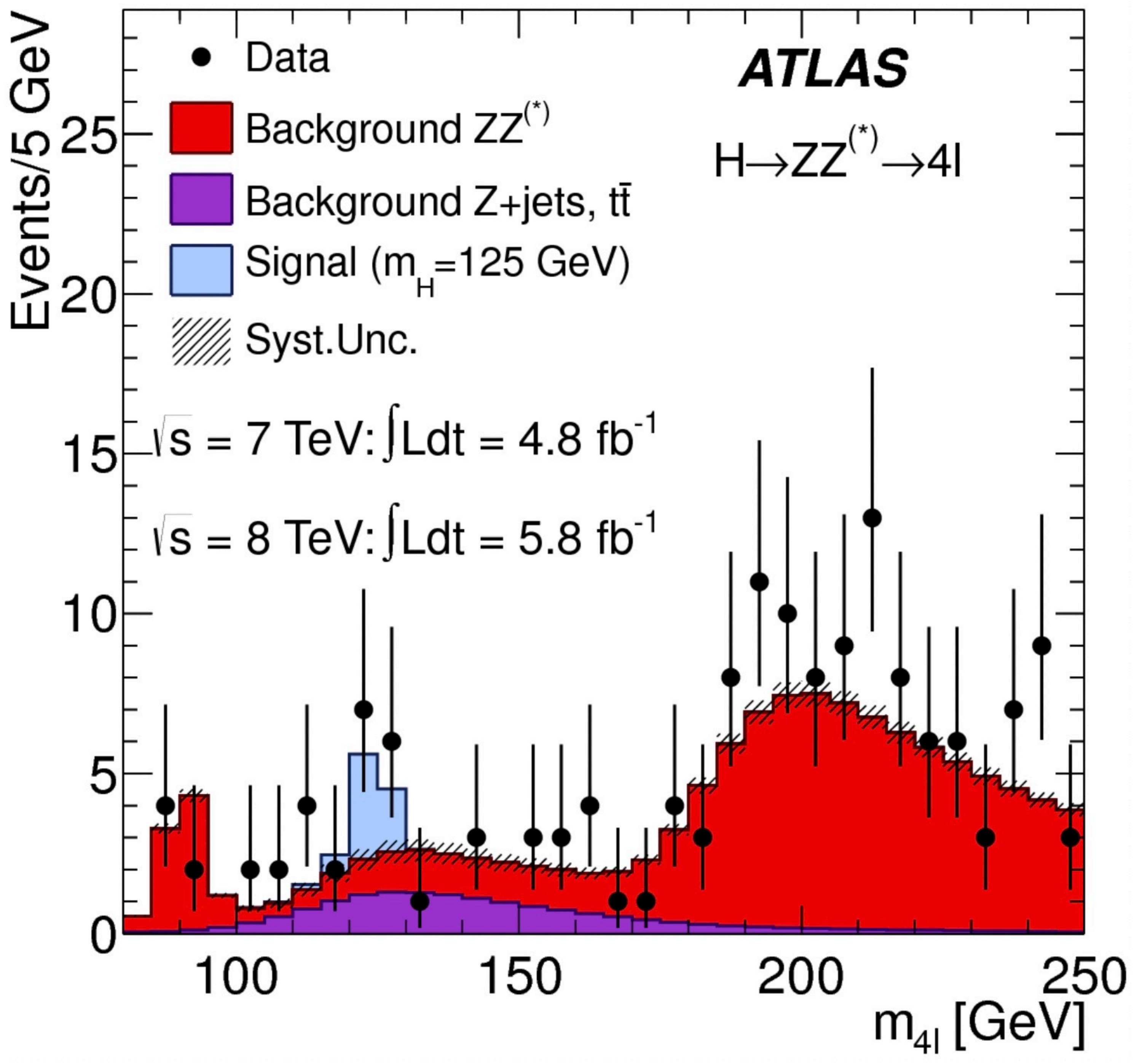
Bayesian Methodology for Particle Physics with `pyhf`

CHEP23, 08.05.2023

Matthew Feickert, Lukas Heinrich, Malin Horstmann (malin.horstmann@tum.de)

HistFactory

- We need a data-generating model for inferring parameters of nature
- HistFactory: Toolbox for creating these models for binned analyses



https://cds.cern.ch/record/1456844?ln=en

https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PAPERS/HIGG-2012-27/

HistFactory

- We need a data-generating model for inferring parameters of nature
- HistFactory: Toolbox for creating these models for binned analyses
- Probability model:

$\vec{\eta}$: Unconstrained Parameters (incl. POI)

$\vec{\chi}$: Nuisance parameters

$$p\left(\vec{n}, \vec{a} \,|\, \vec{\eta}, \vec{\chi}\right) = \prod_{c \,\in\, \text{Channels}} \prod_{b \,\in\, \text{Bins}} \text{Poiss}\left(n_{cb} \,|\, \nu_{cb}(\vec{\eta}, \vec{\chi})\right) \prod_{\chi \in \vec{\chi}} c_\chi\left(a_\chi \,|\, \chi\right)$$

$\vec{n}$: Bin counts

$\vec{a}$: Auxiliary data

Poisson-distributed bin counts

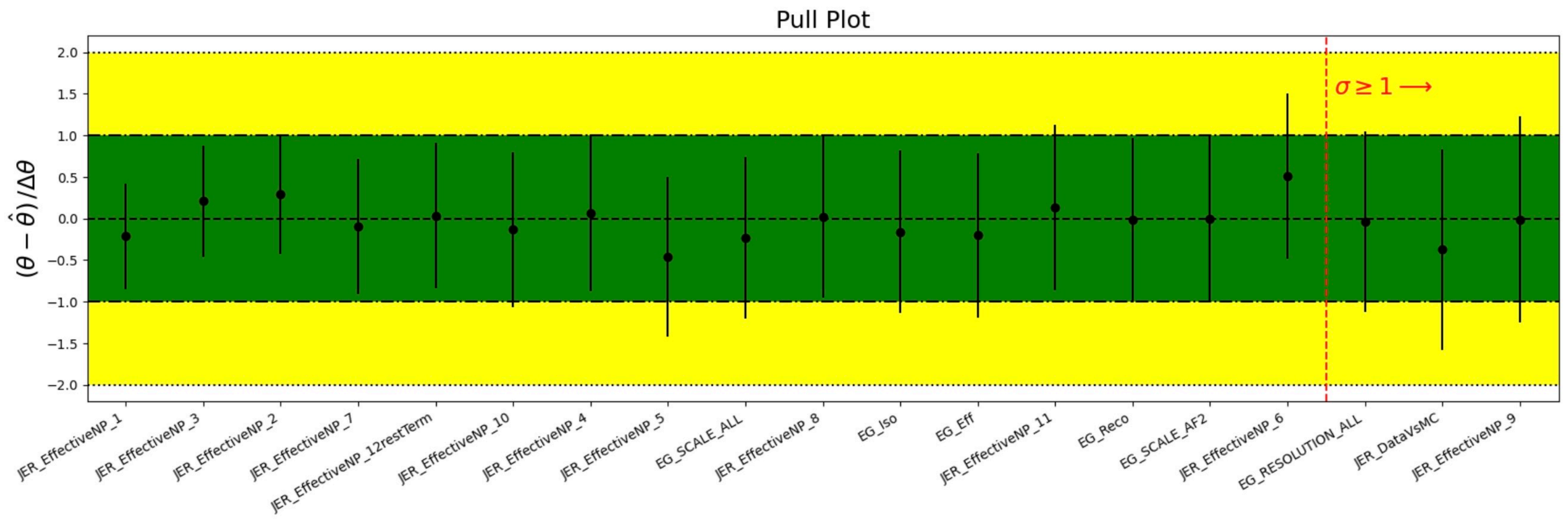
Constraint terms

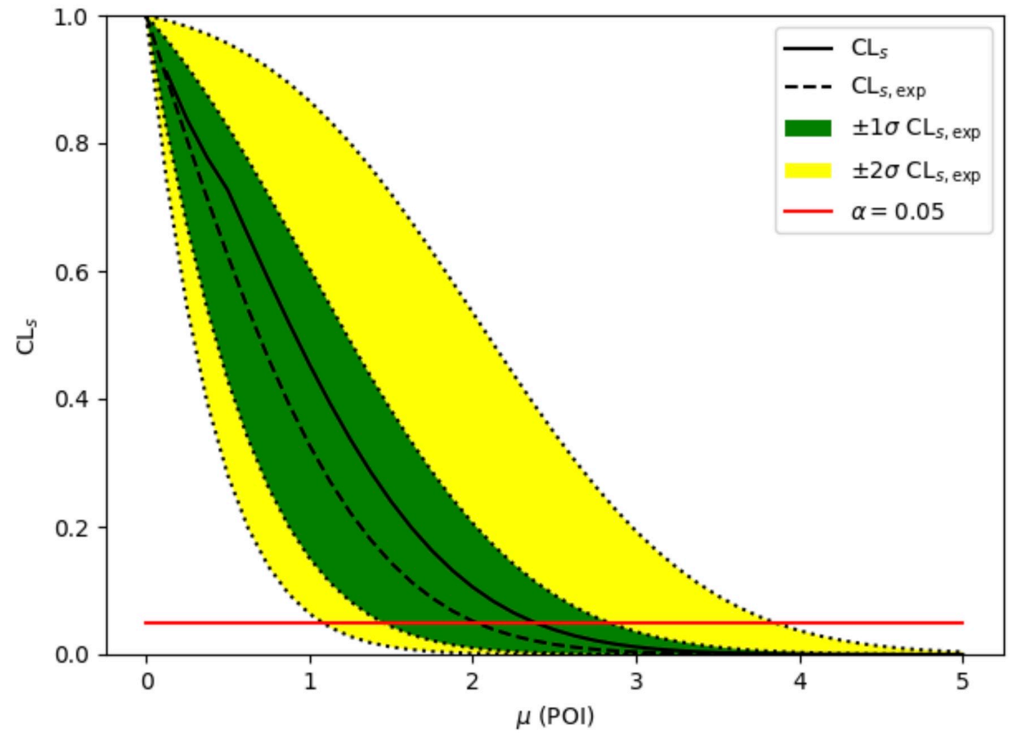
https://cds.cern.ch/record/1456844?ln=en

pyhf

- A pure Python implementation of the HistFactory statistical model, including tools for Frequentist inference
- Supports auto-differentiation via different computational backends (e.g. `jax`)

Before this work:





After this work: Priors and Posteriors

https://github.com/scikit-hep/pyhf

Bayesian Inference

- Updating a prior belief - after taking observations $\vec{n}, \vec{a}$ into consideration - via Bayes theorem:

$$p\left(\vec{\eta}, \vec{\chi} \,|\, \vec{n}, \vec{a}\right) \approx p\left(\vec{n}, \vec{a} \,|\, \vec{\eta}, \vec{\chi}\right) \; p\left(\vec{\eta}\right) \; p\left(\vec{\chi}\right)$$

Posterior distributions

`pyhf` HistFactory statistical model

Prior distributions

Bayesian Inference

- Updating a prior belief - after taking observations $\vec{n}, \vec{a}$ into consideration - via Bayes theorem:

$$p\left(\vec{\eta},\vec{\chi}\,|\,\vec{n},\vec{a}\right) \approx p\left(\vec{n},\vec{a}\,|\,\vec{\eta},\vec{\chi}\right)\; p\left(\vec{\eta}\right)\; p\left(\vec{\chi}\right)$$

Posterior distributions

`pyhf` **HistFactory** statistical model

Prior distributions: How do we get those?

$$p\left(\vec{\eta},\vec{\chi}\,|\,\vec{n},\vec{a}\right) \approx p\left(\vec{n}\,|\,\vec{\eta},\vec{\chi}\right)\; p\left(\vec{\eta}\right)\; p\left(\vec{\chi}\,|\,\vec{a}\right)$$

https://github.com/scikit-hep/pyhf

$$p\left(\vec{\eta},\vec{\chi}\,|\,\vec{n},\vec{a}\right) \approx p\left(\vec{n}\,|\,\vec{\eta},\vec{\chi}\right)\; p\left(\vec{\eta}\right)\; p\left(\vec{\chi}\,|\,\vec{a}\right)$$

Building Prior Distributions

- Turn information from **auxiliary measurements** (means, rates, uncertainties) into prior distributions for $\vec{\chi}$ using **conjugate priors**:

$$p\left(\vec{\chi}\,|\,\vec{a}\right) \approx p\left(\vec{a}\,|\,\vec{\chi}\right)\; p_{\mathrm{ur}}\left(\vec{\chi}\right)$$

$p\left(a\,\|\,\chi\right)$	$\propto \mathrm{Poiss}\left(a\,\|\,\chi\right)$	$\propto \mathcal{N}\left(a\,\|\,\mu = 0,\ \sigma = 1\right)$
$p\left(\chi\,\|\,a\right)$	$\propto \Gamma\left(\chi\,\|\,\alpha = a, \beta = a\right)$	$\propto \mathcal{N}\left(\chi\,\|\,\mu = 0,\ \sigma = 1\right)$

Priors for the nuisance parameters $\vec{\chi}$

pyhf + PyMC

- PyMC: Probabilistic programming Python library for Bayesian analysis based on MCMC methodology
- Already includes a wide range of MCMC techniques, cross-checks, …
- PyMC allows for the implementation of external models
- Example inference code:

```
import pyhf
import pymc

with pyhf.infer.bayes(model, prior, data):
    posterior = pymc.sample(10_000)
    posterior_predictive = pymc.sample_posterior_predictive(posterior)
    prior = pymc.sample_prior_predictive(10_000)
```


https://github.com/pymc-devs/pymc

Example Bayesian Workflow

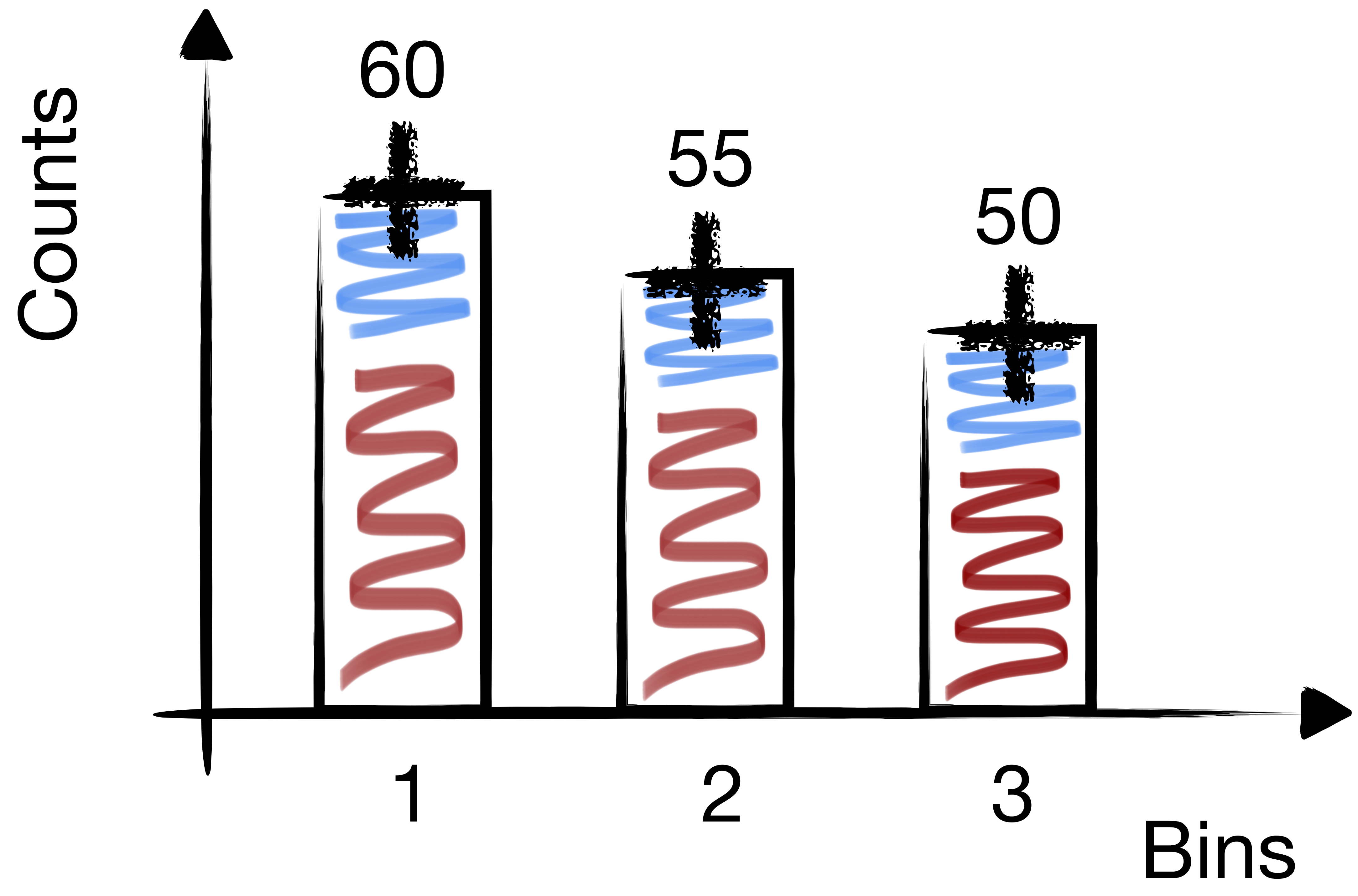
Model

- Bin counts: $N = \eta\ s\ +\ \chi\ b$

Background (Nuisance): [50,40,30]

Signal Strength (Unconstrained): [10, 15, 20]

- Data: [60, 55, 50]



Example Bayesian Workflow

1. Set up statistical model (`pyhf`)
2. Build priors
3. Testing the model:

Checking prior and posterior predictive

Check consistency: Calibration

Inference Quality: Autocorrelation length

…

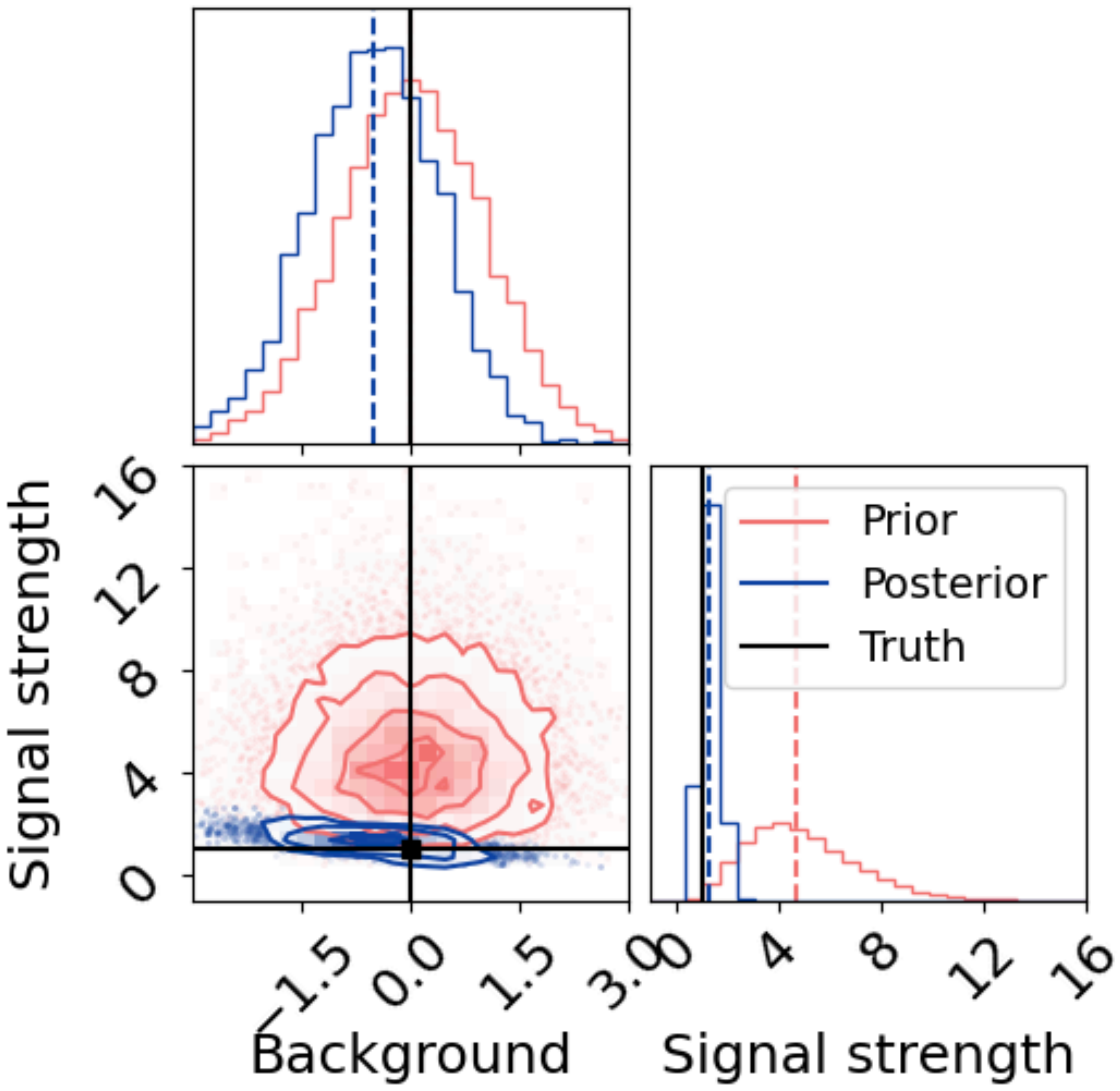
https://betanalpha.github.io/assets/case_studies/principled_bayesian_workflow.html

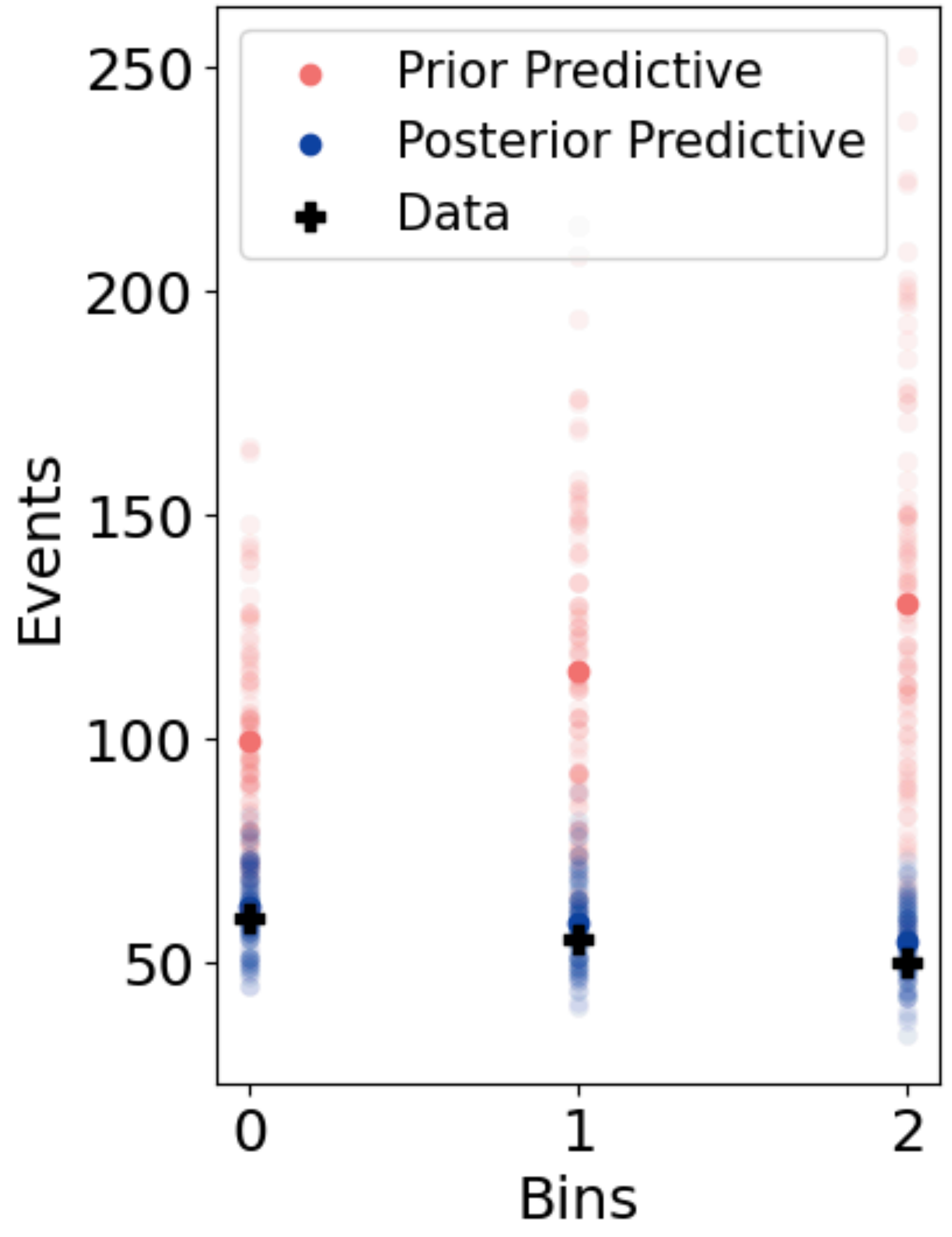
Example Bayesian Workflow

$$p\left(\vec{\eta},\vec{\chi}\,|\,\vec{n},\vec{a}\right) \approx p\left(\vec{n}\,|\,\vec{\eta},\vec{\chi}\right)\; p\left(\vec{\eta}\right)\; p\left(\vec{\chi}\,|\,\vec{a}\right)$$

$$N = \eta\, s \; + \; \chi\, b$$

- Prior and posterior predictive:





Example Bayesian Workflow

Calibration: Testing Computational Faithfulness

- Average posterior draws with observations sampled from prior predictive should converge to prior distribution:

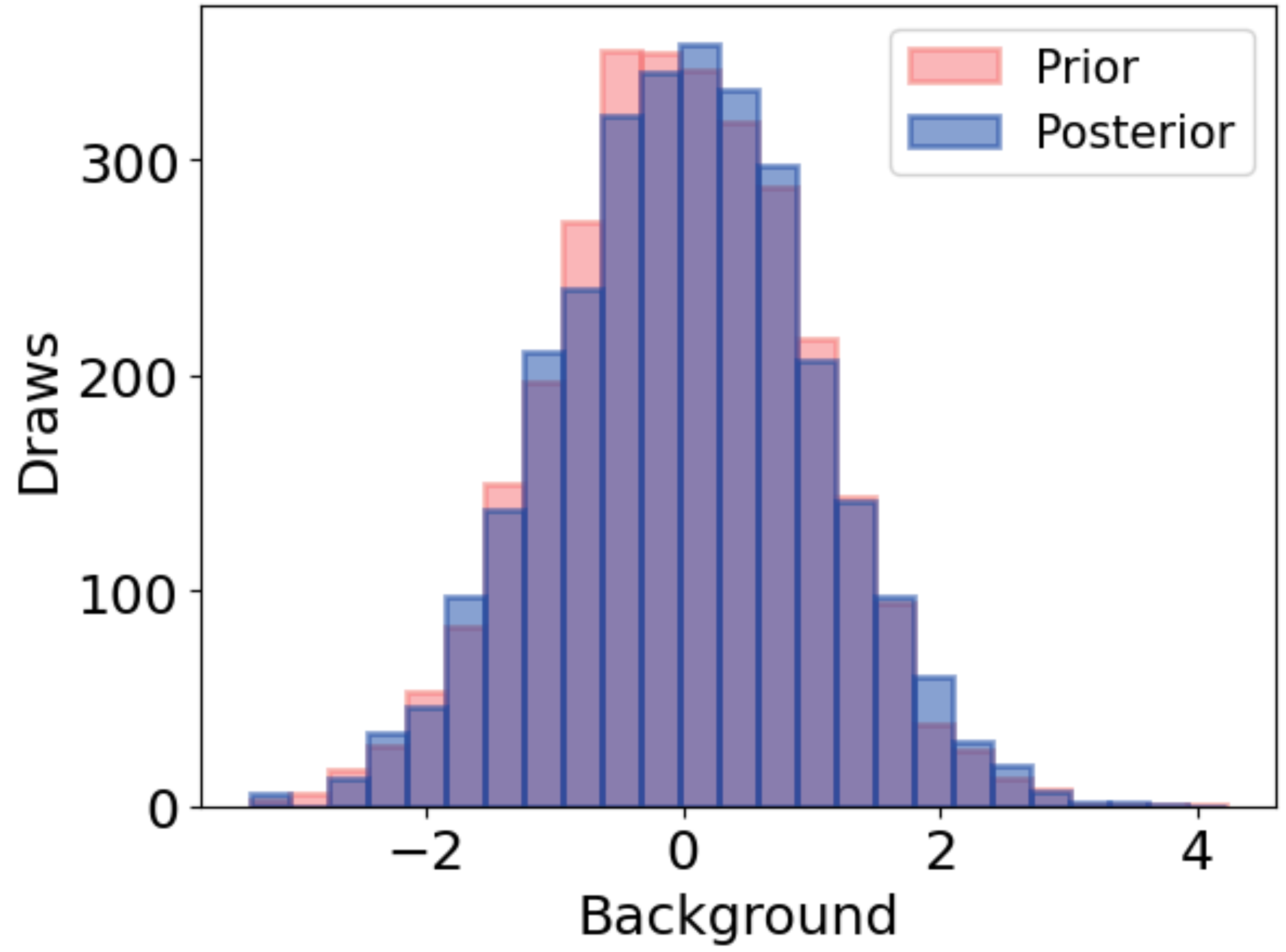
$$p(\theta) \overset{!}{\approx} \int dy d\theta' \; p(\theta \mid y) \; p(y \mid \theta')$$

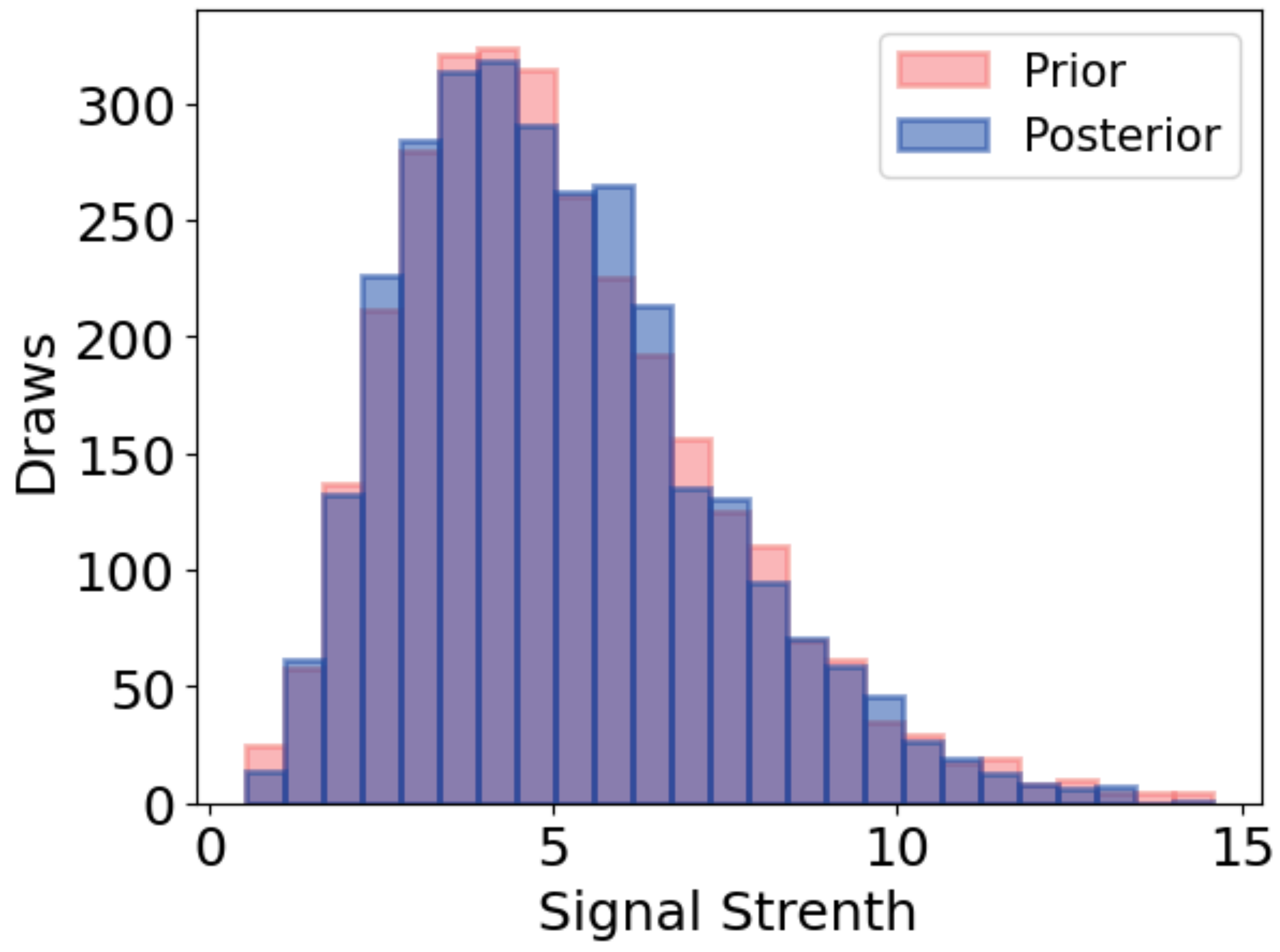
Example Bayesian Workflow

Calibration: Testing Computational Faithfulness

- Average posterior draws with observations sampled from prior predictive should converge to prior distribution:

$$p(\theta) \overset{!}{\approx} \int dy d\theta' \, p(\theta \mid y) \, p(y \mid \theta')$$





https://betanalpha.github.io/assets/case_studies/principled_bayesian_workflow.html

Conclusion and Outlook

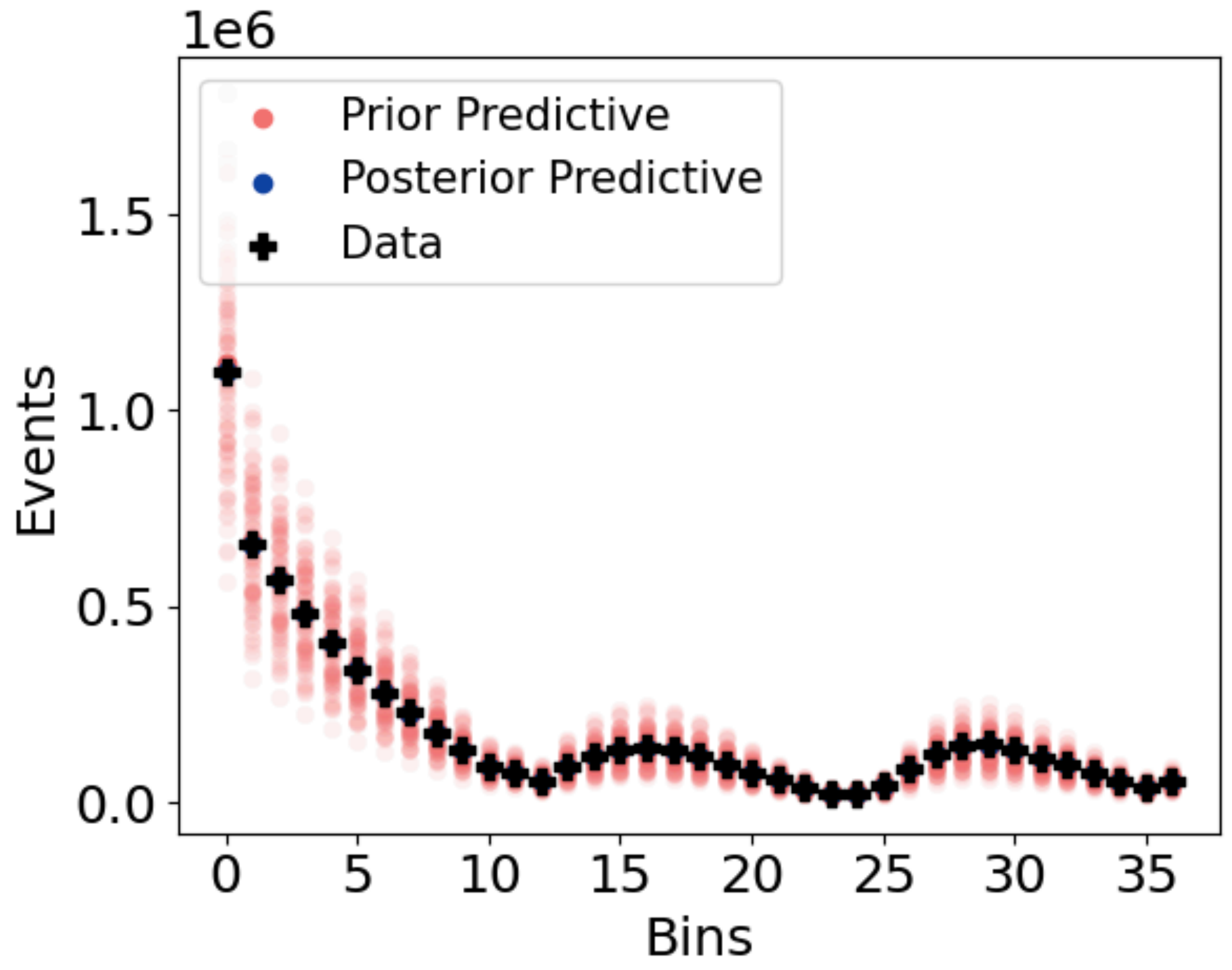
- All of this work is open-source and available under: DOI 10.5281/zenodo.7886632

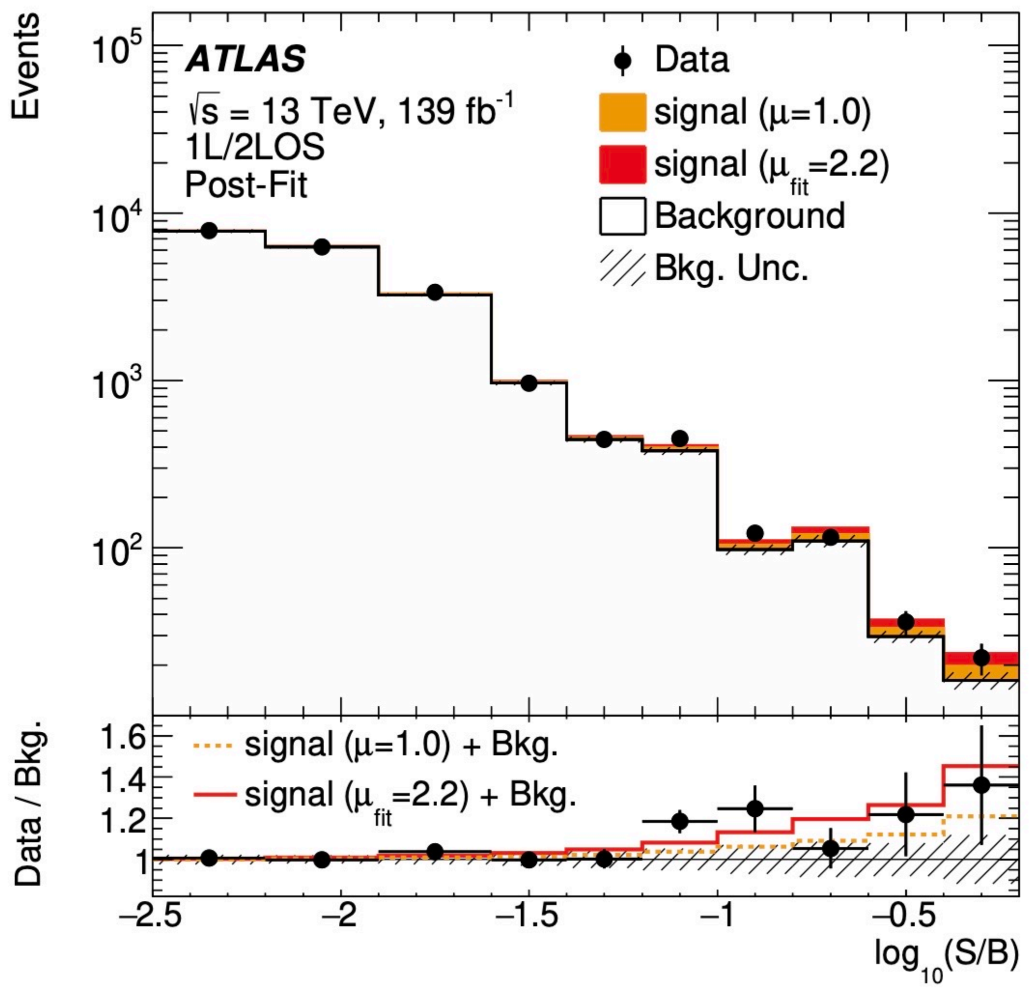
  github.com/malin-horstmann/pyhf_pymc

- Parallel Bayesian and Frequentist inference for HistFactory models is now possible
- Next steps:
  - Intergrate into `pyhf`
  - Continued tests for robustness (i.e. parallel chains)
- An ATLAS public likelihood example:

  https://doi.org/10.17182/hepdata.105039

- Want to try it? launch binder





Backup

Computation Times

	6000 (= 1000)	**12_000 (= 1000)**
Metropolis		11 s
NUTS	14 s	

Building Prior Distributions for $\vec{\eta}, \vec{\chi}$

- Turn information from auxiliary measurements (means, rates, uncertainties) into prior distributions for $\vec{\chi}$:

$$p\left(\vec{\chi}\right) \approx p\left(\vec{a}\,|\,\vec{\chi}\right)\, p_{\text{ur}}\left(\vec{\chi}\right)$$

Ur-priors

- By using rules for conjugate priors, this inference turns trivial:

$p\left(a\,|\,\chi\right)$	$\propto \text{Poiss}\left(a\,|\,\chi\right)$	$\propto \mathcal{N}\left(a\,|\,\mu = 0,\ \sigma = 1\right)$
$p\left(\chi\right)$	$\propto \Gamma\left(\chi\,|\,\alpha = a, \beta = a\right)$	$\propto \mathcal{N}\left(\chi\,|\,\mu = 0,\ \sigma = 1\right)$

Implementing external models in PyMC

- Class: `PyTensor Op`

```python
class VJPOp(Op):
  '''

  '''
  itypes = [pt.dvector,pt.dvector]
  otypes = [pt.dvector]

  def perform(self, node, inputs, outputs):
      (parameters, tangent_vector) = inputs
      results = jitted_vjp_expData(parameters, tangent_vector)

      outputs[0][0] = np.asarray(results)

  vjp_op = VJPOp()
```

```python
class ExpDataOp(Op):
    '''

    '''
    itypes = [pt.dvector]
    otypes = [pt.dvector]

    def perform(self, node, inputs, outputs):
        (parameters, ) = inputs
        results = jitted_processed_expData(parameters)

        outputs[0][0] = np.asarray(results)

    def grad(self, inputs, output_gradients):
        (parameters,) = inputs
        (tangent_vector,) = output_gradients
        return [vjp_op(parameters, tangent_vector)]


expData_op = ExpDataOp()
```

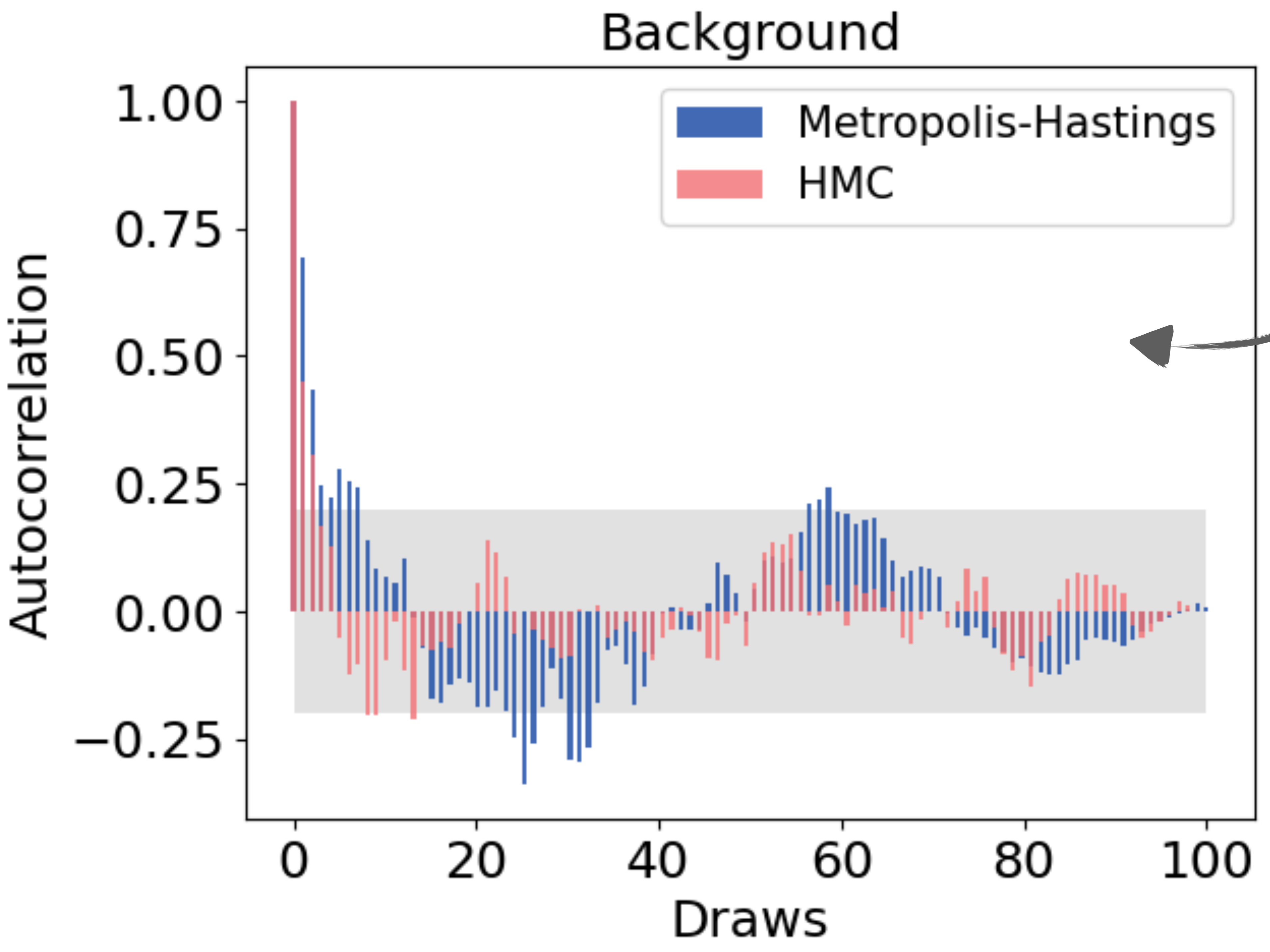
Example Bayesian Workflow

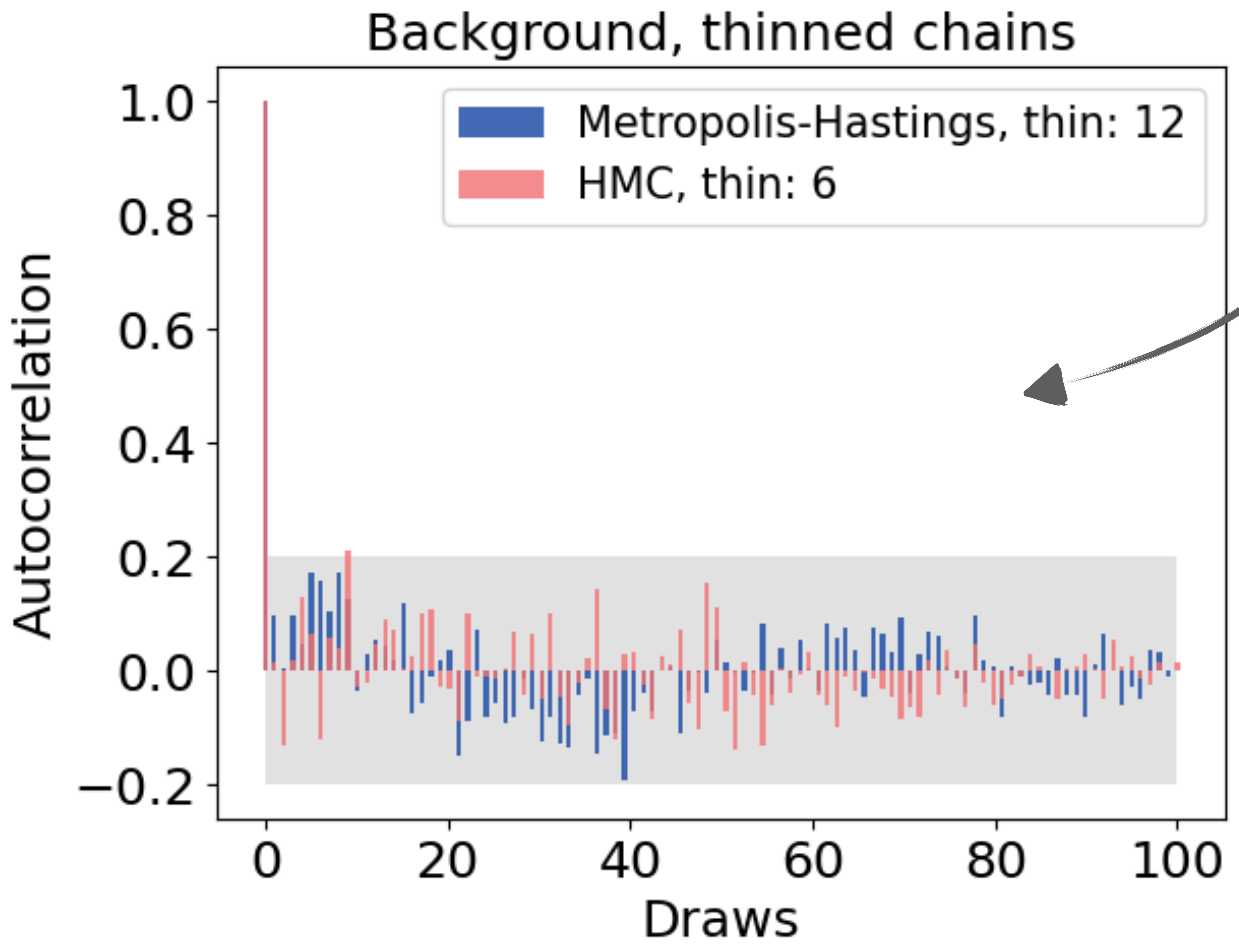
Autocorrelation: A Measure for the Inference Quality

- Gradient-based sampling (HMC) in general higher quality (but computationally heavier)
- Improve quality by thinning chains:

Use all samples

Keep only a few samples





Autocorrelation

- ACF(τ) $=$ IFFT(FFT$(X(\tau))FFT*(X(\tau))$ (Wiener-Khinchin Theorem)

https://python.arviz.org/en/stable/api/generated/arviz.autocorr.html

Ur-Priors:

- Poisson auxiliary measurement:

  $p(\chi \,|\, a) \propto \text{Poiss}(a \,|\, \chi)\; p_{\text{ur}}(\chi)$ and

  $p_{\text{ur}}(\chi) = \Gamma(\chi \,|\, \alpha, \beta)$

- Possible choices of ur-prior:
  - Mean and uncertainty from observation from auxiliary measurement
  - Uninformative