Recent developments in histogram libraries

Hans Dembinski\textsuperscript{1} Jim Pivarski\textsuperscript{2} Henry Schreiner\textsuperscript{2}

November 7, 2019

Max Planck Institute for Nuclear Physics, Heidelberg, Germany\textsuperscript{1}
Princeton University, Princeton, USA\textsuperscript{2}
Histogram libraries (Python)

- theodoregoetz
- matplotlib-hep
- rootplotlib
- pyhistogram
- Cassius
- YODA
- qhist
- fast-histogram
- pygram11
- hdrhistogram
- PyROOT
- histogram
- SimpleHist
- paida
- multihist
- SVGFig
- coffea
- physt
- Vaex
- HistBook
Recent developments in histogram libraries

November 7, 2019
Scikit-HEP histogramming plan (Python)

- **boost-histogram**: Fast filling and manipulation (core library)
- **hist**: Simple analysis frontend
- **aghast**: Conversions between histogram libraries
- **UHI** (Unified Histogram Indexing): Cross-library indexing proposal
Boost.Histogram C++14

- /boostorg/histogram
- Designed by Hans Dembinski
- Header only, with header only Boost
- Heavily unit-tested: 100% line coverage

Features
- Static / Dynamic storage (can avoid allocation!)
- Static fill becomes 57 lines of vectorized assembly
- Supports platforms without exceptions or RTTI
- Adding, scaling, slicing, rebinning, projections...
- High performance filling and bin iteration
- Unique memory-efficient dynamic storage

Customizable:
- storage
- allocator
- accumulator
- axes
- axis metadata
- axis transform
Boost.Histogram C++14

- /boostorg/histogram
- Designed by Hans Dembinski
- Header only, with header only Boost
- Heavily unit-tested: 100% line coverage

Features

- Static / Dynamic storage (can avoid allocation!)
- Static fill becomes 57 lines of vectorized assembly
- Supports platforms without exceptions or RTTI
- Adding, scaling, slicing, rebinning, projections...
- High performance filling and bin iteration
- Unique memory-efficient dynamic storage

Customizable:

- storage
- allocator
- accumulator
- axes
- axis metadata
- axis transform

Accepted in Boost 1.70!

Dembinski, Pivarski, Schreiner

Recent developments in histogram libraries

November 7, 2019
Boost.Histogram C++14

- [link](https://github.com/boostorg/histogram)
- Designed by Hans Dembinski
- Header only, with header only Boost
- Heavily unit-tested: 100% line coverage

**Features**
- Static / Dynamic storage
- Static fill becomes 57 lines of vectorized assembly
- Supports platforms without exceptions or RTTI
- Adding, scaling, slicing, rebinning, projections...
- High performance filling and bin iteration
- Unique memory-efficient dynamic storage

**Customizable:**
- storage
- allocator
- accumulator
- axes
- axis metadata
- axis transform

**Will not go away!**
Boost.Histogram C++14: Histogram concept

**Histogram**
- Axes
- Storage

**Axis types**
- Regular (circular)
- Variable (circular)
- Integer (circular)
- Category

**Accumulators**
- Sum
- Mean

Single class replaces TH1T, TH2T, TH3T, THnT, TProfile, TProfile2D, TProfile3D and is more general!
#include <boost/histogram.hpp>
#include <boost/histogram/ostream.hpp>
#include <random>
#include <iostream>

int main() {
    namespace bh = boost::histogram;

    auto hist = bh::histogram(bh::axis::regular{20, -3, 3}); // C++17 version

    std::default_random_engine eng;
    std::normal_distribution<double> dist{0, 1};
    for(int n = 0; n < 10'000; ++n)
        hist(dist(eng));

    std::cout << hist << std::endl;
    return 0; }

Boost.Histogram C++14: Example output

```cpp
histogram(regular(20, -3, 3, options=underflow | overflow))
```

```
+----------------------------------------------------------+
| [-inf, -3) 9  |
| [ -3, -2.7) 19 |=
| [-2.7, -2.4) 36 |==
| [-2.4, -2.1) 110 |=====|
| [-2.1, -1.8) 191 |========|
| [-1.8, -1.5) 275 |=============|
| [-1.5, -1.2) 518 |=============================|
| [-1.2, -0.9) 644 |==============================|
| [-0.9, -0.6) 914 |================================|}
| [-0.6, -0.3) 1107 |==================================|}
| [-0.3, 0) 1183 |======================================|}
| [ 0, 0.3) 1185 |=======================================|}
| [ 0.3, 0.6) 1120 |=======================================|}
| [ 0.6, 0.9) 874 |=======================================|}
| [ 0.9, 1.2) 663 |=======================================|}
| [ 1.2, 1.5) 491 |=======================================|}
| [ 1.5, 1.8) 322 |=======================================|}
| [ 1.8, 2.1) 172 |=======================================|}
| [ 2.1, 2.4) 79 |=======================================|}
| [ 2.4, 2.7) 38 |=======================================|}
| [ 2.7, 3) 28 |=======================================|}
| [ 3, inf) 22 |=======================================|}
+----------------------------------------------------------+
```
• call is one at a time fill
• fill is new (1.72) array fill

• S is static (fixed types)
• D is dynamic (unlimited storage)
boost-histogram: Python bindings

• Boost.Histogram developed with Python in mind
  ▶ Original prototype bindings based on Boost::Python

• New bindings: /scikit-hep/boost-histogram
  ▶ 0-dependency build (C++14 only)
  ▶ State-of-the-art PyBind11
  ▶ Designed with the original author, Hans Dembinski

• Public beta 0.5.2 currently available, 0.6 coming soon
Design

- 500+ unit tests run on Azure on Linux, macOS, and Windows
- Stays close to Boost.Histogram with Pythonizations

C++17
#include <boost/histogram.hpp>
namespace bh = boost::histogram;

auto hist = bh::histogram(
    bh::axis::regular{2, 0, 1, "x"},
    bh::axis::regular{4, 0, 1, "y"});

hist(.2, .3); // Fill will also be
hist(.4, .5); // available in 1.72
hist(.3, .2);

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

\[ \text{hist1} + \text{hist2} \]

**Scale** a histogram

\[ \text{hist} \times 2.0 \]

**Sum** a histogram contents

\[ \text{hist} \text{.sum()} \]

**Access** an axis

\[
\begin{align*}
\text{ax} &= \text{hist} \text{.axes}[0] \\
\text{ax} \text{.edges} & \quad \# \text{The edges array} \\
\text{ax} \text{.centers} & \quad \# \text{Centers of bins} \\
\text{ax} \text{.widths} & \quad \# \text{Width of each bin}
\end{align*}
\]

\[ \text{hist} \text{.axes} \text{.centers} \quad \# \text{All centers} \\
\# \text{Etc.} \]

**Fill** 2D histogram with values or arrays

\[ \text{hist} \text{.fill(a1, a2, weights=[w1, w2], samples=[s1, s2])} \]

**Convert** contents to Numpy array

\[ \text{hist} \text{.view()} \]

**Convert** to Numpy style histogram tuple

\[ \text{hist} \text{.to_numpy()} \]

**Pickle** supported (multiprocessing)

\[ \text{pickle} \text{.dumps(hist, -1)} \]

**Copy/deepcopy** supported

\[ \text{hist2} = \text{copy} \text{.deepcopy(hist)} \]

**Numpy** functions provided

\[ \text{H, E} = \text{bh} \text{.numpy} \text{.histogram(...)} \]
Design: Unified Histogram Indexing (UHI)

The language here (bh.loc, etc) is defined in such a way that any library can provide them - “Unified”.

Access

\[ v = h[b] \]  # Returns bin contents, indexed by bin number
\[ v = h[bh.loc(b)] \]  # Returns the bin containing the value
\[ v = h[bh.underflow] \]  # Underflow and overflow can be accessed with special tags

Setting

\[ h[b] = v \]
\[ h[bh.loc(b)] = v \]
\[ h[bh.underflow] = v \]
Design: Unified Histogram Indexing (UHI) (2)

```python
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[bh.loc(v):]  # Slices can be in data coordinates, too
h2 = h[:bh.sum]  # Remove an axis by summation
h2 = h[0:5:bh.sum]  # Can add endpoints
h2 = h[:bh.rebin(2)]  # Modification operations (rebin)
h2 = h[a:b:bh.rebin(2)]  # Modifications can combine with slices
h2 = h[a:b, ...]  # Ellipsis work just like normal numpy
```

- Docs are here
- Description may move to a new repository
Flexibility: Axis types

- **bh.axis.Regular**
  - under/overflow
  - growth=True
  - circular=True
  - transform=Log()
  - transform=Sqrt()
  - transform=Pow(v)

- **bh.axis.Integer**
  - under/overflow
  - growth=True

- **bh.axis.Variable**
  - under/overflow
  - growth=True

- **bh.axis.Category**
  - int or str
  - growth=True

```python
bh.axis.Regular(10, 0, 1)
bh.axis.Regular(8, 0, 2*np.pi, circular=True)
bh.axis.Variable([0, .3, .5, 1])
bh.axis.Integer(0, 5)
bh.axis.Category([2, 5, 8, 3, 7])
```
Performance

- Factor of 2 faster than 1D regular binning in Numpy 1.17
  - Currently no specialization, just a 1D regular fill
  - Could be optimized further for common fills
- Factor of 6-10 faster than 2D regular binning Numpy
Distribution

If `pip install boost-histogram` fails, we have failed.

- Docker ManyLinux1 GCC 9.2: `/scikit-hep/manylinuxgcc`
- Used in `/scikit-hep/iMinuit`, `/scikit-hep/awkward1`
- Described on `iscinumpy.gitlab.io`, also see `/scikit-hep/azure-wheel-helpers`

Wheels

- manylinux1 32/64 bit, Py 2.7, 3.5, 3.6, 3.7
- manylinux2010 64 bit, Py 2.7, 3.5, 3.6, 3.7, 3.8
- macOS 10.9+ 64 bit, Py 2.7, 3.6, 3.7, 3.8
- Windows 32/64 bit, Py 2.7, 3.6, 3.7

Source

- SDist on PyPI
- Build directly from GitHub

Conda-Forge

- All (including 3.8) except 2.7 on Windows

```
python3 -m pip install boost-histogram
conda install -c conda-forge boost-histogram
```
**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 via aghast (ROOT, HDF5)
- Implicit multithreading
- Statistical functions (Like TEfficiency)
- Multihistogram plotting (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?
- Looking for a student interested in development

Join in the development! This should combine the best features of other packages.
aghast is a histogramming library that does not fill histograms and does not plot them.

/aghast

- A memory format for histograms, like Apache Arrow
- Converts to and from other libraries
- Uses flatbuffers to hold histograms
- Indexing ideas inspired the UHI

Available Binnings
- IntegerBinning
- RegularBinning
- HexagonalBinning
- EdgesBinning
- IrregularBinning
- CategoryBinning
- SparseRegularBinning
- FractionBinning
- PredicateBinning
- VariationBinning
Conclusions

- Boost.Histogram C++14 is a solid foundation
- boost-histogram for Python is a core histogramming library
- Hist will be developed next and is analysis friendly
- aghast is the glue to ROOT/everything else
Questions

Supported by

- IRIS-HEP, NSF OAC-1836650
- DIANA-HEP, NSF OAC-1558216, NSF OAC-1558233, NSF OAC-1558219
Storage types

- `bh.storage.Int`
- `bh.storage.Double`
- `bh.storage.Unlimited`
- `bh.storage.AtomicInt`
- `bh.storage.Weight`
- `bh.storage.Profile`
- `bh.storage.WeightedProfile`