Missing Data Handling in Amortized Bayesian Inference via Invertible Neural Networks

Zijian Wang

Final Bachelor Thesis Presentation
September 5, 2022

Outline

Problem Setting

- Infer underlying parameters $\boldsymbol{\theta}$ from dataset $\boldsymbol{x}_{1:N} = \{\boldsymbol{x}_i\}_{i=1}^N$
- **Bayesian inference** aims at assessing the posterior distribution:

$$p(\boldsymbol{\theta} \,|\, \boldsymbol{x}_{1:N}) \propto p(\boldsymbol{x}_{1:N} \,|\, \boldsymbol{\theta})\, p(\boldsymbol{\theta})$$

- **Likelihood-free** methods only make use of simulations from the forward model:

$$\boldsymbol{x}_{1:N} \sim p(\boldsymbol{x}_{1:N} \,|\, \boldsymbol{\theta}) \quad \Longleftrightarrow \quad \boldsymbol{x}_{1:N} = g(\boldsymbol{\theta}, \boldsymbol{\xi}) \text{ with } \boldsymbol{\xi} \sim p(\boldsymbol{\xi})$$

- **Amortized** inference means to find a global estimator of $p(\boldsymbol{\theta} \,|\, \boldsymbol{x}_{1:N})$ that works for any dataset $\boldsymbol{x}_{1:N}$

Learning the Posterior via Normalizing Flows

- Goal: Approximate the true posterior $p(\boldsymbol{\theta}\,|\,\boldsymbol{x}) \approx p_{\boldsymbol{\phi}}(\boldsymbol{\theta}\,|\,\boldsymbol{x})$ for all $\boldsymbol{x}$
- Idea: Reparametrize $p_{\boldsymbol{\phi}}$ in terms of an invertible function $f_{\boldsymbol{\phi}}\colon \mathbb{R}^{n_{\boldsymbol{\theta}}} \to \mathbb{R}^{n_{\boldsymbol{\theta}}}$ which performs a normalizing flow between $\boldsymbol{\theta}$ and a Gaussian latent variable $\boldsymbol{z}$:

$$\boldsymbol{\theta} \sim p_{\boldsymbol{\phi}}(\boldsymbol{\theta}\,|\,\boldsymbol{x}) \quad \Longleftrightarrow \quad \boldsymbol{\theta} = f_{\boldsymbol{\phi}}^{-1}(\boldsymbol{z}; \boldsymbol{x}) \text{ with } \boldsymbol{z} \sim \mathcal{N}_{n_{\boldsymbol{\theta}}}(\mathbf{0}, \boldsymbol{I})$$

- BayesFlow: Implement $f_{\boldsymbol{\phi}}$ as conditional invertible neural network (cINN)

Derivation of Loss Function

- Seek neural network parameters ϕ such that the "distance" between $p_{\phi}(\boldsymbol{\theta} \,|\, \boldsymbol{x})$ and $p(\boldsymbol{\theta} \,|\, \boldsymbol{x})$ is small

- Formally, we minimize the average KL divergence over all datasets:

$$\begin{aligned}
\widehat{\boldsymbol{\phi}} &= \operatorname*{argmin}_{\boldsymbol{\phi}} \mathbb{E}_{p(\boldsymbol{x})}[\mathbb{KL}(p(\boldsymbol{\theta} \,|\, \boldsymbol{x}) \,||\, p_{\boldsymbol{\phi}}(\boldsymbol{\theta} \,|\, \boldsymbol{x}))] \\
&= \dots \\
&= \operatorname*{argmin}_{\boldsymbol{\phi}} \left[- \iint p(\boldsymbol{\theta}, \boldsymbol{x}) \log p_{\boldsymbol{\phi}}(\boldsymbol{\theta} \,|\, \boldsymbol{x}) \, d\boldsymbol{\theta} \, d\boldsymbol{x} \right]
\end{aligned}$$

Derivation of Loss Function (cont.)

- Apply the change of variables formula

$$p_{\boldsymbol{\phi}}(\boldsymbol{\theta} \,|\, \boldsymbol{x}) = p_{\boldsymbol{z}}(f_{\boldsymbol{\phi}}(\boldsymbol{\theta}; \boldsymbol{x})) \left|\det \boldsymbol{J}_{f_{\boldsymbol{\phi}}}\right|$$

  and insert the prescribed density $p_{\boldsymbol{z}}(\boldsymbol{z}) \propto \exp\left(-\frac{1}{2}\|\boldsymbol{z}\|_2^2\right)$ to obtain:

$$\widehat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \iint p(\boldsymbol{\theta}, \boldsymbol{x}) \left(\frac{1}{2} \left\| f_{\boldsymbol{\phi}}(\boldsymbol{\theta}; \boldsymbol{x}) \right\|_2^2 - \log \left|\det \boldsymbol{J}_{f_{\boldsymbol{\phi}}}\right| \right) d\boldsymbol{\theta}\, d\boldsymbol{x}$$

- Monte Carlo estimate (batch size M):

$$\widehat{\boldsymbol{\phi}} \approx \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \frac{1}{M} \sum_{m=1}^{M} \left(\frac{1}{2} \left\| f_{\boldsymbol{\phi}}\left(\boldsymbol{\theta}^{(m)}; \boldsymbol{x}^{(m)}\right) \right\|_2^2 - \log \left|\det \boldsymbol{J}_{f_{\boldsymbol{\phi}}}^{(m)}\right| \right)$$

Summary Statistics Learning

- To handle variable-size data, incorporate a summary network $h_{\boldsymbol{\psi}}$ that transforms raw data $\boldsymbol{x}_{1:N}$ into a fixed-size vector $h_{\boldsymbol{\psi}}(\boldsymbol{x}_{1:N})$
- Train the summary network jointly with the cINN:

$$\widehat{\boldsymbol{\phi}}, \widehat{\boldsymbol{\psi}} = \underset{\boldsymbol{\phi},\boldsymbol{\psi}}{\operatorname{argmin}} \, \mathbb{E}_{p(N,\boldsymbol{x}_{1:N})}[\mathbb{KL}(p(\boldsymbol{\theta} \,|\, \boldsymbol{x}_{1:N}) \,||\, p_{\boldsymbol{\phi}}(\boldsymbol{\theta} \,|\, h_{\boldsymbol{\psi}}(\boldsymbol{x}_{1:N})))]$$

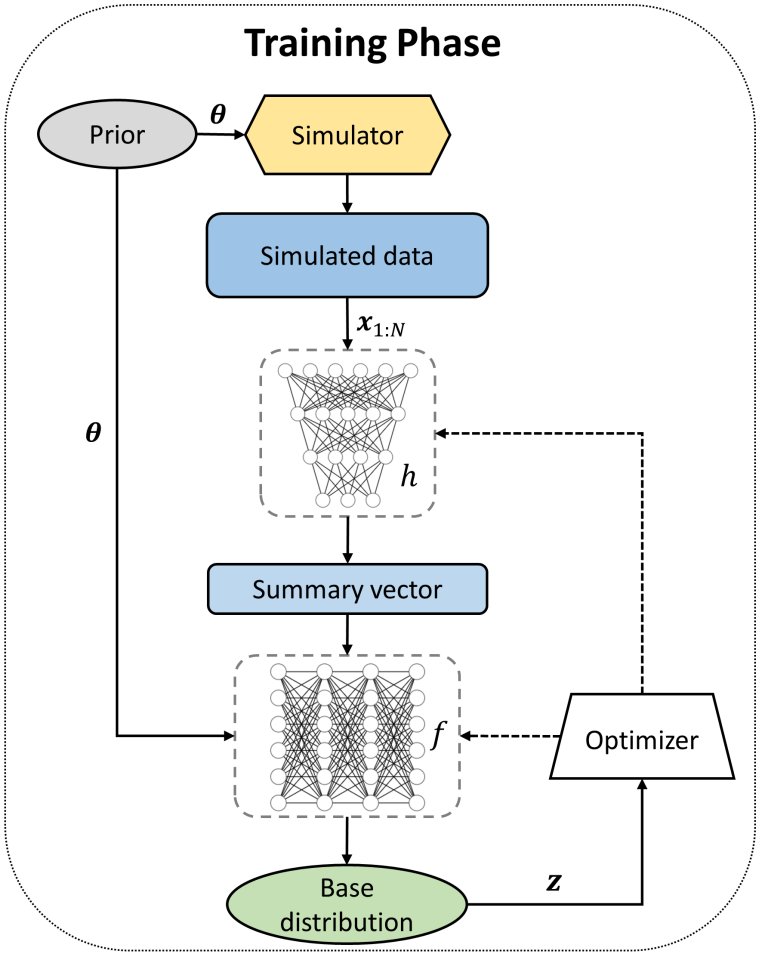
- Monte Carlo estimate of the objective:

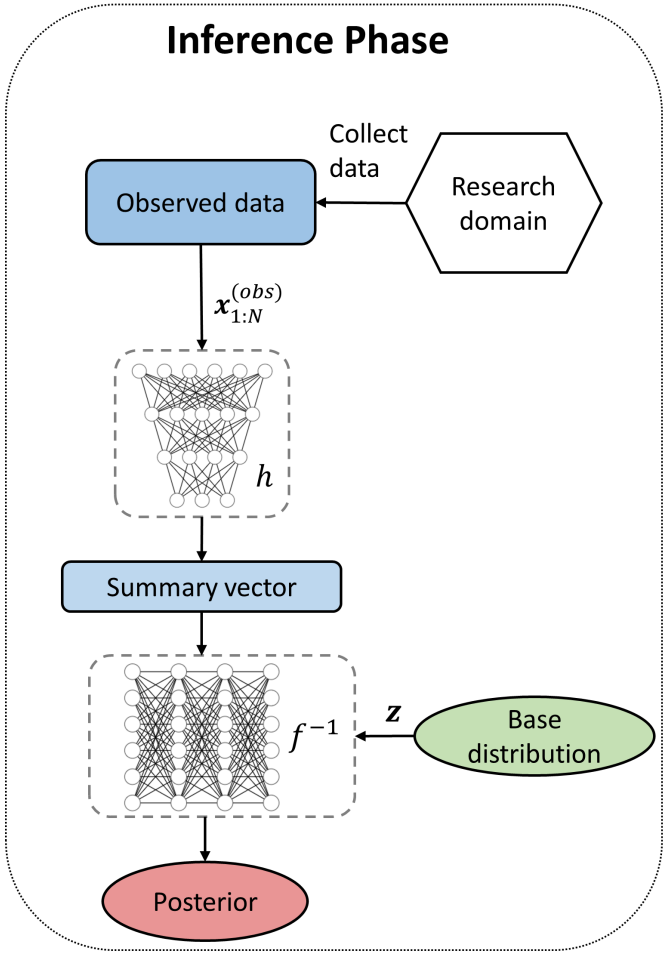
$$\widehat{\boldsymbol{\phi}}, \widehat{\boldsymbol{\psi}} \approx \underset{\boldsymbol{\phi},\boldsymbol{\psi}}{\operatorname{argmin}} \frac{1}{M} \sum_{m=1}^{M} \left(\frac{1}{2} \left\| f_{\boldsymbol{\phi}} \left(\boldsymbol{\theta}^{(m)}; h_{\boldsymbol{\psi}} \left(\boldsymbol{x}_{1:N^{(m)}}^{(m)} \right) \right) \right\|_2^2 - \log \left| \det \boldsymbol{J}_{f_{\boldsymbol{\phi}}}^{(m)} \right| \right)$$

Theoretical Exactness

- Assumption: Both the cINN and the summary network are sufficiently deep s.t. perfect convergence is achieved (analytical KL loss $\to 0$)
- Then it can be proven:
  - The learned posterior $p_{\widehat{\boldsymbol{\phi}}}\left(\boldsymbol{\theta} \,|\, h_{\widehat{\boldsymbol{\psi}}}(\boldsymbol{x}_{1:N})\right)$ approximates the true posterior $p(\boldsymbol{\theta} \,|\, \boldsymbol{x}_{1:N})$ perfectly for almost every dataset $\boldsymbol{x}_{1:N}$
  - The learned summary statistic $h_{\widehat{\boldsymbol{\psi}}}(\boldsymbol{x}_{1:N})$ is "maximally informative"

Amortized Posterior Sampling with BayesFlow





https://bayesflow.readthedocs.io/en/latest/

Outline

Missing Data

- Incomplete datasets in which no numerical value is stored for one or multiple observations

  $\Longrightarrow$ Can substantially complicate reliable statistical inference

- Highly relevant e.g. in biomedical sciences:
  - Clinical trials, where probands do not show up at some of the regular screenings
  - COVID-19 pandemic, where infection numbers are not reported on holidays or due to the overload of health departments

Problem Specification

- Time series model producing datasets $\boldsymbol{x}_{1:N}$ of fixed length N, where the $\boldsymbol{x}_i \in \mathbb{R}^{n_{\boldsymbol{x}}}$ are assigned to time points $0 \leq t_1 < \cdots < t_N$
- Underlying complete dataset $\boldsymbol{x}_{1:N}$ is only partially known:

$$\mathcal{S} \coloneqq \{1 \leq i \leq N : \text{ No value available for } \boldsymbol{x}_i\}, \quad \mathcal{T} \coloneqq \{1, \ldots, N\} \setminus \mathcal{S}$$

- Two possible target distributions:
  - Posterior $p(\boldsymbol{\theta} \,|\, \boldsymbol{x}_{\mathcal{T}})$ cond. on the available data $\boldsymbol{x}_{\mathcal{T}} \coloneqq \{\boldsymbol{x}_i\}_{i \in \mathcal{T}}$
  - Posterior $p(\boldsymbol{\theta} \,|\, \boldsymbol{x}_{1:N})$ cond. on the complete original dataset $\boldsymbol{x}_{1:N}$

$\Longrightarrow$ Focus on the estimation of $p(\boldsymbol{\theta} \,|\, \boldsymbol{x}_{\mathcal{T}})$

Encoding Missing Values

- Goal: Teach neural network to detect missing values and ignore them for the inference
- Investigated approaches to encode missing data:
  - **Augment by $0/1$**: Augment the data matrix $\left[\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\right]$ by a binary row, where 0 indicates absence and 1 indicates presence of a data point. Further, insert a constant value $C \in \mathbb{R}$ for all missing data points.
  - **Insert $\mathbf{C}$**: Simply insert a constant $C \in \mathbb{R}$ for all missing data points.
  - **Time labels**: Augment the data matrix containing only the available observations by a row of time points associated to these observations.

Encoding Missing Values - Examples

- Incomplete dataset:

$$\begin{bmatrix} 0.9 & \text{N/A} & 3.1 & 4.0 & \text{N/A} \end{bmatrix}$$

- Augment by $0/1$ with fill-in value -1:

$$\Longrightarrow \quad \begin{bmatrix} 0.9 & -1.0 & 3.1 & 4.0 & -1.0 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

- Insert -1: $\Longrightarrow \quad \begin{bmatrix} 0.9 & -1.0 & 3.1 & 4.0 & -1.0 \end{bmatrix}$

- Time labels: $\Longrightarrow \quad \begin{bmatrix} 0.9 & 3.1 & 4.0 \\ 0.0 & 1.0 & 1.5 \end{bmatrix}$

Integration into BayesFlow

- Choose maximum number $N_{\varnothing}^{\max} < N$ of missing data points
- Train network on datasets with artificially induced missing values that are encoded via one of the three approaches
- For our test models:

  Number of missing data points $N_{\varnothing} \in \{0, \ldots, N_{\varnothing}^{\max}\}$ and set of missing time indices $\mathcal{S} \subseteq \{1, \ldots, N\}$ are sampled uniformly

Outline

Performance on Complete Data - Brief Overview

- Nearly perfect approximation of the true posterior can be achieved.
- For time series models, LSTM networks with approximately as many hidden units as the total number of entries in the data matrix are a reasonable choice of summary network.
- Hard-bounded posteriors resulting from uniform priors are difficult to learn.

Conversion Reaction Model

- Forward model:

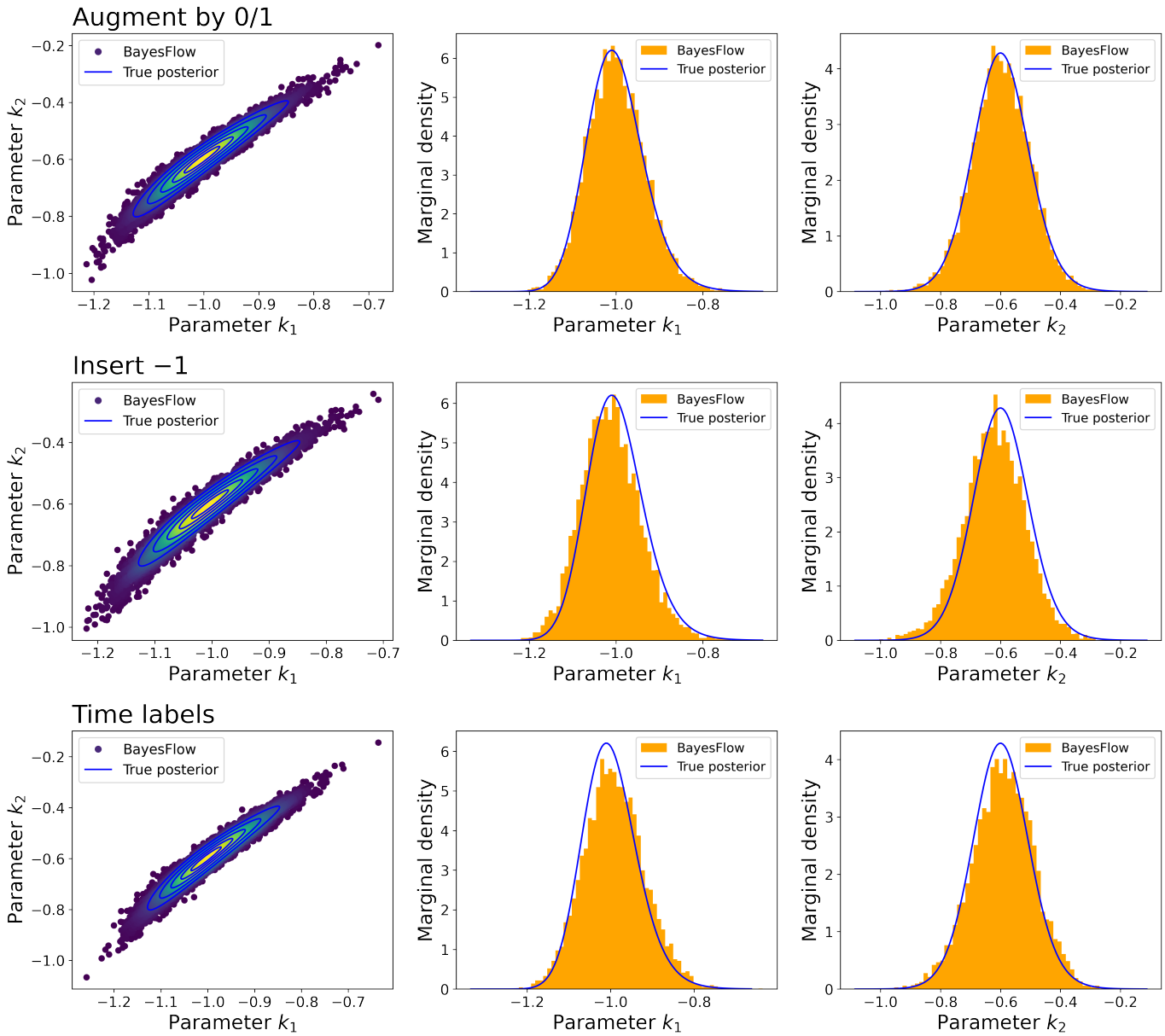
$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -c_1 x_1 + c_2 x_2 \\ c_1 x_1 - c_2 x_2 \end{pmatrix}, \quad \begin{pmatrix} x_1(0) \\ x_2(0) \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$y_t = x_2(t) + \varepsilon_t \text{ with } \varepsilon_t \sim \mathcal{N}(0,\, 0.015^2)$$

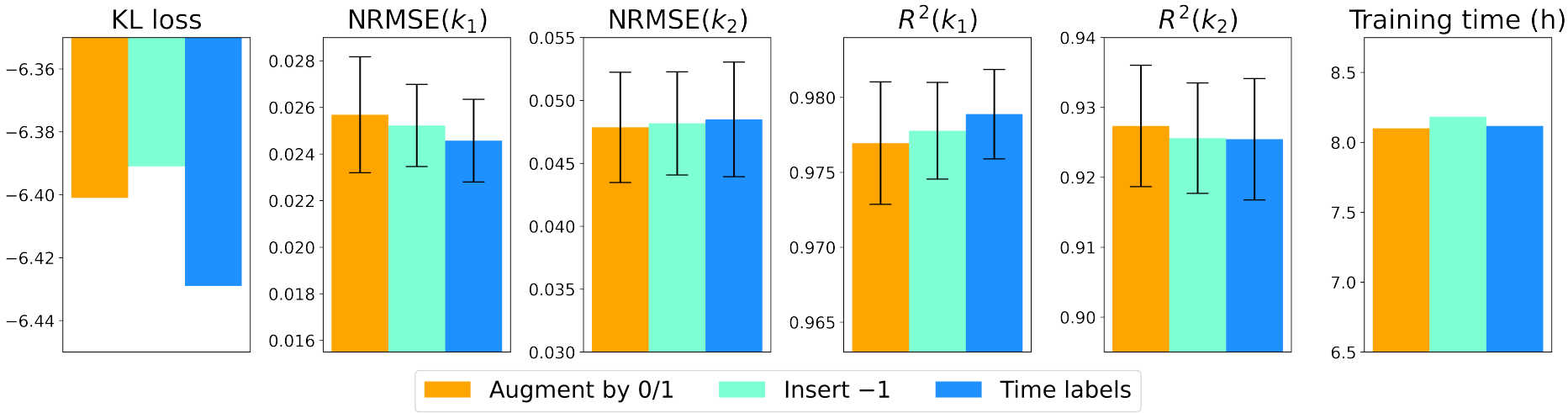
- $N = 11$ observations at $t = 0, 1, \ldots, 10$ and $N_{\varnothing}^{\max} = 6$
- Normal prior on log-scale rate parameters $k_j = \log_{10}(c_j)$:

$$k_1, k_2 \sim \mathcal{N}(-0.75,\, 0.25^2)$$

Comparing the Approaches to Encode Missing Data



Comparing the Approaches to Encode Missing Data (cont.)



$\Longrightarrow$ Similarly good performance of all 3 approaches!

Increased Robustness via Augmentation by 0/1

- "Augment by $0/1$" and "Insert C" require a fill-in value C
- CR model (positive trajectories):
  - Natural to choose an unambiguous fill-in value $C = -1$
  - Both networks correctly learn to interpret -1 as missing values
- More complex dynamics: Choice of fill-in value is non-trivial
- In the following: Test ambiguous value $C = 0.5$ on reduced CR model ($N = 3$ observations at $t = 0, 5, 10$ and $N_{\varnothing}^{\text{max}} = 2$)

Increased Robustness via Augmentation by 0/1 (cont.)

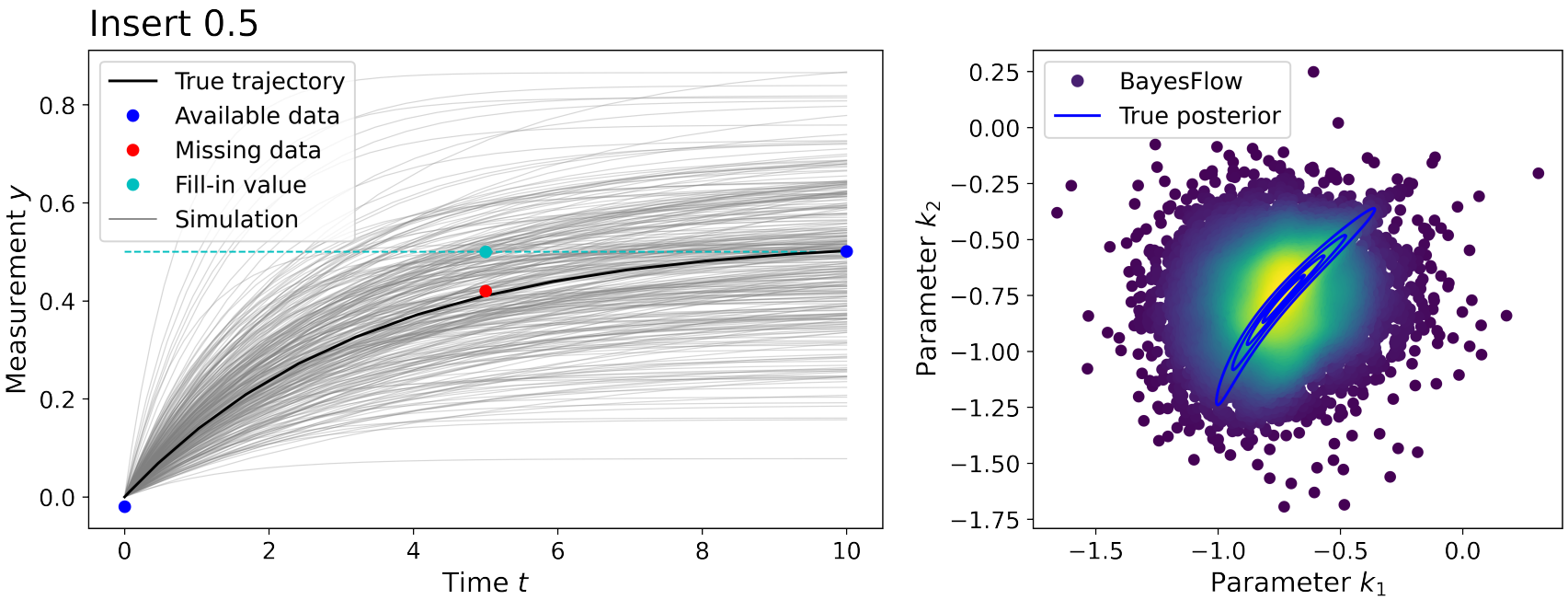


Figure: Test data: $\begin{bmatrix} -0.02 & 0.5 & 0.501 \end{bmatrix}$ (only the second observation is missing). However, the network misinterprets the signal 0.501 as another missing value.

Increased Robustness via Augmentation by 0/1 (cont.)

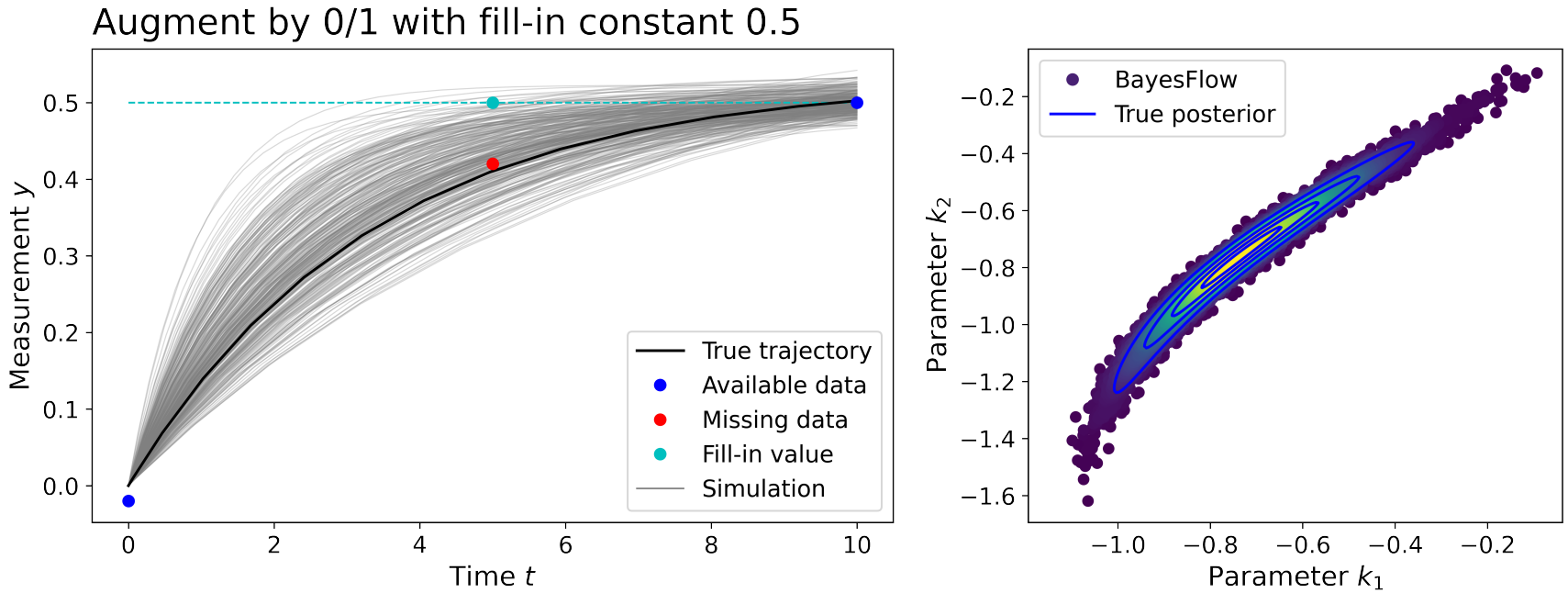


Figure: Test data: $\begin{bmatrix} -0.02 & 0.5 & 0.5 \\ 1 & 0 & 1 \end{bmatrix}$. Based on the augmentation, the network correctly identifies the value 0.5 in the second observation as a missing value and in the third observation as a signal.

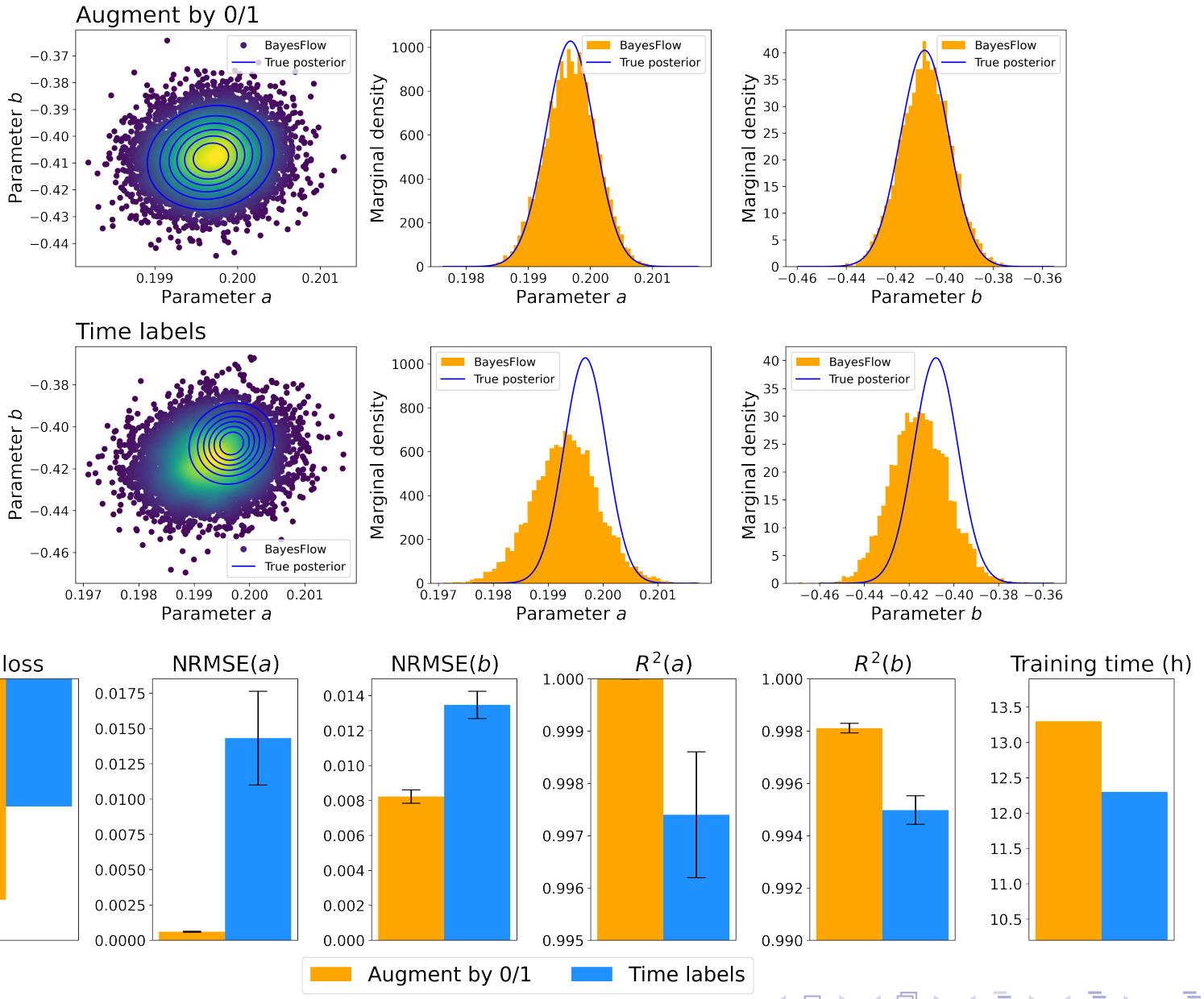
Oscillation Model

- Forward model:

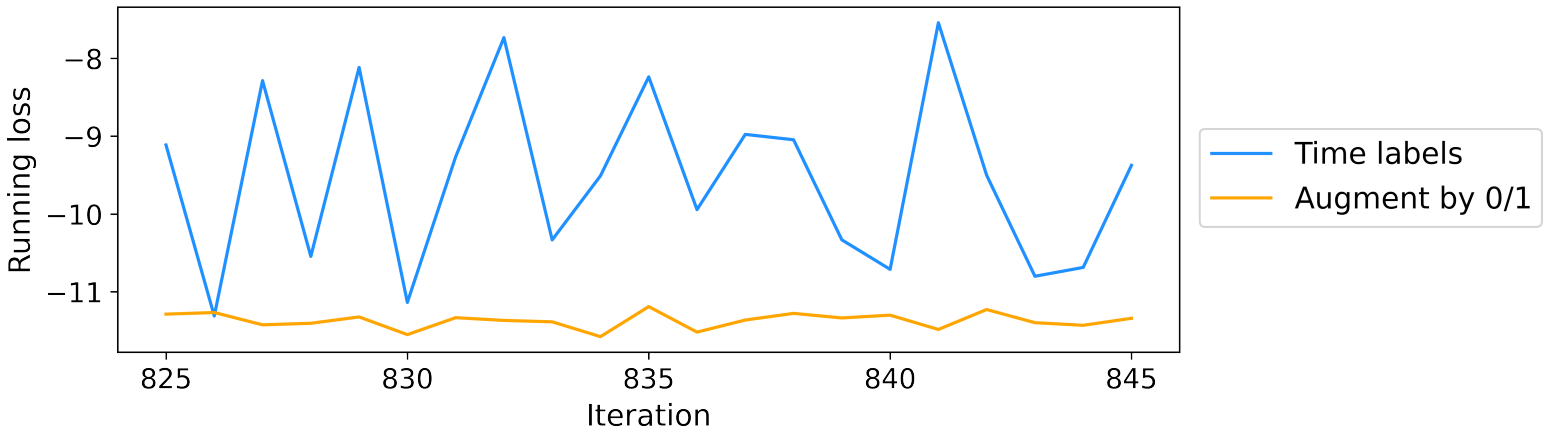
$$x(t) = \sin(2\pi a t) + b$$
$$y_t = x(t) + \varepsilon_t \text{ with } \varepsilon_t \sim \mathcal{N}(0,\, 0.05^2)$$

- Prior for frequency parameter: $a \sim \mathcal{U}(0.1,\, 1)$
- Prior for shift parameter: $b \sim \mathcal{N}(0,\, 0.25^2)$
- $N = 41$ observations at $t_k = \frac{1}{4}k$ for $k = 0, 1, \ldots, 40$ and $N_{\varnothing}^{\text{max}} = 21$
- Models producing oscillatory data are in general hard to fit (highly irregular objective function landscape)

Poor Convergence of the "Time labels" Approach



Poor Convergence of the "Time labels" Approach (cont.)



- Keeping $N_\varnothing$ fixed for each batch gives an inexact loss approximation and causes the running loss to jump
- Effect is especially strong for the more challenging loss function of the oscillation model

  $\Longrightarrow$ Convergence issues

SIR Epidemiology Model

- Forward model:

$$\begin{pmatrix} \dot{S} \\ \dot{I} \\ \dot{R} \end{pmatrix} = \begin{pmatrix} -\beta SI \\ \beta SI - \gamma I \\ \gamma I \end{pmatrix}, \quad \begin{pmatrix} S(0) \\ I(0) \\ R(0) \end{pmatrix} = \begin{pmatrix} 0.999 \\ 0.001 \\ 0 \end{pmatrix}$$

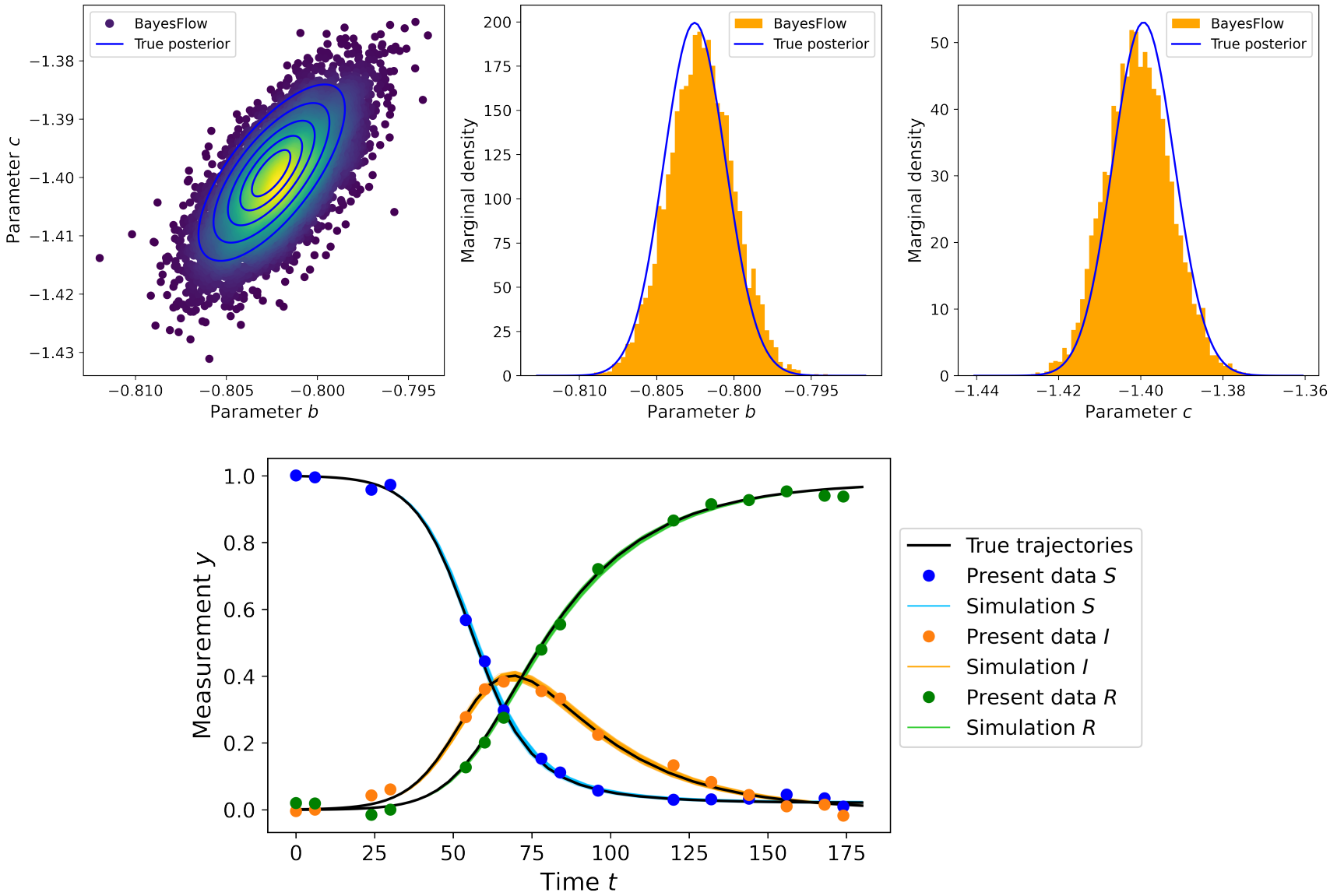
$$y_t = \begin{pmatrix} S(t) \\ I(t) \\ R(t) \end{pmatrix} + \begin{pmatrix} \varepsilon_{t,1} \\ \varepsilon_{t,2} \\ \varepsilon_{t,3} \end{pmatrix} \text{ with } \varepsilon_{t,i} \sim \mathcal{N}(0,\, 0.015^2)$$

- Normal priors on log-scale infection/recovery rate parameters:

$$b = \log_{10}(\beta) \sim \mathcal{N}(-1,\, 0.3^2), \quad c = \log_{10}(\gamma) \sim \mathcal{N}(-1.5,\, 0.3^2)$$

- $N = 31$ observations at $t_k = 6k$ for $k = 0, 1, \ldots, 30$ and $N_{\varnothing}^{\text{max}} = 15$

Performance of "Augment by 0/1" on SIR Model



Outline

Summary

- Extended the BayesFlow method to perform amortized likelihood-free Bayesian inference on incomplete datasets

  $\Longrightarrow$ Sampling from the posterior conditioned on the available data

- Train network on datasets with artificially induced missing data

- Compared three approaches to encode missing values

  $\Longrightarrow$ Most robust approach: "Augment by 0/1"

Outlook I

- Application to real-world problems from systems biology and epidemiology, e.g. NLME models
- Use parallelization to speed up batch simulation
- Explore more sophisticated LSTM network architecture
- Integrate known patterns behind the data missingness
- If data are missing independently in different observed features, perform augmentation for each feature row

Outlook II

- Recover posterior $p(\boldsymbol{\theta} \,|\, \boldsymbol{x}_{1:N})$ cond. on the complete original data
  $\Longrightarrow$ Stronger posterior contraction, better point estimates
- Data imputation based on:
  - Spline interpolation
  - Deep learning, e.g. LSTM networks or autoencoders
- Train BayesFlow on imputed data with an added binary row to label imputed/original values
- Continuous weights indicating trustworthyness of imputed values, e.g. from Gaussian process regression

Thank you for your attention!
I look forward to your questions and feedback!