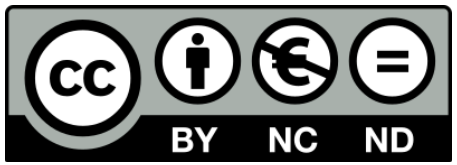
Data Analysis with Python, Numpy and Pandas

Jean-Pierre Messager (jp@xiasma.fr)

19 of September, 2021 – version 0.7a



No commercial use without authorization

LICENSE

Creative Commons Licence Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-ND 4.0)

This is a human-readable summary of (and not a substitute for) the license :

https://creativecommons.org/licenses/by-nc-nd/4.0/legalcode

You are free to:

Share — copy and redistribute the material in any medium or format.

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:

Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

NonCommercial — You may not use the material for commercial purposes.

NoDerivatives — If you remix, transform, or build upon the material, you may not distribute the modified material.

No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

Anaconda

Anaconda is a single package installing Python and a huge set of modules and tools

- *IPython* : improved REPL command line interface
- *numpy* : efficient number and multi-dimensionnal arrays computation
- *pandas* : data indexing, querying and aggregation
- Both provide objects with similar interfaces (*iteration protocol*, method names) to built-in Python or array module objects
- Interface nicely with CSV files, SQL databases, even Excel files
- *scipy* : numerical analysis, linear algebra, statistics
- *scikit-learn* : classification, clustering, regression
- *matplotlib*, *seaborn* : data visualization, imaging

Note that all these package may also be installed *independantly* from Anaconda by *pip*.

Anaconda

Graphical User Interface

- *anaconda-navigator* gives access to most graphical tools
- *Jupyter* : notebook oriented Web interfaces
- *Spyder* : Integrated Development Environment
- PyCharm can also interfaces itself with Anaconda

More packages (set of modules) can be installed by `conda` command.

Notebooks

- You can share notebooks IPython files saved by Jupyter
- Many are available on the Internet

Numpy

Efficient numerical and array types

```
>>> import numpy as np
>>> np.array( [ [1,2,3] , [4,5,6] ] )
array([[1, 2, 3],
       [4, 5, 6]])
>>> t = np.array( [ [1, 2, 3] , [4, 5, 6] ] )
>>> type(t[0][1])
<class 'numpy.int64'>
>>> t = np.array( [ [1, 2, 3] , [4, 5.0, 6] ] )
>>> type(t[0][1])
<class 'numpy.float64'>
```

Mathematical functions

```
>>> np.exp(t)
array([[  2.71828183,   7.3890561 ,  20.08553692],
       [ 54.59815003, 148.4131591 , 403.42879349]])
```

Arrays in numpy

Arrays creation

- Can be generated from sequences, constant or random values, ranges, files
- Can be reshaped, iterated through, flattened
- Items can be addressed by single or multiple index, slices

Examples

```
>>> v = np.array([1,2,3])
>>> v.dtype # dtype('int64')
>>> v.shape # (3,)
>>> t = np.array([ [1,2,3], [4,5,6.0] ])
>>> t.dtype # dtype('float64')
>>> t.shape # (2, 3)
```

Array creation functions

Constant and special arrays

```
>>> np.zeros((2,2))
>>> np.ones((3,4))
>>> np.full((3,3), 42)
>>> np.eye(3) # identity matrix
>>> np.random.random(2,2)
>>> np.random.randn(4) # Normal std distribution
```

From files or Web ressources

```
>>> np.genfromtxt('myfile.csv', delimiter=';')
```

Functions and operators

Use function from numpy (not math!) or operators

```
>>> np.log(t)
>>> t * 42 # overloaded by numpy
>>> t + 2*t
```

Operations on arrays

- Functions acting on elements
- Linear algebra : `np.linalg`
- Compound functions (sum, mean, average, ...)

```
>>> np.sum(t)
21.0
>>> np.sum(t, axis=0)
array([5., 7., 9.])
>>> np.sum(t, axis=1)
array([ 6., 15.])
```

Other Numpy features, Scipy, Matplotlib

Submodules of Numpy

- numpy.linalg for linear algebra
- polynomial, random, fft (Fourier transformation),
- ...

Scipy is a useful complement of numpy

- Provides more statistical functions
- Combinatorics

Matplotlib is the base module to produce graphs

Can be displayed directly by Jupyter

```
%matplotlib inline
import numpy as np
from matplotlib import pyplot
pyplot.hist(np.random.random(100),
            range = (0, 1),
            bins = 100, color = 'blue',
            edgecolor = 'black')
pyplot.show()
```

- Can produce histogram, pie charts, curves, points, ...
- Documentation provides complete examples of all types

Pandas series

Build on top of Numpy

- Dictionary-like indexing
- Both sequence-like and dictionary-like objects

```
>>> data = pd.Series([42.42, 3.14, 1.0],
                     index=['price','size','weight'])
>>> data['price']
42.42
>>> data
price     42.42
size       3.14
weight     1.00
>>> data['size'] == data[1]
True
```

Pandas series

Database-like object

```
>>> data.where( (lambda x: x > 2) )
price     42.42
size       3.14
weight      NaN
dtype: float64
```

Filtering out NaNs

```
>>> data.where( (lambda x: x > 2) ).dropna()
price    42.42
size      3.14
dtype: float64
```

Pandas dataframes

Both are build on top of np.arrays

- Series are general sequence and dictionary-like one dimensional objects
- Dataframes are general sequence and dictionary-like two-dimensional objects

```
>>> prices = pd.Series( { 'spam':12.5, 'egg':4.3 } )
>>> qty    = pd.Series( { 'spam':4, 'egg':12 } )
>>> stock  = pd.DataFrame( { 'price': prices,
                             'qty': qty } )
>>> stock
      price  qty
spam   12.5    4
egg     4.3   12
```

Dataframe manipulation

You can add columns like you add dictionary entries

```
>>> stock['value'] = stock['price'] * stock['qty']
>>> stock
      price  qty  value
spam   12.5    4   50.0
egg     4.3   12   51.6
```

And aggregates values

```
>>> stock.sum()
price     16.8
qty       16.0
value    101.6
```

Data filtering

```
>>> stock[stock.price > 5]
      price  qty  value
spam   12.5    4   50.0
```

How can it works??

- Isn't `stock.price > 5` supposed to be a boolean?
- No it isn't! Not at all!

```
>>> stock.price > 5
spam     True
egg     False
Name: price, dtype: bool
>>> type(stock.price > 5)
<class 'pandas.core.series.Series'>
```

Pandas *massively* overloads comparison operators.

Pandas data sources

Dataframes can be build from

- Local files
- Remote data accessible through http(s) urls
- Remote well known public ressources
- Remote ressources requiring an access key

`pandas_datareader` module allows to directly access public (or not) data sources such as Eurostat, the World Bank, OECD, etc.

`yfinance` module to access Yahoo finance, etc.

Application to nuclear physics

Download data from CERN

```
dataurl = 'http://opendata.cern.ch/record/700/files/
 MuRun2010B.csv'
import pandas as pd
data = pd.read_csv(dataurl)
data
```

Fix an incorrect column name!

```
data['px1'] = data['px1 '] # Sanitizing field name...
del(data['px1 ']) # we could have used rename
```

Nuclear Physics

Compute total momentum on all spatial directions

For particules beams 1 and 2...

```
data['vector sum px'] = data['px1'] + data['px2']
data['vector sum py'] = data['py1'] + data['py2']
data['vector sum pz'] = data['pz1'] + data['pz2']
```

Resultant momentum...

```
data['resultant momentum sum'] = \
  (data['vector sum px']**2 + \
   data['vector sum py']**2 + \
   data['vector sum pz']**2)**(0.5)
```

Nuclear Physics

Compute relativistic invariant mass

```
data['invariant mass'] = ((data['E1']+data['E2'])**2 \
     - data['resultant momentum sum'] ** 2)**(0.5)

min(data['invariant mass']), \
max(data['invariant mass'])

(1.958119099640791, 109.97712963856803)
```

Nuclear Physics

Display result

```
%matplotlib inline
from matplotlib import pyplot
pyplot.hist(data['invariant mass'],
            range = ( min(data['invariant mass']),
                      max(data['invariant mass'])),
            bins = 100, color = 'orange',
            edgecolor = 'red')
pyplot.show()
```

Add `log=True`, notice the peak at 91Gev. We've discovered the Z boson!
Reference: //opendata.cern.ch/record/72/files/3_DiMuonHistogramExcelInstructions.pdf

Database-like operation in Pandas

Similar to SQL

- `[field1, field2 ...]` for field selection
- Slices (numbers, dates)
- `loc` and `iloc` method to filter lines and colums
- operators for where-like conditions
- grouby and join methods
- specific methods for Series of data: `isin`, `pd.to_numeric`
- .type.method (ex: `str.split`)

A query mini-language

```
df.query("(name=='john') or (country=='UK')")
df.query('col1.str.contains("spam")')
names = ['john', 'terry']
df.query('name in @names')
```

Numpy, Pandas, Matplotlib, and data science ressources

- Python Data Science : book and Jupyter notebooks
  https://jakevdp.github.io/PythonDataScienceHandbook/
- Scipy/Numpy Introduction
  https://sites.engineering.ucsb.edu/~shell/che210d/numpy.pdf
- Scikit tutorial
  https://scikit-learn.org/stable/tutorial/index.html
- Matplotlib Pyplot tutorial
  https://matplotlib.org/tutorials/introductory/pyplot.html
- Quantitative Economics with Python
  https://python.quantecon.org/intro.html