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.
HMC	Hamiltonian Monte Carlo.
MCMC	Markov chain Monte Carlo.
NP	nuisance parameter.
pdf	probability density function.
POI	parameter of interest.

Chapter 1

Introduction

Chapter 2

Mathematical Preliminaries

2.1 Frequentist versus Bayesian Inference

2.2 MCMC Sampling

2.2.1 Markov Chains

2.2.2 Metropolis Hastings

2.2.3 Hamiltonian Markov Chains

Chapter 3

HistFactory

3.1 Mathematical Description

3.2 Workspaces ...

Chapter 4

Bayesian Inference with HistFactory

4.1 BAT

Bayesian Analysis Toolkit (BAT)

4.2 batty – BAT to Python Interface

4.3 Priors from `auxdata`

4.4 Gradients for HamiltonianMC

4.5 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 statistic is illustrated in Figure 4.1.

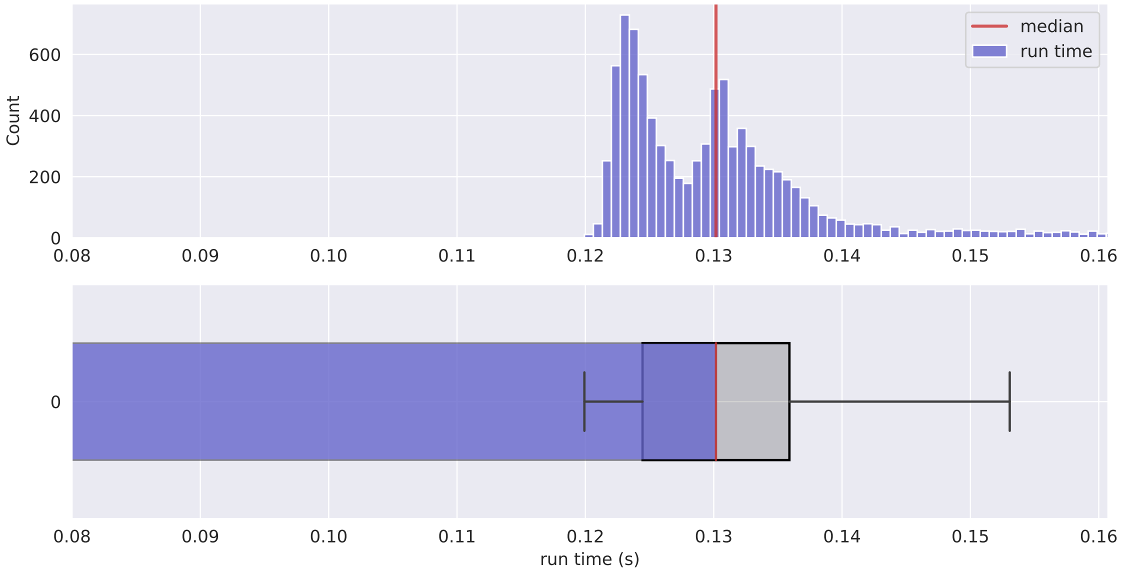


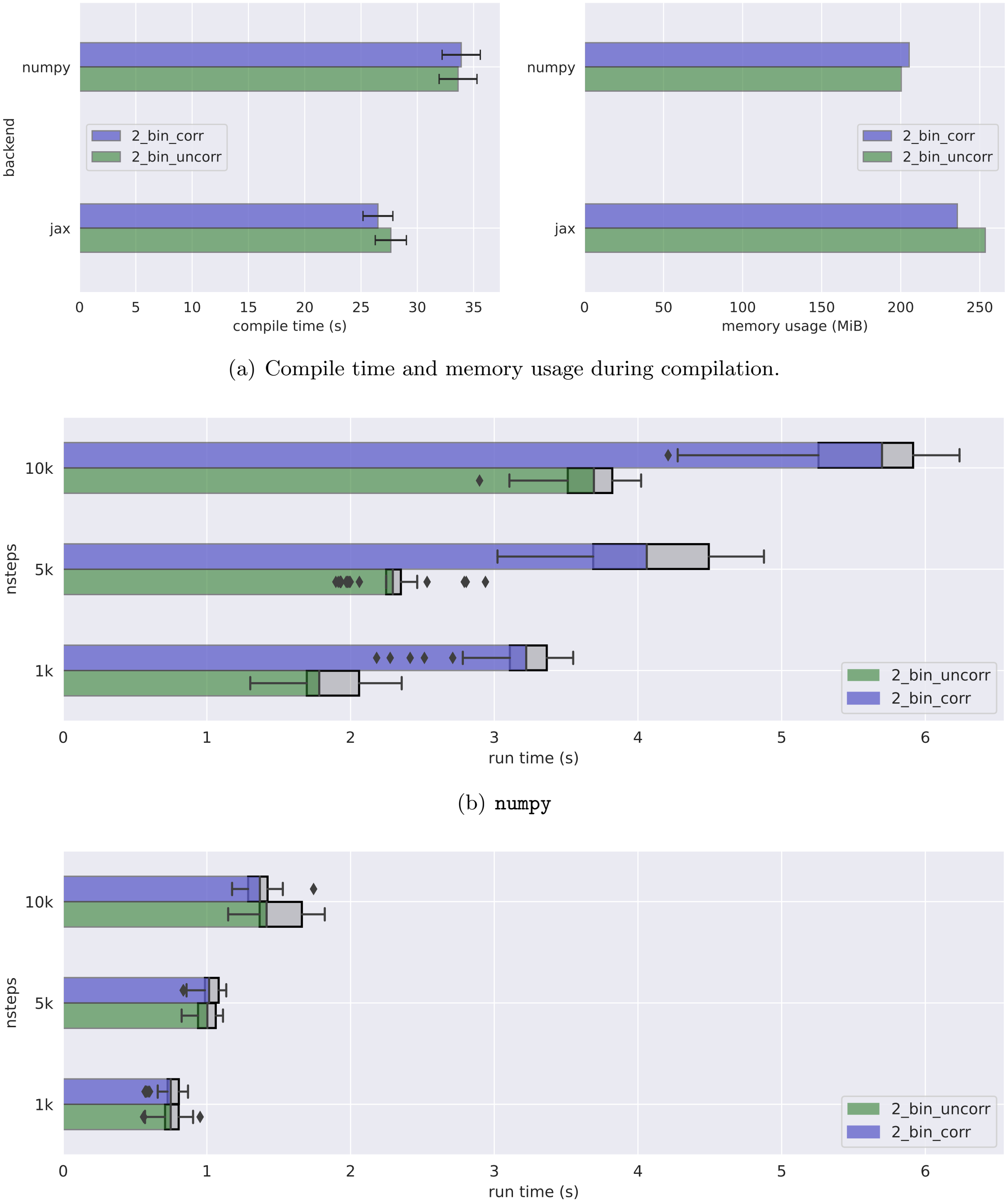
Figure 4.1: Run time statistics of (xx)

- use median instead mean
- use bar + box plot

4.5.1 `pyhf` Backend

- `jax` vs `numpy`
- 2 simplemodels

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



(a) Compile time and memory usage during compilation.

(b) `numpy`

(c) `jax`

Figure 4.2: Compile benchmark 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 XX.

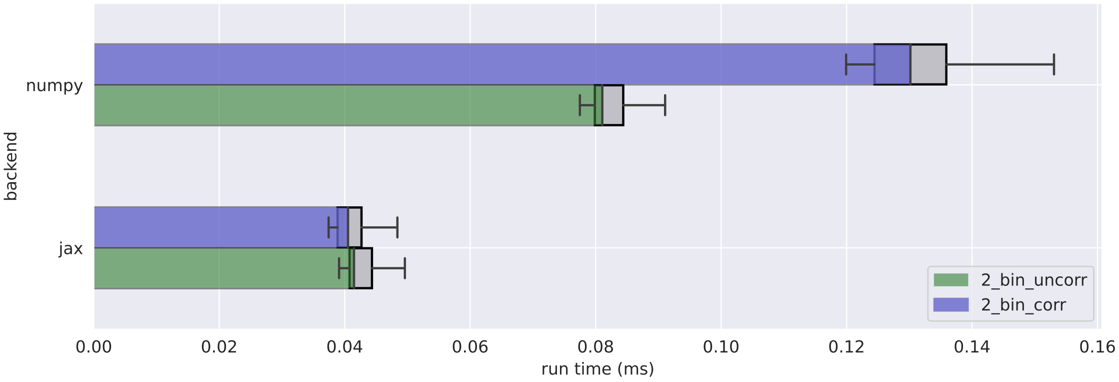


Figure 4.3: Run time of the log likelihood for two different HF models. XX

Chapter 5

Benchmarking the Python-Julia Pipeline against the Julia Implementation

5.1 Overview

Calling Julia code from Python is accomplished by the `PythonCall.jl` module

5.2 Benchmark Setup

-

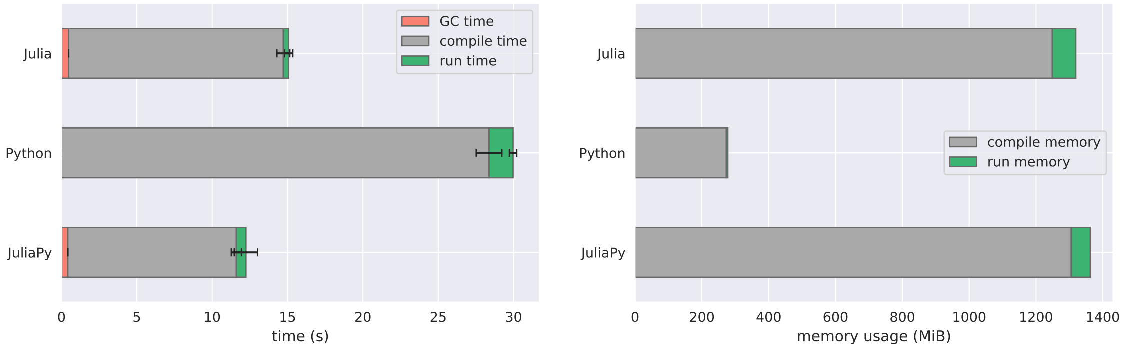
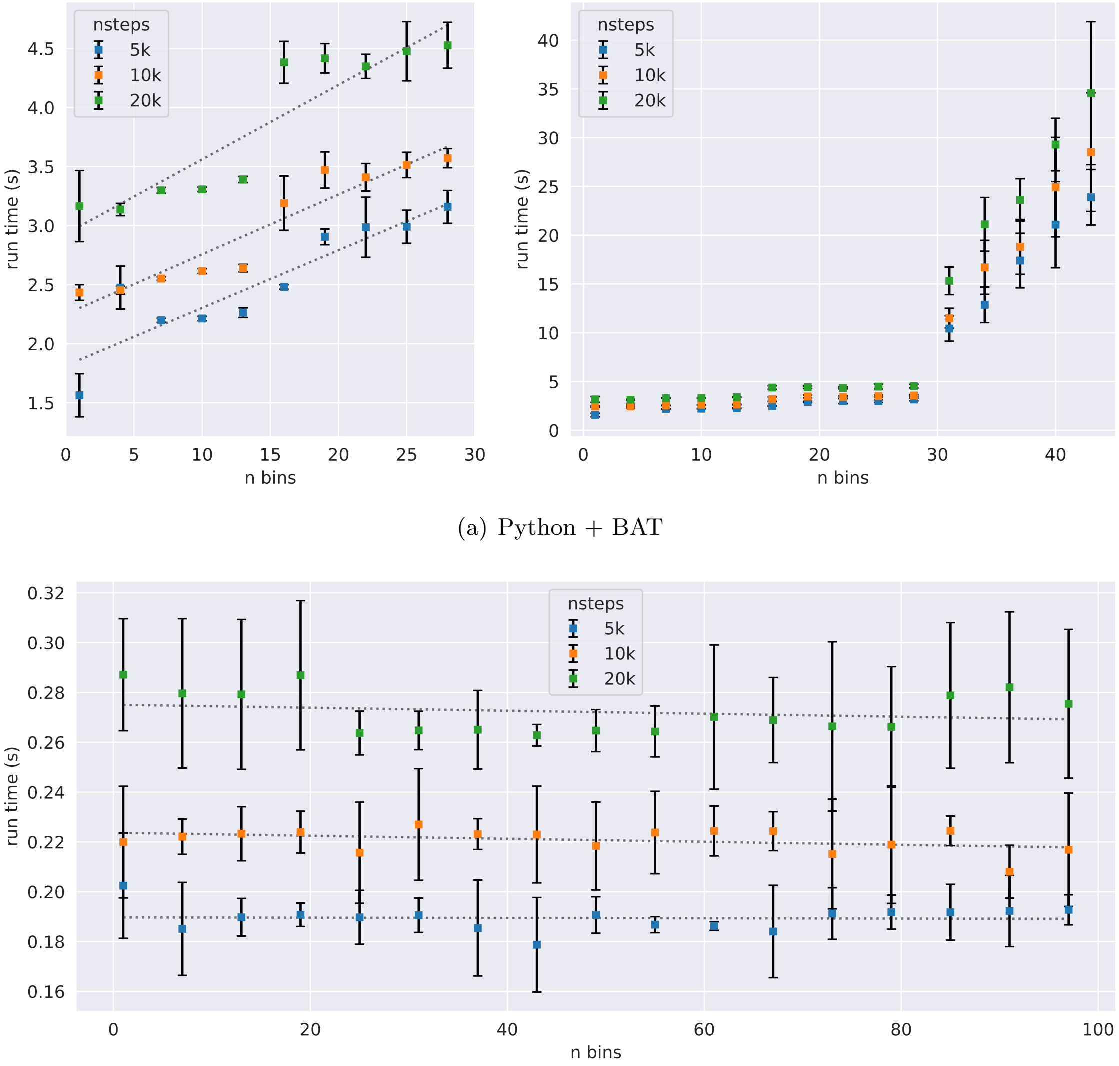


Figure 5.1: Total execution time for `bat_sample()`. (10k steps) ...

5.3 Benchmarks

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

5.3.1 n-Bin-Model



(a) Python + BAT

(b) Julia + LiteHF

Figure 5.2: n-Bin Model. c) Julia + `pyhf` likelihood fehlt (noch keine Benchmarks gemacht, da LZ intensiv, in nÃ¤chster Version.)

- simple model with uncorrelated background and n bins
- pure Julia ”stable” runtime over 1 to 100 dimensional parameter vector

  (Ich check das nochmal ..)

- Python jump at 30 dim parameter vector

5.3.2 Benchmarks for the "complex" Model

TODO: complex Model benchmarks

- In Progress..
- Ich hab mal Plots erzeugt, fur die simplemodels

  Hier hab ich nur gerade was durcheinandergebracht, ist in der naechsten Version drin (vmtl. mit dem complex model.)
- Ergebnis vorab:
  - pure-Julia ist wieder 10 x Schneller als Python + BAT
  - Python + BAT etwa vergleichbare Laufzeit wie Julia + `pyhf` likelihood

List of Figures

Bibliography

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

[2] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 9 2017.

[3] Kyle Cranmer, Akira Shibata, Wouter Verkerke, Lorenzo Moneta, and George Lewis. Histfactory: A tool for creating statistical models for use with roofit and roostats. Technical report, 2012.

[4] Kyle Cranmer. Practical statistics for the LHC. *arXiv preprint arXiv:1503.07622*, 2015.

[5] Oliver Schulz, Frederik Beaujean, Allen Caldwell, Cornelius Grunwald, Vasyl Hafych, Kevin Kröninger, Salvatore La Cagnina, Lars Röhrig, and Lolian Shtembari. Bat.jl: A julia-based tool for bayesian inference. *SN Computer Science*, 2(3):210, Apr 2021.