

Amortized Simulation-Based Inference For Non-Linear Mixed-Effects Models

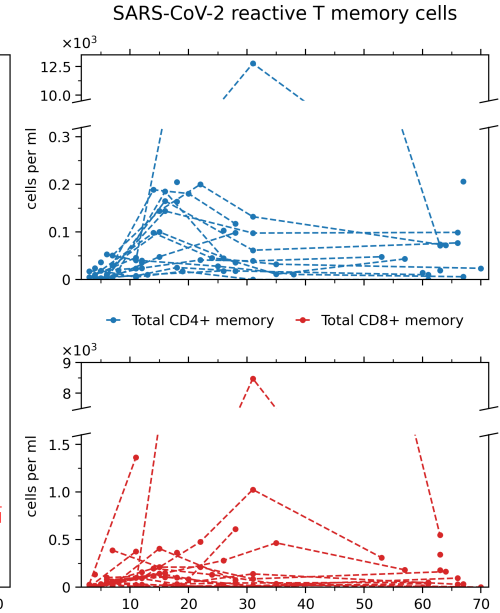
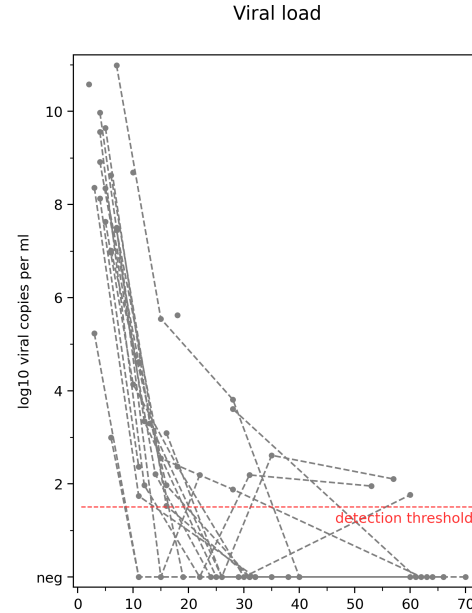
Jonas Arruda & Yannik Schälte - 29th June 2023

One World ABC Seminar

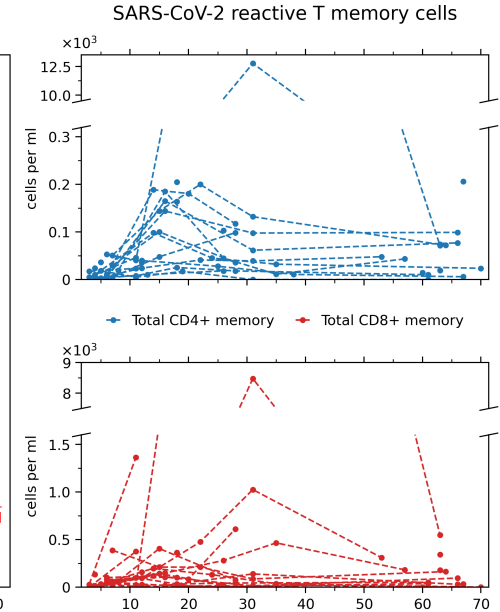
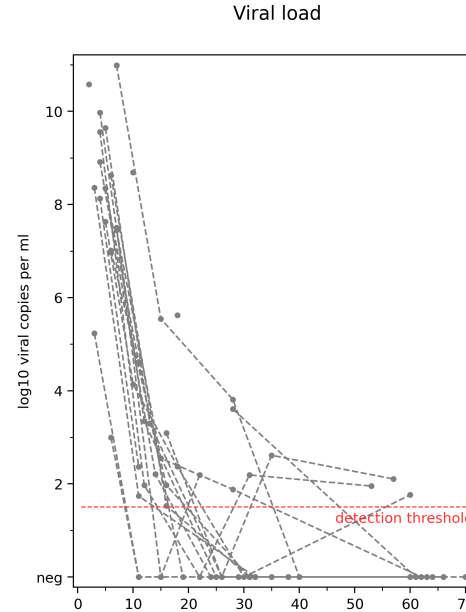
Biology is heterogeneous



Biology is heterogeneous

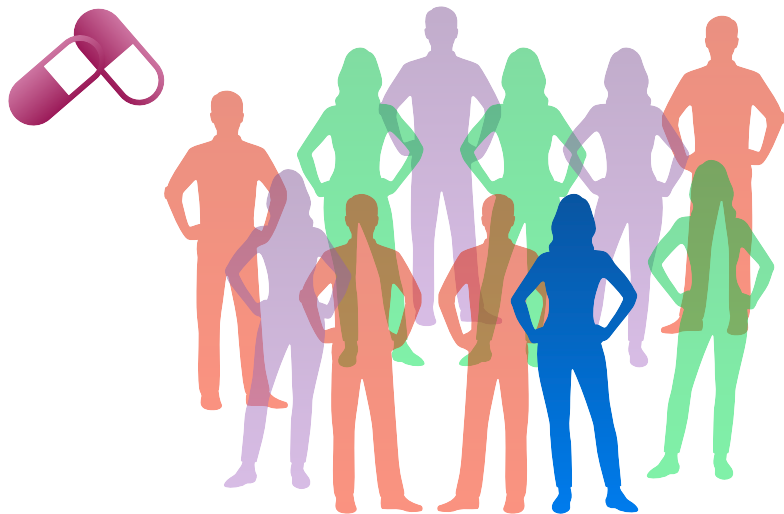


Biology is heterogeneous



How to describe heterogeneous populations?

Mixed Effects Modelling

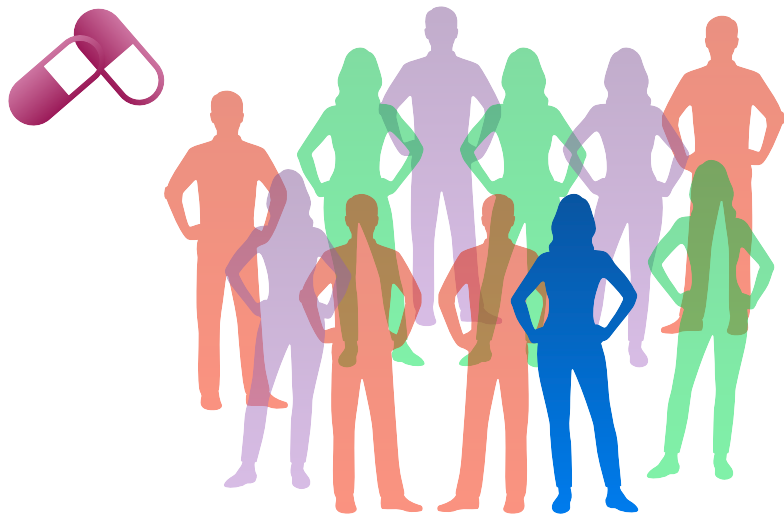


$M(\phi_i)$ generative model for individual i
 $\phi_i = \theta_1 + b_i$

- **Fixed Effects** θ_1
(e.g. mean effect of some drug)
- **Random Effects** $b_i \sim p_{\theta_2}(b)$
(variability of an effect)

Population Model $\phi \sim p(\phi \mid \theta)$
But observations are noisy

Mixed Effects Modelling




$M(\phi_i)$ generative model for individual i
 $\phi_i = \theta_1 + b_i$

- **Fixed Effects** θ_1
(e.g. mean effect of some drug)
- **Random Effects** $b_i \sim p_{\theta_2}(b)$
(variability of an effect)


Population Model $\phi \sim p(\phi \mid \theta)$
But observations are noisy

→ **all individuals** must be estimated **simultaneously**



Mixed Effects Modelling

- What is the likelihood of observing $y_i =$  using our model $M(\phi)$? $\rightarrow p(y_i \mid \phi)$

Mixed Effects Modelling

- What is the likelihood of observing $y_i =$  using our model $M(\phi)$? $\rightarrow p(y_i \mid \phi)$
- How are random effects distributed? $\rightarrow p(\phi \mid \theta)$

Mixed Effects Modelling

- What is the likelihood of observing $y_i =$  using our model $M(\phi)$? $\rightarrow p(y_i \mid \phi)$
- How are random effects distributed? $\rightarrow p(\phi \mid \theta)$
- What is the likelihood of observing $\mathcal{D} =$  ? $\rightarrow p(\mathcal{D} \mid \theta)$

Random effects are **unobserved quantities**

$$p(\mathcal{D} \mid \theta) = \prod_i \int p(y_i \mid \phi) p(\phi \mid \theta) d\phi$$

Current Estimation Methods $p(\mathcal{D} \mid \theta) = \prod_i \int p(y_i \mid \phi) p(\phi \mid \theta) d\phi$

1. “Estimate” the missing variables ϕ_i
2. Find best parameters of the overall model

Current Estimation Methods $p(\mathcal{D} \mid \theta) = \prod_i \int p(y_i \mid \phi) p(\phi \mid \theta) d\phi$

1. “Estimate” the missing variables ϕ_i
2. Find best parameters of the overall model

We can do this

- Deterministically \rightarrow biased
- Stochastically \rightarrow computationally intensive

Deterministic Approach

$$p(\mathcal{D} \mid \theta) = \prod_i \int p(y_i \mid \phi) p(\phi \mid \theta) d\phi$$

Linearisation-based or Laplace methods

1. For every individual i
 - A. Estimate mode $\hat{\phi}_i$ of $p(y_i \mid \phi)p(\phi \mid \theta_t)$ based on the best current θ_t
 - B. Use approximation to the integral based on the mode $\hat{\phi}_i$
2. Compute likelihood for population
3. Repeat until convergence

Deterministic Approach

$$p(\mathcal{D} \mid \theta) = \prod_i \int p(y_i \mid \phi) p(\phi \mid \theta) d\phi$$

Linearisation-based or Laplace methods

1. For every individual i
 - A. Estimate mode $\hat{\phi}_i$ of $p(y_i \mid \phi)p(\phi \mid \theta_t)$ based on the best current θ_t
 - B. Use approximation to the integral based on the mode $\hat{\phi}_i$
2. Compute likelihood for population
3. Repeat until convergence

Biased, **unreliable** and sensitive

(Pinheiro 1994, Comets & Mentré 2001)

Stochastic Approach

$$p(\mathcal{D} \mid \theta) = \prod_i \int p(y_i \mid \phi) p(\phi \mid \theta) d\phi$$

Stochastic expectation maximisation algorithm (SAEM)

1. Expectation step

$$Q(\theta \mid \theta_t) = \mathbb{E}_{\phi \sim p(\phi \mid y_i, \theta_t)} [\log p(y_i \mid \phi) + \log p(\phi \mid \theta)]$$

$\phi \sim p(\phi \mid y_i, \theta_t)$ samples must be generated with a MCMC procedure

Stochastic Approach

$$p(\mathcal{D} \mid \theta) = \prod_i \int p(y_i \mid \phi) p(\phi \mid \theta) d\phi$$

Stochastic expectation maximisation algorithm (SAEM)

1. Expectation step

$$Q(\theta \mid \theta_t) = \mathbb{E}_{\phi \sim p(\phi \mid y_i, \theta_t)} [\log p(y_i \mid \phi) + \log p(\phi \mid \theta)]$$

$\phi \sim p(\phi \mid y_i, \theta_t)$ samples must be generated with a MCMC procedure

2. Maximisation step

$$\theta_{t+1} = \operatorname{argmax}_{\theta} Q(\theta \mid \theta_t)$$

Stochastic Approach

$$p(\mathcal{D} \mid \theta) = \prod_i \int p(y_i \mid \phi) p(\phi \mid \theta) d\phi$$

Stochastic expectation maximisation algorithm (SAEM)

1. Expectation step

$$Q(\theta \mid \theta_t) = \mathbb{E}_{\phi \sim p(\phi \mid y_i, \theta_t)} [\log p(y_i \mid \phi) + \log p(\phi \mid \theta)]$$

$\phi \sim p(\phi \mid y_i, \theta_t)$ samples must be generated with a MCMC procedure

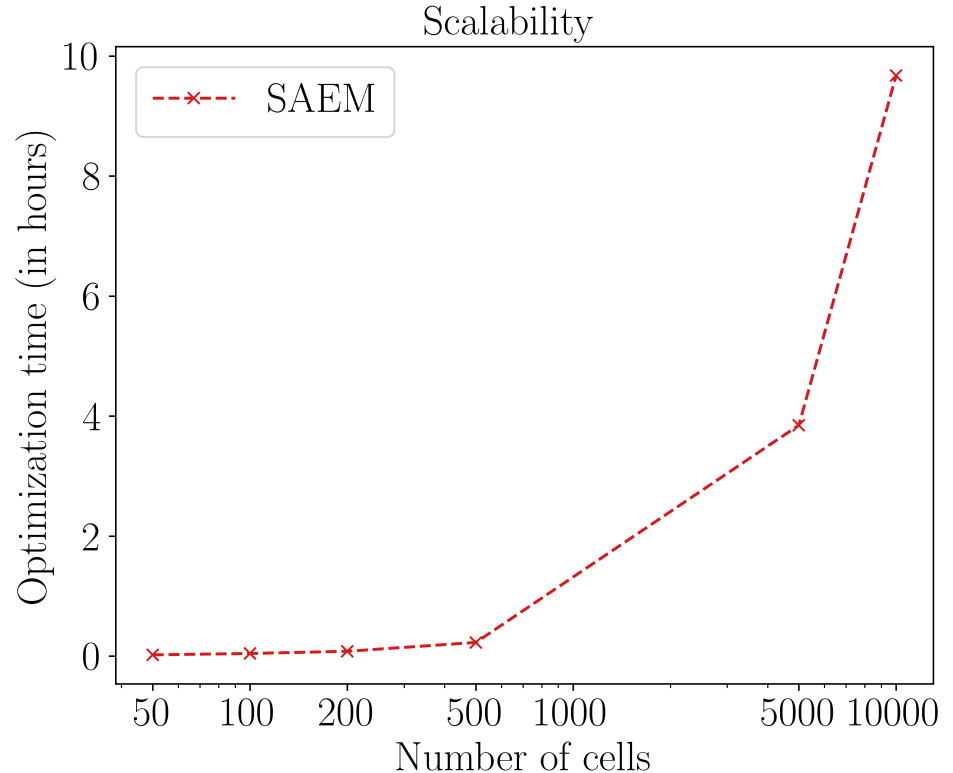
2. Maximisation step

$$\theta_{t+1} = \operatorname{argmax}_{\theta} Q(\theta \mid \theta_t)$$

Refinement in every step: $\hat{Q}(\theta \mid \theta_t) = (1 - \lambda_t) \hat{Q}(\theta \mid \theta_{t-1}) + \lambda_t Q(\theta \mid \theta_t)$

Stochastic Expectation Maximisation Algorithm

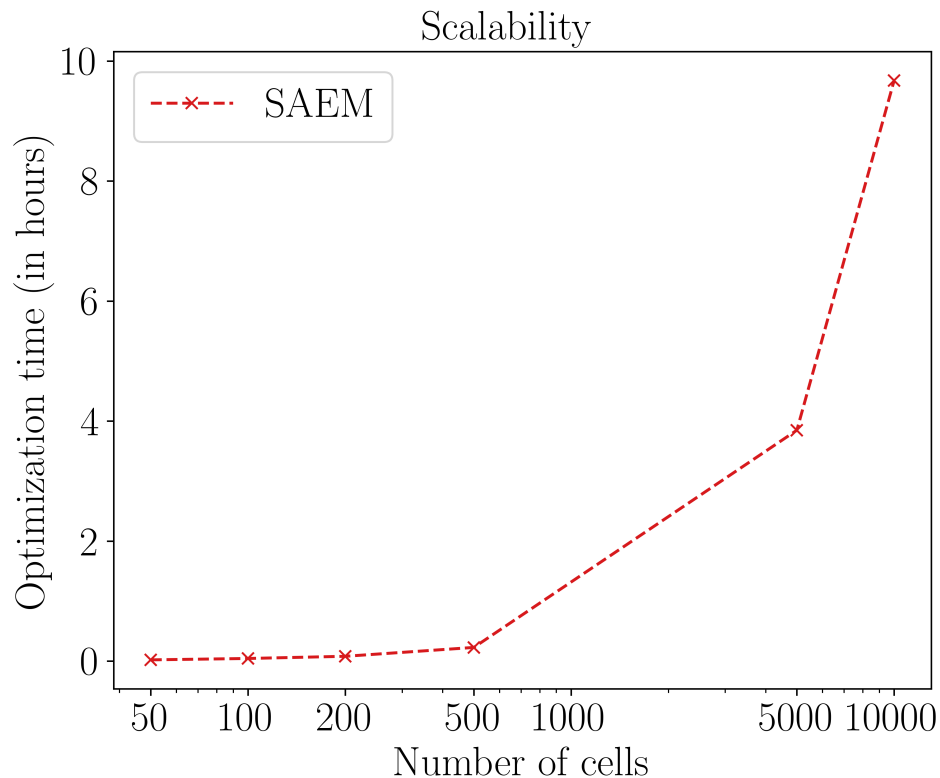
- State of the art method
- EM provides unbiased estimates
(Savic et. al. 2010)
- **Computationally demanding**
- Sensitive to initial values
(local minima)



Stochastic Expectation Maximisation Algorithm

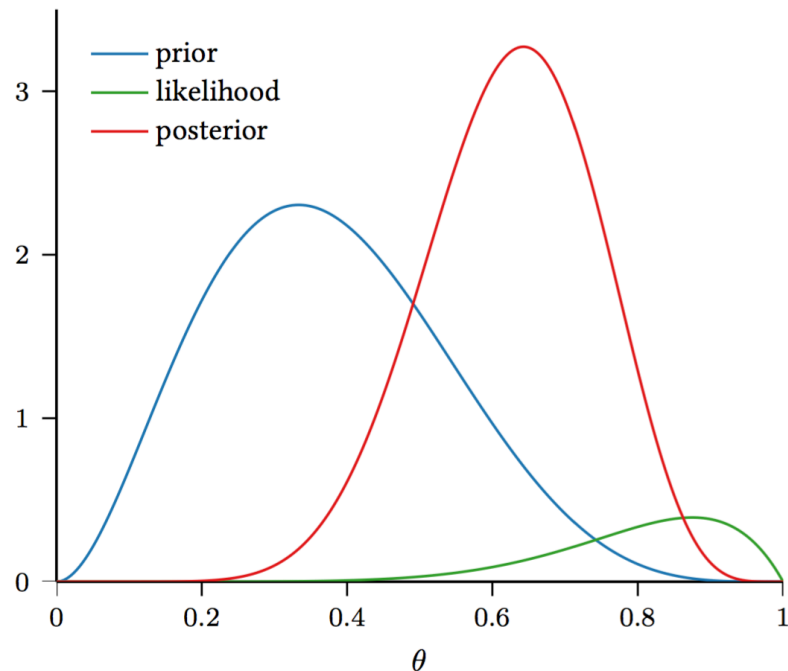
- State of the art method
- EM provides unbiased estimates
(Savic et. al. 2010)
- **Computationally demanding**
- Sensitive to initial values
(local minima)

Can we be more scalable?



Bayesian Inference

- Posterior distribution
 $p(\phi \mid y) \propto p(y \mid \phi)p(\phi)$
- Computed with methods like Markov chain Monte Carlo methods (**MCMC**)
- Computationally demanding



Bayesian Approach for Single Individual

- Back to $p(\mathcal{D} \mid \theta) = \prod_i \int p(y_i \mid \phi) p(\phi \mid \theta) d\phi$

Bayesian Approach for Single Individual

- Back to $p(\mathcal{D} \mid \theta) = \prod_i \int \frac{p(y_i)p(\phi \mid y_i)}{p(\phi)} p(\phi \mid \theta) d\phi$
- Rewrite marginal likelihood $p(y_i \mid \phi)$ in terms of the **individual-specific posterior distribution**

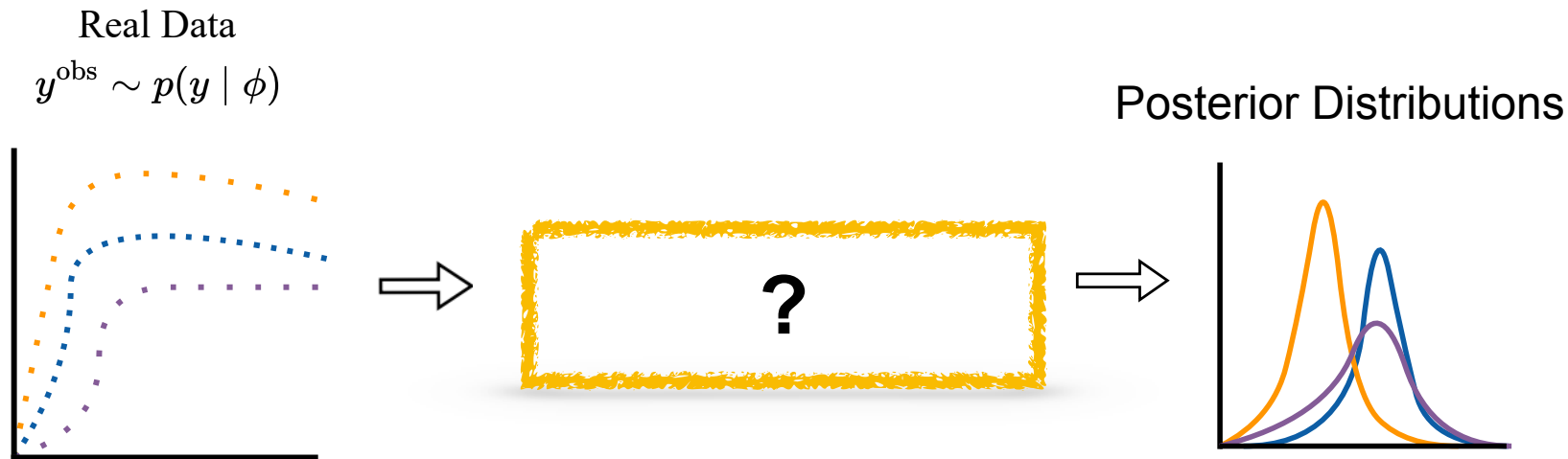
Bayesian Approach for Single Individual

- Back to $p(\mathcal{D} \mid \theta) = \prod_i \int \frac{p(y_i)p(\phi \mid y_i)}{p(\phi)} p(\phi \mid \theta) d\phi$
- Rewrite marginal likelihood $p(y_i \mid \phi)$ in terms of the **individual-specific posterior distribution**
- $\arg \max_{\theta} p(\mathcal{D} \mid \theta) = \arg \max_{\theta} \prod_i \mathbb{E}_{\phi \sim p(\phi \mid y_i)} \left[\frac{p(\phi \mid \theta)}{p(\phi)} \right]$

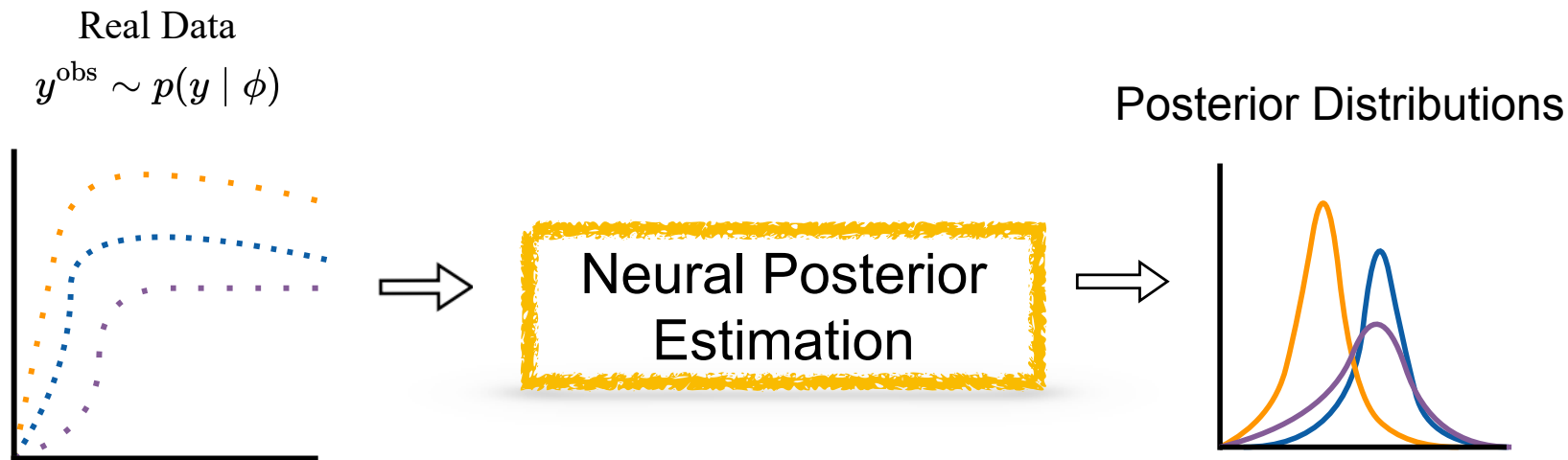
Bayesian Approach for Single Individual

- Back to $p(\mathcal{D} \mid \theta) = \prod_i \int \frac{p(y_i)p(\phi \mid y_i)}{p(\phi)} p(\phi \mid \theta) d\phi$
- Rewrite marginal likelihood $p(y_i \mid \phi)$ in terms of the **individual-specific posterior distribution**
- $$\arg \max_{\theta} p(\mathcal{D} \mid \theta) = \arg \max_{\theta} \prod_i \mathbb{E}_{\phi \sim p(\phi \mid y_i)} \left[\frac{p(\phi \mid \theta)}{p(\phi)} \right]$$
$$\approx \arg \max_{\theta} \prod_i \sum_{\phi_{i,j} \sim p(\phi \mid y_i)} \left[\frac{p(\phi_{i,j} \mid \theta)}{p(\phi_{i,j})} \right]$$
- Use Monte-Carlo approximation by sampling from **individual-specific posterior**

How do we get the individual-specific posterior in a scalable way?

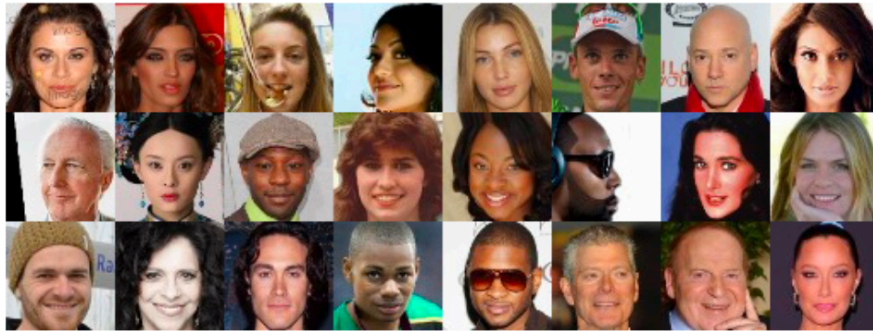


How do we get the individual-specific posterior in a scalable way?

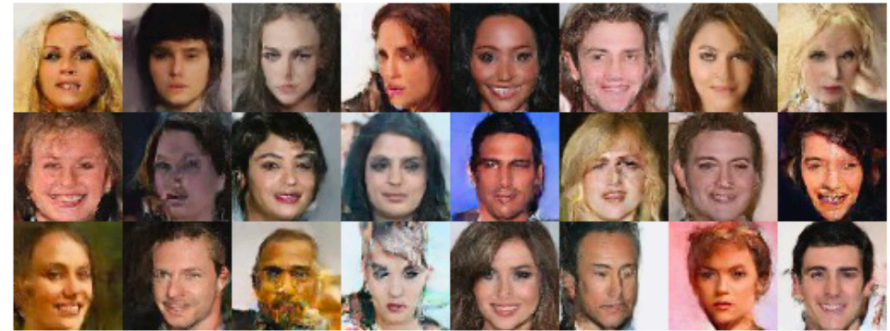


Neural Density Estimation

- Generative models
- Known for faithfully constructing e.g. portrait images

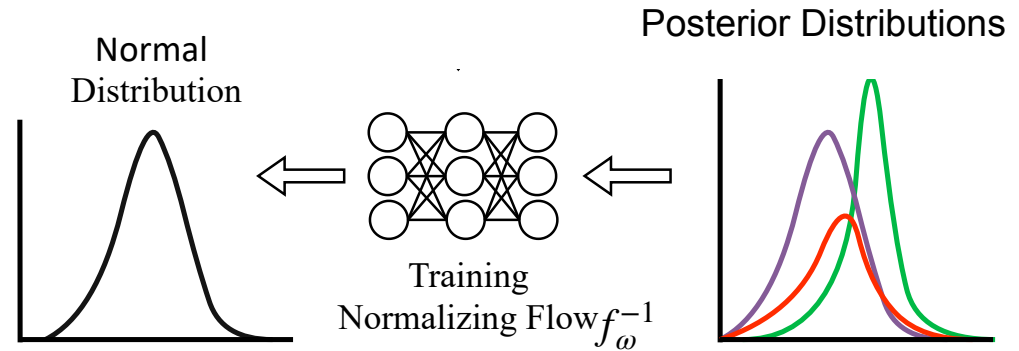


Training Data



Generated Data

Neural Posterior Estimation

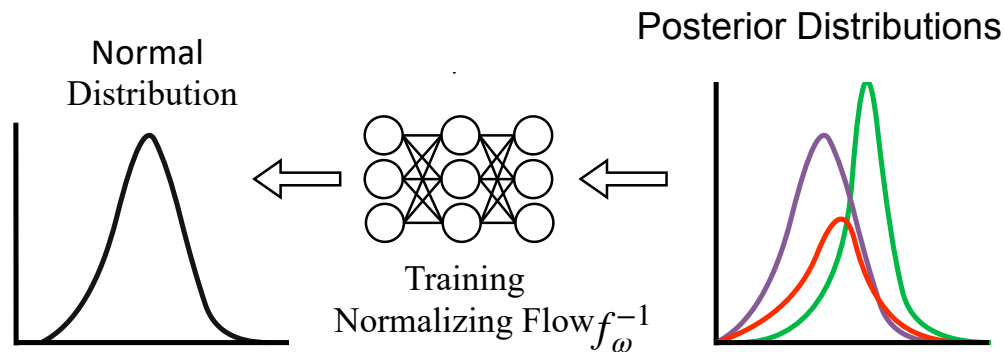


Neural Posterior Estimation

Conditional normalizing flows

- sample from latent distribution $z \sim p(z)$
- get sample from desired distribution by invertible mappings $f_i(z)$

$$x = f_0(z) \circ f_1(z) \circ \dots \circ f_n(z)$$



Neural Posterior Estimation

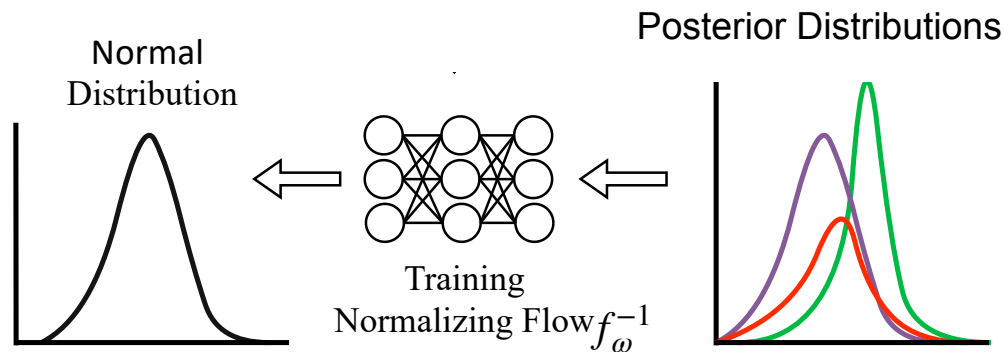
Conditional normalizing flows

- sample from latent distribution $z \sim p(z)$
- get sample from desired distribution by invertible mappings $f_i(z)$

$$x = f_0(z) \circ f_1(z) \circ \dots \circ f_n(z)$$

- eval density by

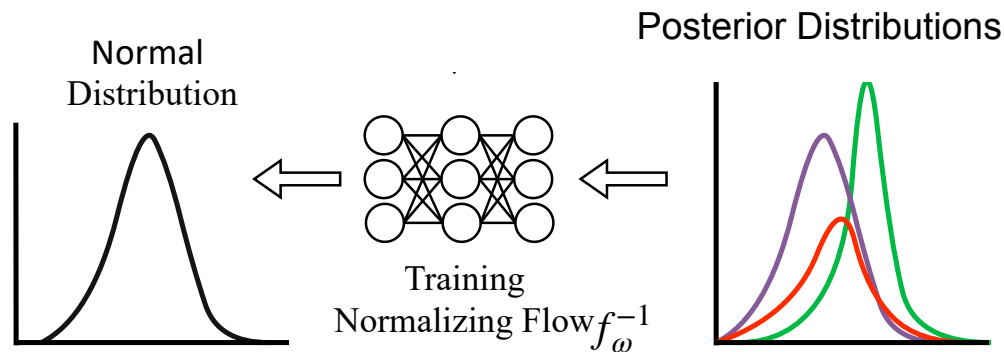
$$p(x) = p(z) | \det J_{f^{-1}} |$$



Neural Posterior Estimation

Conditional normalizing flows

- sample from latent distribution $z \sim p(z)$
- get sample from desired distribution by invertible mappings $f_i(z)$
$$x = f_0(z) \circ f_1(z) \circ \dots \circ f_n(z)$$
- eval density by
$$p(x) = p(z) | \det J_{f^{-1}} |$$
- f_ω parameterized by invertible neural networks



Example of Invertible Neural Networks

- **affine coupling layers**, split input $u = (u_1, u_2)$, s_i, t_i can be any neural network

$$v_1 = u_1 \odot \exp(s_1(u_2)) + t_1(u_2)$$

$$v_2 = u_2 \odot \exp(s_2(v_1)) + t_2(v_1)$$

Example of Invertible Neural Networks

- **affine coupling layers**, split input $u = (u_1, u_2)$, s_i, t_i can be any neural network

$$v_1 = u_1 \odot \exp(s_1(u_2)) + t_1(u_2)$$

$$v_2 = u_2 \odot \exp(s_2(v_1)) + t_2(v_1)$$

- **inversion**

$$u_2 = (v_2 - t_2(v_1)) \odot \exp(-s_2(v_1))$$

$$u_1 = (v_1 - t_1(u_2)) \odot \exp(-s_1(u_2))$$

Example of Invertible Neural Networks

- **affine coupling layers**, split input $u = (u_1, u_2)$, s_i, t_i can be any neural network

$$v_1 = u_1 \odot \exp(s_1(u_2)) + t_1(u_2)$$

$$v_2 = u_2 \odot \exp(s_2(v_1)) + t_2(v_1)$$

- **inversion**

$$u_2 = (v_2 - t_2(v_1)) \odot \exp(-s_2(v_1))$$

$$u_1 = (v_1 - t_1(u_2)) \odot \exp(-s_1(u_2))$$

- Jacobian of the affine transformation is a strictly upper or a lower triangular matrix
→ easy to compute

Learning a Normalizing Flow

Minimize **KL-divergence** between true posterior and approximation
for all possible data y

$$\arg \min_{\omega} \mathbb{E}_{p(y)} \left[\text{KL} \left(p(\phi \mid y) \parallel p_{\omega}(\phi \mid y) \right) \right]$$

Learning a Normalizing Flow

Minimize **KL-divergence** between true posterior and approximation for all possible data y

$$\begin{aligned} & \arg \min_{\omega} \mathbb{E}_{p(y)} \left[\text{KL} \left(p(\phi | y) \| p_{\omega}(\phi | y) \right) \right] \\ &= \arg \min_{\omega} \mathbb{E}_{p(y)} \left[\mathbb{E}_{p(\phi|y)} \left[\log p(\phi | y) - \log p_{\omega}(\phi | y) \right] \right] \end{aligned}$$

Learning a Normalizing Flow

Minimize **KL-divergence** between true posterior and approximation for all possible data y

$$\begin{aligned} & \arg \min_{\omega} \mathbb{E}_{p(y)} \left[\text{KL} \left(p(\phi | y) \| p_{\omega}(\phi | y) \right) \right] \\ &= \arg \min_{\omega} \mathbb{E}_{p(y)} \left[\mathbb{E}_{p(\phi|y)} \left[\log p(\phi | y) - \log p_{\omega}(\phi | y) \right] \right] \\ &= \arg \max_{\omega} \mathbb{E}_{p(y)} \left[\mathbb{E}_{p(\phi|y)} \left[\log p_{\omega}(\phi | y) \right] \right] \end{aligned}$$

Learning a Normalizing Flow

Minimize **KL-divergence** between true posterior and approximation for all possible data y

$$\begin{aligned} & \arg \min_{\omega} \mathbb{E}_{p(y)} \left[\text{KL} \left(p(\phi | y) \| p_{\omega}(\phi | y) \right) \right] \\ &= \arg \min_{\omega} \mathbb{E}_{p(y)} \left[\mathbb{E}_{p(\phi|y)} \left[\log p(\phi | y) - \log p_{\omega}(\phi | y) \right] \right] \\ &= \arg \max_{\omega} \mathbb{E}_{p(y)} \left[\mathbb{E}_{p(\phi|y)} \left[\log p_{\omega}(\phi | y) \right] \right] \\ &= \arg \max_{\omega} \int \int p(y, \phi) \log p_{\omega}(\phi | y) dy d\phi \end{aligned}$$

Learning a Normalizing Flow

Minimize **KL-divergence** between true posterior and approximation
for all possible data y

$$\arg \max_{\omega} \iint p(y, \phi) \log p_{\omega}(\phi \mid y) dy d\phi$$

Learning a Normalizing Flow

Minimize **KL-divergence** between true posterior and approximation for all possible data y

$$\arg \max_{\omega} \iint p(y, \phi) \log p_{\omega}(\phi \mid y) dy d\phi$$

$$= \arg \max_{\omega} \iint p(y, \phi) (\log p(f_{\omega}(\phi, y)) + \log |\det J_{f_{\omega}}|) dy d\phi$$

Learning a Normalizing Flow

Minimize **KL-divergence** between true posterior and approximation for all possible data y

$$\arg \max_{\omega} \iint p(y, \phi) \log p_{\omega}(\phi \mid y) dy d\phi$$

$$= \arg \max_{\omega} \iint p(y, \phi) (\log p(f_{\omega}(\phi, y)) + \log |\det J_{f_{\omega}}|) dy d\phi$$

$$\approx \arg \min_{\omega} \frac{1}{M} \sum_{m=1}^M \left(\frac{\|f_{\omega}(\phi^{(m)}, y^{(m)})\|_2^2}{2} - \log |\det J_{f_{\omega}}| \right)$$

Learning a Normalizing Flow

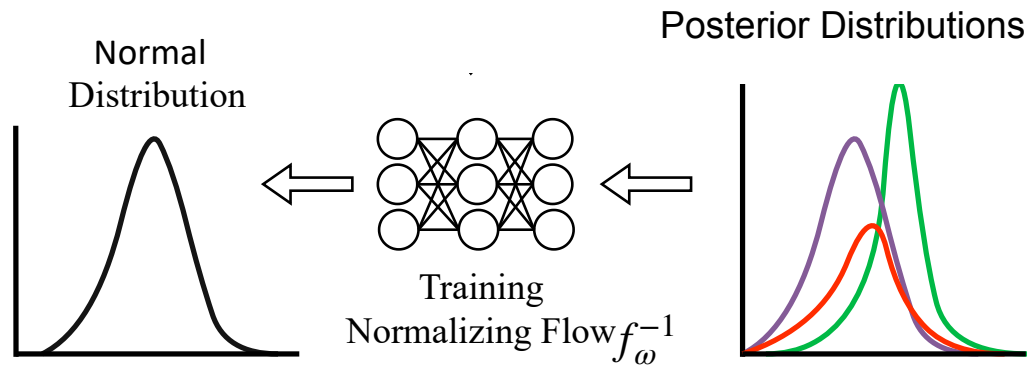
Minimize **KL-divergence** between true posterior and approximation for all possible data y

$$\begin{aligned} & \arg \max_{\omega} \iint p(y, \phi) \log p_{\omega}(\phi \mid y) dy d\phi \\ &= \arg \max_{\omega} \iint p(y, \phi) (\log p(f_{\omega}(\phi, y)) + \log |\det J_{f_{\omega}}|) dy d\phi \\ &\approx \arg \min_{\omega} \frac{1}{M} \sum_{m=1}^M \left(\frac{\|f_{\omega}(\phi^{(m)}, y^{(m)})\|_2^2}{2} - \log |\det J_{f_{\omega}}| \right) \end{aligned}$$

We need sample pairs $(\phi^{(m)}, y^{(m)})$!

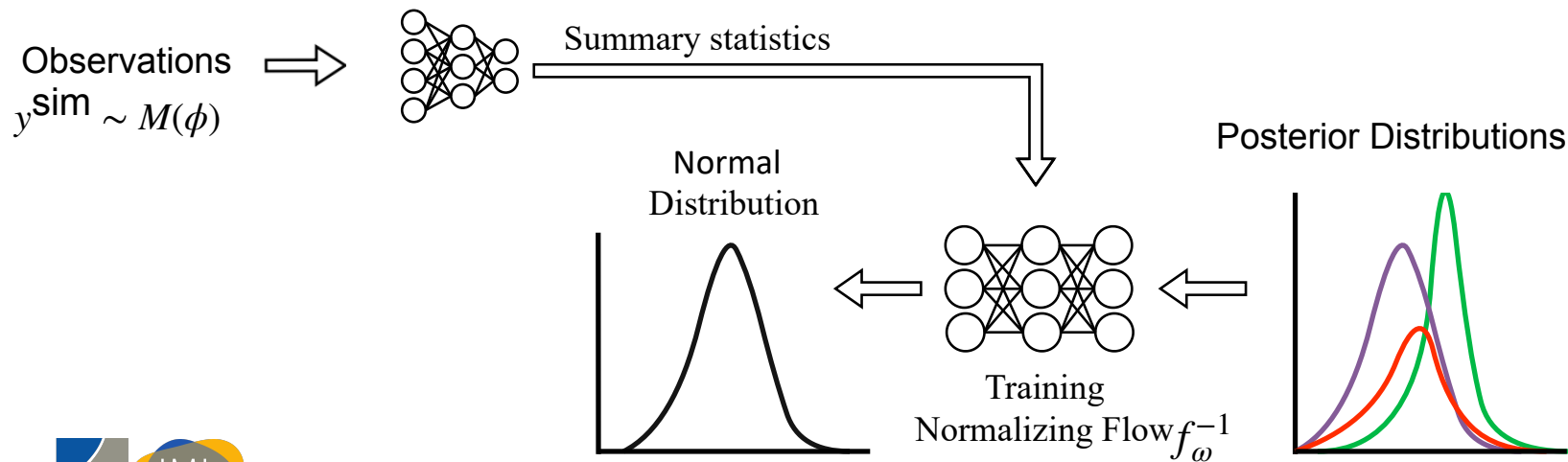
Neural Posterior Density Estimation

- Minimize **KL-divergence** between true posterior and approximation for all possible data y
- Generate samples ϕ from prior and produce simulations y^{sim}

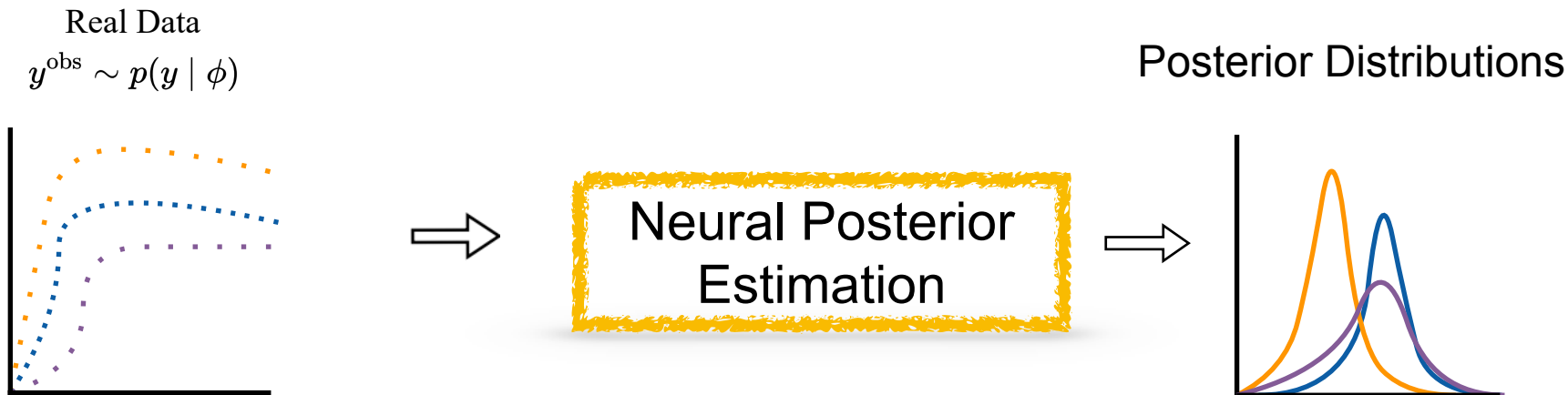


Neural Posterior Density Estimation

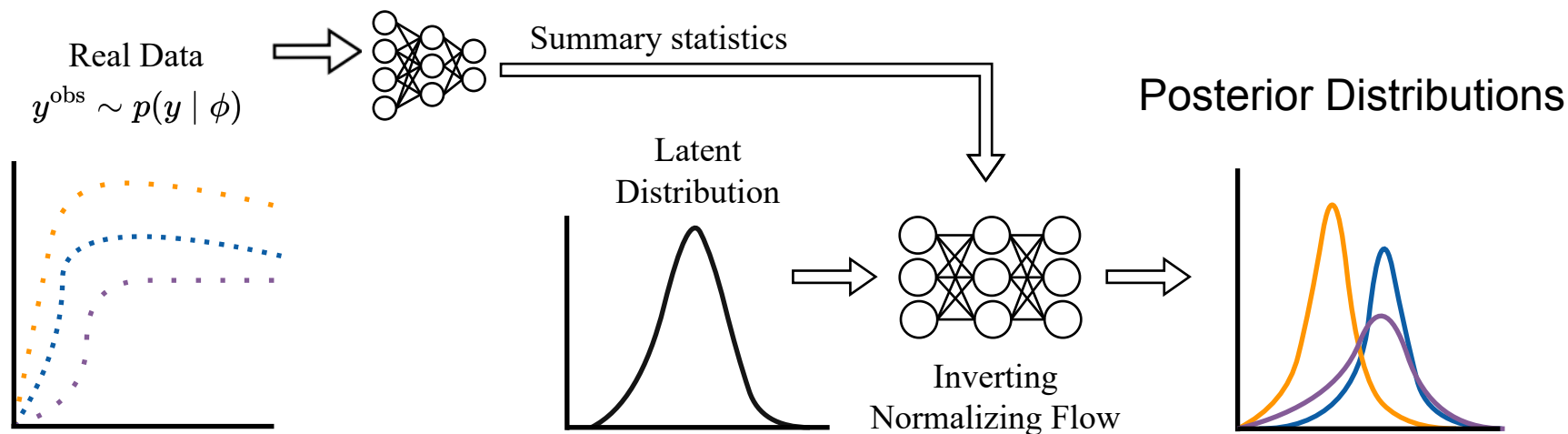
- Minimize **KL-divergence** between true posterior and approximation for all possible data y
- Generate samples ϕ from prior and produce simulations y^{sim}



How do we get the individual-specific posterior in a scalable way?



How do we get the individual-specific posterior in a scalable way?



Choices to be made

- Depth of invertible neural networks
- Type of summary networks
- Duration of training (early stopping)
- Loss function (e.g. Wasserstein instead of KL-divergence)

Choices to be made

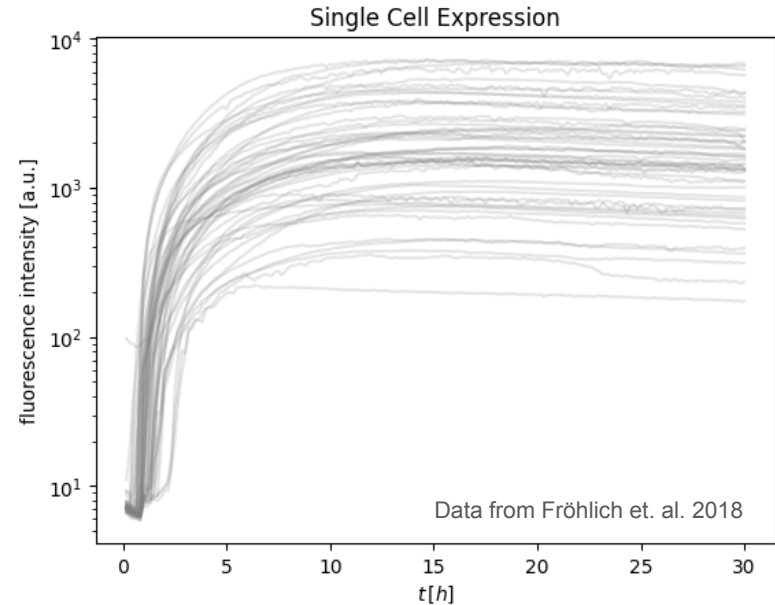
- Depth of invertible neural networks
- Type of summary networks
- Duration of training (early stopping)
- Loss function (e.g. Wasserstein instead of KL-divergence)

Other types of neural density estimators

- Conditional variational autoencoders (Kingma & Welling, 2022)
- Conditional generative adversarial neural networks (Wang & Ročková 2022)

Example Application

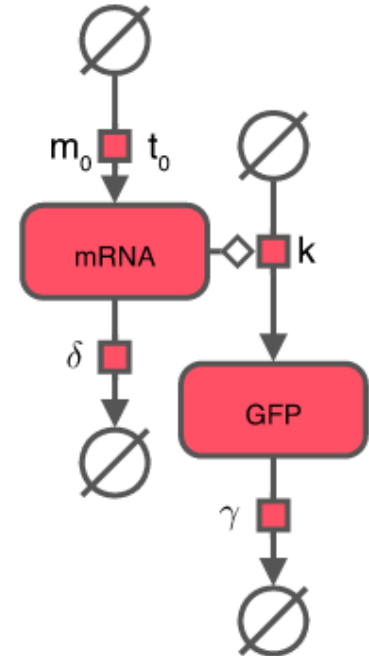
- living cells show molecular and phenotypic **differences at the single-cell level**
- mass cytometry can provide snapshots in thousands to **millions of cells**
- time-lapse microscopy measurements of **single-cells after transfection** with synthetic mRNA to assess mRNA lifetime



Amortized Approach to Non-Linear Mixed-Effects

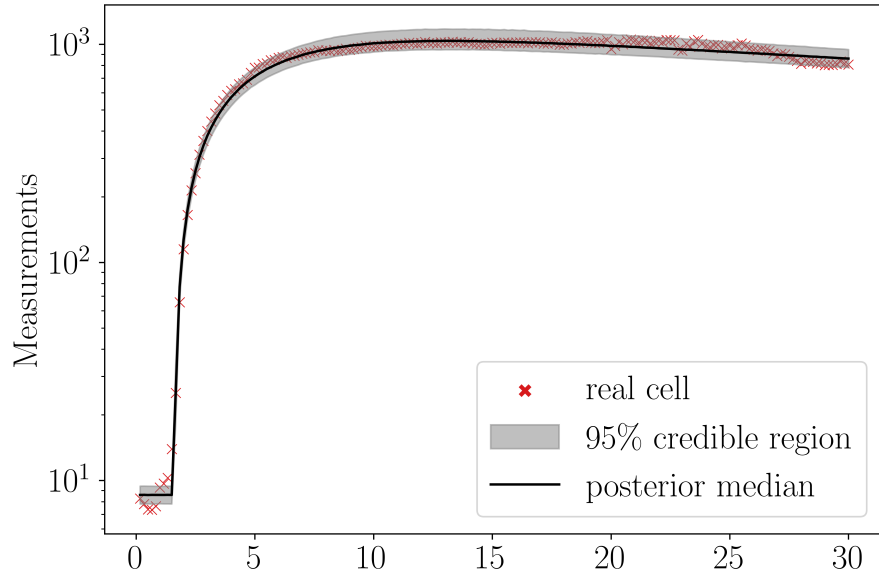
1. **Simulate** individual cells with generative model from prior
2. **Train** neural density estimator with simulations
3. **Infer** population parameter from real data

Generative ODE Model

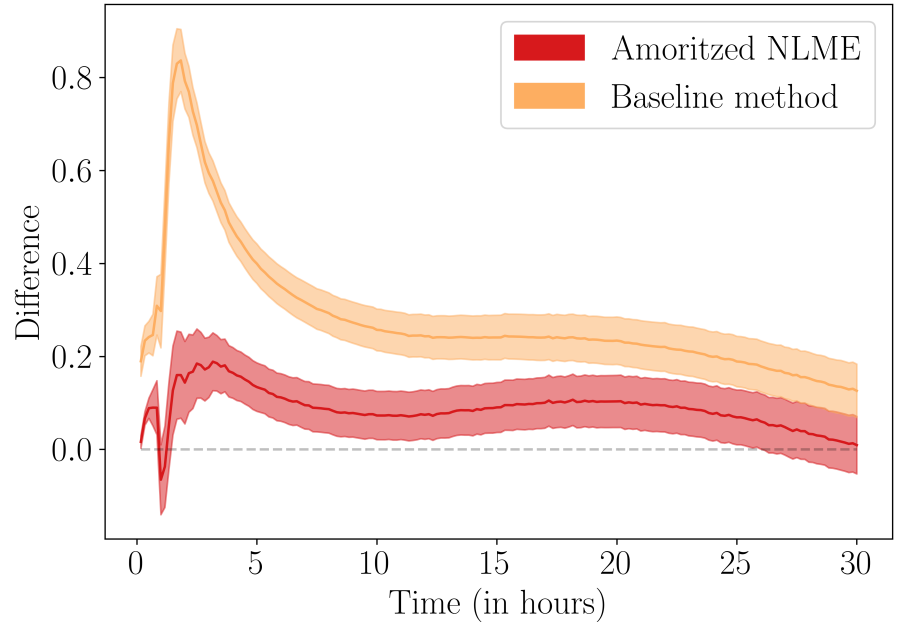


Inference Results

Neural Density Estimator



Population Estimation



Current Limitations

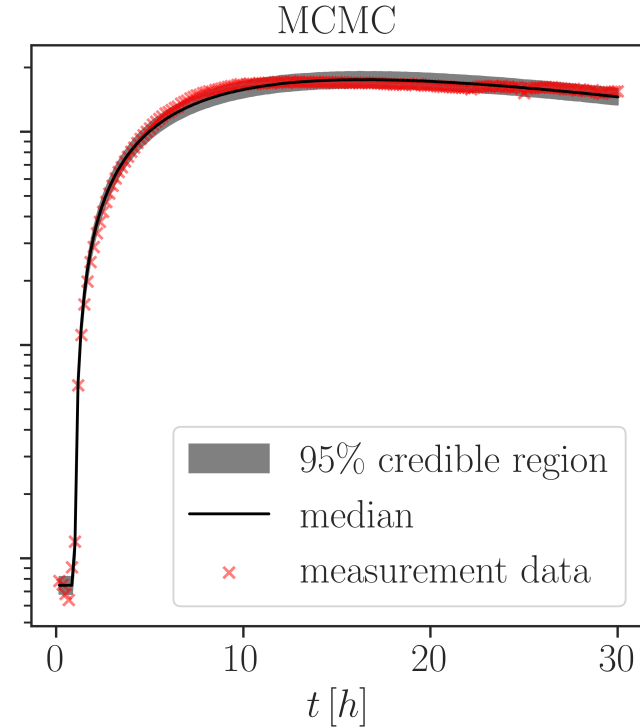
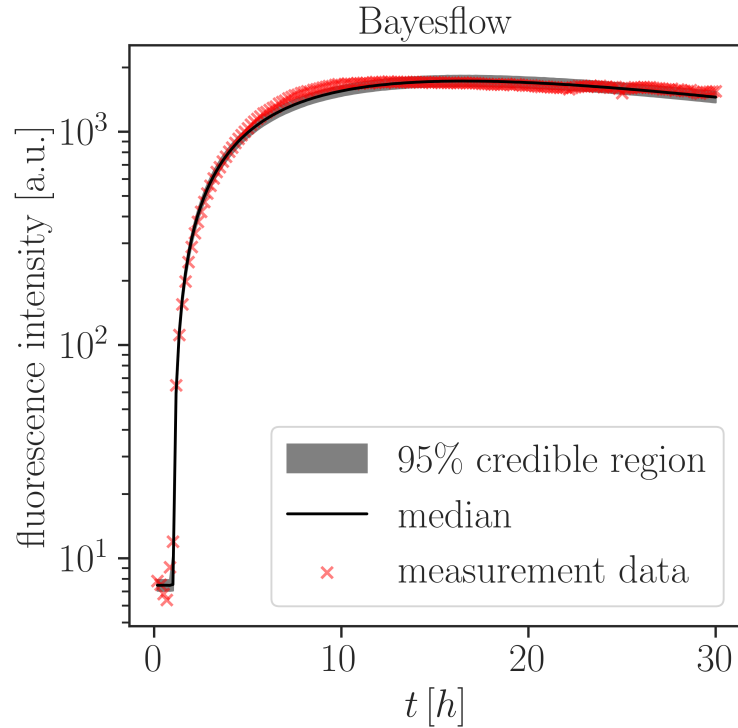
Neural Density Estimator

- Expressivity of the normalizing flow (Hagemann et. al. 2023)
- Misspecification of the model (Schmitt et. al. 2022)
- Non-conservative posteriors (Hermans et. al. 2022)

Population Estimation

- Robust inference of parameters with no variability

Comparison to MCMC



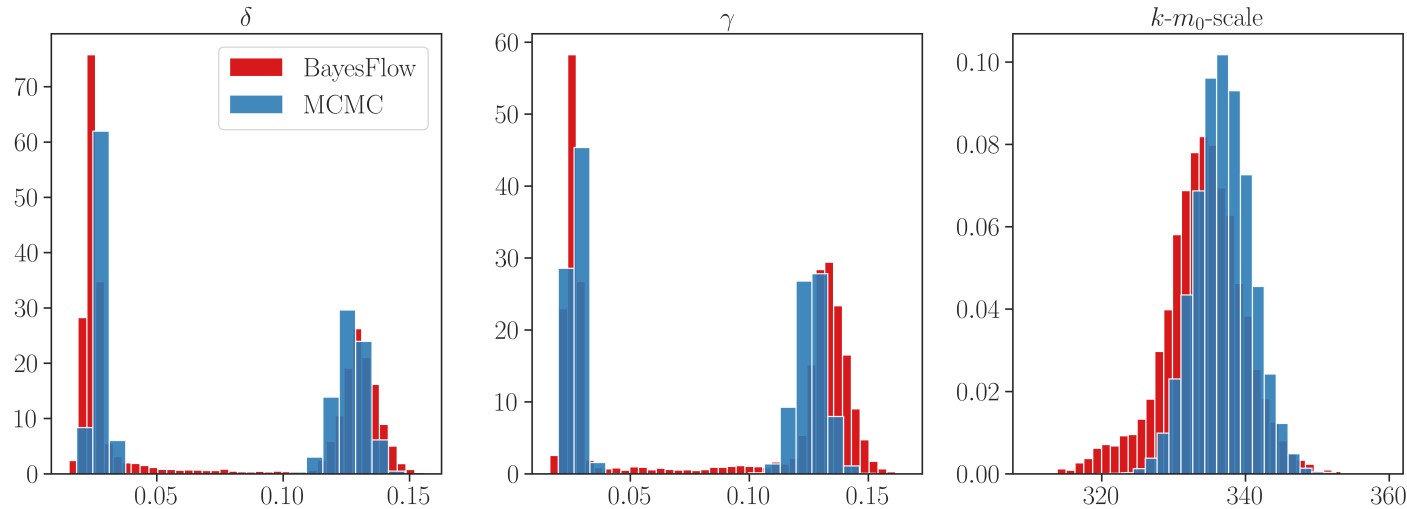
Comparison to MCMC

For ODE Model

Sampling Time for Single Cell

MCMC: 30min run time

BayesFlow: 1s



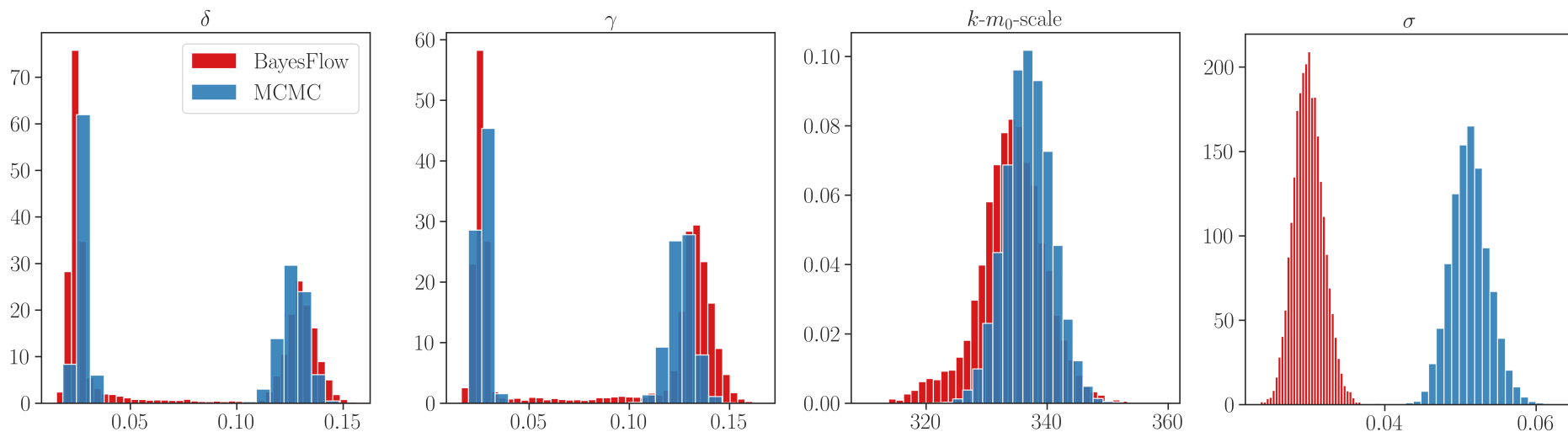
Comparison to MCMC

For ODE Model

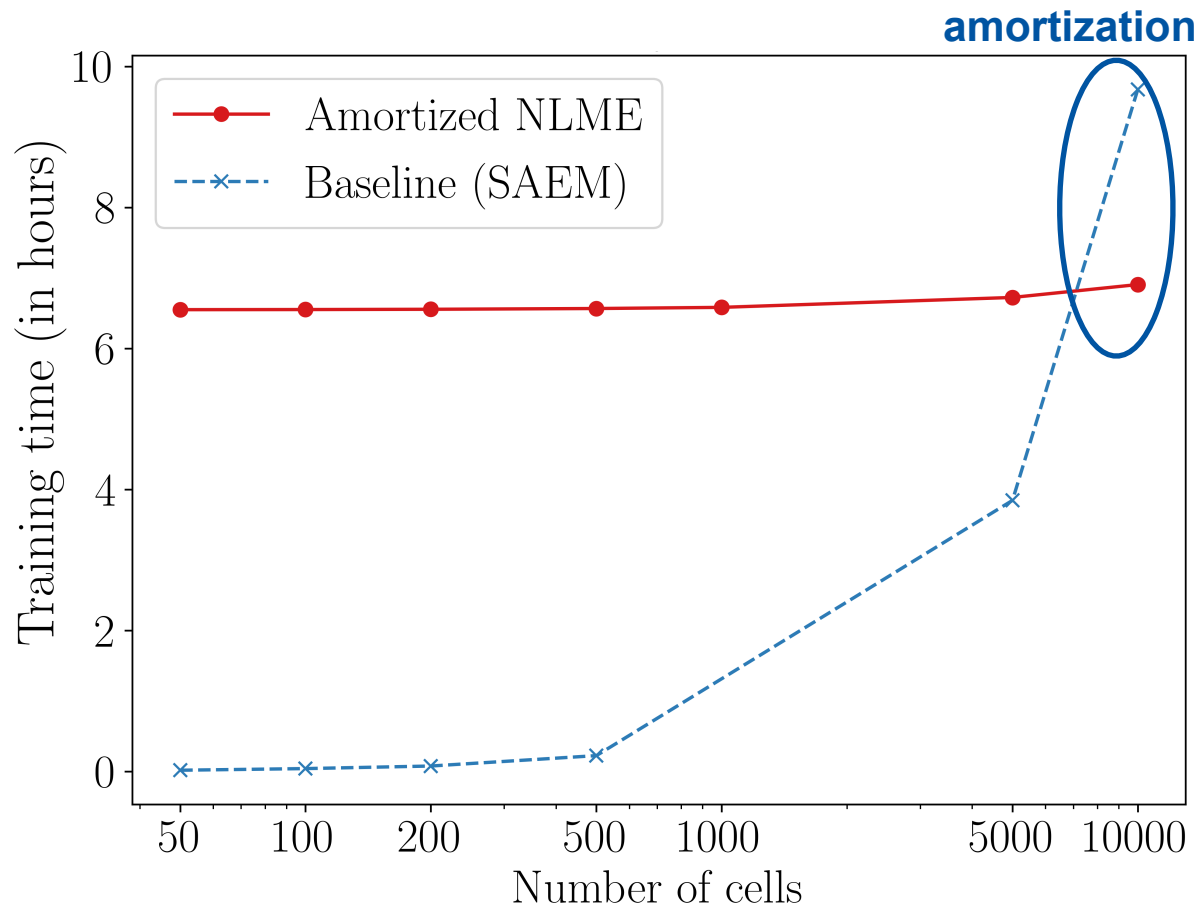
Sampling Time for Single Cell

MCMC: 30min run time

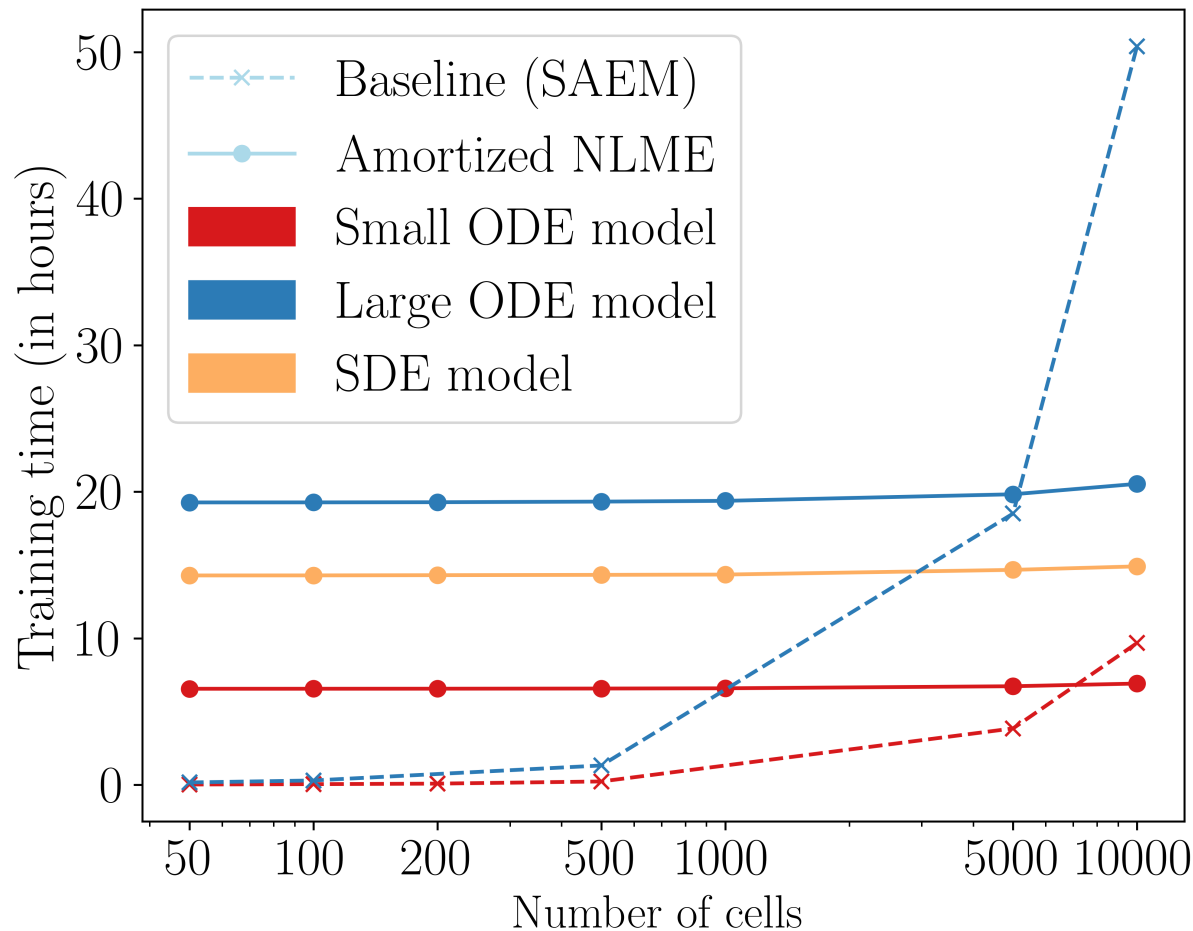
BayesFlow: 1s



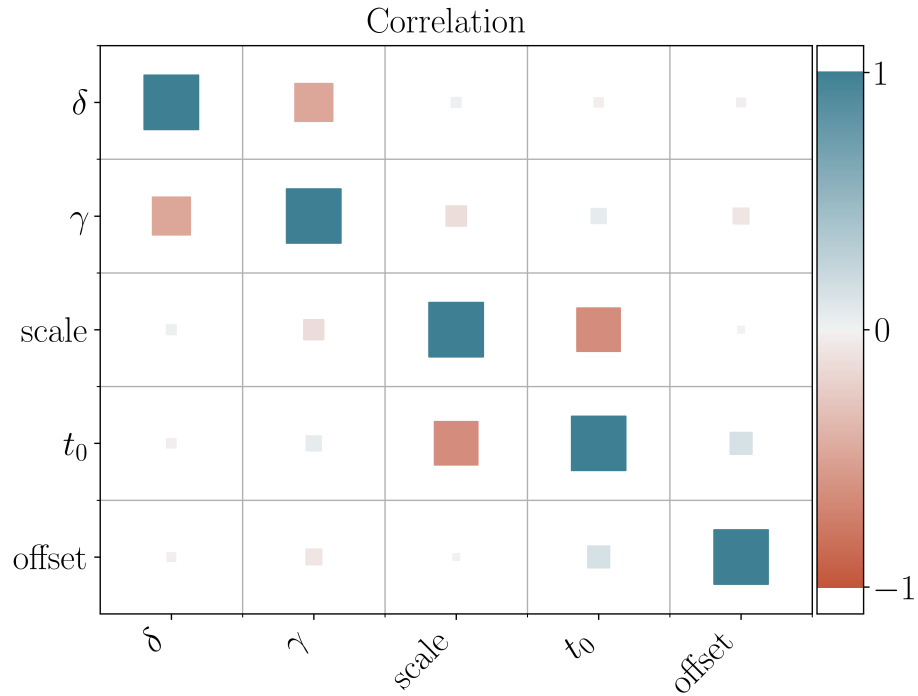
Scalable Method



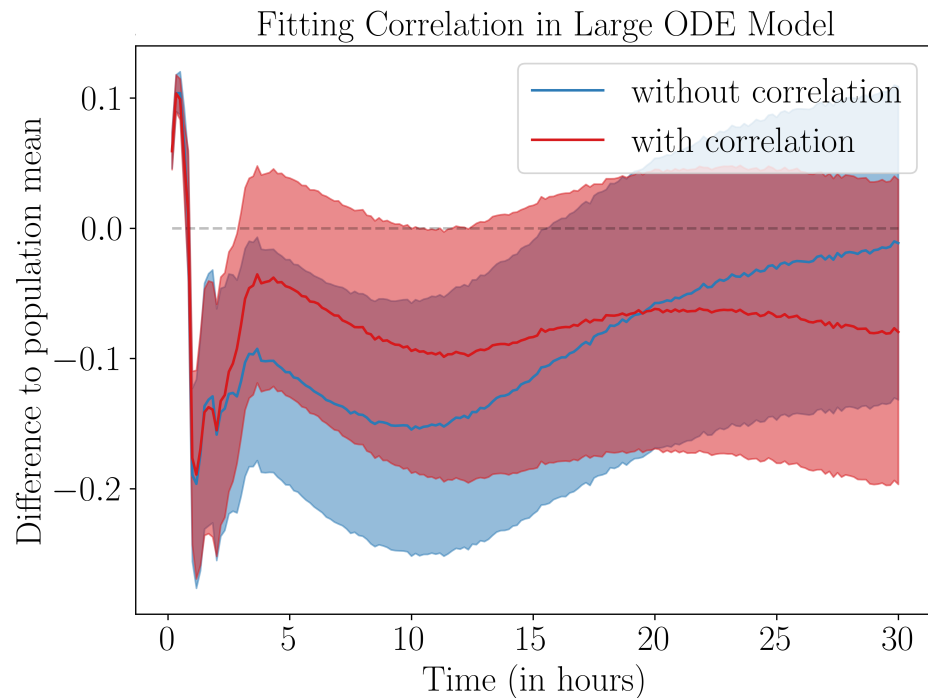
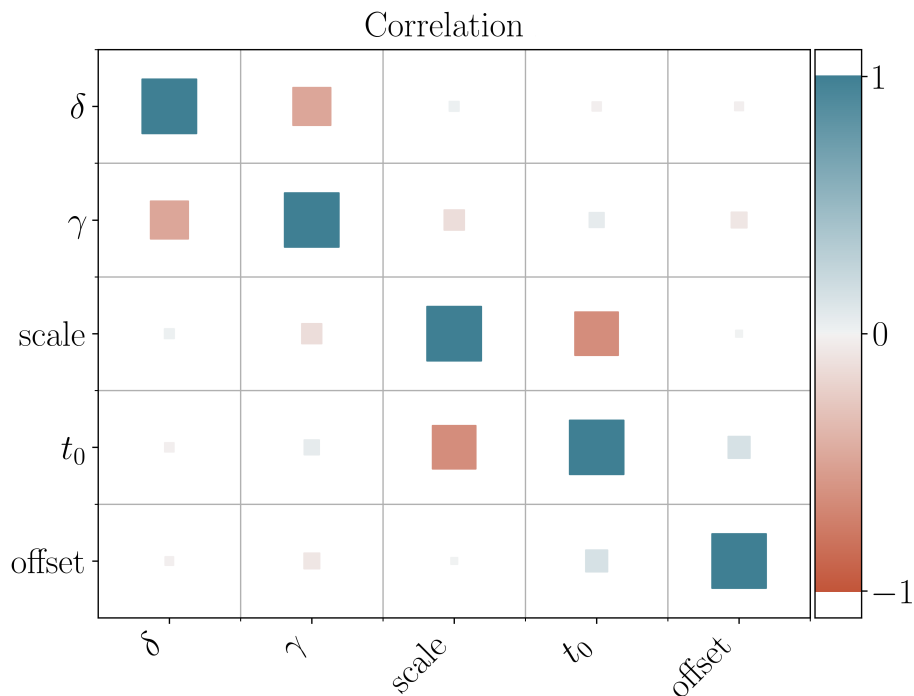
Scalable Method



Flexible Method

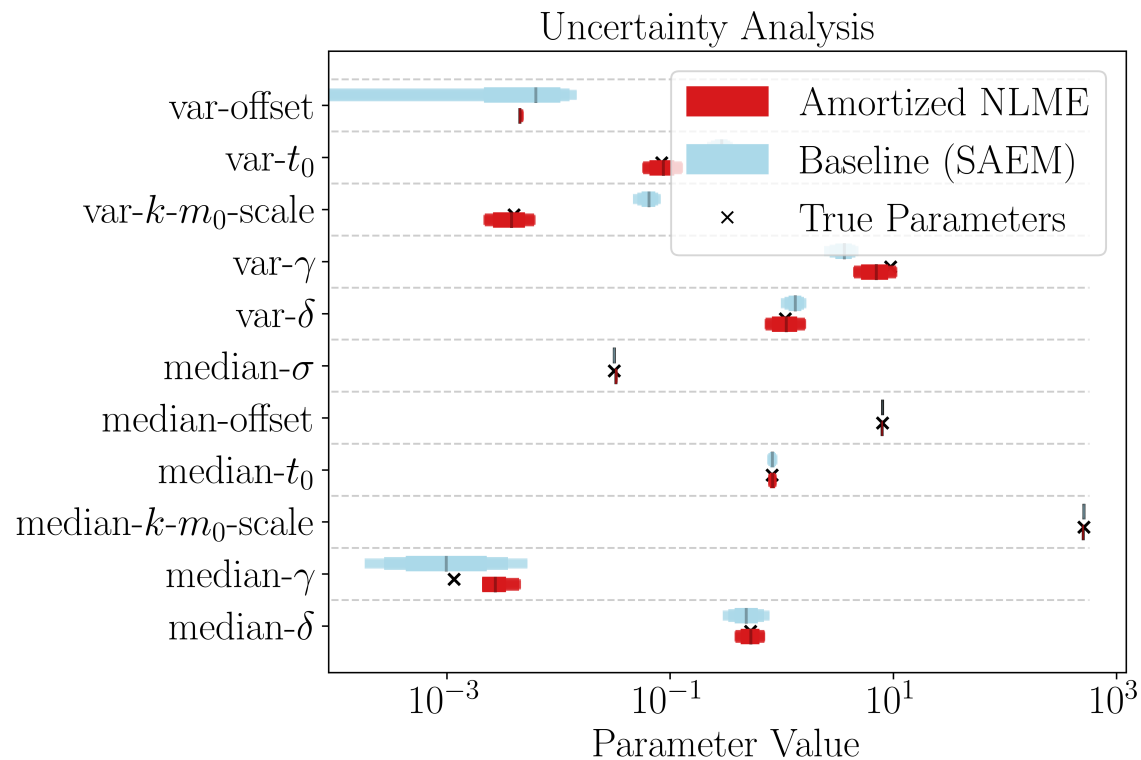


Flexible Method

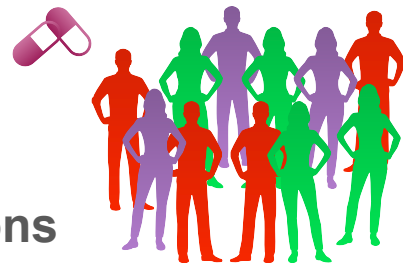


Basically no additional computational cost!

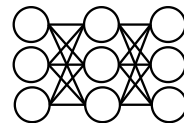
Easy Uncertainty Analysis



Conclusion



- Mixed effect models can describe **heterogenous populations**
- **Scalable** inference is computationally challenging
- Amortized approach by
 - Training a **neural posterior density estimator** on simulated data
 - **Cheap inference**
 - Flexible population model, uncertainty analysis, ...



Paper is coming soon :)

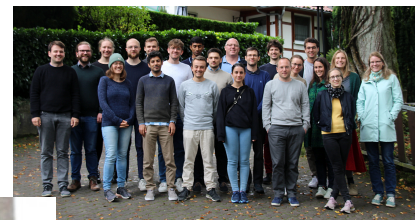


IRU MATHEMATICS
& LIFE SCIENCES

bigs
BONN INTERNATIONAL
GRADUATE SCHOOL
OF MATHEMATICS



Federal Ministry
of Education
and Research



AG Prof. Hasenauer



jonas.arruda@uni-bonn.de



yannik.schaelte@uni-bonn.de