# Python-Gate

May 11, 2017

## 1 Using Python for Gate Analysis

- Python in a high level (in terms of abstraction) versatile programming language well adapted for scientific use
  - Both Imperative and Oriented Object approach (at the same time)
  - Interpeted and dynamic typing (implicite type) and easy object manipulation and indexing
  - At the price of speed and memory managment (compared to C)
  - Various scientific libaries for data analysis and ploting (numpy, scipy, matplotlib)
  - Can read many file format like text and root (with dedicated library)
  - Free and open source
  - Written in C and able to read any C Class

### 1.1 Reading Root files with Python

- 3 Approaches :
  - The Root approach **PyROOT**: directly using Root in Python
  - The inbetween approach : **rootpy** : Redefine some ROOT classes in a more Pythonic way. Seems not be compatible with Python 3. Abbandoned ?
  - The Python approach **root_numpy**: Transfoming Root tree to numpy structured array

### 1.2 The Root Flavor

- ROOT comes with its own Python implementation : PyROOT
  - Pro :
    - No additional libraries needed to acces ROOT files
    - Possibility to use both ROOT and Python libraries for data manipulation and plotting
  - Cons :
    - Still using ROOT ;)

```
In [1]: import ROOT as rt
```

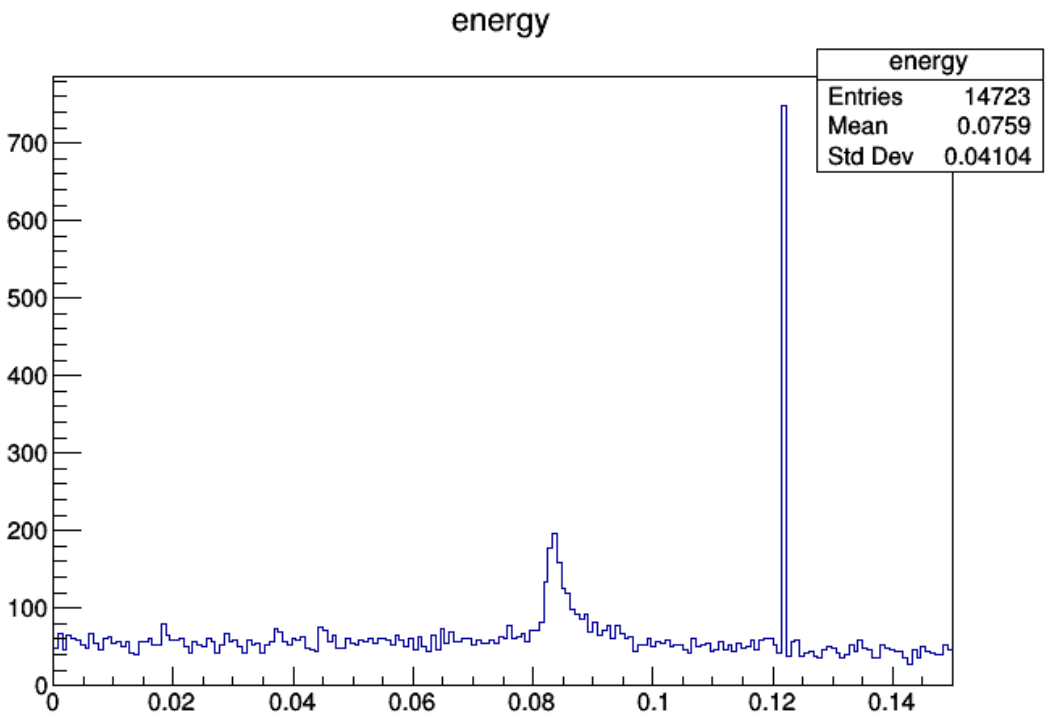
```
Welcome to JupyROOT 6.09/03
```

```
In [2]: tfil = rt.TFile('Simu_TReCam_contact_puit1.root')
        tree = tfil.Get('Singles')
```

```
In [3]: h1 = rt.TH1D('energy', 'energy', 200, 0, 0.15)
        for i in range(tree.GetEntries()):
            tree.GetEntry(i)
            h1.Fill(tree.energy)
```

```
In [4]: c1 = rt.TCanvas( 'c1', '', 200, 10, 700, 500 )
```

```
In [5]: h1.Draw()
        c1.Draw()
```




```
In [6]: tfil.Close()
```

- Works fine but it's just using ROOT with Python syntax, so what's the point ?!
- The data can be converted to an python numpy **ndarray** to benefit of numpy

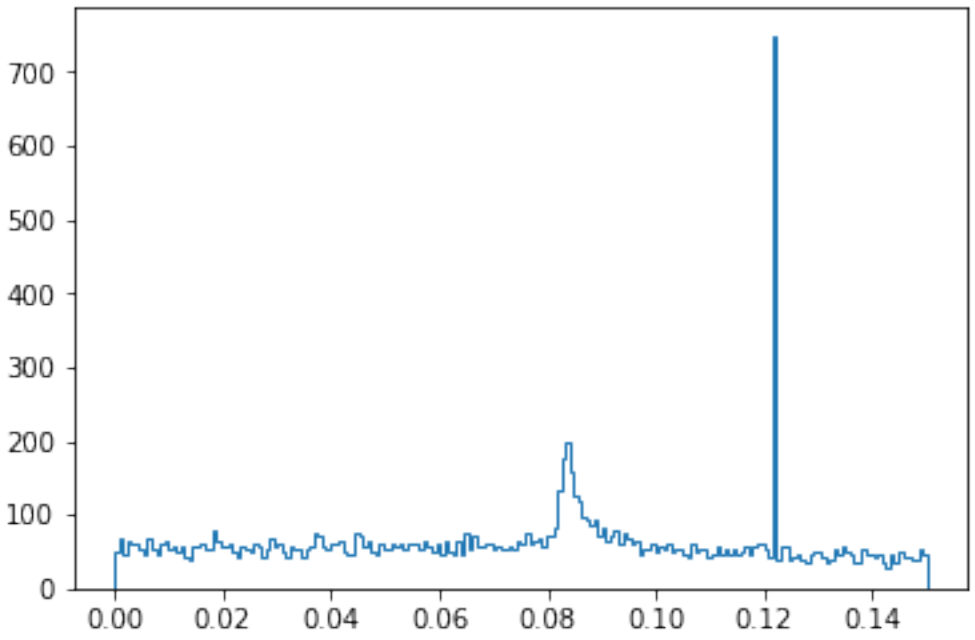
```
In [7]: %matplotlib inline
```

```
In [8]: import numpy as np # Data manipulaion
        import matplotlib.pyplot as plt # Graphic library
        import ROOT as rt

In [9]: tfil = rt.TFile('Simu_TReCam_contact_puit1.root')
        tree = tfil.Get('Singles')

In [10]: ene = np.zeros(tree.GetEntries()) # Initialize a 1 dim array with 0
         for i in range(tree.GetEntries()):
             tree.GetEntry(i)
             ene[i] = tree.energy

In [11]: plt.hist(ene, bins=200, range=(0, 0.15), histtype='step') # Both fill and plot the hist
         plt.show()
```




- This way benefits from both ROOT and numpy/matplotlib advantages, but still use ROOT

### 1.3 The Numpy Flavor

- The **root_numpy** library directly read root files and convert ROOT Trees to numpy structured array (more advanced ndarray)
  - Pro:
    - Very easy to use for end users and benefit from all numpy functiunalities
    - No ROOT
  - Cons:

    * Need third party library -> compatibility with ROOT version
    * Not memory friendly: the whole tree is loaded at once by default (option to limit this)

```
In [12]: import numpy as np
         import matplotlib.pyplot as plt

In [13]: import root_numpy as rnp

In [14]: singles = rnp.root2array('Simu_TReCam_contact_puit1.root', 'Singles')
         # One function to load a whole ROOT Tree in an array

In [15]: print(singles)
```

```
[ (0,     1755, 1,  22.28430748,  10.8377018 ,   3.95043778,   1.79555351e-02,  0.10200048,  22.
 (0,    10369, 1,  22.96845055,   5.18244171,   5.25857925,   1.03164452e-01,  0.0117202 ,  22.9
 (0,    13949, 0,   2.1362288 ,   1.9215287 ,  19.        ,   1.40135429e-01,  0.08359003, -21.9
 ...,
 (0, 71982813, 1,   8.18272209,  20.5252285 ,   3.92328787,   7.19788730e+02,  0.14628485,   8.1
 (0, 71996183, 0,   1.07309175,   0.66100019,  19.        ,   7.19923060e+02,  0.04958908,  -7.5
 (0, 71998014, 1,  -4.37246513, -22.17212105,   5.94269371,   7.19941406e+02,  0.11886024,  -4.3
```

```
In [16]: print(singles.dtype)
```

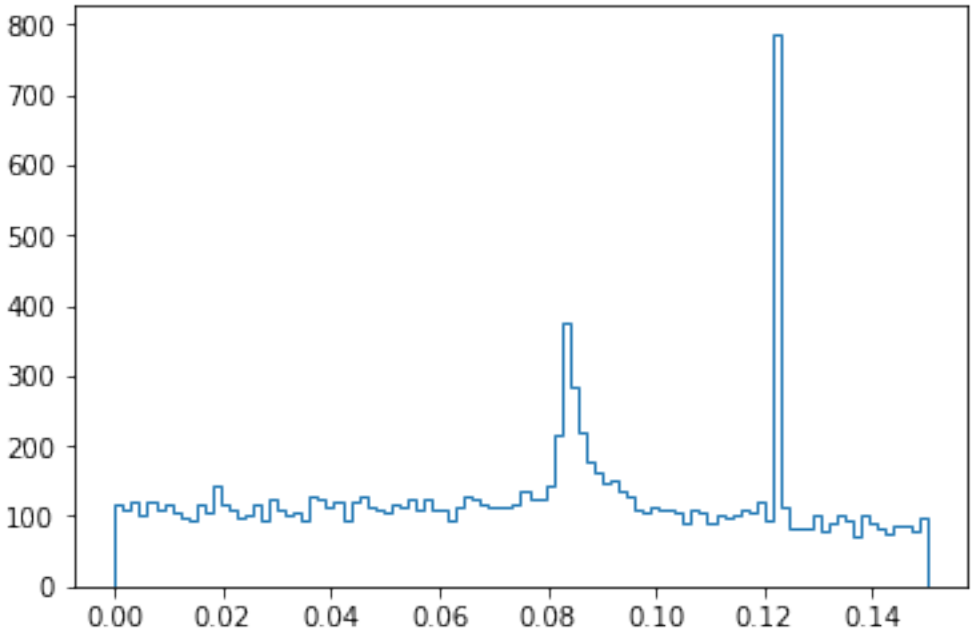
```
[('runID', '<i4'), ('eventID', '<i4'), ('sourceID', '<i4'), ('sourcePosX', '<f4'), ('sourcePosY'
```

- Acces to "leaves" is done using their names

```
In [17]: ene1 = singles['energy']
         posx = singles['globalPosX']
         print(ene)
         print(posx)
```

```
[ 0.10200048  0.0117202   0.08359003 ...,  0.14628485  0.04958908
  0.11886024]
[ 22.26452255  22.96856499 -21.97757721 ...,   8.14372635  -7.54512739
  -4.35010099]
```

```
In [18]: plt.hist(ene1, bins=100, range=(0, 0.15), histtype='step')
         plt.show()
```



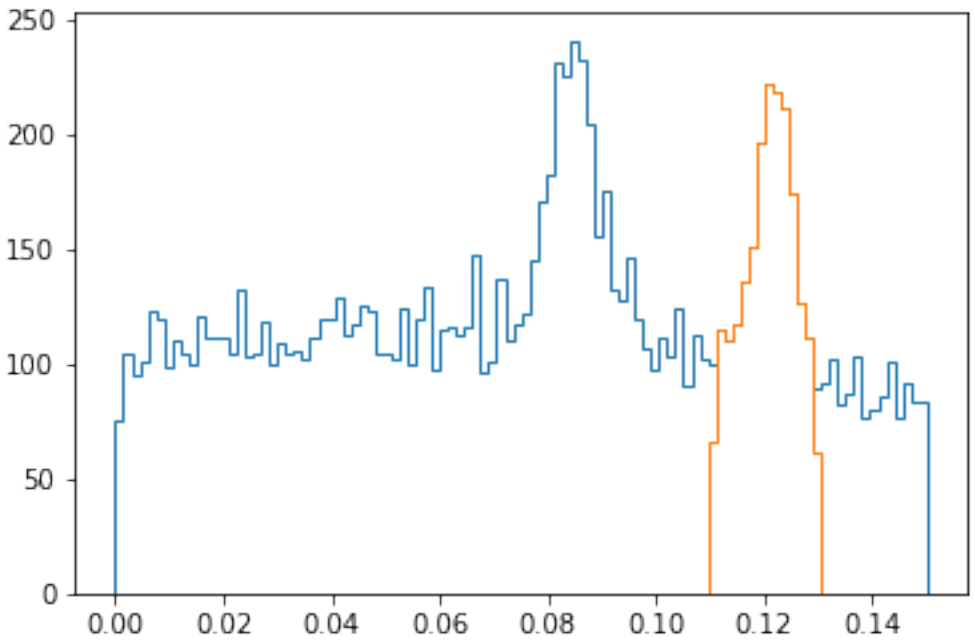

- Numpy array allow easy math operations

```
In [19]: noise = np.random.normal(0, 0.003, len(ene1))
         ene2 = ene1 + noise # Add Gaussian random fluctuation
```

- And easy indexing

```
In [20]: ene3 = ene2[(ene2>0.11) & (ene2<0.13)] # Energy Cut
```

```
In [21]: plt.hist(ene2, bins=100, range=(0, 0.15), histtype='step')
         plt.hist(ene3, bins=100, range=(0, 0.15), histtype='step')
         plt.show()
```




- Numpy also possess its own format to save arrays either in text or binary format
- Adding direct save of Gate simulation in numpy format would solve the issue of root_numpy compatibility with ROOT and get completely rid of ROOT ;)
- I'm going to have a look at it to add it to Gate and also add a couple of exemple for data anlaysis
- Python provide powerful tool for data analysis that can easily be addapted and used for Gate.
  - Easier to use and to learn for end users or lazy physicists (like me)
  - Lack the power and speed of pure C and ROOT libraries for high amount of Data