FELLOW:
JAY GOHIL

MENTORS:
HENRY SCHREINER
HANS DEMBINSKI

Features Extension, Inclusion & Rectification for boost-histogram

Objective

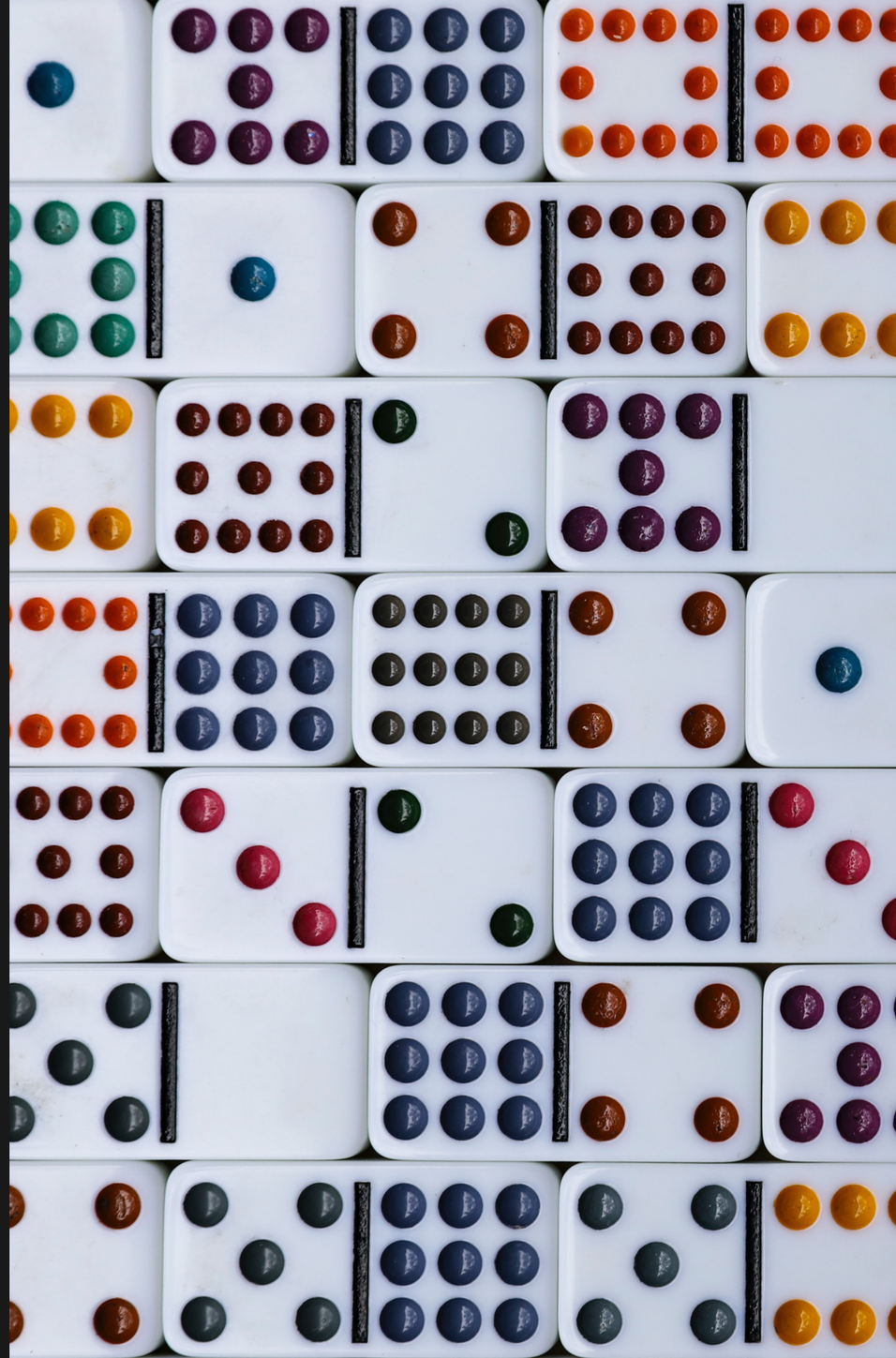
- Add new features to boost-histogram
- Rectify bugs on boost-histogram
- Add an accumulator in Boost.Histogram
- Work on documentation for both

Introduction

Boost.Histogram is one of the most extensive and powerful histogram libraries in C++ which provides easy-to-use, fast, and extensible multi-dimensional histograms and profiles.

boost-histogram is the python package that provides bindings for Boost.Histogram in python with plotting tools, operations, axes manipulation, and much more.

boost-histogram: Link
Boost.Histogram: Link



Better error for empty AND incorrect sample #782

- Addressed #734 issue.
- Added error messages for:
  - empty sample
  - incompatible type sample
  - incorrect dimension sample

```
import boost_histogram as bh
values = [10]*10
histogram = bh.accumulators.Mean()
histogram.fill(values, sample="")
```

Histogram Comparison (Draft) #778 #779

```
import boost_histogram as bh
import numpy as np
histogram1.allclose(histogram2)
```

Public #778

- Addresses #157 issue.
- Compare two histograms based on:
  - Values
  - Edges
  - Dimension
  - Storage type
  - Axes
- Added ufunc for numpy's allclose

Private #779

- Addresses #157 issue, but for internal use.
- All checks similar to #778.
- Has a boolean return type instead of pretty-string.

Minor Updates #760 #781 #783 #786

- Added storage_type function and property, with deprecation warning for _storage_type()
- Updated numpy.testing asserts with pytest.approx (with complete tests' swap under draft)
- Cleared reset() usage confusion with a small doc update

Patch Release v1.3.2

Version 1.3.2 Latest

Compare

henryiii released this 6 days ago · v1.3.2 · 0c8659c

1.3 is the final release series supporting Python 3.6 and manylinux1 - manylinux2010. The next release will move to non EoL Python and manylinux images only.

Changes

- Added `storage_type()` as public API #781, with pending deprecation for `_storage_type`. #786 #790
- Better errors generated for missing or incorrect sample to mean storage. #782
- Better error message when views are set with an incompatible array. #794

Bug fixes

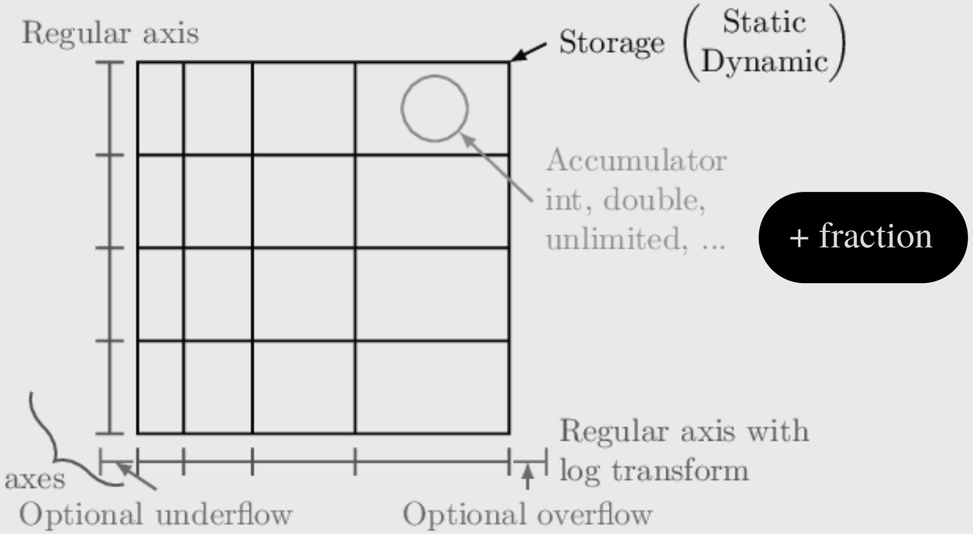
- Patch broken sum with fully empty (0 bin) axis. #718
- Fix zero range `bh.numpy.histogram` to match `numpy.histogram` behavior. #721
- Avoid triggering `__init__` when copying (better support for subclasses with custom init's). #759
- `IntCategory` now supports numbers larger than 2^{24} (now 2^{53}). #792
- Pick a subset now supported inside a larger expression. #793

Backend and docs

- Minor optimizations for UFuncs. #771
- Added Python 3.11 wheels. #789
- Include PyPy 3.9 binary wheels. #730
- Using pybind11 2.10 #767
- Explicit `reset()` documentation. #783
- Minor cleanup and further removal of a little Python 2 back-compat code.
- Warnings have better stacklevel settings.

Fraction Accumulator #361

- Added new Fraction accumulator on boost.histogram.



Fraction Accumulator #361

- Accumulator has the following:
  - successes() and failures()
  - count()
  - value()
  - variance()

```
successes() -> fetch quantity of success/true/1
failures() -> fetch quantity of failure/false/0
count() -> fetch total quantity
value() -> fetch value (fraction of successes)
variance() -> fetch BN based variance
```

Fraction Accumulator #361

- Accumulator has the following:
  - confidence_interval(), and of following default:
    - wald interval
  - Other external classes' intervals:
    - wilson interval
    - clopper pearson interval
    - jeffreys interval

A binomial proportion confidence interval is an **interval estimate** of a success probability **p**.

Wald Interval :

$$\hat{p} \pm z\sqrt{\frac{\hat{p}\,(1-\hat{p})}{n}}$$

Wilson Interval :

$$p \approx \frac{n_S + \frac{1}{2}z^2}{n + z^2} \pm \frac{z}{n + z^2}\sqrt{\frac{n_S\, n_F}{n} + \frac{z^2}{4}}$$

Fraction Accumulator #361

- Added tests for accumulator and intervals
- Added documentation
- Merged now!

```
/**
 Accumulate boolean samples and compute the fraction of true
samples.

 This accumulator should be used to calculate the efficiency or
success fraction of a
 random process as a function of process parameters. It returns
the fraction of
 successes, the variance of this fraction, and a two-sided
confidence interval with 68.3
 % confidence level for this fraction.

 There is no unique way to compute an interval for a success
fraction. This class returns
 the Wilson score interval, because it is widely recommended in
the literature for
 general use. More interval computers can be found in
`boost/histogram/utility`, which
 can be used to compute intervals for other confidence levels.
*/
```

Next Steps

- Merge histogram comparison
- Add fraction accumulator to boost-histogram
- Work on python and c++ end in open source capacity

Thank you for listening!