

ROOT 講習会 第5回

PyROOT

濱田悠斗

神戸大 粒子物理研究室

自己紹介

- 濱田 悠斗
- 神戸大学 粒子物理学研究室 M1
- MIRACLUE実験, XENON
- 趣味：スノーボード
- 甘いもの (小麦系) が好き



第5回でやること

- TTreeの復習
- Pythonとは
- PyROOT
 - PyROOTとは
 - 使ってみる
 - PyROOTを使うモチベーション
 - PyROOT風 TTreeの読み方
 - RDataFrameの紹介

TTreeの復習

- 1 ブランチだけのTTreeを作ってファイルに保存してみよう
 - 作成例： root_lecture/macros/hamada/example01.C
 - データの場所：root_lecture/macros/hamada/data/data1.txt
 - 第2, 3回で使ったscintillator_expのデータ

data1.txt

```
~/VS/root_lecture/macros/hamada git:(main) (0.109s)
head data/data1.txt
2074
3049
2774
2177
3107
1934
762
790
1192
2153
```

data1.root

```
~/VS/root_lecture/macros/hamada git:(main)
root rootfiles/data1.root
root [0]
Attaching file rootfiles/data1.root as _file0...
(TFile *) 0x142e2f9b0
root [1] tree->Scan()
*****
*      Row      * adc_value *
*****
*      0 *      2074 *
*      1 *      3049 *
*      2 *      2774 *
*      3 *      2177 *
*      4 *      3107 *
*      5 *      1934 *
*      6 *       762 *
*      7 *       790 *
*      8 *      1192 *
*      9 *      2153 *
```

TTreeの復習

- 作成例

```
macros > hamada > C example01.C
1 void example01(TString data_path, TString out_file_path){
2     // TTreeを準備
3     TTree* tree = new TTree("tree", "tree");
4     double adc_value;
5     tree->Branch("adc_value", &adc_value, "adc_value/D");
6
7     // treeにデータを読み込む
8     ifstream ifs(data_path);
9     while(ifs >> adc_value){
10         tree->Fill();
11     }
12
13     // treeを保存
14     TFile* out_file = new TFile(out_file_path, "recreate");
15     tree->Write();
16     out_file->Close();
17 }
```

```
~/VS/root_lecture/macros/hamada git:(main) (2.967s)
root 'example01.C("data/data1.txt", "rootfiles/data1.root")'
```


Pythonとは

- インタプリタ型のプログラミング言語
 - ROOTはコンパイル型言語C++のインタプリタ (のようなもの)
- 可読性を重視して作られた言語
 - インデントでブロックが表現されたり
- 動的に型付けされる
- 例えば, 配列の和を返す関数だとこうなる

macros > hamada >  calc_sum.C

```
1  int calc_sum(vector<int> arr) {  
2      int ret = 0;  
3      for (int i = 0; i < arr.size(); i++) {  
4          ret += arr[i];  
5      }  
6      return ret;  
7  }
```

C++

macros > hamada >  calc_sum.py

```
1  def calc_sum(arr):  
2      ret = 0  
3      for val in arr:  
4          ret += val  
5      return ret
```

Python

PyROOTとは

- ROOTのPythonインタフェース
 - PythonからROOTを呼び出せる
 - PythonでROOTの各機能が実装されているわけではない
- ROOTのホームページにある解説ページ
 - <https://root.cern/manual/python>
 - マニアックなことが書いてある

PyROOT: 使ってみよう

- さっき作ったtreeの中身をTH1DにFillしてみよう
 - まずは復習がてら, C++で

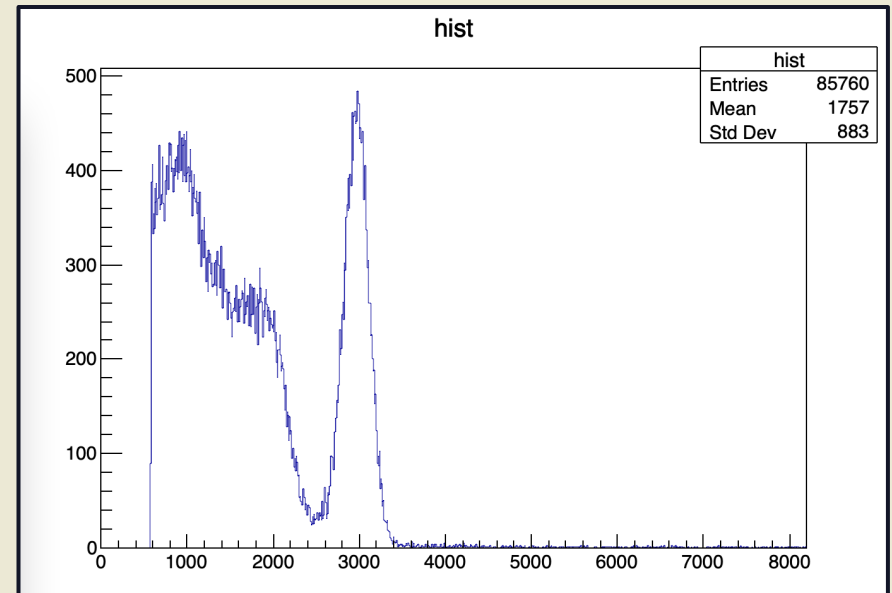
```
macros > hamada > C example02.C
```

```
1 void example02(TString rootfile_path) {
2     // Treeを読み込む
3     TFile* root_file = TFile::Open(rootfile_path);
4     TTree* tree = (TTree*)root_file->Get("tree");
5
6     // branchから値を読み出す準備
7     double adc_value;
8     tree->SetBranchAddress("adc_value", &adc_value);
9
10    // ヒストグラムの準備
11    const int NBINS = 1024;
12    const double XMIN = 0;
13    const double XMAX = 8192;
14    TH1D* hist = new TH1D("hist", "hist", NBINS, XMIN, XMAX);
15
16    // Treeからヒストグラムにデータを入れる
17    int n_entries = tree->GetEntries();
18    for (int i_entry = 0; i_entry < n_entries; i_entry++) {
19        tree->GetEntry(i_entry);
20        hist->Fill(adc_value);
21    }
22
23    // ヒストグラムをdraw
24    hist->Draw();
25 }
```

実行

```
~/VS/root_lecture/macros/hamada git:(main)
root 'example02.C("rootfiles/data1.root")'
root [0]
Processing example02.C("rootfiles/data1.root")...
Info in <TCanvas::MakeDefCanvas>: created default TCanvas with name c1
root [1]
```

こんな絵が出るはず



PyROOT: 使ってみよう

- PyROOTで書くと…
 - 素直な書き換え

macros > hamada > C example02.C

```
1 void example02(TString rootfile_path) {
2     // Treeを読み込む
3     TFile* root_file = TFile::Open(rootfile_path);
4     TTree* tree = (TTree*)root_file->Get("tree");
5
6     // branchから値を読み出す準備
7     double adc_value;
8     tree->SetBranchAddr("adc_value", &adc_value);
9
10    // ヒストグラムの準備
11    const int NBINS = 1024;
12    const double XMIN = 0;
13    const double XMAX = 8192;
14    TH1D* hist = new TH1D("hist", "hist", NBINS, XMIN, XMAX);
15
16    // Treeからヒストグラムにデータを入れる
17    int n_entries = tree->GetEntries();
18    for (int i_entry = 0; i_entry < n_entries; i_entry++) {
19        tree->GetEntry(i_entry);
20        hist->Fill(adc_value);
21    }
22
23    // ヒストグラムをdraw
24    hist->Draw();
25 }
```

C++

macros > hamada > example02.py > ...

```
1 import ROOT
2
3 # Treeを読み込む
4 root_file_path = "rootfiles/data1.root"
5 root_file = ROOT.TFile.Open(root_file_path)
6 tree = root_file.Get("tree")
7
8 # ヒストグラムの準備
9 NBINS = 1024
10 XNIN = 0
11 XMAX = 8192
12 hist = ROOT.TH1D("hist", "hist", NBINS, XNIN, XMAX)
13
14 # Treeからヒストグラムにデータを入れる
15 n_entries = tree.GetEntries()
16 for i_entry in range(n_entries):
17     tree.GetEntry(i_entry)
18     hist.Fill(tree.adc_value)
19
20 # ヒストグラムをdraw
21 hist.Draw()
```

Python

実行はこんな感じ

```
~/VS/root_lecture/macros/hamada git:(main)
python3 -i example02.py
```

PyROOTを使うモチベーション

- Pythonは文字列の扱いが簡単
 - C++ と比べて
- Pythonの豊富なライブラリ群を扱える
 - 標準ライブラリが頼もしい
 - “batteries included”
 - 数値計算, 機械学習等の外部ライブラリが豊富 / 成熟している
 - C++も外部ライブラリは多くはないが, ROOTとの連携が大変
 - scipy, numpy, tensorflow, pytorch, etc
- 動的型付け, 簡単な文法
 - 時間をかけずに簡単な解析をしたい時に便利

PyROOTを使うモチベーション: 文字列

- 一部を紹介
- slice
 - 直感的な書き方ができる
- replace
 - C++で書くと長い
- split
 - C++にない (TStringにはある)

```
s = 'abc123def456'  
print(s[-1])  
print(s[3:5])  
print(s[3:])  
print(s[:5])
```

```
6  
12  
123def456  
abc12
```

```
s = 'abc123def456'  
print(s.replace('abc', 'ABC'))
```

```
ABC123def456
```

```
macros > hamada > C example_str.C
```

```
1 void example_str()  
2 {  
3     string s = "abc123def456";  
4     string s_replaced = regex_replace(s, regex("abc"), "ABC");  
5     cout << s_replaced << endl;  
6 }
```

```
s = '/home/username/dir1/dir2/test.text'  
s_split = s.split('/')  
print(s_split)
```


```
['', 'home', 'username', 'dir1', 'dir2', 'test.text']
```

PyROOTを使うモチベーション: 標準lib

- 一部を紹介
- CSV
 - コンマ区切りのテキストデータを読み込む
- pathlib
 - パスの文字列を便利に扱える
- argparse
 - コマンドライン引数を受け取る
- json
 - jsonを読める

PyROOTを使うモチベーション: pathlib

- pathを便利に扱えるクラスを提供している
 - ここで紹介している機能はほんの一部

```
macros > hamada >  example_pathlib.py > ...  
1  from pathlib import Path  
2  
3  p = Path('rootfiles')  
4  print(p.is_dir()) # True  
5  print(p.is_file()) # False  
6  print(p.is_absolute()) # False  
7  print(p.resolve()) # /Users/yuto/VS/root_lecture/macros/hamada/rootfiles  
8  
9  data1_path = p / 'data1.root'  
10 print(data1_path.as_posix()) # rootfiles/data1.root  
11 print(data1_path.name) # data1.root  
12 print(data1_path.stem) # data1  
13 print(data1_path.suffix) # .root
```

PyROOT風 TTreeの読み方

- よりPythonっぽい

macros > hamada > example02.py > ...

```
1  import ROOT
2
3  # Treeを読み込む
4  root_file_path = "rootfiles/data1.root"
5  root_file = ROOT.TFile.Open(root_file_path)
6  tree = root_file.Get("tree")
7
8  # ヒストグラムの準備
9  NBINS = 1024
10 XNIN = 0
11 XMAX = 8192
12 hist = ROOT.TH1D("hist", "hist", NBINS, XNIN, XMAX)
13
14 # Treeからヒストグラムにデータを入れる
15 n_entries = tree.GetEntries()
16 for i_entry in range(n_entries):
17     tree.GetEntry(i_entry)
18     hist.Fill(tree.adc_value)
19
20 # ヒストグラムをdraw
21 hist.Draw()
```

macros > hamada > example03.py > ...

```
1  import ROOT
2
3  # Treeを読み込む
4  root_file_path = "rootfiles/data1.root"
5  root_file = ROOT.TFile.Open(root_file_path)
6  tree = root_file.tree # Pythonっぽい ←
7
8  # ヒストグラムの準備
9  NBINS = 1024
10 XNIN = 0
11 XMAX = 8192
12 hist = ROOT.TH1D("hist", "hist", NBINS, XNIN, XMAX)
13
14 # Treeからヒストグラムにデータを入れる
15 n_entries = tree.GetEntries()
16 for event in tree: # pythonっぽい ←
17     hist.Fill(event.adc_value)
18
19 # ヒストグラムをdraw
20 hist.Draw()
```

PyROOT風 TTreeの読み方

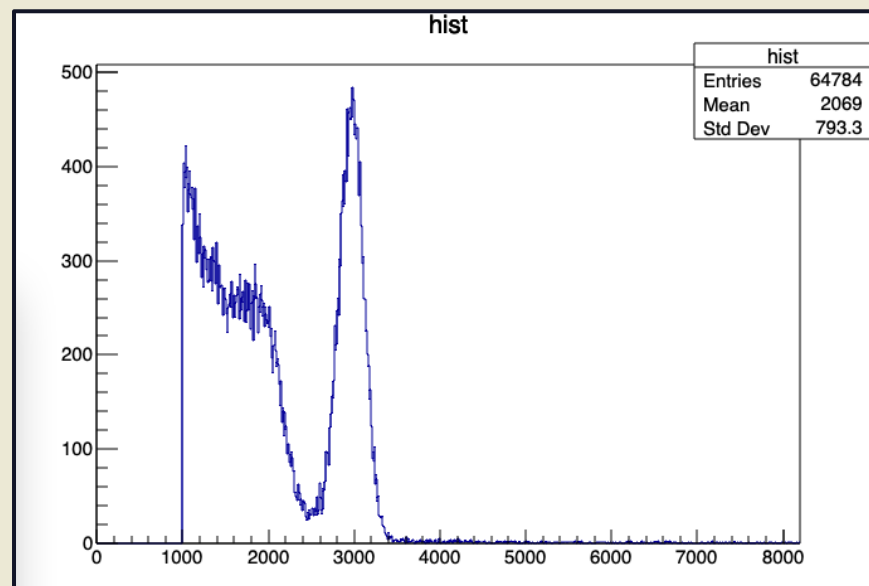
- ちょっと手を動かす
 - root_lecture/macros/hamada/example03.py
 - adc_valueが1000より大きい時にだけFillするように変更してみよう

macros > hamada > example03.py > ...

```
1  import ROOT
2
3  # Treeを読み込む
4  root_file_path = "rootfiles/data1.root"
5  root_file = ROOT.TFile.Open(root_file_path)
6  tree = root_file.tree # Pythonっぽい
7
8  # ヒストグラムの準備
9  NBINS = 1024
10 XNIN = 0
11 XMAX = 8192
12 hist = ROOT.TH1D("hist", "hist", NBINS, XNIN, XMAX)
13
14 # Treeからヒストグラムにデータを入れる
15 n_entries = tree.GetEntries()
16 for event in tree: # pythonっぽい
17     hist.Fill(event.adc_value)
18
19 # ヒストグラムをdraw
20 hist.Draw()
```

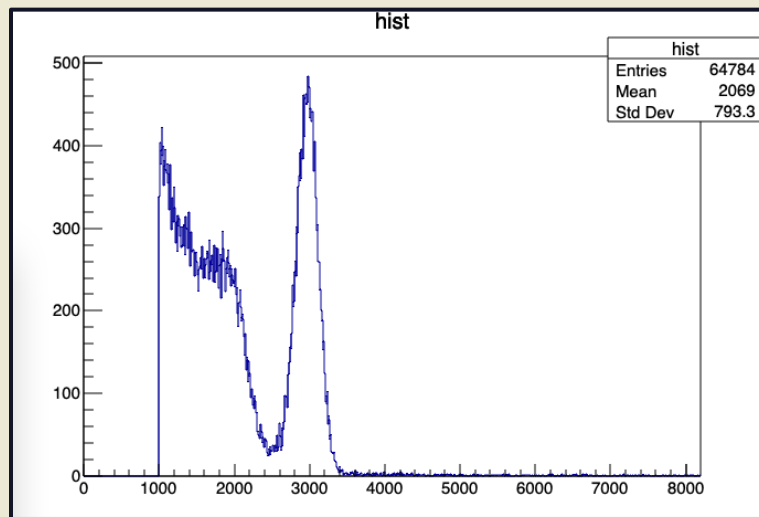
この辺に追記

こんな感じの絵が見えるはず



PyROOT風 TTreeの読み方: もっと速く

- Pythonのforは速くない
 - C++ と比べて
- ROOTの内部機能を使えば裏でC++がうごく
 - tree->Draw()を使えば良い
- tree->Draw()で前ページと同じ絵を出してみよう



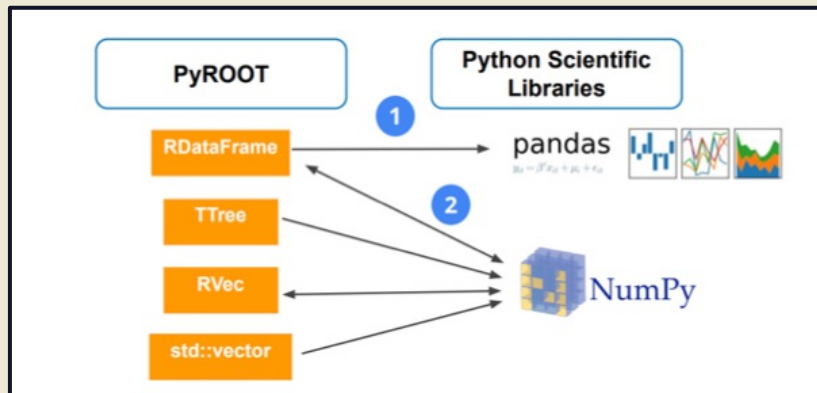
PyROOT風 TTreeの読み方: もっと速く

- こんな感じで書けばヒストグラムのインスタンスをとってこれる

```
macros > hamada > example04.py > ...
1  import ROOT
2
3  # Treeを読み込む
4  root_file_path = "rootfiles/data1.root"
5  root_file = ROOT.TFile.Open(root_file_path)
6  tree = root_file.tree # Pythonっぽい
7
8  # ヒストグラムの準備
9  NBINS = 1024
10 XMIN = 0
11 XMAX = 8192
12
13 # tree->Draw() の結果をhistという名前のオブジェクトに保存させる
14 tree.Draw(f"adc_value >> hist({NBINS}, {XMIN}, {XMAX})", "adc_value > 1000")
15
16 # histを取ってくる
17 hist = ROOT.gDirectory.Get("hist")
18 hist.GetAxis().SetRangeUser(0, 4000)
19 hist.Draw()
```

RDataFrame

- treeを読むためのTTree以外のクラス
 - Pythonでよく使うライブラリたちへのコンバートが簡単
 - 前述のtree->Draw()の代わりになる
 - 速度を落とさず処理ができるということ



```
# Run input pipeline with C++ performance that can process TBs of data
df = ROOT.RDataFrame('tree', 'file.root')
    .Filter('pT_j0 > 30')
    .Filter('n_jet >= 2')
    .Define('r_j0', 'sqrt(eta_j0*eta_j0 + phi_j0*phi_j0)')

# Read out final selection with defined variables as NumPy arrays
col_dict = df.AsNumpy(['r_j0', 'eta_j0', 'phi_j0'])
print(col_dict)

{'r_j0': ndarray([0.26,1.,4.45]), 'eta_j0': ndarray(0.1,-1.,2.1),
'phi_j0': ndarray([-0.5,0.,0.2])}

# Wrap data with pandas
p = pandas.DataFrame(col_dict)
print(p)

   r_j0  eta_j0  phi_j0
0  0.26   0.1   -0.5
1  1.0   -1.0    0.0
2  4.45   2.1    0.2
```

<https://doi.org/10.1051/epjconf/202024506004>

RDataFrame: 使ってみよう

- RDataFrameを使って example03.py と同等の処理を書こう
 - example05.pyに模範解答があります

```
macros > hamada > example03.py > ...
1  import ROOT
2
3  # Treeを読み込む
4  root_file_path = "rootfiles/data1.root"
5  root_file = ROOT.TFile.Open(root_file_path)
6  tree = root_file.tree # Pythonっぽい
7
8  # ヒストグラムの準備
9  NBINS = 1024
10 XNIN = 0
11 XMAX = 8192
12 hist = ROOT.TH1D("hist", "hist", NBINS, XNIN, XMAX)
13
14 # Treeからヒストグラムにデータを入れる
15 n_entries = tree.GetEntries()
16 for event in tree: # pythonっぽい
17     hist.Fill(event.adc_value)
18
19 # ヒストグラムをdraw
20 hist.Draw()
```

おわり

- PyROOTは情報が少ないので気軽に質問ください
- YMAPに入ってROOT講習会の講師をしましょう