Amortized Bayesian Inference for Multilevel Models

Paul Bürkner, TU Dortmund University

Bayesian Inference

We want to estimate (approximate) the posterior

$$p(\theta \mid y) = \frac{p(\theta) \times p(y \mid \theta)}{p(y)}$$

- The marginal likelihood $p(y)$ is rarely analytically tractable
- So we use MCMC (or similar) to estimate the posterior
- For every new dataset y_{obs}, we need to run MCMC again

Simulation-Based Bayesian Inference (SBI)

What do we do if the likelihood $p(y \mid \theta)$ is not tractable either?

$\Rightarrow$ We can use simulation-based inference (SBI)

Basic algorithm for SBI (Approximate Bayesian Computation, ABC):

- Simulate $(y^{(s)}, \theta^{(s)}) \sim p(y, \theta) = p(\theta) \times p(y \mid \theta)$ from the model
- Compare $y^{(s)}$ to the observed data y_{obs}
- Keep only those parameter draws $\theta^{(s)}$ that produced data $y^{(s)}$ close enough to y_{obs}
- For every new dataset y_{obs}, we need to run ABC again

Non-Amortized Bayesian Inference

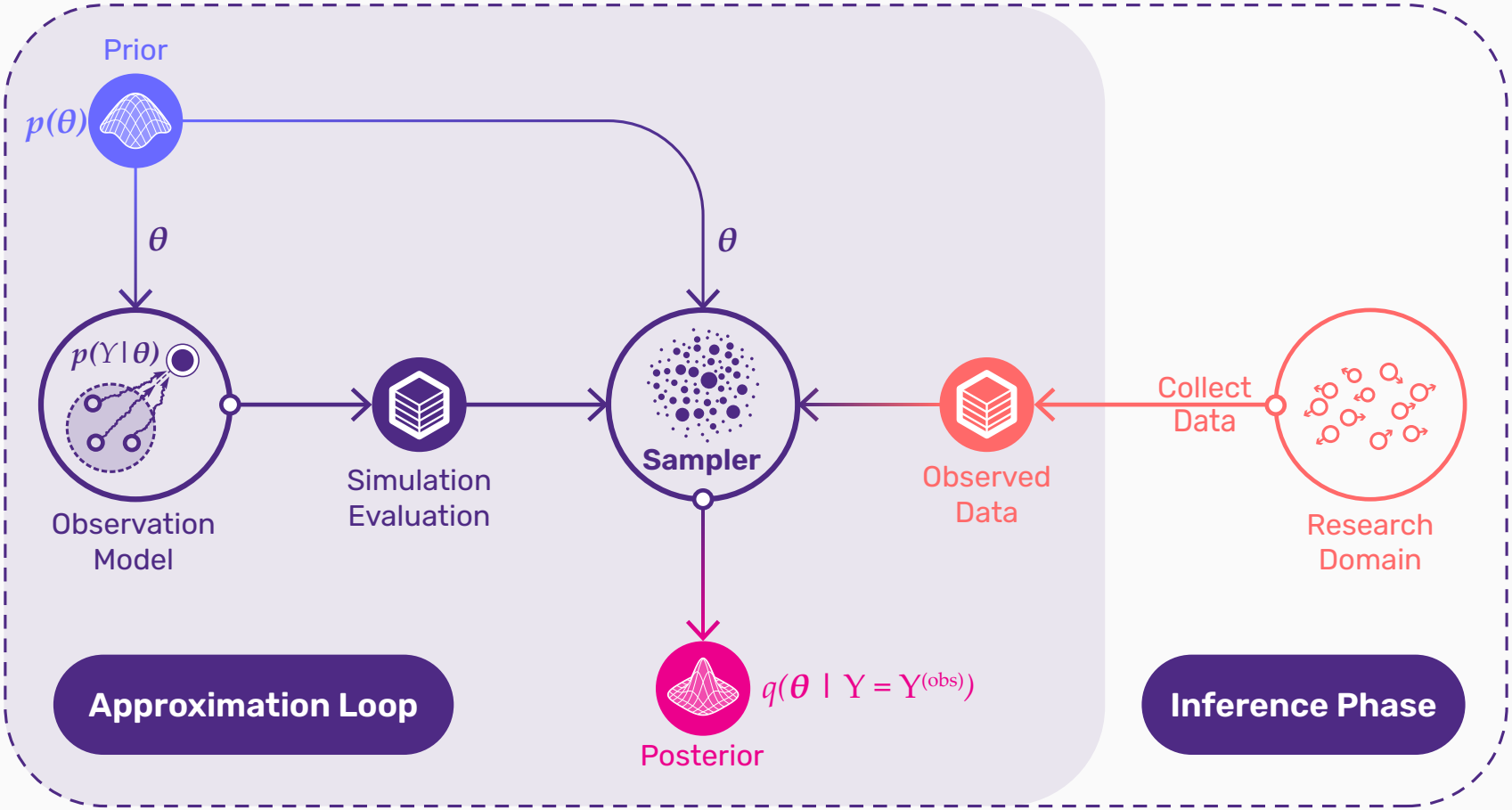


Figure 1: Designed by Jerry Huang

Amortized Bayesian Inference (ABI)

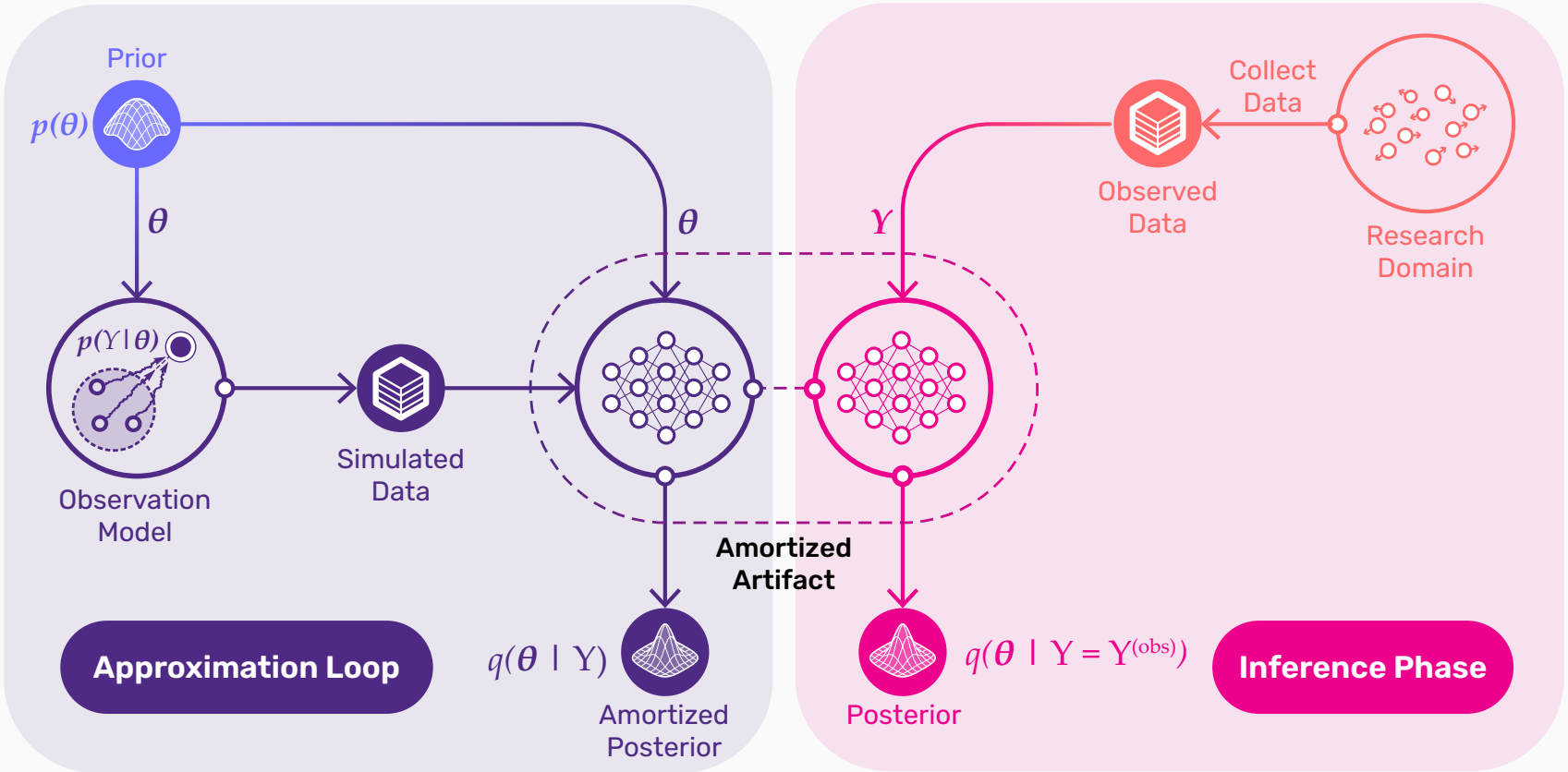
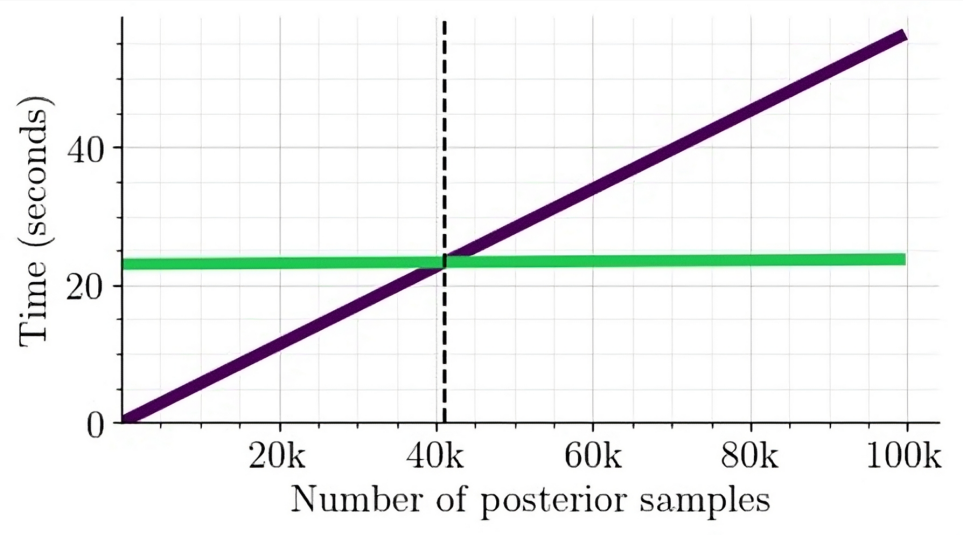


Figure 2: Designed by Jerry Huang

Breakeven points



MCMC:

- No upfront training
- 5.66 seconds for 10k posterior draws

Amortized Inference:

- 23 seconds upfront training
- 0.07 seconds for 10k posterior draws

Neural Posterior Estimation (NPE): Starting Point

Define a generative neural network $q_\phi(\theta \mid y)$ that learns to approximate the analytic posterior $p(\theta \mid y)$

For given dataset y_{obs} minimize the KL divergence between $p(\theta \mid y_{\text{obs}})$ and $q_\phi(\theta \mid y_{\text{obs}})$ with respect to the neural network parameters ϕ:

$$
\begin{aligned}
\text{minimize}_\phi \quad & \mathbb{KL}\left[p(\theta \mid y_{\text{obs}}) \mid\mid q_\phi(\theta \mid y_{\text{obs}})\right] \\
& = \mathbb{E}_{\theta \sim p(\theta \mid y_{\text{obs}})}\left[\log \frac{p(\theta \mid y_{\text{obs}})}{q_\phi(\theta \mid y_{\text{obs}})}\right] \\
& \propto -\mathbb{E}_{\theta \sim p(\theta \mid y_{\text{obs}})}\left[\log q_\phi(\theta \mid y_{\text{obs}})\right] \\
& \approx -\frac{1}{S}\sum_{s=1}^{S} \log q_\phi(\theta^{(s)} \mid y_{\text{obs}}) \quad \text{with} \quad \theta^{(s)} \sim p(\theta \mid y_{\text{obs}})
\end{aligned}
$$

How to we minimize this KL divergence without having to sample from the analytic posterior $p(\theta \mid y_{\text{obs}})$?

Amortized Bayesian Inference via NPE

Don't just learn to estimate a posterior $q_\phi(\theta \mid y_{\text{obs}})$ for a single dataset y_{obs}, but learn a *posterior functional* $q_\phi(\theta \mid y)$ for *arbitrary data* y:

$$
\begin{aligned}
\text{minimize}_\phi \quad & \mathbb{E}_{y \sim p(y)} \left[\mathbb{KL} \left[p(\theta \mid y) \mid\mid q_\phi(\theta \mid y) \right] \right] \\
& = \mathbb{E}_{y \sim p(y)} \left[\mathbb{E}_{\theta \sim p(\theta \mid y)} \left[\log \frac{p(\theta \mid y)}{q_\phi(\theta \mid y)} \right] \right] \\
& \propto -\mathbb{E}_{(y,\theta) \sim p(y,\theta)} \left[\log q_\phi(\theta \mid y) \right] \\
& \approx -\frac{1}{S} \sum_{s=1}^{S} \log q_\phi(\theta^{(s)} \mid y^{(s)}) \quad \text{with} \quad (y^{(s)}, \theta^{(s)}) \sim p(y, \theta)
\end{aligned}
$$

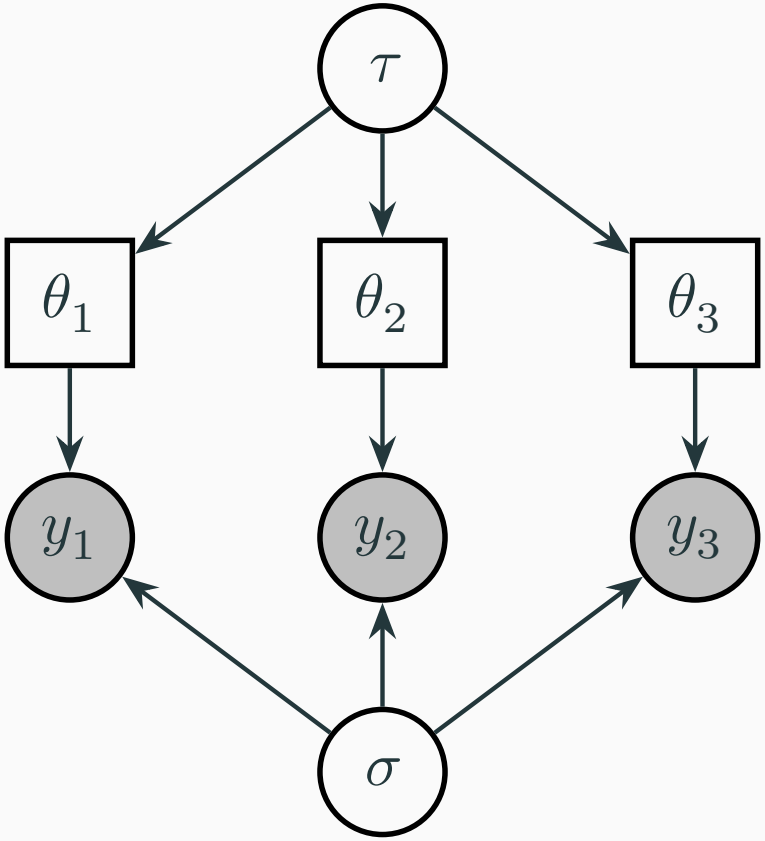
Training the Networks

1. Define our generative statistical model $p(x, \theta)$ of interest
   - typically $p(x, \theta) = p(\theta) \times p(y \mid \theta)$
2. (Optionally) define a summary network $h_\psi(y)$ to learn summary statistics from data
3. Define an inference network $q_\phi(\theta \mid h_\psi(y))$ to learn the posterior given the summary statistics
4. Train the networks
   4.1 Sample from $(y^{(s)}, \theta^{(s)}) \sim p(y, \theta)$
   4.2 Optimize weights ϕ and ψ so that $p(\theta \mid y) \approx q_\phi(\theta \mid h_\psi(y))$

How can we apply ABI to multilevel models?

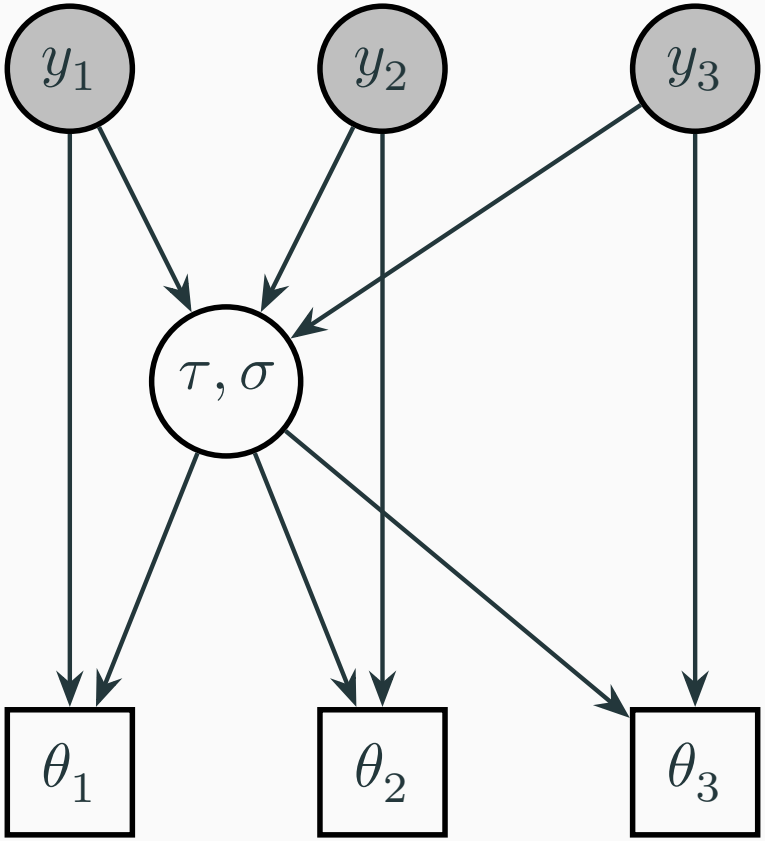
Main issue: The number of parameters varies with the size of the dataset

Factorization for Simple Multilevel Models



Factorization of the joint distribution:

$$p(y, \theta, \tau, \sigma) = p(\tau, \sigma) \times \prod_{j=1}^{J} p(\theta_j \mid \tau) \times p(y_j \mid \theta_j, \sigma)$$



Factorization of the posterior distribution:

$$p(\tau, \sigma, \theta \mid y) = p(\tau, \sigma \mid y) \times \prod_{j=1}^{J} p(\theta_j \mid \tau, \sigma, y_j)$$

Neural Archictecture for Two-Level Models

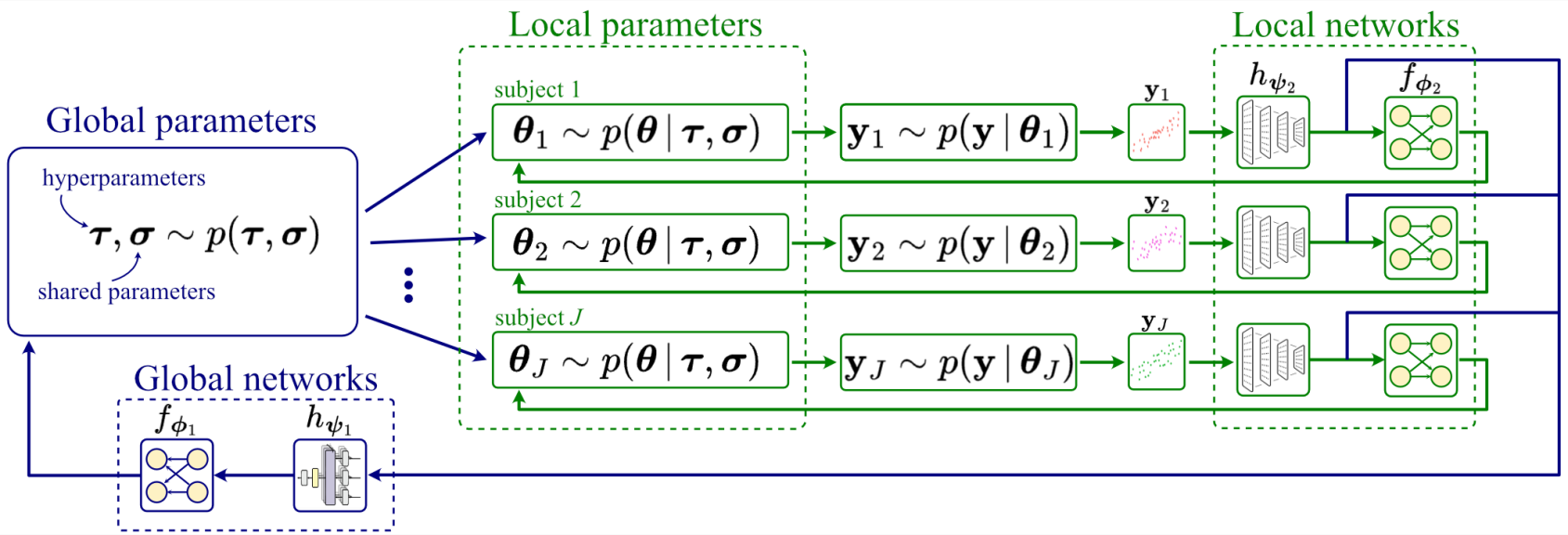


Figure 3: Source: Habermann et al. (2025). Amortized Bayesian Multilevel Models. *Bayesian Analysis*. https://arxiv.org/abs/2408.13230

Example: European Air Passenger Traffic

Likelihood of the difference in air passenger traffic for country j between time $t+1$ and t:

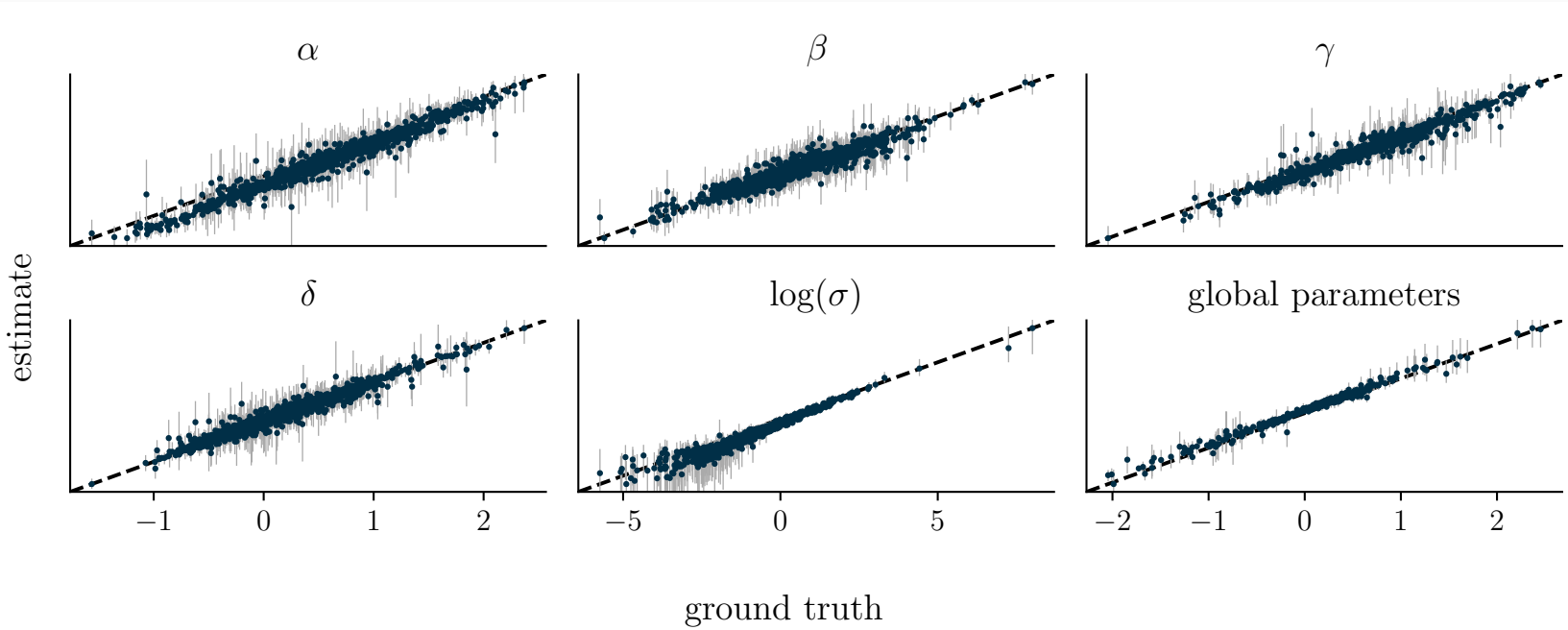
$$y_{j,t+1} \sim \text{Normal}(\alpha_j + y_{j,t}\beta_j + u_{j,t}\gamma_j + w_{j,t}\delta_j, \sigma_j)$$

Local priors:

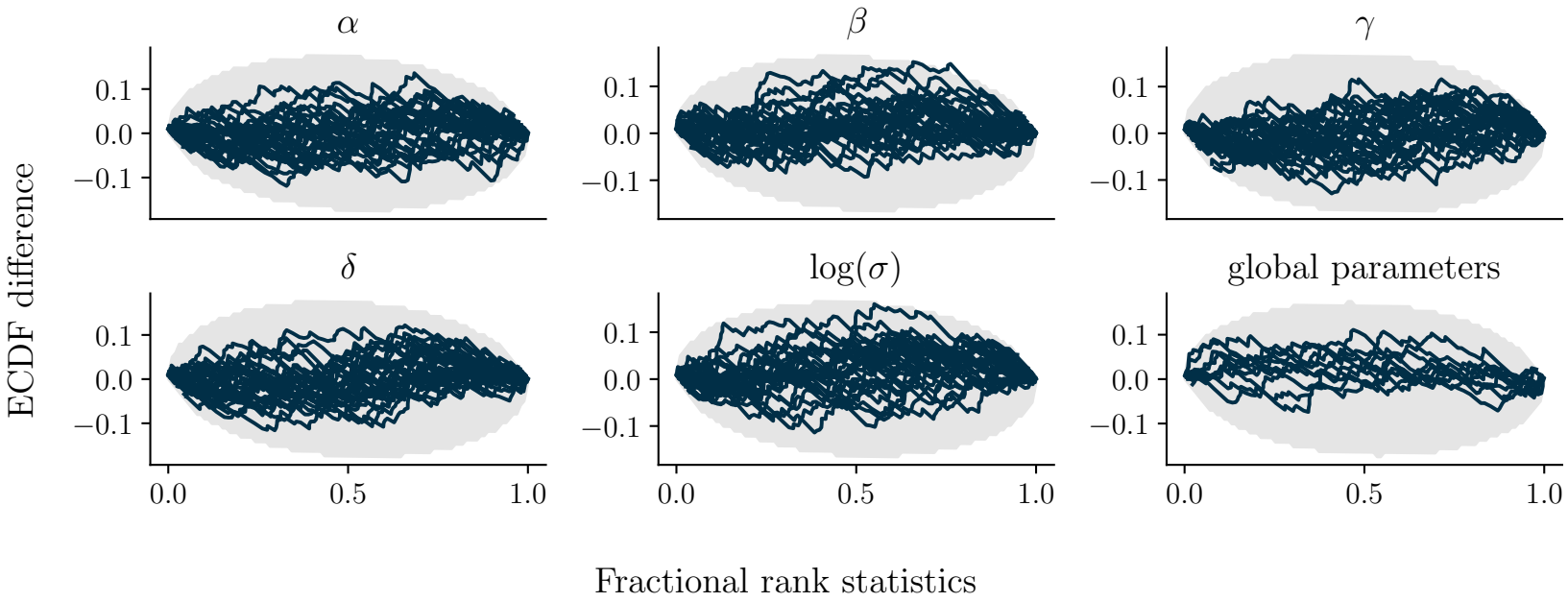
$$\begin{aligned}
\alpha_j &\sim \text{Normal}(\mu_\alpha, \sigma_\alpha) \\
\beta_j &\sim \text{Normal}(\mu_\beta, \sigma_\beta) \\
\gamma_j &\sim \text{Normal}(\mu_\gamma, \sigma_\gamma) \\
\delta_j &\sim \text{Normal}(\mu_\delta, \sigma_\delta) \\
\log(\sigma_j) &\sim \text{Normal}(\mu_\sigma, \sigma_\sigma)
\end{aligned}$$

Global priors not shown for simplicity

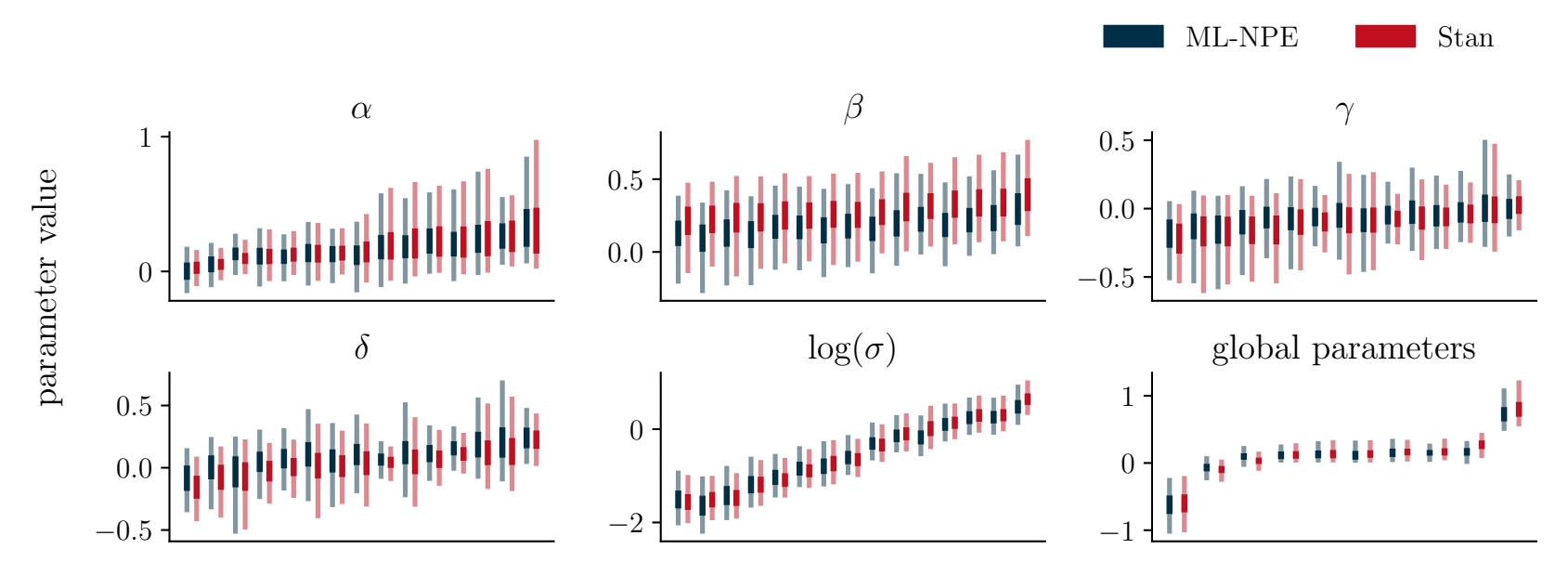
Results: Recovery



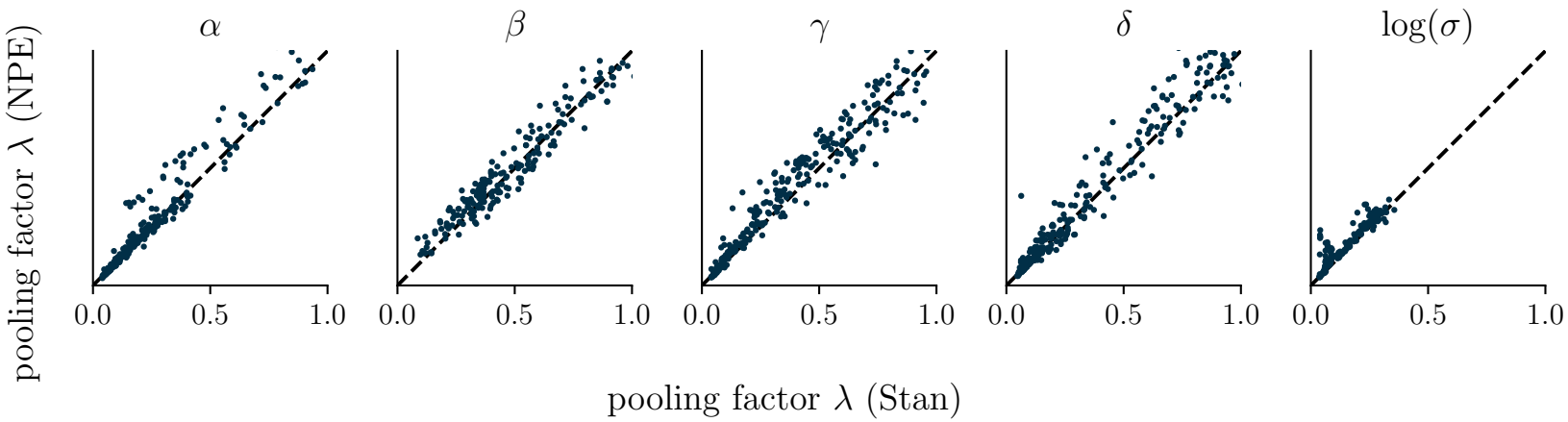
Results: Calibration



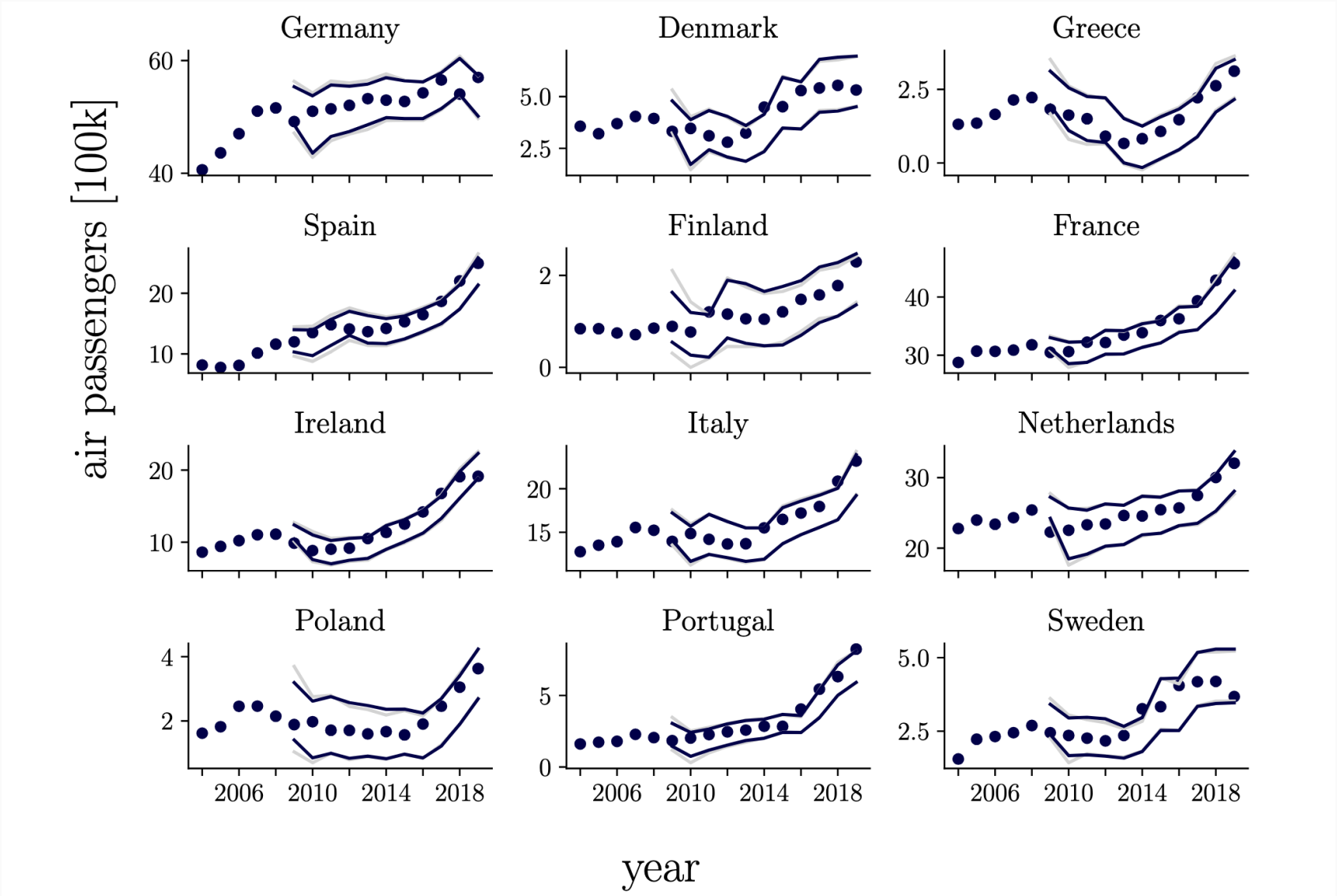
Results: Comparison with Stan (Parameter Estimates)



Results: Comparison with Stan (Shrinkage)



Results: Posterior Predictions (1-Step Ahead)



Software: BayesFlow 2.0

- Python library built on `keras3`
  - `TensorFlow`, `JAX`, or `PyTorch` as a backend
- Implementation of common neural architectures to make ABI easier
- Helper functions for simulation, configuration, training, validation, diagnostics, ...
- Website: bayesflow.org
- Github: github.com/bayesflow-org/bayesflow
- Forums: discuss.bayesflow.org
- Install release version: `pip install bayesflow`
- Install dev version: `pip install git+https://github.com/bayesflow-org/bayesflow@dev`

BayesFlow 2.0: Workflow

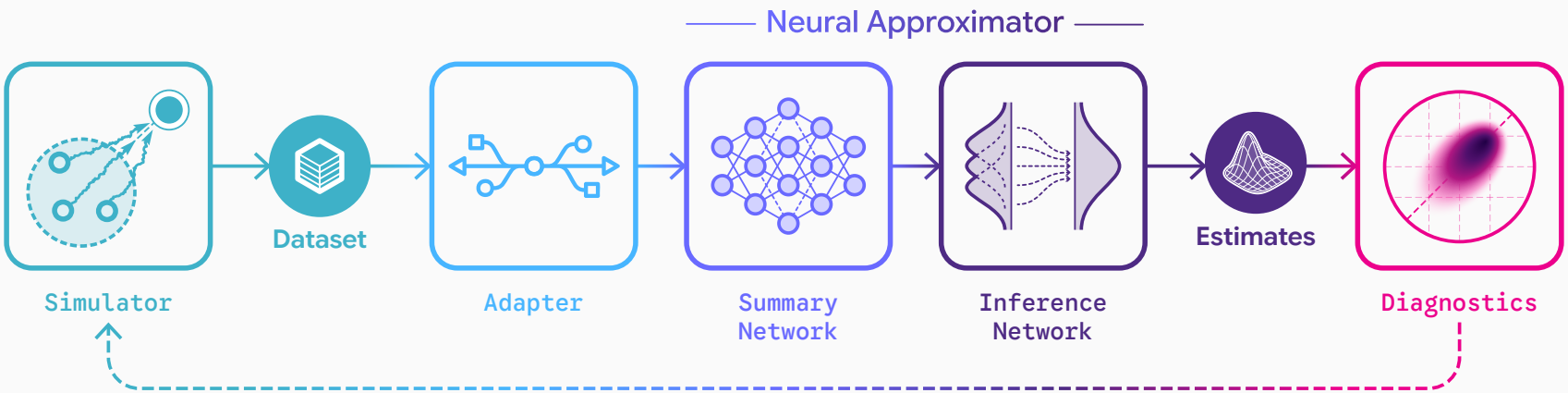


Figure 4: Kühmichel et al. (2026). BayesFlow 2.0: Multi-Backend Amortized Bayesian Inference in Python. *ArXiv preprint*. https://arxiv.org/abs/2602.07098

Summary

- Amortized Bayesian Inference (ABI) is a powerful method for fast posterior estimation across datasets
- ABI on multilevel models is non-trivial but possible via inverse factorizations of the posterior

Many challenges remain:

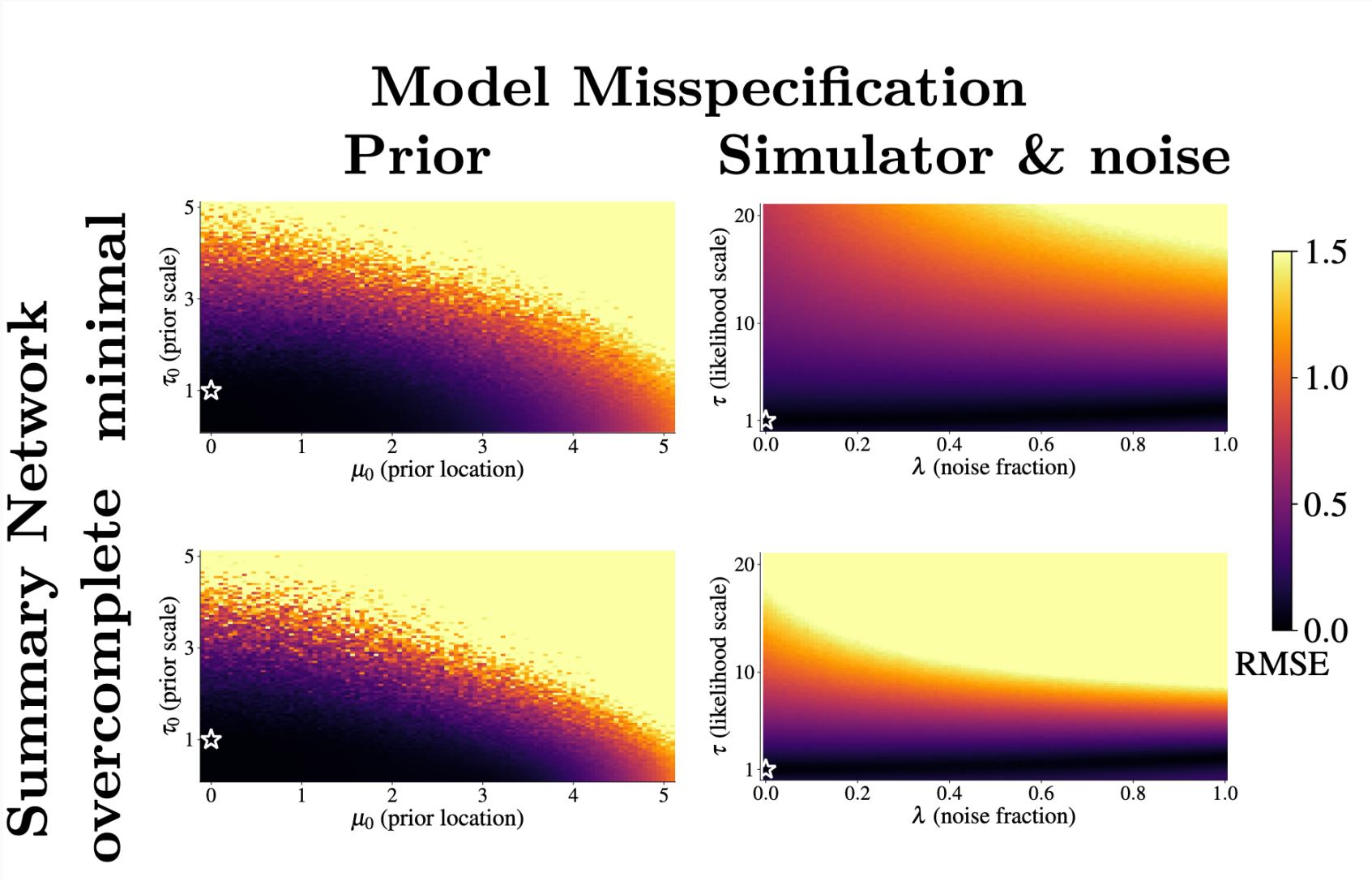
- Robustness of ABI to model misspecification
- Architectures for more complex graphical models
- Better out-of-the-box applicability

Collaboration Opportunities

- Applications of ABI
- Software: BayesFlow
- ABI robustness methods
- Improved architectures for ABI
- Mathematical theory of ABI losses

Appendix

Standard NPE on misspecified models fails



Source: https://arxiv.org/abs/2406.03154

Bayesian Self-consistency

For any set of parameter values $\theta^{(1)}, \dots, \theta^{(L)}$, the following holds:

$$p(y) = \frac{p(y \mid \theta^{(1)})\, p(\theta^{(1)})}{p(\theta^{(1)} \mid y)} = \dots = \frac{p(y \mid \theta^{(L)})\, p(\theta^{(L)})}{p(\theta^{(L)} \mid y)}.$$

This implies that the variance of the log-ratios must be zero:

$$\text{Var}_{l=1}^{L} \left[\log \left(\frac{p(y \mid \theta^{(l)})\, p(\theta^{(l)})}{p(\theta^{(l)} \mid y)} \right) \right] = 0$$

Our initial paper on Bayesian self-consistency: https://arxiv.org/abs/2310.04395

Bayesian self-consistency loss

Replace the true posterior $p(\theta \mid y)$ with the neural approximate posterior $q(\theta \mid y)$.

For any (**unlabled**) dataset y^* and any parameter generating distribution $\tilde{p}(\theta)$, we define:

$$\text{SCLoss(q)} = \text{Var}_{\theta \sim \tilde{p}(\theta)} \left[\log p(y^* \mid \theta) + \log p(\theta) - \log q(\theta \mid y^*) \right]$$

$\Rightarrow$ We can use **real-world data** as y^* to train our SC loss!

The SC-Loss alone doesn't work well most of the time so we combine it with the standard NPE loss:

$$\text{SemiSupervisedLoss(q)} = \text{NPELoss(q)} + \lambda \cdot \text{SCLoss(q)}.$$

Bayesian Self-Consistency losses a strictly proper

Let C be a score that is globally minimized if and only if its functional argument is constant across the support of the posterior $p(\theta \mid y)$ almost everywhere. Then, C applied to the Bayesian self-consistency ratio with known likelihood

$$C\left(\frac{p(y \mid \theta)\, p(\theta)}{q(\theta \mid y)}\right)$$

is a strictly proper loss: It is globally minimized if and only if $q(\theta \mid y) = p(\theta \mid y)$ almost everywhere.

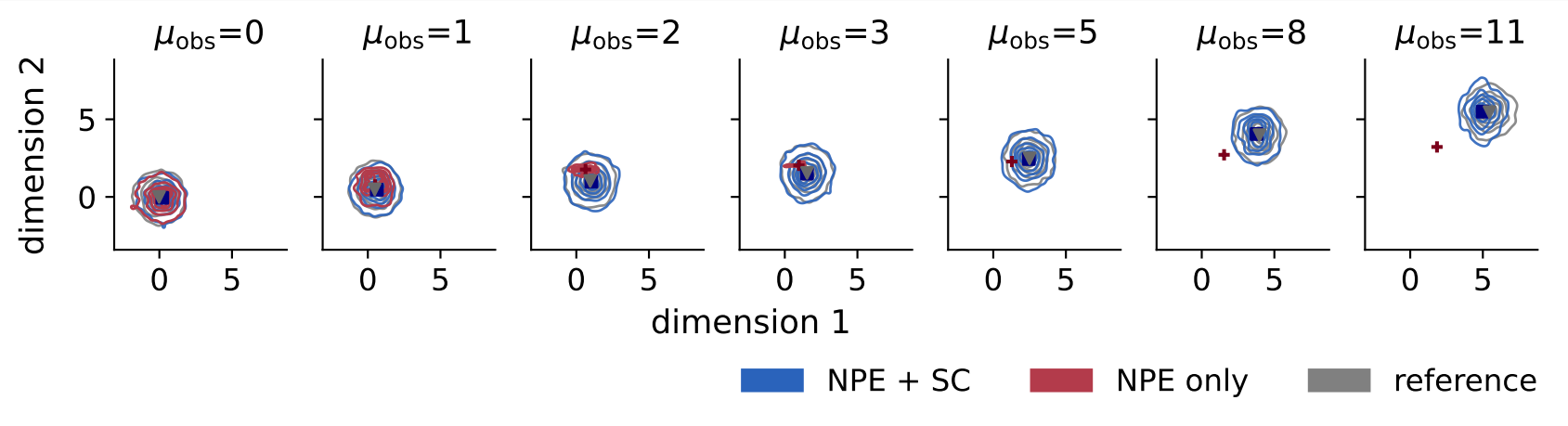
This implies that also the semi-supervised loss is strictly proper.

Case Study 1: Multivariate normal model

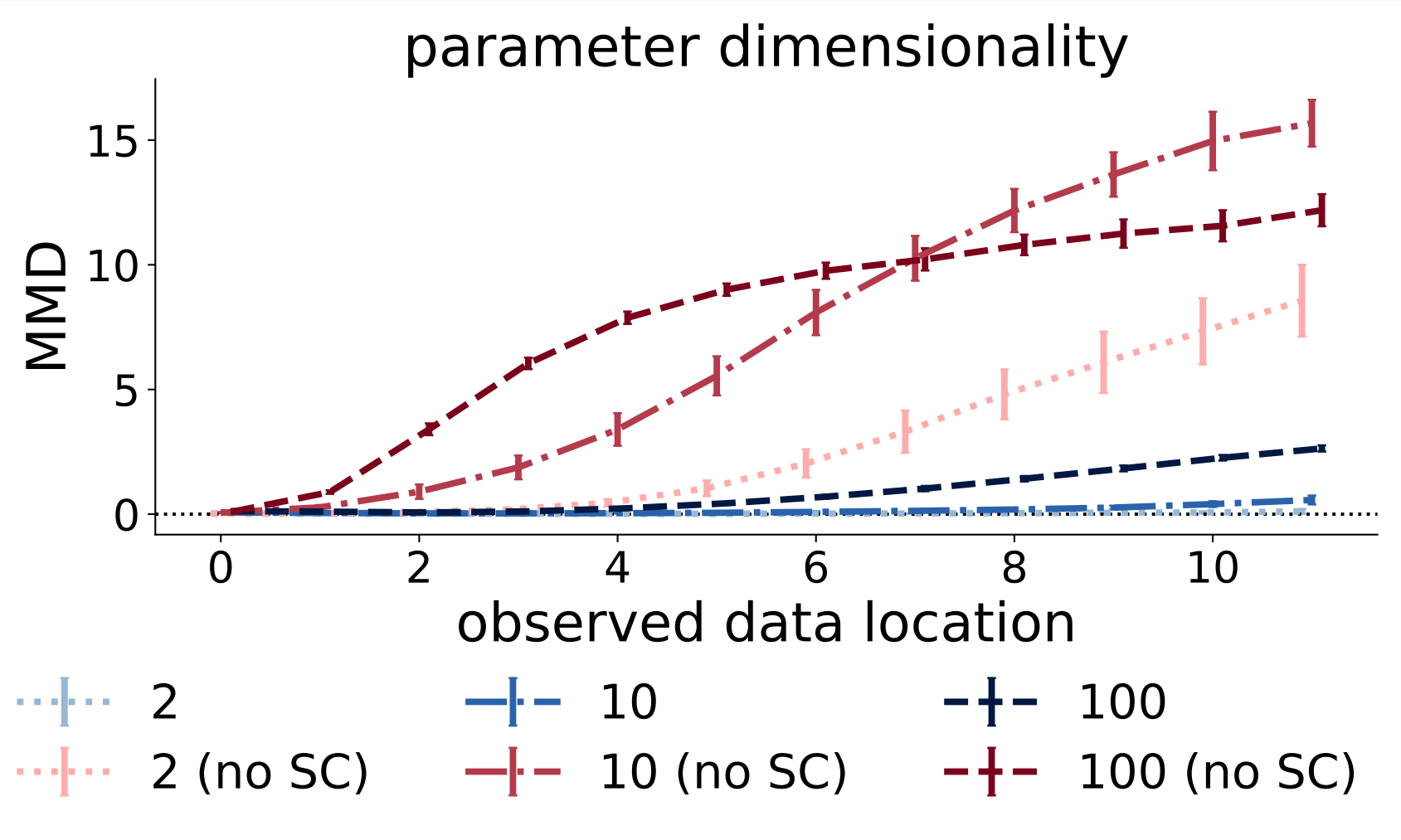
$$\theta \sim \text{Normal}(\mu_{\text{prior}}, I_D), \qquad y \sim \text{Normal}(\theta, I_D)$$

- For the NPE loss, we simulate from the model with $\mu_{\text{prior}} = 0$
- For the SC loss, we simulate **few unalabled datasets** from the model with $\mu_{\text{prior}} = 2$

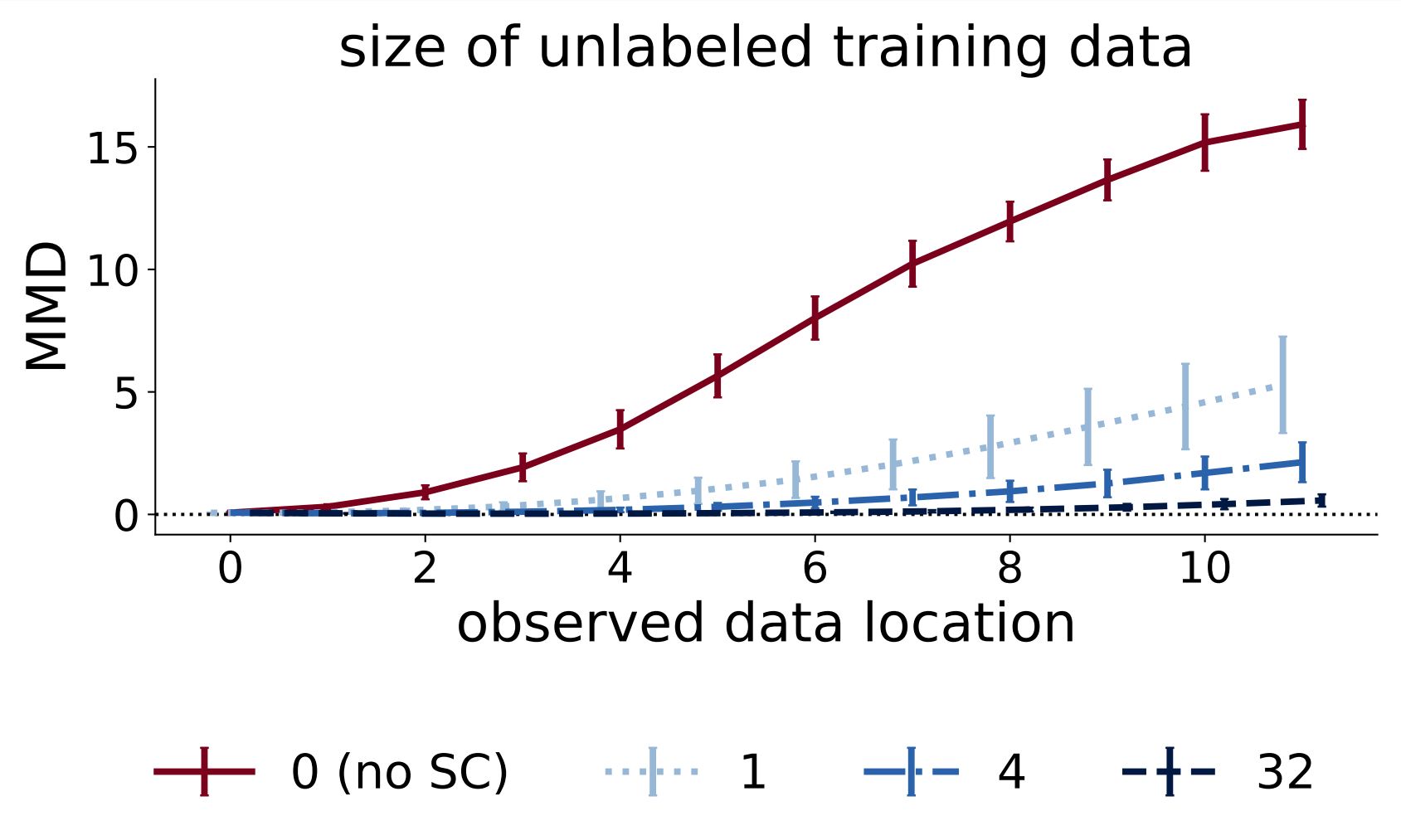
Illustrative results:



Case Study 1: More Results



Case Study 1: More Results



Case Study 2: Time Series of Air Traffic data

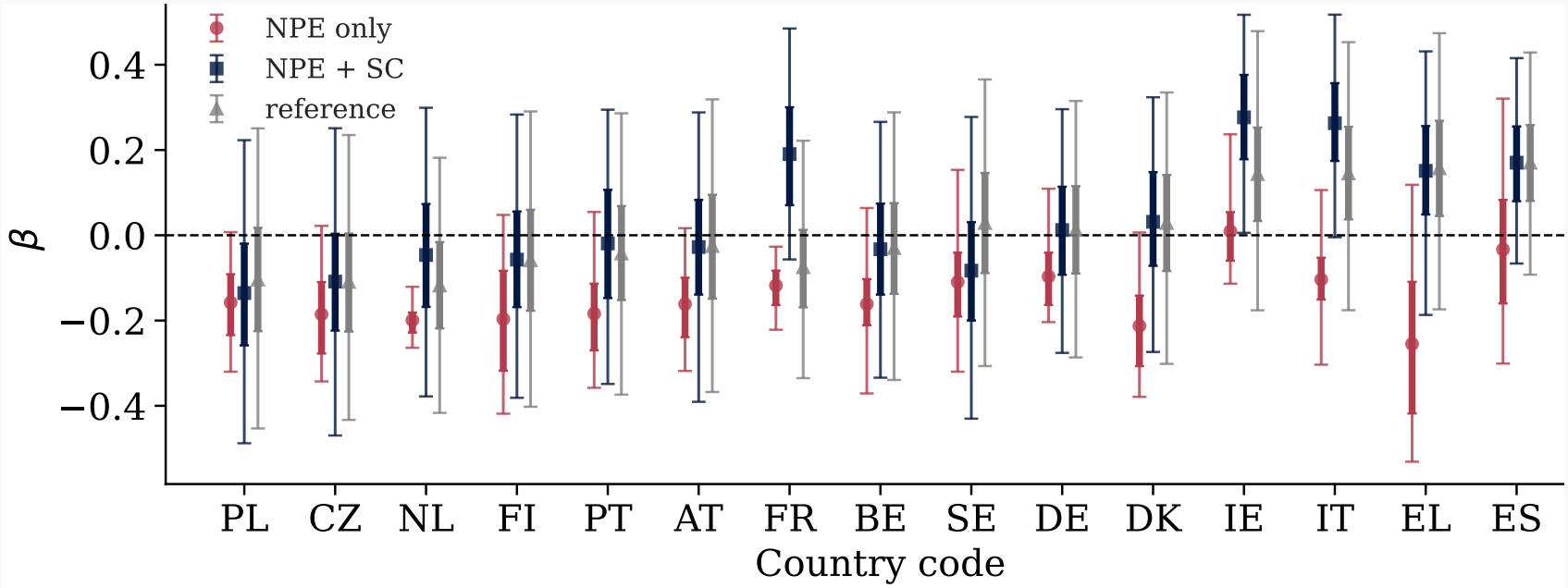
Predicting the change in air traffic for different European countries

$$y_{j,t+1} \sim \text{Normal}(\alpha_j + \beta_j y_{j,t} + \dots, \sigma_j)$$

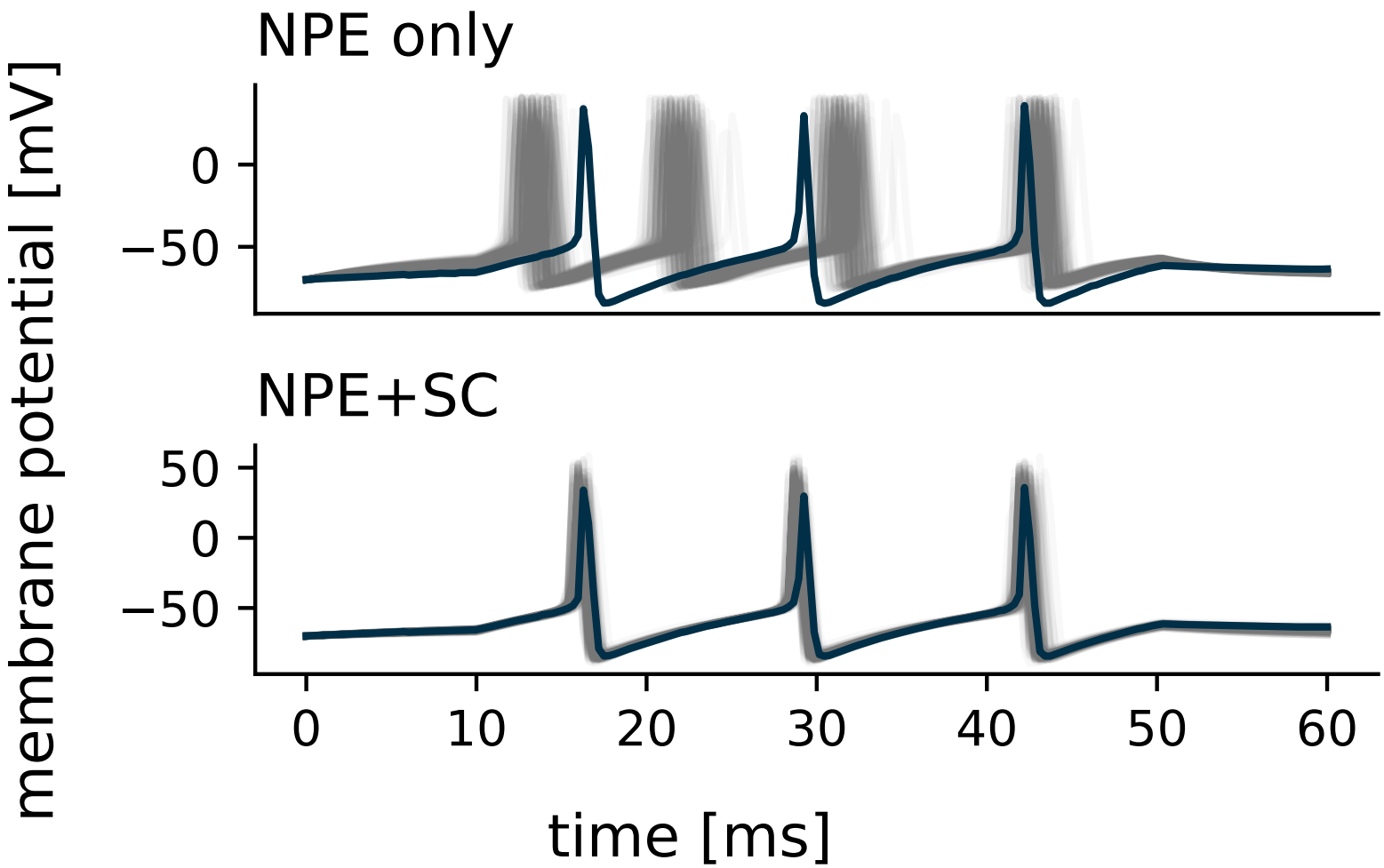
- $y_{j,t}$ number of passengers for country j at year t
- α_j intercept parameter
- β_j auto-correlation parameter
- σ_j residual standard deviation

We have data of 15 countries, 4 of which are used as training data in our SC loss.

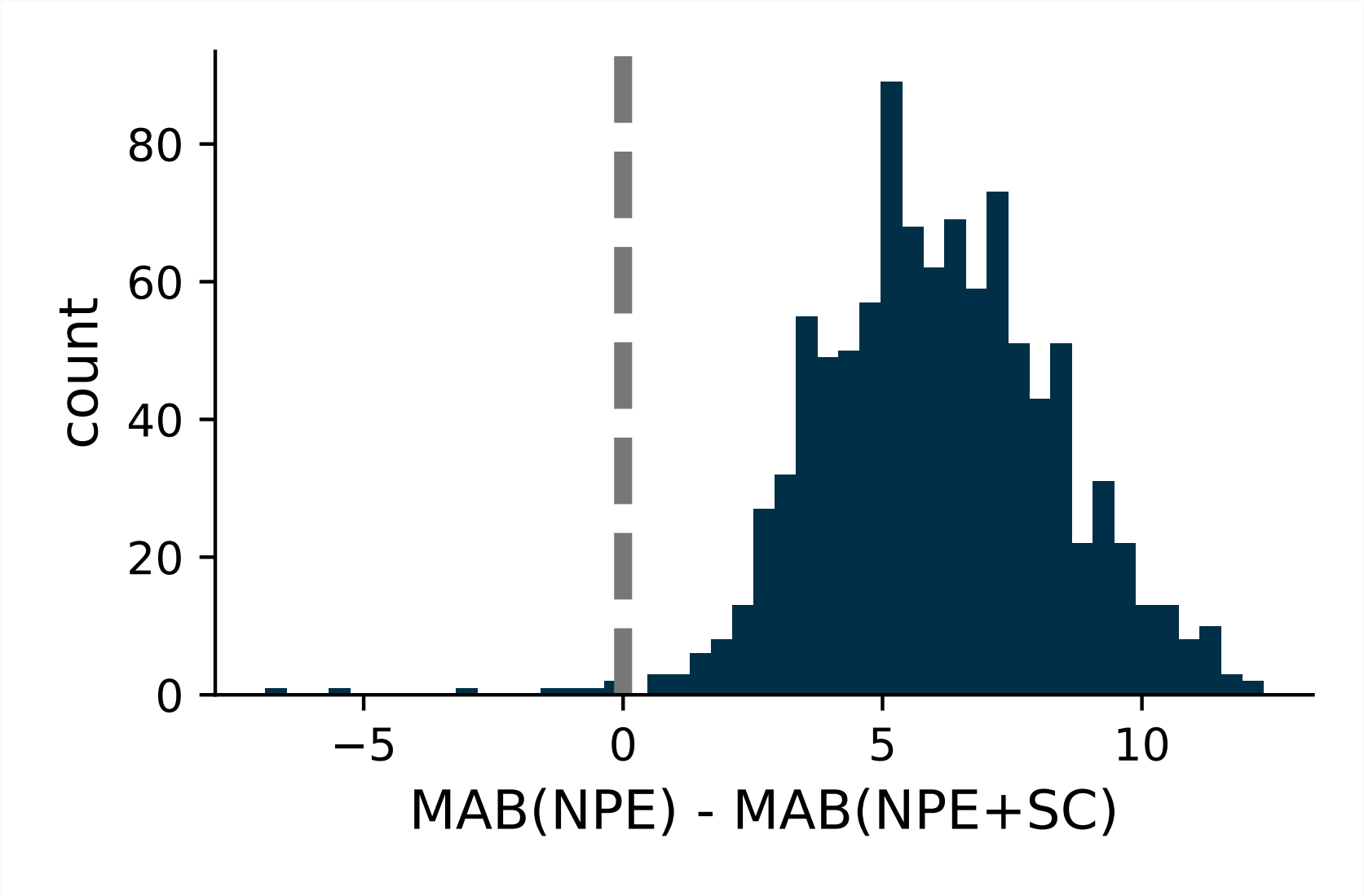
Case Study 2: Results



Case Study 3: Hodgkin-Huxley model of neuron activation



Case Study 3: More results



Summary: Self-Consistency

- The SC loss can strongly improve robustness to model misspecification
- The SC loss requires no data labels so we may even use **real-world data** for training
- The SC loss is strictly proper so it has the same target (the true posterior) as the NPE loss
- Challenge 1: The SC loss requires a known or estimated likelihood density: stronger robustness in the known case
- Challenge 2: We need neural approximators that have fast density evaluation

Our Current Research on SC losses

- Model comparison: SC works great when done correctly (https://arxiv.org/abs/2508.20614)
- Continual learning: SC losses may lead to catastrophic forgetting if applied alone but we can mitigate that
- Unknown likelihood densities: We have some promising ideas how to adjust SC losses and training

What is the target of inference?

- **Target 1**: The analytic posterior $p(\theta \mid y_{\text{obs}}) \propto p(y_{\text{obs}} \mid \theta)\, p(\theta)$ of the assumed probabilistic model given the observed data y_{obs}.
- **Target 2**: A posterior $p(\theta \mid \tilde{y}_{\text{obs}}) \propto p(\tilde{y}_{\text{obs}} \mid \theta)\, p(\theta)$ of the assumed probabilistic model given *adjusted data* $\tilde{y}_{\text{obs}}$.
    - Equivalent: A posterior given an *adjusted likelihood*
- **Target 3**: A posterior $\tilde{p}(\theta \mid y_{\text{obs}}) \propto p(y_{\text{obs}} \mid \theta)\, \tilde{p}(\theta)$ from an *adjusted prior* $\tilde{p}(\theta)$ given the observed data y_{obs}.

Source: Elsemüller et al. (2025). *Transactions in Machine Learning Research*. https://arxiv.org/abs/2502.04949

Ways to achieve robust/trustworthy inference

- Unsupervised likelihood-based losses
  - Example: SC losses
  - Aims at Target 1
  - Drawback: Requires the likelihood density
- Unsupervised domain adaptation
  - Example: Minimize the distance of simulated and real-world data in the summary space
  - Aims at Target 2
  - Drawback: Adjusts the target posterior implicitly
- Supervised real-world data calibration
  - Calibrate posterior on **labeled** real-world data
  - Aims at Target 2 (I think)
  - Drawback: Requires real-world data with labels