Hands-on Simulation Based Inference

David Prelogović

Setup

- Go to https://github.com/dprelogo/SBI-tutorial
- Follow installation instructions

What is Inference?

Bayesian inference, obviously

Bayesian inference

- How to properly learn from the new data?
- Update prior knowledge about the parameters $P(\boldsymbol{\theta})$ by including data $\mathcal{D}$

$$\underbrace{P(\boldsymbol{\theta}|\mathcal{D})}_{\textit{posterior}} = \frac{\overbrace{P(\mathcal{D}|\boldsymbol{\theta})}^{\textit{likelihood}} \cdot \overbrace{P(\boldsymbol{\theta})}^{\textit{prior}}}{\underbrace{P(\mathcal{D})}_{\textit{evidence}}}$$

Bayesian inference

- How to properly learn from the new data?
- Update prior knowledge about the parameters $P(\boldsymbol{\theta})$ by including data $\mathcal{D}$

Good "physically-motivated" model

=

good prior

$$\textit{posterior}\ P(\boldsymbol{\theta}|\mathcal{D}) = \frac{\textit{likelihood}\ P(\mathcal{D}|\boldsymbol{\theta})\ \textit{prior}\ P(\boldsymbol{\theta})}{\textit{evidence}\ P(\mathcal{D})}$$

Bayesian inference

- How to properly learn from the new data?
- Update prior knowledge about the parameters $P(\boldsymbol{\theta})$ by including data $\mathcal{D}$

Good "physically-motivated" model
=
good prior

$$\textit{posterior}\ P(\boldsymbol{\theta}|\mathcal{D}) = \frac{\overbrace{P(\mathcal{D}|\boldsymbol{\theta})}^{\textit{likelihood}} \cdot \overbrace{P(\boldsymbol{\theta})}^{\textit{prior}}}{\underbrace{P(\mathcal{D})}_{\textit{evidence}}}$$

mostly ignored
(except for model selection)

Bayesian inference

- How to properly learn from the new data?
- Update prior knowledge about the parameters $P(\boldsymbol{\theta})$ by including data $\mathcal{D}$

???

Good "physically-motivated" model
=
good prior

$$\textit{posterior}\ P(\boldsymbol{\theta}|\mathcal{D}) = \frac{\textit{likelihood}\ P(\mathcal{D}|\boldsymbol{\theta}) \cdot \textit{prior}\ P(\boldsymbol{\theta})}{P(\mathcal{D})}$$

evidence

mostly ignored
(except for model selection)

Bayesian inference

- How to properly learn from the new data?
- Update prior knowledge about the parameters $P(\boldsymbol{\theta})$ by including data $\mathcal{D}$

???

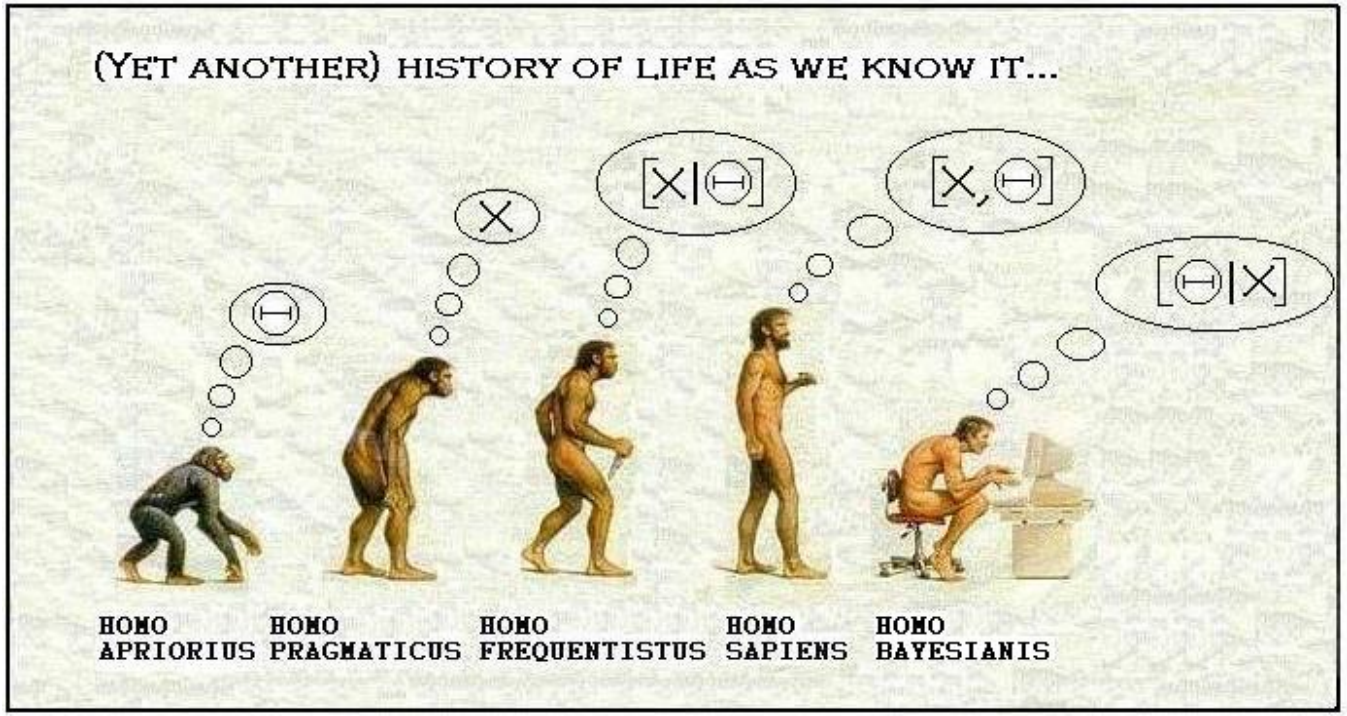
Good "physically-motivated" model
=
good prior

$$\textit{posterior}\ P(\boldsymbol{\theta}|\mathcal{D}) = \frac{\textit{likelihood}\ P(\mathcal{D}|\boldsymbol{\theta}) \cdot \textit{prior}\ P(\boldsymbol{\theta})}{\textit{evidence}\ P(\mathcal{D})}$$

mostly ignored
(except for model selection)

- Very often: data compressed to a set of summaries $\mathcal{D} \xrightarrow{f} \mathcal{S}$

What is Classical inference?



Classical inference

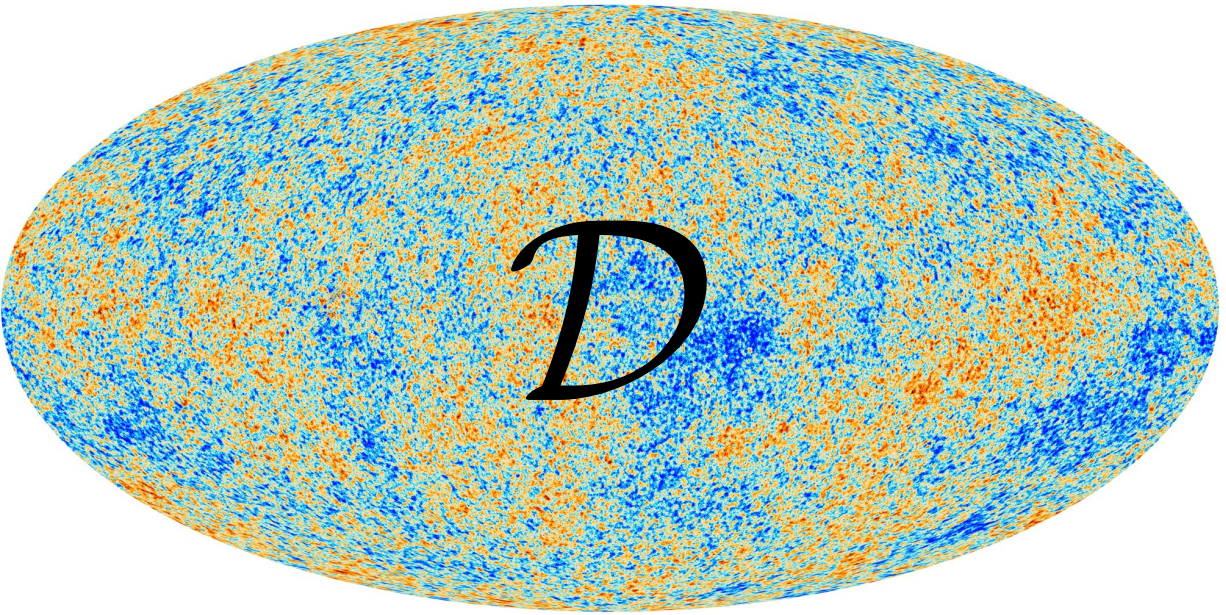
- Knowing your likelihood
- Eg.

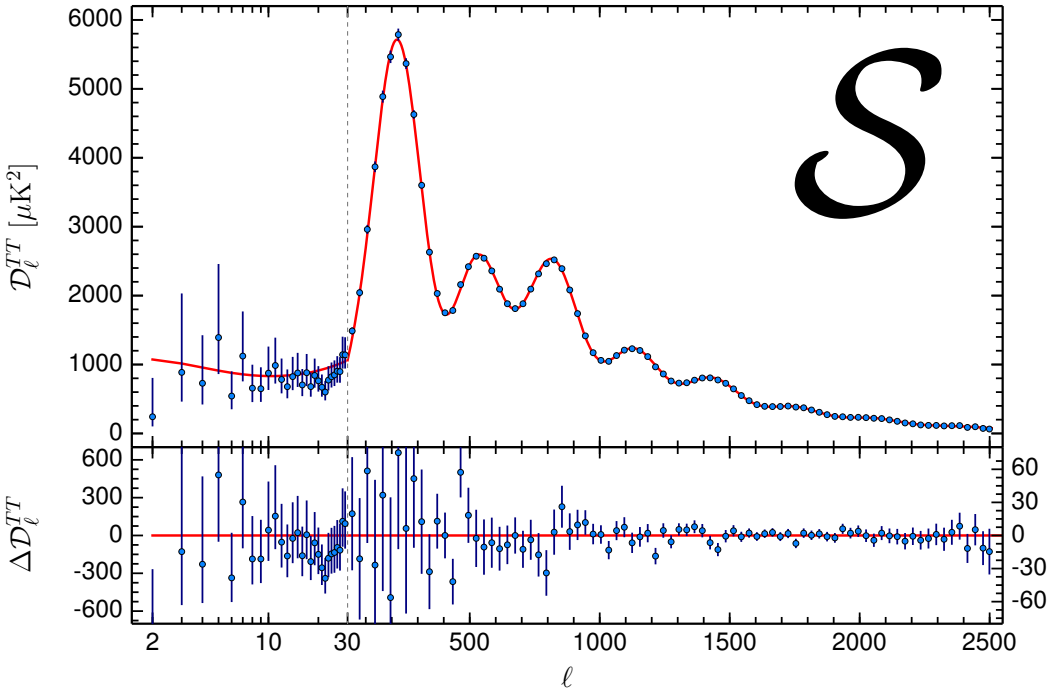
$$P(\mathcal{D}|\boldsymbol{\theta}) = \frac{1}{(2\pi)^{n/2}\sqrt{|\Sigma(\boldsymbol{\theta})|}} e^{-\frac{1}{2}(\mathbf{d}-\boldsymbol{\mu}(\boldsymbol{\theta}))^T \Sigma(\boldsymbol{\theta})^{-1}(\mathbf{d}-\boldsymbol{\mu}(\boldsymbol{\theta}))}$$

$$P(\mathcal{D}|\boldsymbol{\theta}) = \frac{1}{(2\pi)^{n/2}\sqrt{\Pi_i \sigma_i(\boldsymbol{\theta})^2}} e^{-\frac{1}{2}\sum_i (x_i-\mu_i(\boldsymbol{\theta}))^2/\sigma_i(\boldsymbol{\theta})^2}$$

-> i.e. χ^2

Example: CMB

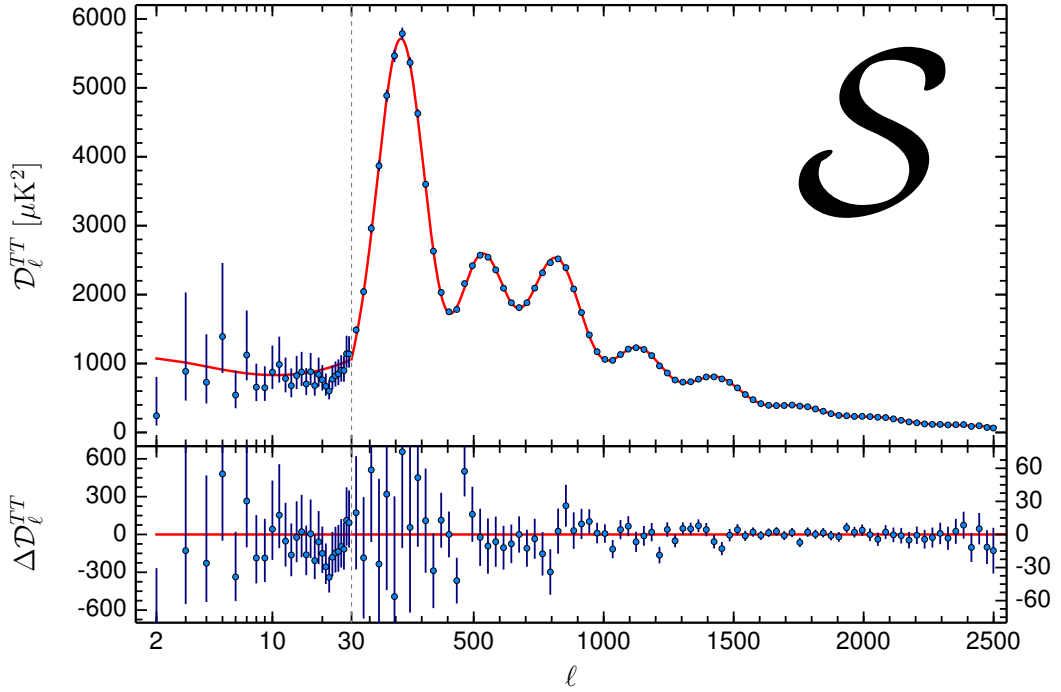




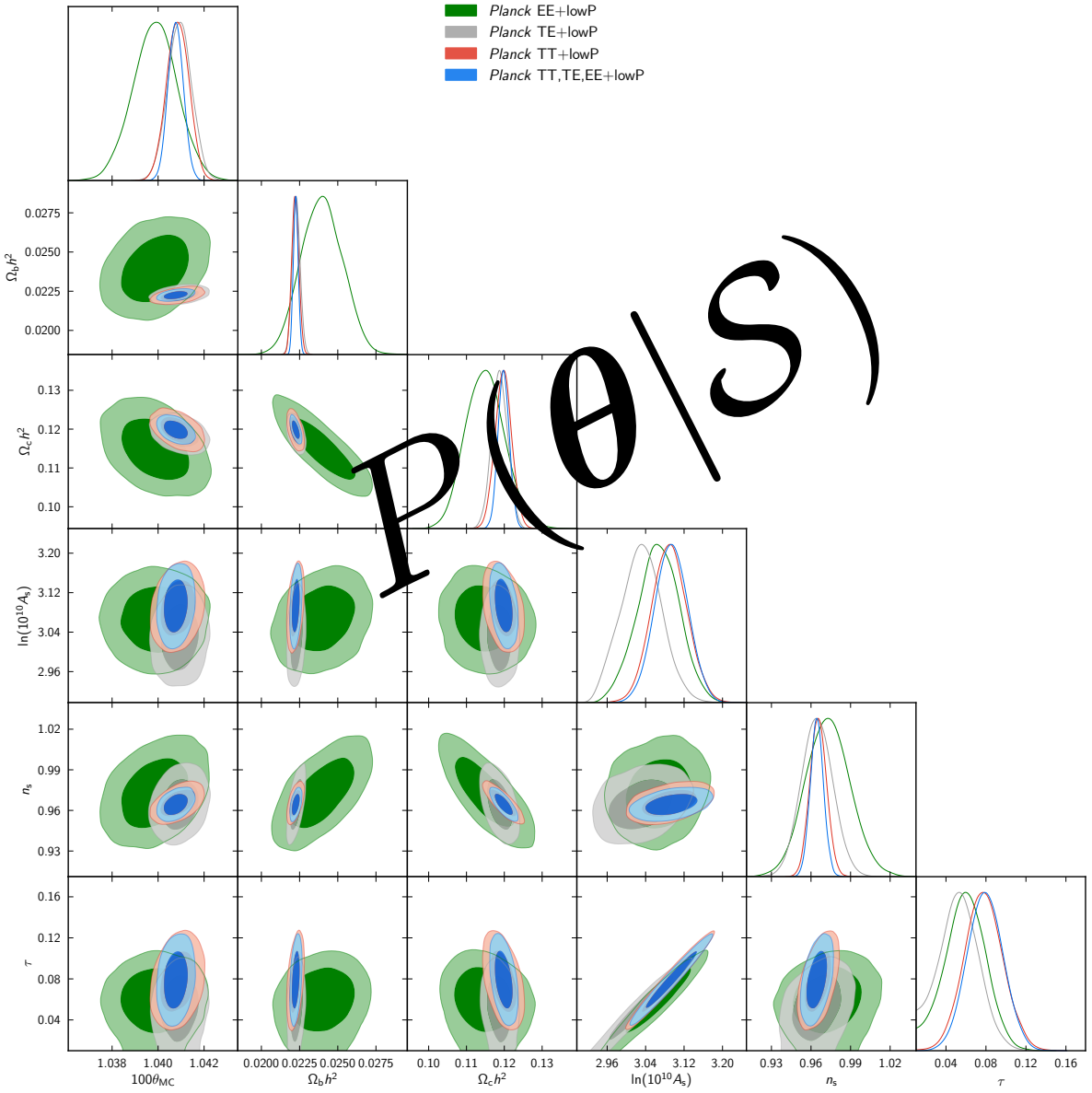
- Full sky map compressed to PS

Planck 2015

Example: CMB



S



$P(\theta|S)$

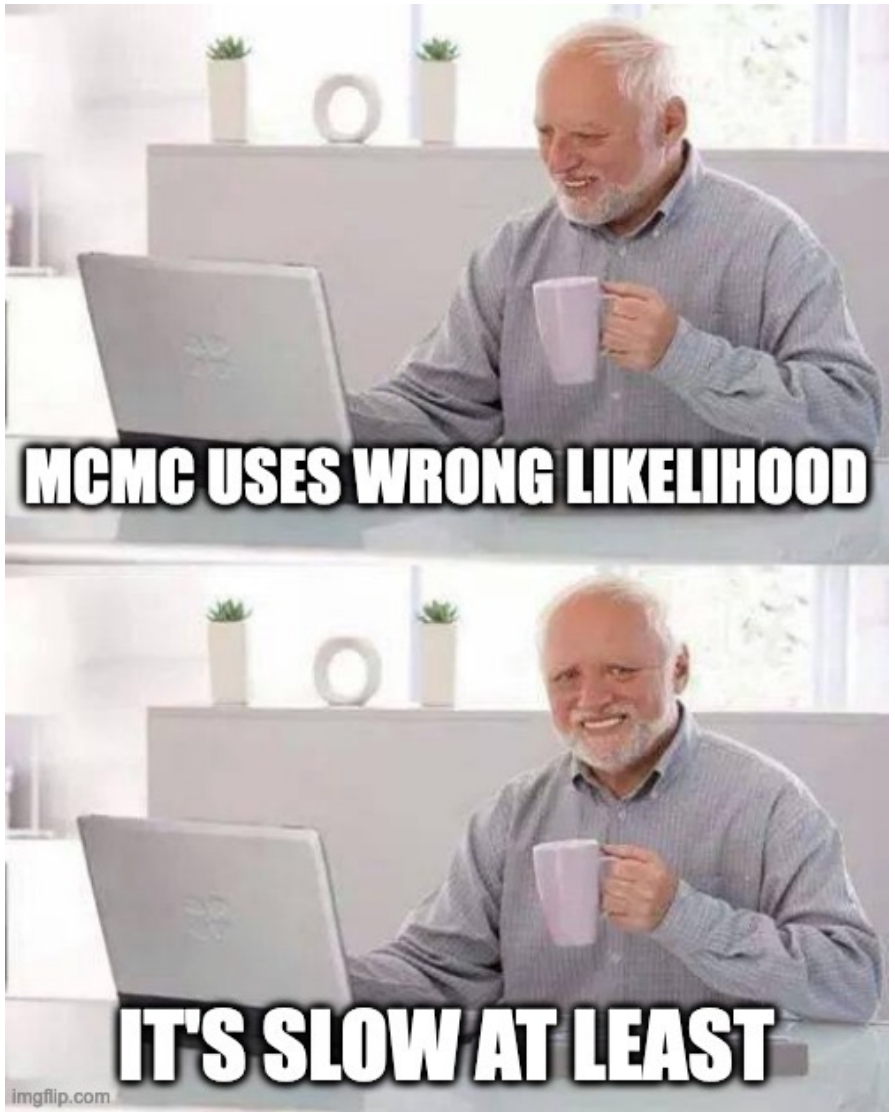
- Full sky map compressed to PS
- From it we infer the cosmology
  - Known compression, known likelihood

Planck 2015

How do we construct $P(\boldsymbol{\theta}|\mathcal{S})$

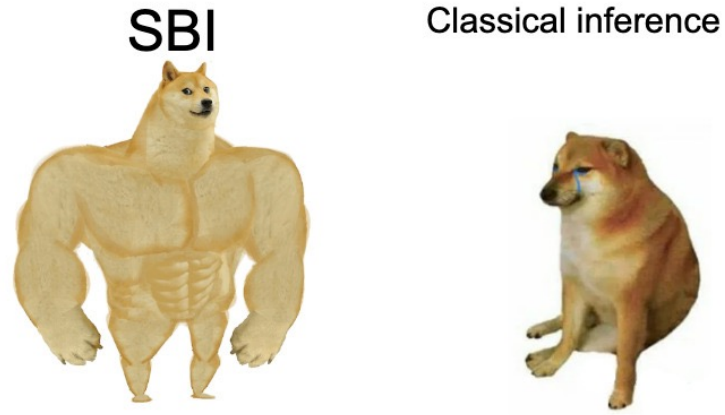
- https://chi-feng.github.io/mcmc-demo/app.html

Why isn’t classical inference enough?



Why isn’t classical inference enough?

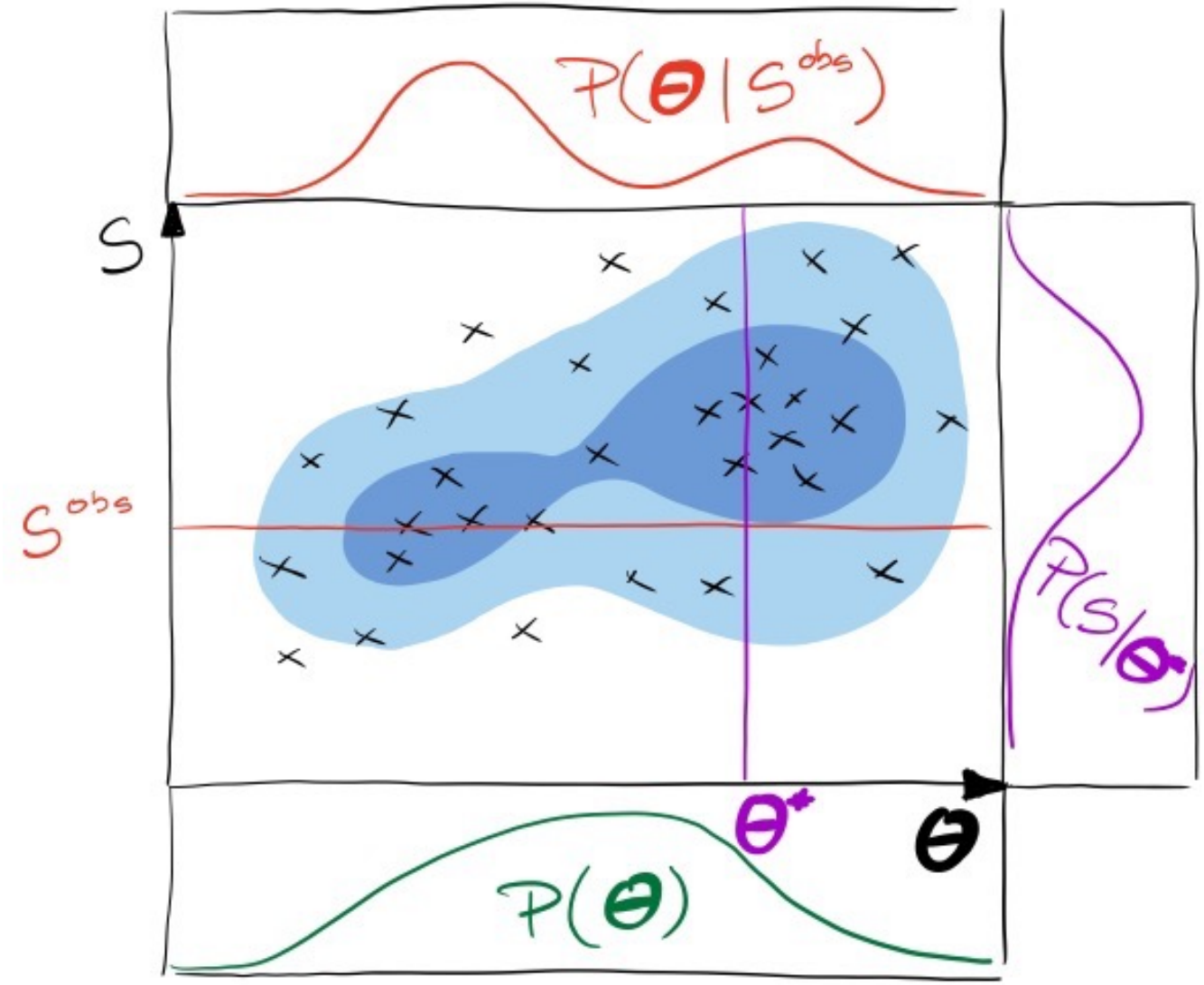
- Likelihood is not analytic or not tractable?
- Parameter space is huge?
- Maybe there’s only few important parameters, a lot of nuisance ones?
- New data comes in, should I run expensive inference from scratch?



What is Simulation Based Inference?

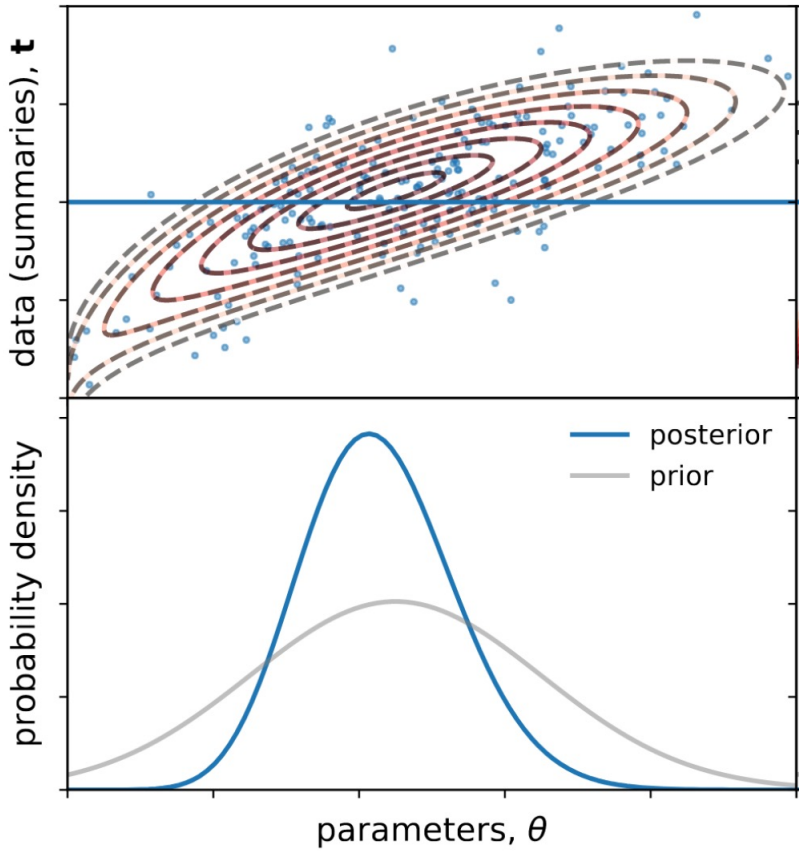
Simulation-Based Inference (SBI)

$$P(\boldsymbol{\theta}|\mathcal{D}) = \frac{\overbrace{P(\mathcal{D}|\boldsymbol{\theta})}^{\textit{likelihood}} \cdot \overbrace{P(\boldsymbol{\theta})}^{\textit{prior}}}{\underbrace{P(\mathcal{D})}_{\textit{evidence}}}$$



Credit: Tom Charnock

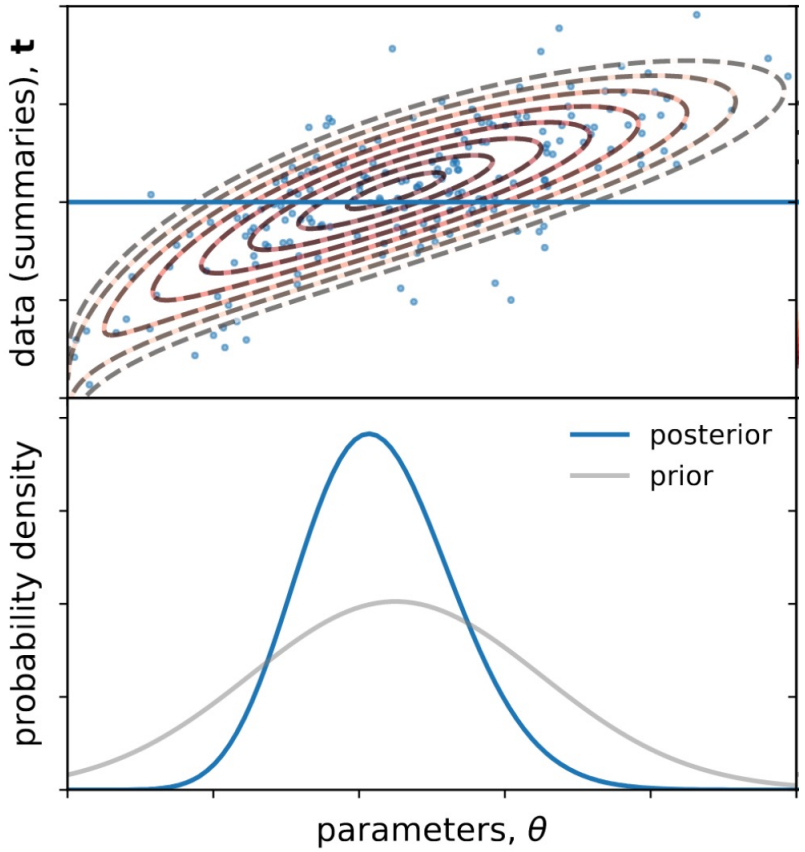
Simulation-Based Inference (SBI)



- Pull from the prior $\tilde{\theta} \sim p(\theta)$
- Pull from the likelihood $\tilde{t} \sim p(t|\tilde{\theta})$
  - Use simulator to simulate the data
    $$\tilde{d} = \text{simulator}(\tilde{\theta})$$
  - Optionally compress the data to summary
    $$\tilde{t} = \text{compressor}(\tilde{d})$$
- Repeat many times to construct a set
  $$\left\{(\tilde{t}_1, \tilde{\theta}_1), (\tilde{t}_2, \tilde{\theta}_2), \ldots, (\tilde{t}_N, \tilde{\theta}_N)\right\}$$
- It is assumed the simulator is perfect, i.e. it pulls from the actual likelihood
  $$p(d|\tilde{\theta})$$

Alsing+2019

Approximate Bayesian Computation (ABC)



- Define an "ε-ball" in the data-space

$$B_\epsilon(\mathbf{x}_o) = \{\mathbf{x} : \|\mathbf{x} - \mathbf{x}_o\| \leq \epsilon\}$$

- Likelihood is then approximated by

$$p(\mathbf{x}_o \,|\, \boldsymbol{\theta}) \approx \frac{\Pr(\|\mathbf{x} - \mathbf{x}_o\| \leq \epsilon \,|\, \boldsymbol{\theta})}{|B_\epsilon(\mathbf{x}_o)|}$$

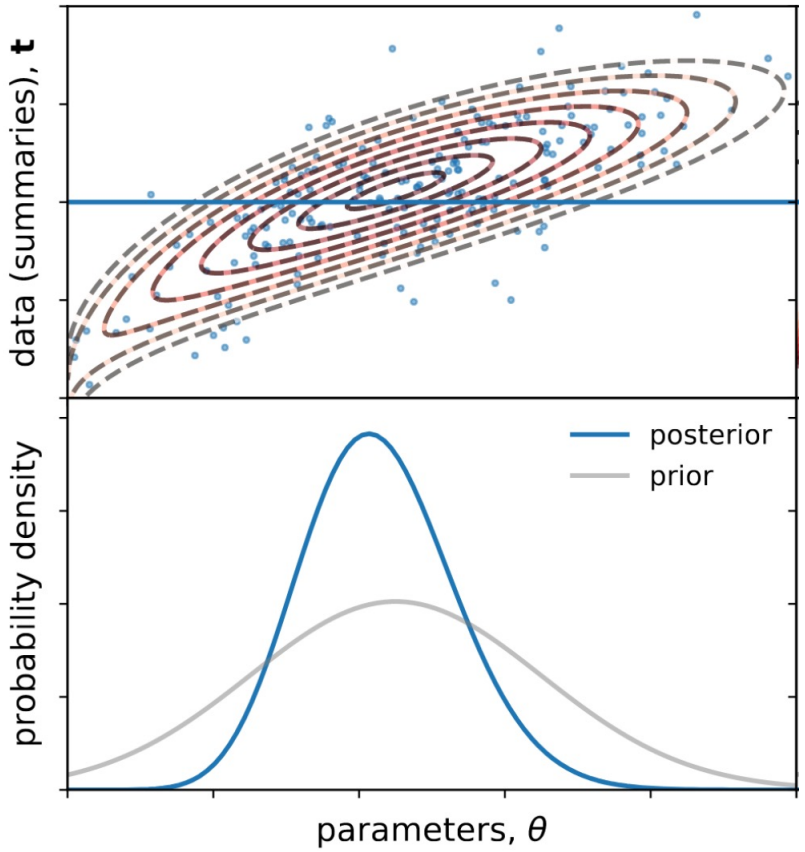
- Inverting with Bayes rule gives:

$$p(\boldsymbol{\theta} \,|\, \mathbf{x} = \mathbf{x}_o) \approx \frac{\Pr(\|\mathbf{x} - \mathbf{x}_o\| \leq \epsilon \,|\, \boldsymbol{\theta})\, p(\boldsymbol{\theta})}{\int \Pr(\|\mathbf{x} - \mathbf{x}_o\| \leq \epsilon \,|\, \boldsymbol{\theta}')\, p(\boldsymbol{\theta}')\, \mathrm{d}\boldsymbol{\theta}'} = p(\boldsymbol{\theta} \,|\, \|\mathbf{x} - \mathbf{x}_o\| \leq \epsilon)$$

Alsing+2019

Papamakarios 2019

Approximate Bayesian Computation (ABC)



Algorithm Rejection ABC

repeat

Sample $\boldsymbol{\theta} \sim p(\boldsymbol{\theta})$

Simulate $\mathbf{x} \sim p(\mathbf{x} \mid \boldsymbol{\theta})$

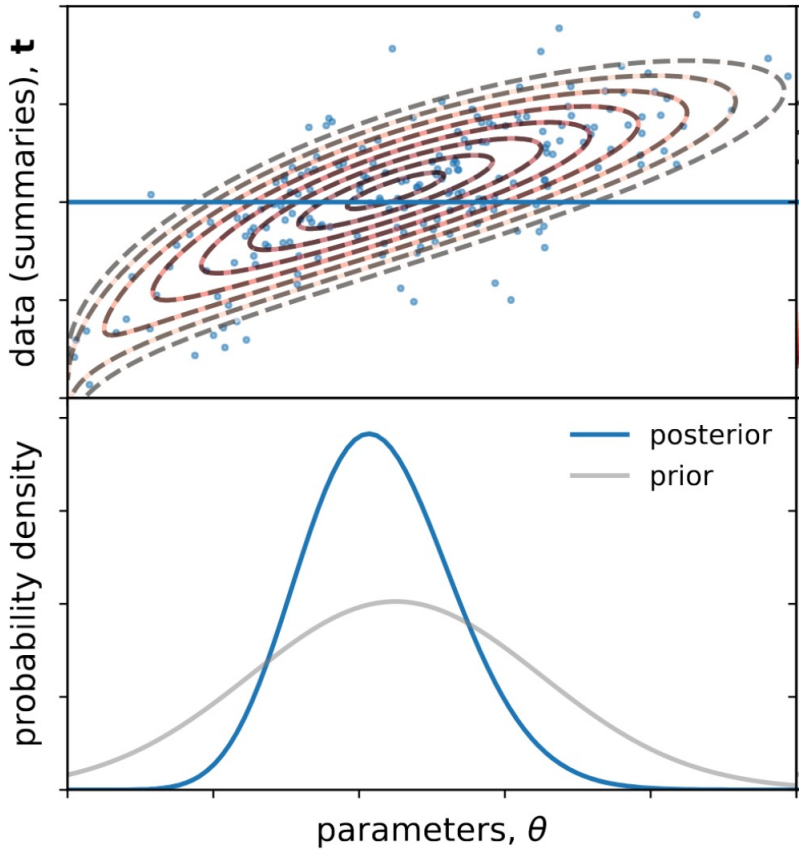
until $\|\mathbf{x} - \mathbf{x}_o\| \leq \epsilon$

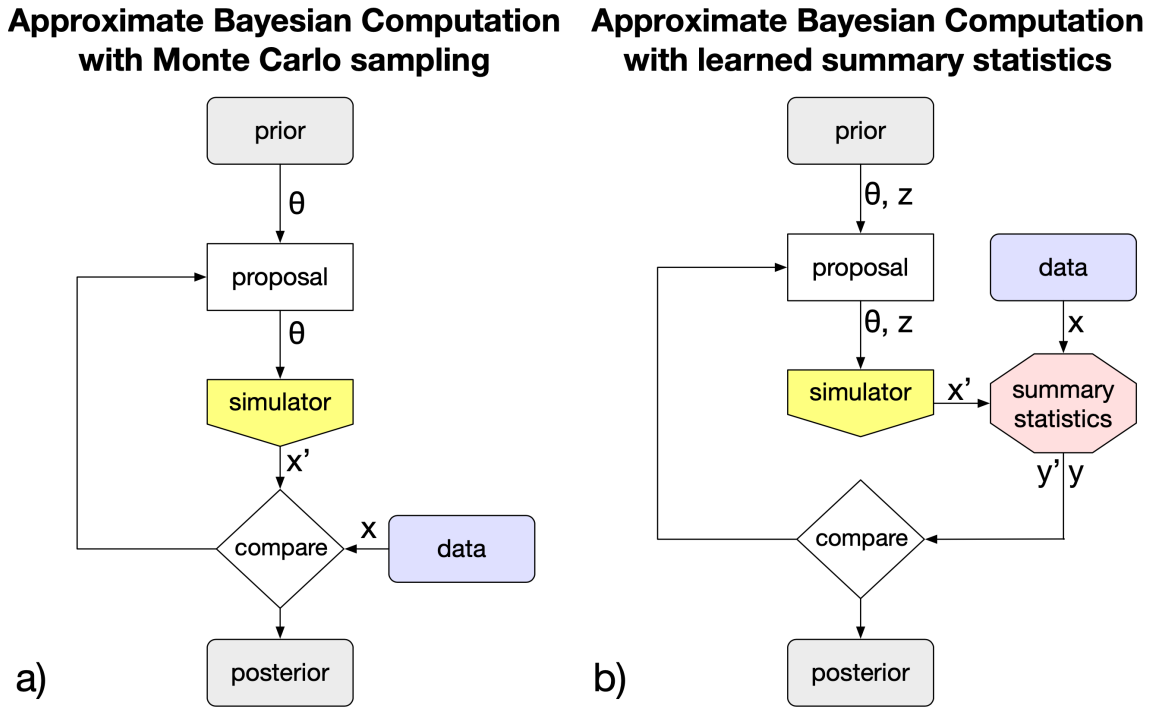
return $\boldsymbol{\theta}$

Alsing+2019

Papamakarios 2019

Approximate Bayesian Computation (ABC)





- For new data the whole procedure has to be repeated
- Curse of dimensionality

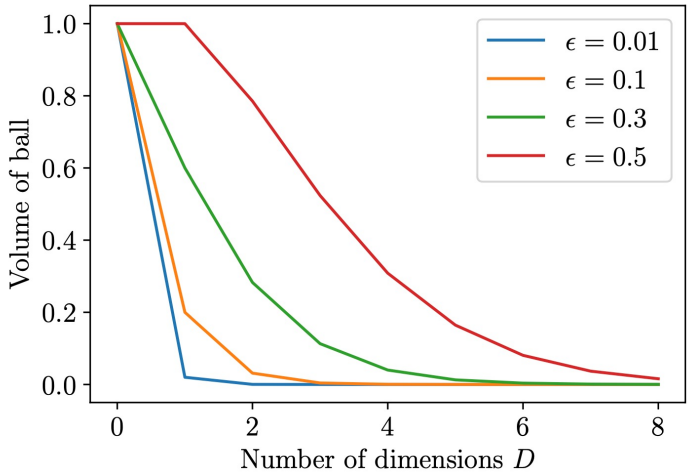
Alsing+2019

Cranmer+2020

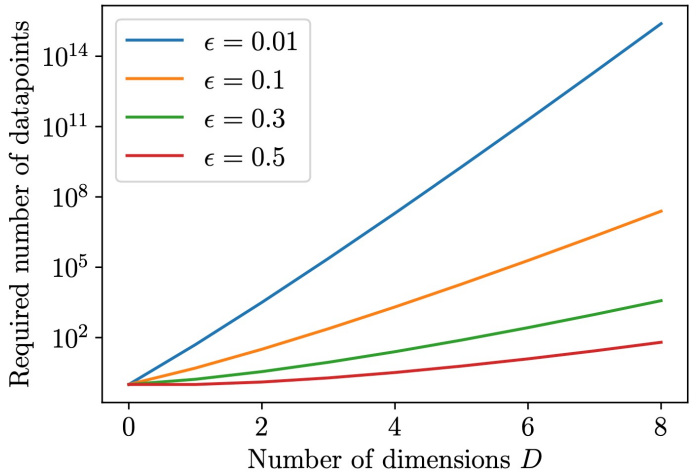
ABC & curse of dimensionality

- For an ε-cube around the data point:

$$\Pr(\|\mathbf{x} - \mathbf{x}_o\| \leq \epsilon \,|\, \boldsymbol{\theta}) \approx p(\mathbf{x}_o \,|\, \boldsymbol{\theta})\, |B_\epsilon(\mathbf{x}_o)| = p(\mathbf{x}_o \,|\, \boldsymbol{\theta})\, (2\epsilon)^D$$



(a) Volume of a D-dimensional ball of radius ϵ.



(b) Expected number of datapoints we need to generate until one falls in the ball.

Solutions:

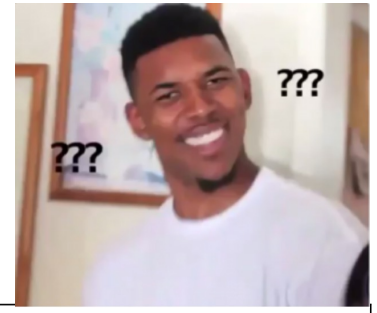
1. Summary statistics
2. High ε = lower-quality posterior
3. Smarter samplers (MCMC ABC , Sequential MC ABC)

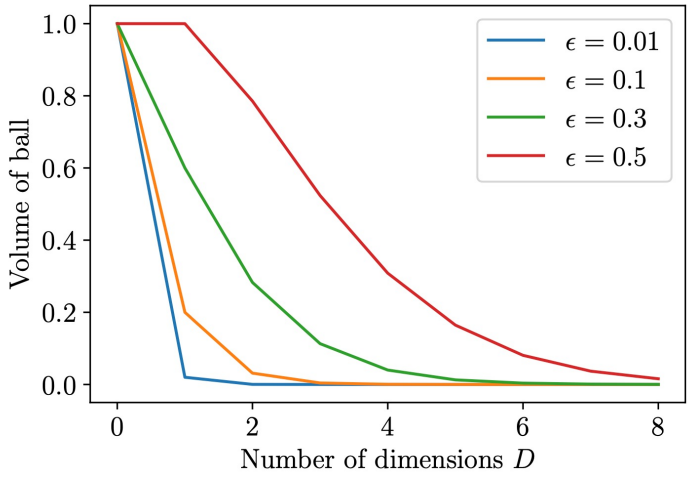
Papamakarios 2019

ABC & curse of dimensionality

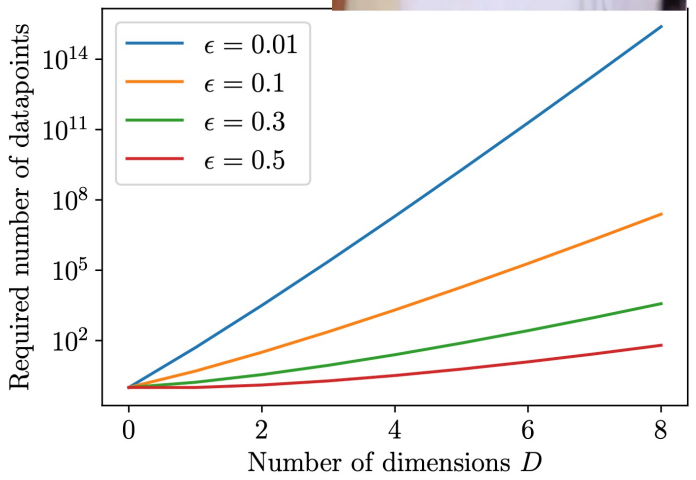
- For an ε-cube around the data point:

$$\Pr(\|\mathbf{x} - \mathbf{x}_o\| \leq \epsilon \,|\, \boldsymbol{\theta}) \approx p(\mathbf{x}_o \,|\, \boldsymbol{\theta})\, |B_\epsilon(\mathbf{x}_o)| = p(\mathbf{x}_o \,|\, \boldsymbol{\theta})\, (2\epsilon)^D$$





(a) Volume of a D-dimensional ball of radius ϵ.



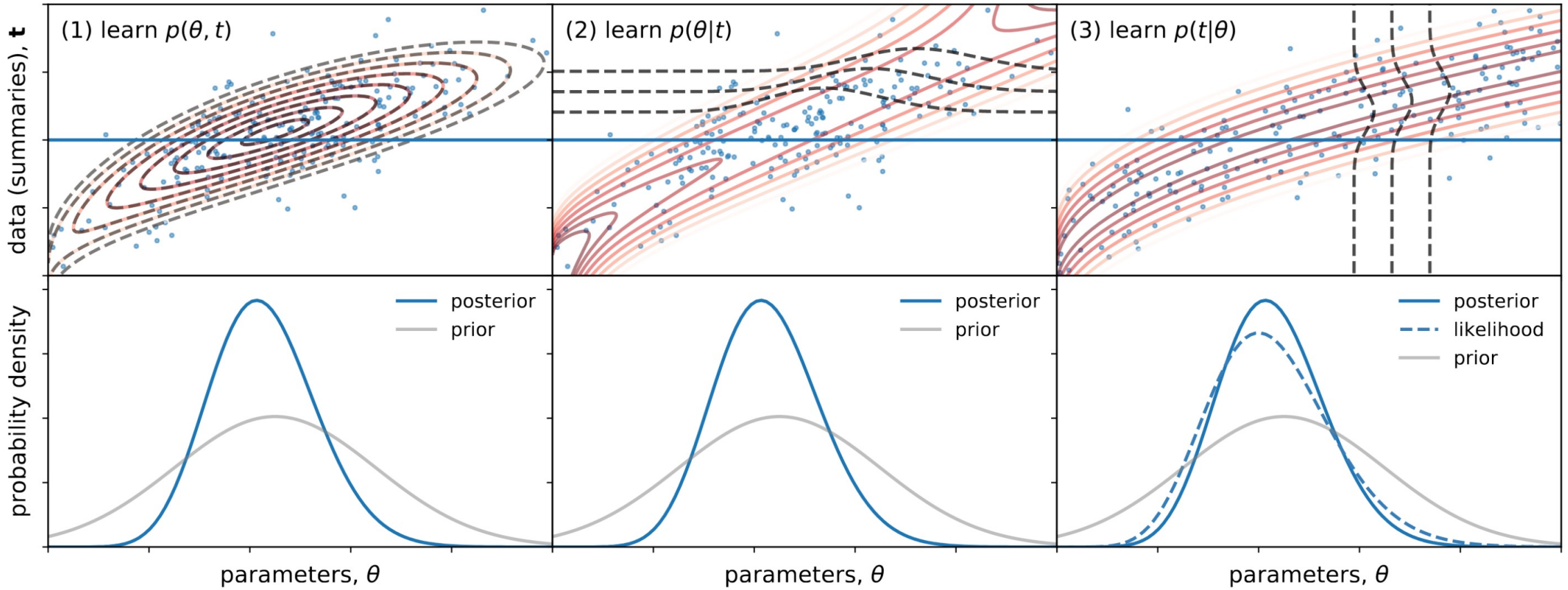
(b) Expected number of datapoints we need to generate until one falls in the ball.

Solutions:

1. Summary statistics
2. High ε = lower-quality posterior
3. Smarter samplers (MCMC ABC , Sequential MC ABC)

Papamakarios 2019

SBI - Learning the distribution

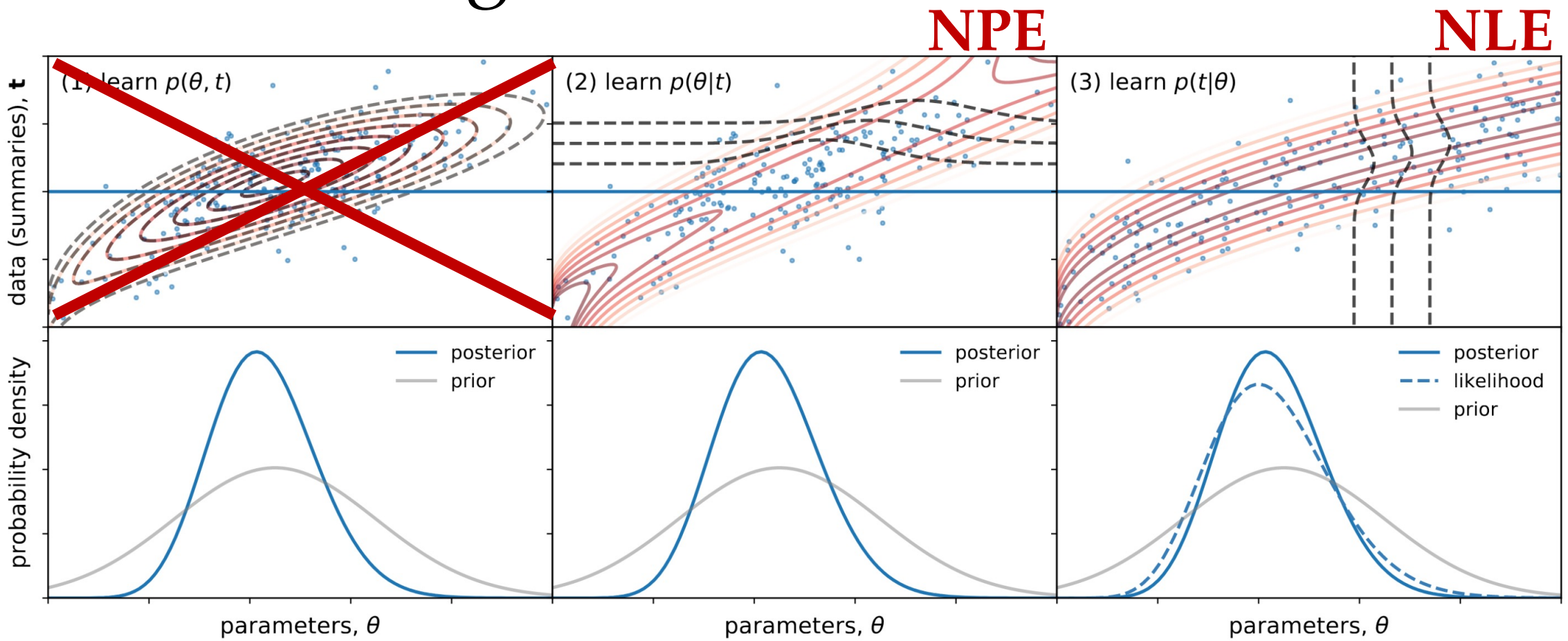


- Or: learn likelihood-evidence ratio $p(t|\theta)/p(t)$

→ *(Neural) density estimators*

Alsing+2019

SBI - Learning the distribution



- Or: learn likelihood-evidence ratio $p(t|\theta)/p(t)$ **NRE**

→ *(Neural) density estimators*

Alsing+2019

NPE vs NLE

NPE vs NLE

- NPE includes prior implicitly through the data
- NLE requires explicit prior to get the posterior

- Depending on the
  - dimensionality of the data vs. parameter space
  - Complexity of the likelihood vs. posterior

one or the other could work better

Density estimators (non-parametric)

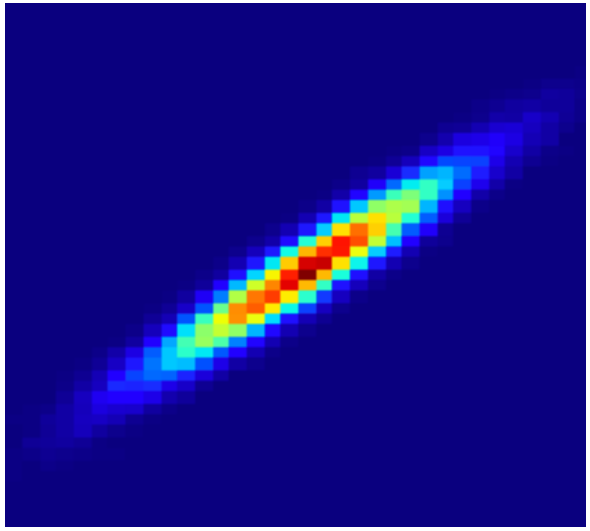
- Histograms

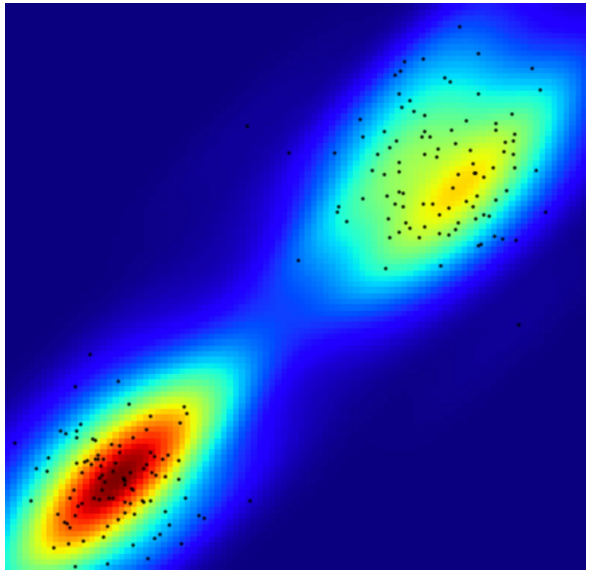
- Kernel density estimators
  - can be parametric if the width is tuned

$$q_\epsilon(\mathbf{x}) = \frac{1}{N}\sum_n k_\epsilon(\mathbf{x}-\mathbf{x}_n)$$

$$k_1(\mathbf{u}) = \frac{1}{(2\pi)^{D/2}}\exp\left(-\frac{1}{2}\|\mathbf{u}\|^2\right)$$

- Both will in general be cursed by dimensionality



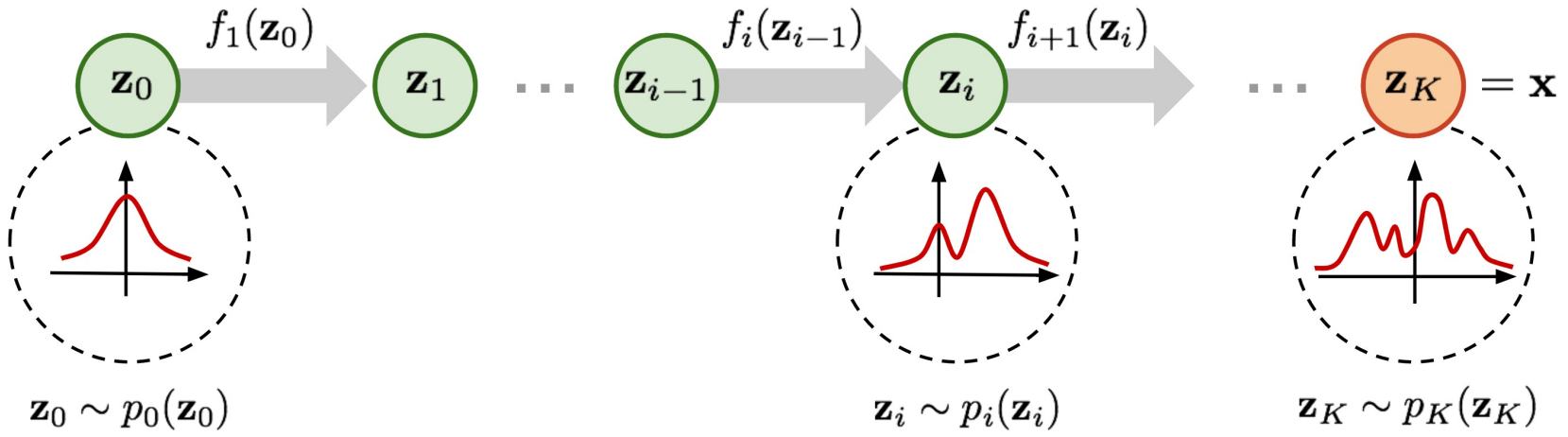


Density estimators (parametric)

- Gaussian $q_{\boldsymbol{\phi}}(\mathbf{x}) = \frac{1}{|\det(2\pi\boldsymbol{\Sigma})|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \quad \text{where} \quad \boldsymbol{\phi} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$
  - "MLE" for ϕ: $\boldsymbol{\mu}^* = \frac{1}{N}\sum_n \mathbf{x}_n \quad \text{and} \quad \boldsymbol{\Sigma}^* = \frac{1}{N}\sum_n (\mathbf{x}_n - \boldsymbol{\mu}^*)(\mathbf{x}_n - \boldsymbol{\mu}^*)^T$
- Mixture models $q_{\boldsymbol{\phi}}(\mathbf{x}) = \sum_k \alpha_k \, q^{(k)}_{\boldsymbol{\phi}_k}(\mathbf{x}) \quad \text{where} \quad \sum_k \alpha_k = 1 \quad \text{and} \quad \alpha_k \geq 0$
  - Gaussian mixture model: $q^{(k)}_{\phi_k} = \mathcal{N}\left(\mathbf{x}|\phi_k = (\boldsymbol{\mu}, \boldsymbol{\Sigma})\right)$

Density estimators (parametric)

- Normalizing flows
  - Deforming a know distribution to fit the unknown, target distribution



Credit: Lilian Weng

Density estimators (parametric)

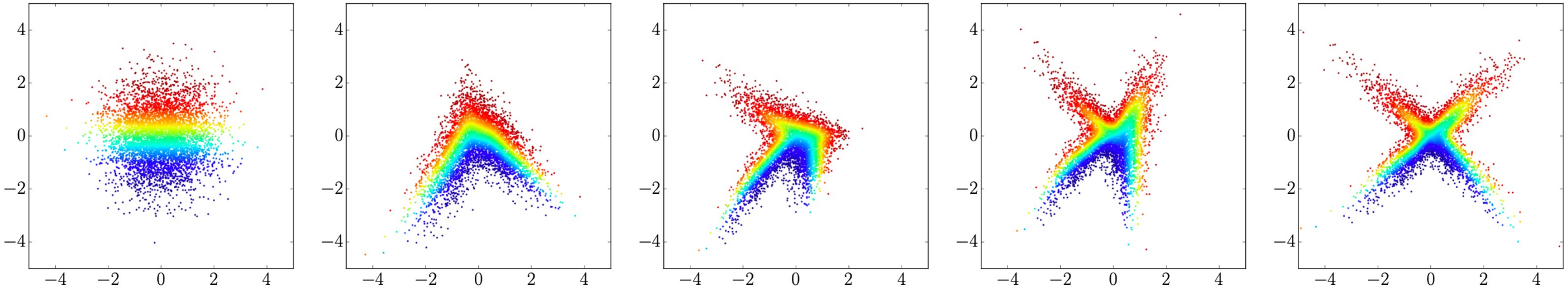
- Normalizing flows
  - Deforming a know distribution to fit the unknown, target distribution

$$\mathbf{z} \sim \pi(\mathbf{z}), \mathbf{x} = f(\mathbf{z}), \mathbf{z} = f^{-1}(\mathbf{x})$$

$$p(\mathbf{x}) = \pi(\mathbf{z}) \left| \det \frac{d\mathbf{z}}{d\mathbf{x}} \right| = \pi\left(f^{-1}(\mathbf{x})\right) \left| \det \frac{df^{-1}}{d\mathbf{x}} \right|$$

Density estimators (parametric)

- Normalizing flows
  - Deforming a know distribution to fit the unknown, target distribution



$$\mathbf{z}_0 \xrightarrow{} \boxed{T_1} \xrightarrow{\mathbf{z}_1} \boxed{T_2} \xrightarrow{\mathbf{z}_2} \cdots \xrightarrow{\mathbf{z}_{K-1}} \boxed{T_K} \xrightarrow{} \mathbf{z}_K$$

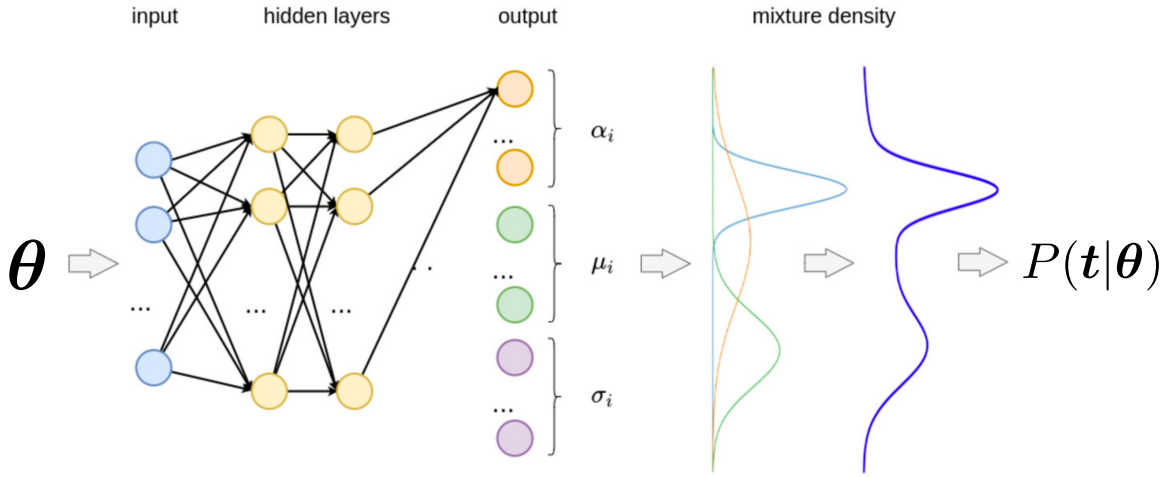
$$\log|\det J_{T_1}(\mathbf{z}_0)| + \log|\det J_{T_2}(\mathbf{z}_1)| + \cdots + \log|\det J_{T_K}(\mathbf{z}_{K-1})| = \log|\det J_T(\mathbf{z}_0)|$$

Papamakarios+2021

Conditioning the density

- Example #1: KDE -> example
- Example #2 Gaussian / Gaussian mixture

- Example #3: Normalizing flow
  - Conditioning each transformation on parameters θ



Repin+2021

Neural Likelihood Estimation

Neural Likelihood Estimation

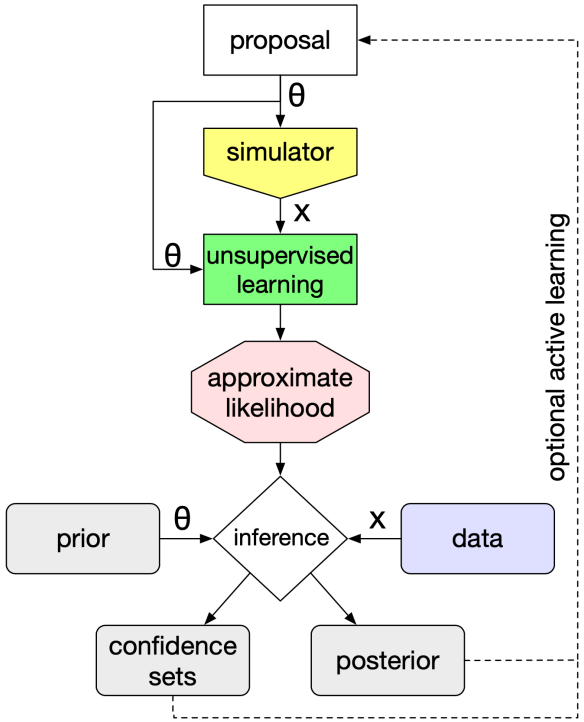
- Take a constructed set $\left\{(\tilde{t}_1, \tilde{\theta}_1), (\tilde{t}_2, \tilde{\theta}_2), \ldots, (\tilde{t}_N, \tilde{\theta}_N)\right\}$ pulled from the total distribution $p(t, \theta) = p(t|\theta) \cdot p(\theta)$

- Pick a (conditional) neural density of your choice

- Minimize KL divergence

$$D_{\mathrm{KL}}(P\|Q) = \int_{-\infty}^{\infty} p(x) \log p(x)\, \mathrm{d}x - \int_{-\infty}^{\infty} p(x) \log q(x)\, \mathrm{d}x$$

$$\approx -H_P - \frac{1}{N} \sum_{i=1}^{N} \log q(x_i)\,,$$



- (possibly) pull additional samples – sequential (active) learning

Neural Likelihood Estimation

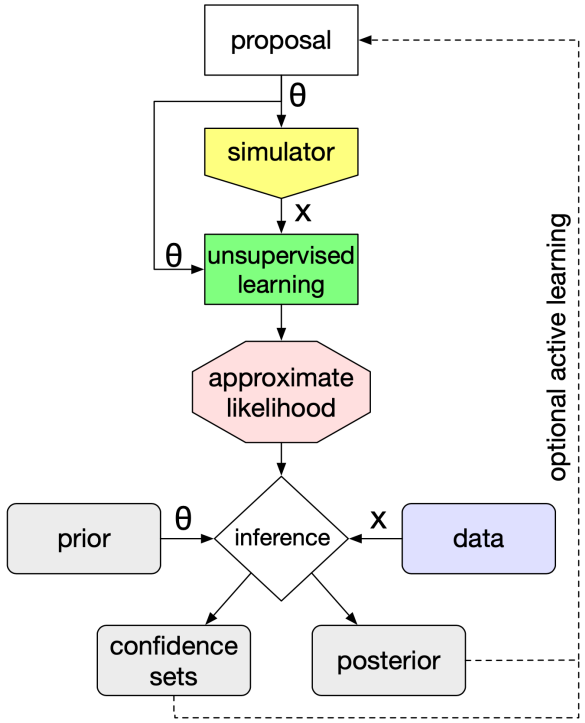
- (possibly) pull additional samples – sequential (active) learning

Algorithm Sequential Neural Likelihood (SNL)

Input : observed data $\mathbf{x}_o$, estimator $q_{\boldsymbol{\phi}}(\mathbf{x} \mid \boldsymbol{\theta})$, number of rounds R, simulations per round N

Output : approximate posterior $\hat{p}(\boldsymbol{\theta} \mid \mathbf{x}_o)$

set $\hat{p}_0(\boldsymbol{\theta} \mid \mathbf{x}_o) = p(\boldsymbol{\theta})$ and $\mathcal{D} = \{\}$
for $r = 1 : R$ **do**
  for $n = 1 : N$ **do**
    sample $\boldsymbol{\theta}_n \sim \hat{p}_{r-1}(\boldsymbol{\theta} \mid \mathbf{x}_o)$ with MCMC
    simulate $\mathbf{x}_n \sim p(\mathbf{x} \mid \boldsymbol{\theta}_n)$
    add $(\boldsymbol{\theta}_n, \mathbf{x}_n)$ into $\mathcal{D}$
  (re-)train $q_{\boldsymbol{\phi}}(\mathbf{x} \mid \boldsymbol{\theta})$ on $\mathcal{D}$ and set
  $\hat{p}_r(\boldsymbol{\theta} \mid \mathbf{x}_o) \propto q_{\boldsymbol{\phi}}(\mathbf{x}_o \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})$
return $\hat{p}_R(\boldsymbol{\theta} \mid \mathbf{x}_o)$



Neural Likelihood Estimation

- (possibly) pull additional samples – sequential (active) learning

Algorithm Sequential Neural Likelihood (SNL)

Input : observed data $\mathbf{x}_o$, estimator $q_{\boldsymbol{\phi}}(\mathbf{x} \mid \boldsymbol{\theta})$, number of rounds R, simulations per round N

Output : approximate posterior $\hat{p}(\boldsymbol{\theta} \mid \mathbf{x}_o)$

set $\hat{p}_0(\boldsymbol{\theta} \mid \mathbf{x}_o) = p(\boldsymbol{\theta})$ and $\mathcal{D} = \{\}$
for $r = 1 : R$ **do**
  for $n = 1 : N$ **do**
    sample $\boldsymbol{\theta}_n \sim \hat{p}_{r-1}(\boldsymbol{\theta} \mid \mathbf{x}_o)$ with MCMC
    simulate $\mathbf{x}_n \sim p(\mathbf{x} \mid \boldsymbol{\theta}_n)$
    add $(\boldsymbol{\theta}_n, \mathbf{x}_n)$ into $\mathcal{D}$
  (re-)train $q_{\boldsymbol{\phi}}(\mathbf{x} \mid \boldsymbol{\theta})$ on $\mathcal{D}$ and set
  $\hat{p}_r(\boldsymbol{\theta} \mid \mathbf{x}_o) \propto q_{\boldsymbol{\phi}}(\mathbf{x}_o \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})$
return $\hat{p}_R(\boldsymbol{\theta} \mid \mathbf{x}_o)$

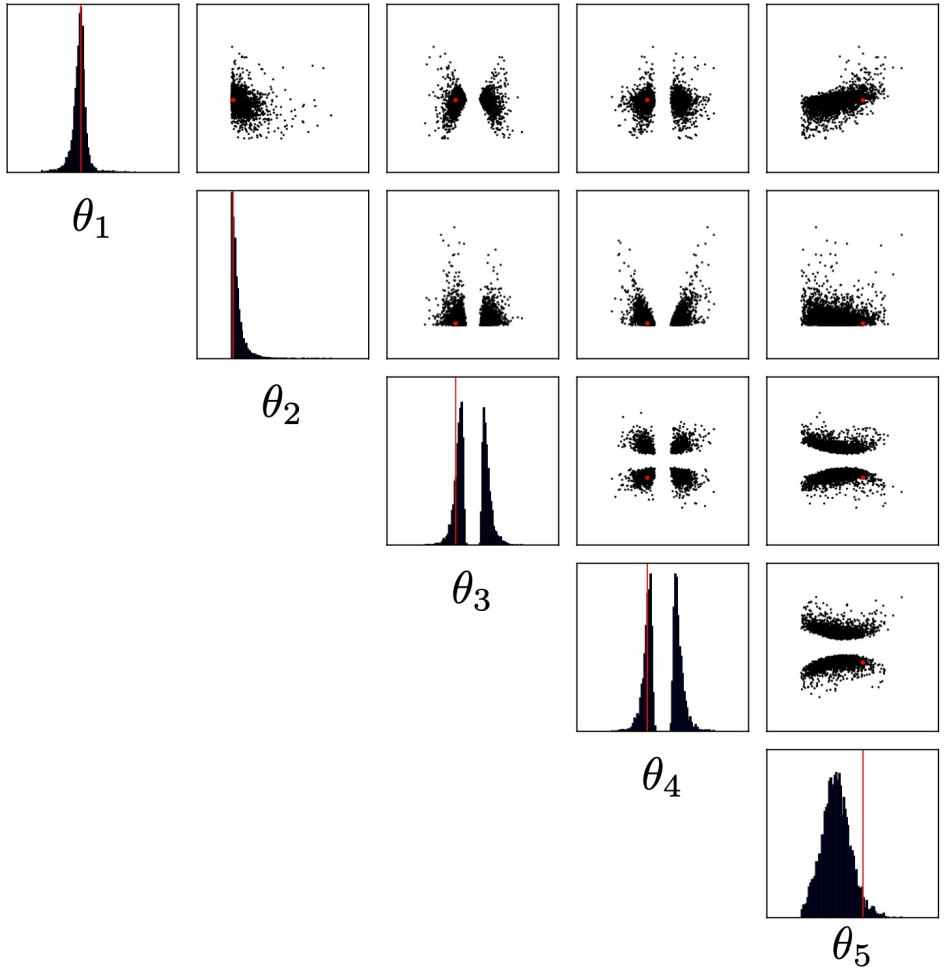
Pro:

- Provides better posterior as samples were chosen in the relevant parameter space

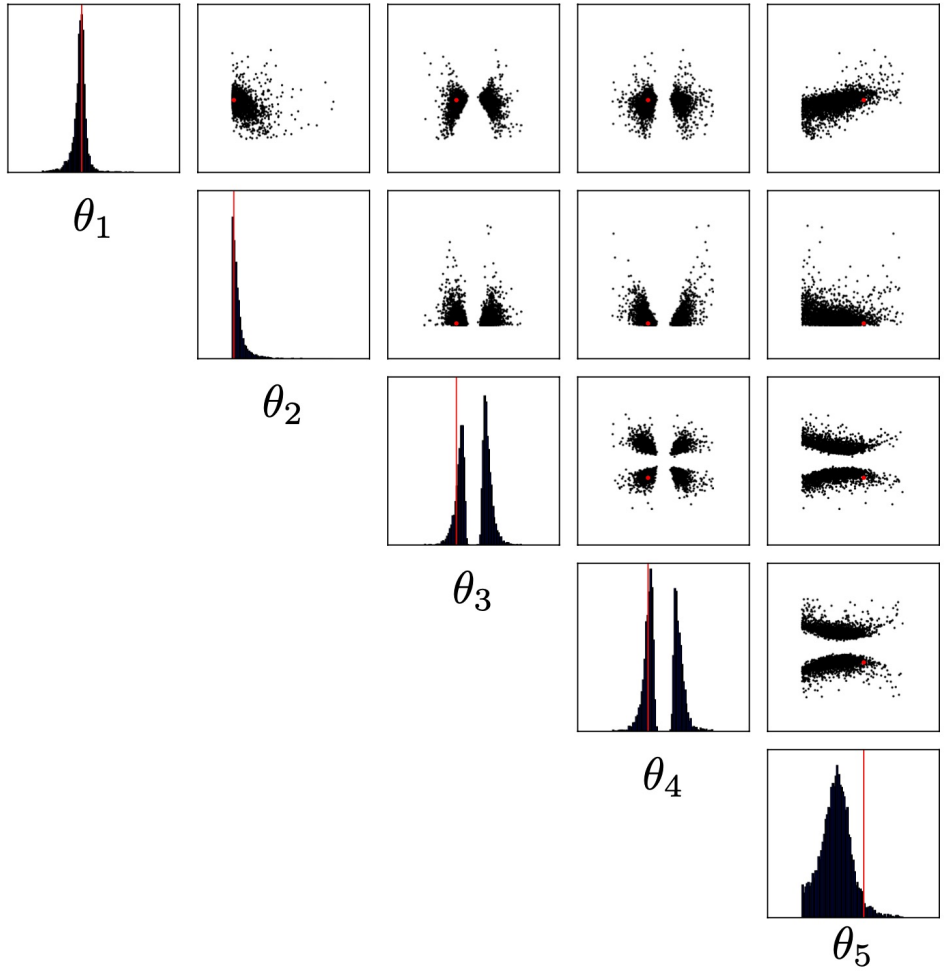
Con:

- It will not perform so well for samples other than x_0

Neural Likelihood Estimation

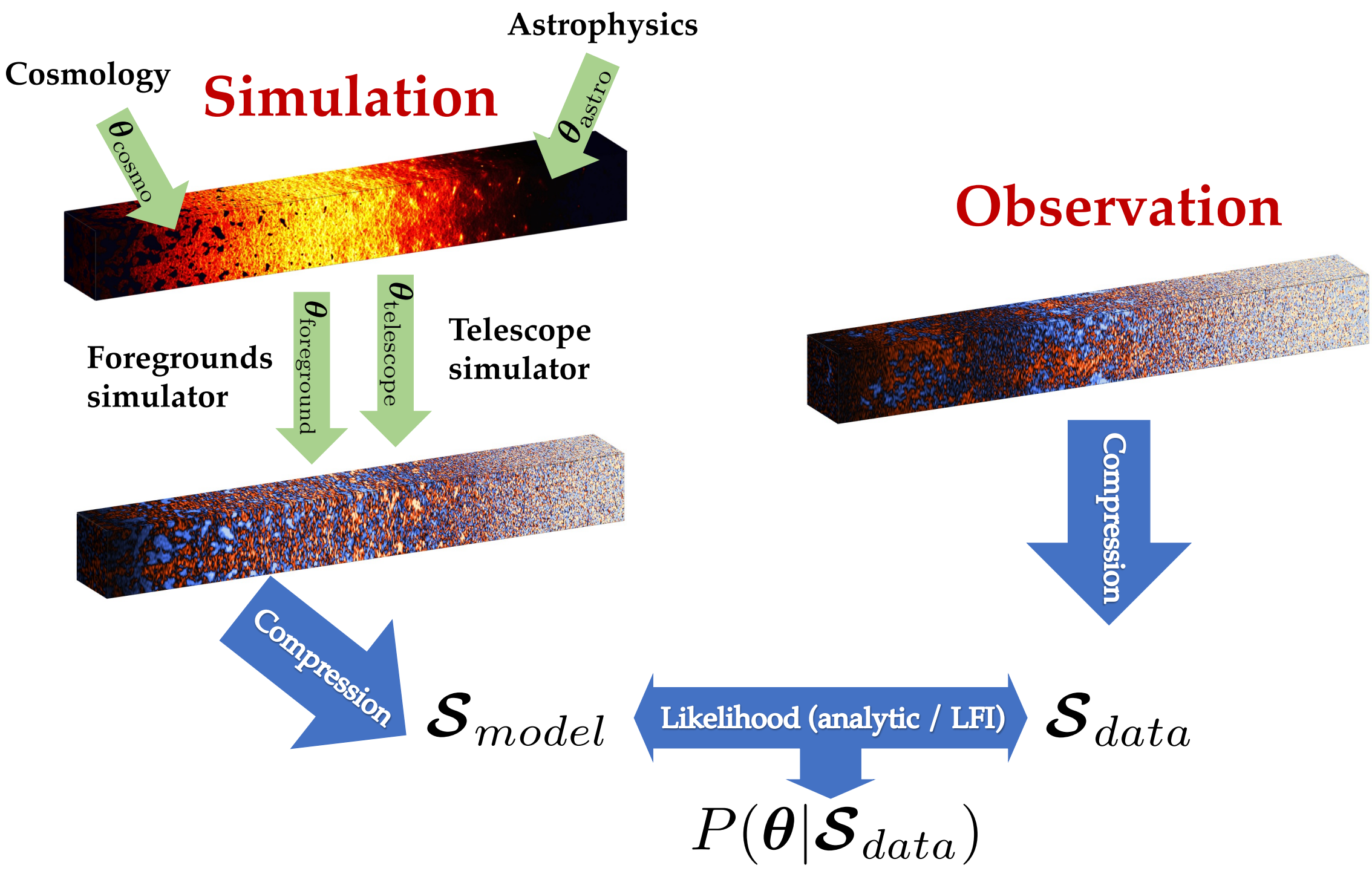


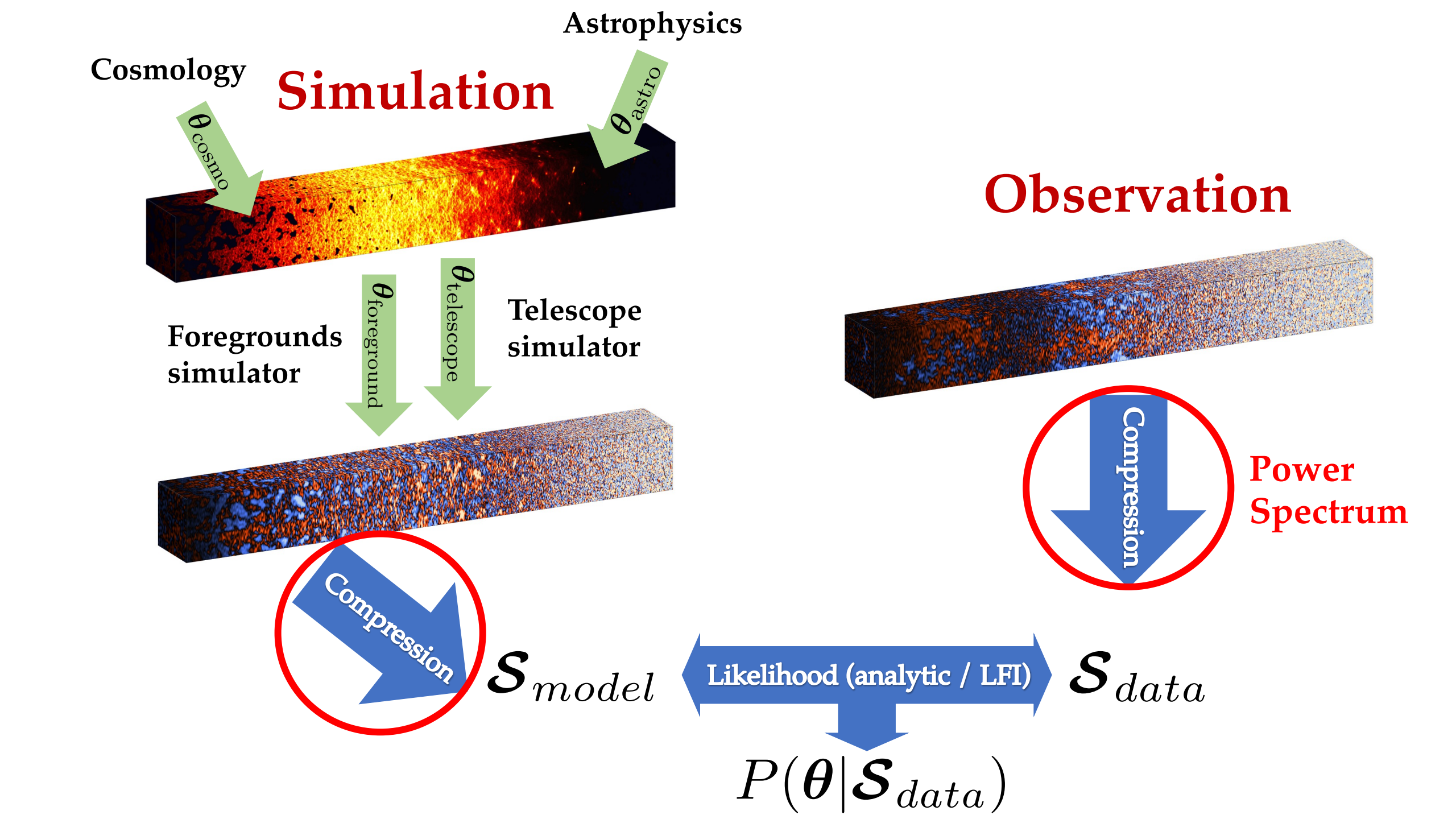
(a) MCMC samples from exact posterior. True parameters indicated in red.



(b) MCMC samples from SNL posterior. True parameters indicated in red.

Example: SBI with NLE for 21-cm PS


Astrophysics
Cosmology
Simulation
θ_{cosmo}
θ_{astro}
Observation
$\theta_{\mathrm{foreground}}$
$\theta_{\mathrm{telescope}}$
Foregrounds simulator
Telescope simulator
Compression
Compression
$\mathcal{S}_{model}$
Likelihood (analytic / LFI)
$\mathcal{S}_{data}$
$P(\theta|\mathcal{S}_{data})$


Astrophysics
Cosmology
Simulation
θ_{cosmo}
θ_{astro}
$\theta_{foreground}$
$\theta_{telescope}$
Foregrounds simulator
Telescope simulator
Observation
Compression
Power Spectrum
Compression
$\mathcal{S}_{model}$
Likelihood (analytic / LFI)
$\mathcal{S}_{data}$
$P(\theta|\mathcal{S}_{data})$

Likelihood of the 21-cm PS

- Analytically not available
- Numerically non-tractable
- Eg. – power spectrum

$$\mathcal{L}(\boldsymbol{\sigma}, \boldsymbol{\mu}; \mathbf{x}) = \frac{1}{(2\pi)^{n/2}\sqrt{\Pi_i \sigma_i^2}} e^{-\frac{1}{2}\sum_i (x_i - \mu_i)^2/\sigma_i^2}$$

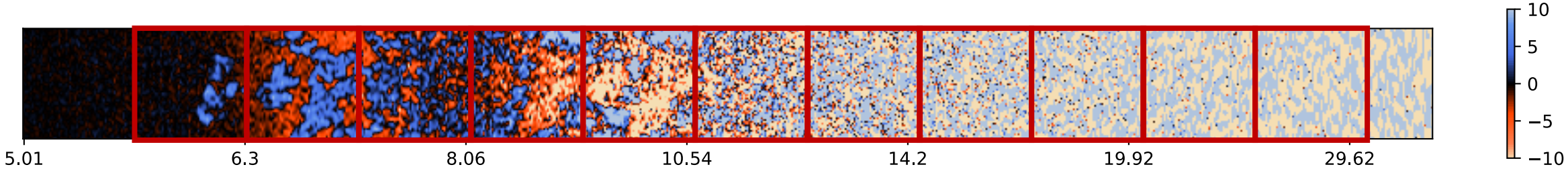
$$\mathcal{L}(\boldsymbol{\Sigma}, \boldsymbol{\mu}; \mathbf{x}) = \frac{1}{(2\pi)^{n/2}\sqrt{|\Sigma|}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu})}$$

??? higher order terms & non-constant σ/Σ ???

–> Neural Likelihood Estimation

Likelihood of the 21-cm PS

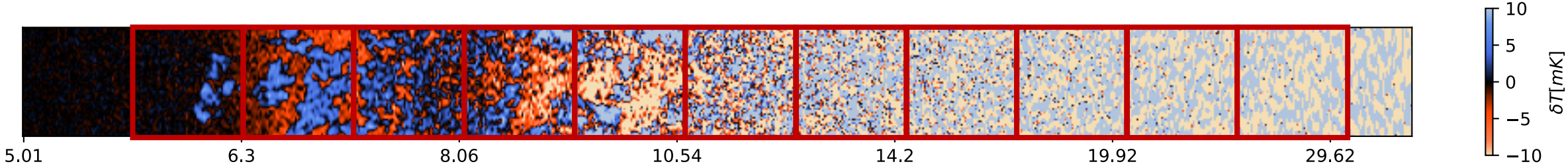
- Is inclusion of the covariance worthwhile?

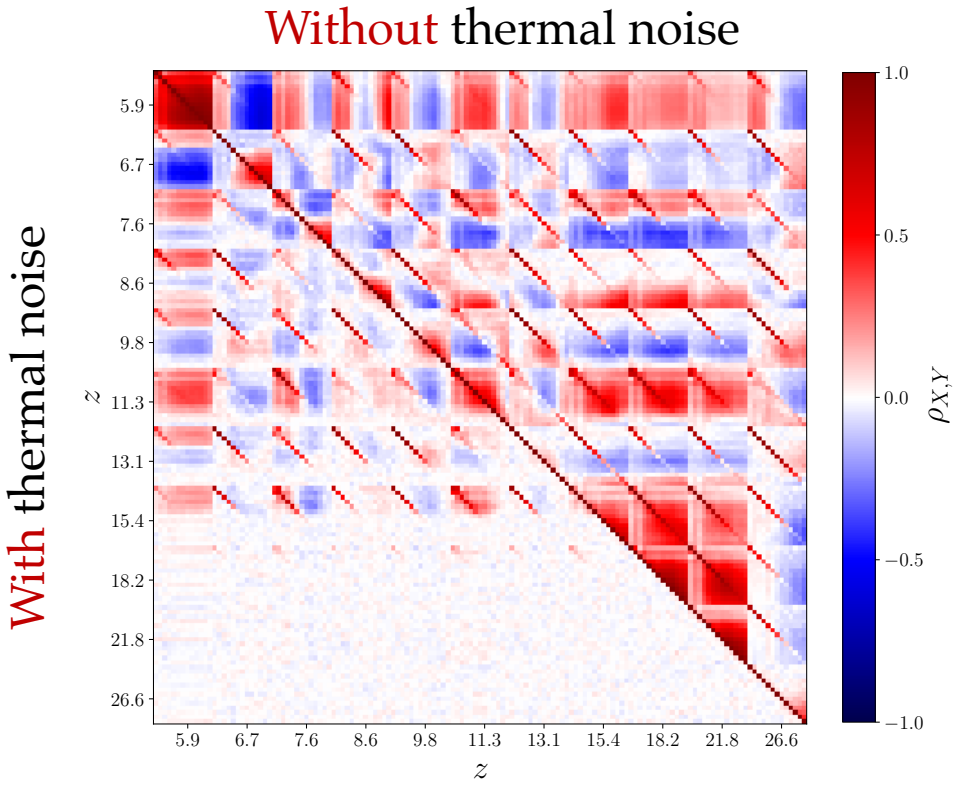


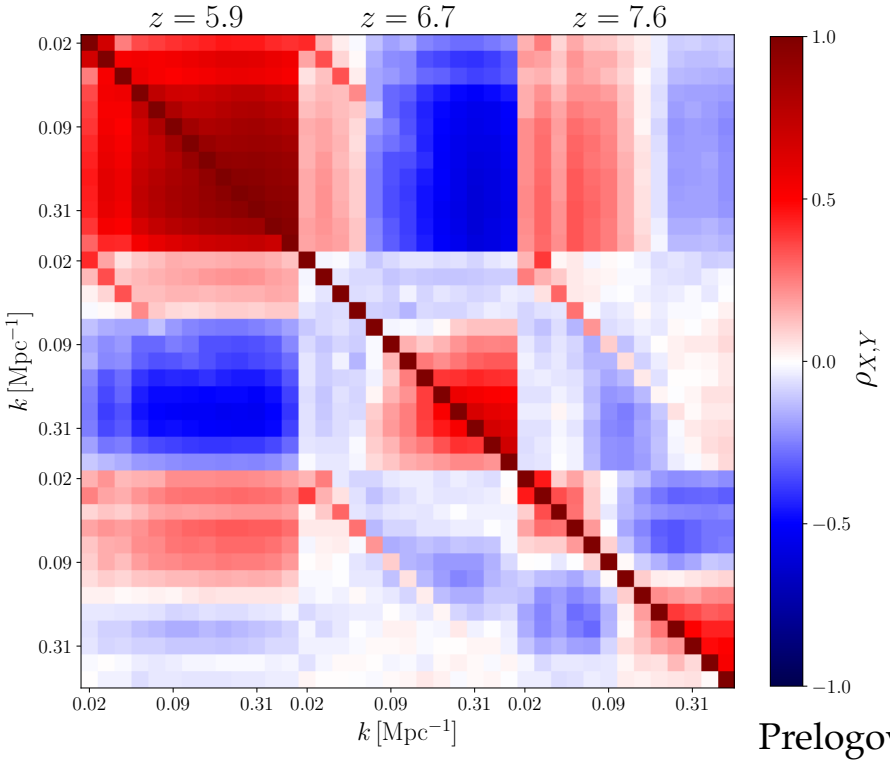
Prelogović+ in.prep.

Likelihood of the 21-cm PS

- Is inclusion of the covariance worthwhile?







Prelogović+ in.prep.

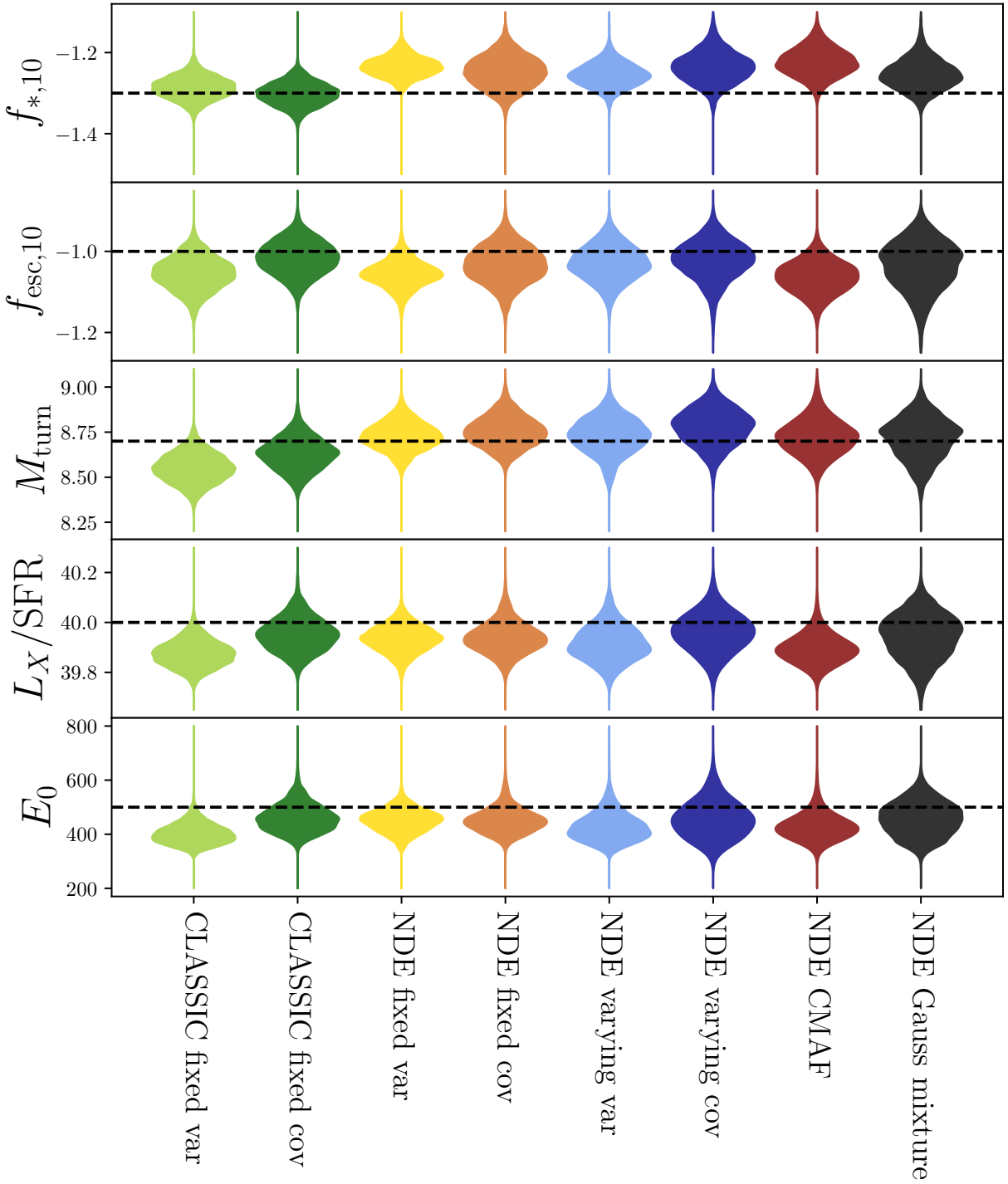
Likelihood of the 21-cm PS

- Classically, Σ is estimated only at the fiducial
  - Estimating it at every point in parameter space is too expensive
  - Even for a proper Gaussian, we rely on NLE
- Gaussian Mixture Density Network
- Conditional Masked Autoregressive Flows

Likelihood of the 21-cm PS

	CLASSIC		NDE				
	fixed var	fixed cov	fixed var	fixed cov	varying var	varying cov	CMAF
$f_{*,10}$	1.34	2.71	0.50	0.03	0.05	0.11	0.36
$f_{\rm esc,10}$	0.53	1.01	0.84	0.16	0.13	0.07	0.57
$M_{\rm turn}$	2.37	0.49	0.32	0.38	0.03	0.19	0.05
$L_X/{\rm SFR}$	1.19	0.50	0.59	0.26	0.16	0.06	0.48
E_0	1.38	0.15	0.26	0.18	0.17	0.02	0.21

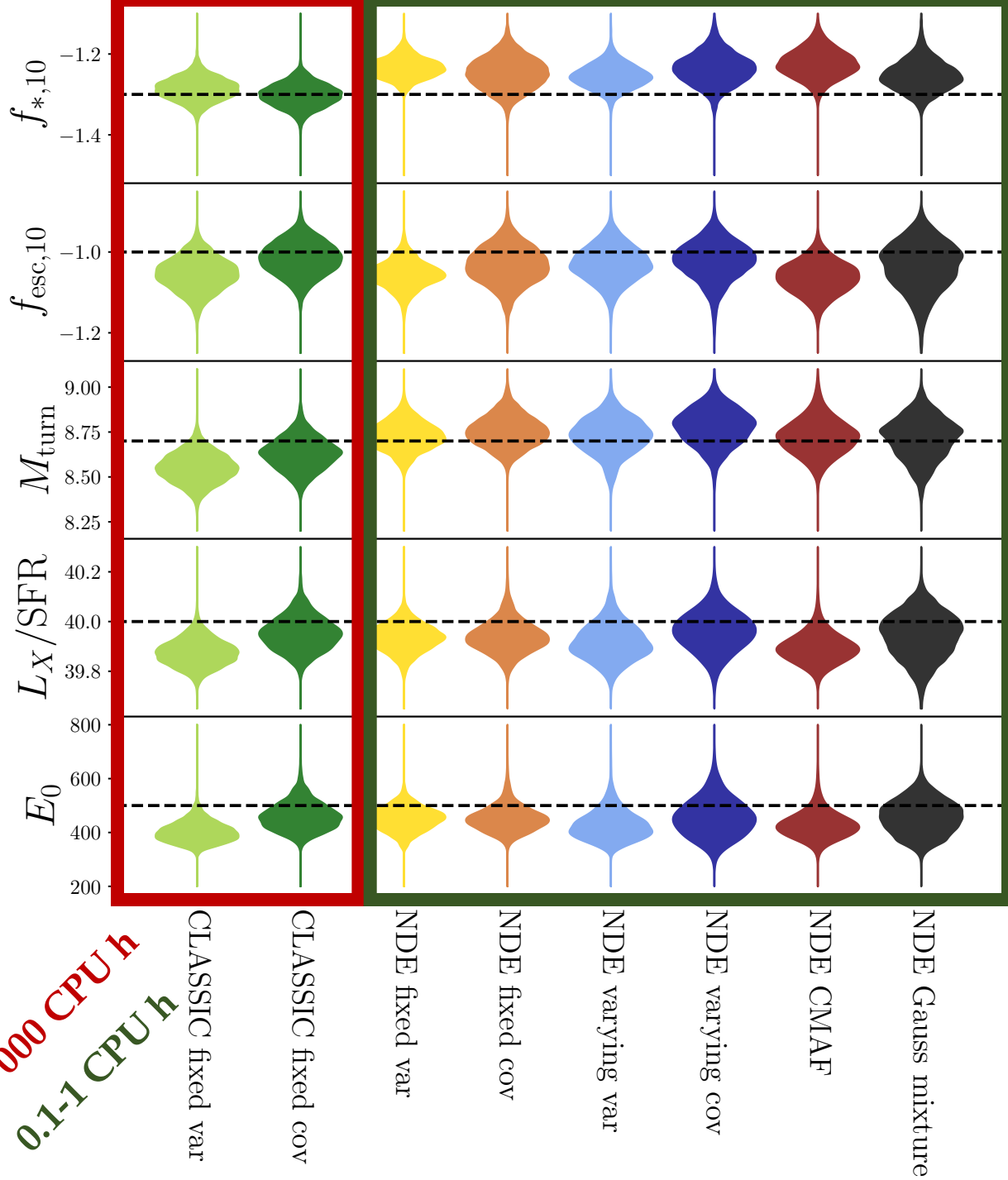
- Excluding correlations gives over-confident and biased posteriors
- NDE varying cov close to the NDE Gauss mixture
- NDE CMAF not performing very well
  - Possible issue is high expressivity, so it needs **sequential learning**



Prelogović+ in.prep.

Likelihood of the 21-cm PS

	CLASSIC		NDE				
	fixed var	fixed cov	fixed var	fixed cov	varying var	varying cov	CMAF
$f_{*,10}$	1.34	2.71	0.50	0.03	0.05	0.11	0.36
$f_{\rm esc,10}$	0.53	1.01	0.84	0.16	0.13	0.07	0.57
$M_{\rm turn}$	2.37	0.49	0.32	0.38	0.03	0.19	0.05
$L_X/{\rm SFR}$	1.19	0.50	0.59	0.26	0.16	0.06	0.48
E_0	1.38	0.15	0.26	0.18	0.17	0.02	0.21



- Excluding correlations gives over-confident and biased posteriors
- NDE varying cov close to the NDE Gauss mixture
- NDE CMAF not performing very well
  - Possible issue is high expressivity, so it needs **sequential learning**

Prelogović+ in.prep.

Likelihood of the 21-cm PS

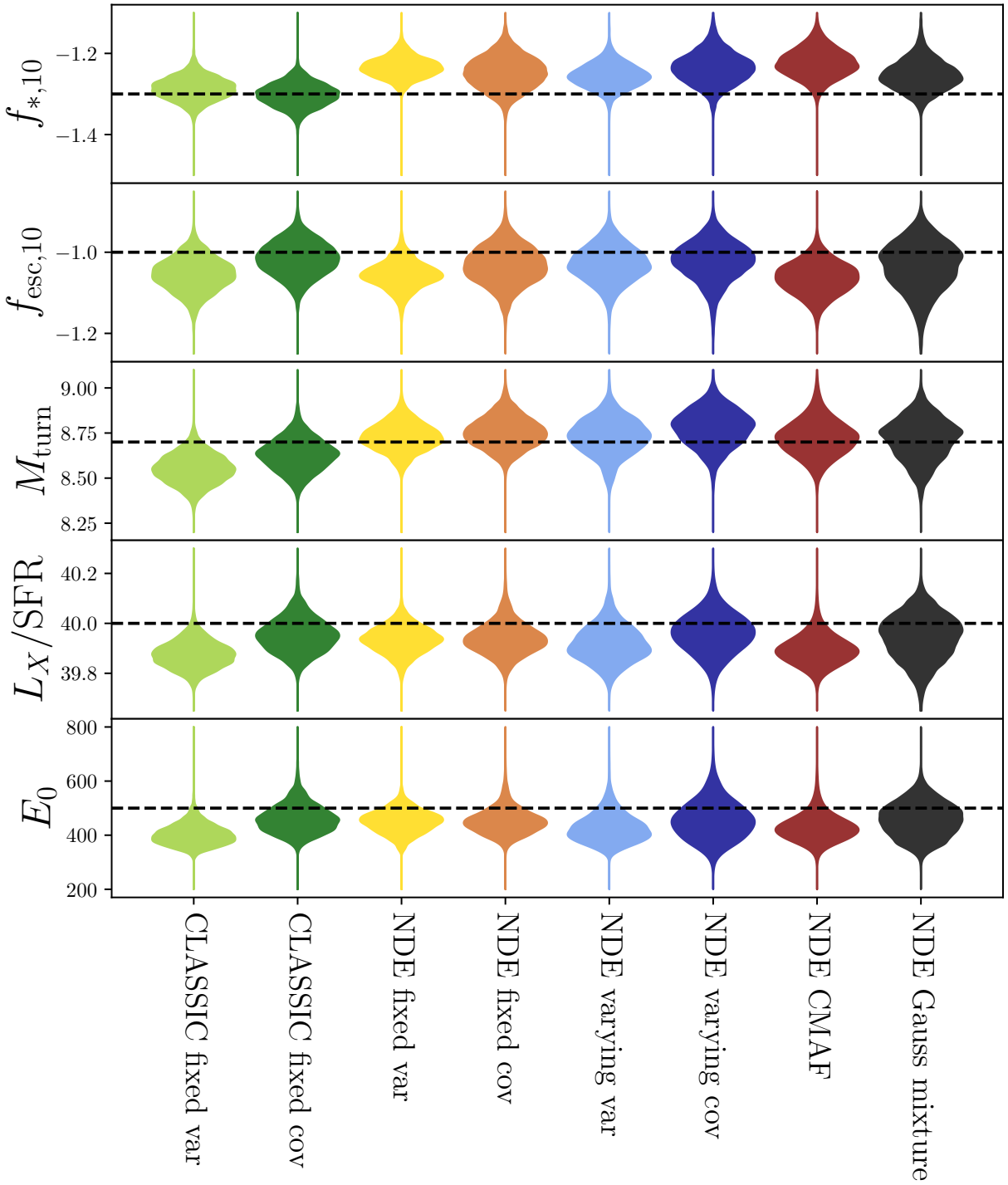
	CLASSIC		NDE				
	fixed var	fixed cov	fixed var	fixed cov	varying var	varying cov	CMAF
$f_{*,10}$	1.34	2.71	0.50	0.03	0.05	0.11	0.36
$f_{\rm esc,10}$	0.53	1.01	0.84	0.16	0.13	0.07	0.57
$M_{\rm turn}$	2.37	0.49	0.32	0.38	0.03	0.19	0.05
$L_X/{\rm SFR}$	1.19	0.50	0.59	0.26	0.16	0.06	0.48
E_0	1.38	0.15	0.26	0.18	0.17	0.02	0.21

BUT:

This is only qualitative description, and only for the mock observation

- How does it perform for other points in the parameter space?
- Did the training converge?
- Can we quantify the best model?

–> Simulation-Based Calibration



Prelogović+ in.prep.

Simulation Based Calibration (SBC)

- ***"prior" = "data averaged posterior"*** $P(\boldsymbol{\theta}) = \int P(\boldsymbol{\theta}|\tilde{\boldsymbol{y}})\, P(\tilde{\boldsymbol{y}}|\tilde{\boldsymbol{\theta}}) P(\tilde{\boldsymbol{\theta}})\, \mathrm{d}\tilde{\boldsymbol{y}}\, \mathrm{d}\tilde{\boldsymbol{\theta}}$

Simulation Based Calibration (SBC)

- ***"prior" = "data averaged posterior"*** $P(\boldsymbol{\theta}) = \int P(\boldsymbol{\theta}|\tilde{\boldsymbol{y}})\, P(\tilde{\boldsymbol{y}}|\tilde{\boldsymbol{\theta}}) P(\tilde{\boldsymbol{\theta}})\, \mathrm{d}\tilde{\boldsymbol{y}}\, \mathrm{d}\tilde{\boldsymbol{\theta}}$

1. Pull from prior $\tilde{\boldsymbol{\theta}} \sim P(\boldsymbol{\theta})$

Simulation Based Calibration (SBC)

- ***"prior" = "data averaged posterior"*** $P(\boldsymbol{\theta}) = \int P(\boldsymbol{\theta}|\tilde{\boldsymbol{y}})\, P(\tilde{\boldsymbol{y}}|\tilde{\boldsymbol{\theta}}) P(\tilde{\boldsymbol{\theta}})\, \mathrm{d}\tilde{\boldsymbol{y}}\, \mathrm{d}\tilde{\boldsymbol{\theta}}$

1. Pull from prior $\tilde{\boldsymbol{\theta}} \sim P(\boldsymbol{\theta})$
2. Pull the data from the likelihood $\tilde{\boldsymbol{y}} \sim P(\boldsymbol{y}|\tilde{\boldsymbol{\theta}}) \quad \Leftrightarrow \quad \tilde{\boldsymbol{y}} = \mathrm{simulator}(\tilde{\boldsymbol{\theta}})$

Simulation Based Calibration (SBC)

- ***"prior" = "data averaged posterior"*** $P(\boldsymbol{\theta}) = \int P(\boldsymbol{\theta}|\tilde{\boldsymbol{y}})\, P(\tilde{\boldsymbol{y}}|\tilde{\boldsymbol{\theta}}) P(\tilde{\boldsymbol{\theta}})\, \mathrm{d}\tilde{\boldsymbol{y}}\, \mathrm{d}\tilde{\boldsymbol{\theta}}$

1. Pull from prior $\tilde{\boldsymbol{\theta}} \sim P(\boldsymbol{\theta})$
2. Pull the data from the likelihood $\tilde{\boldsymbol{y}} \sim P(\boldsymbol{y}|\tilde{\boldsymbol{\theta}}) \quad \Leftrightarrow \quad \tilde{\boldsymbol{y}} = \mathrm{simulator}(\tilde{\boldsymbol{\theta}})$
3. Calculate the posterior the sample $P(\boldsymbol{\theta}|\tilde{\boldsymbol{y}})$

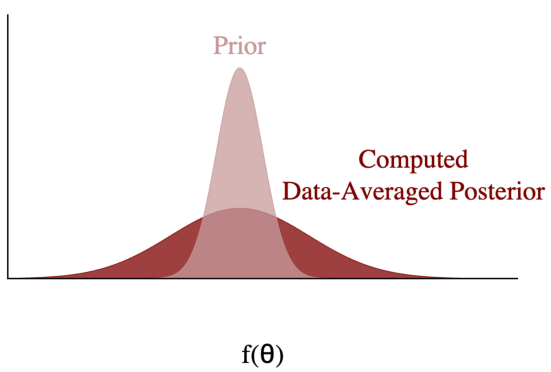
Simulation Based Calibration (SBC)

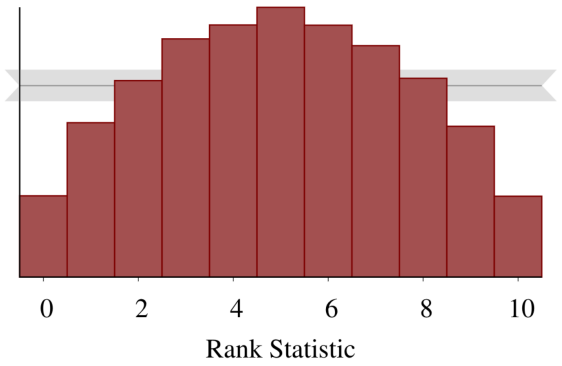
- ***"prior" = "data averaged posterior"*** $P(\boldsymbol{\theta}) = \int P(\boldsymbol{\theta}|\tilde{\boldsymbol{y}})\, P(\tilde{\boldsymbol{y}}|\tilde{\boldsymbol{\theta}}) P(\tilde{\boldsymbol{\theta}})\, \mathrm{d}\tilde{\boldsymbol{y}}\, \mathrm{d}\tilde{\boldsymbol{\theta}}$

1. Pull from prior $\tilde{\boldsymbol{\theta}} \sim P(\boldsymbol{\theta})$
2. Pull the data from the likelihood $\tilde{\boldsymbol{y}} \sim P(\boldsymbol{y}|\tilde{\boldsymbol{\theta}}) \quad \Leftrightarrow \quad \tilde{\boldsymbol{y}} = \mathrm{simulator}(\tilde{\boldsymbol{\theta}})$
3. Calculate the posterior the sample $P(\boldsymbol{\theta}|\tilde{\boldsymbol{y}})$
4. Repeat and average posteriors $P(\boldsymbol{\theta}) \approx \frac{1}{N} \sum_{i=1}^{N} P_i(\boldsymbol{\theta}|\tilde{\boldsymbol{y}}_i)$

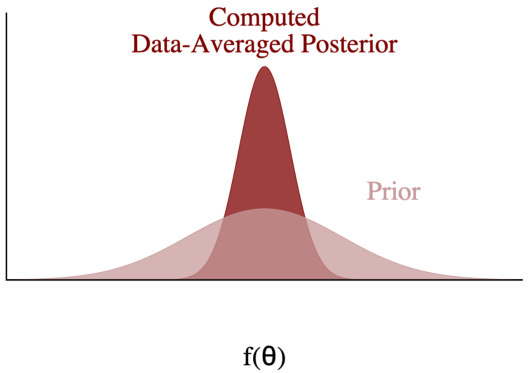
Simulation Based Calibration (SBC)

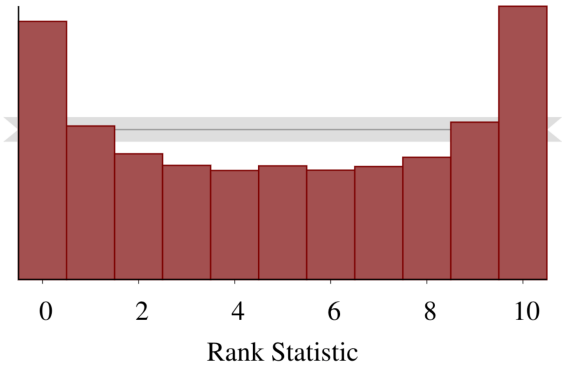
- ***"prior" = "data averaged posterior"*** $P(\boldsymbol{\theta}) = \int P(\boldsymbol{\theta}|\tilde{\boldsymbol{y}})\, P(\tilde{\boldsymbol{y}}|\tilde{\boldsymbol{\theta}}) P(\tilde{\boldsymbol{\theta}})\, \mathrm{d}\tilde{\boldsymbol{y}}\, \mathrm{d}\tilde{\boldsymbol{\theta}}$
- SBC – casting integral into 1D rank statistics distribution

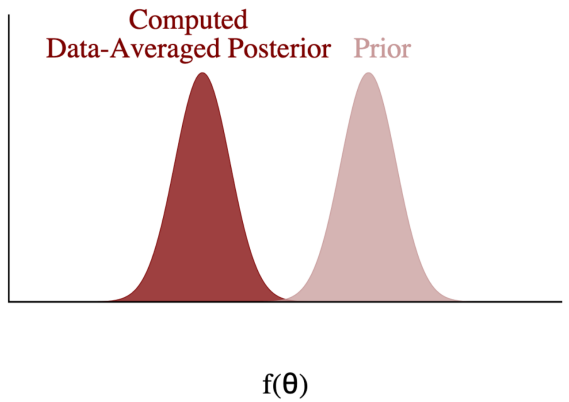


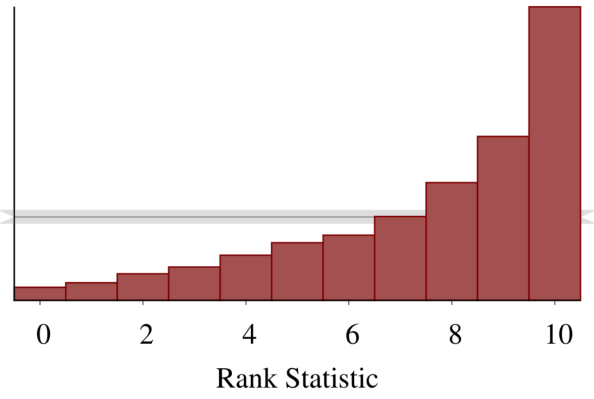


$$r\left(f(\boldsymbol{\theta}_1), \ldots f(\boldsymbol{\theta}_L), f(\tilde{\boldsymbol{\theta}})\right) = \sum_{l=1}^{L} \mathbb{I}\left[f(\boldsymbol{\theta}_l) < f(\tilde{\boldsymbol{\theta}})\right]$$









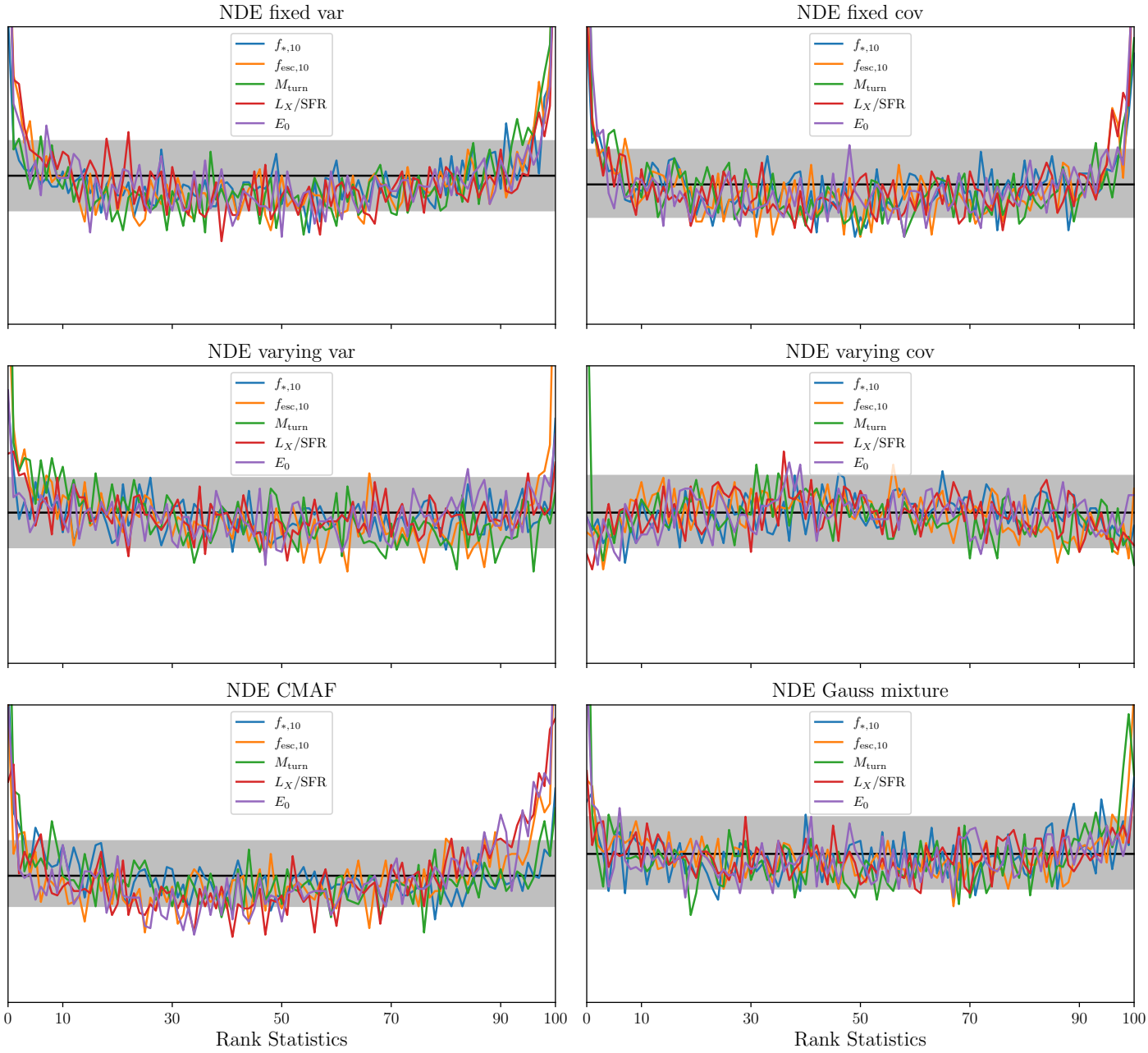
Talts+2018

SBC for 21-cm PS

- *Expensive to compute!*
  - *10 000 posteriors*

- However, once likelihood is trained, no new simulations are needed
  - "amortized inference"
- Procedure would be useful for classic inference too, but is impossible to compute

- NDE fixed var & cov – overconfident
- NDE varying var & cov – good
- NDE CMAF – overconfident
- NDE Gauss mixture – the best



Prelogović+ in.prep.

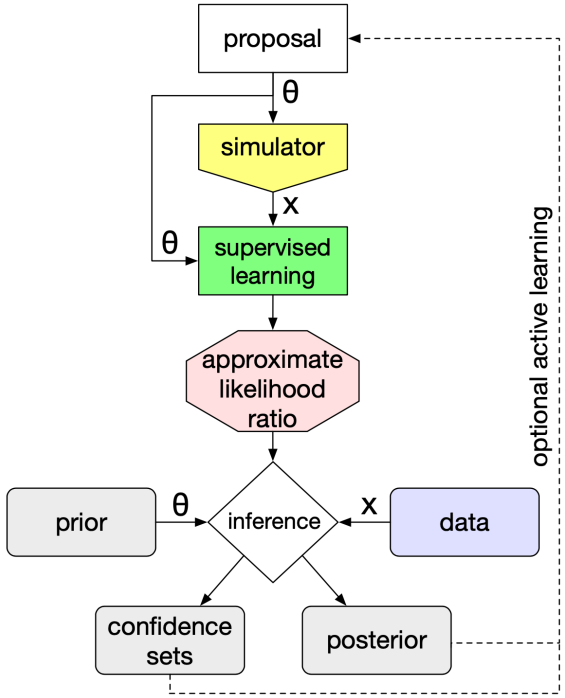
Neural Ratio Estimation

and marginalization

NRE vs NLE/NPE

- No need to estimate a full distribution
- Ratio is a simple scalar
- Any neural network will do
- Doesn’t suffer from the curse of dimensionality (in data)

Neural Ratio Estimation (NRE)



- The main focus:

$$r(\boldsymbol{x}, \boldsymbol{\theta}) \equiv \frac{p(\boldsymbol{x} \mid \boldsymbol{\theta})}{p(\boldsymbol{x})} = \frac{p(\boldsymbol{\theta} \mid \boldsymbol{x})}{p(\boldsymbol{\theta})} = \frac{p(\boldsymbol{x}, \boldsymbol{\theta})}{p(\boldsymbol{x})p(\boldsymbol{\theta})}$$

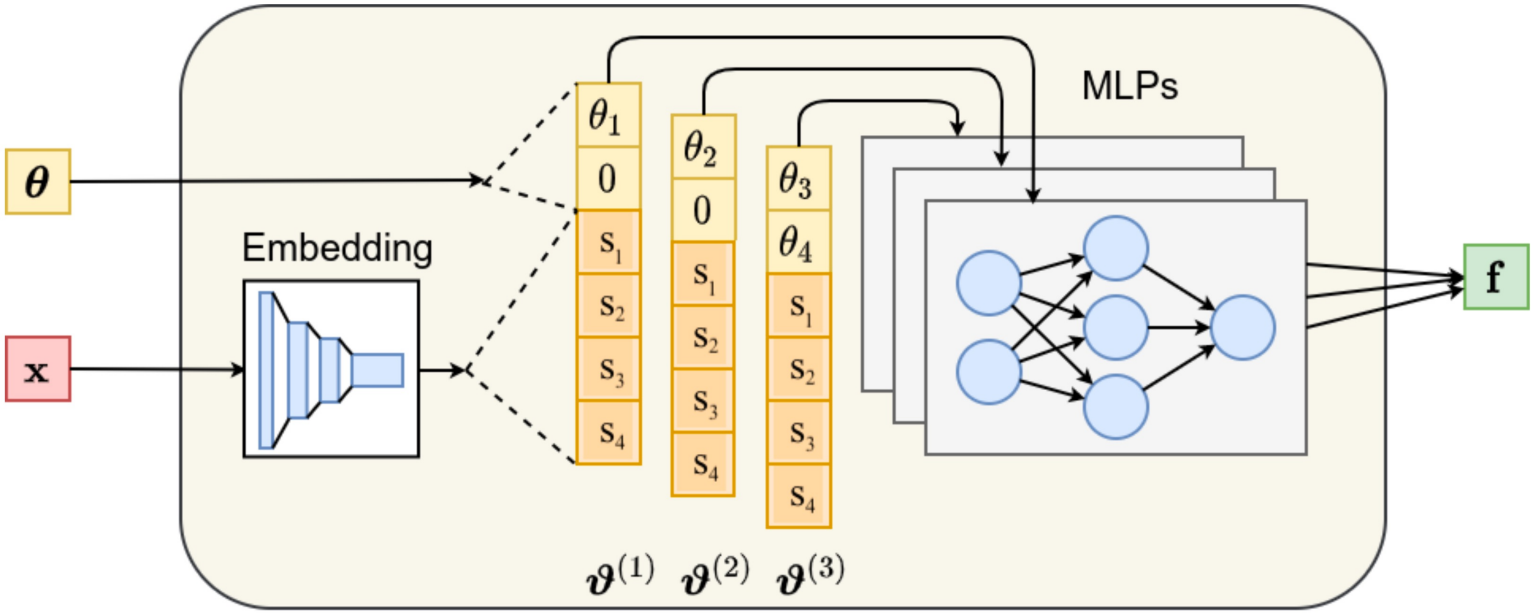
- Introduce a binary random variable y, which classifies

$$\tilde{p}(\boldsymbol{x}, \boldsymbol{\theta} \mid y) = \begin{cases} p(\boldsymbol{x}, \boldsymbol{\theta}) & \text{if } y = 1 \\ p(\boldsymbol{x})p(\boldsymbol{\theta}) & \text{if } y = 0 \end{cases}$$

- Take a classifier $\tilde{p}(y = 1 \mid \boldsymbol{x}, \boldsymbol{\theta})$
- Compute likelihood trick - *blackboard*
- Train it with all DL machinery developed for classifiers

Marginal NRE (MNRE)

- Nuisance parameters
  - Often we don't care about them
- Important parameters
  - Often we're interested only in 1D and 2D posteriors



Cole+2022

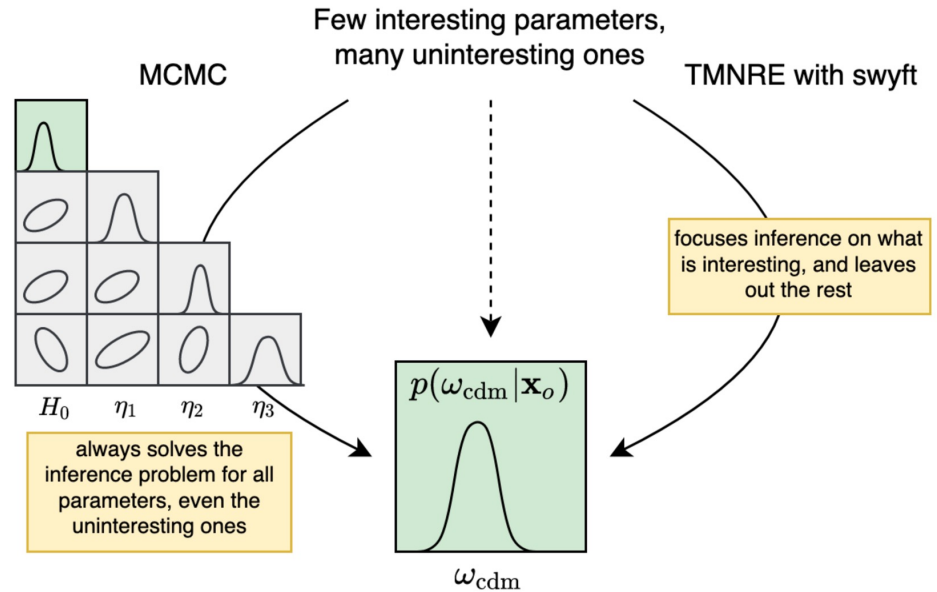
Marginal NRE (MNRE)

- For example, we're interested in only CDM energy density
- All other parameters are not important for us

$$p(\omega_{cdm} \mid \boldsymbol{x}) = \int d\boldsymbol{\eta}\, p(\omega_{cdm}, \eta_1, \ldots, \eta_N \mid \boldsymbol{x})$$

- Running the full inference can be arbitrarily expensive, depending on the model complexity

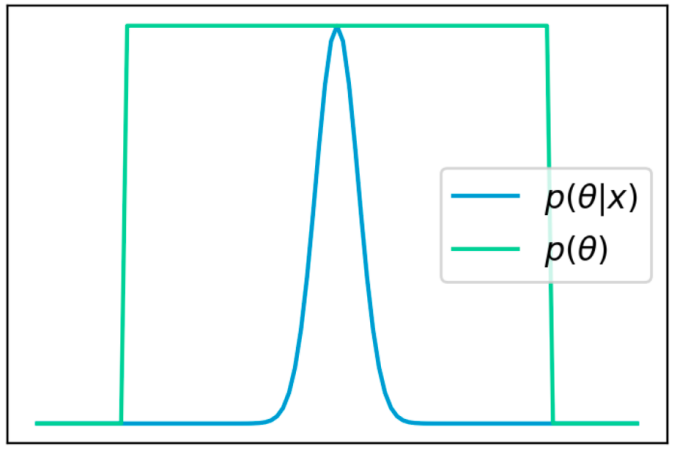
- Importantly, prior on η must be good!
- For other parameters important,
  but not crucial

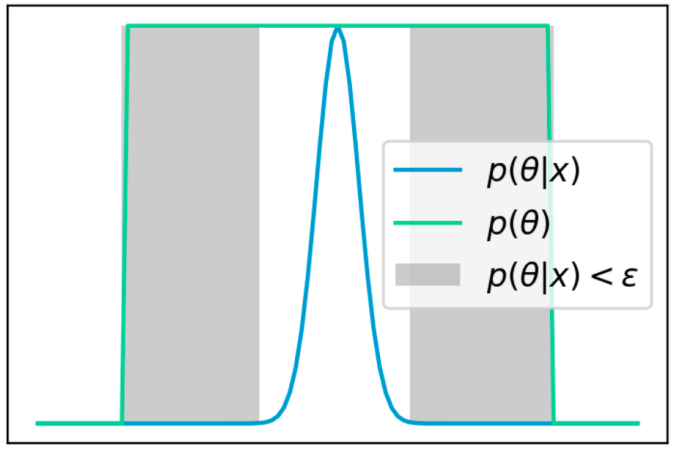


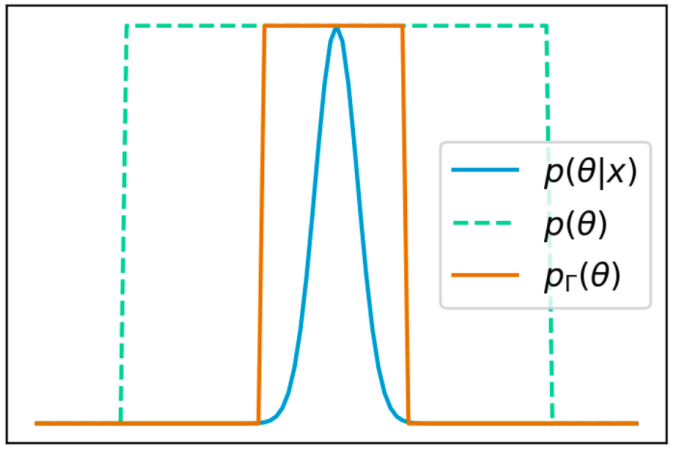
Cole+2022

Truncated Marginal NRE (TMNRE)

- Similar to the sequential (active) learning for NLE
- In several iterations focusing on the relevant prior volume
- Controlled by the posterior estimation around some x_0



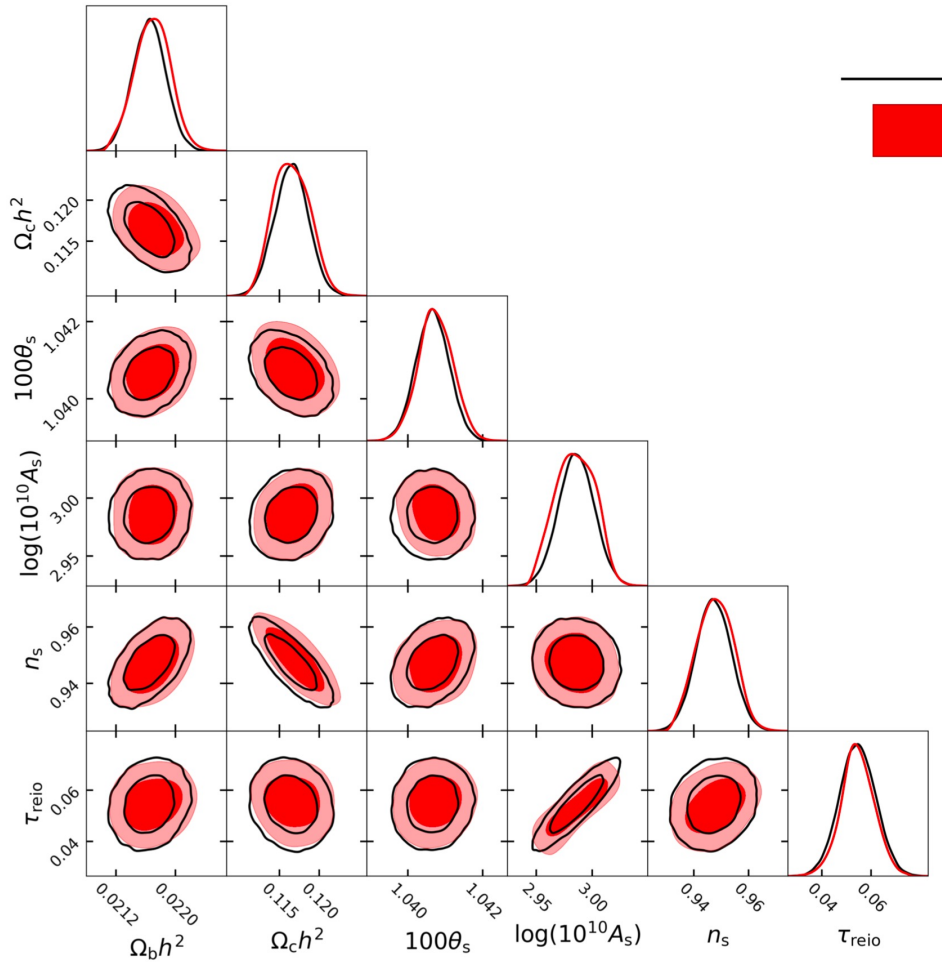


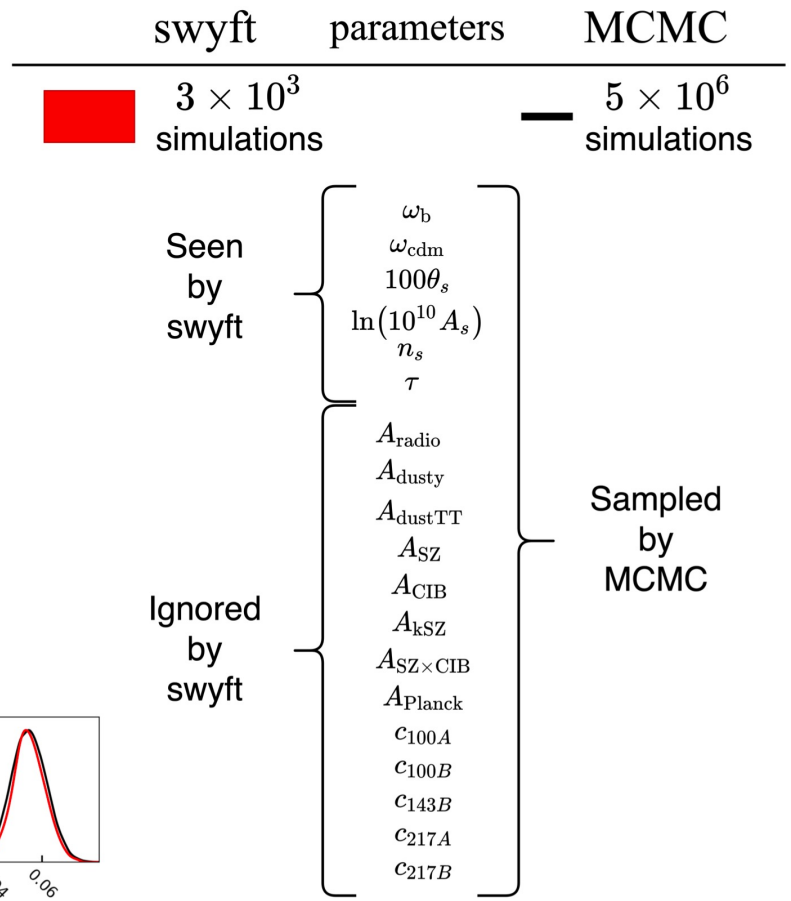


Cole+2022

Truncated Marginal NRE (TMNRE)

- Example: MCMC vs TMNRE for CMB

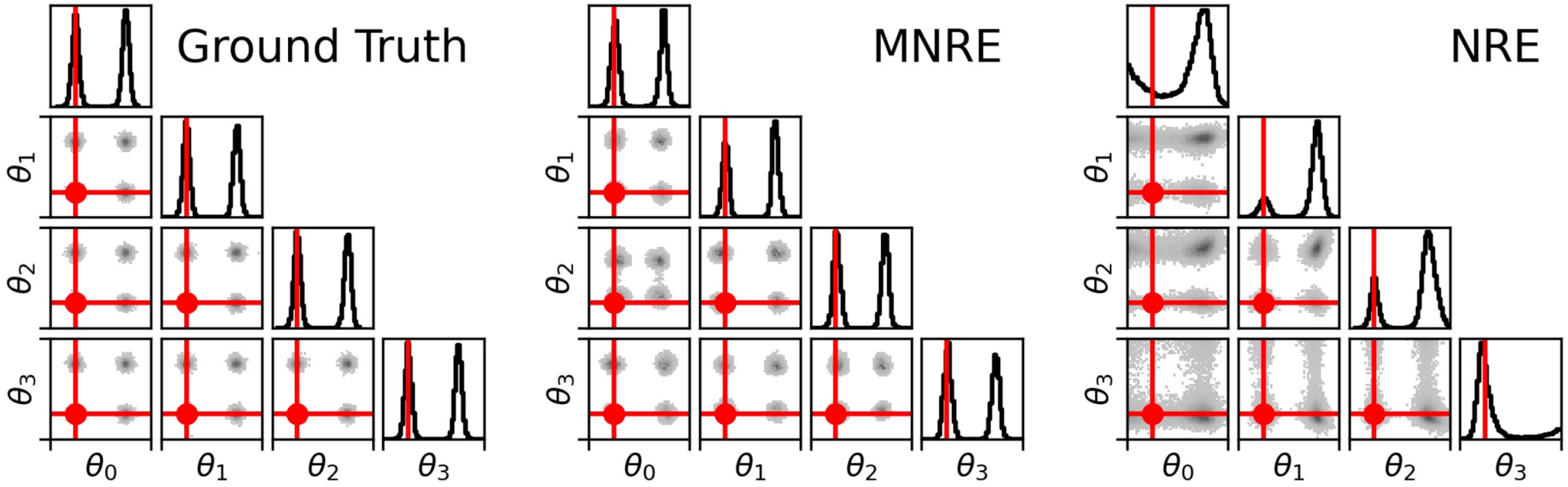




Cole+2022

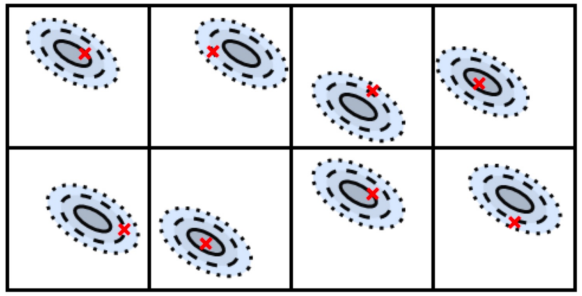
Truncated Marginal NRE (TMNRE)

- Example: 10-dim eggbox

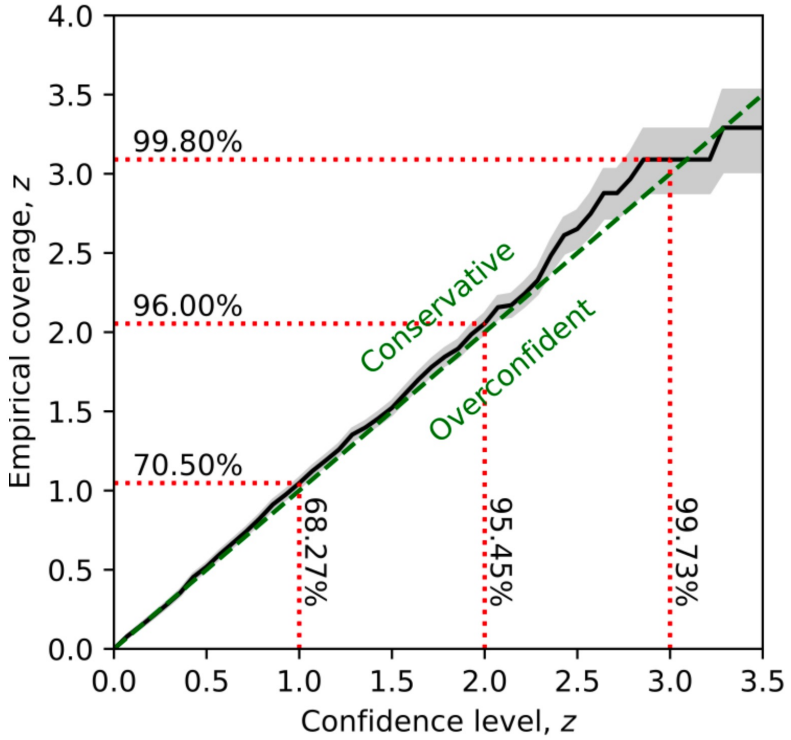


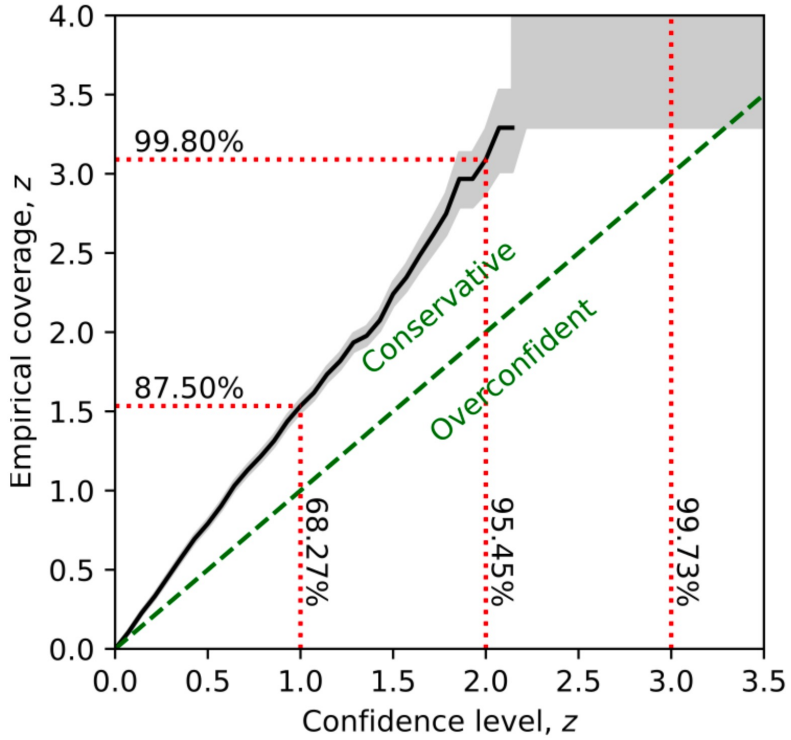
Kurt Miller+2021

TMNRE coverage test



- SBC could be used to test posteriors in equivalent way as for NLE
- Another idea is to take all posteriors, and calculate confidence region for the true sample

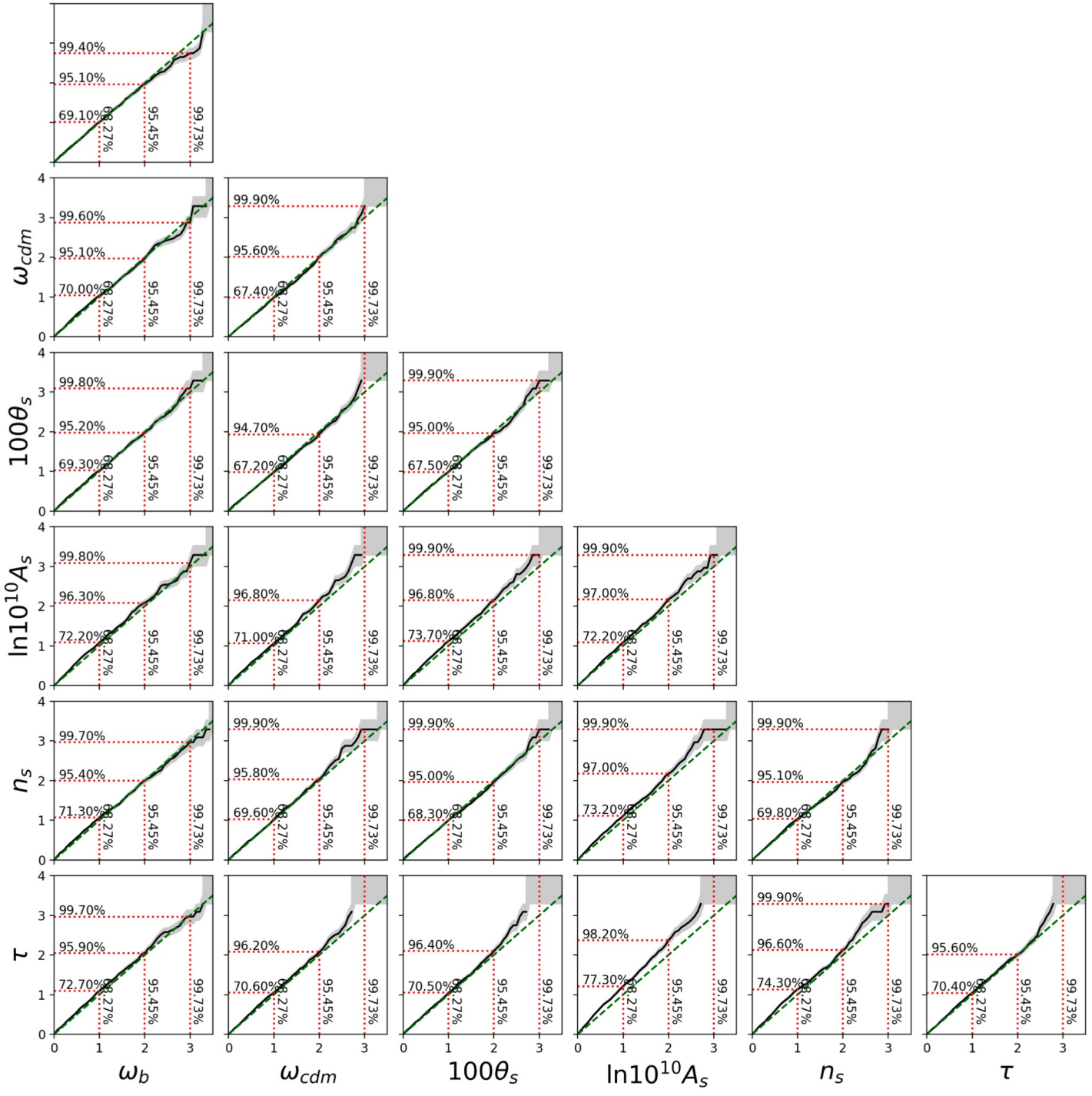




Cole+2022

TMNRE coverage test

- Example: CMB
  - It can be run for any 1D or 2D posterior



Cole+2022

TMNRE vs. (Sequential) NLE

Similarities:

- ***Amortization*** – trained on full prior; once trained, any data can be passed
- ***Sampling*** – after training, both methods efficiently sample posterior (MCMC)
- ***Compression*** – both can have advantage of it, but trained differently
- ***Marginalization*** – both can target marginal posteriors (NLE with "nuisance-hardened" compression)
- ***Active learning*** – truncation vs. sequential posteriors / acquisition function

Differences:

- ***Ratio vs. density estimation*** – classifier is usually easier to train than a density estimator; only with a density estimator one can sample likelihood
- ***Marginalization and active learning interplay*** – truncation might work better for nuisance parameters than nuisance-hardened compression