

plothist package

Plot and compare histograms in a scalable way and a beautiful style

Cyrille Praz , Tristan Fillinger 

01/07/24

PyHEP 2024



GitHub

pypi package

1.2.5

docs

main



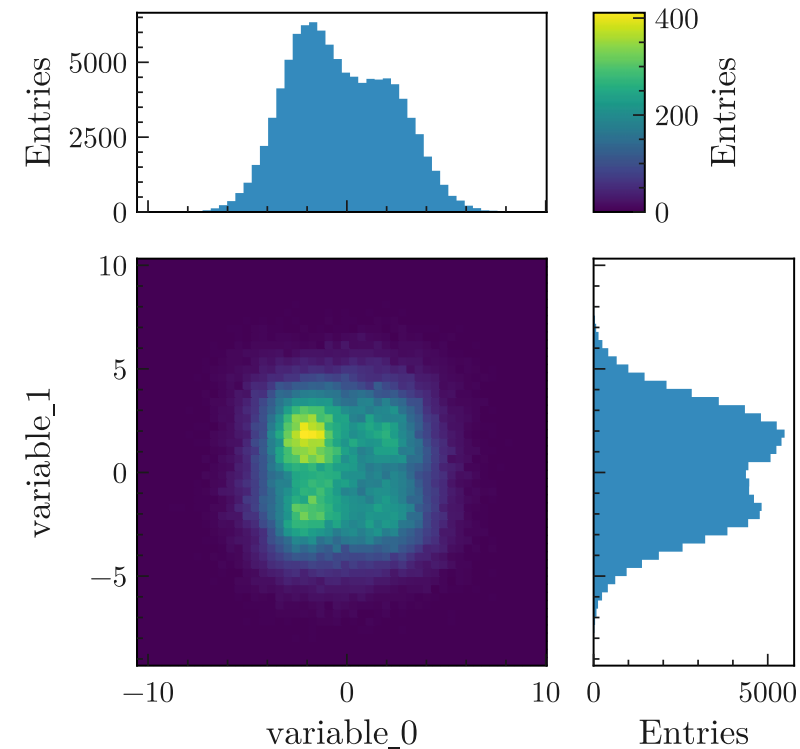
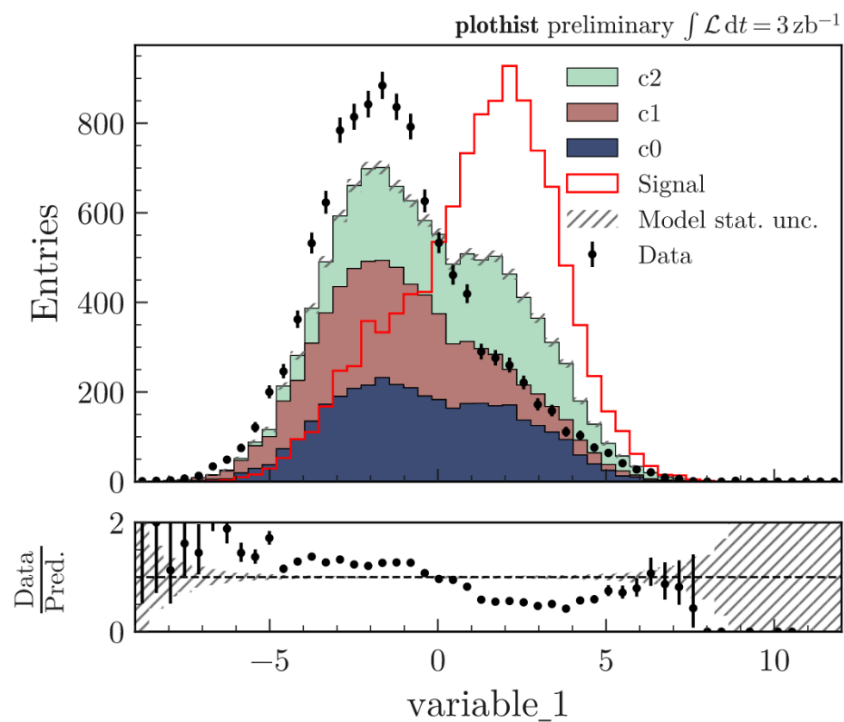
Discussions Ask



DOI 10.5281/zenodo.12160362

License BSD 3-Clause



code style black



Presentation of the basic functionalities of the package [\(this presentation\)](#)

- Concept
- 1D, 2D histograms
- High-energy physics example
- Variable registry

Live demonstration of the package [\(right after this presentation\)](#)

- Interactive Jupyter notebooks  
- High-energy physics examples
 - model made of functions
 - model made of histograms
- 2D histograms with variable registry

Concept

Goal of the package

- Provide tools to [make standard high-energy physics plots](#) in a [scalable](#) way and a [publication-ready](#) style, allowing analysts to [focus on Physics](#) rather than spending time on making and tuning plots.

Method

- [Wrapper functions](#) around [matplotlib](#) to plot [boost_histogram.Histogram](#) objects

Goal of the package

- Provide tools to [make standard high-energy physics plots](#) in a [scalable](#) way and a [publication-ready](#) style, allowing analysts to [focus on Physics](#) rather than spending time on making and tuning plots.

Method

- [Wrapper functions](#) around [matplotlib](#) to plot [boost_histogram.Histogram](#) objects

Main features

Style

- [Default style](#) is [publication-ready](#) (with little to [no effort](#))

Goal of the package

- Provide tools to **make standard high-energy physics plots** in a **scalable** way and a **publication-ready** style, allowing analysts to **focus on Physics** rather than spending time on making and tuning plots.

Method

- **Wrapper functions** around `matplotlib` to plot `boost_histogram.Histogram` objects

Main features

Style

- **Default style** is **publication-ready** (with little to **no effort**)

Scalability

- **Scalable wrt data size** by separating histogram creation from plotting, allowing batching/parallelism
- **Scalable wrt number of variables** by storing plotting parameters in a **variable registry**

Goal of the package

- Provide tools to [make standard high-energy physics plots](#) in a [scalable](#) way and a [publication-ready](#) style, allowing analysts to [focus on Physics](#) rather than spending time on making and tuning plots.

Method

- [Wrapper functions](#) around [matplotlib](#) to plot [boost_histogram.Histogram](#) objects

Main features

Style

- [Default style](#) is [publication-ready](#) (with little to [no effort](#))

Scalability

- [Scalable wrt data size](#) by separating histogram creation from plotting, allowing batching/parallelism
- [Scalable wrt number of variables](#) by storing plotting parameters in a [variable registry](#)

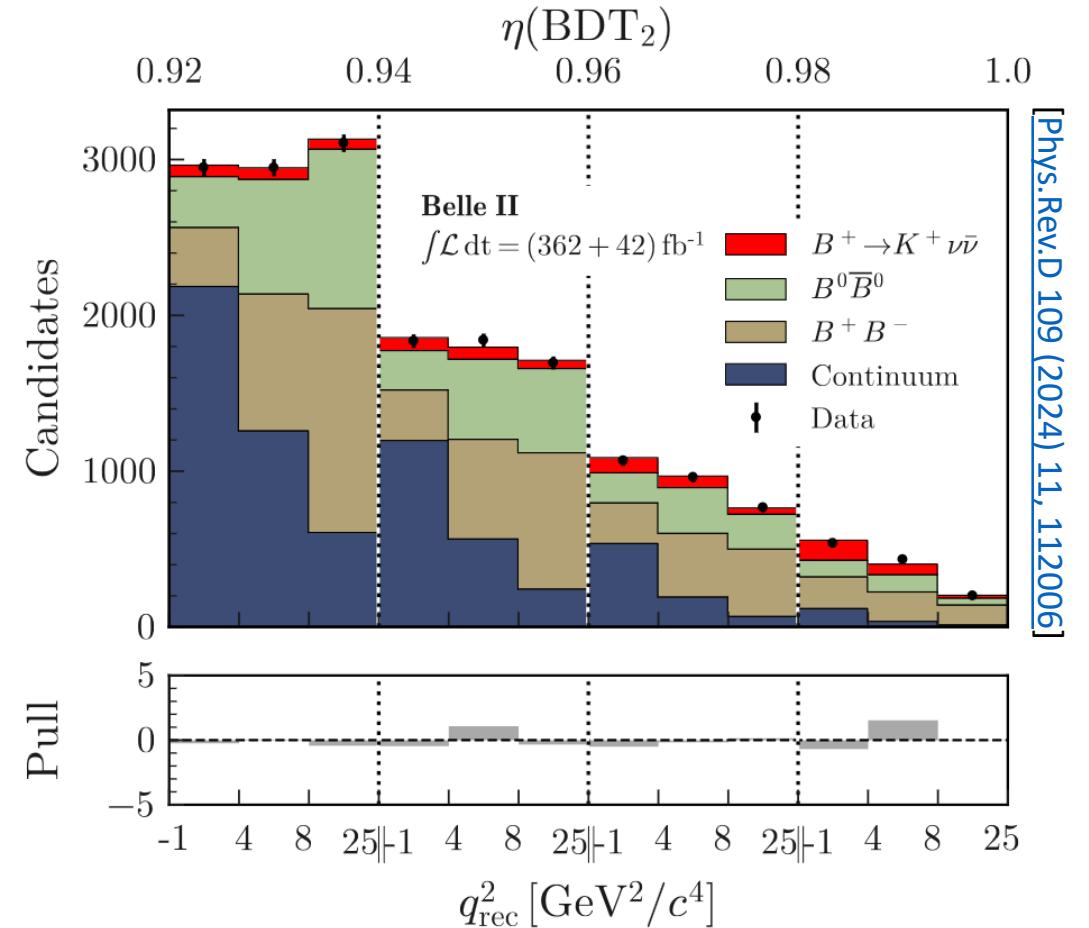
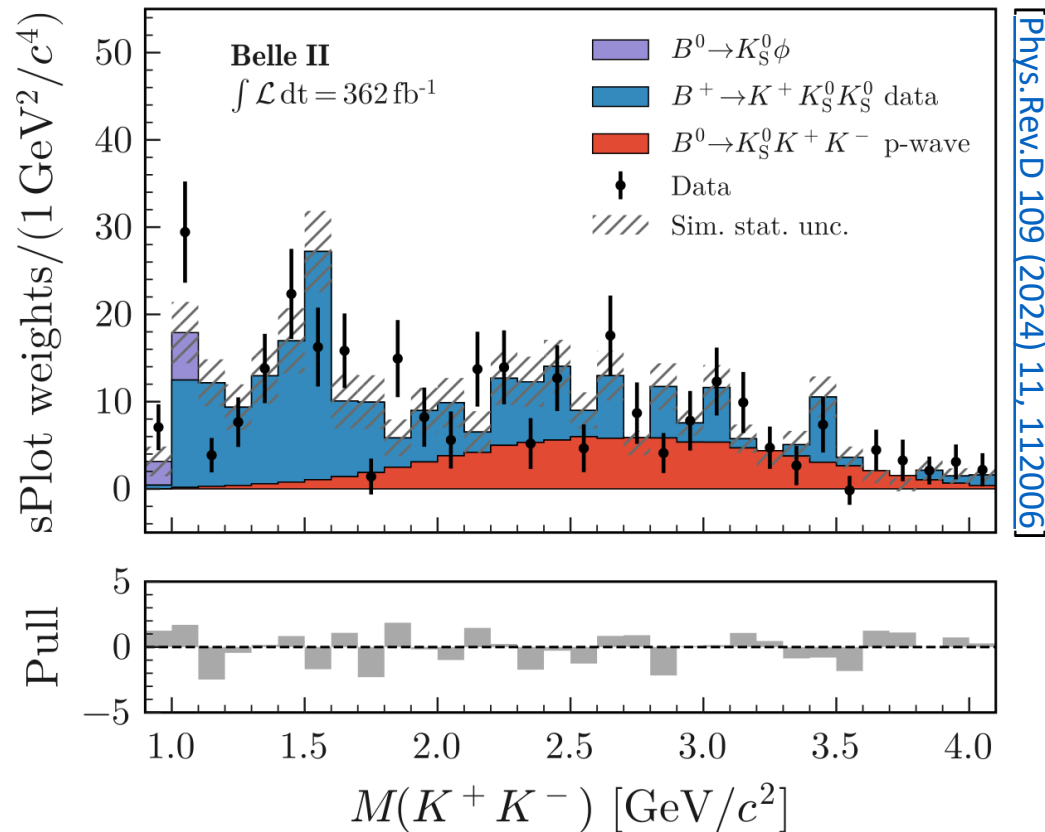
User-friendly

- A [gallery of examples](#) with complete codes
- Comprehensive and easy-to-navigate [documentation](#)
- [Installable](#) in one command line via [pip](#): `pip3 install plothist`

Default style

Style *already compatible* with **Physical Review Letters / Physical Review D** (with little to *no effort*)

Example from recent Belle II paper published in PRD



Functionality overview

🏠 plothist

latest

Search docs

Installation and update

Font installation

SIMPLE EXAMPLES

Plot 1D histograms

Plot 2D histograms

Plot functions

ADVANCED EXAMPLES

Plot and compare model and data

Other advanced examples

UTILITIES

Variable registry

Style and colors

Utility functions

Plot result of a fit

DOCUMENTATION

Example gallery

Package references

Notes on statistics

Documentation: <https://plothist.readthedocs.io>

🏠 / Example gallery

🔗 Edit on GitHub

Example gallery

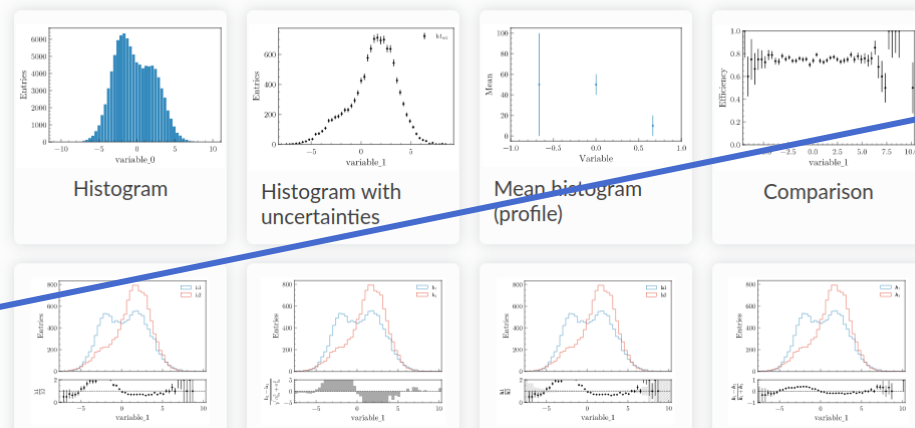
Gallery of images that are used in the doc.

Note

Click on an image to see the source code that generates it.

Plot 1D histograms

Gallery of images that are used in the 1D histogram section.



🏠 / Package references

🔗 Edit on GitHub

Package references

histogramming.py

`plothist.histogramming.create_axis(data, bins, range=None)`

Create an axis object for histogram binning based on the input data and parameters.

- Parameters:
- **data** (*array-like*) – The input data for determining the axis range.
 - **bins** (*int or array-like*) – The number of bins or bin edges for the axis.
 - **range** (*None or tuple, optional*) – The range of the axis. If *None*, it will be determined based on the data.

Returns: An axis object for histogram binning.

Return type: Axis object

Raises: **ValueError** – If the range parameter is invalid or not finite.

`plothist.histogramming.flatten_2d_hist(hist)`

Flatten a 2D histogram into a 1D histogram.

Parameters: **hist** (*Histogram object*) – The 2D histogram to be flattened.

Returns: The flattened 1D histogram.

Return type: Histogram object

Raises: **ValueError** – If the input histogram is not 2D.

`plothist.histogramming.make_2d_hist(data, bins=(10, 10), range=(None, None), weights=1)`

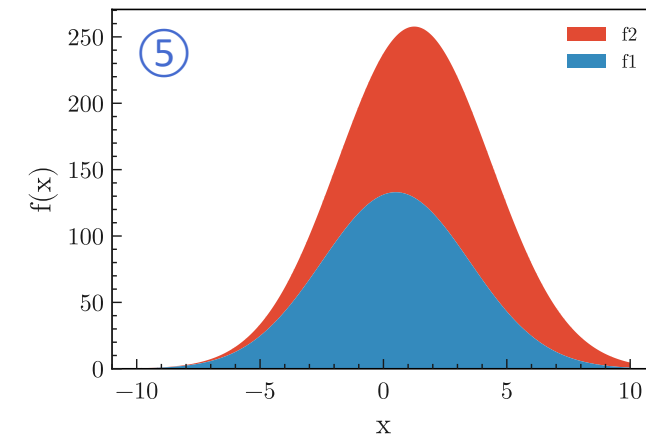
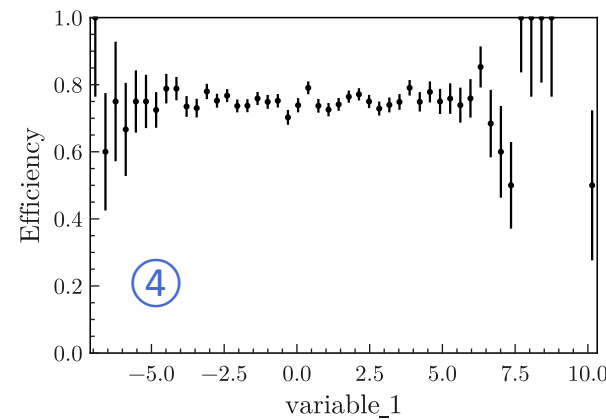
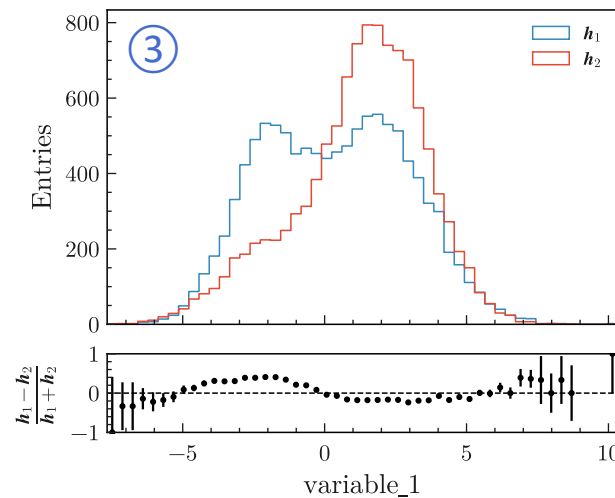
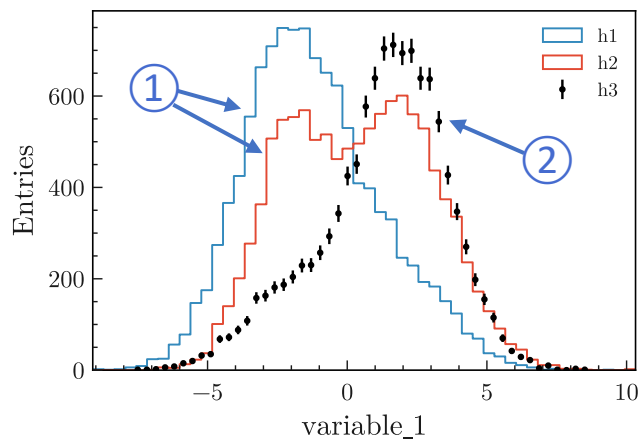
Create a 2D histogram object and fill it with the provided data.

- Parameters:
- **data** (*array-like*) – 2D array-like data used to fill the histogram.
 - **bins** (*tuple, optional*) – Binning specification for each dimension of the histogram (default is (10, 10)). Each element of the tuple represents the number of bins for the corresponding dimension. Also support explicit bin edges specification (for non-constant bin size).

1D examples: overview

Create simple 1D histogram plots or compare them

<code>make_hist</code>	to create <code>boost_histogram</code> objects that are used in <code>plothist</code>
① <code>plot_hist</code>	to plot 1D histogram(s), takes <code>matplotlib</code> arguments for the style
② <code>plot_error_hist</code>	to plot 1D histogram with error bars (can be asymmetrical)
③ <code>plot_two_hist_comparison</code>	to compare 2 histograms
④ <code>plot_comparison</code>	to compare 2 histograms and only plot the comparison
⑤ <code>plot_function</code>	to plot 1D function(s), takes <code>matplotlib</code> arguments for the style



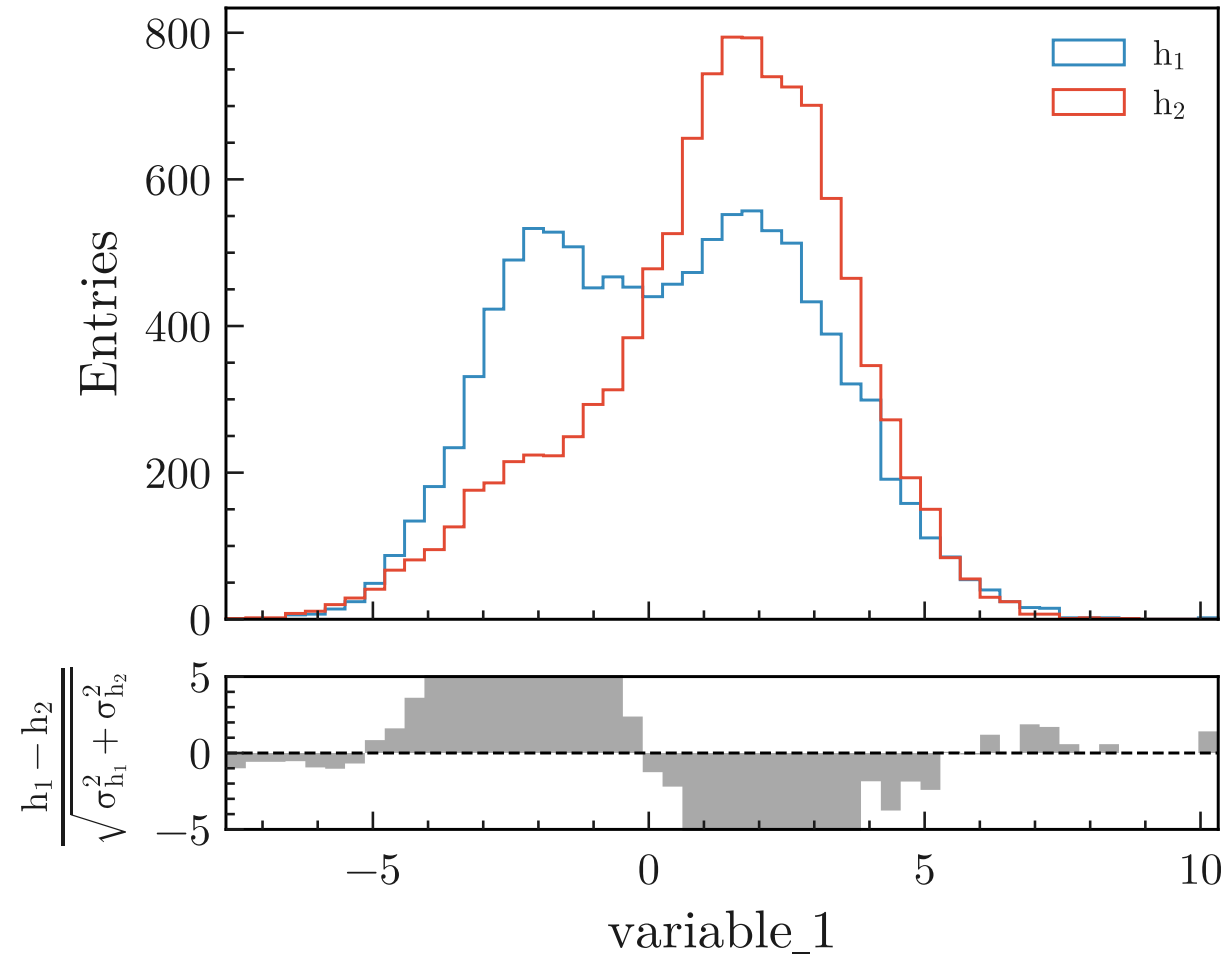
1D examples: histogram comparison

Example

```
from plothist import make_hist, plot_two_hist_comparison

h1 = make_hist(df["x1"], bins=50, range=[-7.5, 10.1])
h2 = make_hist(df["x2"], bins=50, range=[-7.5, 10.1])

fig, ax_main, ax_comparison = plot_two_hist_comparison(
    h1,
    h2,
    xlabel      ="variable_1",
    ylabel      ="Entries",
    h1_label    ="$h_1$",
    h2_label    ="$h_2$",
    comparison  ="pull",
)
```



Available comparisons

Ratio: $\frac{h_1}{h_2}$

Difference: $h_1 - h_2$

Asymmetry: $\frac{h_1 - h_2}{h_1 + h_2}$

Pull: $\frac{h_1 - h_2}{\sqrt{\sigma_{h_1}^2 + \sigma_{h_2}^2}}$

Relative difference: $\frac{h_1 - h_2}{h_2}$

Efficiency: $\frac{h_1}{h_2}$ with h_1 a subset of h_2

2D examples: overview

Create simple 2D histogram plots

`make_2d_hist`

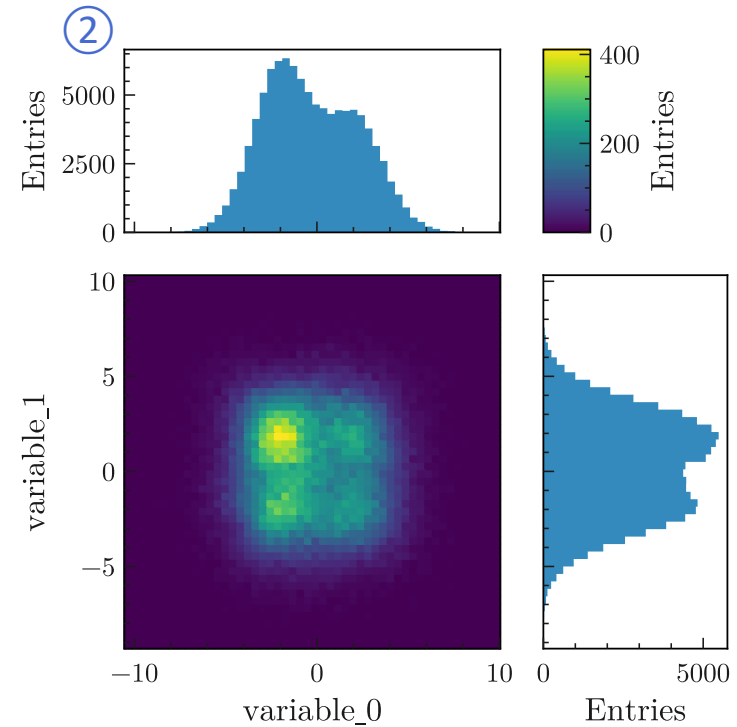
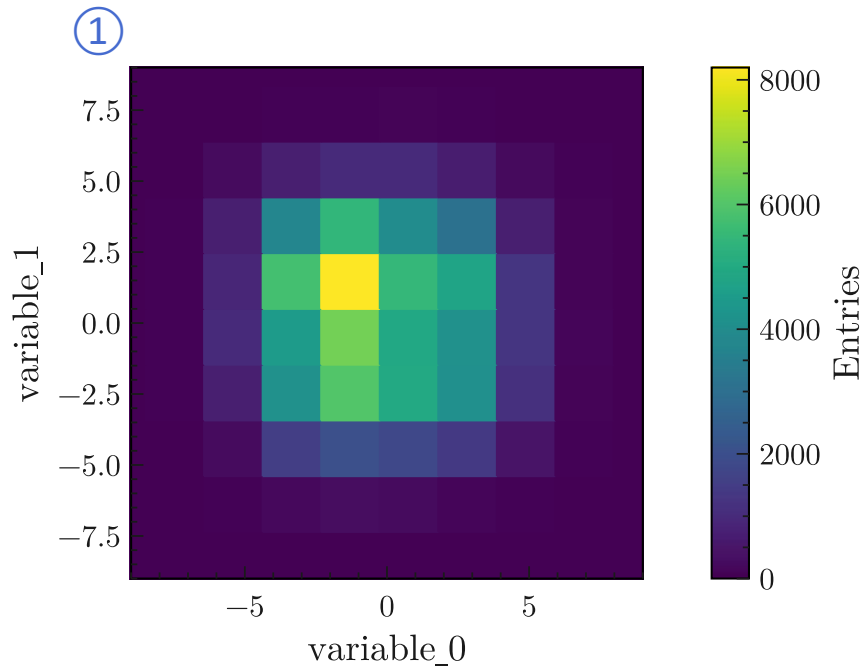
to create `boost_histogram` objects that are used in `plothist`

① `plot_2d_hist`

to plot 2D histogram, takes `matplotlib` arguments for the style

② `plot_2d_hist_with_projections`

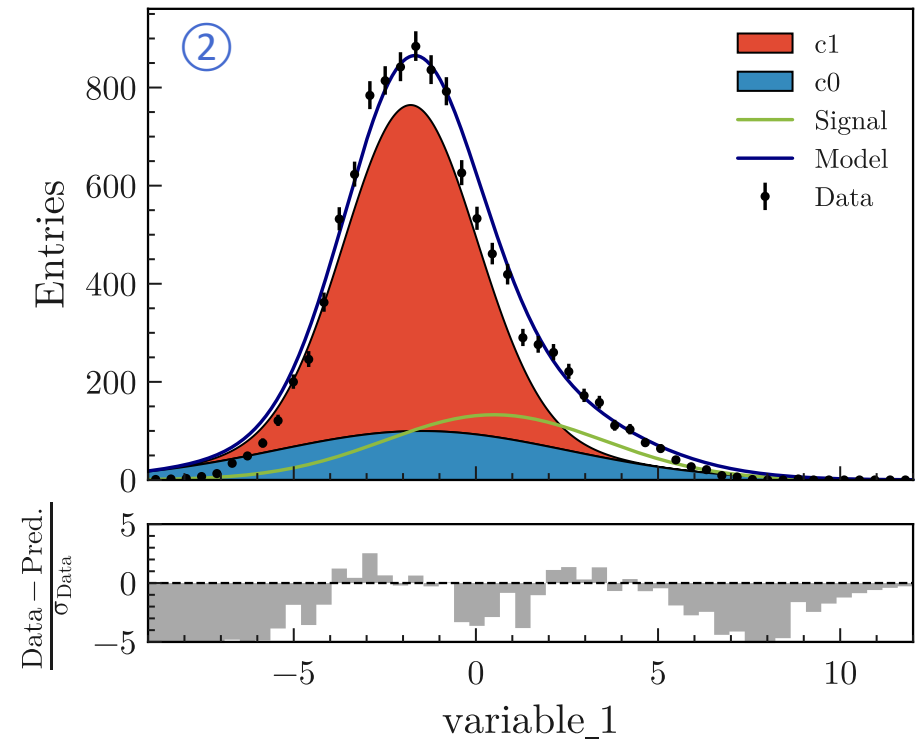
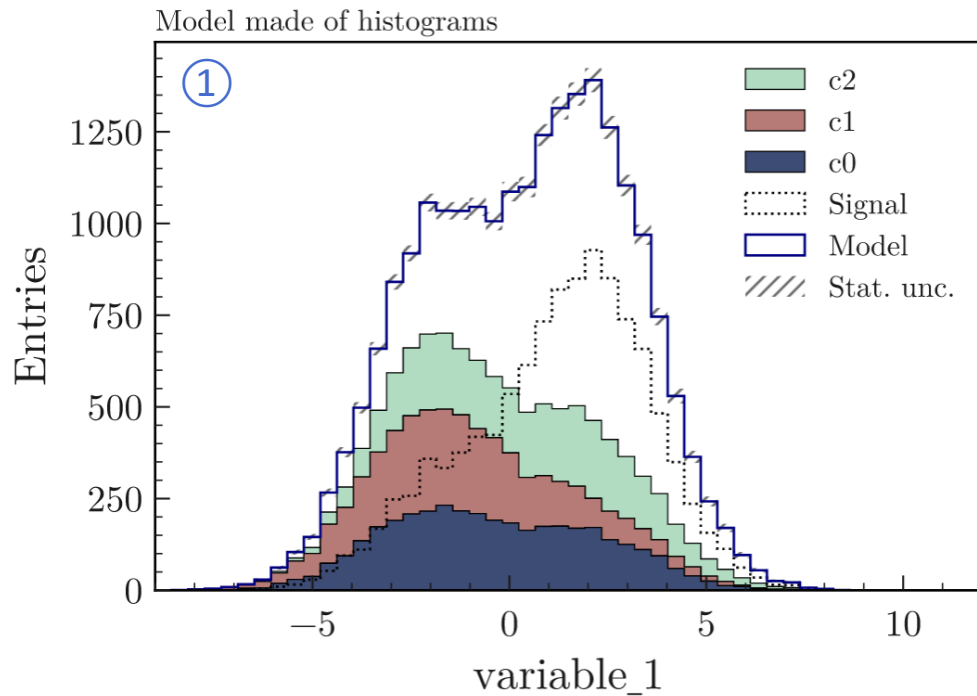
to plot 2D hist with the 1D projections



High-energy physics examples: overview

Create Data/model plots in a few lines of code

- | | |
|---|---|
| ① <code>plot_model</code> | to plot stacked and/or unstacked histograms or functions together |
| ② <code>plot_data_model_comparison</code> | to compare stacked and/or unstacked histograms or functions with data |



High-energy physics examples: Data vs Model comparison

Example

```
from plothist import plot_data_model_comparison, add_luminosity

fig, ax_main, ax_comparison = plot_data_model_comparison(
    data_hist           =data_hist,
    stacked_components  =background_hists,
    stacked_labels      =background_categories_labels,
    stacked_colors      =background_categories_colors,
    xlabel              ="variable [TeV/$(c^2$)]",
    ylabel              ="Candidates per 0.42 [TeV/$(c^2$)]",
    comparison         ="pull"
)

add_luminosity(
    collaboration="plothist",
    ax=ax_main,
    lumi=3,
    lumi_unit="zb",
    preliminary=True
)
```

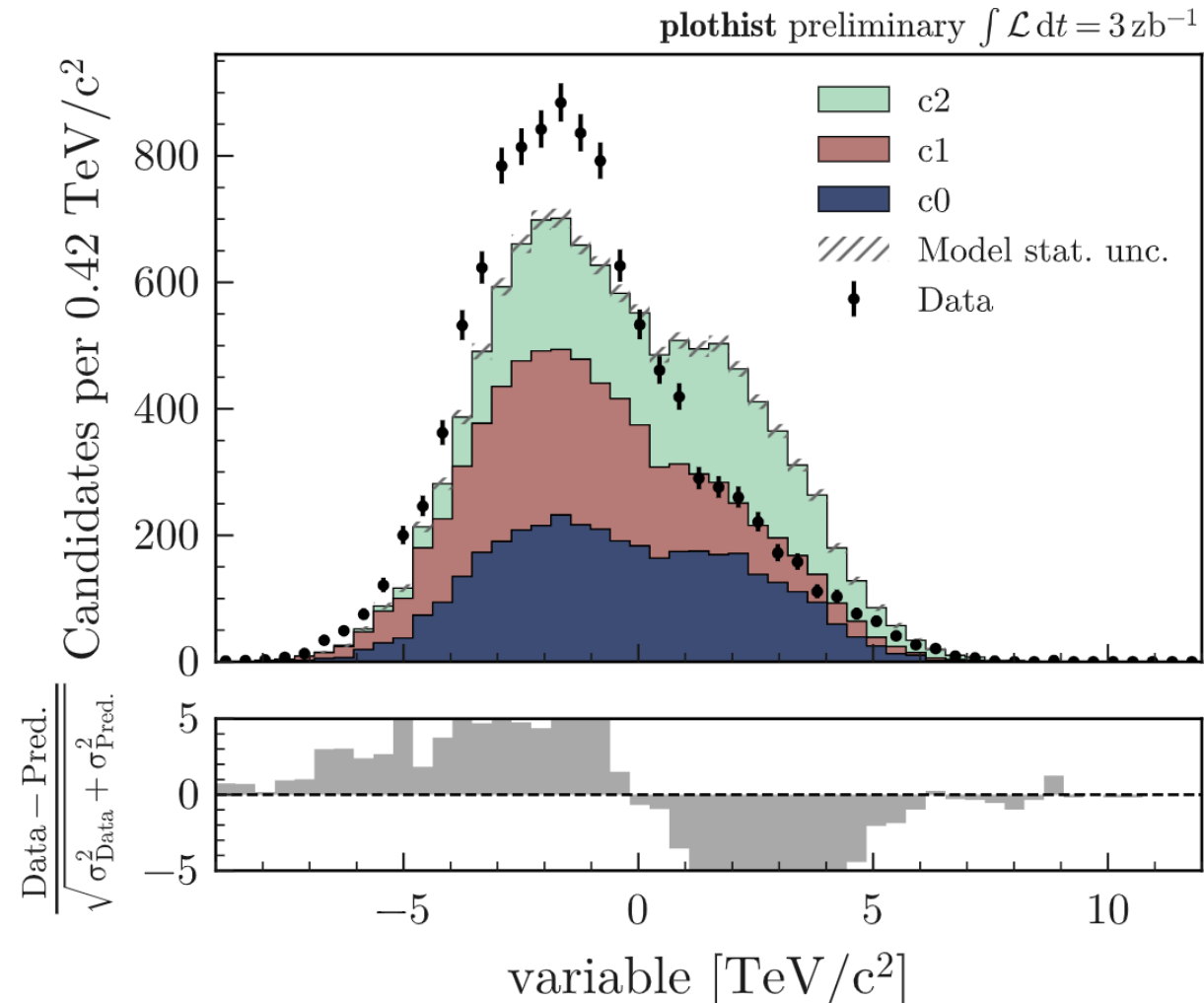
Available comparisons

$$\text{Ratio: } \frac{h_1}{h_2}$$

$$\text{Pull: } \frac{h_1 - h_2}{\sqrt{\sigma_{h_1}^2 + \sigma_{h_2}^2}} \quad \text{or} \quad \frac{h_1 - h_2}{\sigma_{h_1}}$$

$$\text{Difference: } h_1 - h_2$$

$$\text{Relative difference: } \frac{h_1 - h_2}{h_2}$$



Variable registry

Functionalities

- Manage any number of variable using unique identifiers (*keys*)
- Store any information in a *database* (YAML file)

```
variable_keys = ["variable_0", "variable_1", "variable_2"]  
create_variable_registry(variable_keys)
```

variable_registry.yaml

```
variable_0:  
  name: variable_0  
  bins: 50  
  range:  
  - min  
  - max  
  label: variable_0  
  log: false  
  legend_location: best  
  legend_ncols: 1  
  docstring: ''  
  
  ...  
  
variable_1:  
  ...
```

Variable registry

Functionalities

- Manage any number of variable using unique identifiers (*keys*)
- Store any information in a *database* (YAML file)
- Retrieve information with only the keys

```
variable_keys = ["variable_0", "variable_1", "variable_2"]
create_variable_registry(variable_keys)

variable = get_variable_from_registry("variable_0")

# variable is a dictionary
# Get the name: variable["name"]
# Get the range: variable["range"]
# ...
```

variable_registry.yaml

```
variable_0:
  name: variable_0
  bins: 50
  range:
    - min
    - max
  label: variable_0
  log: false
  legend_location: best
  legend_ncols: 1
  docstring: ''

...

variable_1:
  ...
```

Variable registry

Functionalities

- Manage any number of variable using unique identifiers (*keys*)
- Store any information in a *database* (YAML file)
- Retrieve information with only the keys
- Update or add automatically *information* (like the range)

```
variable_keys = ["variable_0", "variable_1", "variable_2"]
create_variable_registry(variable_keys)
variable = get_variable_from_registry("variable_0")
update_variable_registry_ranges(df, variable_keys)
```

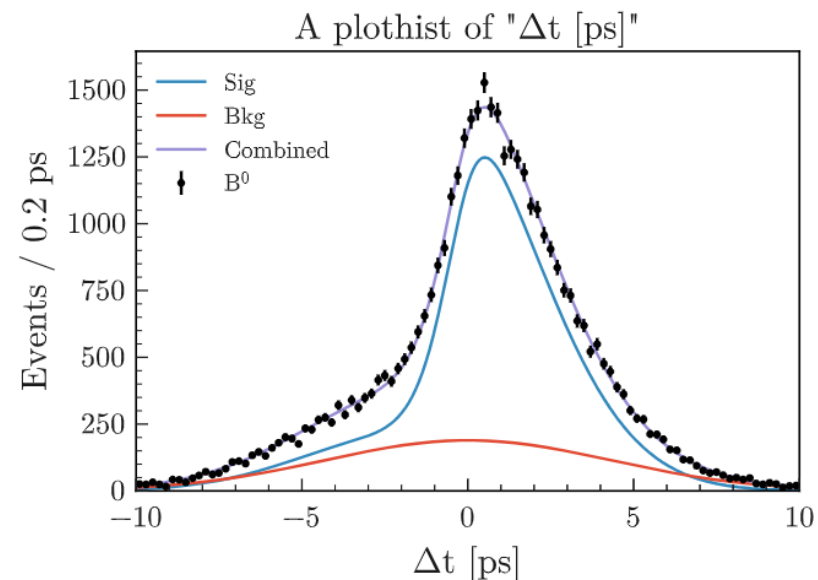
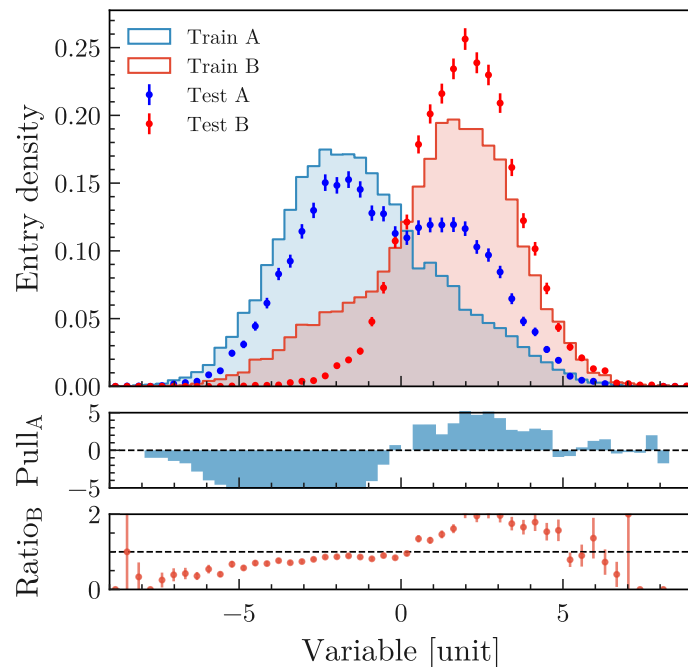
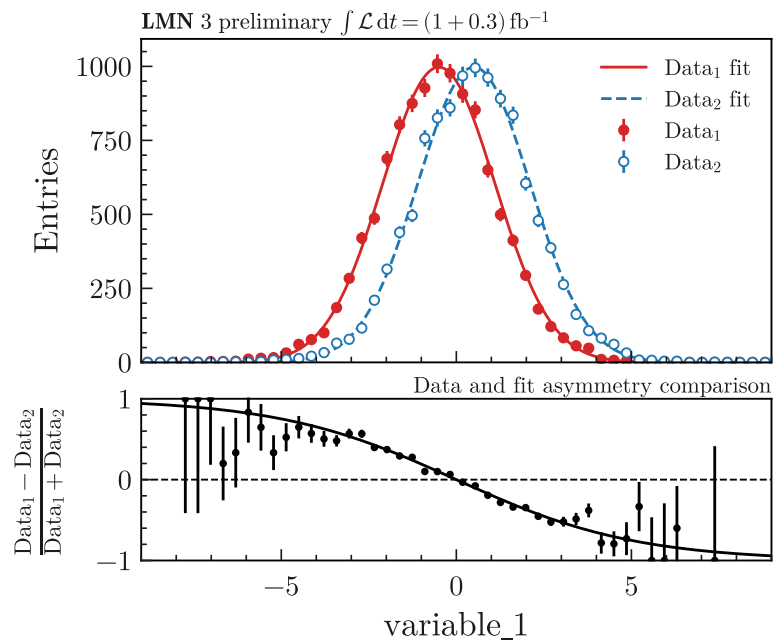
variable_registry.yaml

```
variable_0:
  name: variable_0
  bins: 50
  range:
    - -10.55227774892869 # min(df["variable_0"])
    - 10.04658448558009 # max(df["variable_0"])
  label: variable_0
  log: false
  legend_location: best
  legend_ncols: 1
  docstring: ''
  ...
variable_1:
  ...
```


And more!

Complex [examples](#), code still simple and easy-to-navigate

[Tutorial](#) to transfer [RooFit](#), [zfit](#) or [pyhf](#) plot to [plothist](#)



Utility functions

<code>install_latin_modern_fonts</code>	from the terminal to install the LaTeX fonts
<code>add_luminosity</code>	to easily add luminosity + collaboration text on the plot
<code>get_color_palette</code>	to sample any color palette
<code>add_text</code>	to easily add text on a plot
<code>set_fitting_ylabel_fontsize</code>	to automatically set the ylabel font size to fit the plot
<code>And more!</code>	

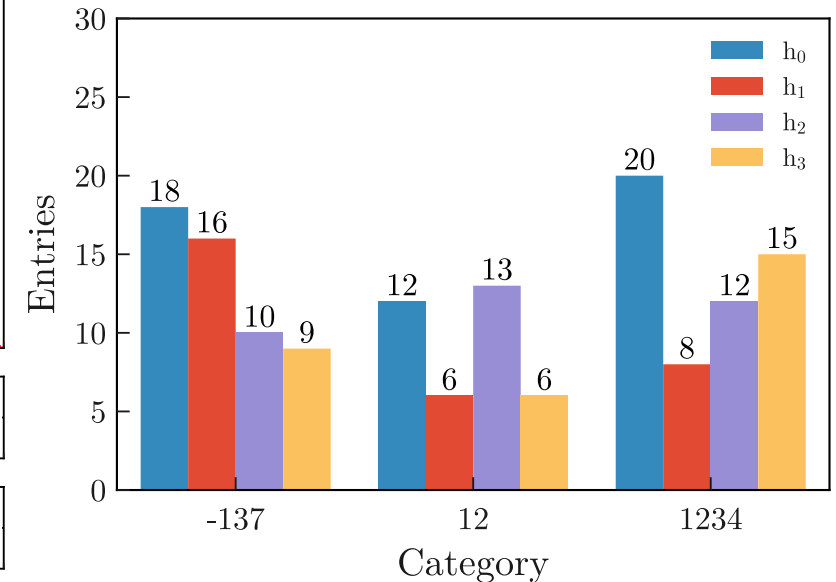
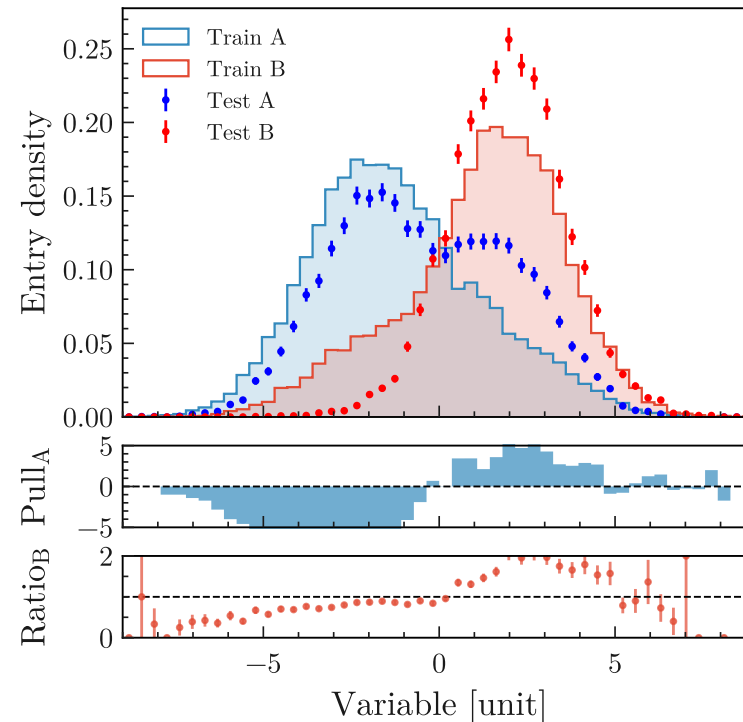
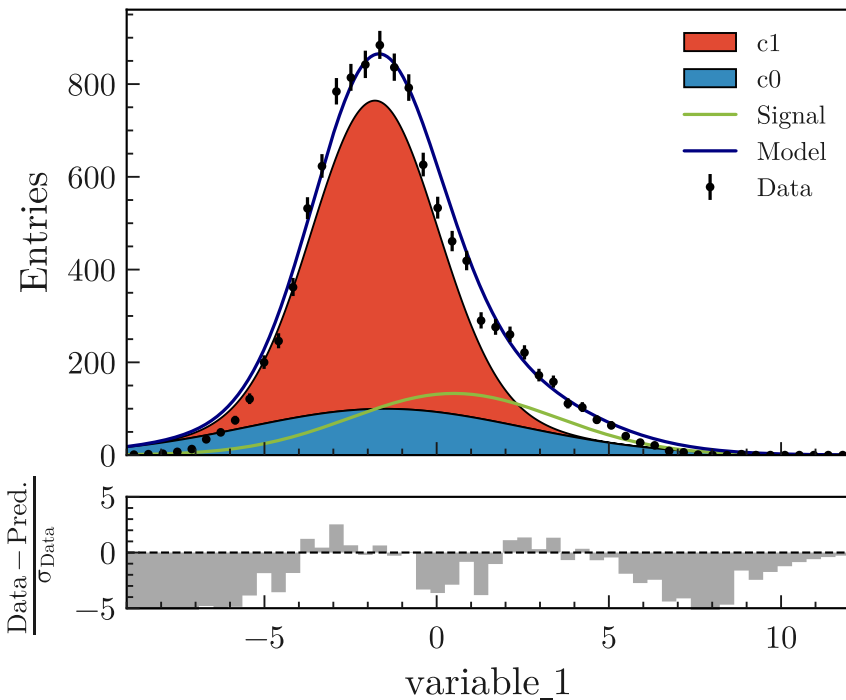
Outlook

- Every example shown here (and many others) available in the doc:

<https://plothist.readthedocs.io>

https://plothist.readthedocs.io/en/latest/example_gallery

- plothist already used by collaborators of multiple experiments
Main feedback: plothist is a time saver, so they can spend more time on physics than on making and tuning plots
- We are starting to discuss coordinating plothist with scikit-HEP packages (see backup)



Thank you for your attention!

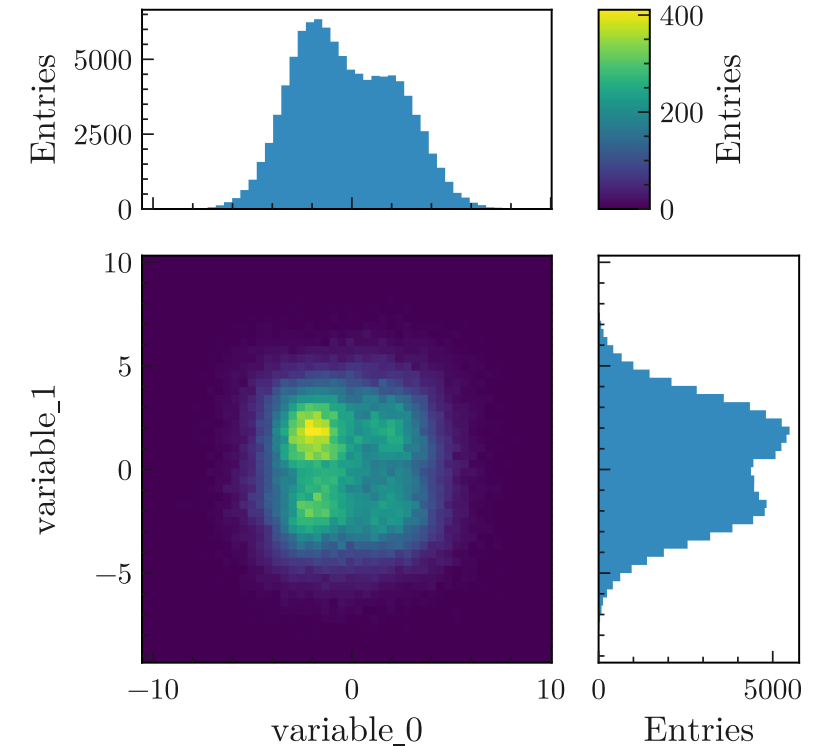
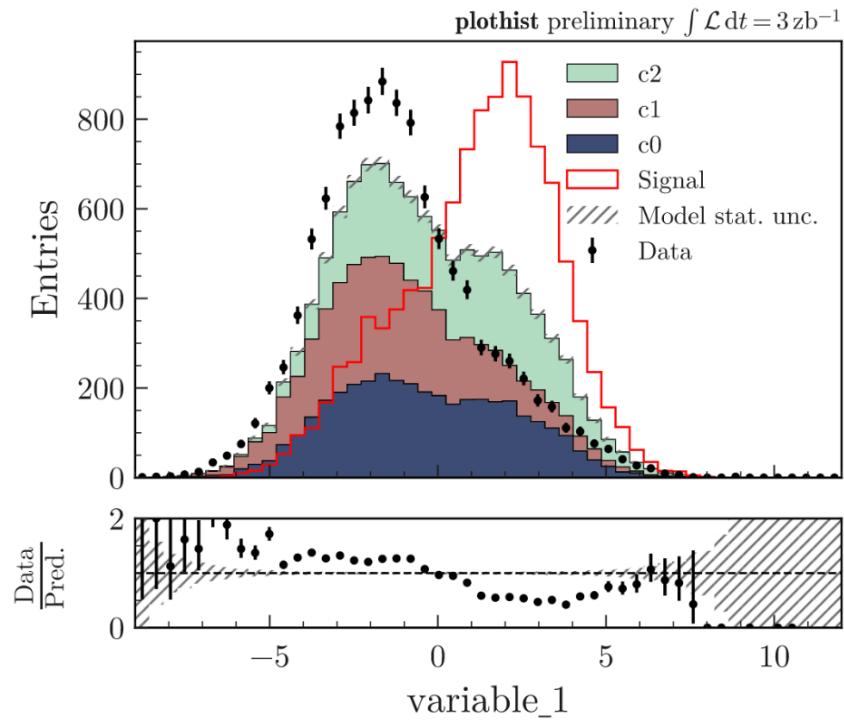
Next: Jupyter notebook interactive session





Tristan Fillinger, Cyrille Praz



01/07/24

PyHEP 2024



Comparison/compatibility with other scikit-HEP packages

	Hist 	plothist 
Plotting	Plotting methods (including 1-hist-to-1-hist comparisons)	Allows to compare data with models consisting of any number of stacked and unstacked components that are either histograms or functions
Variable information (name, units, ...)	Stored in the metadata of the class	Stored in a YAML file to offer more flexibility

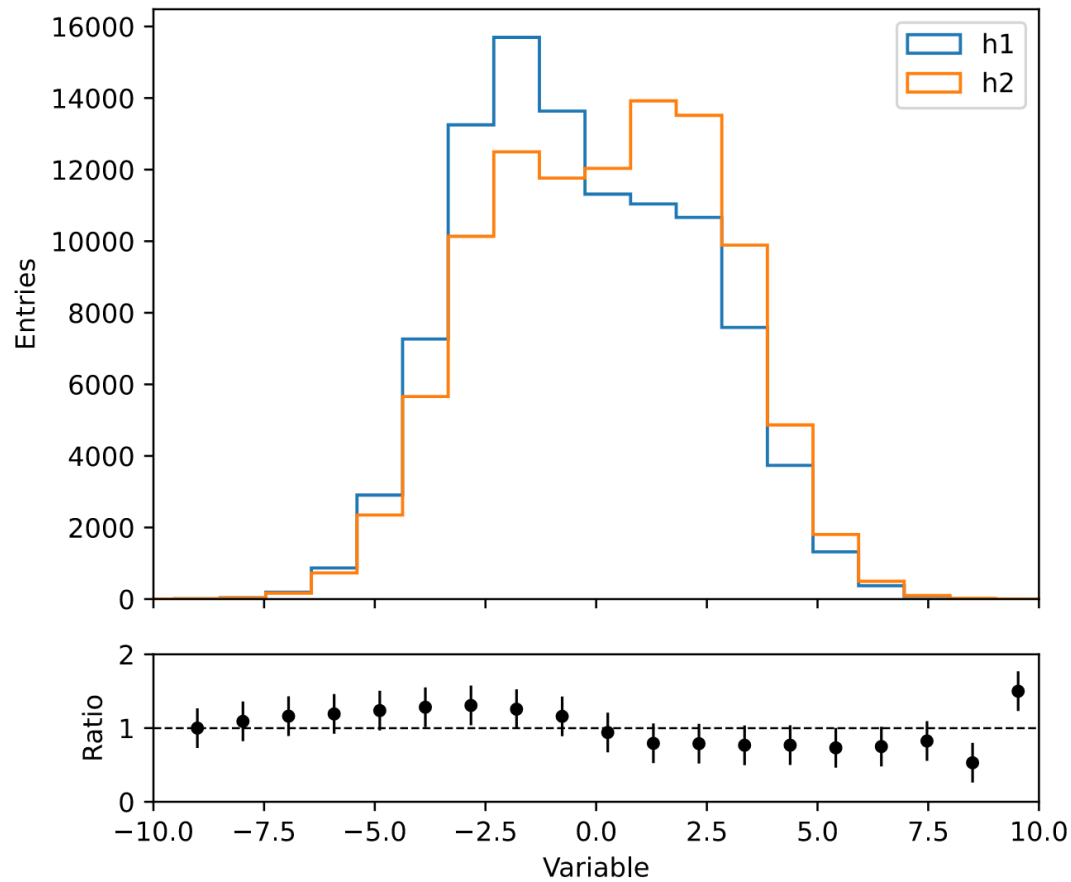
	mplhep 	plothist 
Histogramming	Plot numpy, boost-histogram (incl. Hist), PlottableProtocol histograms	Plot boost-histogram (incl. Hist) histograms
Plotting	Only simple plots, no histogram comparison or data-model comparison	Provide high level functions to create out-of-the-box data-model comparisons
Style	Supports multiple collaboration styles	One default style, compatible with Physical Review Letters / Physical Review D (with tools to add more)

We are starting to discuss coordinating the plothist and mplhep packages

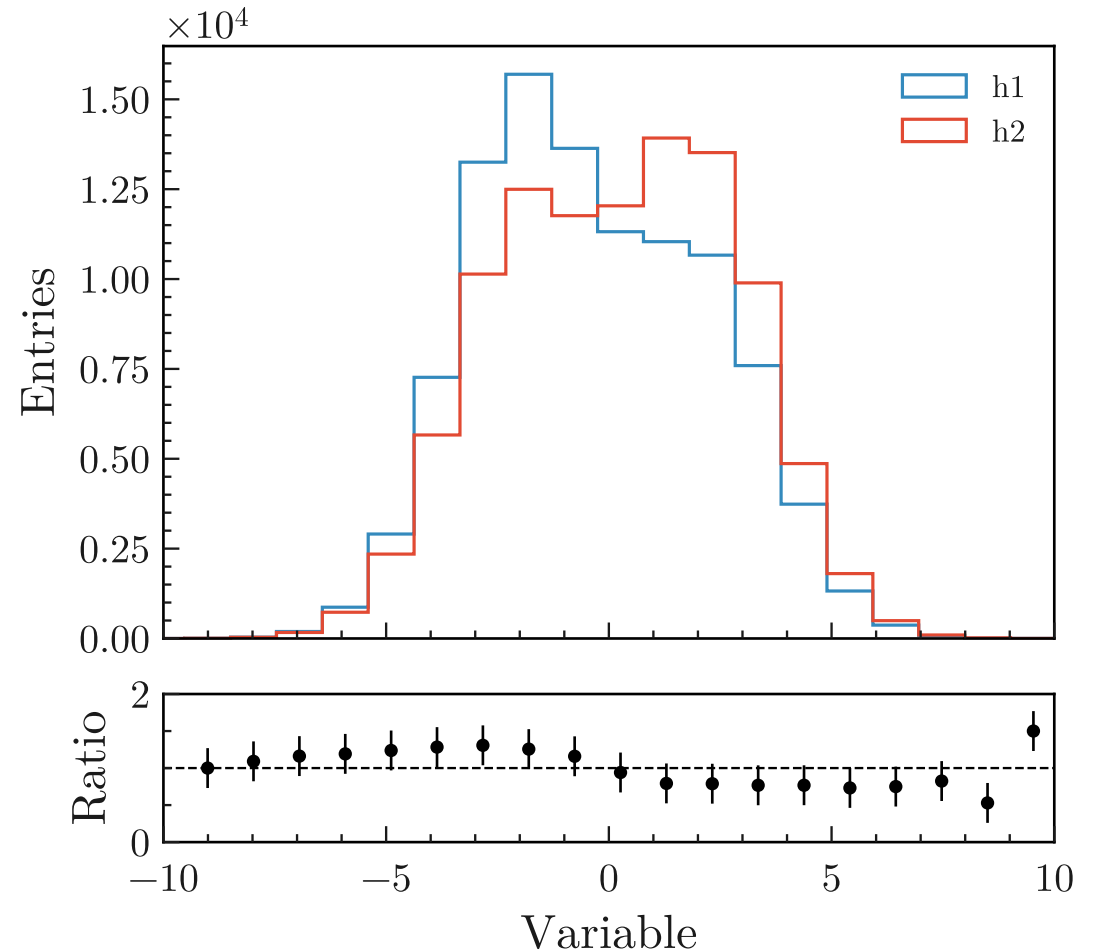
Without using plothist functions

If you just want the style, add `import plothist` to your python script

Simple matplotlib script



Same script, just `import plothist` added



Style

Fonts

Latin Modern (LaTeX)

Colors

Provide a function to sample colors from:

- Default palette
- Cubehelix palette
- Any matplotlib palettes

