`boost-histogram` and `hist` Roadmap

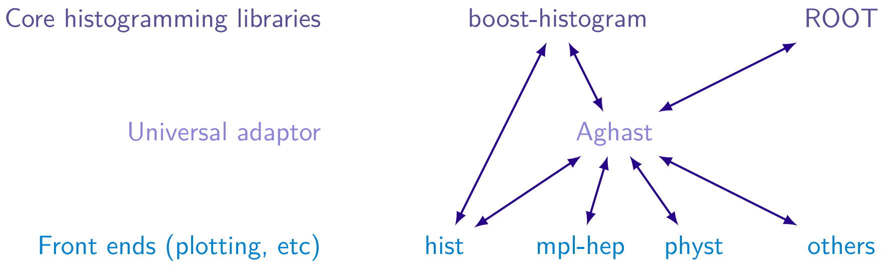
Henry Schreiner

September 12-13, 2019

Institute for Research & Innovation in Software for High Energy Physics

Princeton University

Future of histograms in Python



Intro to Boost.Histogram C++14

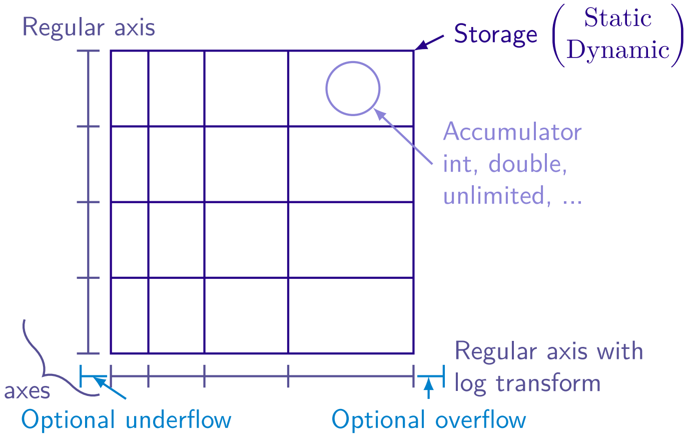
- Multidimensional templated header-only histogram library: /boostorg/histogram
- Designed by Hans Dembinski, inspired by ROOT, GSL, and histbook

Histogram

- Axes
- Storage

Axes types

- Regular, Circular
- Variable
- Integer
- Category



boost-histogram (Python)

/scikit-hep/boost-histogram

- 0-dependency build (C++14 only)
- State-of-the-art PyBind11
- 280+ unit tests run on Azure on Linux, macOS, and Windows
- Binary wheels on Azure for all major platforms
- Read the docs (in progress)

Design

Flexibility

Speed

Distribution

Design

Resembles the original Boost.Histogram where possible, with changes where needed for Python performance and idioms.

C++

```cpp
#include <boost/histogram.hpp>
namespace bh = boost::histogram;

// The make_ can be dropped in C++17
auto hist = bh::make_histogram(
  bh::axis::regular<>{2, 0, 1, "x"},
  bh::axis::regular<>{4, 0, 1, "y"});

hist(.2, .3); // .fill being added
hist(.4, .5); // in Boost 1.72
hist(.3, .2);
```

Python

```python
import boost.histogram as bh


hist = bh.histogram(
  bh.axis.regular(2, 0, 1, metadata="x"),
  bh.axis.regular(4, 0, 1, metadata="y"))

hist.fill(
    [.2, .4, .3],
    [.3, .5, .2])
```

Design: Manipulations

Combine two histograms

```
hist1 + hist2
```

Scale a histogram

```
hist * 2.0
```

Sum a histogram contents

```
hist.sum()
```

Access an axis

```
axis0 = hist.axis(0)
axis0.edges() # The edges array
axis0.bin(1) # The bin accessors
```

Fill 2D histogram with values or arrays

```
hist.fill(x, y)
```

Convert to Numpy, 0-copy if possible

```
hist.view()
```

Unified Histogram Indexing

Access:

```
v = h[b]                        # Returns bin contents, indexed by bin number
v = h[loc(b)]                        # Returns the bin containing the value
```

Setting (planned):

```
h[...] = np.ndarray(...)        # Setting with an array or histogram sets the
                                # contents if the sizes match
h[b] = v                        # Values can be set, too
```

Unified Histogram Indexing (2)

Slicing:

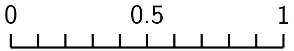
```
h == h[:]                                              # Slice over everything
h2 = h[a:b]                    # Slice of histogram (includes flow bins)
h2 = h[:b]                                     # Leaving out endpoints is okay
h2 = h[loc(v):]                 # Slices can be in data coordinates, too
h2 = h[::project]                                      # Projection operations
h2 = h[::rebin(2)]                           # Modification operations (rebin)
h2 = h[a:b:rebin(2)]              # Modifications can combine with slices
h2 = [a:b, ...]                       # Ellipsis work just like normal numpy
```

Slicing (planned)

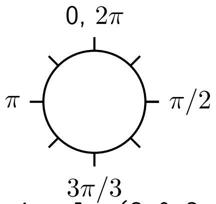
```
h2 = h[a:b:project]       # Adding endpoints to projection operation removes
                          # under or overflow from the calculation
h2 = h[0:end:project]          # Projection without flow bins, special tag
```

Flexibility: 22 axis types

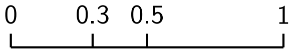
- `regular`
  - `uoflow`, `uflow`, `oflow`, `noflow`, `growth`
- `regular_` +
  - `log`, `sqrt`, `pow`
- `circular`
- `integer`
  - `uoflow`, `uflow`, `oflow`, `noflow`, `growth`
- `variable`
  - `uoflow`, `uflow`, `oflow`, `noflow`
- `category`
  - `int` or `str`, `growth`



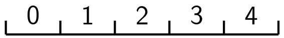
`bh.axis.regular(10,0,1)`



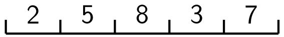
`bh.axis.circular(8,0,2*np.pi)`



`bh.axis.variable([0,.3,.5,1])`



`bh.axis.integer(0,5)`



`bh.axis.category[2,5,8,3,7])`

Flexibility: 7 storage types

- `bh.storage.int`
- `bh.storage.double`
- `bh.storage.unlimited`
- `bh.storage.atomic_int`
- `bh.storage.weight`
- `bh.storage.profile`
- `bh.storage.weighted_profile`

Plans (See #18)

- Finish UHI implementation
- Add non-double fill
- Clean up some bugs/missing functionality with access
- Add `from_numpy` and numpy style shortcut(s)
- Release to PyPI

Release

- Becoming stable enougth for internal use in AS!
- Planned release before PyHEP in mid October

Bikeshedding (API discussion)

Let's discuss API! (On GitHub issues or gitter)

- Download: `pip install boost-histogram` (Release before PyHEP)
- Use: `import boost.histogram as bh`
- Create: `hist = bh.histogram(bh.axis.regular(12,0,1))`
- Access values, convert to numpy, etc.

Documentation

- The documentation will also need useful examples, feel free to contribute!

A slide about hist

`hist` is the 'wrapper' piece that does plotting and interacts with the rest of the ecosystem.

Plans

- Easy plotting adaptors (mpl-hep)
- Serialization formats (ROOT, HDF5)
- Auto-multithreading
- Statistical functions (Like TEfficiency)
- Multihistograms (HistBook)
- Interaction with fitters (ZFit, GooFit, etc)
- Bayesian Blocks algorithm from SciKit-HEP
- Command line histograms for stream of numbers

Call for contributions

- What do you need?
- What do you want?
- What would you like?

Join in the development! This should combine the best features of other packages.

Discussion

These are just a few question to facilitate discussion.

- Do you plan to use boost-histogram and or hist as part of another package?
- Do you have any unusual histogramming needs?
- What part of `boost-histogram` sounds most useful/exciting?
- Does the boost-histogram API look reasonable?
- What are the analysis/plotting features you need in Hist?

Support

- Supported by IRIS-HEP, NSF OAC-1836650