Development and Benchmarking of a Bayesian Inference Pipeline for LHC Physics

Scientific Thesis for the procurance of the degree

Bachelor of Science

from the Physics Department at the Technical University of Munich.

Supervised by *Prof. Dr. Lukas Heinrich*
ORIGINS Data Science Lab

Dr. Oliver Schulz
Max Planck Institute for Physics

Submitted by *Christian Gajek*

Submitted on September 22, 2022

Abstract

TODO

Contents

Abbreviations

BAT Bayesian Analysis Toolkit.

HEP High Energy Physics.
HF HistFactory.

pdfs probability density functions.

Chapter 1

Mathematical Preliminaries

1.1 Frequentist versus Bayesian Inference

1.2 MCMC Sampling

1.2.1 Markov Chains

1.2.2 Metropolis Hastings

1.2.3 Hamiltonian Markov Chains

Chapter 2

HistFactory

HistFactory (HF) is a tool for binned statistical analysis which is widely used in High Energy Physics (HEP) to measure the consistency of collision events with theoretical predictions. It has been employed for the discovery of the Higgs Boson [1] and is used in searches for new physics [2] by research groups around the planet. The relationship between theoretical predictions and collision events is formalized as *statistical model* $f(\boldsymbol{x} \mid \boldsymbol{\phi})$. It describes the probability of observing data $\boldsymbol{x}$ given the model parameters $\boldsymbol{\phi}$. Typically, models in HEP are complex with hundreds of parameters. HF enables a standardized way to build parametrized probability density functions (pdfs) and infer parameter properties from it.

2.1 Formalism

In HEP the phase space of observed events

In a frequentist framework the constraint terms can be viewed as *auxiliary measurements* paired with the channel data $\boldsymbol{n}$. observation $\boldsymbol{x} = (\boldsymbol{n}, \boldsymbol{a})$

$$f(\boldsymbol{x} \mid \boldsymbol{\phi}) = f(\boldsymbol{x} \mid \boldsymbol{\eta}, \boldsymbol{\chi}) = f(\boldsymbol{x} \mid \boldsymbol{\psi}, \boldsymbol{\theta}) \tag{2.1}$$

The Poisson distribution

$$\mathrm{Pois}(n \mid \nu) = \frac{\nu^n}{n!} e^{-\nu} \tag{2.2}$$

$$f(\boldsymbol{n}, \boldsymbol{a} \mid \boldsymbol{\eta}, \boldsymbol{\chi}) = \underbrace{\prod_{c \,\in\, \text{channels}} \prod_{b \,\in\, \text{bins}_c} \mathrm{Pois}\left(n_{cb} \mid \nu_{cb}(\boldsymbol{\eta}, \boldsymbol{\chi})\right)}_{\substack{\text{simultaneous measurement} \\ \text{of multiple channels}}} \underbrace{\prod_{\chi \,\in\, \boldsymbol{\chi}} c_\chi(a_\chi \mid \chi)}_{\substack{\text{constraint terms} \\ \text{for auxiliary measurements}}} \tag{2.3}$$

The event rates ν_{cb} are defined as

$$\begin{aligned}
\nu_{cb}(\boldsymbol{\eta},\boldsymbol{\chi}) &= \sum_{s\,\in\,\text{samples}} \nu_{scb}(\boldsymbol{\eta},\boldsymbol{\chi}) \\
&= \sum_{s\,\in\,\text{samples}} \underbrace{\left(\prod_{\kappa\in\boldsymbol{\kappa}} \kappa_{scb}(\boldsymbol{\eta},\boldsymbol{\chi})\right)}_{\text{multiplicative modifiers}} \left(\nu^0_{scb}(\boldsymbol{\eta},\boldsymbol{\chi}) + \underbrace{\sum_{\Delta\in\boldsymbol{\Delta}} \Delta_{scb}(\boldsymbol{\eta},\boldsymbol{\chi})}_{\text{additive modifiers}}\right) \qquad (2.4)
\end{aligned}$$

2.2 Modifiers

Table 2.1: HistFactory modifiers and constraints [3]

Description	Modification	Constraint Term c_χ	Input
Uncorrelated Shape `shapesys`	$\kappa_{scb}(\gamma_b) = \gamma_b$	$\prod_b \text{Pois}\left(r_b = \sigma_b^{-2} \,\middle|\, \rho_b = \sigma_b^{-2}\gamma_b\right)$	σ_b
Correlated Shape `histosys`	$\Delta_{scb}(\alpha) = f_p\left(\alpha \,\middle|\, \Delta_{scb,\alpha=\pm1}\right)$	$\text{Normal}\left(a = 0 \,\middle|\, \alpha, \sigma = 1\right)$	$\Delta_{scb,\alpha=\pm1}$
Normalisation Uncert. `normsys`	$\kappa_{scb}(\alpha) = g_p\left(\alpha \,\middle|\, \kappa_{scb,\alpha\pm1}\right)$	$\text{Normal}\left(a = 0 \,\middle|\, \alpha, \sigma = 1\right)$	$\kappa_{scb,\alpha\pm1}$
MC Stat. Uncertainty `staterror`	$\kappa_{scb}(\gamma_b) = \gamma_b$	$\prod_b \text{Normal}\left(a_{\gamma_b} = 1 \,\middle|\, \gamma_b, \delta_b\right)$	$\delta_b^2 = \sum_s \delta_{sb}^2$
Luminosity `lumi`	$\kappa_{scb}(\lambda) = \lambda$	$\text{Normal}\left(l = \lambda_0 \,\middle|\, \lambda, \sigma_\lambda\right)$	$\lambda_0,\ \sigma_\lambda$
Normalisation `normfactor`	$\kappa_{scb}(\mu_b) = \mu_b$		
Data-driven Shape `shapefactor`	$\kappa_{scb}(\gamma_b) = \gamma_b$		

2.3 Workspaces

While flexible enough to describe a wide range of LHC measurements, statistical models in HistFactory are described in plain-text `JSON` format. This scheme fully specifies the model structure as well as necessary constrained data in a single document, and hence is implementation independent. The `JSON` file describes a *workspace* which consists of *channels*, *measurements* and *observations*.

-
-
- **Observations** are the actual observed events in a particle detector for

2.4 HistFactory Implementations

Currently, HistFactory is available in three different programming languages

- a C++ implementation, (cite?)
- a Python version `pyhf` [4], and
- a Julia implementation `LiteHF` [5]

In chapter 4, this thesis uses `pyhf` and `LiteHF` for run time benchmarks. To verify the coincidence of `pyhf` and `LiteHF` the log posterior measure is evaluated for both implementations. The log likelihood in Python for a parameter vector θ is computed by the `logpdf` of the `main_model`, i.e.

```python
def llh(param: np.ndarray) -> float
    """pyhf log likelihood from the main_model."""
    return model.main_model.logpdf(main_data, param)
```

In `LiteHF` one can access the log likelihood by

```julia
llh = pyhf_loglikelihoodof(pyhfmodel.expected, pyhfmodel.observed)
```

For visualization purposes, the verification step is illustrated for the 2D `2_bin_corr` model in section A.2. The prior vector is chosen to be

$$p(\boldsymbol{\theta}) \sim \begin{bmatrix} \text{Uniform}(0,\, 5) \\ \text{Normal}(0,\, 1) \end{bmatrix} \tag{2.5}$$

and the log posterior measure is evaluated for 10^5 points $\boldsymbol{x} \sim p(\boldsymbol{\theta})$ with equal initial `seed`. Up to numerical precision the samples coincident for both implementations. In Figure 2.1 1000 randomly chosen data points are illustrated for `pyhf` (blue) and `LiteHF` (orange).

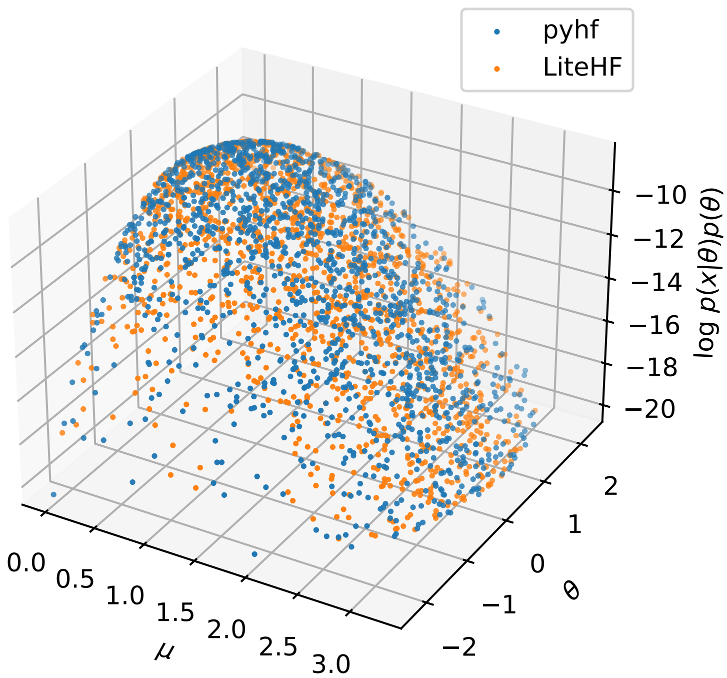


Figure 2.1: Verify the equivalence of `pyhf` and `LiteHF` implementation by evaluating the log posterior measure $\log p\,(\boldsymbol{x}\,|\,\boldsymbol{\theta})\,p(\boldsymbol{\theta})$ at 1000 random points $x \sim p(\boldsymbol{\theta})$ for the `2_bin_corr` model (see section A.2). The 2 dimensional prior vector is set to $p(\boldsymbol{\theta}) \sim [\text{Uniform}(0, 5),\ \text{Normal}(0, 1)]^T$.

Chapter 3

Bayesian Inference with HistFactory

3.1 BAT

Bayesian Analysis Toolkit (BAT)

3.2 batty – BAT to Python Interface

3.3 Priors from `auxdata`

3.3.1 Implementation

`model.config.par_map` is a ordered mapping from the parameter name to the slice in the parameter vector.

3.4 Gradients for HamiltonianMC

3.5 Python HistFactory Benchmarks

While running code on a PC, the execution time of a function varies due to other active processes. A typical run time statistics is illustrated in Figure 3.1.

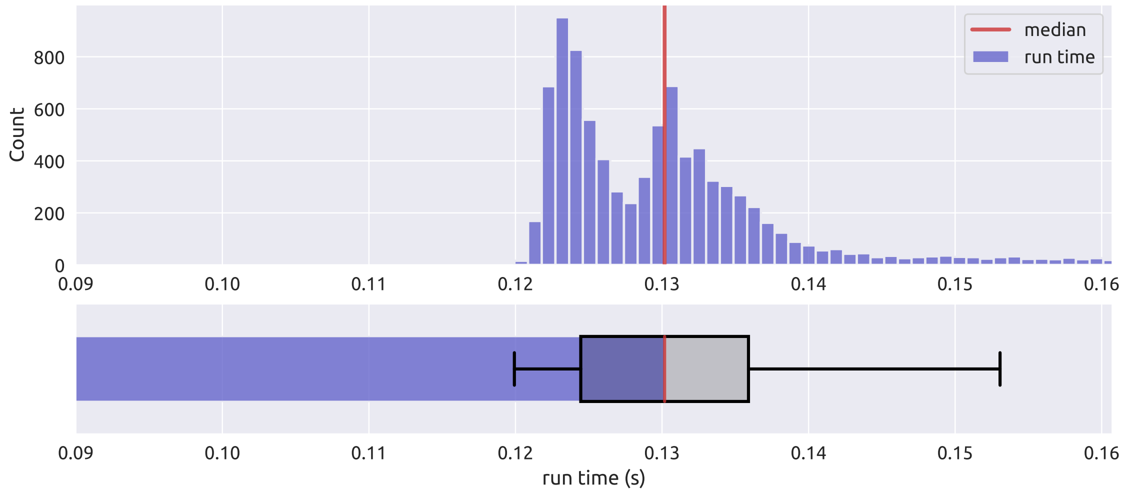


Figure 3.1: Run time statistics of (xx)

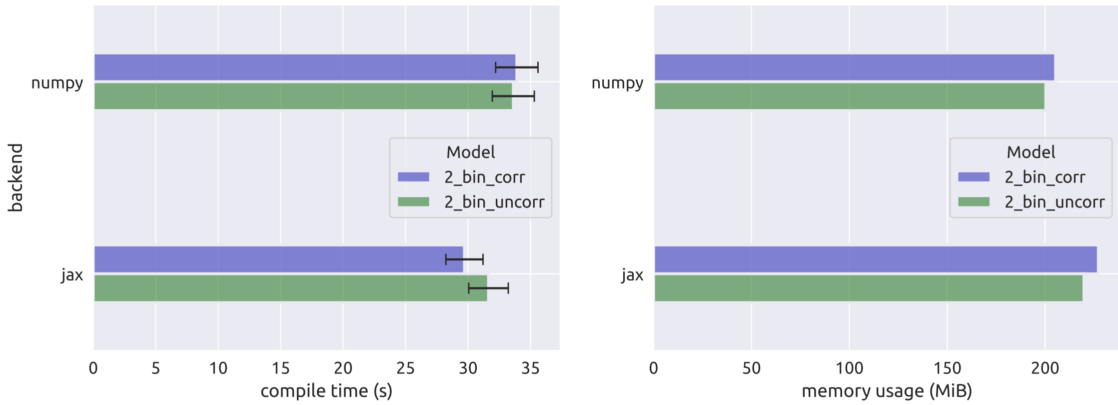
- use median instead mean
- use bar + box plot

3.5.1 `pyhf` Backends

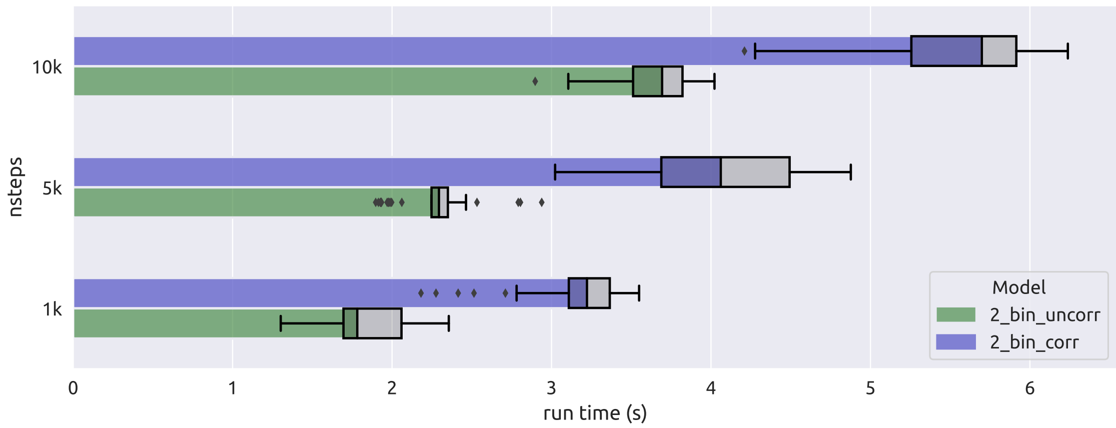
- `jax` vs `numpy`
- 2 simplemodels

- `jax` jitted log likelihood is 4 times faster with `numpy` backend

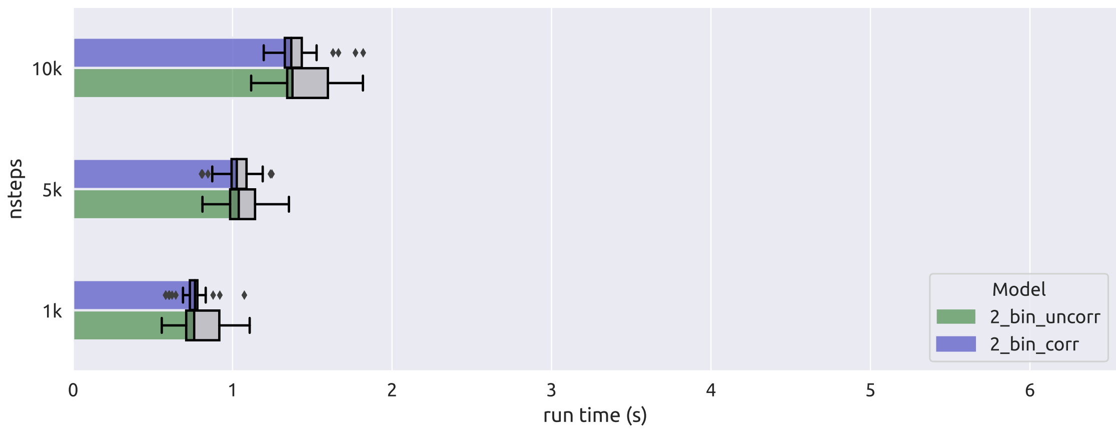
3.6 Bayesian Inference Examples



(a) Compile time and memory usage during compilation



(b) `pyhf` with `numpy` backend



(c) `pyhf` with `jax` backend

Figure 3.2: Compile time and run time performance of `bat_sample()` once using the `numpy` backend of `pyhf` and once with `jax` backend. The run time is evaluated for two different models `2_bin_corr` and `2_bin_uncorr`.

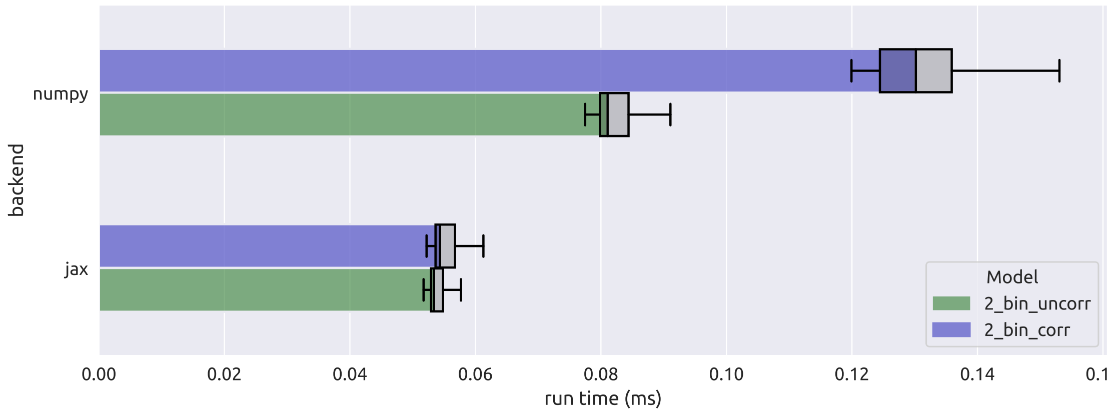


Figure 3.3: Run time of the log likelihood for two different HF models. XX

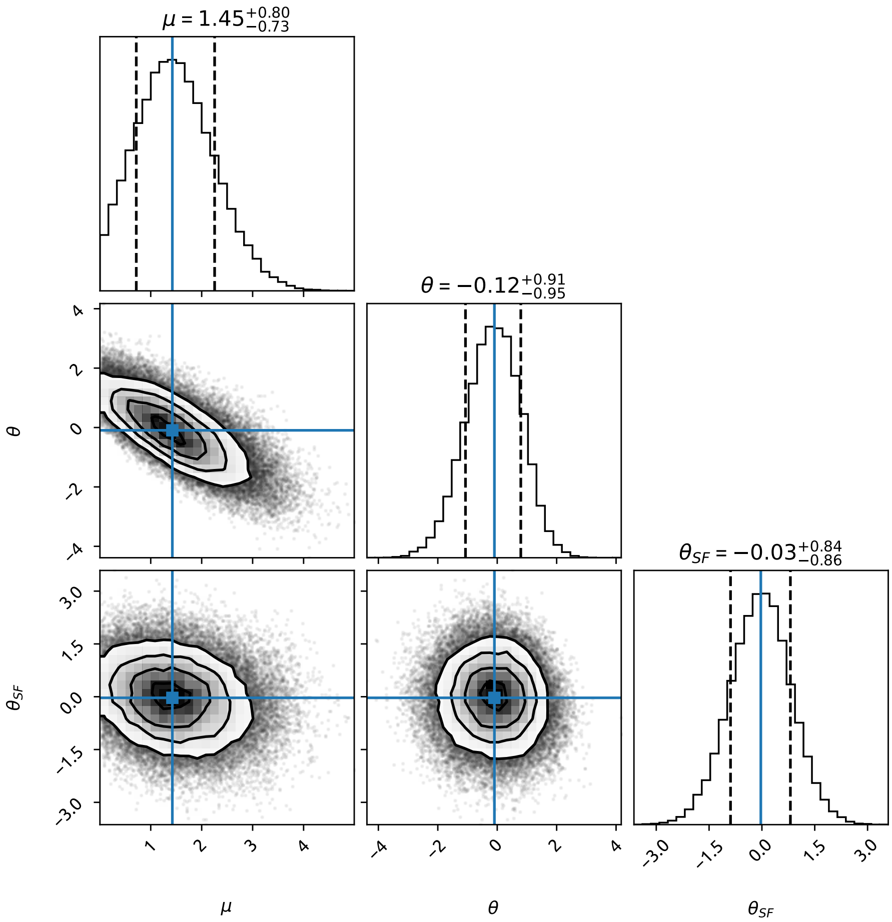
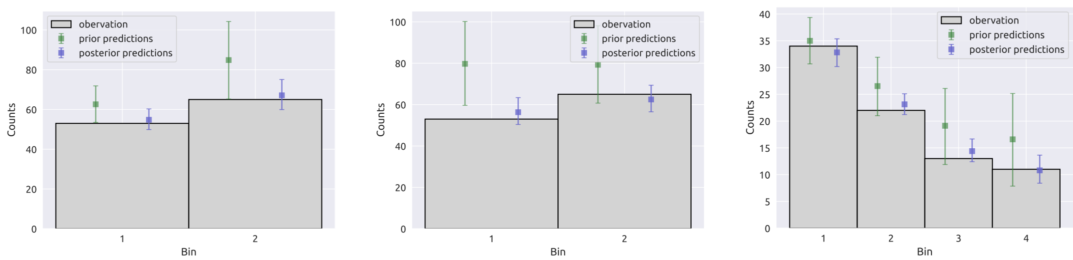


Figure 3.4: Corner plot for the `4_bin` Model (section A.3) for 100k samples. (dashed line 0.16, 0.84 quantile, blue lines mode) XX



(a) `2_bin_uncorr` model (b) `2_bin_corr` model (c) `4_bin` model

Figure 3.5: Posterior predictive check for all used models. Median = Dot, uncertainty as quantile (0.16, 0.84). (wird noch formatiert..)

Chapter 4

Benchmarking the Python-Julia Pipeline against the Julia Implementation

4.1 Overview

4.2 Benchmark Setup

•

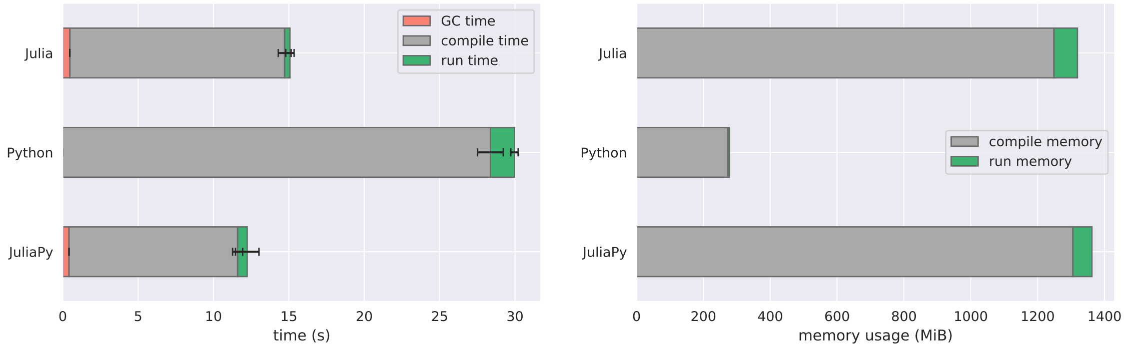


Figure 4.1: Total execution time for `bat_sample()`. (10k steps) ...

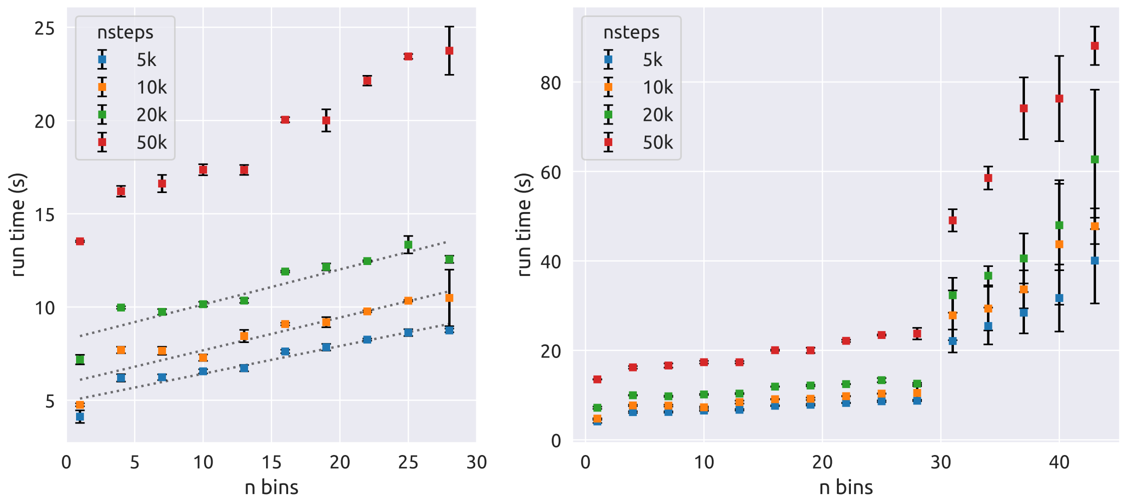
4.3 Benchmarks

4.3.1 `2_bin_corr` Model

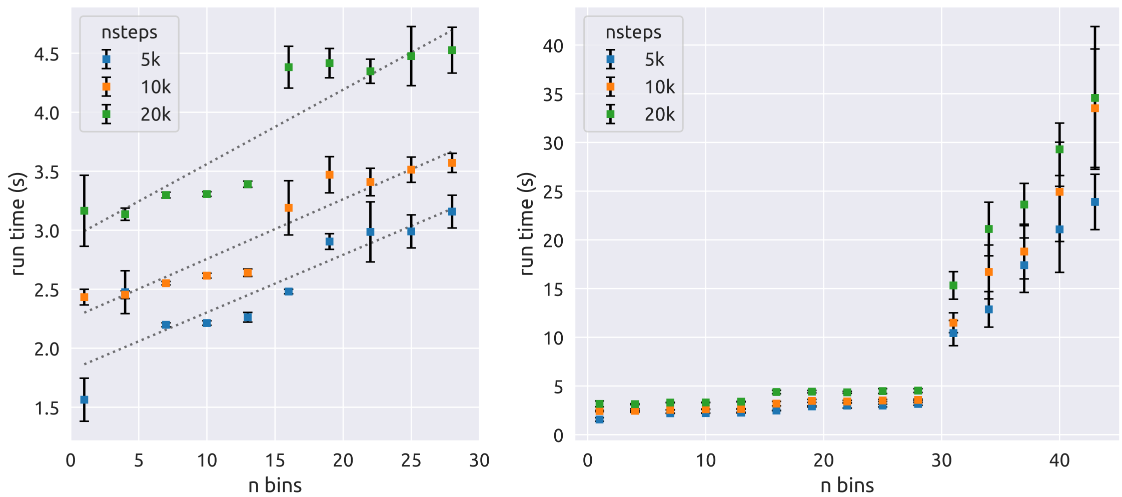
The pure-Julia implementation is about **10 times faster** than the Python-Julia pipeline.

4.3.2 n-Bin-Model

The runtime measurements in this section examine how the dimensionality of the parameter space affects the runtime. The n-bin model is implemented as `simplemodel` with uncorrelated background according to the `2_bin_uncorr` model in section A.1. Repeatedly adding bins to the histogram is a simple method to increase die dimensionality of the inference problem (n bins corresponds to a $(n+1)$-dimensional parameter vector).



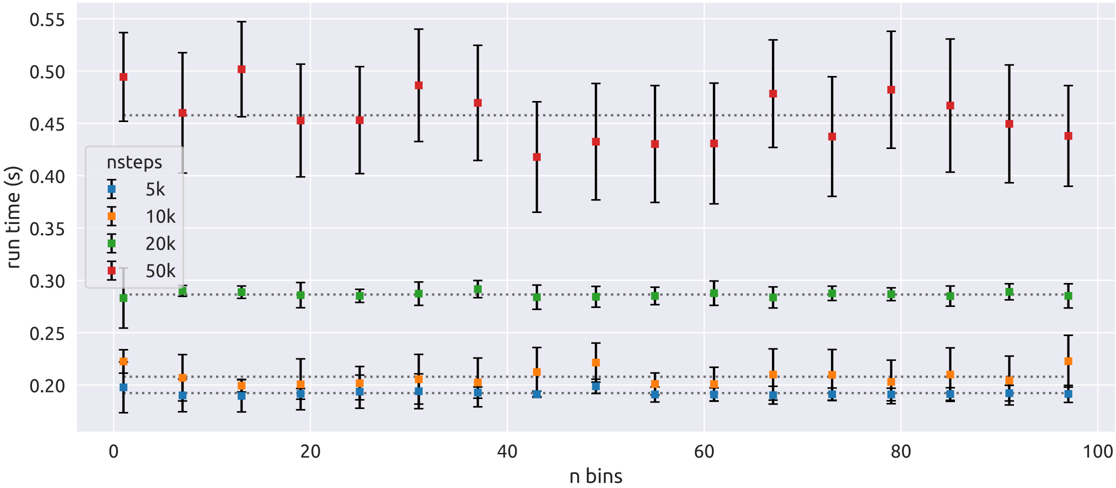
(a) Python + BAT (no `jax`) (linie fehlt)



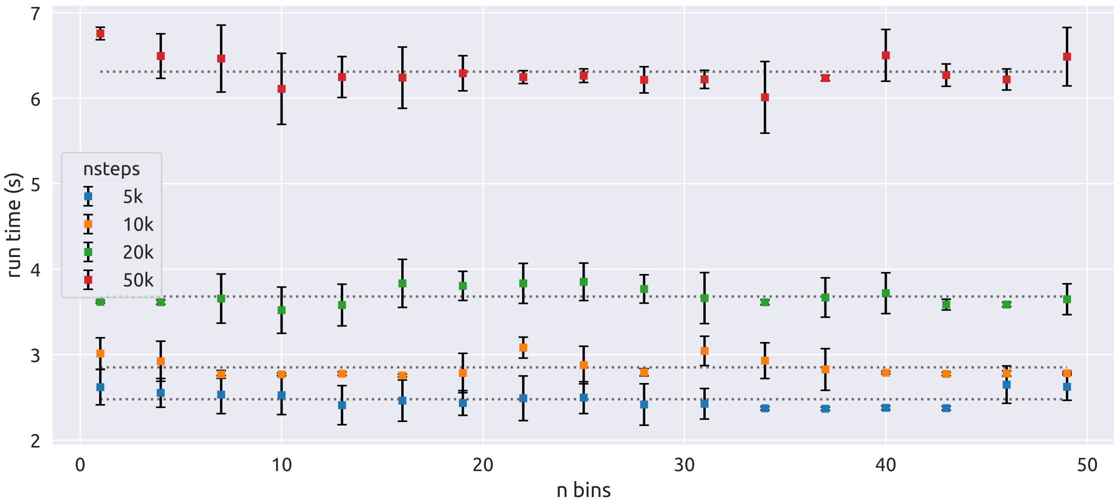
(b) Python + BAT (with `jax`)

Figure 4.2: Run time benchmark of the n-bin Model in Python using Metropolis Hastings. average over 30 runs, uncertainty as std without outliers → better quantiles?

- Python jump at 30 dim parameter vector



(a) Julia + LiteHF



(b) Julia + Python likelihood

Figure 4.3: Run time benchmark of the n-bin Model in Julia.

- Julia runtime constant

4.3.3 `4_bin` Model

TODO

4.3.4 Complex Model

maybe short...

4.4 Discussion

TODO

Appendix A

HistFactory Models

A.1 2_bin_uncorr Model

- parameter $[\mu, \gamma_1, \gamma_2]$
- signal model [5.0, 10.0] with **normfactor** modifier
- bkg model [50.0, 60.0] with **shapesys** modifier, relative uncert. [10 %, 20 %]

```
{
  "channels": [
    { "name": "singlechannel",
      "samples": [
        { "name": "signal",
          "data": [5.0, 10.0],
          "modifiers": [ { "name": "mu", "type": "normfactor", "data": null} ]
        },
        { "name": "background",
          "data": [50.0, 60.0],
          "modifiers": [
            { "name": "uncorr_bkguncrt",
              "type": "shapesys",
              "data": [5.0, 12.0] }
          ]
        }
      ]
    }
  ],
  "observations": [
    { "name": "singlechannel", "data": [50.0, 60.0] }
  ],
  "measurements": [
    { "name": "Measurement", "config": {"poi": "mu", "parameters": []} }
  ],
  "version": "1.0.0"
}
```

A.2 2_bin_corr Model

- parameter [μ, γ]
- ...

```
{
    "channels": [
      { "name": "singlechannel",
        "samples": [
          { "name": "signal",
            "data": [12.0,11.0],
            "modifiers": [ { "name": "mu", "type": "normfactor", "data": null } ]
          },
          { "name": "background",
            "data": [ 50.0, 52.0 ],
            "modifiers": [
              { "name": "correlated_bkg_uncertainty",
                "type": "histosys",
                "data": { "hi_data": [45.0, 57.0], "lo_data": [55.0, 47.0] }
              }
            ]
          }
        ]
      }
    ],
    "observations": [
      { "name": "single_channel", "data": [53.0, 65.0] }
    ],
    "measurements": [
      {
        "name": "Measurement",
        "config": {
        "poi": "mu",
        "parameters": []
        }
      }
    ],
    "version": "1.0.0"
  }
```

A.3 4_bin Model

- parameter $[\mu, \theta, \theta_{SF}]$
- ...


```
{
  "channels": [
    { "name": "singlechannel",
      "samples": [
        { "name": "signal MC",
          "data": [2, 3, 4, 5],
          "modifiers": [ { "name": "mu", "type": "normfactor", "data": null } ]
        },
        {
          "name": "bkg MC",
          "data": [30, 19, 9, 4],
          "modifiers": [
            { "name": "theta",
              "type": "histosys",
              "data": { "hi_data": [31, 21, 12, 7], "lo_data": [29, 17, 6, 1] }
            },
            { "name": "SF_theta",
              "type": "normsys",
              "data": {"hi": 1.1,"lo": 0.9}
            }
          ]
        }
      ]
    }
  ],
  "observations": [
    { "name": "mychannel", "data": [34, 22, 13, 11]}
  ],
  "measurements": [
    { "name": "Measurement", "config": {"poi": "mu","parameters": []} }
  ],
  "version": "1.0.0"
}
```

List of Figures

Bibliography

[1] ATLAS Collaboration. Measurements of higgs boson production and couplings in diboson final states with the atlas detector at the lhc. *Physics Letters B*, 726:88–119, 2013.

[2] Albert M Sirunyan, Armen Tumasyan, Wolfgang Adam, Federico Ambrogi, Thomas Bergauer, Johannes Brandstetter, Marko Dragicevic, Janos Erö, Alberto Escalante Del Valle, Martin Flechl, et al. Search for supersymmetry in proton-proton collisions at 13 TeV in final states with jets and missing transverse momentum. *Journal of High Energy Physics*, 2019(10):1–61, 2019.

[3] `pyhf` documentation `v0.6.3`. https://pyhf.readthedocs.io/en/v0.6.3/intro.html.

[4] Lukas Heinrich, Matthew Feickert, and Giordon Stark. Python HistFactory pyhf. https://github.com/scikit-hep/pyhf, 2021.

[5] Jerry Ling, Oliver Schulz, and Gabriel Rabanal. LiteHF.jl, Light-weight HistFactory in pure Julia. https://github.com/JuliaHEP/LiteHF.jl.