Personal Report: Comparing Different Frameworks for HEP Analysis: ROOT, PyROOT, and CutLang from an Undergraduate's Perspective

IBRAHIM H.I. ABUSHAWISH
Istanbul University, Physics Department
Instructor: Prof. Aytül ADIGÜZEL
Report Submission Date: DD.MM.YYYY

***Abstract*—This report presents a comparative study where I gained familiarity with CutLang by comparing it to familiar frameworks like ROOT and PyROOT in High Energy Physics (HEP) analysis. By implementing the same analysis across these frameworks, I developed a deeper understanding of CutLang's unique features and operation. The comparison focused on basic implementations, examining syntax, structure, and methodologies. While ROOT and PyROOT offer extensive documentation and community support, CutLang provides a simpler, more intuitive syntax. This comparative approach proved effective as a learning strategy, making CutLang's concepts more accessible through the context of previously known frameworks.**

***Index Terms*—High Energy Physics (HEP), ROOT, PyROOT, CutLang, Data Analysis, Performance Comparison, Physics Software**

I. Introduction

This study, primarily conducted for educational purposes, aimed to understand CutLang by comparing it with familiar frameworks like ROOT and PyROOT in High Energy Physics (HEP) analysis. As a student learning CutLang, comparing it with previously known frameworks provided valuable insights into its unique features and operation. The comparison focused on basic implementations across these frameworks, emphasizing learning patterns rather than complex analysis.

The comparison examined syntax, structure, and implementation approaches, serving as a practical learning exercise to grasp CutLang's concepts through familiar ROOT and PyROOT contexts. This educational approach helped bridge the gap between traditional HEP analysis tools and newer frameworks like CutLang.

The installation of the frameworks was straightforward, with ROOT and PyROOT being installed via the CERN software repository [2]. CutLang was installed from the GitHub repository [3], and the installation process was well-documented.

II. Methodology

The First step was to install the frameworks, which was straightforward. The datasets were loaded using the TTree class in ROOT and firstly analyzed using C++ in ROOT framework. The same analysis was then implemented in PyROOT, adapting the C++ code to Python syntax while maintaining the ROOT functionality. Finally, the analysis was translated to CutLang using its ADL (Analysis Description Language) syntax, and the file was compiled to perform the same analysis tasks.

The datasets used in this study were obtained from the ATLAS Open Data [4].

III. Results

The analysis was implemented in three different frameworks, each producing histograms showing photon energy distributions. While all frameworks achieve the same analytical goal, their approaches differ significantly in syntax, structure, and implementation methodology.

```cpp
void plot_photon_energy() {
    // Open file and get tree
    TFile *file = TFile::Open("../DATA/data_A.GamGam.root");
    ... # Error handling
    TTree *tree = (TTree*)file->Get("mini");
    ... # Error handling
    TH1F *hPhotonE = new TH1F("hPhotonE", "Photon Energy Distribution;Energy (MeV);Events", 50, 0, 400000);
    // Set up branch for reading
    std::vector<Double_t> *photon_E = nullptr;
    tree->SetBranchAddress("photon_E", &photon_E);

    // Loop over entries
    Long64_t nentries = tree->GetEntries();
    for (Long64_t i = 0; i < nentries; i++) {
        tree->GetEntry(i);
        if (!photon_E) continue;  // Avoid null pointer access
        for (float energy : *photon_E) {
            hPhotonE->Fill(energy);
        }
    }

    // Draw histogram
    TCanvas *c1 = new TCanvas("c1", "Photon Energy", 800, 600);
    hPhotonE->SetLineColor(kBlue);
    hPhotonE->Draw();
    // Save the plot
    c1->SaveAs("photon_energy.png");

    // Clean up
    delete c1;
    delete hPhotonE;
    file->Close();
}
```

Listing 1: ROOT C++ Implementation

Fig. 1b represents the PyROOT implementation, which bridges traditional ROOT functionality with Python's modern programming paradigms. This approach offers several advantages:

```python
import ROOT
file = ROOT.TFile.Open("../DATA/data_A.GamGam.root")
... # Error handling
tree = file.Get("mini")
... # Error handling
hPhotonE = ROOT.TH1F("hPhotonE", "Photon Energy Distribution;Energy (MeV);Events", 50, 0, 400000)

photon_E = ROOT.std.vector('float')()
tree.SetBranchAddress("photon_E", photon_E)

nentries = tree.GetEntries()
print(nentries)
for i in range(nentries):
    tree.GetEntry(i)
    for energy in photon_E:
        hPhotonE.Fill(energy)

c1 = ROOT.TCanvas("c1", "Photon Energy", 800, 600)
hPhotonE.SetLineColor(ROOT.kBlue)
hPhotonE.Draw()

c1.SaveAs("photon_energy_pyroot.png")

file.Close()
```

Listing 2: PyROOT Implementation

Fig. 1c showcases CutLang's unique approach using ADL (Analysis Description Language). This framework introduces a domain-specific language that emphasizes physics concepts over programming constructs:

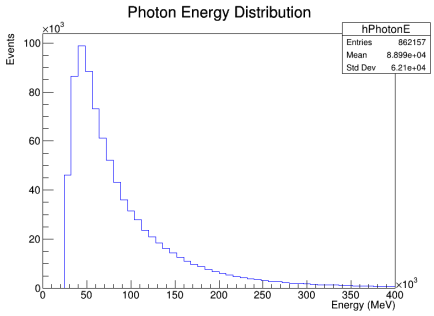
```
#object
object goodPhotons
    take PHO
    select E(PHO) > 0  # Ensuring energy is positive

# histogram
histoList photonHistos
    histo hPhotonE, "Photon Energy Distribution;Energy (GeV);Events", 50, 0, 400, E(goodPhotons)

# event
region test
    select ALL
    select Size(goodPhotons)>0
    photonHistos
```

Listing 3: CutLang ADL Implementation

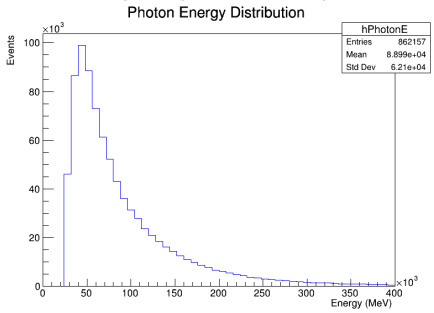
Common features across all implementations include: - Tree-based data access - Similar binning and range parameters - Comparable histogram visualization - Event-by-event processing capability

Key differences include: - Memory management (manual in ROOT, automatic in PyROOT and CutLang) - Syntax complexity (highest in ROOT, lowest in CutLang) - Language paradigm (procedural in ROOT, object-oriented in PyROOT, declarative in CutLang) - Development speed (fastest in CutLang, followed by PyROOT, then ROOT)

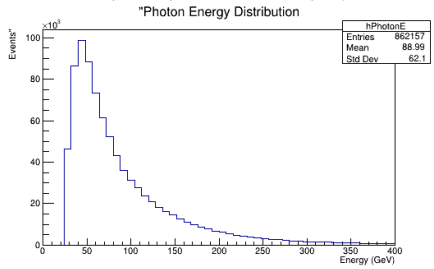
All three implementations produce comparable results in terms of physics analysis, though each framework offers distinct advantages depending on the user's requirements and expertise level.



(a) Histogram generated using ROOT



(b) Histogram generated using PyROOT



(c) Histogram generated using CutLang

Fig. 1: Comparison of histograms generated by different frameworks showing the distribution of photon energy. All three frameworks produce similar results.

IV. Discussion

This educational study, aimed at understanding CutLang through comparative analysis, provided valuable insights into different framework approaches. The primary goal was to learn CutLang by comparing it with familiar frameworks like ROOT and PyROOT, making the learning process more structured and relatable. The implementation across different frameworks revealed distinct challenges and learning curves. While ROOT and PyROOT benefit from extensive documentation and community support [2], CutLang, being relatively new, presents limited but growing resources [3]. Installation processes varied significantly in complexity:

- ROOT/PyROOT: Installation on Windows proved challenging, requiring careful attention to system configurations and dependencies. Linux installation was more straightforward through package managers.
- CutLang: Installation initially presented challenges due to root permission requirements, but following the GitHub documentation made the process straightforward, despite being a newer framework with a smaller user base.

Resource availability showed clear disparities:

- ROOT/PyROOT: Abundant tutorials, Stack Overflow discussions, and official documentation made problem-solving relatively faster.
- CutLang: While official documentation is comprehensive, fewer community resources meant longer problem-solving times. However, the framework's simpler syntax partially compensated for this limitation.

This comparison suggests that while established frameworks offer better support resources, newer alternatives like CutLang can still be viable options, particularly for users seeking more intuitive syntax despite limited community resources.

V. Conclusion

This report compared three frameworks used in High Energy Physics (HEP) analysis: ROOT, PyROOT, and CutLang. The comparison focused on syntax, structure, and implementation approaches.

ROOT and PyROOT, being well-established, offer extensive documentation and community support, while CutLang, though newer, provides a simpler and more intuitive syntax. Each framework has its strengths: ROOT for its comprehensive features, PyROOT for its integration with Python, and CutLang for its user-friendly ADL syntax.

The choice of framework depends on the user's requirements and expertise level, with all three producing comparable results in terms of physics analysis.

The comparison-based learning approach proved effective, as understanding CutLang through the lens of familiar frameworks made the learning process more structured and intuitive. By implementing the same analysis in ROOT, PyROOT, and CutLang, the similarities and differences between these frameworks became clearer, facilitating a deeper understanding of CutLang's unique features and advantages.

VI. Acknowledgments

I would like to acknowledge the use of Large Language Models (LLMs) such as GitHub Copilot in assisting with the search, study, and writing of this report. These tools provided valuable support in generating content and improving the overall quality of the report.

References

[1] *Source code and additional experiments are available in the GitHub repository.* https://github.com/ibeuler/ROOTBench
[2] *Installing ROOT*: https://root.cern/install/#install-via-a-package-manager
[3] *CutLang GitHub Repository:* https://github.com/unelg/CutLang
[4] *ATLAS Open Data:* https://atlas-opendata.web.cern.ch/atlas-opendata/samples/2020/GamGam/