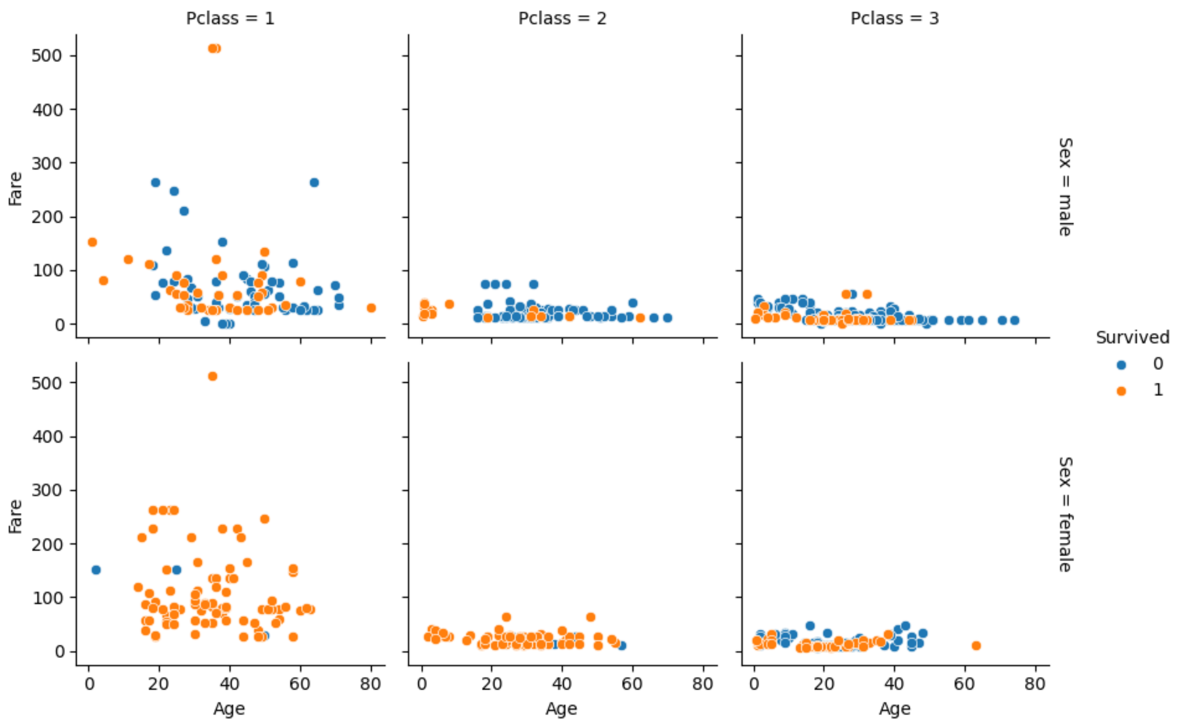
More often than not, we have to visualise relationships between four or more dimensions. As I'm sure you can imagine, thinking in four dimensions can be difficult. Five dimensions? Even more so. Imagine a dataset containing 20 columns - that's 20 dimensions that we have to visualise! Luckily, there are workarounds to this.

Seaborn contains a FacetGrid - essentially a grid containing multiple plots. While this can easily be done in matplotlib, the interface presented in Seaborn is much easier to use.

Let's use the Titanic dataset as an example here. We have multiple factors contributing to whether someone survived or not. Let's take a look at the relationship between a person's Sex, Passenger Class, Age, Fare, and whether they survived or not:

```python
import pandas as pd
import seaborn as sns

# Load data
titanic_df = pd.read_csv('Titanic.csv')

# Plot FacetGrid plot
plt.figure()
fg = sns.FacetGrid(titanic_df, row="Sex", col="Pclass", hue="Survived",
margin_titles=True)
fg.map(sns.scatterplot, "Age", "Fare")
fg.add_legend()
plt.show()
plt.close()
```

This produces the following diagram:



This tells us a lot. The first (and most obvious) observation is that Pclass 1 has the most expensive fare. By comparing the number of orange and blue dots in the first and second row of graphs we see that mostly females survived, especially in Pclass 1 and 2.

Correlation Heatmaps

A correlation matrix is a very important tool to have as a data scientist. This is quite possibly the best way to get a basic understanding of all of the features in a single plot.

Before we talk about a correlation matrix, let's talk about correlation, and more specifically, the correlation coefficient. In short, the correlation coefficient is a single number that tells you the relationship between two variables. If one variable increases consistently as another variable increases, this means that they have a positive correlation. If one variable decreases as another increases, this is a negative correlation. If two variables have no effect on each other, this means that they have a zero correlation.

The correlation coefficient has a value between -1 and 1. A correlation of 1 is basically a straight line going upwards from left to right. A correlation of -1 is a straight line going down from left to right.

Now that we understand correlation, what is a correlation matrix? It is essentially a 2D square matrix (which means that there are as many rows as there are columns). The value at position (0, 1) in the correlation matrix will show you the correlation coefficient between features 0 and 1.

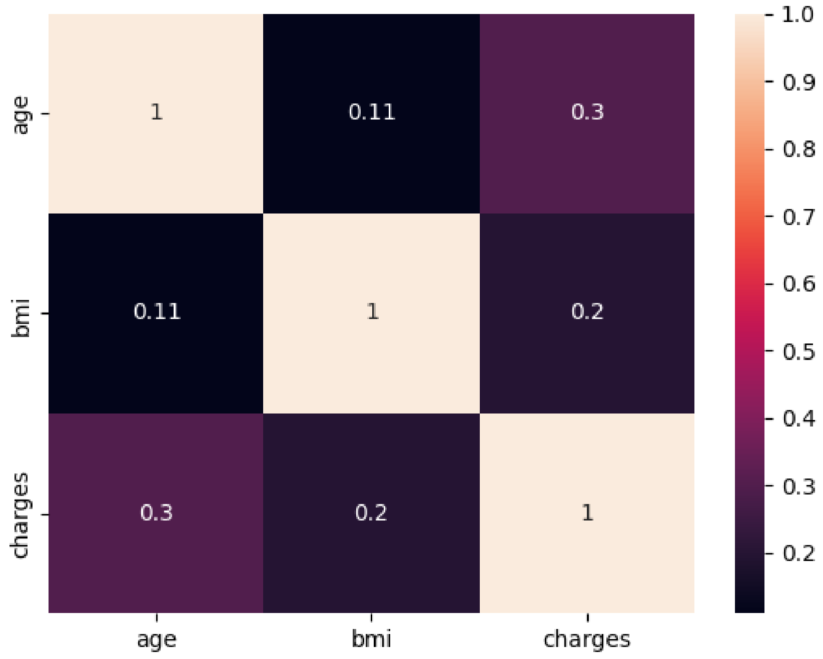
The correlation matrix can be viewed as a set of numbers, but numbers have never been very easy to read. The best way to view this matrix is with a heatmap. Thankfully, seaborn (as always) makes this easy.

Let's take a look at the insurance dataset. Specifically, we want to see the relationship between age, BMI, and insurance charges:

```python
import pandas as pd
import seaborn as sns

# Load data
insurance_df = pd.read_csv('insurance.csv')

# Select columns of interest
age_bmi_charges = insurance_df[['age', 'bmi', 'charges']]
```

```
# Plot correlation matrix
plt.figure()
corr_coeff_mat = age_bmi_charges.corr()
sns.heatmap(corr_coeff_mat, annot=True)
plt.show()
plt.close()
```

This produces the following graph:



What can we tell from this graph? Well, keep in mind that a correlation close to zero means that there's little to be seen in terms of a relationship and values closer to one indicate a strong relationship. The range of values from zero to one is assigned different shades on the heatmap. A more purple colour indicates a weak relationship and red to beige indicates a stronger relationship. In this example, the relationships between features are weak as the colour blocks showing relationships between different features are all shades of purple. Nevertheless, there are some observations that are interesting. For example, being older has a slightly higher impact on insurance charges than your BMI.

Extra resource

You can find plenty of Python packages related to data visualisation with a quick search. For geographic or map visualisations you can use **geoplotlib**. Or you can create a **wordcloud** for fun.

Instructions

First, read and run the **example files** provided. Feel free to write and run your own example code before doing the compulsory task to become more comfortable with the concepts covered in this task.

Compulsory Task 1

Follow these steps:

- Open the Jupyter notebook named **data_viz_task.ipynb**.
- Generate the following graphs from the `Cars93.csv` dataset. Then, answer the accompanying questions in the markdown cells in the notebook:
    - A box plot for the revs per mile for the Audi, Hyundai, Suzuki, and Toyota car manufacturers. Which of these manufacturers has the car with the highest revs per mile?
    - A histogram of MPG in the city. On the same axis, show a histogram of MPG on the highway. Is it generally more fuel efficient to drive in the city or on the highway?
    - A lineplot showing the relationship between the 'Wheelbase' and 'turning circle'. What is this relationship? What happens when the wheelbase gets larger?
    - A bar plot showing the mean horsepower for each car Type (Small, Midsize, etc.). Does a larger car mean more horsepower?

Compulsory Task 2

Follow these steps:

- Create a new Jupyter notebook in this task's folder and name it **wine.ipynb**.
- Read in the **wine.csv** file.
- Create a multi-plot grid:

- Filter the dataset to only contain “Cabernet Sauvignon”, “Pinot Noir”, and “Chardonnay” wines.
- Use the variety column.
- For each of these three wine varieties, show a histogram plot of the “points” column.

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

Hunter, J.D. (2007). Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95.

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