# Practical Modelling in Parallel:

## Benchmarking the Highly-Parallel Performance of Bayesian Sampling in both Python and Julia.

Cian Roche
6.338 Project
Massachusetts Institute of Technology

March 29, 2022

The estimation of model parameters via Bayesian sampling techniques such as MCMC or nested sampling is an integral part of modern science, appearing frequently in the fields of astrophysics, biology, linguistics, nuclear physics, cryptography, political science, and many more. As problems become more high-dimensional and intractable, parallel computing grows only more valuable as a resource in these fields. However, most of the researchers making use of these sampling techniques do not have broad backgrounds in computer science, or perhaps have only engaged in formal education in interpreted languages like python and R. To such a population, the barrier to using a compiled language such as C++ or Julia is relatively high, due to less accessible documentation for non-experts, relatively smaller amounts of example code available easily online, and potentially greater complexity in the coding and debugging processes.

This said, the performance gains when using a compiled language are a significant advantage, in particular for the large-scale computations which are good candidates for parallelism. As a result, modern researchers are faced with the decision between a **potentially** smoother development experience, and **potentially** much faster and more efficient code. As the representative languages we choose python (due to its simplicity and popularity) and Julia (due to its speed and relative user-friendliness for a compiled language). The aim of this project is to illuminate, in the context of Bayesian sampling as a means of model fitting, if indeed Julia outpaces python considerably when appropriate use is made of packages like Cython and Numpy, and if the implementation of a highly-parallel Bayesian sampler is in fact more time-consuming in Julia for a non-expert. Importantly, most researchers would not code their own sampling routines, instead relying on existing packages. As a result, we benchmark appropriate packages in both Python (emcee) and Julia (AffineInvariantMCMC and/or Mamba) for performing Bayesian sampling.

# Contents

# 1 Background

## 1.1 Markov Chain Monte Carlo

### 1.1.1 Metropolis-Hastings

### 1.1.2 Affine-Invariant MCMC

## 1.2 Nested Sampling

## 1.3 Test Problem Definition

## 1.4 Parallel Computing Environment

Will use 16 nodes of MIT engaging cluster (64 CPUs per node), to which I have unrestricted access.

# 2 Python Implementation

## 2.1 Details of emcee

## 2.2 Test Problem Benchmarks

## 2.3 Bottlenecks and Optimizations

# 3 Julia Implementation

## 3.1 Details of AffineInvariantMCMC

## 3.2 Test Problem Benchmarks

## 3.3 Bottlenecks and Optimizations

# 4 Comparison

# 5 Conclusions and Recommendations

# Appendix