UNIVERSITÄT BONN

HELMHOLTZ MUNICH

Amortized inference for pharmacological mixed-effects models with incomplete data

JOACHIM HERZ STIFTUNG

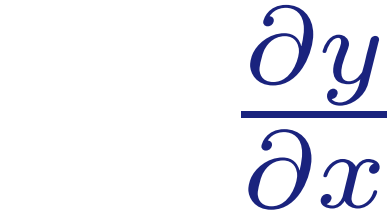
Yannik Schälte

JHS Add-On 8th Cohort First Meeting, Hamburg 2023-02-11

@yannik_schaelte

yannikschaelte

yannikschaelte.github.io/pres_npe_hjs_2023



mechanistic modeling

$$\frac{\partial y}{\partial x}$$

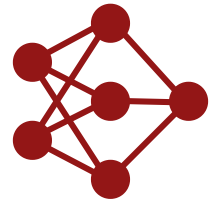
mechanistic modeling

- theory-driven
- interpretation
- testability

$$\frac{\partial y}{\partial x}$$

mechanistic modeling

- theory-driven
- interpretation
- testability

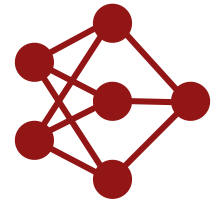


machine learning

$$\frac{\partial y}{\partial x}$$

mechanistic modeling

- theory-driven
- interpretation
- testability



machine learning

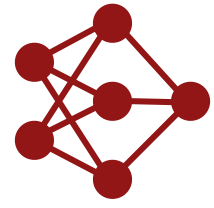
- data-driven
- large data
- automation

$$\frac{\partial y}{\partial x}$$

mechanistic modeling

- theory-driven
- interpretation
- testability

+



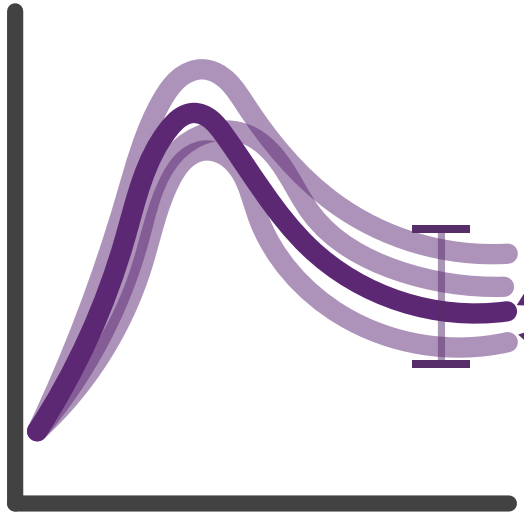
machine learning

- data-driven
- large data
- automation

sciML

model-based data-efficient ML

MIXED-EFFECTS MODELING



fixed effects α

random effects β

MIXED-EFFECTS MODELING

dynamical model: $\dot{x} = f(x, \theta)$

observables: $y = h(x, \theta) + \varepsilon$

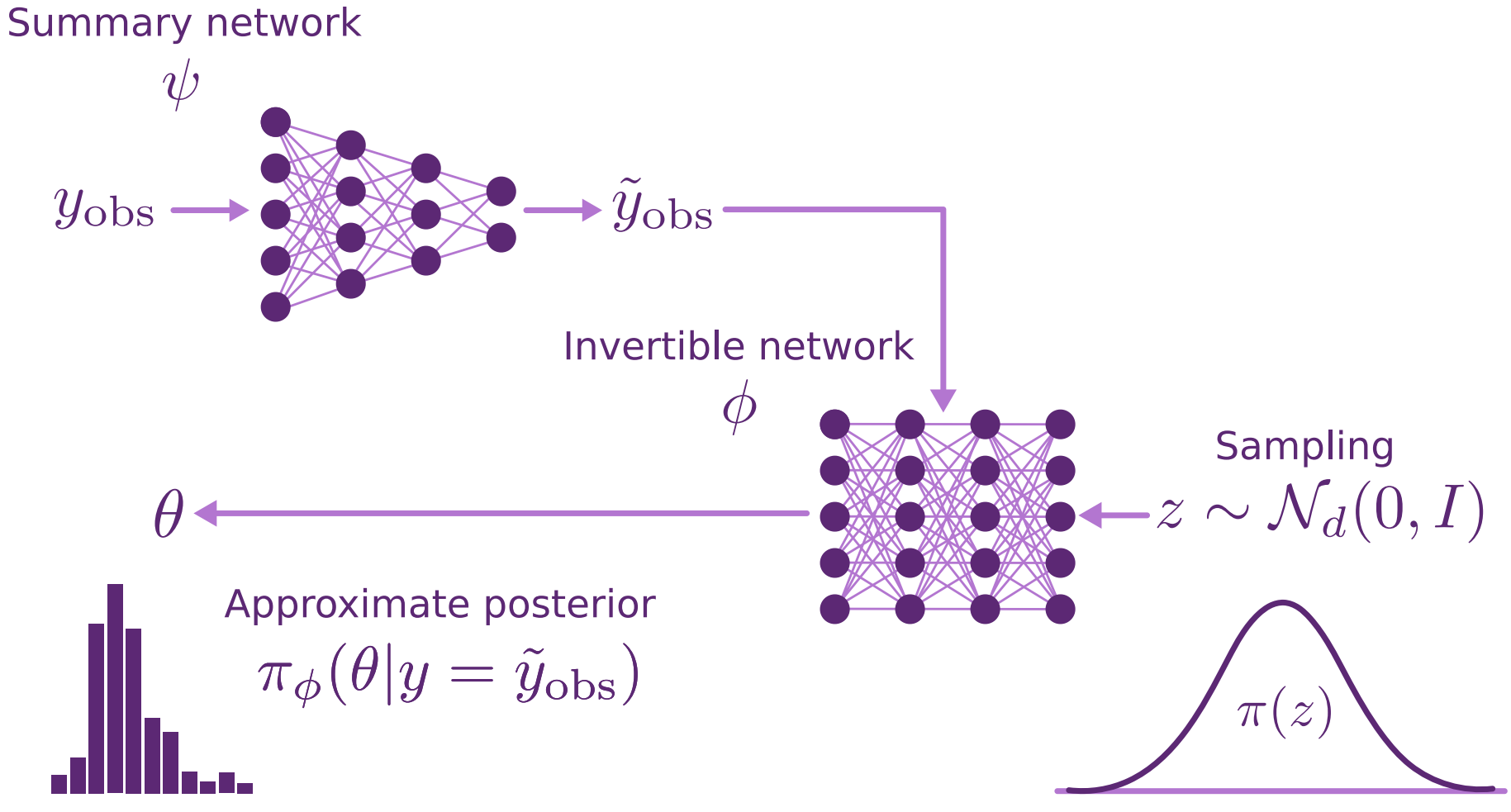
parameters: $\theta = A\alpha + B\beta, \quad \beta \sim \mathcal{N}(0, \Sigma)$

THE PROBLEM

THE PROBLEM

AMORTIZED INFERENCE VIA INVERTIBLE NEURAL NETWORKS

you have to solve many similar problems? amortize the solution!



similar to: Radev et al., 2021

- parameter estimation requires repeatedly evaluating an integral

$$\pi(y_{\text{obs}}|\alpha, \Sigma) = \prod_i \int \pi(y_i|\theta)\pi(\theta|\alpha, \Sigma)d\theta$$

- parameter estimation requires repeatedly evaluating an integral

$$\pi(y_{\text{obs}}|\alpha,\Sigma) = \prod_i \int \pi(y_i|\theta)\pi(\theta|\alpha,\Sigma)d\theta$$

OUR METHOD

- parameter estimation requires repeatedly evaluating an integral

$$\pi(y_{\text{obs}}|\alpha,\Sigma) = \prod_i \int \pi(y_i|\theta)\pi(\theta|\alpha,\Sigma)d\theta$$

OUR METHOD

- parameter estimation requires repeatedly evaluating an integral

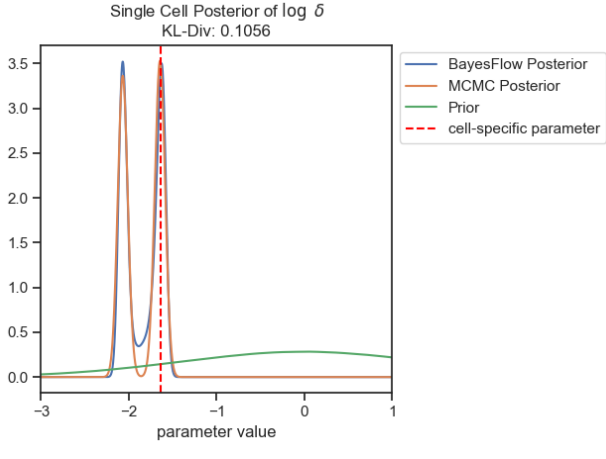
$$\pi(y_{\text{obs}}|\alpha,\Sigma) = \prod_i \int \pi(y_i|\theta)\pi(\theta|\alpha,\Sigma)d\theta$$

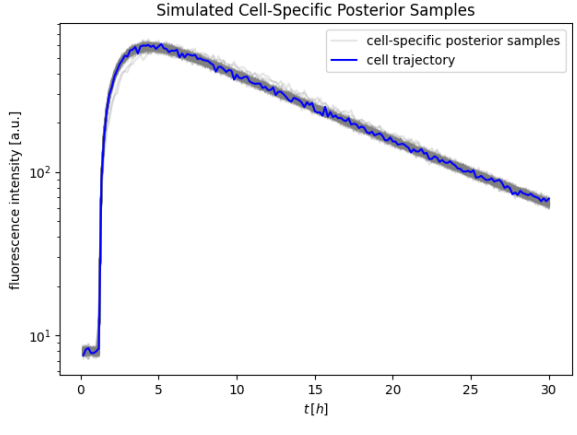
OUR METHOD

RESULTS

RESULTS

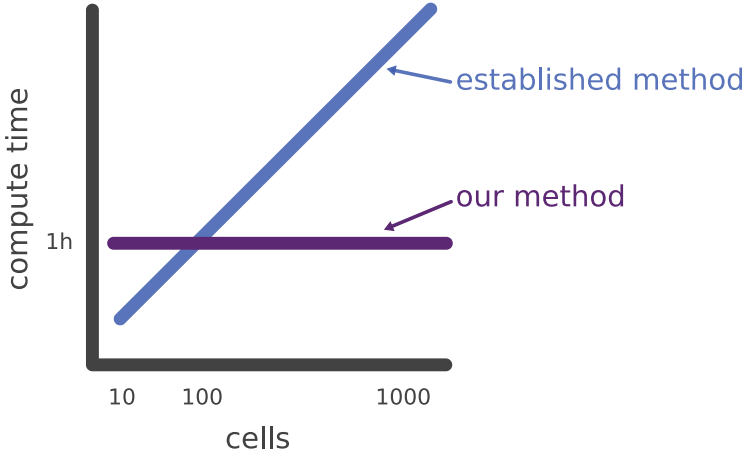
✓ NN **fits the posterior** well

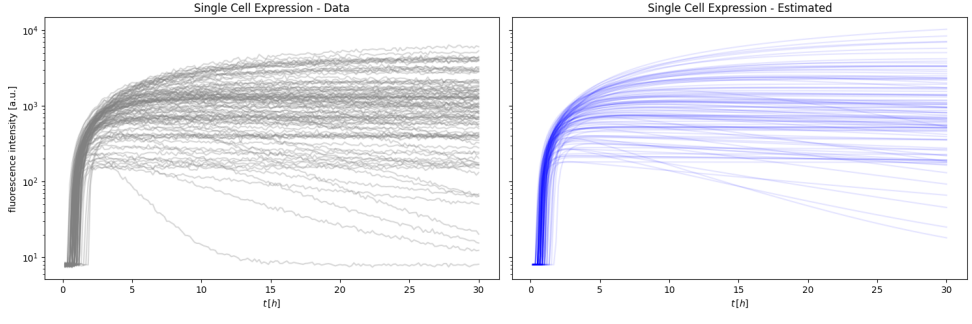




RESULTS

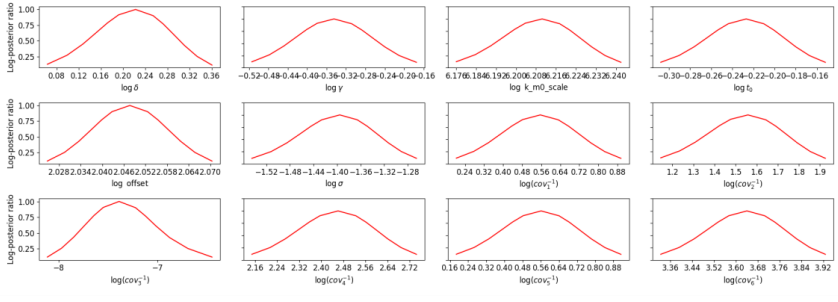
- ✓ NN **fits the posterior** well
- ✓ after training once $(1 - 2h)$, the optimization is **very fast** $(s - min)$

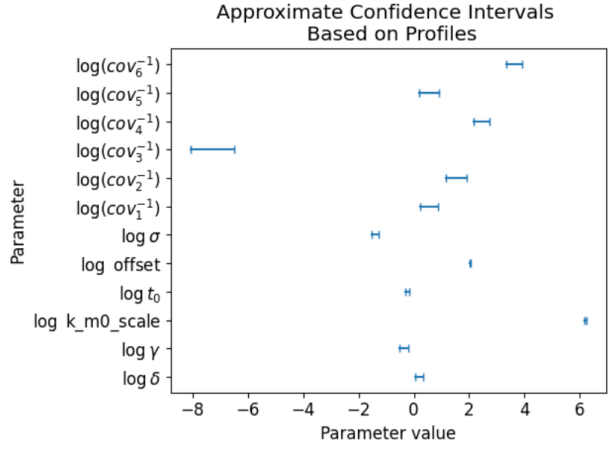




RESULTS

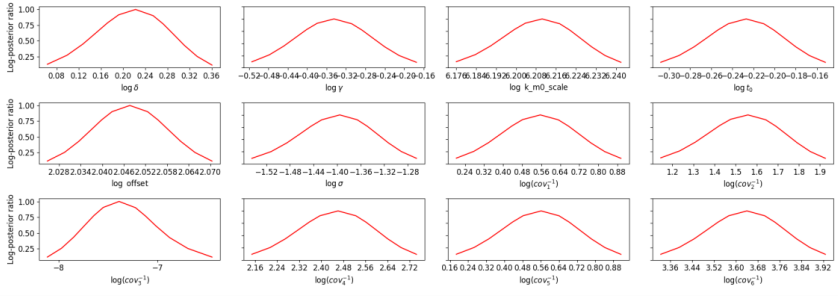
- ✓ NN **fits the posterior** well
- ✓ after training once $(1 - 2h)$, the optimization is **very fast** $(s - min)$
- ✓ even **uncertainty analysis** easily possible

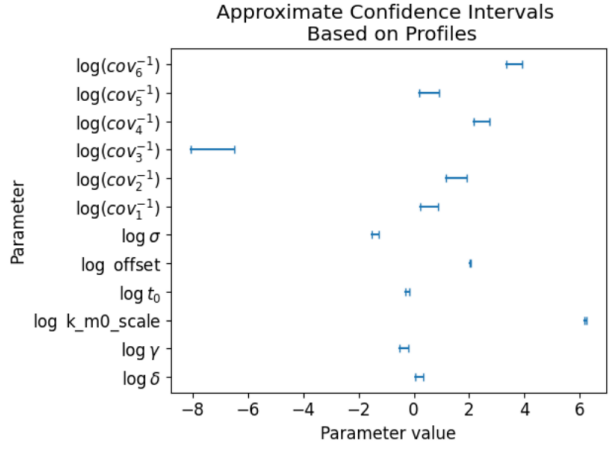




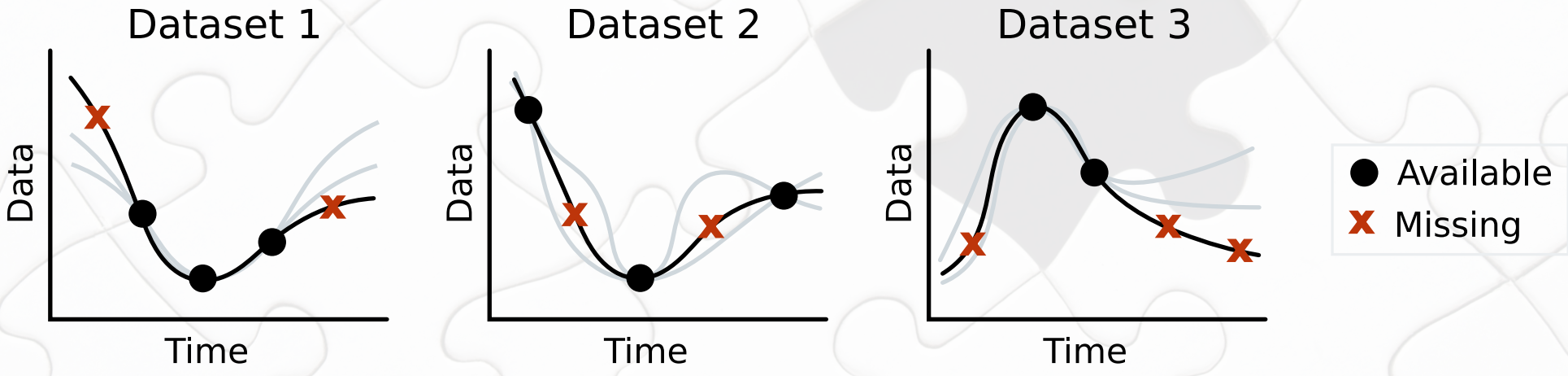
RESULTS

- ✓ NN **fits the posterior** well
- ✓ after training once $(1 - 2h)$, the optimization is **very fast** $(s - min)$
- ✓ even **uncertainty analysis** easily possible
- ✓ can handle multiple **experiment conditions**



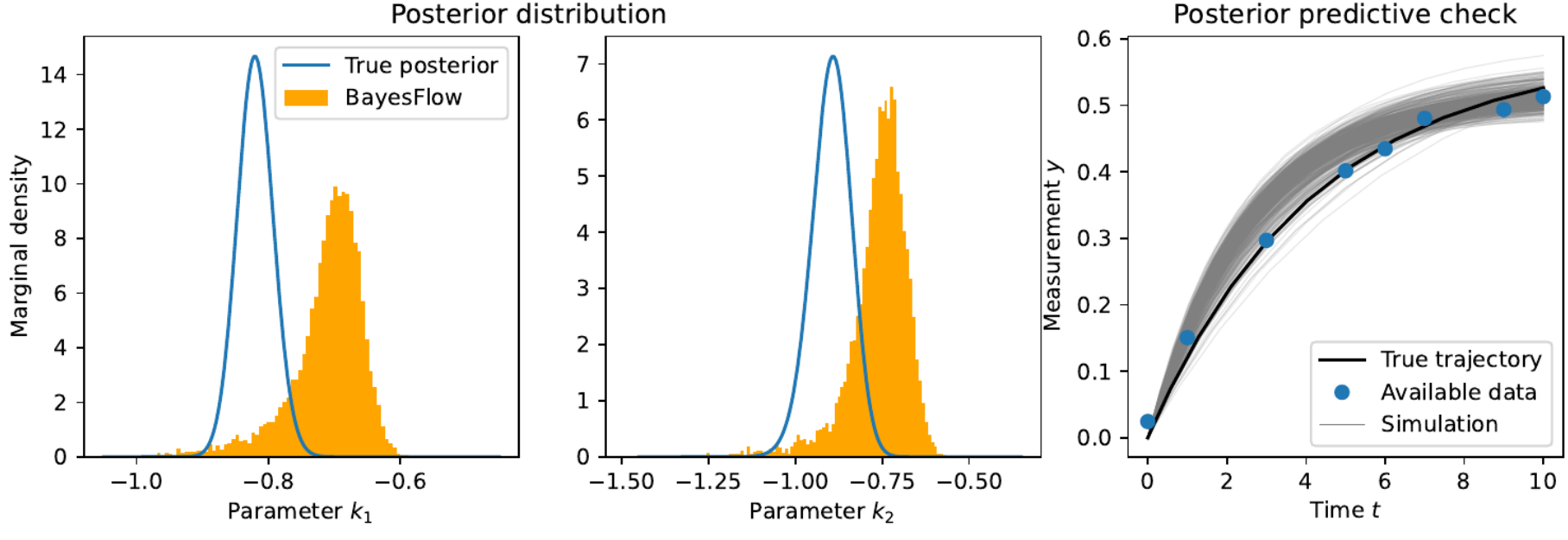


HOW TO HANDLE MISSING DATA IN AMORTIZED INFERENCE?



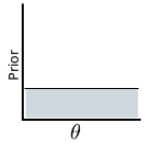
HOW TO HANDLE MISSING DATA IN AMORTIZED INFERENCE?

PROBLEM: BAYESFLOW CANNOT INTERPRET THE DATA



ENCODE MISSING DATA

Training phase



Sample from prior
$\theta \sim \pi(\theta)$

Sample simulated full data
$x \sim \pi(x|\theta)$

Example

$x = \begin{bmatrix} 1.8 & 0.9 & 0.1 & 3.3 & 4.1 \end{bmatrix}$

Sample missing entries
$\tau \sim \pi(\tau)$

$\tau = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \end{bmatrix}$

Simulated incomplete data
$x_{\text{avai}} = x \odot \tau + \text{NA} \cdot (1 - \tau)$

$x_{\text{avai}} = \begin{bmatrix} 1.8 & \text{NA} & 0.1 & 3.3 & \text{NA} \end{bmatrix}$

Encode missing data
$x_{\text{aug}} = x_{\text{aug}}(x_{\text{avai}})$

$x_{\text{aug}} = \begin{bmatrix} 1.8 & -1.0 & 0.1 & 3.3 & -1.0 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}$

Train BayesFlow networks

loss, gradient

Missingness encoding

Insert c

$x_{\text{aug}} = \begin{bmatrix} 1.8 & -1.0 & 0.1 & 3.3 & -1.0 \end{bmatrix}$

Augment by 0/1

$x_{\text{aug}} = \begin{bmatrix} 1.8 & -1.0 & 0.1 & 3.3 & -1.0 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}$

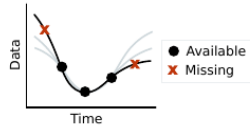
Time labels

$x_{\text{aug}} = \begin{bmatrix} 1.8 & 0.1 & 3.3 \\ 1 & 3 & 4 \end{bmatrix}$

Inference phase

Experiment

Example



Incomplete data
$x_{\text{avai}}^{\text{obs}}$

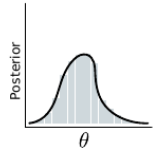
$x_{\text{avai}}^{\text{obs}} = \begin{bmatrix} \text{NA} & 3.9 & 2.5 & 3.4 & \text{NA} \end{bmatrix}$

Encode missing data
$x_{\text{aug}}^{\text{obs}} = x_{\text{aug}}(x_{\text{avai}}^{\text{obs}})$

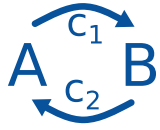
$x_{\text{aug}}^{\text{obs}} = \begin{bmatrix} -1.0 & 3.9 & 2.5 & 3.4 & -1.0 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$

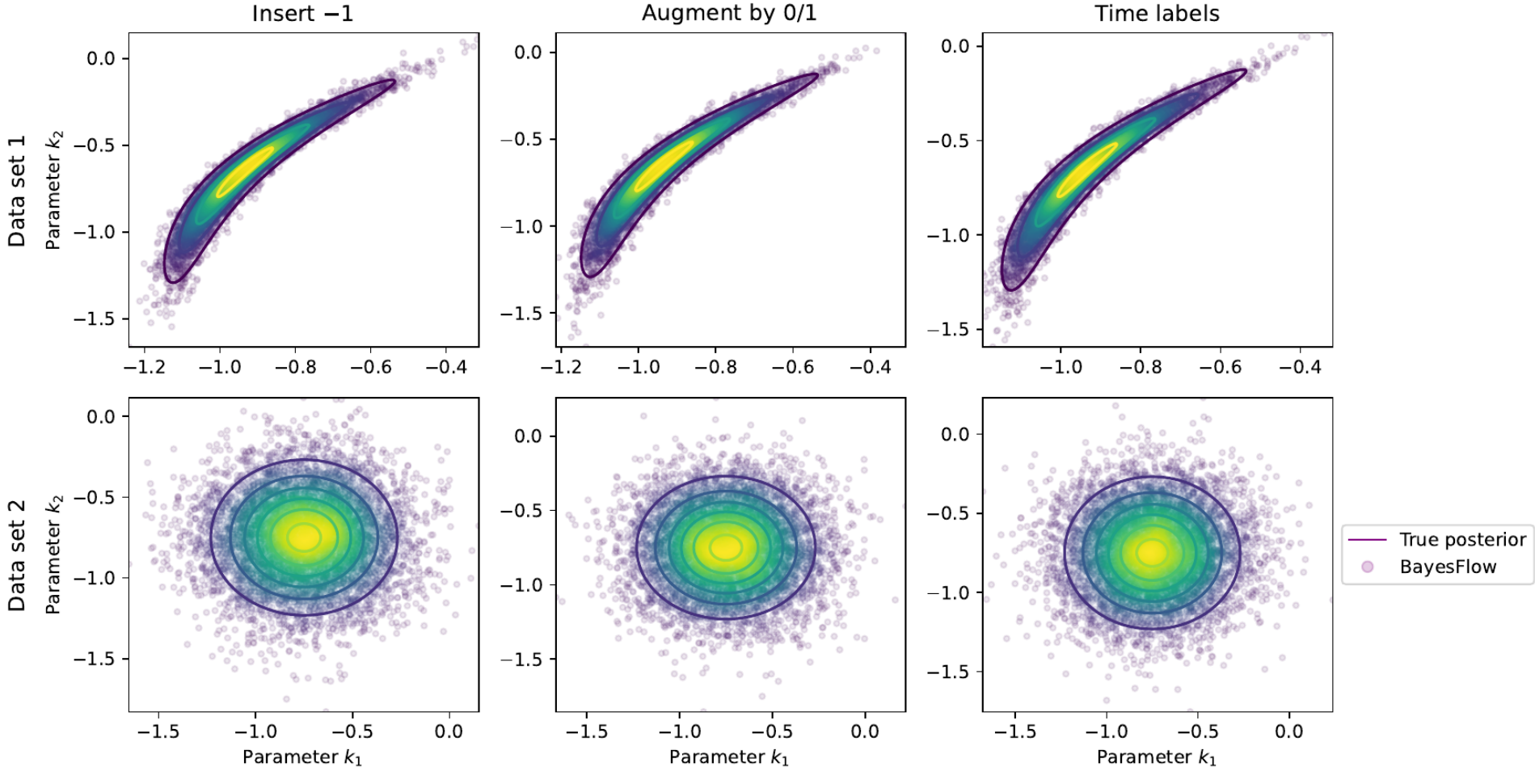
Sample from BayesFlow
$\theta \sim \pi(\theta|x_{\text{avai}}^{\text{obs}})$

Posterior approximation
$\{\theta_l\}_{l=1}^{L}$

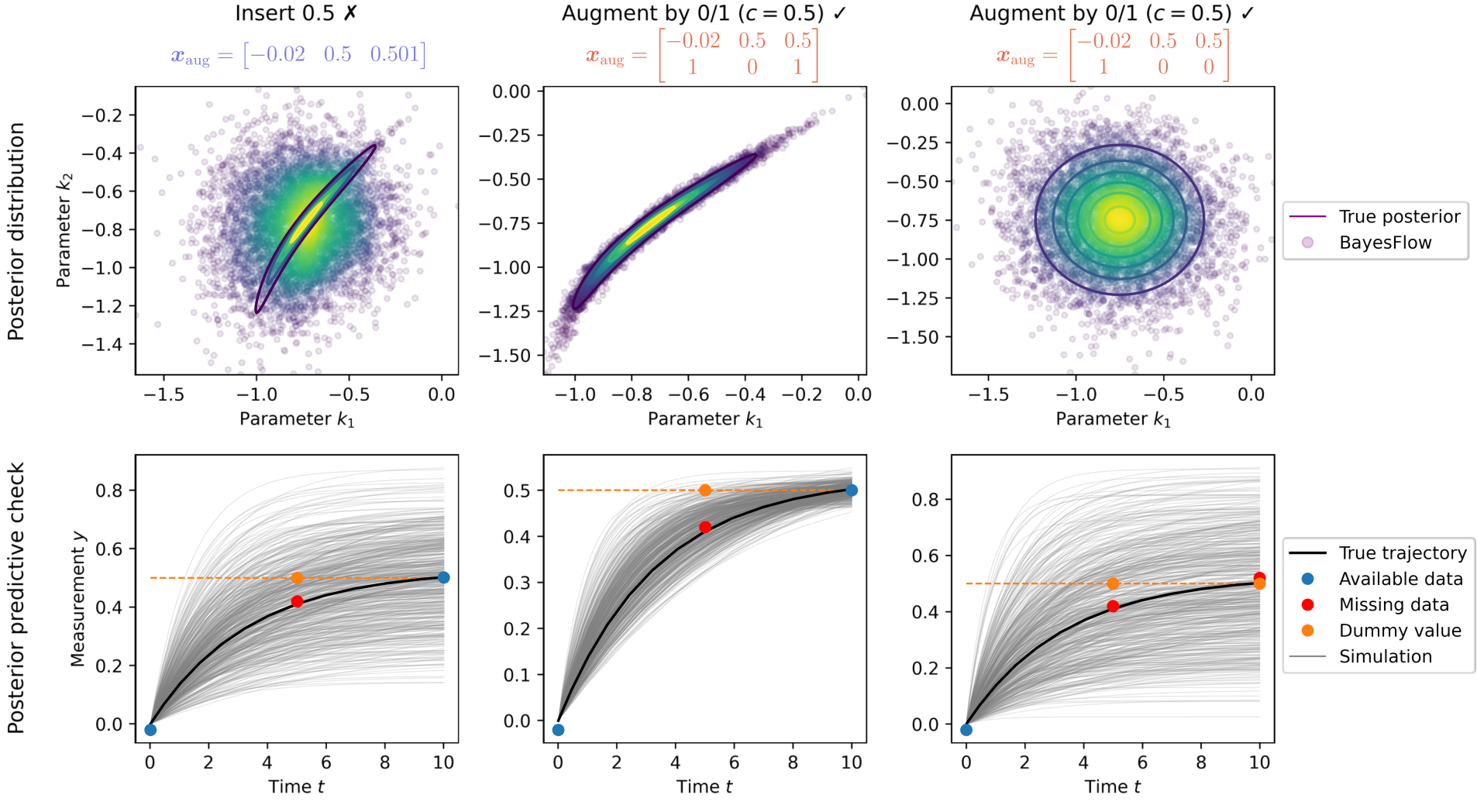


All approaches perform well on simple test problem

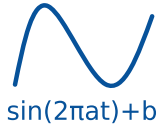


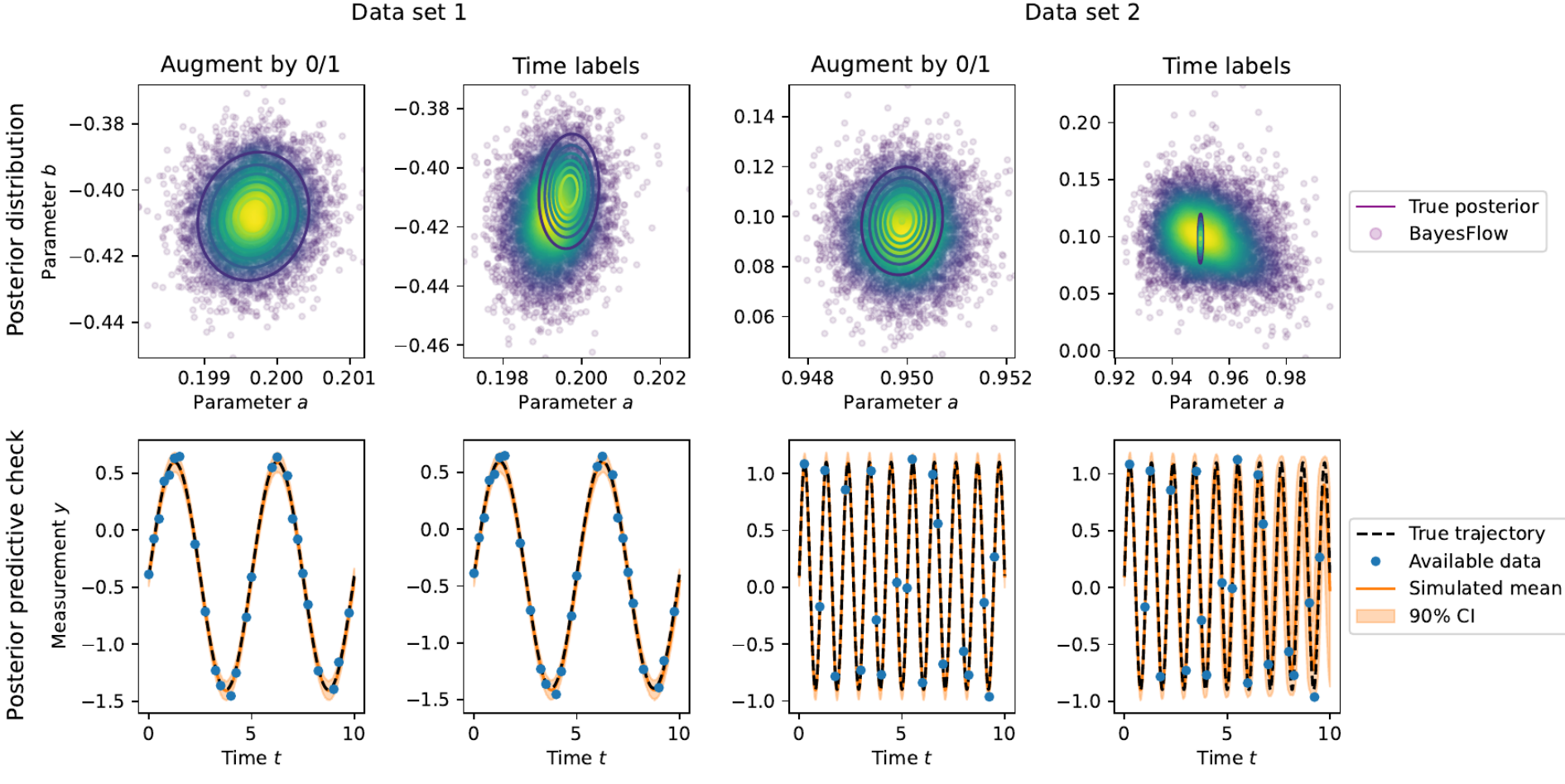


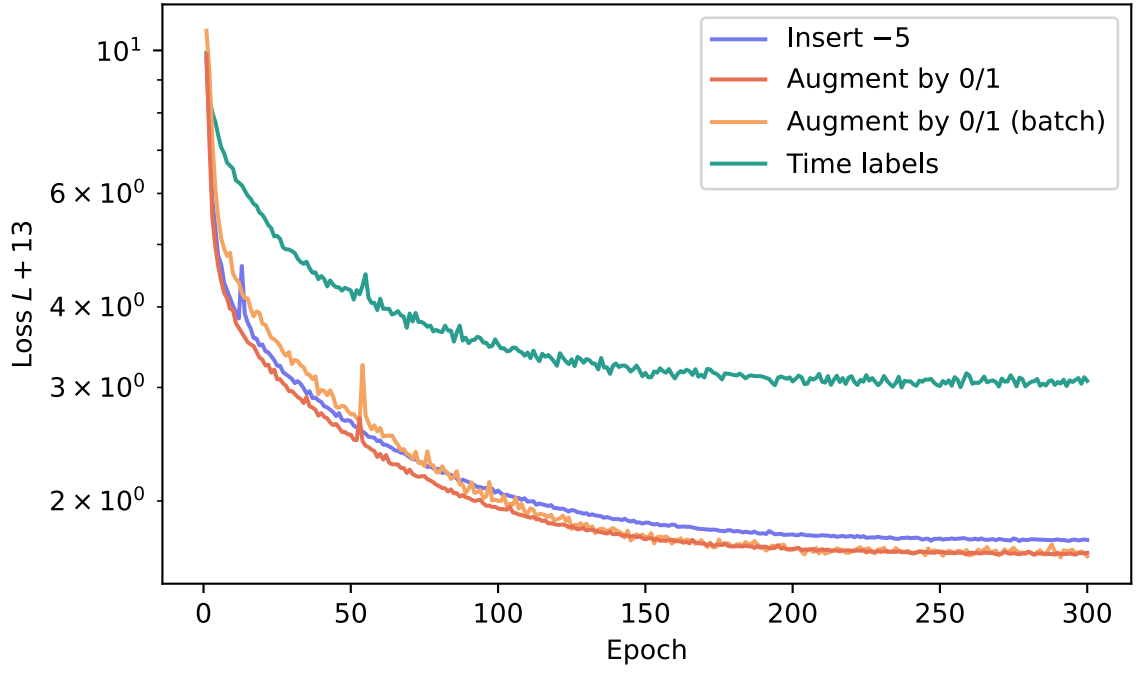
Binary indicator augmentation more robust for ambiguous fill-in values



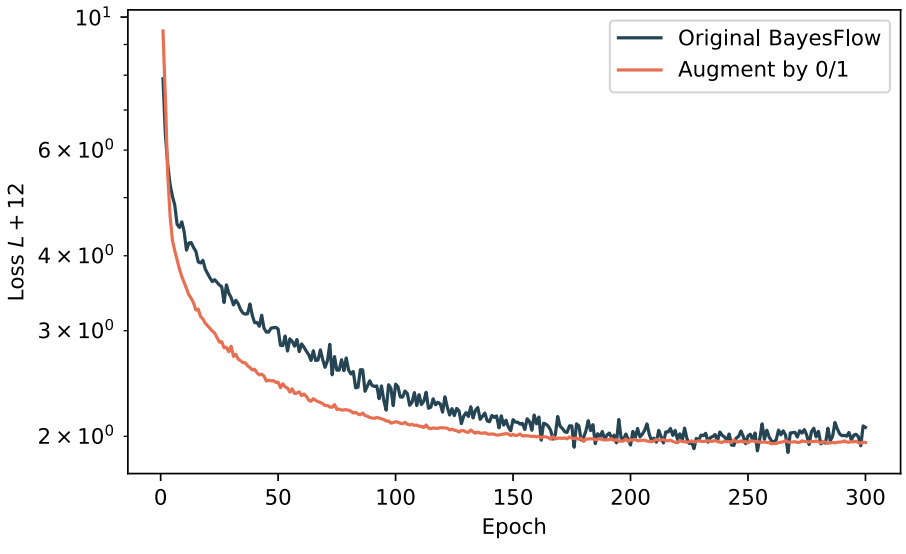
Time labels encoding performs not robustly in case of oscillatory data

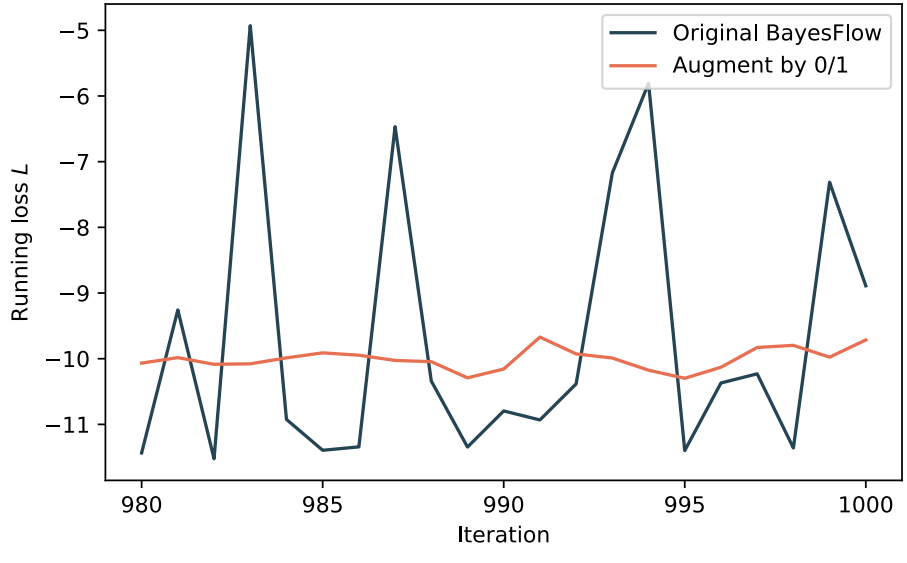





Insert −5
Augment by 0/1
Augment by 0/1 (batch)
Time labels
10^1
6×10^0
4×10^0
3×10^0
2×10^0
Loss $L + 13$
0
50
100
150
200
250
300
Epoch

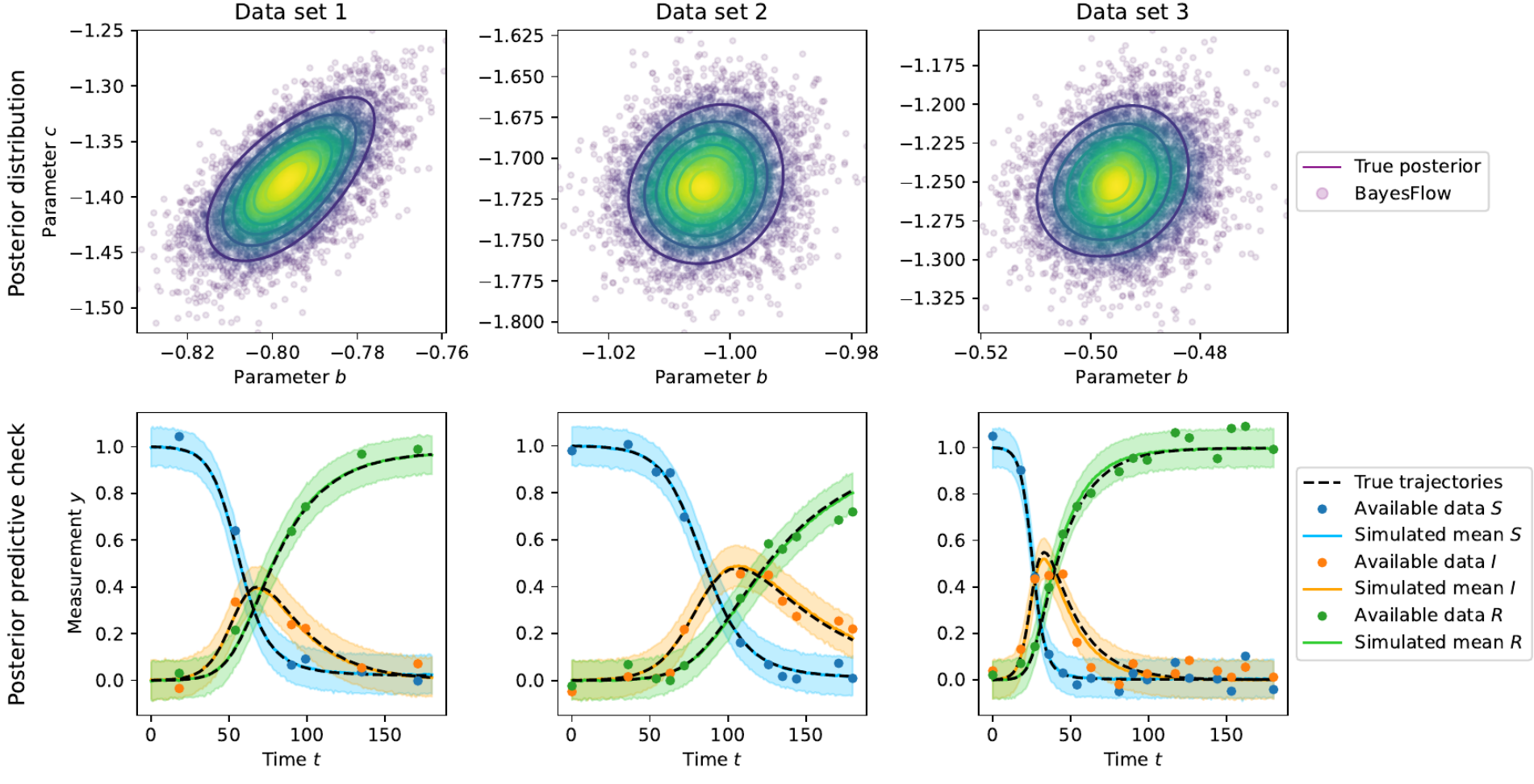
Variable dataset size as a special case of missing data



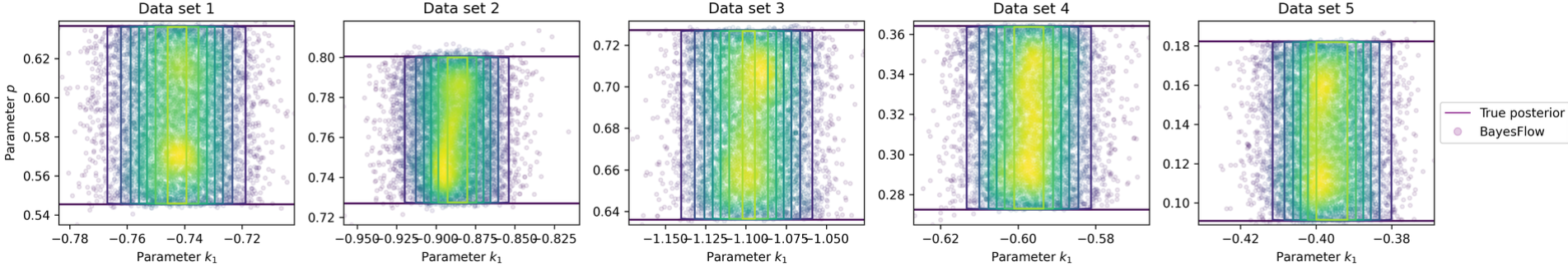


Augment by 0/1 improves performance due to better cost function approximation with individual-specific missingness

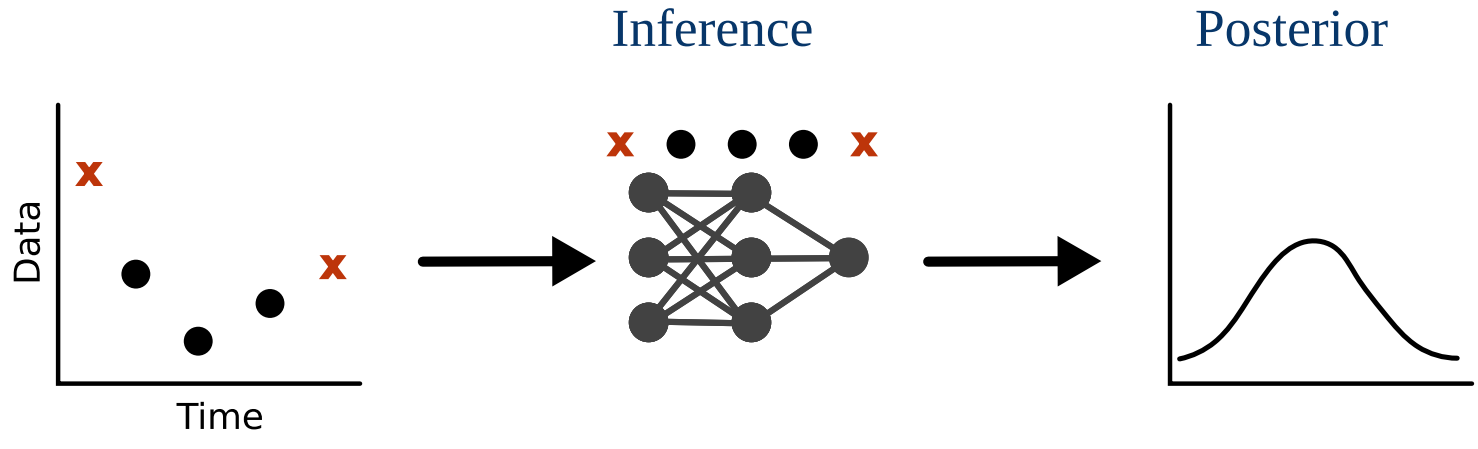
Scales to more complex inference problems



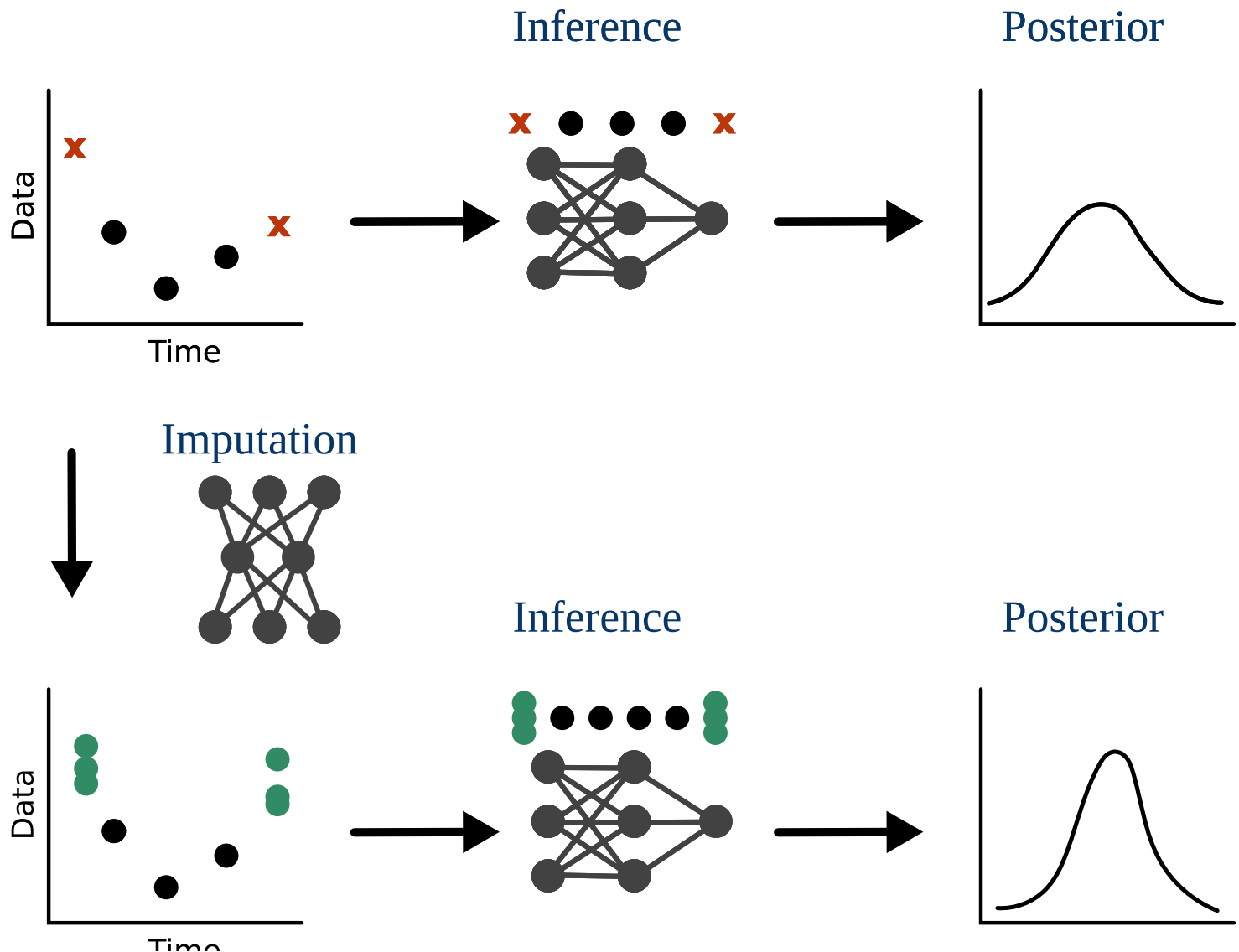
Able to unravel parameter-dependent missingness



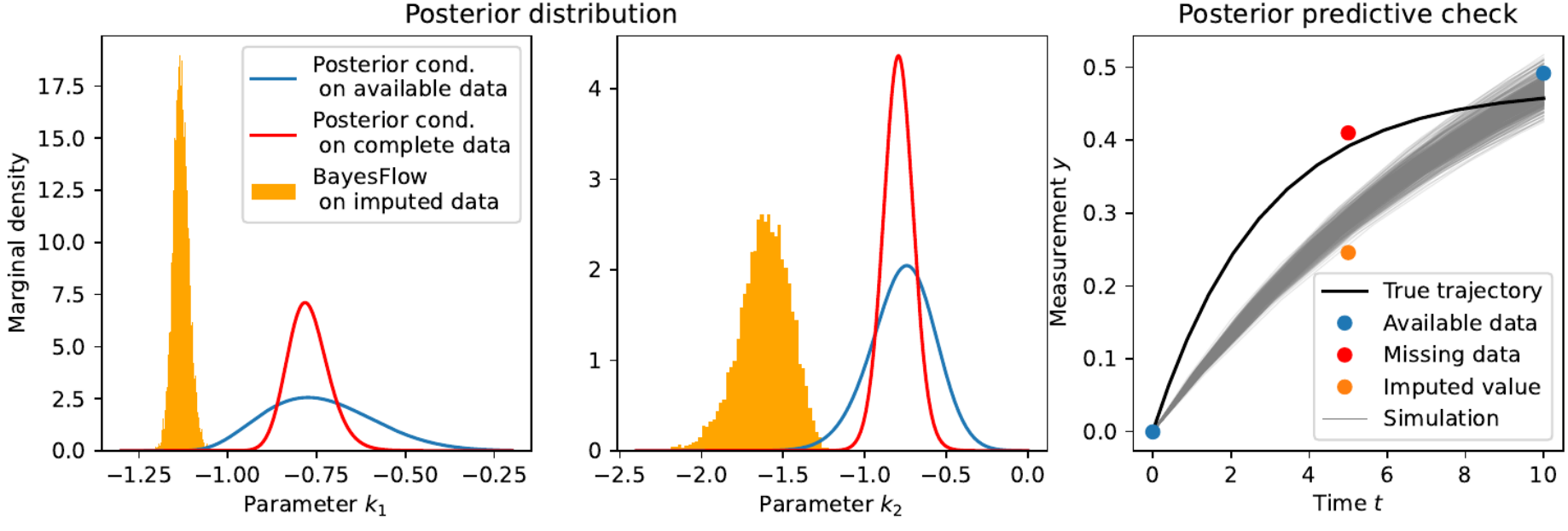
CAN WE JUST IMPUTE MISSING VALUES?



CAN WE JUST IMPUTE MISSING VALUES?



Inappropriate imputation can lead to biased results



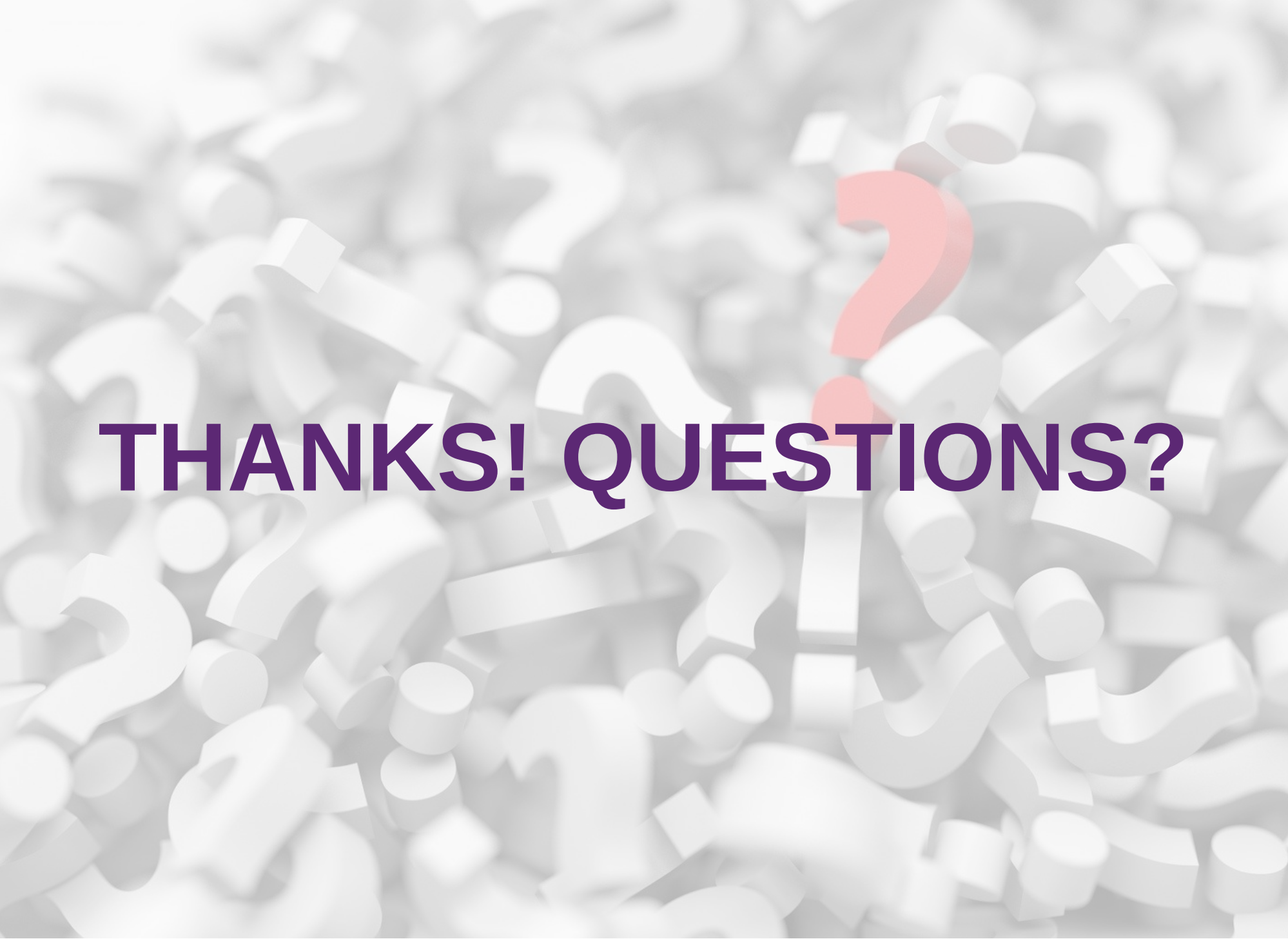
THERE ARE NO FREE DATA

Imputation means that instead of working with available data x, we try to reconstruct the full data $\bar{x}$, and estimate parameter probabilities $\pi(\theta|\bar{x})$ instead of $\pi(\theta|x)$. However, the true full data are unknown, therefore we need to take uncertainty in $\bar{x}$ into account, considering a full distribution of values $\pi(\bar{x}|x)$.

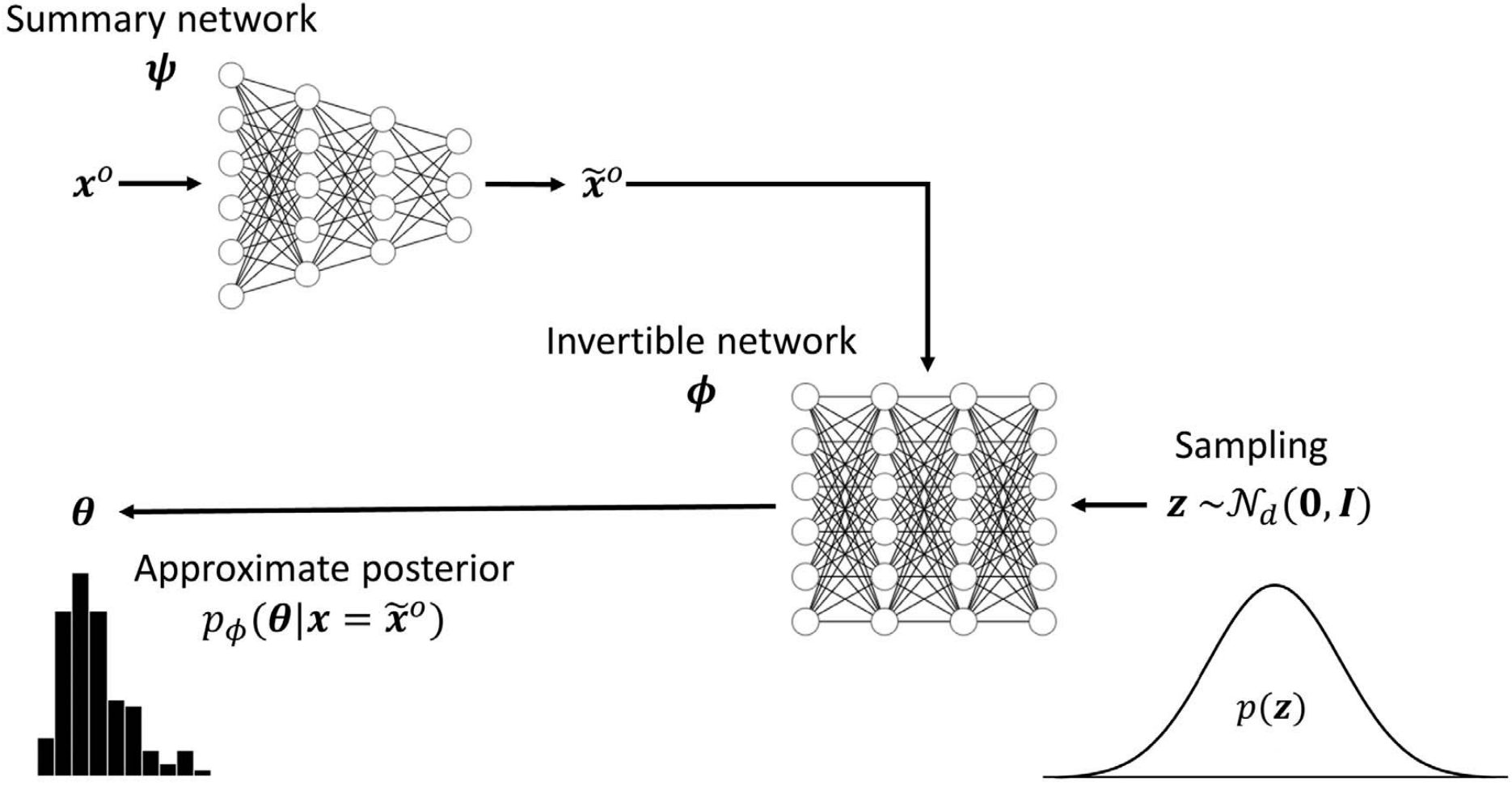
We must either make up a distribution (introducing a bias), or use a faithful approximation $p(\bar{x}|x) = \pi(\bar{x}|x)$ where $\pi(\bar{x}|x)\pi(x) = \pi(\bar{x}, x)$.

However, if we integrate out over all possible realizations of full data, we obtain $\int \pi(\theta|\bar{x})\pi(\bar{x}|x)d\bar{x} = \pi(\theta|x)$ (or similarly $\pi(\theta|x, \tau)$).

TLDR: When doing uncertainty quantification properly, we just recover the same posterior.


THANKS! QUESTIONS?

BAYESFLOW



from: Radev et al, IEEE Transactions on Neural Networks 2020

THE PROBLEM

- Classical simulation-based inference is case-based + slow + ε-approximate
- What if we want to fit the same model to multiple datasets?

THE IDEA

- Learn a global estimator for the probabilistic mapping $y \mapsto \theta$ via cINNs
- Once trained, amortize inference on arbitrarily many datasets
- Embed data via summary statistics model

GENERATIVE MODELS

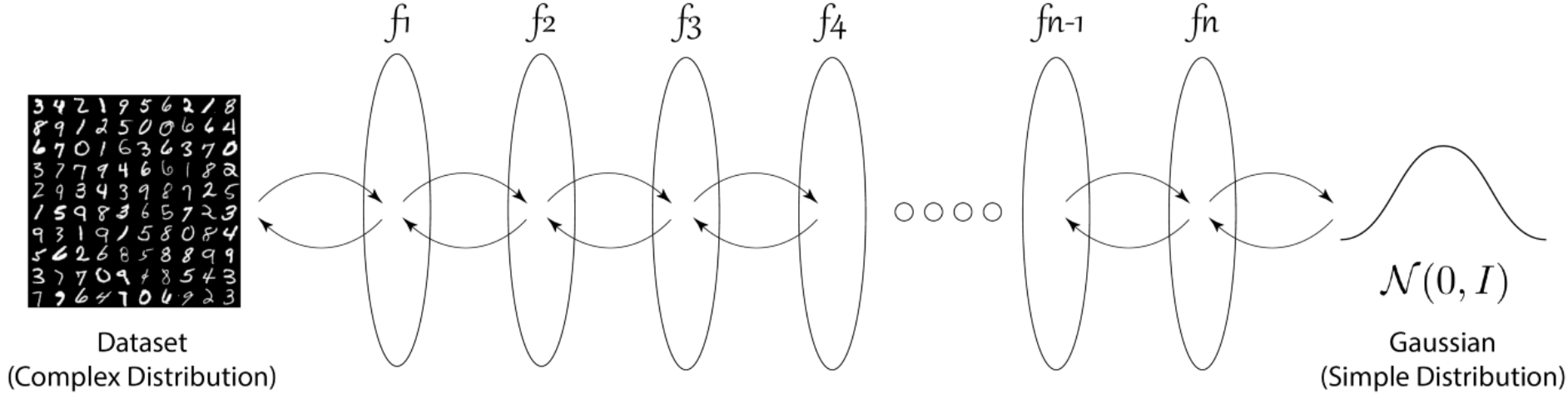
generate new data instances, $x \sim \pi(X|Y = y)$

from: Kingma et al, NeurIPS 2019

e.g.: GANs, VAEs, Flows

NORMALIZING FLOWS

generative models based on an invertible transformation



Let $z \sim \mathcal{N}(0, I)$ and $f : z \mapsto x$ bijective. Then via change of variable,s the pdf of $x = f(z)$ is given as

$$p_x(x) = p_z(f^{-1}(x)) \cdot |\det(\frac{df^{-1}}{dx}(x))|.$$

THE PROBLEM

- forward model $x_i \sim p(x|\theta) \Leftrightarrow x_i = g(\theta, \xi_i)$ with $\xi_i \sim p(\xi)$
- Bayesian inference $p(\theta|x_{1:N}) \propto p(x_{1:N}|\theta)p(\theta)$
- aim: train an invertible neural network that approximates the true posterior $p_\phi(\theta|x) \approx p(\theta|x)\ \forall \theta, x$

THE METHOD

Goal: Approximate the true posterior $p_\phi(\theta|x) \approx p(\theta|x)\ \forall \theta, x.$

Parameterize p_ϕ in terms of a cINN given via a bijective $f_\phi : \mathbb{R}^D \to \mathbb{R}^D, \theta \mapsto z$, which implements a normalizing flow between θ and a Gaussian latent variable z,

$$\theta \sim p_\phi(\theta|x) \Leftrightarrow \phi = f_\phi^{-1}(z; x) \quad \text{with} \quad z \sim \mathcal{N}_D(z|0, I).$$

Seek neural network parameters $\hat{\phi}$ that minimize the KL divergence between true and approximate posterior $\forall x$, giving the objective ...

THE METHOD

$$\hat{\phi} = \arg\min_{\phi} \mathbb{E}_{p(x)}[\mathrm{KL}(p(\theta|x)\,||\,p_\phi(\theta|x))]$$

$$= \arg\max_{\phi} \iint p(x,\theta)\log p_\phi(\theta|x)dxd\theta$$

$$= \arg\max_{\phi} \iint p(x,\theta)(\log p(f_\phi(\theta;x)) + \log|\det J_{f_\phi}|)dxd\theta$$

Approximate via Monte-Carlo sample:

SUMMARY STATISTICS LEARNING

If data $x_{1:N}$ are high-dimensional: Jointly learn a summary network $\tilde{x} = h_\psi(x_{1:N})$, giving the objective

$$\hat{\phi}, \hat{\psi} = \arg\max_{\phi,\psi)} \mathbb{E}_{p(x,\theta,N)}[\log p_\phi(\theta | h_\psi(x_{1:N})]$$

with Monte-Carlo estimate

$$\hat{\phi}, \hat{\psi} = \arg\min_{\phi,\psi} \frac{1}{M} \sum_{m=1}^{M} \left(\frac{|f_\phi(\theta^{(m)}; h_\psi(x_{1:N}^{(m)})|_2^2}{2} - \log |\det(J_{f_\phi}^{(m)})| \right)$$

Learning phase:

- create plenty of synthetic data $(y_i, \theta_i) \sim \pi(y, \theta)$
- train a cINN in forward mode

Inference phase:

- sample many latent $z_i \sim \pi(z)$
- run cINN backwards, $\theta_i = g(z_i; y_{\text{obs}}) \sim \pi(\theta | y_{\text{obs}})$

✓ fast + accurate amortized Bayesian inference