Lorentz Vector Manipulation and Histogram Handling

IRIS-HEP Future Analysis Systems & Facilities Workshop

Henry Schreiner, 10-25-2020
Boost-Histogram
What exists

- Fast, Pythonic histograms in Python
- Built on Boost.Histogram using cutting-edge pybind11 & cibuildwheel
- Int, Double, Weight, & “Profile” storages (and more)
- Can be filled in threads, merged, manipulated, and stored
- Already supported in Uproot 4, mplhep, and histoprint
Boost-histogram
Aside: Update to pybind11 2.6.0

- Python 3.9 support, PyPy 7.3.x (2.7, 3.6, and 3.7 alpha) support
- Fixed warnings on latest AppleClang
- 40% faster accumulator fills, simpler implementation
- Segfaults when passing an object with a throwing repr fixed
- kwargs replaced older workarounds (partially at the moment)
- Using new public py::type instead of pybind11::detail usage
- Enhanced CMake support, finds conda and venv now, uses pybind11_find_import
- Using setuptools support from pybind11 (previously vendored)
Boost-Histogram
What remains

• Most of the planned work for 1.0 was done in 0.10 through 0.11.1
• What makes 1.0?
  • Protocol for Histograms (at least plotting)
  • to_numpy normalization with other packages
  • Slightly more UHI support
  • Slightly better integer storage support
  • Will end Python 2.7 support
Hist
What exists

• Extends boost-histogram with user friendly features
  • Named axes (optionally enforced)
  • Fast histogram construction
  • UHI+: simpler slicing and manip, especially in a notebook
  • Plotting: Simple adaptor to mplhep
  • Beautiful notebook reprs
• GSoC concluded, Python 3.6+ Hist 2.0 released
Reprs in Jupyter

Hist has custom reprs when displaying in a Jupyter.

```python
from hist import Hist
import numpy as np

Hist.new.Reg(50, 1, 2).Double().fill(np.random.normal(1.5, 0.3, 10_000))
```

This is an example of a pull plot:

```python
from uncertainties import unumpy as unp
def pdf(x, a=1 / np.sqrt(2 * np.pi), x0=0, sigma=1, offset=0):
    exp = unp.exp if a.dtype == np.dtype('O') else np.exp
    return a * exp(-((x - x0)**2) / (2 * sigma ** 2)) + offset

plt.figure(figsize=(10, 6))
h.project("S","pull_pull(pdf)
plt.show()
```

(The uncertainty is non-significant as we filled a great quantities of observation points above.)
Hist
What remains

• Anything that doesn’t fit in boost-histogram
  • Statistical functions
  • Bayesian blocks algorithm
• Design a cool logo
• Developing a Plotting Protocol
  • Next iteration coming soon
  • Feedback welcome, will help broaden and shape (exp. for fitters!)
    • [https://github.com/scikit-hep/boost-histogram/issues/423](https://github.com/scikit-hep/boost-histogram/issues/423)
Vector
What exists

• Basic examples of Lorentz vectors:
  • Core - standalone functions (usable by Numba)
  • Common - class structure, basis for others
    • NumPy - Holds arrays
    • Awkward - Usable in Awkward 1
    • Single (free) - useful for Numba
    • Numba - Works in Numba functions
Vector
What exists - 2

- Lorentz
  - Coordinate free (all) (can be overridden)
  - xyzt
- Alternate coordinate properties, like pt, eta, phi
- A few single values, like mag and mag2
- A vector-returning function, __add__ (vector - scalar __mul__ mostly done)
@numba.njit
def do_cool_stuff(input, output):
    for muons in input:
        output.begin_list()

        for i in range(len(muons)):
            output.begin_list()

            for j in range(i + 1, len(muons)):
                zboson = muons[i] + muons[j]

                output.begin_tuple(2)
                output.index(0)
                output.append(zboson)
                output.index(1)
                output.append(zboson.mag)
                output.end_tuple()

            output.end_list()

        output.end_list()
Vector
What remains

- Filling out the remaining scalar and vector functions
- Adding a Protocol to verify an implementation is complete
- Redesigning unit tests to share between impl and coord sys
- Filling out more coordinate systems
- Adding 2D & 3D vectors, handling rotations
  - Interactions between vectors, like boosting
- TensorFlow impl
Vector
Questions

• Clean, “one way to do it”, with optional mix-ins for “momentum”, etc?
• Selecting best names for core methods/properties
  • Protocol will be a good place to get feedback!
• Balance of “sharing code” with simple, separate implementations
  • Protocol can help with the latter!
• ROOT compatibility layer/mix-in
  • Already testing against ROOT using Conda-Forge ROOT