WEEK-02: DATA ANALYSIS_AND_VISUALISATION_WITH_PYTHON

Matplotlib

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

Basic plots in Matplotlib :

Matplotlib comes with a wide variety of plots. Plots helps to understand trends, patterns, and to make correlations. They're typically instruments for reasoning about quantitative information. Some of the sample plots are covered here.

Line plot :

Example 1

```
# importing matplotlib module
from matplotlib import pyplot as plt
# x-axis values
x = [5, 2, 9, 4, 7]
# Y-axis values
y = [10, 5, 8, 4, 2]
# Function to plot
plt.plot(x,y)
# function to show the plot
plt.show()
```

Bar plot:

Example 2

```
# importing matplotlib module
from matplotlib import pyplot as plt
# x-axis values
x = [5, 2, 9, 4, 7]
# Y-axis values
y = [10, 5, 8, 4, 2]
# Function to plot the bar
plt.bar(x,y)
# function to show the plot
plt.show()
```

Hist plot:

A histogram is basically used to represent data provided in a form of some groups.It is accurate method for the graphical representation of numerical data distribution.It is a type of bar plot where X-axis represents the bin ranges while Y-axis gives information about frequency.

Creating a Histogram

To create a histogram the first step is to create bin of the ranges, then distribute the whole range of the values into a series of intervals, and count the values which fall into each of the intervals.Bins are clearly identified as consecutive, non-overlapping intervals of variables.The matplotlib.pyplot.hist() function is used to compute and create histogram of x.

In Matplotlib, we use the hist() function to create histograms. The hist() function will use an array of numbers to create a histogram, the array is sent into the function as an argument. For simplicity we use NumPy to

randomly generate an array with 250 values, where the values will concentrate around 170, and the standard deviation is 10.

You can read from the histogram that there are approximately:

2 people from 140 to 145cm
5 people from 145 to 150cm
15 people from 151 to 156cm
31 people from 157 to 162cm
46 people from 163 to 168cm
53 people from 168 to 173cm
45 people from 173 to 178cm
28 people from 179 to 184cm
21 people from 185 to 190cm
4 people from 190 to 195cm

Example: Say you ask for the height of 250 people, you might end up with a histogram like this:

The hist() function will read the array and produce a histogram:

Simple histogram:

Example 3

```
import matplotlib.pyplot as plt
import numpy as np
x = np.random.normal(170, 10, 250)
print(x)
plt.hist(x)
plt.show()
```

Ref: https://www.w3schools.com/python/matplotlib_histograms.asp

Example 4

```
# importing matplotlib module
from matplotlib import pyplot as plt
# Y-axis values
y = [10, 5, 8, 4, 2]
# Function to plot histogram
plt.hist(y)
# Function to show the plot
plt.show()
```

Ref: https://www.geeksforgeeks.org/plotting-histogram-in-python-using-matplotlib/

Scatter plot:

Example 5

```
# importing matplotlib module
from matplotlib import pyplot as plt
# x-axis values
x = [5, 2, 9, 4, 7]
# Y-axis values
y = [10, 5, 8, 4, 2]
# Function to plot scatter
plt.scatter(x, y)
# function to show the plot
plt.show()
```

SciPy

What is SciPy?

- SciPy is a scientific computation library that uses NumPy underneath.
- SciPy stands for Scientific Python.
- It provides more utility functions for optimization, stats and signal processing.
- Like NumPy, SciPy is open source so we can use it freely.
- SciPy was created by NumPy's creator Travis Olliphant.

Why Use SciPy?

- If SciPy uses NumPy underneath, why can we not just use NumPy?
- SciPy has optimized and added functions that are frequently used in NumPy and Data Science.

Which Language is SciPy Written in?

- SciPy is predominantly written in Python, but a few segments are written in C.

Import SciPy

- Once SciPy is installed, import the SciPy module(s) you want to use in your applications by adding the from scipy import module statement:

  from scipy import constants

constants: SciPy offers a set of mathematical constants, one of them is liter which returns 1 liter as cubic meters.

Unit Categories

- The units are placed under these categories:
- Metric
- Binary
- Mass
- Angle
- Time
- Length
- Pressure
- Volume
- Speed
- Temperature
- Energy
- Power
- Force

Example 6

```
from scipy import constants
print(constants.peta)    #1000000000000000.0
print(constants.tera)    #1000000000000.0
print(constants.giga)    #1000000000.0
print(constants.mega)    #1000000.0
print(constants.kilo)    #1000.0
print(constants.hecto)   #100.0
print(constants.deka)    #10.0
print(constants.deci)    #0.1
print(constants.centi)   #0.01
print(constants.milli)   #0.001
print(constants.micro)   #1e-06
print(constants.nano)    #1e-09
print(constants.pico)    #1e-12
```

Sparse Data

- Sparse data is data that has mostly unused elements (elements that don't carry any information).
- It can be an array like this one: [1, 0, 2, 0, 0, 3, 0, 0, 0, 0, 0, 0]
- **Sparse Data:** is a data set where most of the item values are zero.
- **Dense Array:** is the opposite of a sparse array: most of the values are *not* zero.
- In scientific computing, when we are dealing with partial derivatives in linear algebra we will come across sparse data.

Work With Sparse Data

SciPy has a module, scipy.sparse that provides functions to deal with sparse data.

- There are primarily two types of sparse matrices that we use:
- CSC - Compressed Sparse Column. For efficient arithmetic, fast column slicing.
- CSR - Compressed Sparse Row. For fast row slicing, faster matrix vector products We will use the CSR matrix in this tutorial.

CSR Matrix

We can create CSR matrix by passing an arrray into function scipy.sparse.csr_matrix().

Example 7

Create a CSR matrix from an array:

```
import numpy as np
from scipy.sparse import csr_matrix
arr = np.array([0, 0, 0, 0, 0, 1, 1, 0, 2])
print(csr_matrix(arr))
```

note

The example above returns:

```
  (0, 5) 1
  (0, 6) 1
  (0, 8) 2
```

From the result we can see that there are 3 items with value.
The 1. item is in row 0 position 5 and has the value 1.
The 2. item is in row 0 position 6 and has the value 1.
The 3. item is in row 0 position 8 and has the value 2.

Viewing stored data (not the zero items) with the data property:

Example 8

```
import numpy as np
from scipy.sparse import csr_matrix
arr = np.array([[0, 0, 0], [0, 0, 1], [1, 0, 2]])
print(csr_matrix(arr).data)
```

Converting from csr to csc with the tocsc() method:

Example 9

```
import numpy as np
from scipy.sparse import csr_matrix
arr = np.array([[0, 0, 0], [0, 0, 1], [1, 0, 2]])
newarr = csr_matrix(arr).tocsc()
print(newarr)
```

Pandas

Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

Use of Pandas

Pandas allows us to analyze big data and make conclusions based on statistical theories. Pandas can clean messy data sets, and make them readable and relevant. Relevant data is very important in data science.

Data Science: is a branch of computer science where we study how to store, use and analyze data for deriving information from it.

Pandas gives you answers about the data. Like:

- Is there a correlation between two or more columns?
- What is average value?
- Max value?
- Min value?

Pandas are also able to delete rows that are not relevant, or contains wrong values, like empty or NULL values. This is called *cleaning* the data.

Once Pandas is installed, import it in your applications by adding the import keyword:

import pandas

Example 10

Get your own Python Server

```
import pandas
mydataset = {   'cars': ["BMW", "Volvo", "Ford"],
  'passings': [3, 7, 2] }
myvar = pandas.DataFrame(mydataset)
print(myvar)
```

Create an alias with the as keyword while importing:

import pandas as pd

Now the Pandas package can be referred to as pd instead of pandas.

Example 11

```
import pandas as pd
mydataset = {
  'cars': ["BMW", "Volvo", "Ford"],
  'passings': [3, 7, 2]
}
myvar = pd.DataFrame(mydataset)
print(myvar)
```

Indices in a pandas series:

- A pandas series is similar to a list, but differs in the fact that a series associates a label with each element. This makes it look like a dictionary.
- If an index is not explicitly provided by the user, pandas creates a RangeIndex ranging from 0 to N-1.
- Each series object also has a data type.

Example 12

```
import panadad as pd
new_series= pd.series([5,6,7,8,9,10])
print(new_series)
```

- As you may suspect by this point, a series has ways to extract all of the values in the series, as well as individual elements by index.

Example 13

```
import panadad as pd
new_series= pd.series([5,6,7,8,9,10])
print(new_series.values)
print('_____')
print(new_series[4])
```

- You can also provide an index manually.

Example 14

```
import panadad as pd
new_series= pd.series([5,6,7,8,9,10], indec=[‘a’,’b’,’c’,’d’,’e’,’f’)
print(new_series.values)
print(‘_____________’)
print(new_series[‘f’])
```

- It is easy to retrieve several elements of a series by their indices or make group assignments.

Example 15

```
import panadad as pd
new_series= pd.series([5,6,7,8,9,10], indec=[‘a’,’b’,’c’,’d’,’e’,’f’)
print(new_series)
print(‘_____________’)
print(new_series[‘a’,’b’,‘f’])=0
print(new_series)
```

- Filtering and maths operations are easy with Pandas as well.

Example 16

```
import panadad as pd
new_series= pd.series([5,6,7,8,9,10], indec=[‘a’,’b’,’c’,’d’,’e’,’f’)
new_series2= new_series[new_series>0]  # check with 7 instead of 0
print(new_series2)
print(‘_____________’)
new_series2= new_series[new_series>0]  * 2
print(new_series2)
```

DataFrame:

- A Pandas DataFrame is a 2 dimensional data structure, like a 2 dimensional array, or a table with rows and columns.
- Simplistically, a data frame is a table, with rows and columns.
- Each column in a data frame is a series object.
- Rows consist of elements inside series.

Case ID	*Variable one*	*Variable two*	*Variable 3*
1	*123*	*ABC*	*10*
2	*456*	*DEF*	*20*
3	*789*	*XYZ*	*30*

- Pandas data frames can be constructed using Python dictionaries.

Example 17

Get your own Python Server

Create a simple Pandas DataFrame:

```
import pandas as pd
data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}
#load data into a DataFrame object:
df = pd.DataFrame(data)
print(df)
```

Locate Row

As you can see from the result above, the DataFrame is like a table with rows and columns. Pandas use the loc attribute to return one or more specified row(s)

Example 18

Return row 0:

```
#refer to the row index:
print(df.loc[0])
```

Named Indexes

With the index argument, you can name your own indexes.

Example 19

Add a list of names to give each row a name:

```
import pandas as pd
data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}
df = pd.DataFrame(data, index = ["day1", "day2", "day3"])
print(df)
```

- You can also create a data frame from a list.

Example 20

```
import pandas as pd
list2=[[0,1,2],[3,4,5],[6,7,8]]
df= pd.DataFrame(list2)
print(df)
df.columns=['V1','V2','V3']
print(df)
```

Example 21

```
import pandas as pd
df=pd.DataFrame({
    'Country': ['Kazakhstan','Russia','Belarus','Ukraine'],
    'Population': [17.04,143.5,9.5,45.5]
    'Square':[2724902, 17125191,207600,603628]
   })
print(df)
```

- You can ascertain the type of a column with the type() function.

```
print(type(df['Country']))
```

- A Pandas data frame object as two indices; a column index and row index.
- Again, if you do not provide one, Pandas will create a RangeIndex from 0 to N-1.

Example 22

```
import pandas as pd
df=pd.DataFrame({
    'Country': ['Kazakhstan','Russia','Belarus','Ukraine'],
    'Population': [17.04,143.5,9.5,45.5]
    'Square':[2724902, 17125191,207600,603628]
   })
print(df.columns)
print('_____')
print(df.index)
```

- There are numerous ways to provide row indices explicitly.
- For example, you could provide an index when creating a data frame:

Example 23

```
import pandas as pd
df=pd.DataFrame({
    'Country': ['Kazakhstan','Russia','Belarus','Ukraine'],
    'Population': [17.04,143.5,9.5,45.5]
    'Square':[2724902, 17125191,207600,603628]
   }, index = ['KZ','RU','BY','UA'])
print(df)
```

- or do it during runtime.
- Here, I also named the index ‘country code’.

Example 24

```
import pandas as pd
df=pd.DataFrame({
    ‘Country’: [‘Kazakhstan’,’Russia’,’Belarus’,’Ukraine’],
    ‘Population’: [17.04,143.5,9.5,45.5]
    ‘Square’:[2724902, 17125191,207600,603628]
   })
print(df)
print(‘_____’)
df.index = =[‘KZ’,’RU’,’BY’,’UA’]
df.index.name = ‘Country Code’
print(df)
```

- Row access using index can be performed in several ways.
- First, you could use .loc() and provide an index label.

Example 25

```
print(df.loc[‘KZ’]
```

- Second, you could use .iloc() and provide an index number

```
print(df.iloc[0]
```

- A selection of particular rows and columns can be selected this way.

```
print(df.loc[[‘KZ’,’RU’],’population’])
```

- You can feed .loc() two arguments, index list and column list, slicing operation is supported as well:

```
print(df.loc[[‘KZ’,’BY’, :])
```

Filtering

- Filtering is performed using so-called Boolean arrays.

Example 26

```
print(df[df.population > 10][[‘Country’,’Square’]])
```

Deleting columns

You can delete a column using the drop() function.

```
print(df)
df = df.drop([‘Population’], axis = ‘columns’)
print(df)
```

Load Files Into a DataFrame

If your data sets are stored in a file, Pandas can load them into a DataFrame.

Example 27

Load a comma separated file (CSV file) into a DataFrame:

```
import pandas as pd
df = pd.read_csv('data.csv')
print(df)
```

Read CSV Files

A simple way to store big data sets is to use CSV files (comma separated files). CSV files contains plain text and is a well know format that can be read by everyone including Pandas. In our examples we will be using a CSV file called 'data.csv'.

Example 28

Get your own Python Server

Load the CSV into a DataFrame:

```
import pandas as pd
df = pd.read_csv('data.csv')
print(df.to_string())  # use to_string() to print the entire DataFrame.
print(df)   #Print the DataFrame without the to_string() method
```

Data Cleaning

Data cleaning means fixing bad data in your data set.

Bad data could be:

- Empty cells
- Data in wrong format
- Wrong data
- Duplicates

Plotting

Pandas uses the plot() method to create diagrams. We can use Pyplot, a submodule of the Matplotlib library to visualize the diagram on the screen.

Example 29

Get your own Python Server

Import pyplot from Matplotlib and visualize our DataFrame:

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('data.csv')
df.plot()
plt.show()
```

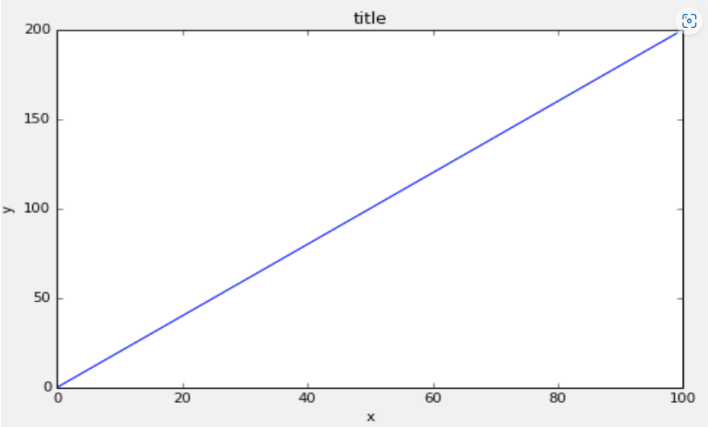
Questions

1. Follow along with these steps:

a) Create a figure object called fig using plt.figure()

b) Use add_axes to add an axis to the figure canvas at [0,0,1,1]. Call this new axis ax.

c) Plot (x,y) on that axes and set the labels and titles to match the plot below:



2. Create a figure object and put two axes on it, ax1 and ax2. Located at [0,0,1,1] and [0.2,0.5,.2,.2] respectively. Now plot (x,y) on both axes. And call your figure object to show it.

3. Use the company sales dataset csv file, read Total profit of all months and show it using a line plot Total profit data provided for each month. Generated line plot must include the following properties: –

a. X label name = Month Number
b. Y label name = Total profit

4. Use the company sales dataset csv file, get total profit of all months and show line plot with the following Style properties. Generated line plot must include following Style properties: –

a. Line Style dotted and Line-color should be red
b. Show legend at the lower right location.
c. X label name = Month Number
d. Y label name = Sold units number
e. Add a circle marker.
f. Line marker color as read
g. Line width should be 3

Additional Questions

1. Use the company sales dataset csv file, read all product sales data and show it using a multiline plot. Display the number of units sold per month or each product using multiline plots. (i.e., Separate Plotline for each product).
2. Use the company sales dataset csv file, calculate total sale data for last year for each product and show it using a Pie chart.
   Note: In Pie chart display Number of units sold per year for each product in percentage.