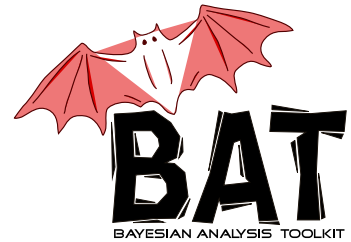
The Bayesian Analysis Toolkit (BAT)

Oliver Schulz
on behalf of the BAT team

MAX-PLANCK-GESELLSCHAFT

Max-Planck-Institut für Physik
(Werner-Heisenberg-Institut)



oschulz@mpp.mpg.de

Workshop on state of the art in sampling and clustering, Oct. 9th, 2020

Introduction

- The Bayesian Analysis Toolkit (BAT):
  A software package for Bayesian inference
- Typical tasks: Given a set of data and prior knowledge
  - estimate parameters
  - compare models (Bayes factors)

  according to Bayes theorem

$$P(\vec{\lambda}|\vec{D}) = \frac{P(\vec{D}|\vec{\lambda})P_0(\vec{\lambda})}{\int P(\vec{D}|\vec{\lambda})P_0(\vec{\lambda})\,d\vec{\lambda}}$$

Introduction

- The Bayesian Analysis Toolkit (BAT):
  A software package for Bayesian inference
- Typical tasks: Given a set of data and prior knowledge
  - estimate parameters
  - compare models (Bayes factors)

  according to Bayes theorem

$$P(\vec{\lambda}|\vec{D}) = \frac{P(\vec{D}|\vec{\lambda})P_0(\vec{\lambda})}{\int P(\vec{D}|\vec{\lambda})P_0(\vec{\lambda})\,d\vec{\lambda}}$$

- Functionalities
  - Posterior space exploration via
    Markov chain Monte-Carlo (MCMC)
  - Integration of non-normalized posterior
    (i.e. evidence calculation)
  - User-friendly plotting and reporting

Development History

- Original: BAT-C++ v1.0
- Very successful over the years, > 250 citations (INSPIRE)

  [Caldwell et al., DOI: 10.1016/j.cpc.2009.06.026 (2009).]

  - Written in C++, based on CERN ROOT
  - Proven in many real-life use cases
  - Wide user base, esp. in high-energy, nuclear and astro-physics

Development History

- Original: BAT-C++ v1.0
- Very successful over the years, > 250 citations (INSPIRE)

  [Caldwell et al., DOI: 10.1016/j.cpc.2009.06.026 (2009).]

  - Written in C++, based on CERN ROOT
  - Proven in many real-life use cases
  - Wide user base, esp. in high-energy, nuclear and astro-physics
- By now reached flexibility limit of original software design
- Started complete re-design in 2017

Design goals for new version of BAT

- Core philosophy: User provides likelihood
  (typically expensive, high data volumes, etc.)
  BAT does the rest
- Easy to use with defaults, but allow for detailed fine-tuning
- Multiple MCMC algorithms
  (BAT-C++ only supports Metropolis-Hastings)
- Deep support for parallel operation:
  - Parallelize both likelihood and MCMC chains
  - Local (multiple threads) plus distributed (compute clusters)
- Auto-differentiation for mode-finding, HMC, etc.
- Choice of programming language?

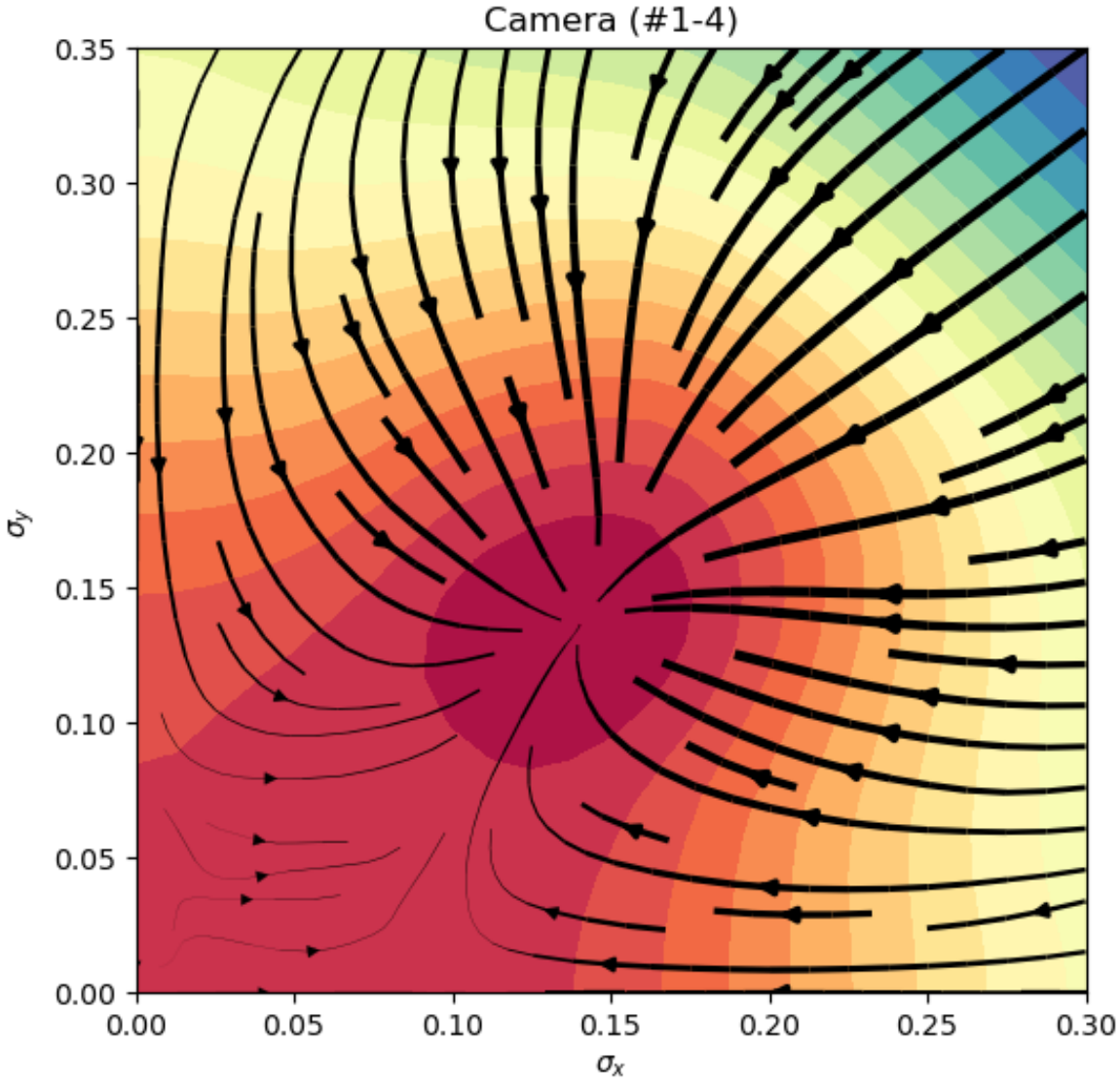
Choice of Programming Language

- New version: BAT.jl, written in Julia
- C++: Complex, takes long to learn, multi-platform tricky, interactive use requires ROOT/Cling
- Python: Hard to write high-performance likelihoods without falling back to C/C++

Choice of Programming Language

- New version: BAT.jl, written in Julia
- C++: Complex, takes long to learn, multi-platform tricky, interactive use requires ROOT/Cling
- Python: Hard to write high-performance likelihoods without falling back to C/C++
- Julia provides:
  - C/C++ performance with Python-like simplicity
  - excellent multi-threading
  - native cluster-computing
  - excellent auto-differentiation
  - native GPU computing
  - ability to easily call C, Fortran, C++, Python, R, ...
  - superior code composability due to multiple dispatch

Autodifferentiation in Julia

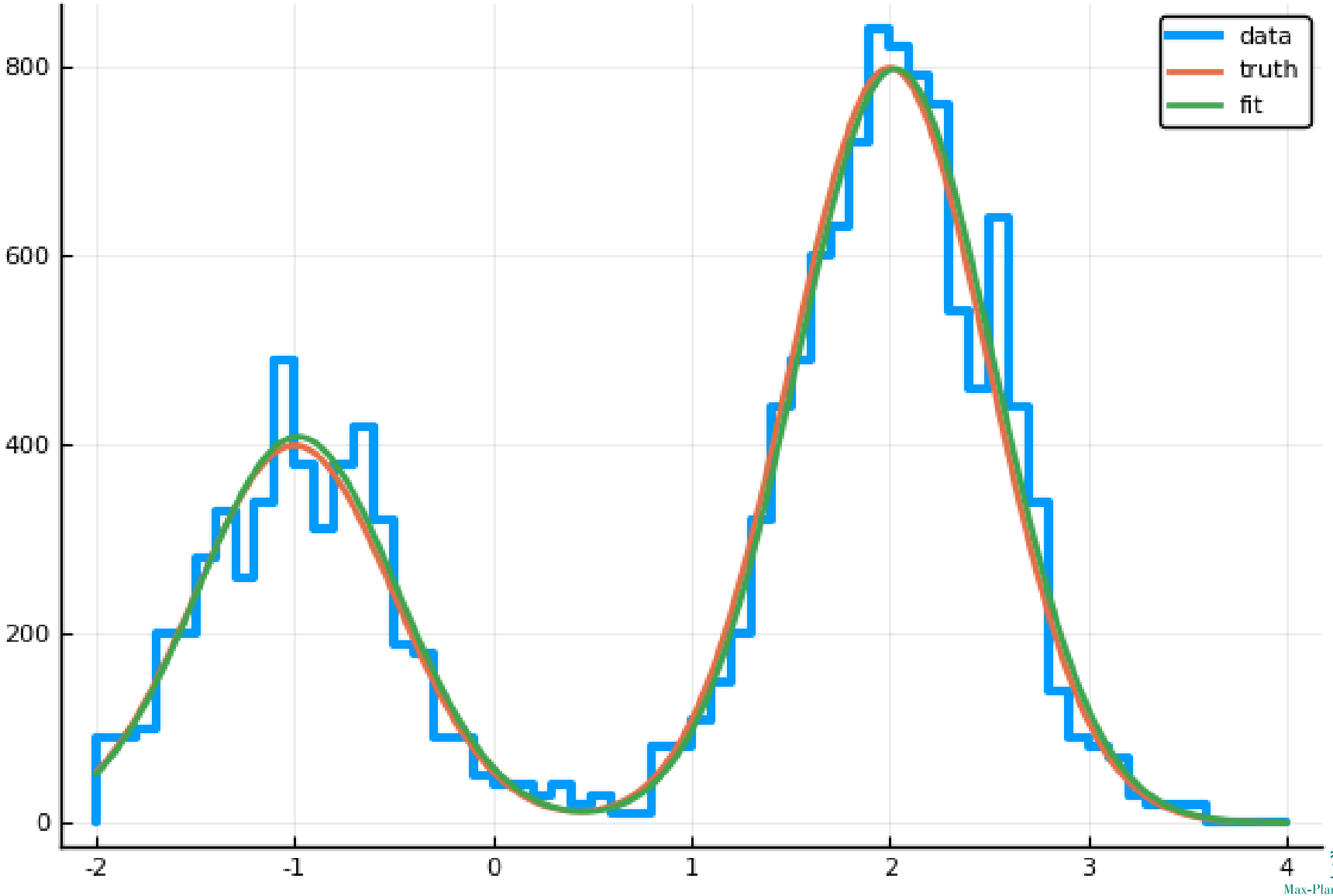


Julia powerful enough to allow for AD of (almost arbitrary code).

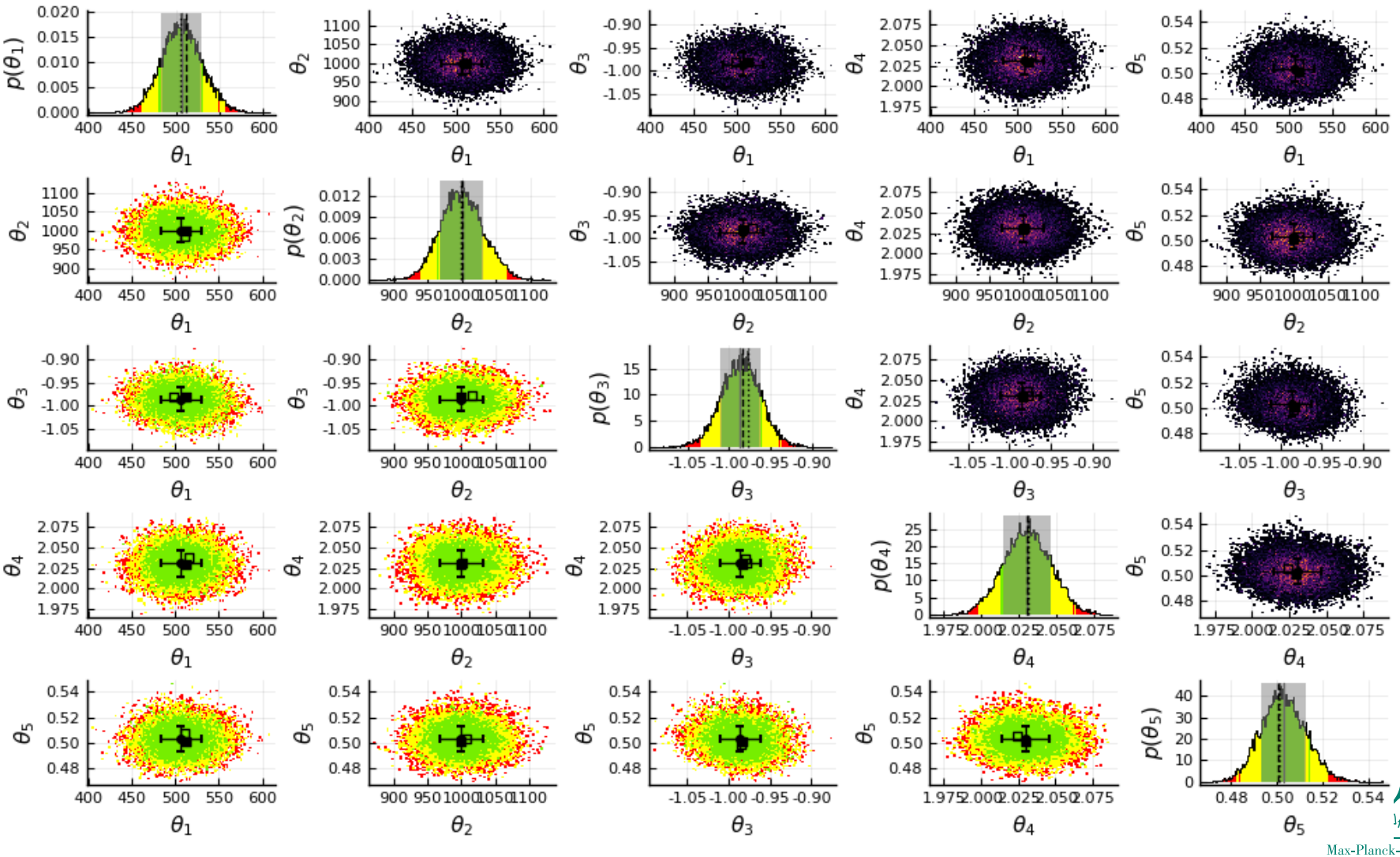
BAT.jl

- MCMC with Metropolis-Hastings (like BAT-C++), also HMC (Stan-like) now
- New ARP sample weighting scheme
- New posterior integration algorithm AHMI
- Release of v1.2 imminent
- https://github.com/BAT/BAT.jl

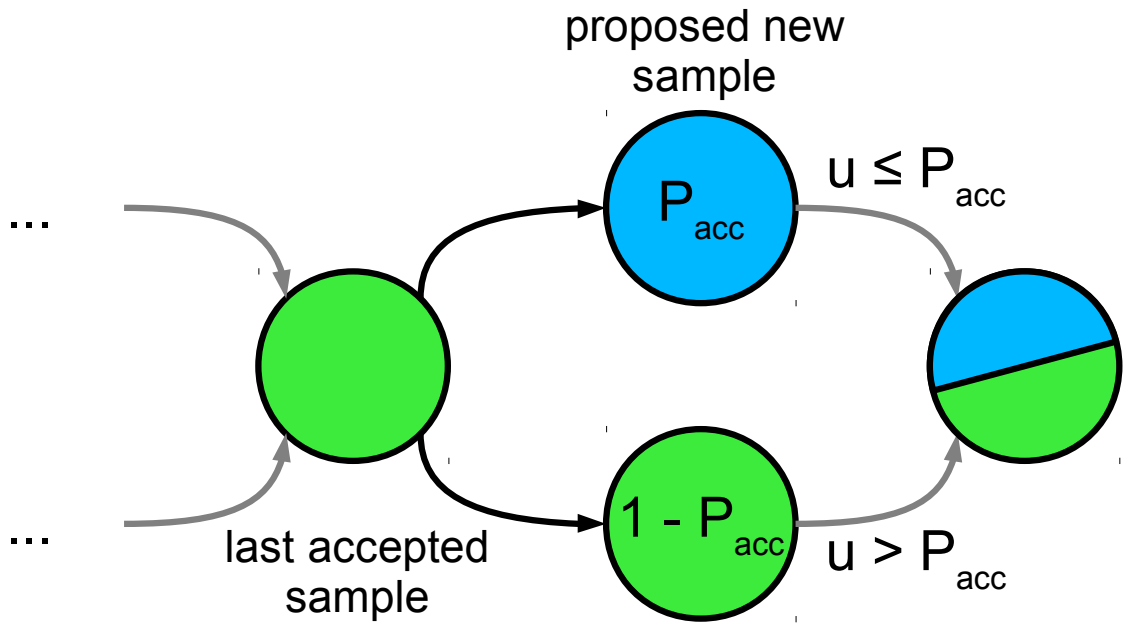
Simple BAT.jl example: Fit Histogram



BAT.jl plotting: Posterior projections

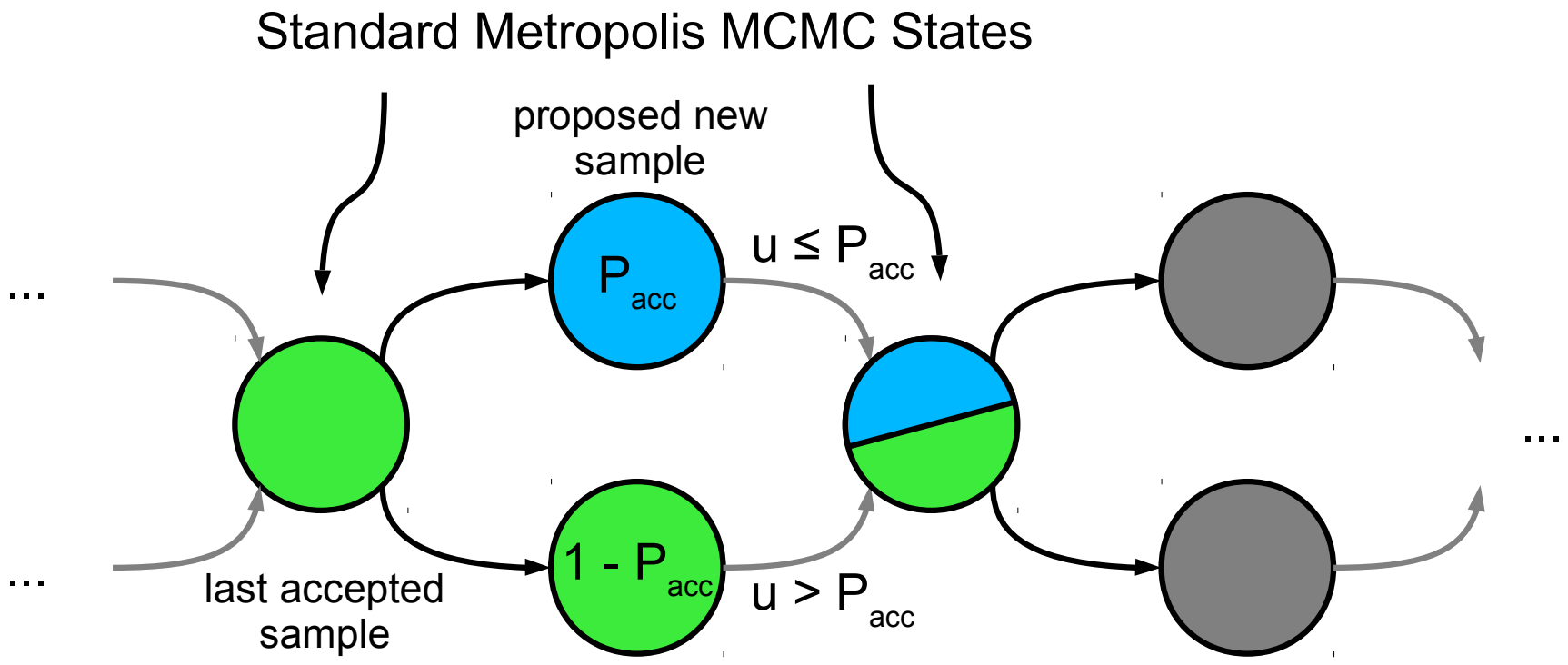


Accept/Reject Probabiliy (ARP) Weights



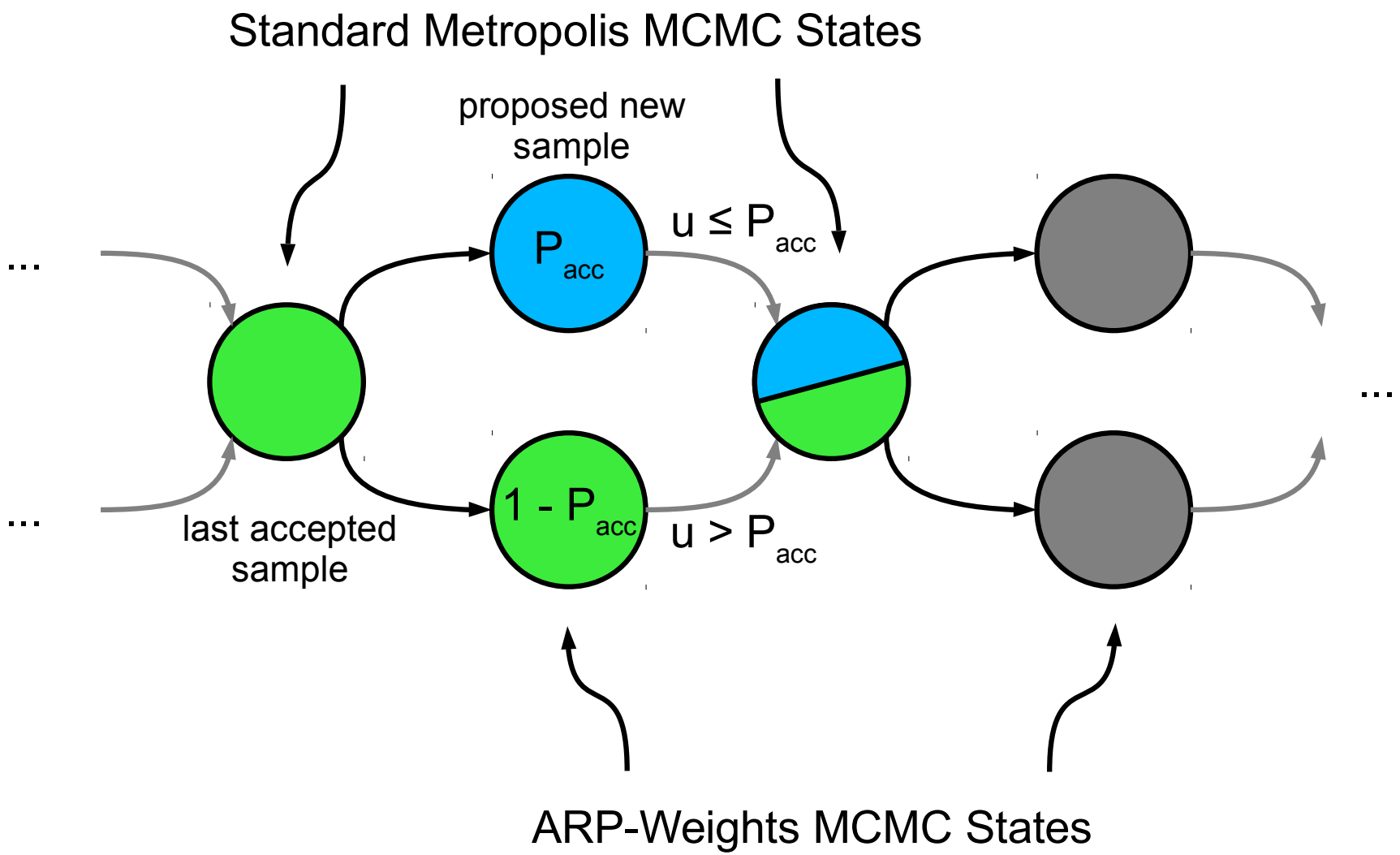
- Schrödinger's cat for samples

Accept/Reject Probabiliy (ARP) Weights



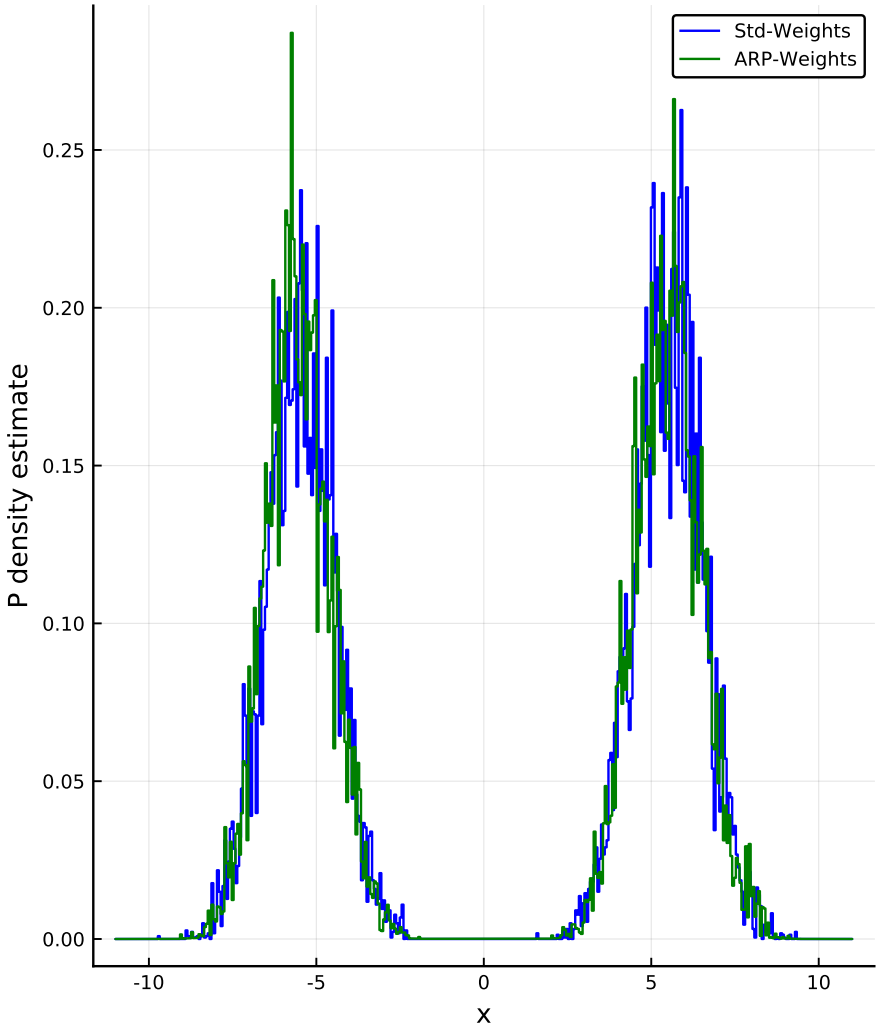
- Schrödinger's cat for samples

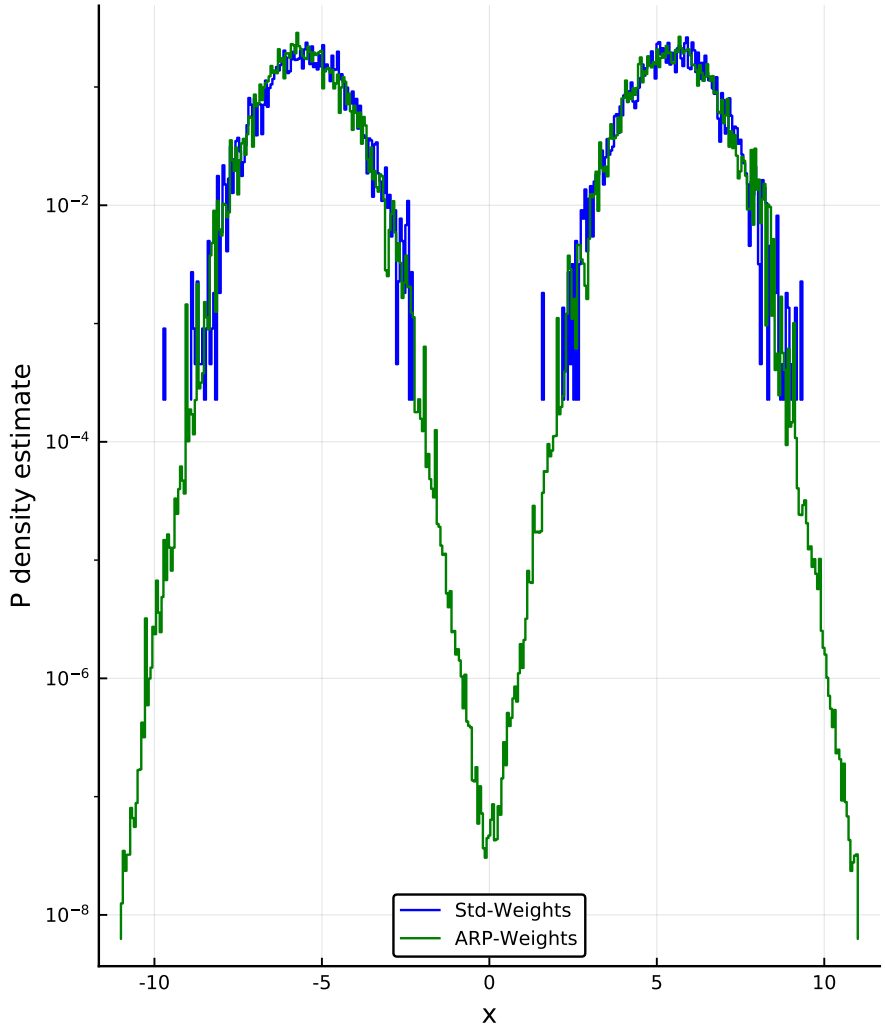
Accept/Reject Probabiliy (ARP) Weights



- Schrödinger's cat for samples

ARP Sample Weights





- New sample weighting scheme keeps rejected samples, allows insight in low-probability posterior areas

Max-Planck-Institut für Physik
(Werner-Heisenberg-Institut)

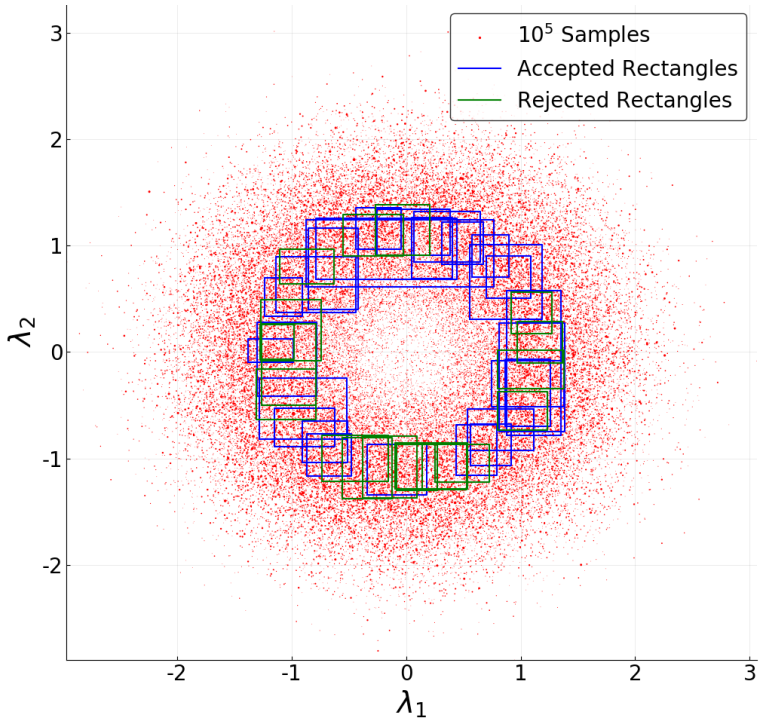
Adaptive Harmonic Mean Integration

- Given N samples Λ_i drawn from posterior, can estimate integral via harmonic mean:

$$\hat{I} \equiv \frac{NV}{\sum_{i=1}^{N} \frac{1}{f(\Lambda_i)}} \; .$$

- Problem: variance of estimator diverges in the general case
- AHMI algorithm:
  - Whiten sample distribution
  - Generate set of hyper-rectangles with bounded variance
  - Compute individual harmonic mean integrals
  - Estimate correlation of integrals
  - Combine integrals into robust overall integral with error estimate

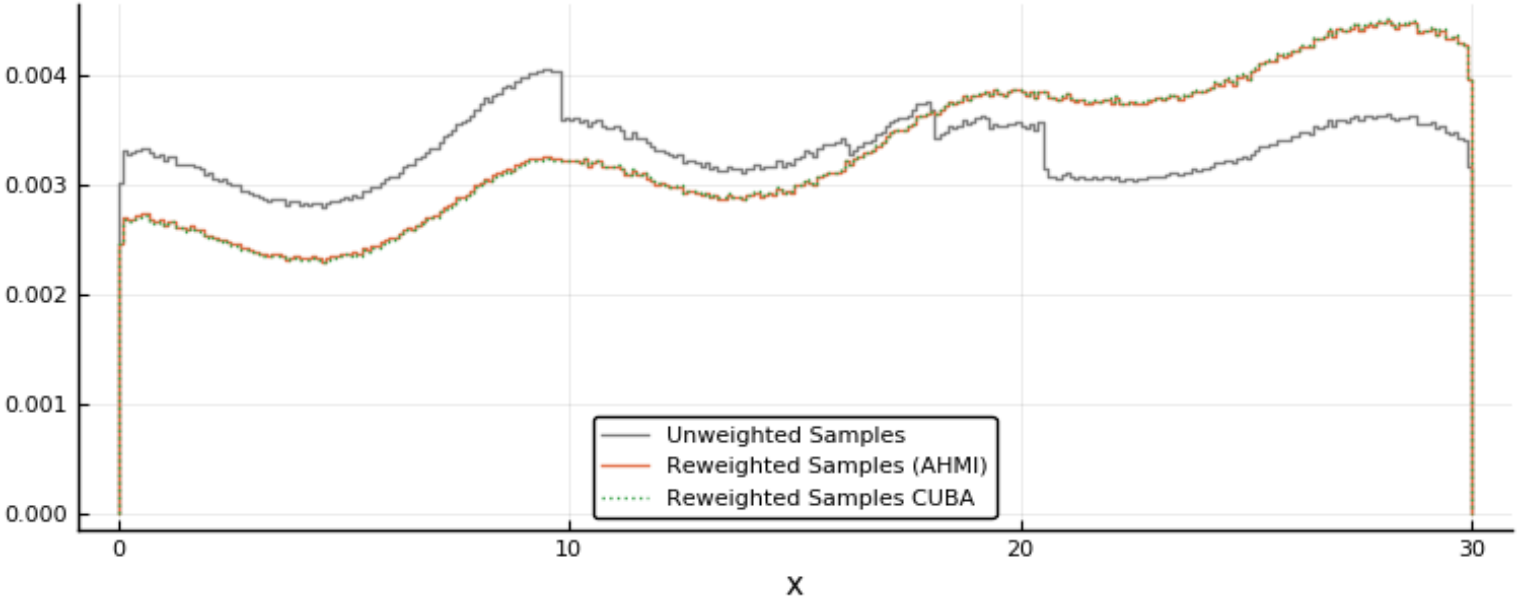
Adapive Harmonic Mean Integration (AHMI)

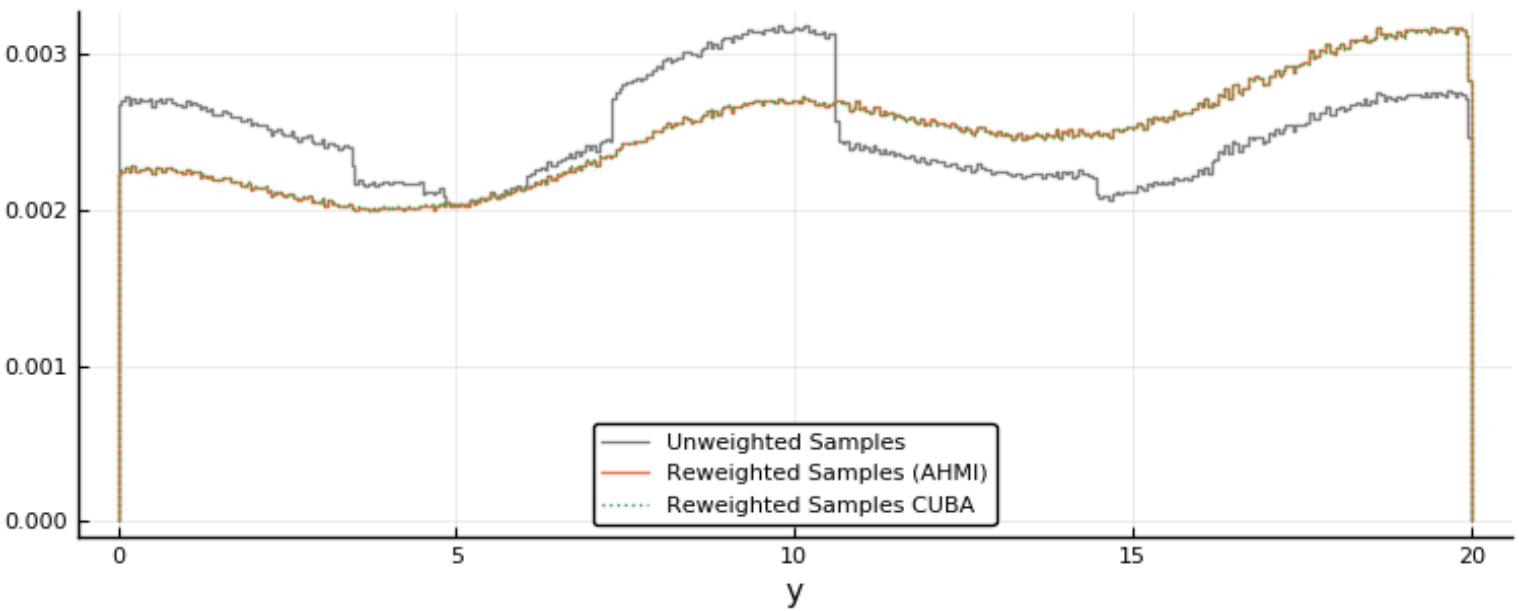


- Computes posterior integral/evidence from samples via harmonic mean [arXiv:1808.08051 (2018)]
- Operates in hyper-rectangles with limited posterior variance to control integral variance

Max-Planck-Institut für Physik
(Werner-Heisenberg-Institut)

Space partitioning AHMI reweighting





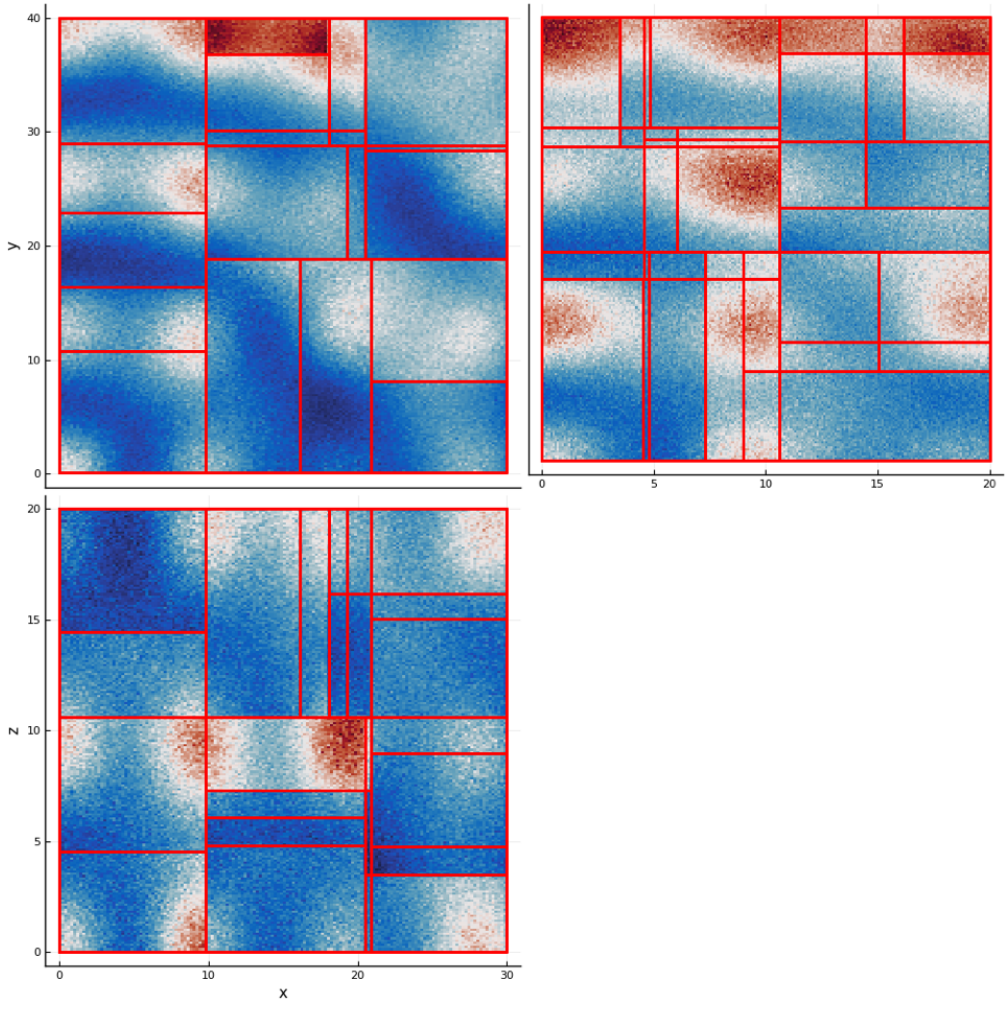
Parameter space partitioning (Experimental)

- MCMC expensive, need maximum parallelization
- Parallelization potential of likelihood often limited
- Increasing number of chains doesn't help (burn-in cost)

Parameter space partitioning (Experimental)

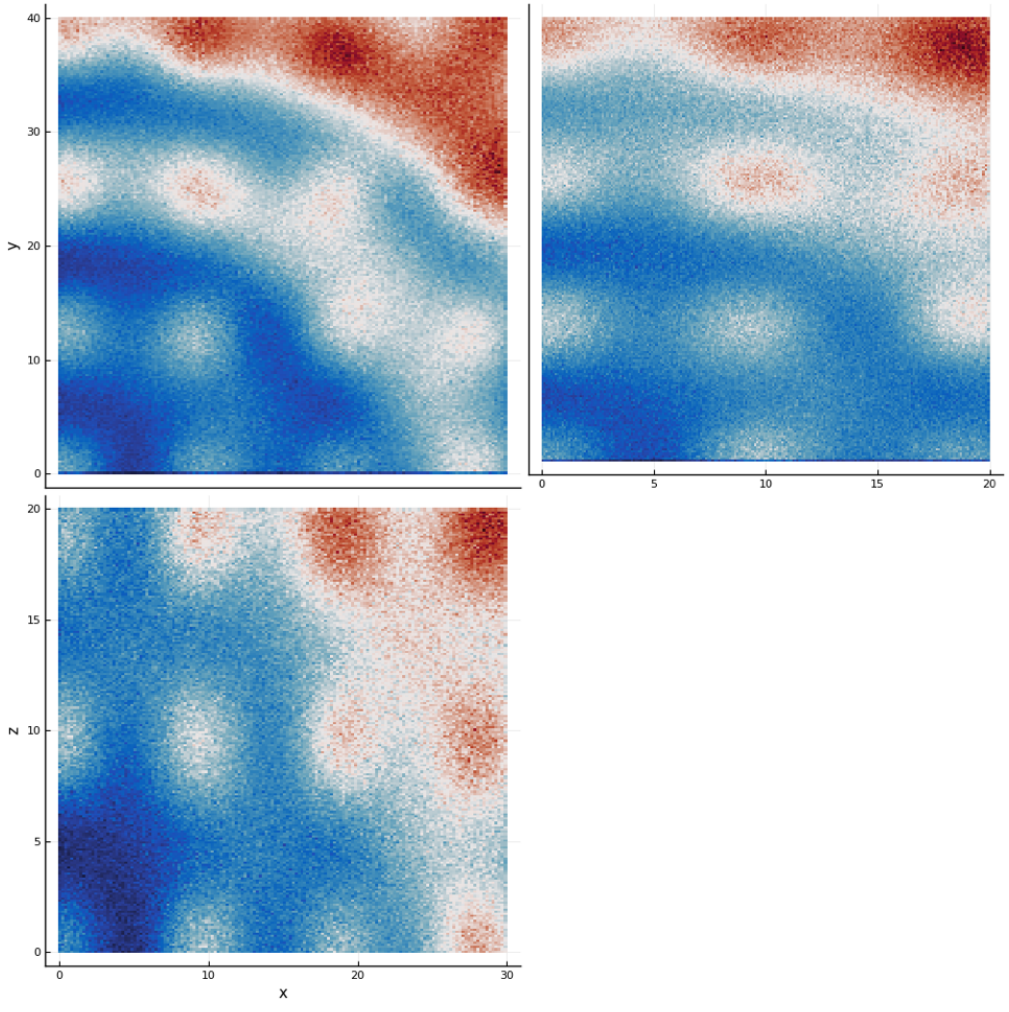
- MCMC expensive, need maximum parallelization
- Parallelization potential of likelihood often limited
- Increasing number of chains doesn't help (burn-in cost)
- New concept: partition parameter space
  run separate set of chains in each subspace
- Rationale: posterior in small subspaces simpler,
  fast burn-in
- Challenge: find good partitioning for given posterior,
  work in progress

Parameter Space Partitioning, Raw



- Subspaces contains unequal probability mass: can't just stitch MCMC results together

Parameter Space Partitioning, Reweight



- Solution: Use AHMI to integrate posterior in each subspace, then reweight by integral

Conclusions and Outlook

- BAT concept:
  user brings domain knowledge and likelihood,
  BAT provides robust sampling, integration and visualization
- Current BAT (C++) is a success story,
  but flexibility limit reached
- Release of BAT.jl (Julia) v2.0 imminent,
  focus on ease of use, parallelization and modern algorithms
- Future improvements: Lot's of ideas, but will strive for quality, not quantity (of algorithms, etc.), ...

Appendix