IRIS-HEP Fellowship Proposal

Project: Advancing the Hist library

Project Area: Analysis Systems

Aman Goel
aman.goel185@gmail.com | University of Delhi, India

Mentor: Henry Schreiner
henryfs@princeton.edu | Princeton University

1 Background

Hist is a powerful Histogramming tool for analysis based on boost-histogram (the Python binding of the Histogram library in Boost). It is a friendly analysis-focused project that uses boost-histogram as a backend to do the work, but provides plotting tools, shortcuts, and new ideas.[1] It provides everything that boost-histogram provides and has enhancements such as:

- Augmentation of axes with names
- Augmentation by class with reduced typing construction
- UHI+ implementation
- Quick plotting routines
- Extended histogram features
- New modules
- In-notebook representation[2]

These features have enabled fast and efficient histogramming,[3] which can be further enhanced by implementing new features that would benefit analyses.

2 Proposed Project

To advance the Hist library further, this project proposes the implementation of various features, testing, improved documentation and tutorials. This would help in improving the performance, usability, accessibility and scope of functionality of the library. The features that would be implemented are as follows:

1. **Stacked Histograms**

   - A function that would plot several histograms in a stack
   - *HistStack* would hold multiple histograms in which the axes are required to match
   - API implementation should be sufficient since it is supported in mplhep
   - Would address the issues *#34* and *#169*

   Stacked Histograms would help in making the histograms more expressive and allow the representation of various categories of data involved.

2. **Interpolator**

   - A class that would be able to interpolate histograms
   - It would take a histogram and would return the interpolated value at that point
   - Various interfaces for different kinds of interpolation such as `hist.interp.Linear(h)` and `hist.interp.Cubic(h)`
   - Would address the issue *#165*

   The interpolator would help in better estimation and interpretation of data.

3. **Statistical Functions**

   - Implementation of something similar to *TEfficiency from ROOT*
   - Used to handle efficiency histograms
   - Would compute the calculation of efficiencies and their uncertainties
   - Would address this *To do card* in *Hist plans*

   This would be useful in combining several efficiencies and in providing various statistical methods for calculating frequentist and Bayesian confidence intervals.[4]

4. **Better Serialization**

   - Implementation via *Aghast*
   - Would enable the saving and loading of *ROOT* histograms
   - Would address the issue *#2* and this *To do card* in *Hist plans*

   This would enable cross-platform development and help push Aghast forward too.

5. **Integration with fitters**

   - Implementation to ensure proper interaction with fitters
   - Can be done for various fitters such as *ZFit*, *PyHF* and *GooFit*
   - Would address this *To do card* in *Hist plans*

   This would help in better interaction amongst Hist and various fitters.

6. **Implicit Multithreading**

   - To be based on the user's hardware
   - An ideal default can be estimated if `threads=None`, based on problem size and threads available
   - Would address this *To do card* in *Hist plans*

   This would help in improving performance while keeping in mind the constraints of the user's hardware.

7. **Documentation and Tutorials**

   - Improvement of the project documentation upon discussion with the mentor
   - Would update the documentation and add detailed instructions
   - Implementation of `doctest` can be done
   - Would address the issues *#111* and *#155*
   - More improvements can be made if feasible as follows:
     - Cover more setups and detailed guide in the installation section
     - A complete index inspired from Astropy[5]
     - Acknowledgement guidelines referring to *DOI* inspired from Astropy[6]
     - Cover as many features and functionalities as possible in the demo
     - Detailed About, Current Status and Future Prospects section in the documentation inspired from poliastro[7]
     - References section inspired from poliastro[8]

   This would help in increasing the accessibility, readability, usability and user as well as developer experience of Hist.

3 Timeline

Time Zone: Indian Standard Time - UTC +5:30

Commitments: I do not have any prior commitments for the summer, nor any planned vacations and will be able to continuously devote 5-6 hours each day ($\sim$ 40 hours per week) for the entire summer. My work hours are flexible, and I can work for more than 40 hours a week, if required. I will be present for all the teleconferences/discussions, if any, by adjusting my time to that which would suit the mentor.

Exams: I will have my end semester examinations during the third and fourth weeks of May (might be affected due to COVID-19). All the time lost (if any) during these two weeks will be made up for in the following weeks.

As my seventh semester at college begins in August (might be affected due to COVID-19), I may not be able to devote time in the mornings (IST) but would definitely be available later in the evening/night (IST). Rest assured, the time difference will not be a problem.

Week(s)	**Plan of Action**
1 - 2	– Familiarization with the codebase – Fix small issues to get an idea of the workflow – Read developer documentation and understand the community environment – Discuss the plan of action with the mentor and the community
3	– Work on the implementation of *Stacked Histograms* – Write tests for the function and document the progress
4 - 5	– Work on the implementation of *Interpolator* – Write tests and document the progress – Clean up previous code and fix bugs, if any – Keep buffer time in case of changes in the timeline
6	– Work on the implementation of *Statistical Functions* – Write tests and document the progress
7	– Work on the implementation of *Better Serialization* via *Aghast* – Write tests and document the progress
8 - 9	– Work on *Integration with fitters* upon discussion with the mentor – Write tests and document the progress – Clean up previous code and fix bugs, if any
10	– Work on the implementation of *Implicit Multithreading* – Write tests and document the progress
11 - 12	– Work on improving *Documentation and Tutorials* – If time permits, implement more features upon discussion with the mentor – Clean up code and fix bugs, if any

4 Student Background

About: I am Aman Goel, a junior from Cluster Innovation Centre, University of Delhi, India currently pursuing B.Tech in Information Technology & Mathematical Innovations with a minor in Management. I have been passionate about technology, space and physics since I was a kid. I have been particularly intrigued by Higgs boson and hence came across LHC. I am proficient in Python and hence, I am currently volunteering to teach Python in Stanford University's Code in Place 2021.

Relevance: My interest in this project stems from the fact it lies at the intersection of physics, technology and mathematics. I believe it would help me pursue my passion and enhance my knowledge. It would also allow me to make meaningful contributions to the community.

I have previously worked with Matplotlib[9] for plotting various visualizations. I have studied statistics and have done a Statistical Analysis project as a part of my coursework. I have also worked remotely before and am equipped to collaborate and communicate efficiently for projects virtually. I have sound experience with Linux, git and open source which would aid me to pursue this project.

I want to pursue my masters in the field of physics and computing, and I believe this would be the perfect opportunity to help me take a step in the same direction. It would help me understand more about data manipulation, particle physics, computing and mathematics which I believe are quite essential to learn.

References

[1] Welcome to Hist's documentation! (n.d.). Retrieved from https://hist.readthedocs.io/en/latest/index.html

[2] Scikit-Hep. (n.d.). Scikit-hep/hist. Retrieved from https://github.com/scikit-hep/hist

[3] Open IRIS-HEP fellow projects. (n.d.). Retrieved from https://iris-hep.org/fellow_projects.html

[4] TEfficiency Class Reference. (n.d.). Retrieved from https://root.cern.ch/doc/master/classTEfficiency.html

[5] Index. (n.d.). Retrieved from https://docs.astropy.org/en/stable/genindex.html

[6] Acknowledging or Citing Astropy. (n.d.). Retrieved from https://www.astropy.org/acknowledging.html

[7] About poliastro. (n.d.). Retrieved from https://docs.poliastro.space/en/stable/about.html

[8] References. (n.d.). Retrieved from https://docs.poliastro.space/en/stable/references.html

[9] Visualization with Python. (n.d.). Retrieved from https://matplotlib.org/