Proposal for GSoC 2020

Hist: histogramming for analysis powered by boost-histogram

About Me

BASIC INFO

- **English Name:** Nino Lau
- **Chinese Name:** 刘硕 (Shuo Liu)
- **Email:** ninomyemail@gmail.com
- **GitHub Account:** LovelyBuggies
- **Website:** lovelybuggies.com.cn
- **Timezone:** +8:00 GMT|UTC

EDUCATION

- **Sun Yat-sen University**, Guangzhou, China
  School of Data and Computer Sciences, expected graduation in July, 2020

CONTRIBUTION

- **hist: view my involvements in this repo**
  - Upgrade the test environment by conda (PR #31 for Issue #29)
- **boost-histogram: the contributor -- view my involvements**
  - Correct some small typos (PR #315)
  - Correct some syntax errors (PR #307, PR #317)
  - Fix the setup problem (PR #345 for Issue #344)
  - Forbid float weights for int storage (draft PR #346 for Issue #289)
  - Remove the redundant parts (PR #349)
- **Nino-hist: the author of Nino-hist -- draft for hist plans**
  - Associate meanings to hist axes metadata
  - Allow filling histograms and getting items by names
  - Design the themed and customized pull plot methods
  - Integrate external dependencies
- **aghast: view my involvements in this repo**
  - Solve the "KeyError" bug (PR #31 for Issue #29)
  - Upgrade the test environment by conda (PR #31 for Issue #30)
  - Better aghast's coding style by pre-commit (PR #33 for Issue #34)
- **mplhep: view my involvements in this repo**
  - Correct some syntax errors (PR #101)
- **vector: view all my involvements in this repo**
  - Set up notebooks environment (PR #9 for Issue #8)

RELATED EXPERIENCE

- IEEE member (#96688161), two-year research experience, familiar with Matplotlib plotting and data analysis
- Research assistant at CASIA in 2018
- Research intern in Inplus Lab, Sun Yat-sen University
- Active member in Scikit-HEP GitHub community
- Participant in 2018 MCM (The Mathematical Contest in Modeling)

Project

Project Name

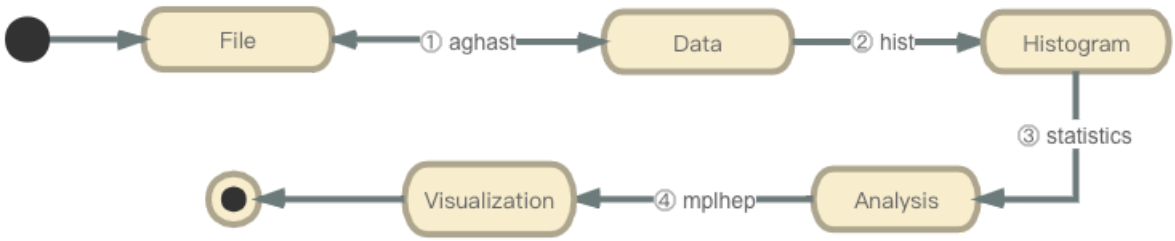
Hist: histogramming for analysis powered by boost-histogram.

Project Description

The Scikit-HEP project is a collection of several dozen packages intended to facilitate the use of Python in High Energy Physics (HEP), started in 2016. One of the major fronts of development is in histogramming; a majority of HEP analysis is heavily reliant on histograms. To this end, a new Python package was introduced for histogramming in Scikit-HEP, boost-histogram, developed in close collaboration with the author of the Boost C++ libraries' Histogram package. This package is intended to be a core package for histogramming with no dependencies, much like numpy is for data structures.

However, Scikit-HEP is a large toolset, it would be laborious for users to learn and use these packages. In addition, some packages of Scikit-HEP are still in heavy development and their documentation is not exhaustive enough. Thus, we are preparing an implementation of an analysis-friendly package, called hist, built on top of boost-histogram and providing commonly expected histogramming features. This package would allow a wider range of dependencies and features not admissible in the core package.

The flowchart of hist is shown as follows:



Project Outline

1. **File & Data**
   aghast provides communication between histogramming libraries, it would be used as a serialization and deserialization engine to support reading and writing data to ROOT files in hist design. Currently aghast is in early beta, and does not natively support boost-histogram. We will integrate boost-histogram support into aghast, and will provide a user-friendly method to save and load histograms without a user knowing about aghast.

2. **Data to Histogram**
   After obtaining data, we need to histogram data to manipulate it. hist is based on boost-histogram package, we create hist modules by deriving them from those in boost-histogram. We would introduce some new features to its super classes to facilitate HEP analysis.

a. Axes

Based on boost-histogram axes, we would add some new axis shortcuts, such as the bool axis type, and their manipulations. We would also customize the histogram axes by associating meanings to axis metadata (names, titles, units), and enable some methods corresponding to the name property. A draft of this is already present at Nino-hist/User_Guide/Axes.

b. Histogram

Based on boost-histogram, we would introduce two customized histograms to our hist package, i.e., NamedHist and Hist. NamedHist enforces manipulations by names. For example, we are allowed to fill the NamedHist histogram by the names of its axes, or get the axis items with names. Hist focuses on the use of analysis. We could plot the observation and prediction in a Hist object. More histogram designs would be implemented if needed. A draft of hypothetical histograms of hist is already present at Nino-hist/User_Guide/NamedHist and Nino-hist/User_Guide/Hist.

3. Analysis Shortcuts

hist is analysis-oriented, so a crucial point of our job is to provide some analysis shortcuts. Many ideas have been proposed: to add smart reprs for IPython; to enable the command-line histograms for streams of numbers; to allow complex number for location search; to support Python sum; to allow auto full-range slicing, etc. In the future, we would come up with more ideas in hist plans and implement them to facilitate HEP analysis.

4. External Packages Integration

Scikit-HEP provides many wonderful tools for data analysis, such as GooFit, zfit, Pyhf, etc. We need to ensure proper interactions with those tools for data analysis. We can come up with a numpy buffer based interface for communicating histogram data between libraries. We do not rule out tools other than Scikit-HEP, if they show good functionalities. A draft of external integration for hist is already present at Nino-hist/User_Guide/Integration.

a. Fitting

GooFit and zfit are affiliated projects of Scikit-HEP. GooFit is a massively-parallel framework for maximum-likelihood fits, implemented in CUDA/OpenMP. We are aiming to use GooFit to allow multi-thread histogramming based on the problem size and threads available and to analyze data via maximum-likelihood fitting. Zfit is a model manipulation and fitting library based on TensorFlow and optimized for simple manipulation of probability density functions, which could be an alternative fitter for hist.

b. Statistics

We are going to integrate some statistical models for hist. Pyhf is a statistical model for multi-bin histogram-based analysis. In addition, we might also integrate packages like iminuit, key features from ROOT.TEfficiency, and support Scipy for hist statistics.

5. Visualization

We need to show hist histograms to get more intuitive profiles of the histograms , i.e., the bins and weights. We plan to show the histograms via mplhep and Matplotlib. We will create some useful plots like pull plot and make them flexible to use, i.e., simple to start but easy to customize. A draft of the pull plot for hist is already present at Nino-hist/User_Guide/Hist.

a. Matplotlib

Matplotlib is a comprehensive library for creating visualizations in Python. We would build some common used plotting functions in NamedHist and Hist based on Matplotlib during this project.

b. mplhep

mplhep is a set of helpers for matplotlib to more easily produce plots typically needed in HEP. We need to guarantee hist could communicate with mplhep properly for visualize histograms. As far as I know, numpy could be a great medium. We could transform our histograms to numpy arrays with meanings associated. Then we could use mplhep to visualize them.

6. Coding Standard

We need to pay more attention to coding standards during the development. I have read Martin's Clean Code before and know the significance of a good coding style. We would strictly obey the development rules in the open source community. For example, we should not commit unrelated changes to an issue and make sure pull requests have a clean and readable code style, and we should follow the TDD rules. The project already has some static checks, so I think we would not run into problems during this part. A draft developing guideline is already shown at Nino-hist/Development and some drafted test files are present in Nino-hist/tests.

7. Documentation

We are going to make detailed documents for hist, including the documentation of hist, some blog posts for hist, and a summary article for GSoC.

a. hist's documentation

For the documentation of our hist package, we plan to write the documents in numpy-style. Specifically, we need to use Sphinx to build hist's documentation and host our projects on Read the Docs. Except for the normal Sphinx extension, we also need to add nbsphinx extension to support notebooks on Read the Docs. Our documentation would include but not limit to usage, user guide, examples, references, etc. A demo website Nino-hist has been made.

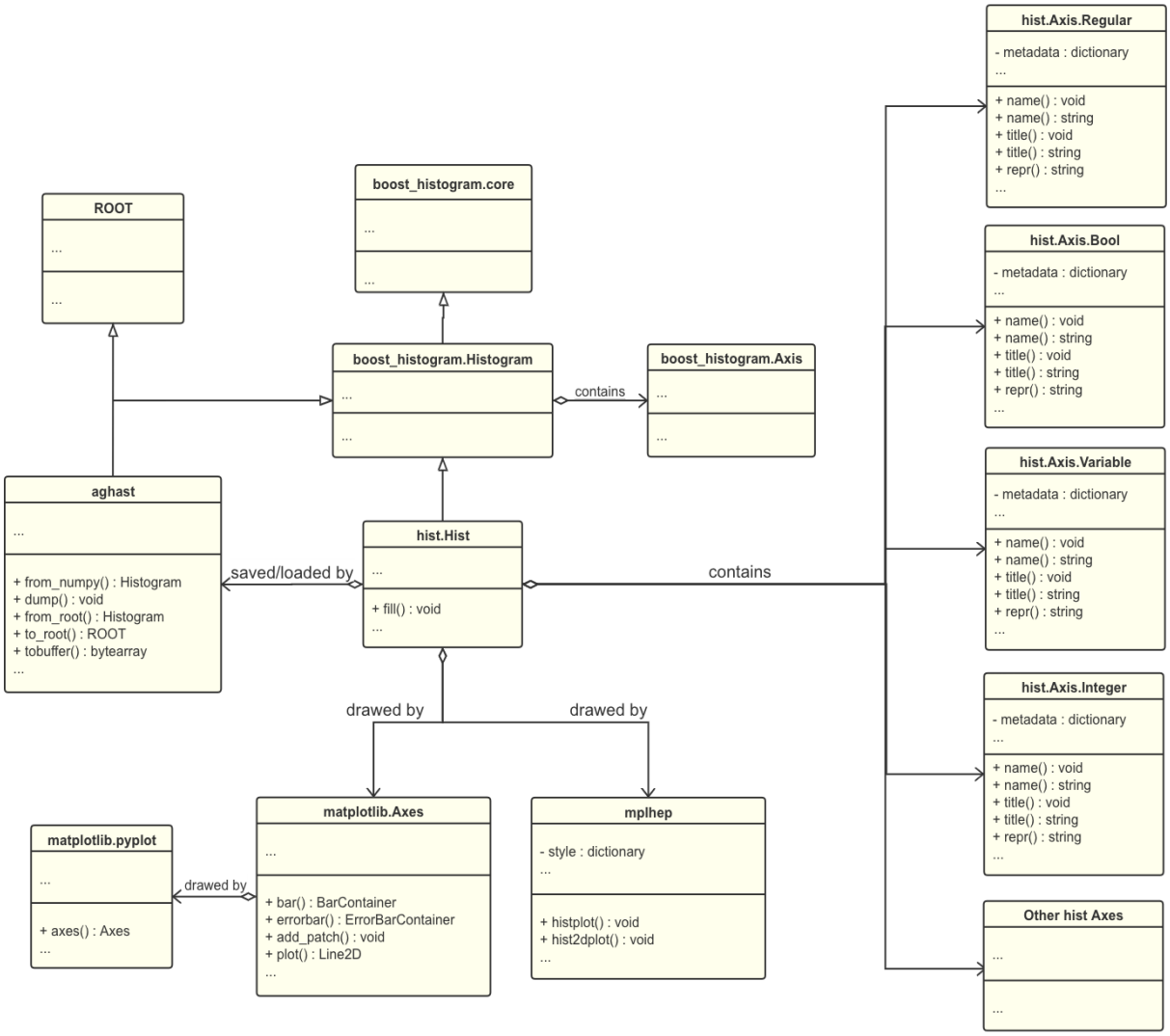
b. hist's pages

Both boost-histogram and hist do not have development logs, currently. Some blog posts might count, too. So we prepare to create the website pages for hist and make some posts to record the important details in our development.

c. Summary article

After the coding part is completed, a summary article would be written to describe our achievements. The article would include the whole procedure of the HEP analysis and details about our implementation.

Project UML



The dependencies attributes, methods, and derived properties as ellipsis.

Timeline

Date	Progress
April 28 - May 19	Community Bonding Period • Figure out the features and interaction logic inside our project • Discuss with mentors about the new shortcuts • Explore other Scikit-HEP modules and think about the possibility of integrating them to our work • Read about the Clean Code and TDD principles • Get familiar with community rules (on GitHub, Gitter, StackOverflow)

Date	Progress
May 20 - June 2	hist Package Implementation & Test • Add name and title to hist axis • Construct hist class and allow its manipulation by keyword arguments • Design unit tests for the new features
June 3 - 15	mplhep Interface Implementation & Test • Allow mplhep to draw hist histograms • Design the themed pull plot method • Design the customized pull plot method • Design unit tests for the new features
June 16 - 20	Evaluation Period • Create Read the Docs for hist package • Provide examples and API reference of documentation • Discuss with mentors about the next steps
June 21 - 30	Analysis Shortcuts Implementation & Test • Add shortcuts for hist • Design unit tests for the new features
July 1 - 12	External Package Integration & Test • Allow hist interact with the fitters properly • Allow hist interact with some statistical models properly • Design unit tests for the integration
July 14 - 18	Evaluation Period • Provide examples and API reference of documentation • Discuss with mentors about the next steps
July 19 - August 10	aghast Interface Implementation & Test • Support de/serialization to ROOT files by Aghast/uproot • Design unit tests for the new features • Provide examples and API reference of documentation
August 11 - 18	Students Submit Code and Final Evaluations • Register hist package on PyPI • Write a summary article throughout the project

The actual schedule may vary slightly according to the progress.

Extra Information

Working Time

I would be based in Beijing, China during this summer. Therefore, I would be working in GMT +8 timezone. I believe it would take about 10 weeks for me to complete the project. But before student projects would be announced in early May, I could have a head start with some early preparation. As for the working time, I think it would be good for me to work 40~50 hours a week.

Reason for Participation

In the past two years of research experience, I have witnessed the importance of numpy, Scipy, Matplotlib, and other toolkits for scientific research. The research needs for different disciplines require us to design some professional yet sophisticated packages. However, excessive programming and development tasks largely delay the research progress and distract researchers. I have always aspired to design and implement convenient interfaces or toolkits to facilitate research (especially for disciplines which do not require proficiency in programming). hist will design a user-friendly toolkit for HEP research, and that is why I consider hist project greatly valuable.

In addition, I have dreamed of being a great developer since the first day I learned programming. GSoC provides me a chance to make contributions to open source projects, with mentorship from great developers all over the world. This is my first time to develop open-source projects. During this period, I have no other large tasks scheduled and have plenty of time for this project. I cherish this opportunity and would definitely put my 100% to devote to this project!