

# Next-generation statistical inference tools

Simulation-based inference, marginal statistics & accelerated nested sampling

Will Handley

[<wh260@cam.ac.uk>](mailto:wh260@cam.ac.uk)

Royal Society University Research Fellow  
Institute of Astronomy, University of Cambridge  
Kavli Institute for Cosmology, Cambridge  
Gonville & Caius College  
[willhandley.co.uk/talks](http://willhandley.co.uk/talks)

13<sup>th</sup> November 2024



UNIVERSITY OF  
CAMBRIDGE

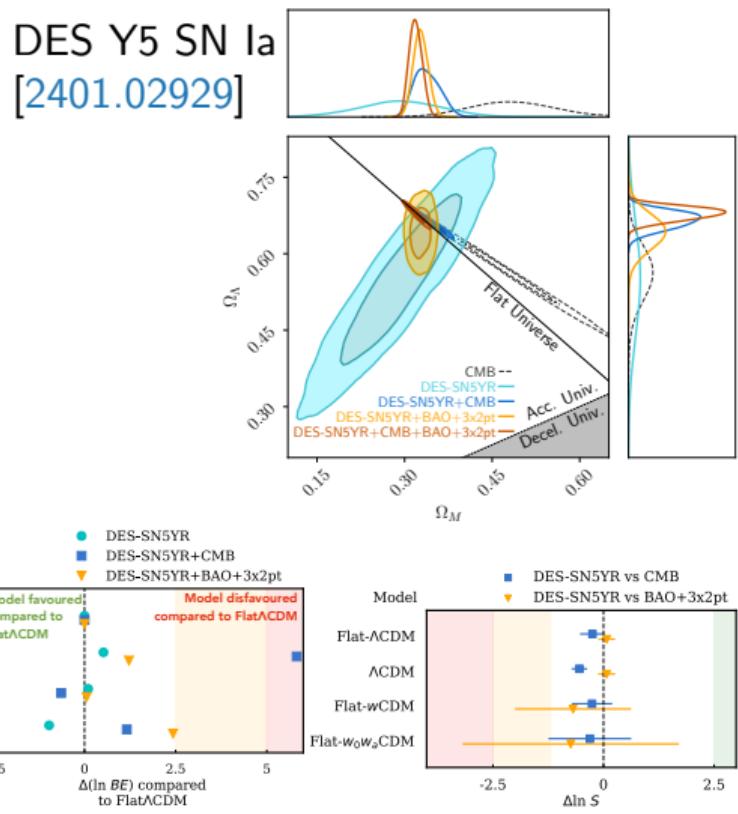


# LBI: Likelihood-based inference

The standard approach if you are fortunate enough to have a likelihood function  $P(D|\theta)$ :

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}$$

1. Define prior  $\pi(\theta)$ 
  - ▶ spend some time being philosophical
2. Sample posterior  $P(\theta|D)$ 
  - ▶ use out-of-the-box MCMC tools such as emcee or MultiNest
  - ▶ make some triangle plots
3. Optionally compute evidence  $\mathcal{Z}(D)$ 
  - ▶ e.g. nested sampling or parallel tempering
  - ▶ do some model comparison (i.e. science)
  - ▶ talk about tensions



# LBI: Likelihood-based inference

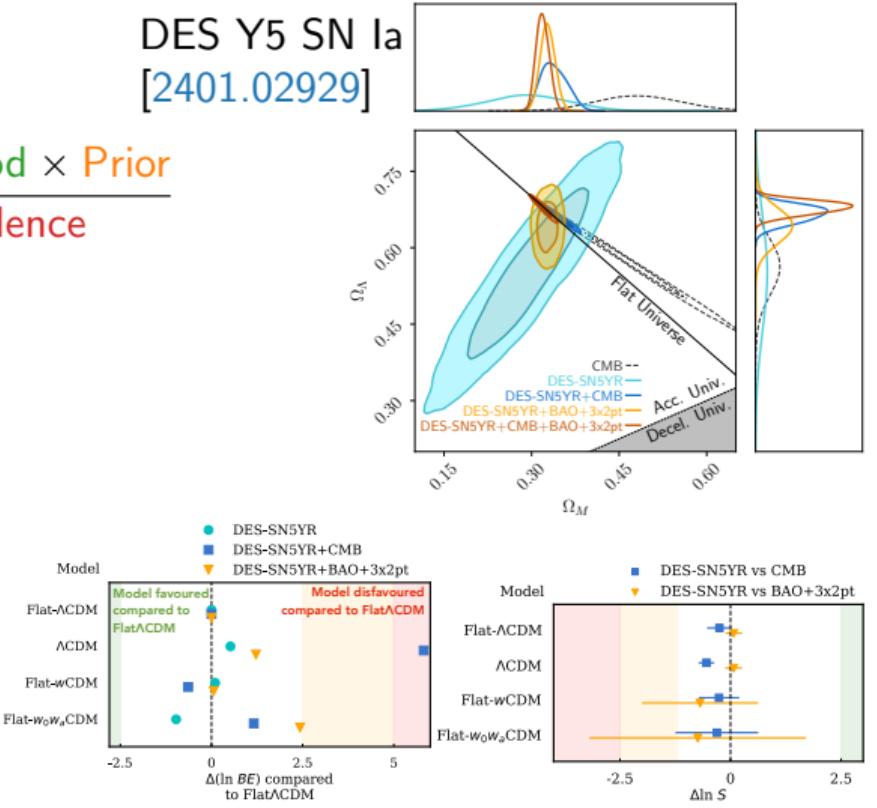
The standard approach if you are fortunate enough to have a likelihood function  $P(D|\theta)$ :

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}$$

$$\text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Evidence}}$$

1. Define prior  $\pi(\theta)$ 
  - ▶ spend some time being philosophical
2. Sample posterior  $P(\theta|D)$ 
  - ▶ use out-of-the-box MCMC tools such as emcee or MultiNest
  - ▶ make some triangle plots
3. Optionally compute evidence  $\mathcal{Z}(D)$ 
  - ▶ e.g. nested sampling or parallel tempering
  - ▶ do some model comparison (i.e. science)
  - ▶ talk about tensions

DES Y5 SN Ia  
[2401.02929]



# LBI: Likelihood-based inference

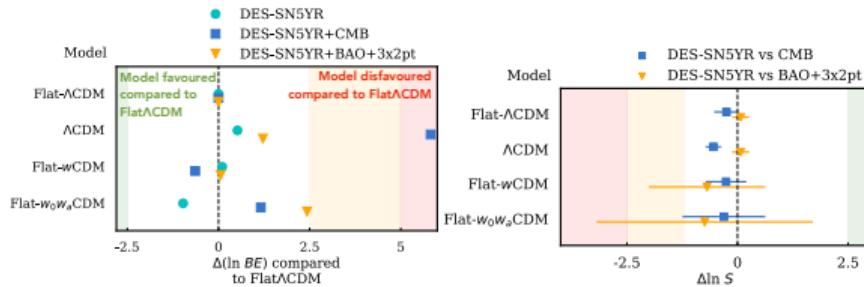
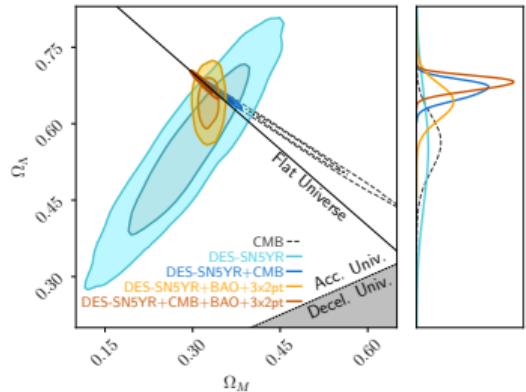
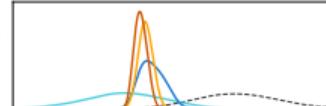
The standard approach if you are fortunate enough to have a likelihood function  $\mathcal{L}(D|\theta)$ :

$$\mathcal{P}(\theta|D) = \frac{\mathcal{L}(D|\theta)\pi(\theta)}{\mathcal{Z}(D)}$$

$$\text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Evidence}}$$

1. Define prior  $\pi(\theta)$ 
  - ▶ spend some time being philosophical
2. Sample posterior  $\mathcal{P}(\theta|D)$ 
  - ▶ use out-of-the-box MCMC tools such as emcee or MultiNest
  - ▶ make some triangle plots
3. Optionally compute evidence  $\mathcal{Z}(D)$ 
  - ▶ e.g. nested sampling or parallel tempering
  - ▶ do some model comparison (i.e. science)
  - ▶ talk about tensions

DES Y5 SN Ia  
[2401.02929]

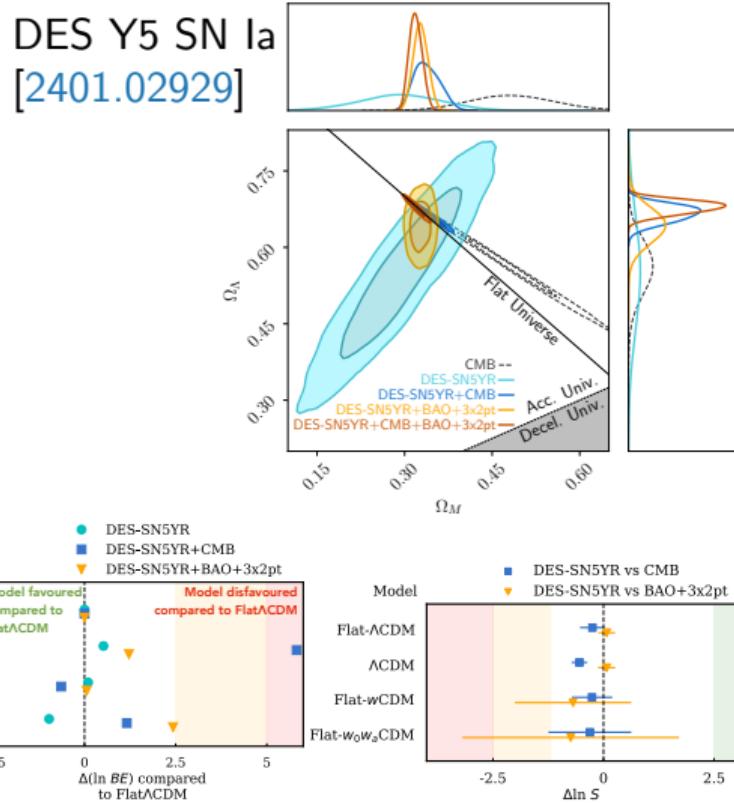


# LBI: Likelihood-based inference

The standard approach if you are fortunate enough to have a likelihood function  $\mathcal{L}(D|\theta)$ :

$$P(\theta|D)P(D) = P(\theta, D) = P(D|\theta)P(\theta),$$

1. Define prior  $\pi(\theta)$ 
  - ▶ spend some time being philosophical
2. Sample posterior  $P(\theta|D)$ 
  - ▶ use out-of-the-box MCMC tools such as emcee or MultiNest
  - ▶ make some triangle plots
3. Optionally compute evidence  $\mathcal{Z}(D)$ 
  - ▶ e.g. nested sampling or parallel tempering
  - ▶ do some model comparison (i.e. science)
  - ▶ talk about tensions

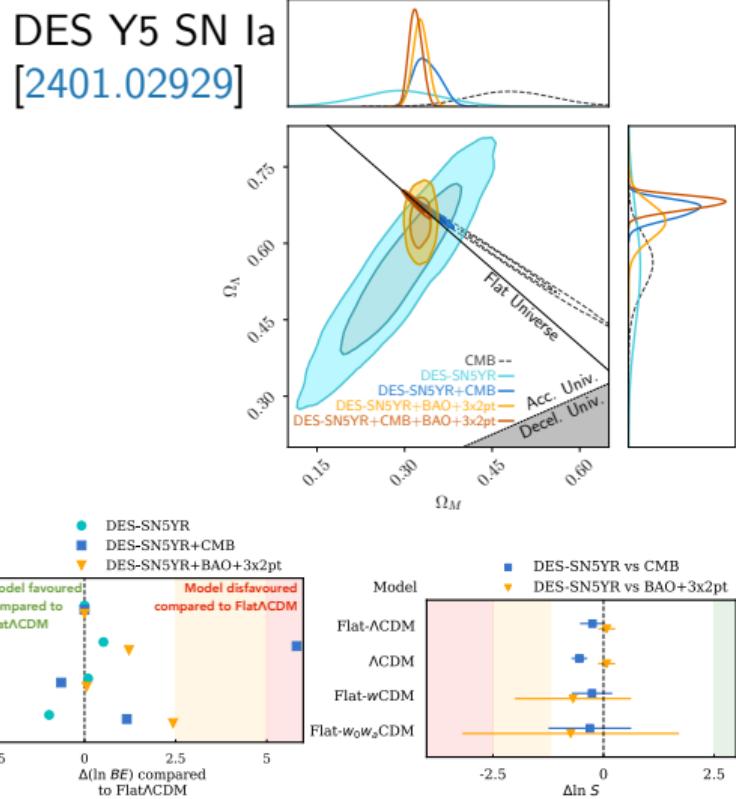


# LBI: Likelihood-based inference

The standard approach if you are fortunate enough to have a likelihood function  $\mathcal{L}(D|\theta)$ :

$$\mathcal{P} \times \mathcal{Z} = \mathcal{J} = \mathcal{L} \times \pi, \quad \text{Joint} = \mathcal{J} = P(\theta, D)$$

1. Define prior  $\pi(\theta)$ 
  - ▶ spend some time being philosophical
2. Sample posterior  $P(\theta|D)$ 
  - ▶ use out-of-the-box MCMC tools such as emcee or MultiNest
  - ▶ make some triangle plots
3. Optionally compute evidence  $\mathcal{Z}(D)$ 
  - ▶ e.g. nested sampling or parallel tempering
  - ▶ do some model comparison (i.e. science)
  - ▶ talk about tensions



# The three pillars of Bayesian inference

## Parameter estimation

What do the data tell us about the parameters of a model?  
e.g. the size or age of a  $\Lambda$ CDM universe

$$P(\theta|D, M) = \frac{P(D|\theta, M)P(\theta|M)}{P(D|M)}$$

$$\mathcal{P} = \frac{\mathcal{L} \times \pi}{\mathcal{Z}}$$

$$\text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Evidence}}$$

## Model comparison

How much does the data support a particular model?  
e.g.  $\Lambda$ CDM vs a dynamic dark energy cosmology

$$P(M|D) = \frac{P(D|M)P(M)}{P(D)}$$

$$\frac{\mathcal{Z}_M \Pi_M}{\sum_m \mathcal{Z}_m \Pi_m}$$

$$\text{Posterior} = \frac{\text{Evidence} \times \text{Prior}}{\text{Normalisation}}$$

## Tension quantification

Do different datasets make consistent predictions from the same model? e.g. CMB vs Type IA supernovae data

$$\mathcal{R} = \frac{\mathcal{Z}_{AB}}{\mathcal{Z}_A \mathcal{Z}_B}$$

$$\begin{aligned} \log \mathcal{S} &= \langle \log \mathcal{L}_{AB} \rangle_{\mathcal{P}_{AB}} \\ &\quad - \langle \log \mathcal{L}_A \rangle_{\mathcal{P}_A} \\ &\quad - \langle \log \mathcal{L}_B \rangle_{\mathcal{P}_B} \end{aligned}$$

# The three pillars of Bayesian inference

## Parameter estimation

What do the data tell us about the parameters of a model?  
e.g. *the masses and spins of a BBH collision*

$$P(\theta|D, M) = \frac{P(D|\theta, M)P(\theta|M)}{P(D|M)}$$

$$\mathcal{P} = \frac{\mathcal{L} \times \pi}{\mathcal{Z}}$$

$$\text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Evidence}}$$

## Model comparison

How much does the data support a particular model?  
e.g. *IMRPhenom vs EOBNR waveform models*

$$P(M|D) = \frac{P(D|M)P(M)}{P(D)}$$

$$\frac{\mathcal{Z}_M \Pi_M}{\sum_m \mathcal{Z}_m \Pi_m}$$

$$\text{Posterior} = \frac{\text{Evidence} \times \text{Prior}}{\text{Normalisation}}$$

## Tension quantification

Do different datasets make consistent predictions from the same model? e.g. *Automated glitch detection*

$$\mathcal{R} = \frac{\mathcal{Z}_{AB}}{\mathcal{Z}_A \mathcal{Z}_B}$$

$$\begin{aligned} \log \mathcal{S} &= \langle \log \mathcal{L}_{AB} \rangle_{\mathcal{P}_{AB}} \\ &\quad - \langle \log \mathcal{L}_A \rangle_{\mathcal{P}_A} \\ &\quad - \langle \log \mathcal{L}_B \rangle_{\mathcal{P}_B} \end{aligned}$$

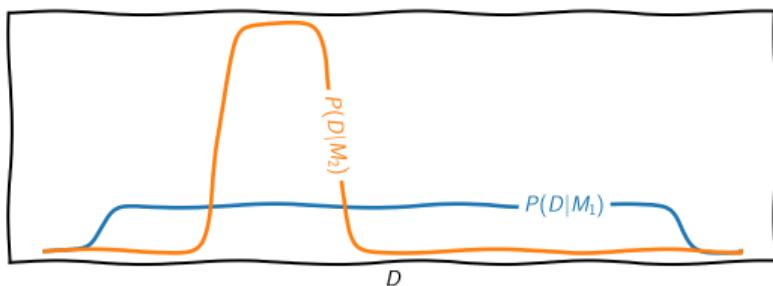
# Model comparison $\mathcal{Z} = P(D|M)$

- ▶ Bayesian model comparison allows mathematical derivation of key philosophical principles.

Viewed from data-space  $D$ :

## Popper's falsificationism

- ▶ Prefer models that make bold predictions.
- ▶ if proven true, model more likely correct.



- ▶ Falsificationism comes from normalisation

Viewed from parameter-space  $\theta$ :

## Occam's razor

- ▶ Models should be as simple as possible
- ▶ ... but no simpler

- ▶ Occam's razor equation:

$$\log \mathcal{Z} = \langle \log \mathcal{L} \rangle_{\mathcal{P}} - \mathcal{D}_{KL}$$

- ▶ “Occam penalty”: KL divergence between prior  $\pi$  and posterior  $\mathcal{P}$ .

$$\mathcal{D}_{KL} \sim \log \frac{\text{Prior volume}}{\text{Posterior volume}}$$

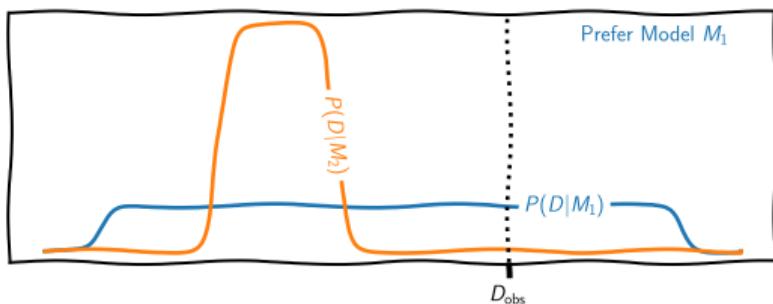
# Model comparison $\mathcal{Z} = P(D|M)$

- ▶ Bayesian model comparison allows mathematical derivation of key philosophical principles.

Viewed from data-space  $D$ :

## Popper's falsificationism

- ▶ Prefer models that make bold predictions.
- ▶ if proven true, model more likely correct.



- ▶ Falsificationism comes from normalisation

Viewed from parameter-space  $\theta$ :

## Occam's razor

- ▶ Models should be as simple as possible
- ▶ ... but no simpler

- ▶ Occam's razor equation:

$$\log \mathcal{Z} = \langle \log \mathcal{L} \rangle_{\mathcal{P}} - \mathcal{D}_{\text{KL}}$$

- ▶ “Occam penalty”: KL divergence between prior  $\pi$  and posterior  $\mathcal{P}$ .

$$\mathcal{D}_{\text{KL}} \sim \log \frac{\text{Prior volume}}{\text{Posterior volume}}$$

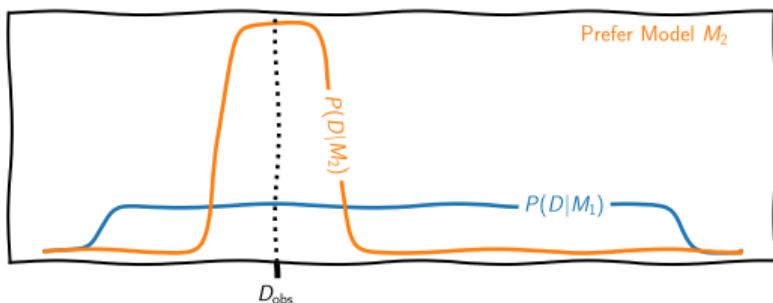
# Model comparison $\mathcal{Z} = P(D|M)$

- ▶ Bayesian model comparison allows mathematical derivation of key philosophical principles.

Viewed from data-space  $D$ :

## Popper's falsificationism

- ▶ Prefer models that make bold predictions.
- ▶ if proven true, model more likely correct.



- ▶ Falsificationism comes from normalisation

Viewed from parameter-space  $\theta$ :

## Occam's razor

- ▶ Models should be as simple as possible
- ▶ ... but no simpler

- ▶ Occam's razor equation:

$$\log \mathcal{Z} = \langle \log \mathcal{L} \rangle_{\mathcal{P}} - \mathcal{D}_{\text{KL}}$$

- ▶ “Occam penalty”: KL divergence between prior  $\pi$  and posterior  $\mathcal{P}$ .

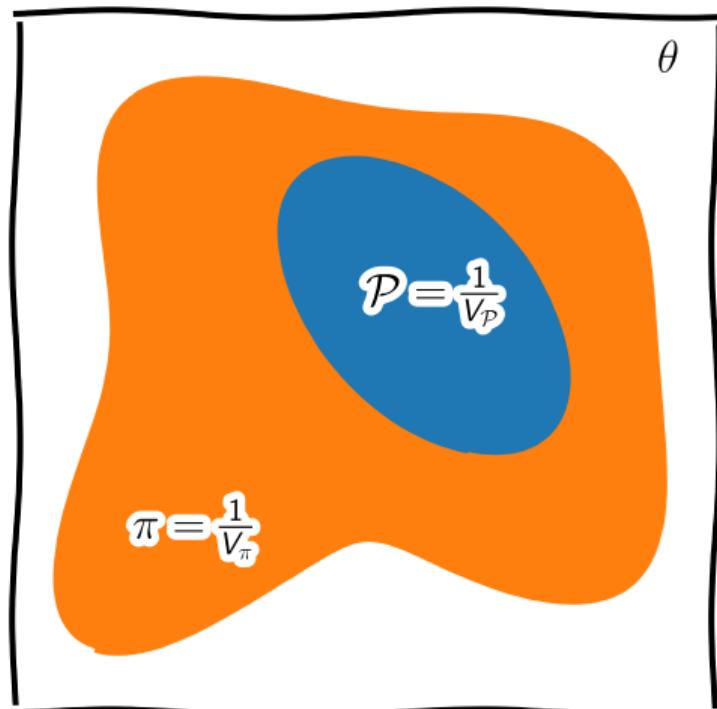
$$\mathcal{D}_{\text{KL}} \sim \log \frac{\text{Prior volume}}{\text{Posterior volume}}$$

# Why do sampling?

- ▶ The cornerstone of numerical Bayesian inference is working with **samples**.
- ▶ Generate a set of representative parameters drawn in proportion to the posterior  $\theta \sim \mathcal{P}$ .
- ▶ The magic of marginalisation  $\Rightarrow$  perform usual analysis on each sample in turn.
- ▶ The golden rule is **stay in samples** until the last moment before computing summary statistics/triangle plots because

$$f(\langle X \rangle) \neq \langle f(X) \rangle$$

- ▶ Generally need  $\sim \mathcal{O}(12)$  independent samples to compute a value and error bar.

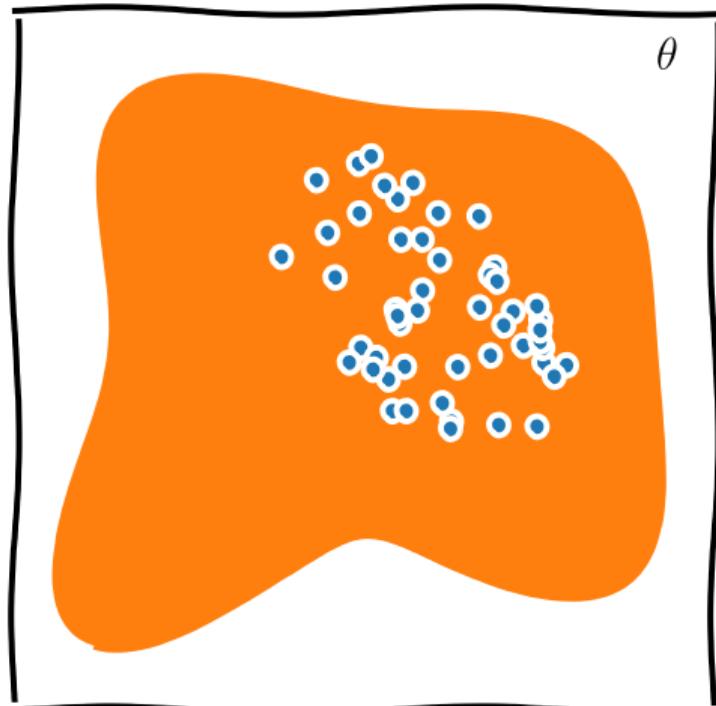


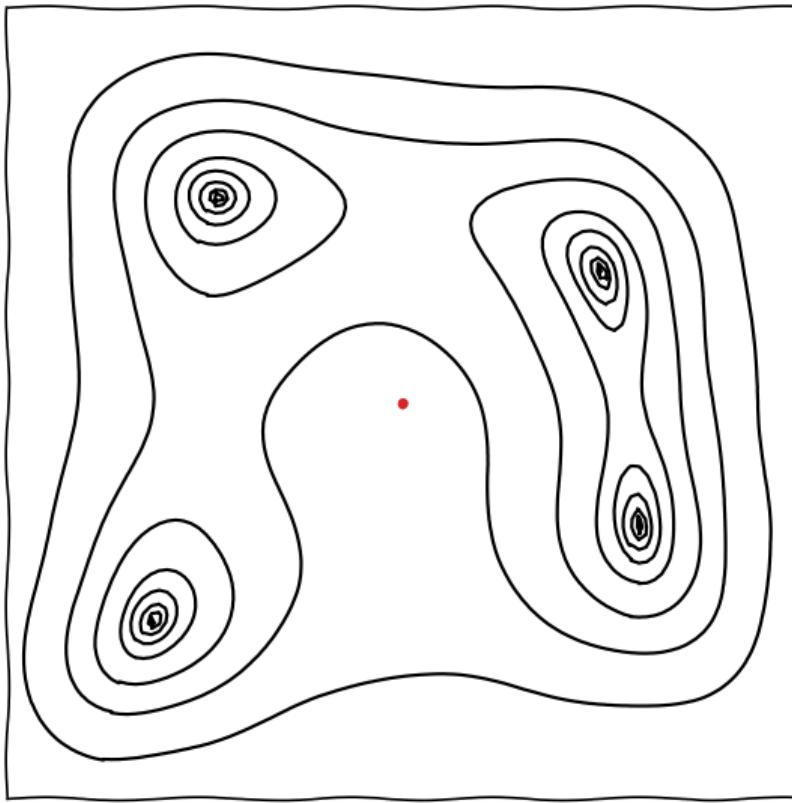
# Why do sampling?

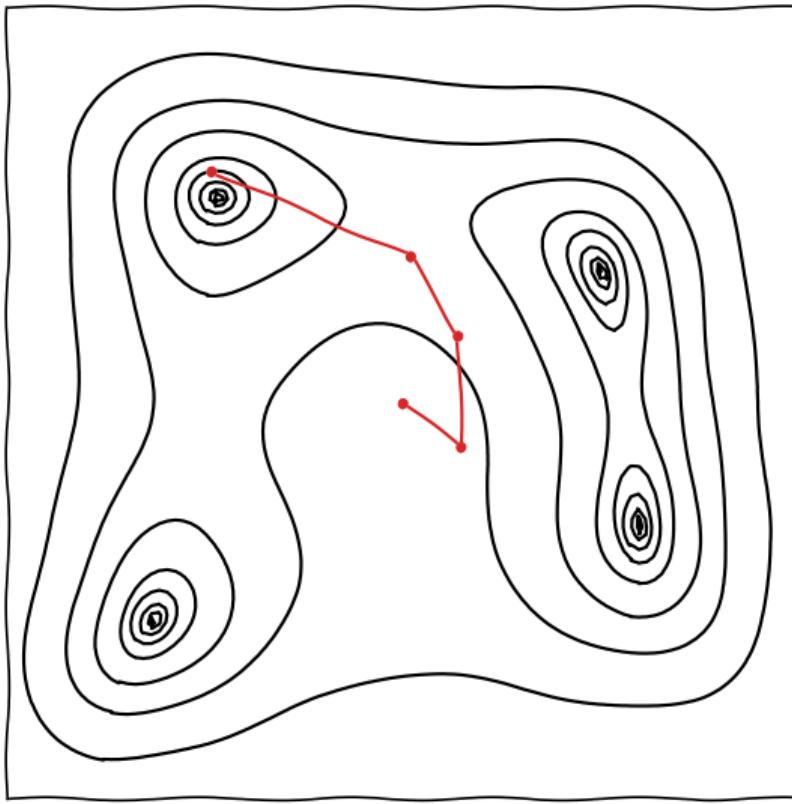
- ▶ The cornerstone of numerical Bayesian inference is working with **samples**.
- ▶ Generate a set of representative parameters drawn in proportion to the posterior  $\theta \sim \mathcal{P}$ .
- ▶ The magic of marginalisation  $\Rightarrow$  perform usual analysis on each sample in turn.
- ▶ The golden rule is **stay in samples** until the last moment before computing summary statistics/triangle plots because

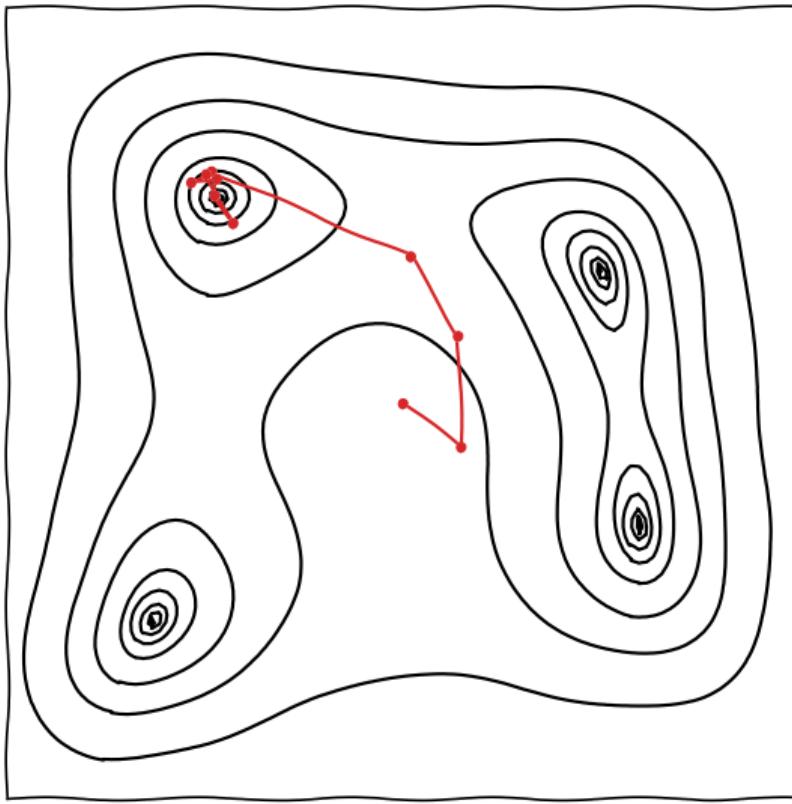
$$f(\langle X \rangle) \neq \langle f(X) \rangle$$

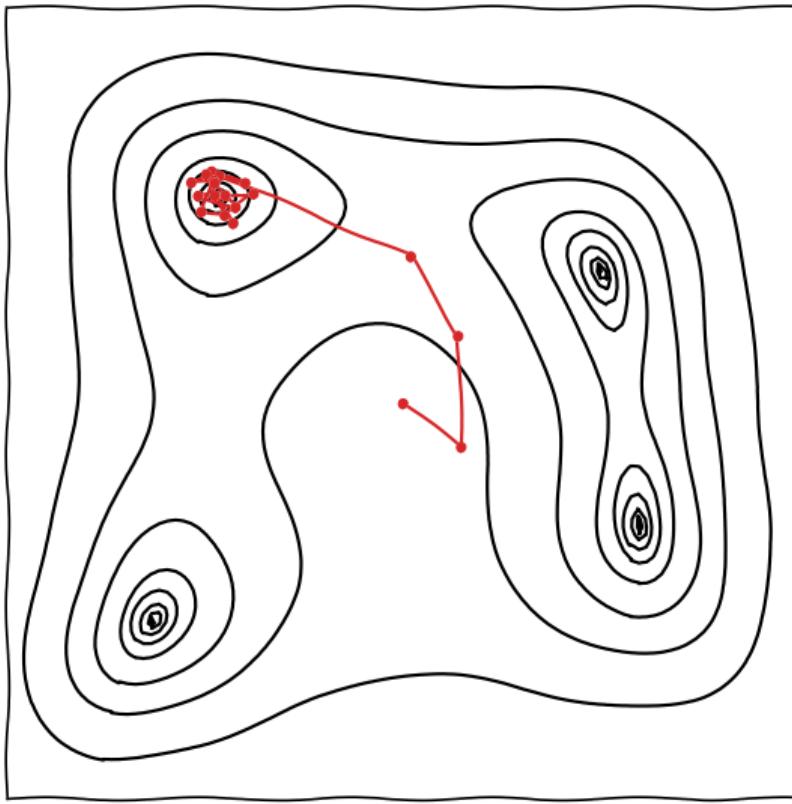
- ▶ Generally need  $\sim \mathcal{O}(12)$  independent samples to compute a value and error bar.

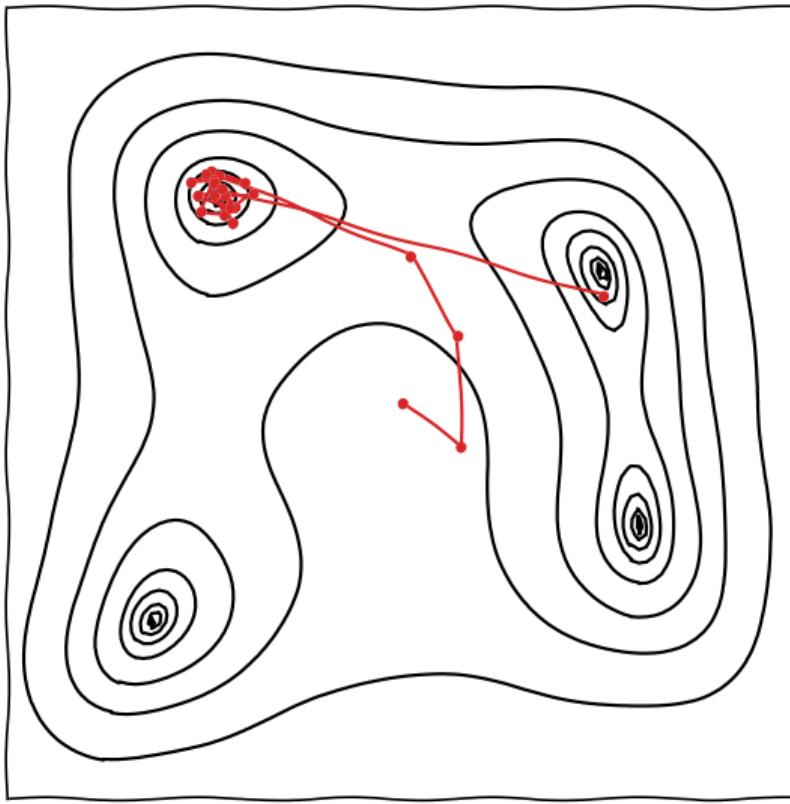


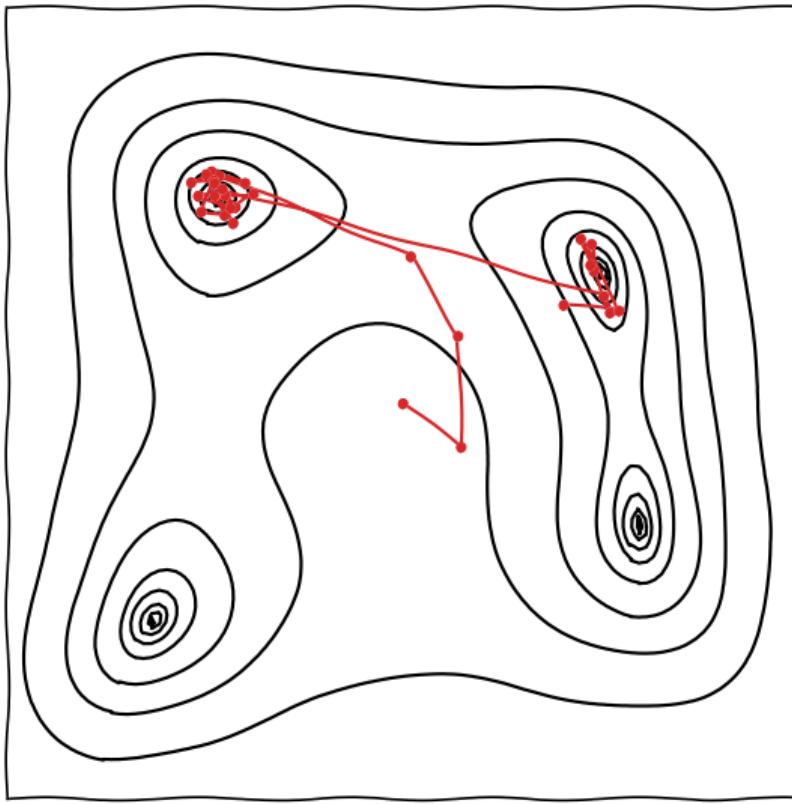




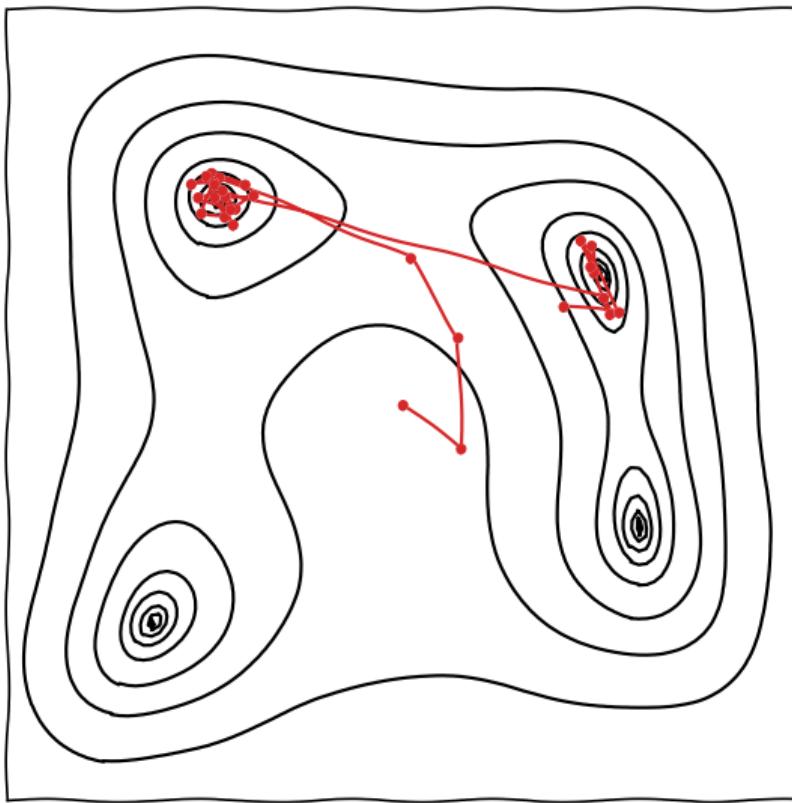




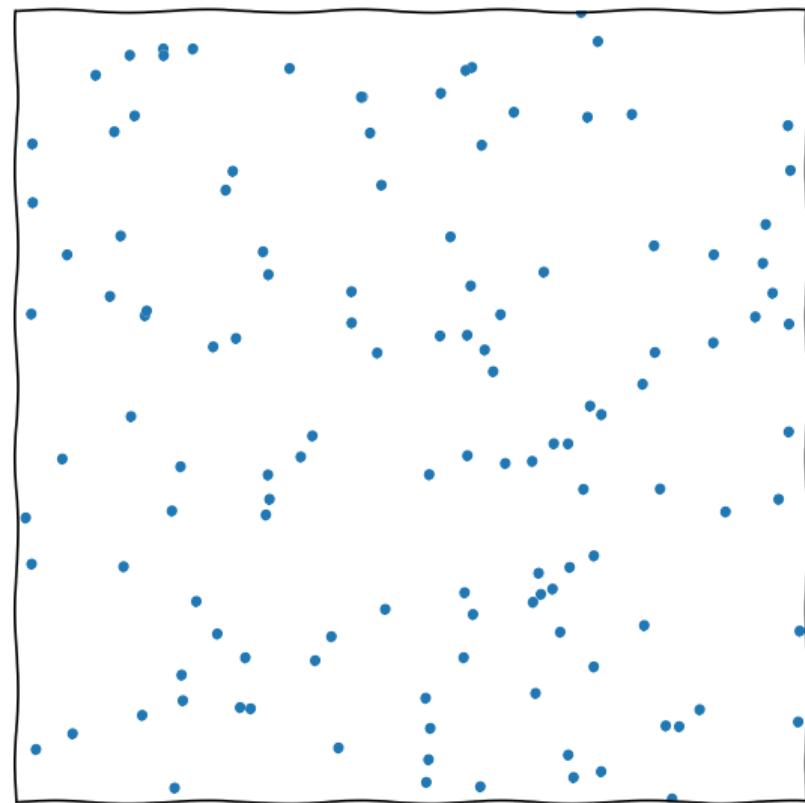




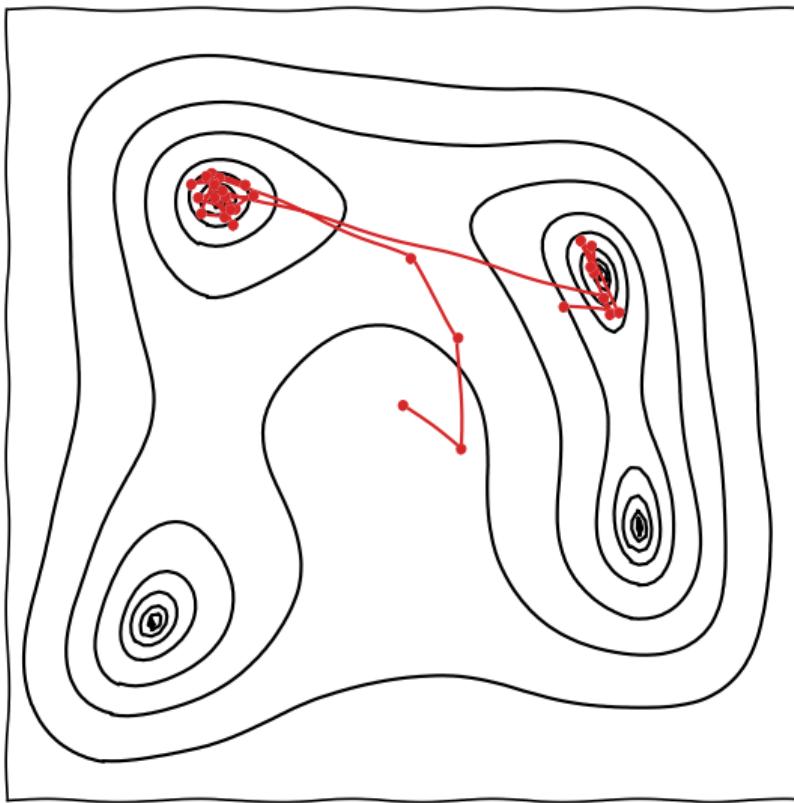
## MCMC



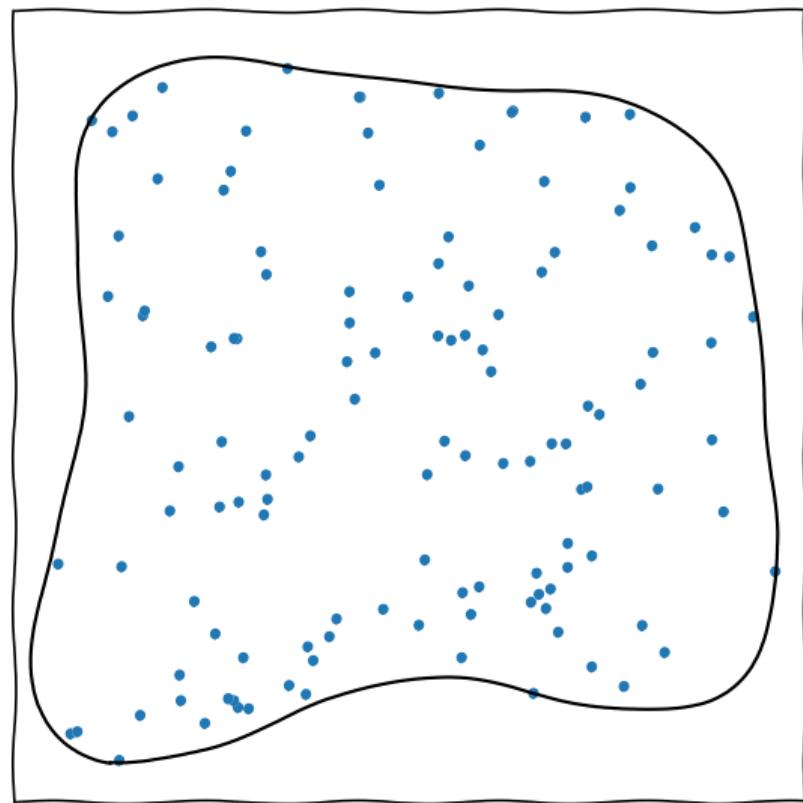
## Nested sampling



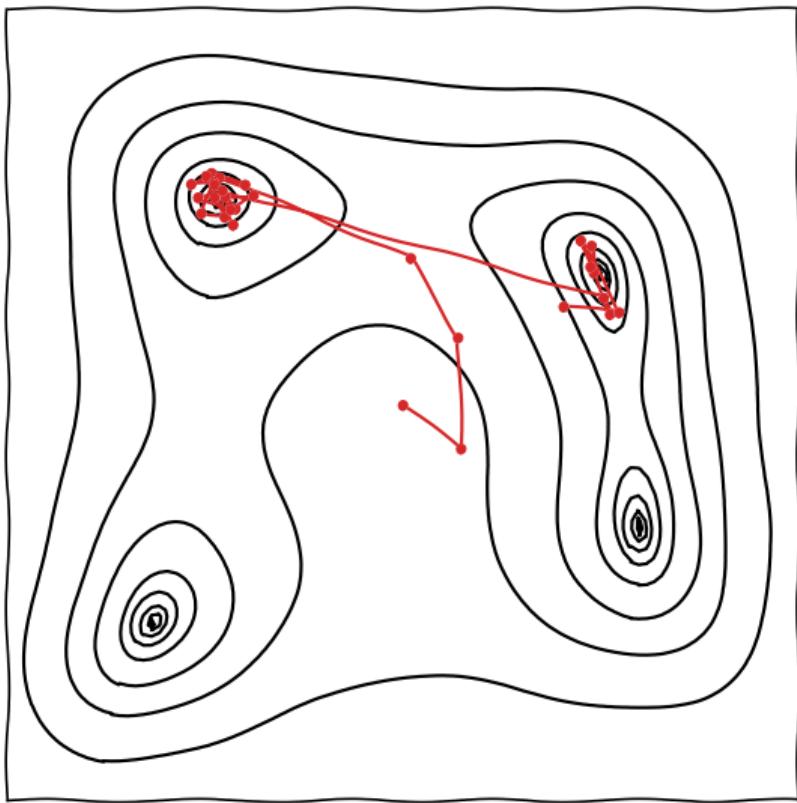
## MCMC



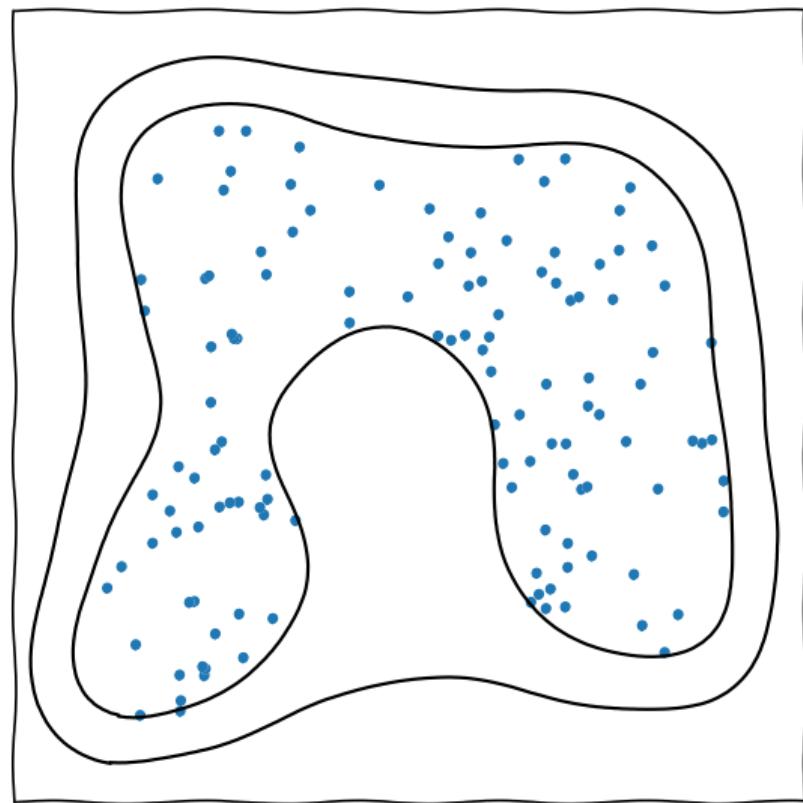
## Nested sampling



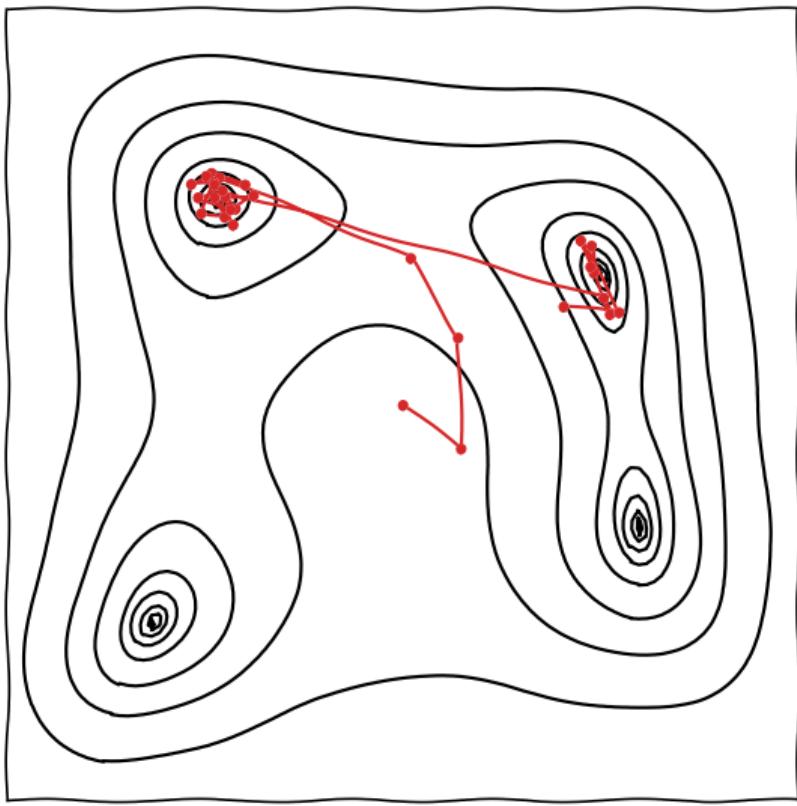
## MCMC



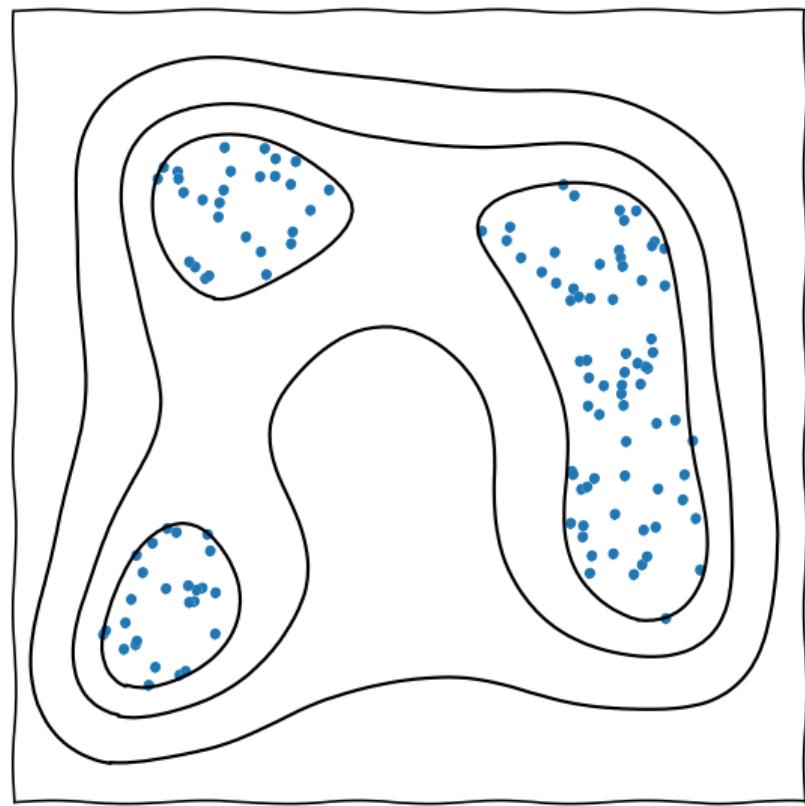
## Nested sampling



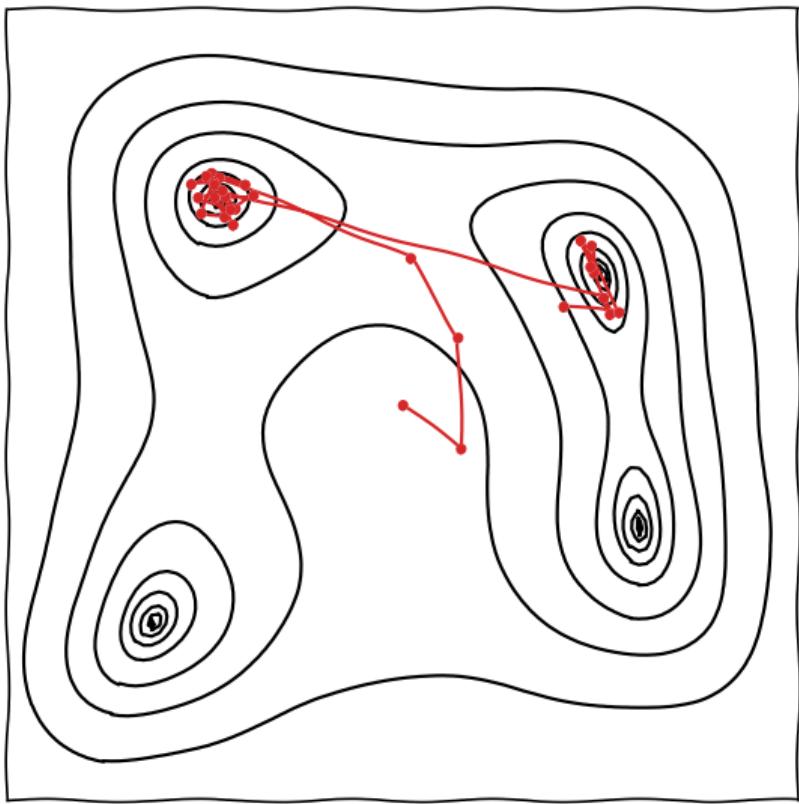
## MCMC



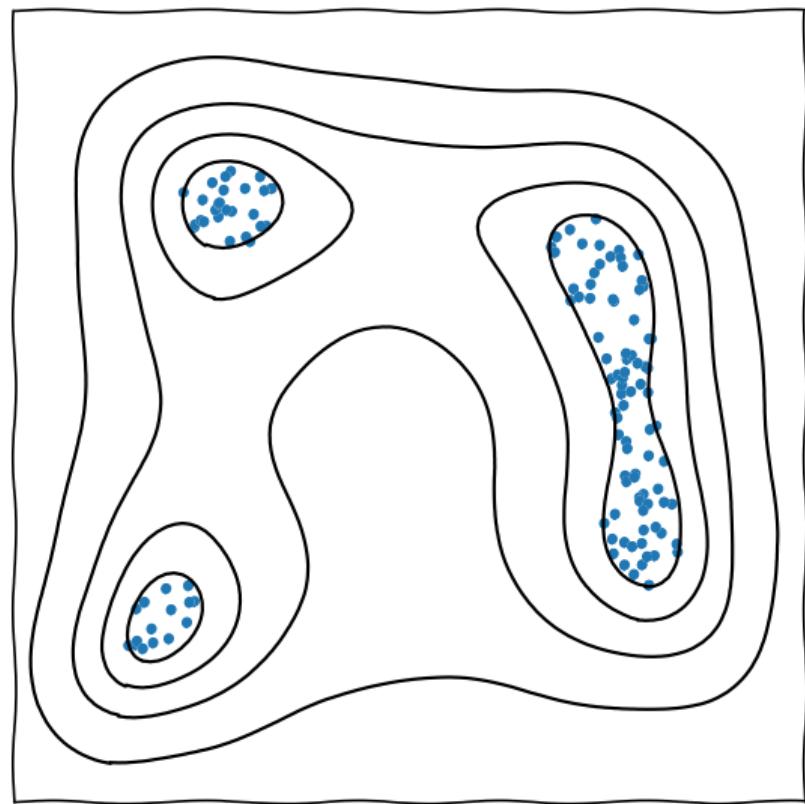
## Nested sampling



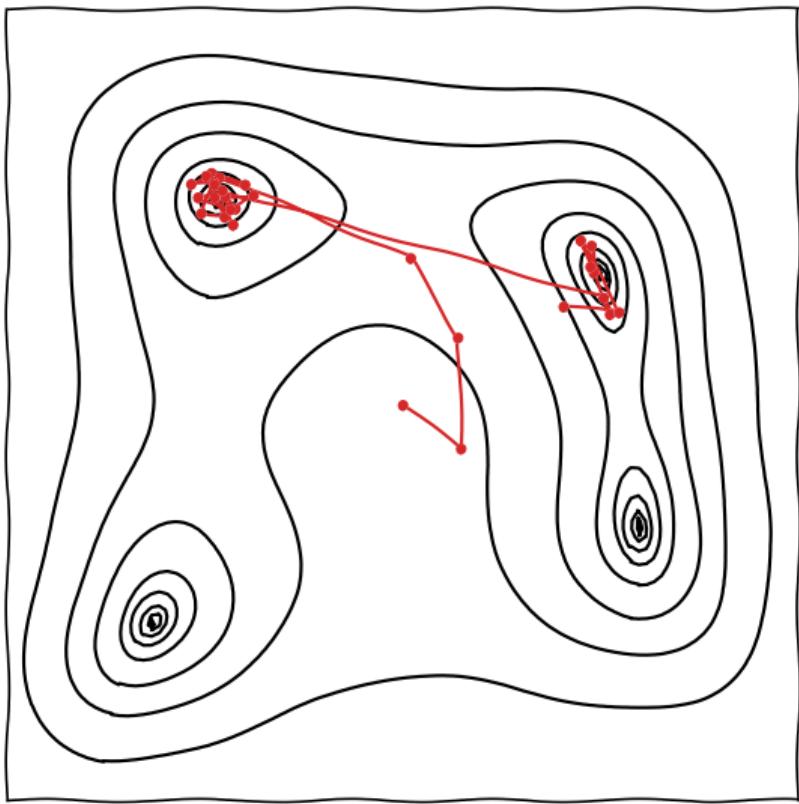
## MCMC



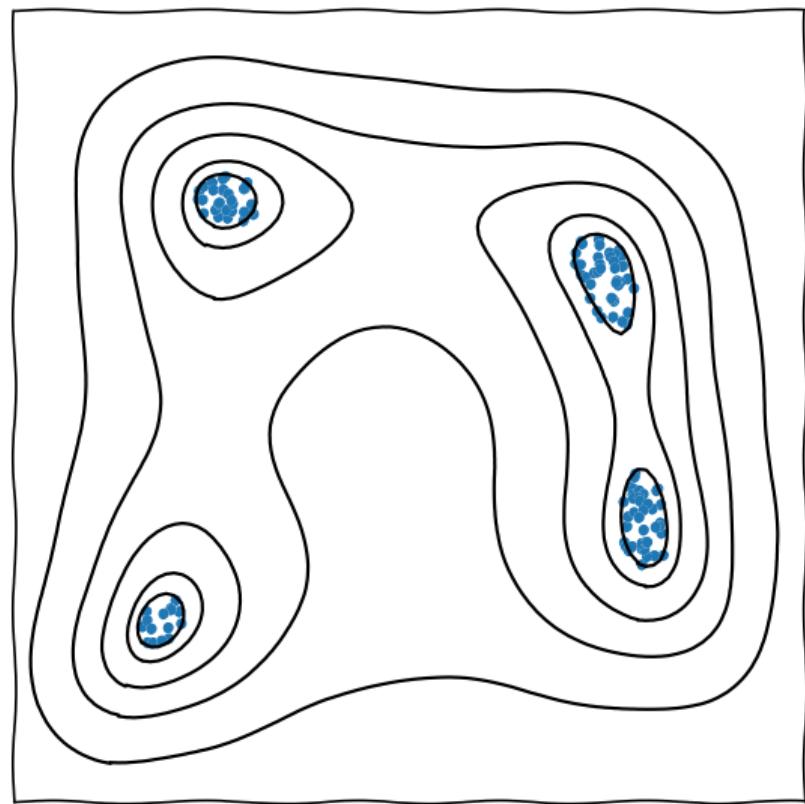
## Nested sampling



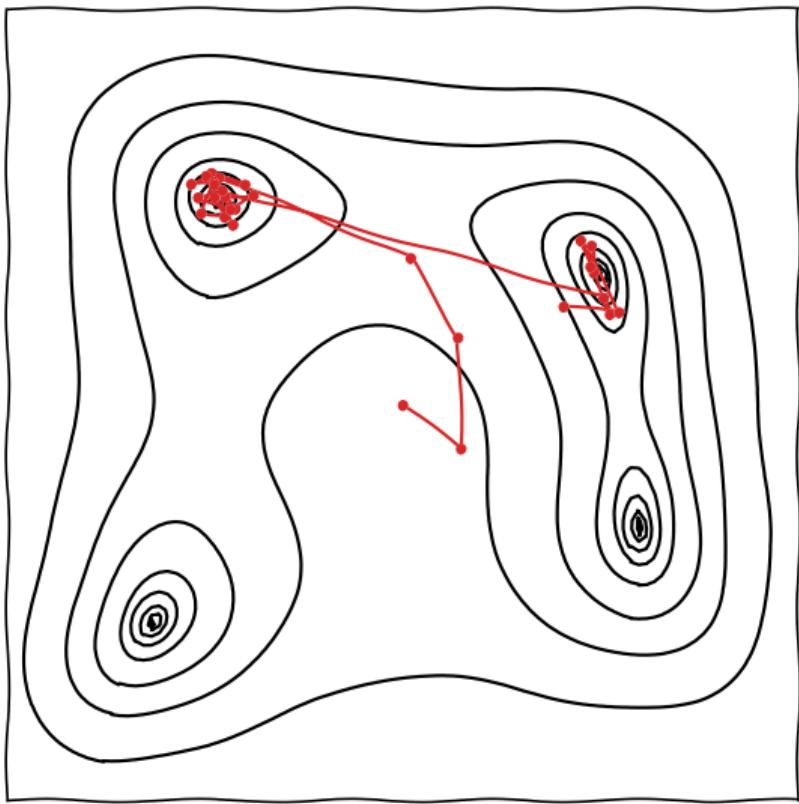
## MCMC



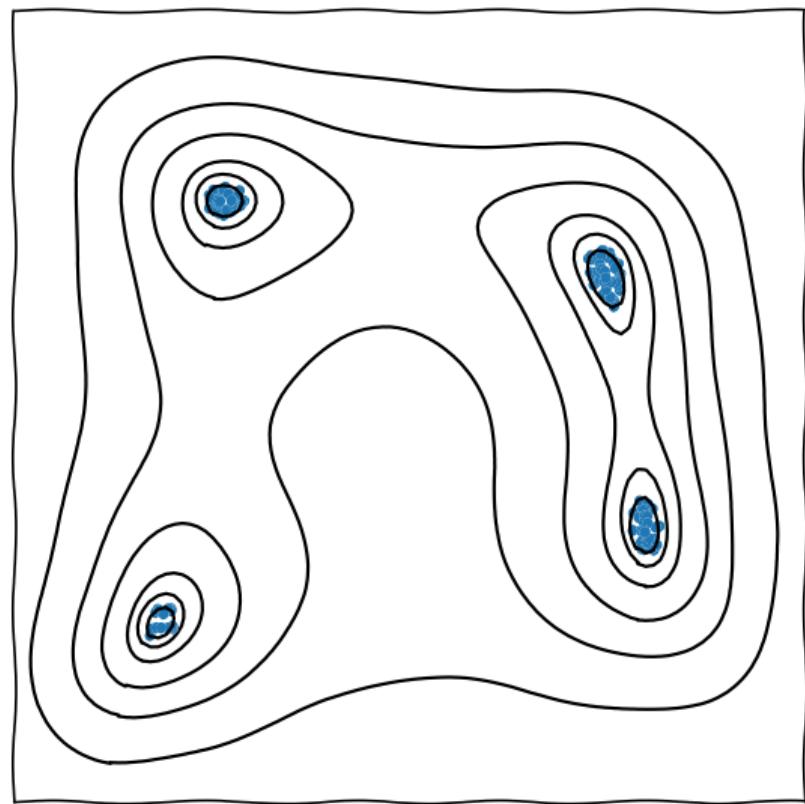
## Nested sampling



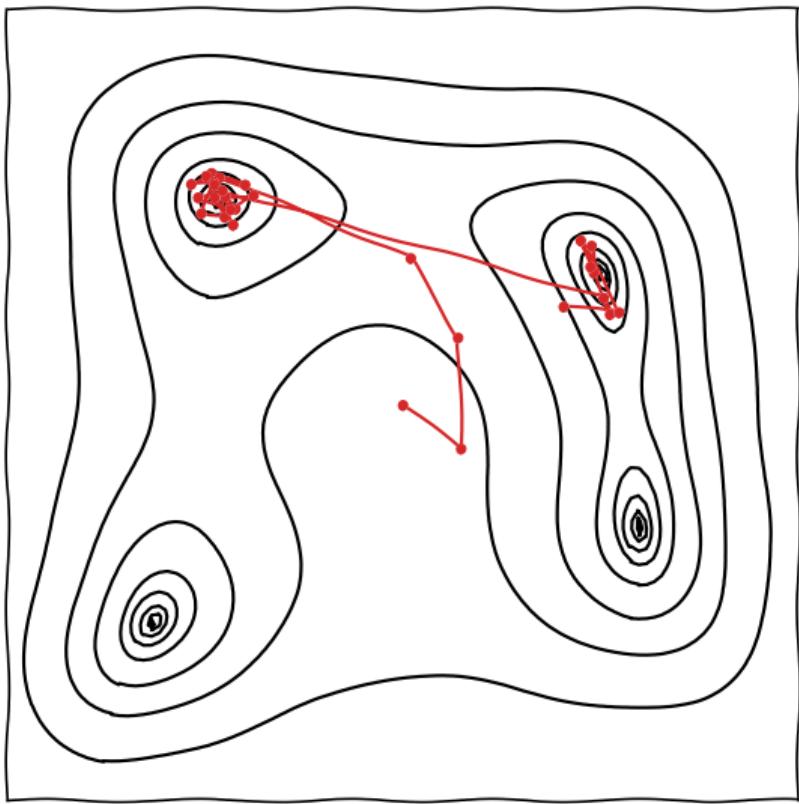
## MCMC



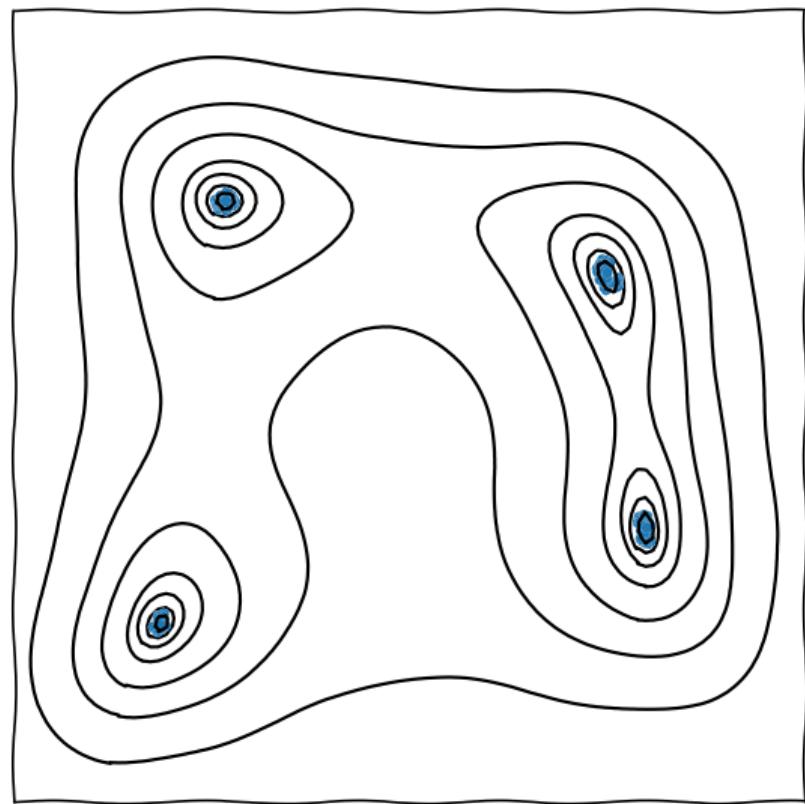
## Nested sampling



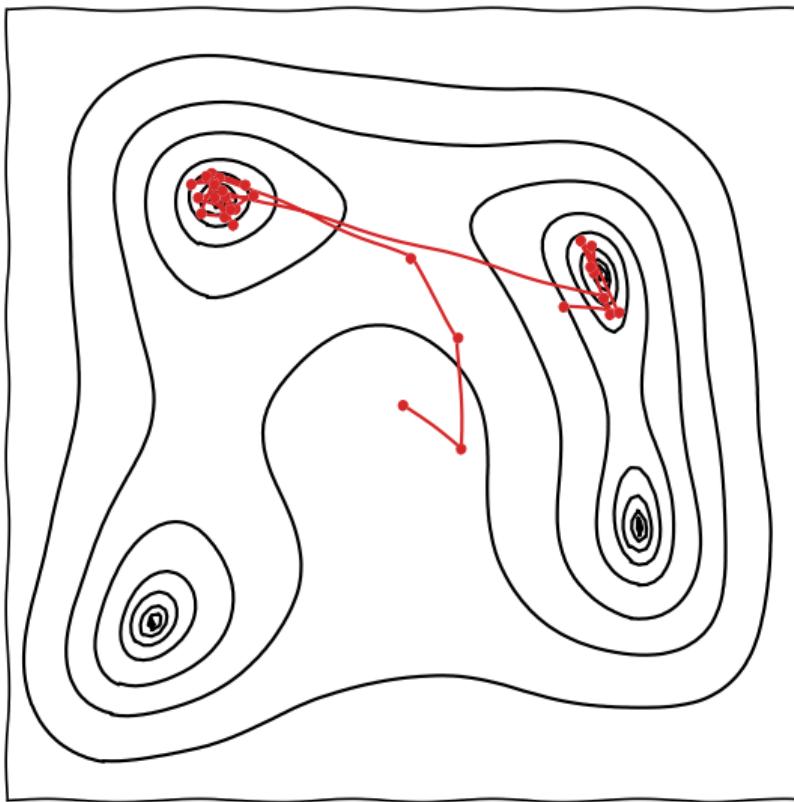
## MCMC



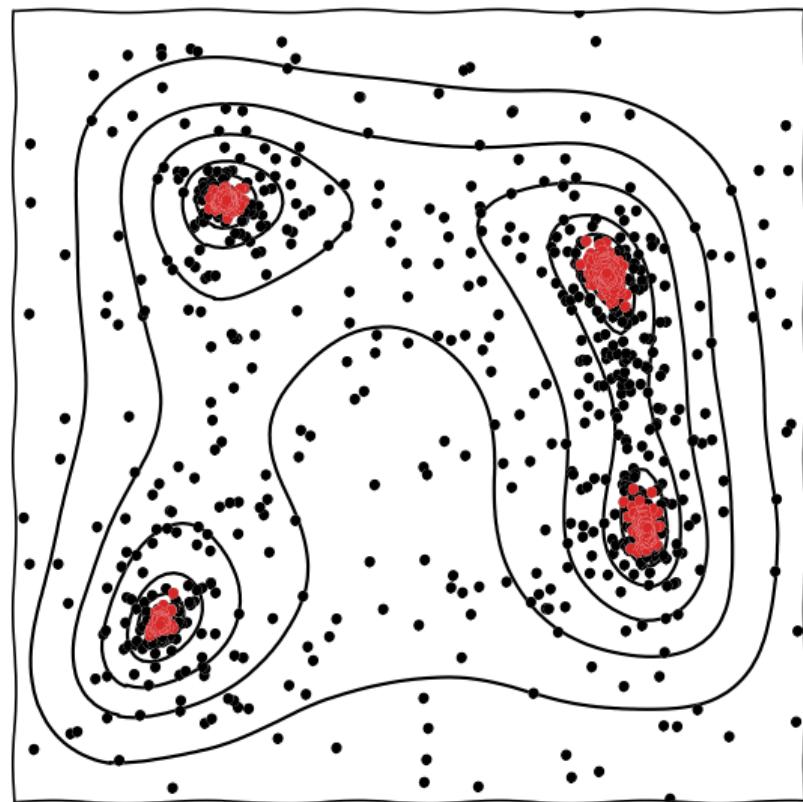
## Nested sampling



## MCMC

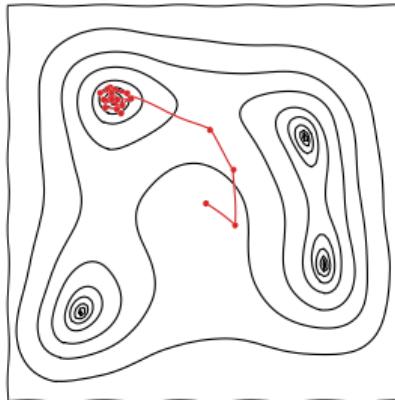


## Nested sampling



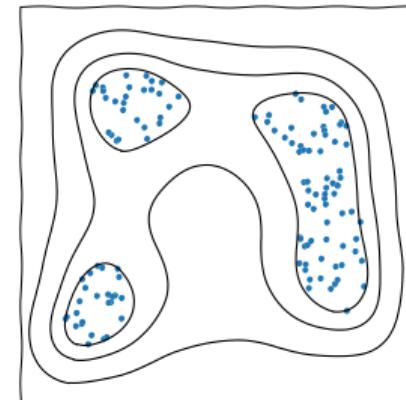
## MCMC

- ▶ Single “walker”
- ▶ Explores posterior
- ▶ Fast, if proposal matrix is tuned
- ▶ Parameter estimation, suspiciousness calculation
- ▶ Channel capacity optimised for generating posterior samples



## Nested sampling

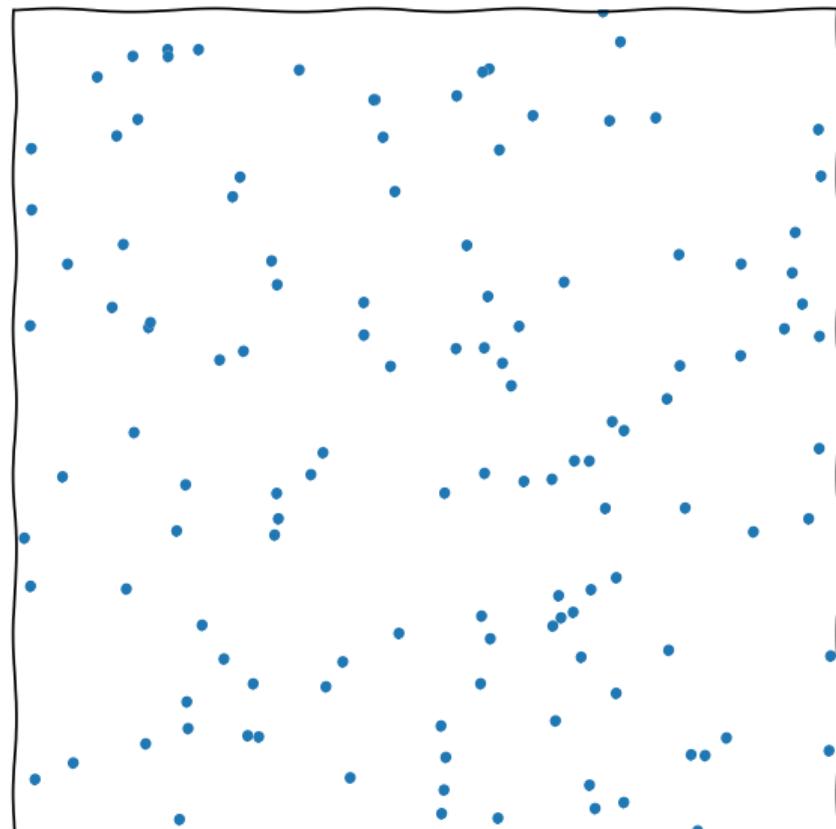
- ▶ Ensemble of “live points”
- ▶ Scans from prior to peak of likelihood
- ▶ Slower, no tuning required
- ▶ Parameter estimation, model comparison, tension quantification
- ▶ Channel capacity optimised for computing partition function



## The nested sampling meta-algorithm: live points

- ▶ Start with  $n$  random samples over the space.
- ▶ Delete outermost sample, and replace with a new random one at higher integrand value.
- ▶ The “live points” steadily contract around the peak(s) of the function.
- ▶ We can use this evolution to estimate volume *probabilistically*.
- ▶ At each iteration, the contours contract by  $\sim \frac{1}{n}$  of their volume.
- ▶ This is an exponential contraction, so

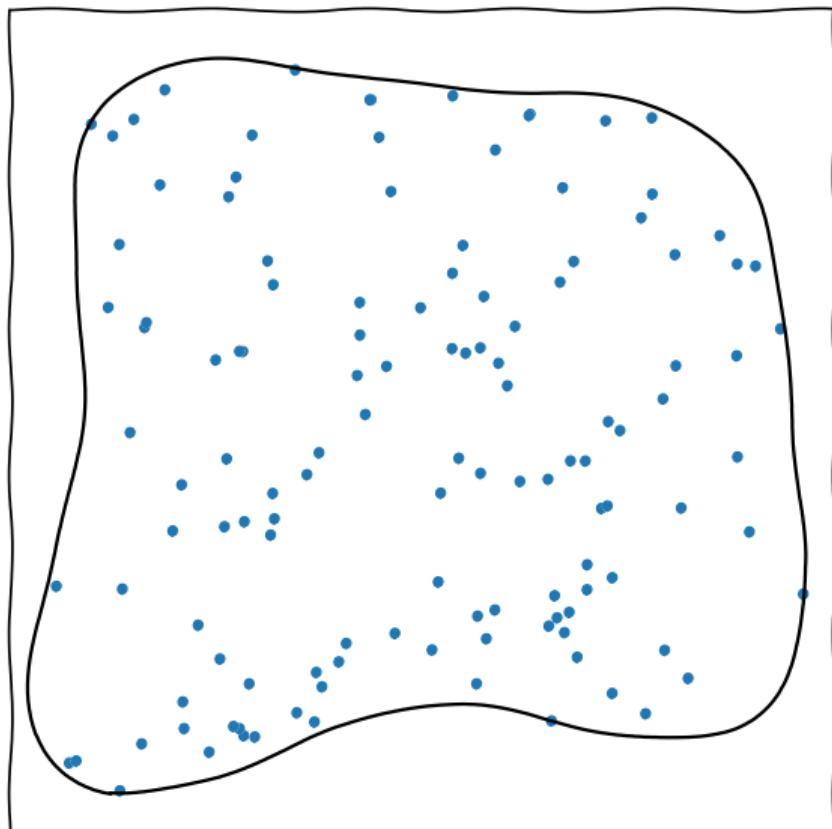
$$\int f(x)dV \approx \sum_i f(x_i)\Delta V_i, \quad V_i = V_0 e^{-i/n}$$



## The nested sampling meta-algorithm: live points

- ▶ Start with  $n$  random samples over the space.
- ▶ Delete outermost sample, and replace with a new random one at higher integrand value.
- ▶ The “live points” steadily contract around the peak(s) of the function.
- ▶ We can use this evolution to estimate volume *probabilistically*.
- ▶ At each iteration, the contours contract by  $\sim \frac{1}{n}$  of their volume.
- ▶ This is an exponential contraction, so

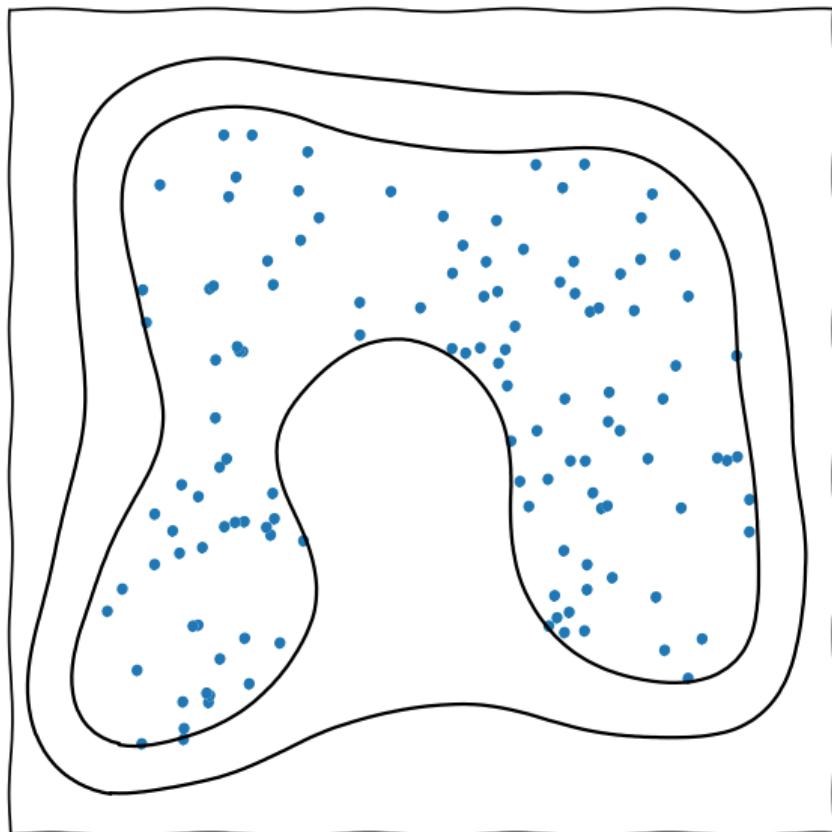
$$\int f(x)dV \approx \sum_i f(x_i)\Delta V_i, \quad V_i = V_0 e^{-i/n}$$



## The nested sampling meta-algorithm: live points

- ▶ Start with  $n$  random samples over the space.
- ▶ Delete outermost sample, and replace with a new random one at higher integrand value.
- ▶ The “live points” steadily contract around the peak(s) of the function.
- ▶ We can use this evolution to estimate volume *probabilistically*.
- ▶ At each iteration, the contours contract by  $\sim \frac{1}{n}$  of their volume.
- ▶ This is an exponential contraction, so

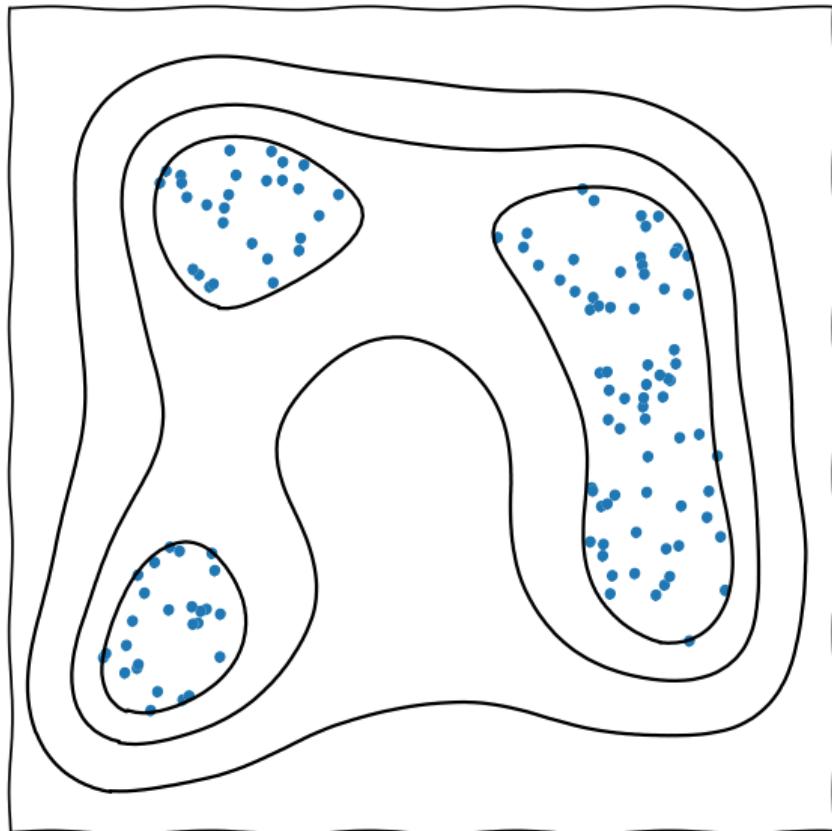
$$\int f(x)dV \approx \sum_i f(x_i)\Delta V_i, \quad V_i = V_0 e^{-i/n}$$



## The nested sampling meta-algorithm: live points

- ▶ Start with  $n$  random samples over the space.
- ▶ Delete outermost sample, and replace with a new random one at higher integrand value.
- ▶ The “live points” steadily contract around the peak(s) of the function.
- ▶ We can use this evolution to estimate volume *probabilistically*.
- ▶ At each iteration, the contours contract by  $\sim \frac{1}{n}$  of their volume.
- ▶ This is an exponential contraction, so

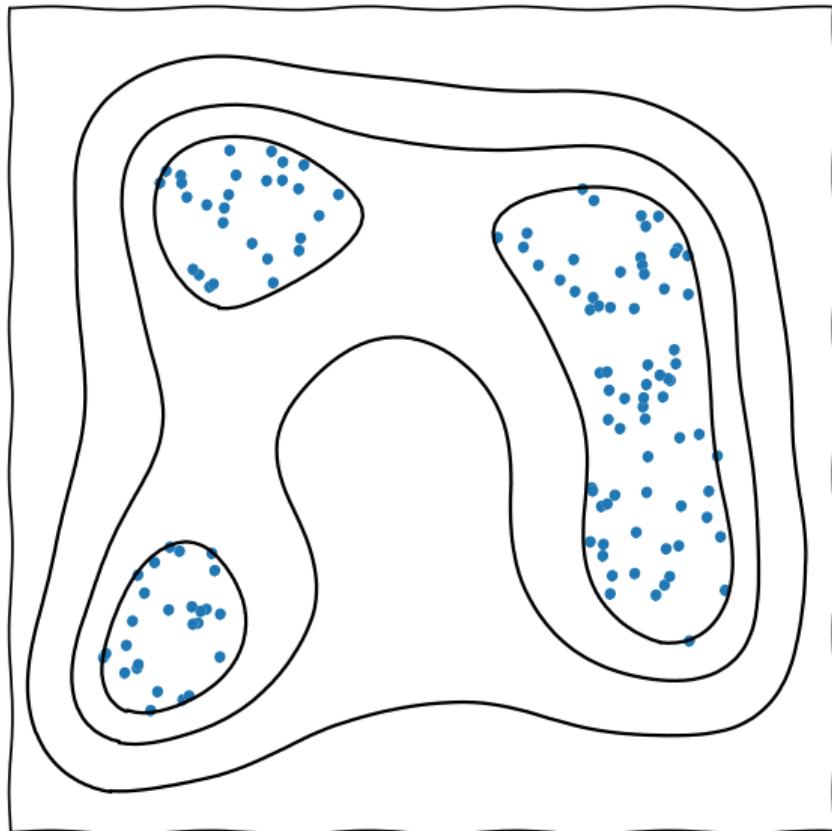
$$\int f(x)dV \approx \sum_i f(x_i)\Delta V_i, \quad V_i = V_0 e^{-i/n}$$



## The nested sampling meta-algorithm: live points

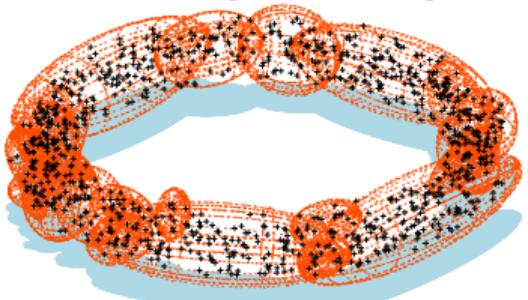
- ▶ Start with  $n$  random samples over the space.
- ▶ Delete outermost sample, and replace with a new random one at higher integrand value.
- ▶ The “live points” steadily contract around the peak(s) of the function.
- ▶ We can use this evolution to estimate volume *probabilistically*.
- ▶ At each iteration, the contours contract by  $\sim \frac{1}{n} \pm \frac{1}{n}$  of their volume.
- ▶ This is an exponential contraction, so

$$\int f(x)dV \approx \sum_i f(x_i)\Delta V_i, \quad V_i = V_0 e^{-(i \pm \sqrt{i})/n}$$

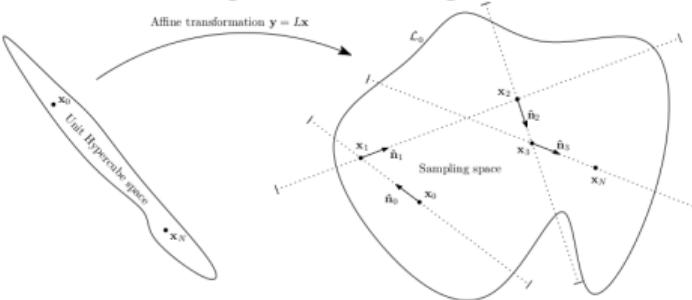


# Implementations of Nested Sampling [2205.15570](NatReview)

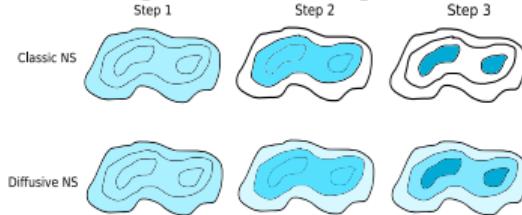
MultiNest [0809.3437]



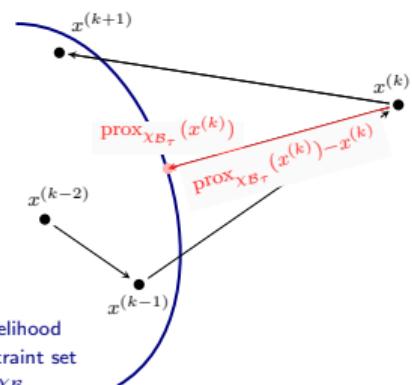
PolyChord [1506.00171]



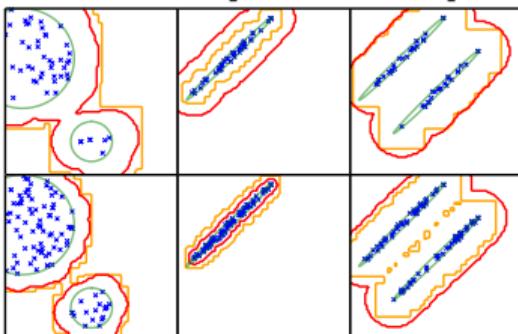
DNest [1606.03757]



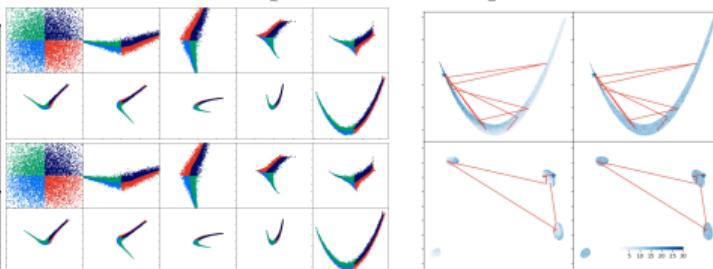
ProxNest [2106.03646]



UltraNest [2101.09604]



NeuralNest [1903.10860]



nessai [2102.11056] cpnest

nora [2305.19267]

jaxns [2012.15286]

nautilus [2306.16923]

<wh260@cam.ac.uk>

willhandley.co.uk/talks

dynesty [1904.02180]

# Types of nested sampler

- ▶ Broadly, most nested samplers can be split into how they create new live points.
- ▶ i.e. how they sample from the hard likelihood constraint  $\{\theta \sim \pi : \mathcal{L}(\theta) > \mathcal{L}_*\}$ .

## Rejection samplers

- ▶ e.g. MultiNest, UltraNest.
- ▶ Constructs bounding region and draws many invalid points until  $\mathcal{L}(\theta) > \mathcal{L}_*$ .
- ▶ Efficient in low dimensions, exponentially inefficient  $\sim \mathcal{O}(e^{d/d_0})$  in high  $d > d_0 \sim 10$ .

- ▶ Nested samplers usually come with:

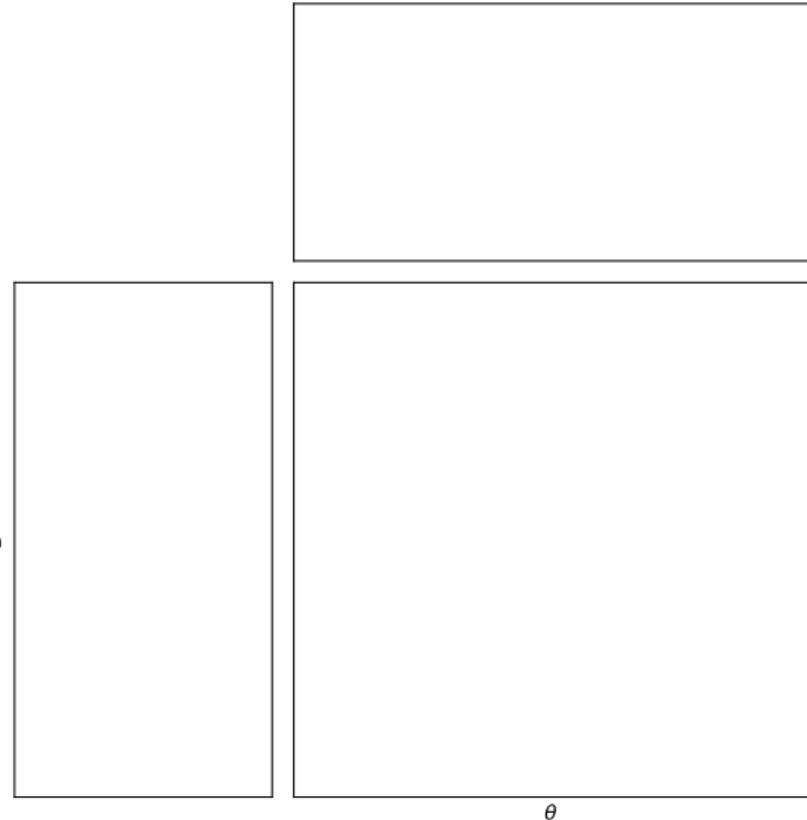
- ▶ *resolution* parameter  $n_{\text{live}}$  (which improve results as  $\sim \mathcal{O}(n_{\text{live}}^{-1/2})$ ).
- ▶ set of *reliability* parameters [2101.04525], which don't improve results if set arbitrarily high, but introduce systematic errors if set too low.
- ▶ e.g. Multinest efficiency  $\text{eff}$  or PolyChord chain length  $n_{\text{repeats}}$ .

## Chain-based samplers

- ▶ e.g. PolyChord, ProxNest.
- ▶ Run Markov chain starting at a live point, generating many valid (correlated) points.
- ▶ Linear  $\sim \mathcal{O}(d)$  penalty in decorrelating new live point from the original seed point.

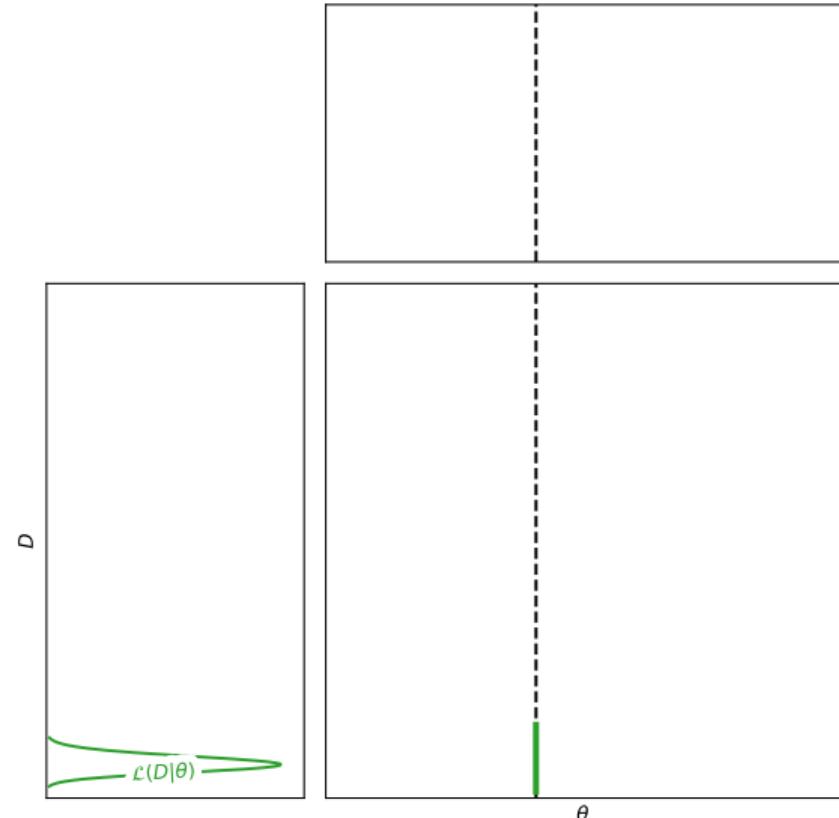
# SBI: Simulation-based inference

- ▶ What do you do if you don't know  $\mathcal{L}(D|\theta)$ ?
- ▶ If you have a simulator/forward model  
 $\theta \rightarrow D$  defines an *implicit likelihood*  $\mathcal{L}$ .
- ▶ Simulator generates samples from  $\mathcal{L}(\cdot|\theta)$ .
- ▶ With a prior  $\pi(\theta)$  can generate samples from  
joint distribution  $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$   
the “probability of everything”.
- ▶ Task of SBI is take joint  $\mathcal{J}$  samples and  
learn posterior  $\mathcal{P}(\theta|D)$  and evidence  $\mathcal{Z}(D)$   
and possibly likelihood  $\mathcal{L}(D|\theta)$ .
- ▶ Present state of the art achieves this using  
*machine learning* (neural networks).
  - ▶ My group's research tries to removes machine learning [github.com/handley-lab/lsbi](https://github.com/handley-lab/lsbi).



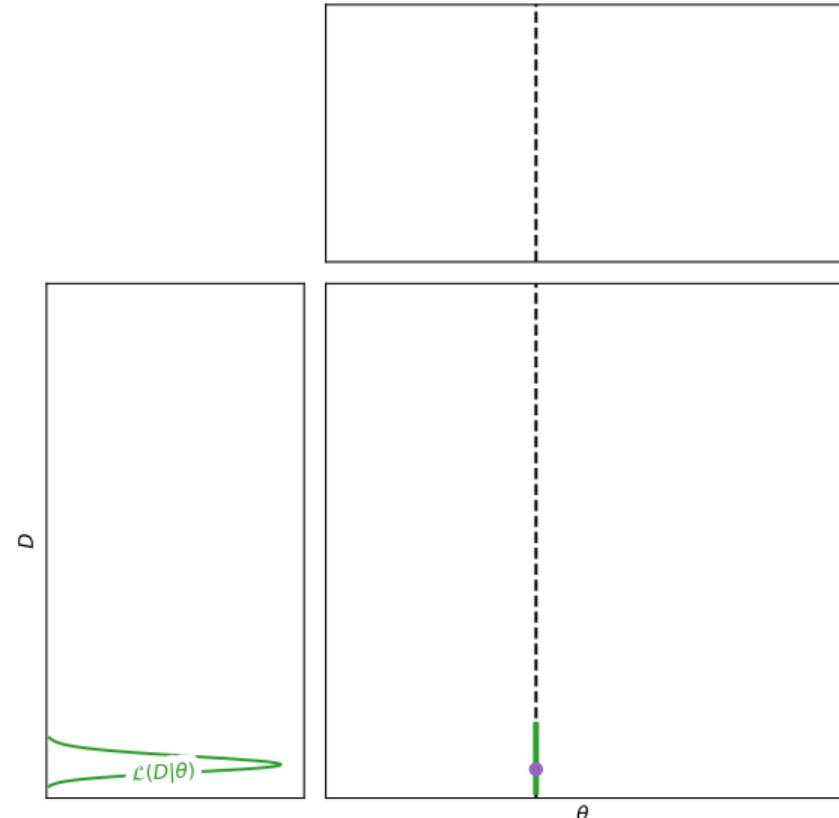
# SBI: Simulation-based inference

- ▶ What do you do if you don't know  $\mathcal{L}(D|\theta)$ ?
- ▶ If you have a simulator/forward model  
 $\theta \rightarrow D$  defines an *implicit likelihood*  $\mathcal{L}$ .
- ▶ Simulator generates samples from  $\mathcal{L}(\cdot|\theta)$ .
- ▶ With a prior  $\pi(\theta)$  can generate samples from joint distribution  $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$   
the “*probability of everything*”.
- ▶ Task of SBI is take joint  $\mathcal{J}$  samples and learn posterior  $\mathcal{P}(\theta|D)$  and evidence  $\mathcal{Z}(D)$   
and possibly likelihood  $\mathcal{L}(D|\theta)$ .
- ▶ Present state of the art achieves this using *machine learning* (neural networks).
  - ▶ My group's research tries to removes machine learning [github.com/handley-lab/lsbi](https://github.com/handley-lab/lsbi).



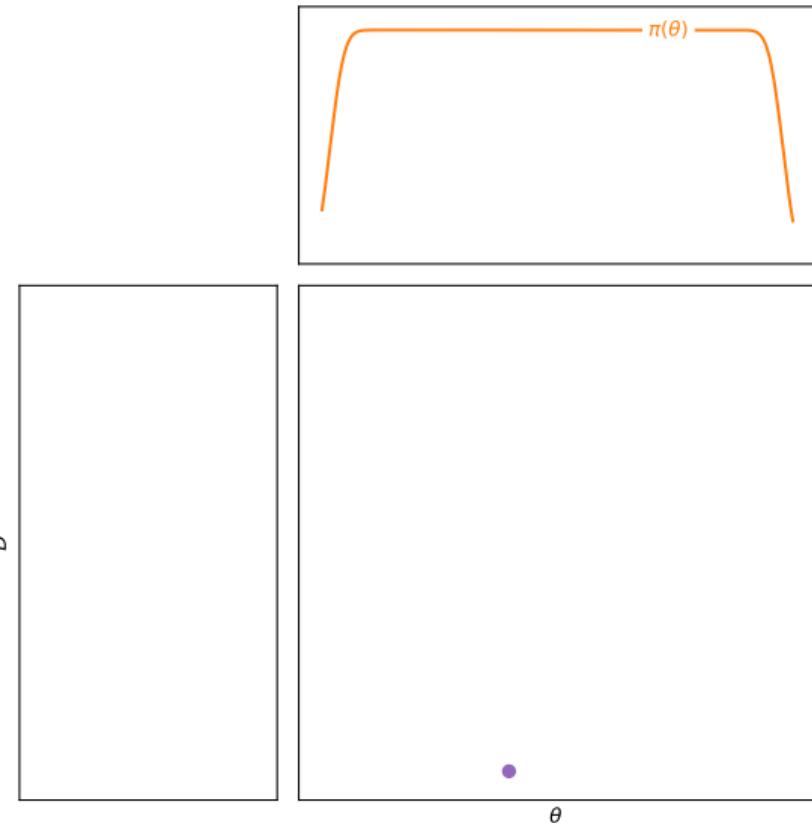
# SBI: Simulation-based inference

- ▶ What do you do if you don't know  $\mathcal{L}(D|\theta)$ ?
- ▶ If you have a simulator/forward model  
 $\theta \rightarrow D$  defines an *implicit likelihood*  $\mathcal{L}$ .
- ▶ Simulator generates samples from  $\mathcal{L}(\cdot|\theta)$ .
- ▶ With a prior  $\pi(\theta)$  can generate samples from joint distribution  $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$   
the “*probability of everything*”.
- ▶ Task of SBI is take joint  $\mathcal{J}$  samples and learn posterior  $\mathcal{P}(\theta|D)$  and evidence  $\mathcal{Z}(D)$   
and possibly likelihood  $\mathcal{L}(D|\theta)$ .
- ▶ Present state of the art achieves this using *machine learning* (neural networks).
  - ▶ My group's research tries to removes machine learning [github.com/handley-lab/lsbi](https://github.com/handley-lab/lsbi).



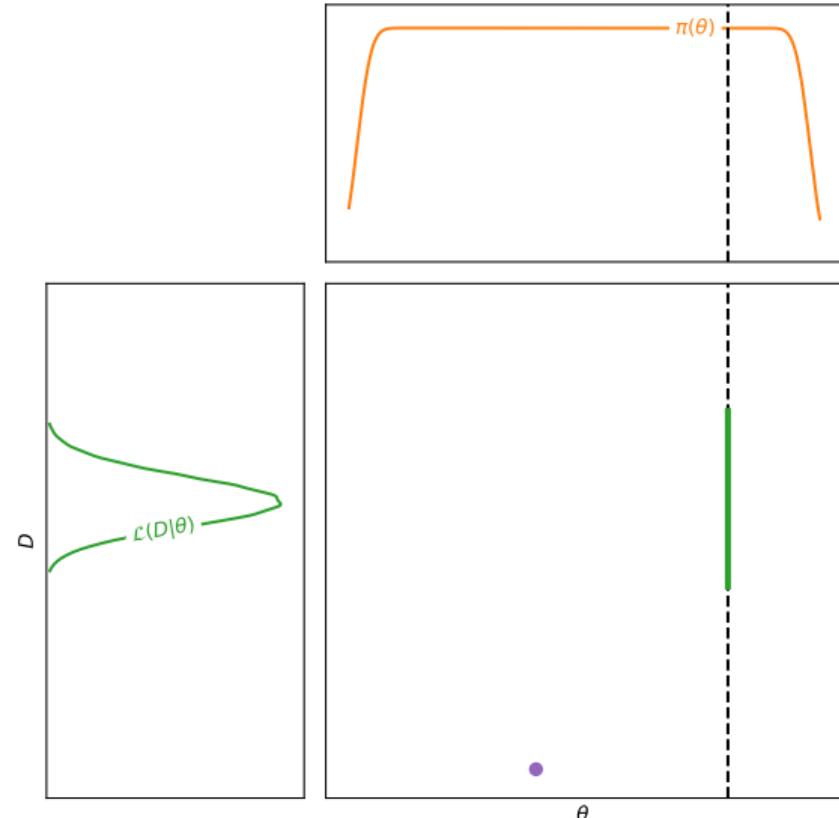
# SBI: Simulation-based inference

- ▶ What do you do if you don't know  $\mathcal{L}(D|\theta)$ ?
- ▶ If you have a simulator/forward model  
 $\theta \rightarrow D$  defines an *implicit likelihood*  $\mathcal{L}$ .
- ▶ Simulator generates samples from  $\mathcal{L}(\cdot|\theta)$ .
- ▶ With a prior  $\pi(\theta)$  can generate samples from joint distribution  $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$   
the “probability of everything”.
- ▶ Task of SBI is take joint  $\mathcal{J}$  samples and learn posterior  $\mathcal{P}(\theta|D)$  and evidence  $\mathcal{Z}(D)$   
and possibly likelihood  $\mathcal{L}(D|\theta)$ .
- ▶ Present state of the art achieves this using *machine learning* (neural networks).
  - ▶ My group's research tries to removes machine learning [github.com/handley-lab/lsbi](https://github.com/handley-lab/lsbi).



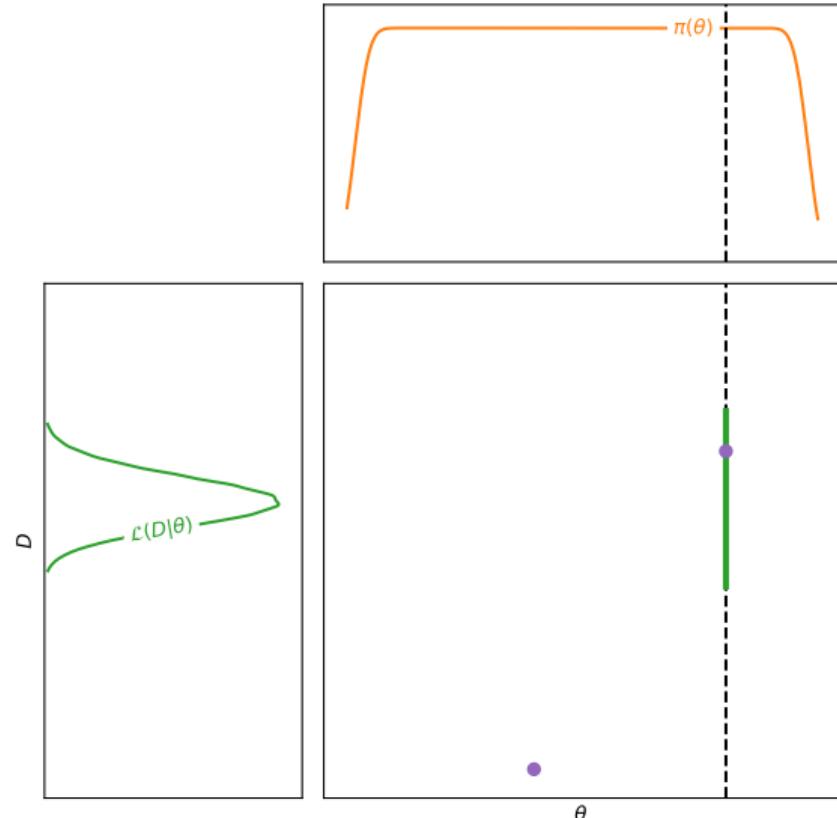
# SBI: Simulation-based inference

- ▶ What do you do if you don't know  $\mathcal{L}(D|\theta)$ ?
- ▶ If you have a simulator/forward model  
 $\theta \rightarrow D$  defines an *implicit likelihood*  $\mathcal{L}$ .
- ▶ Simulator generates samples from  $\mathcal{L}(\cdot|\theta)$ .
- ▶ With a prior  $\pi(\theta)$  can generate samples from joint distribution  $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$   
the “probability of everything”.
- ▶ Task of SBI is take joint  $\mathcal{J}$  samples and learn posterior  $\mathcal{P}(\theta|D)$  and evidence  $\mathcal{Z}(D)$  and possibly likelihood  $\mathcal{L}(D|\theta)$ .
- ▶ Present state of the art achieves this using *machine learning* (neural networks).
  - ▶ My group's research tries to remove machine learning [github.com/handley-lab/lstbi](https://github.com/handley-lab/lstbi).



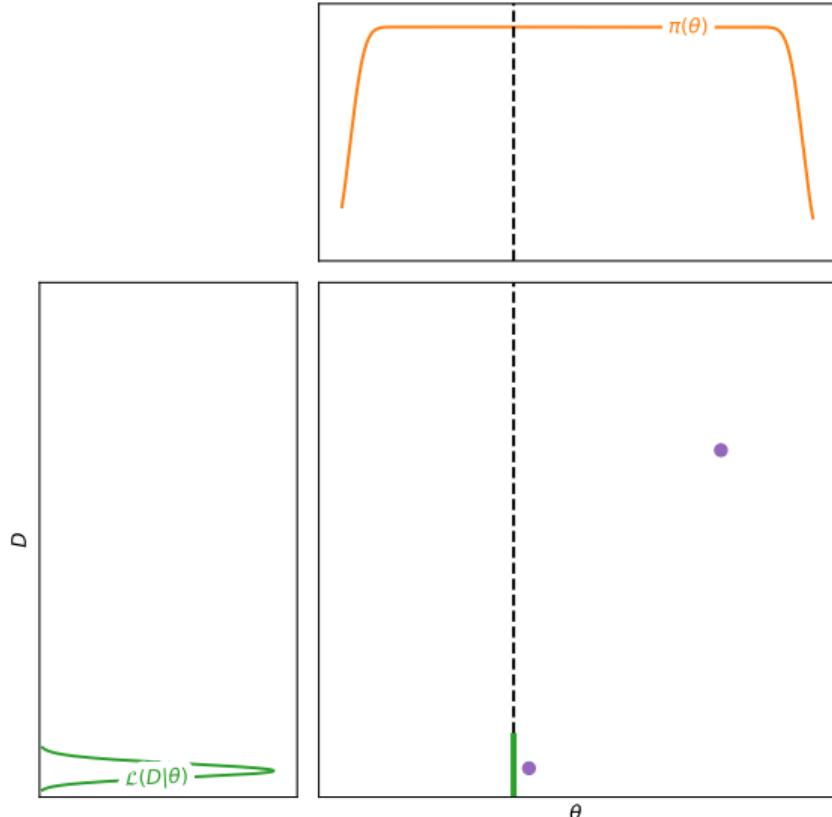
# SBI: Simulation-based inference

- ▶ What do you do if you don't know  $\mathcal{L}(D|\theta)$ ?
- ▶ If you have a simulator/forward model  
 $\theta \rightarrow D$  defines an *implicit likelihood*  $\mathcal{L}$ .
- ▶ Simulator generates samples from  $\mathcal{L}(\cdot|\theta)$ .
- ▶ With a prior  $\pi(\theta)$  can generate samples from joint distribution  $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$   
the “probability of everything”.
- ▶ Task of SBI is take joint  $\mathcal{J}$  samples and learn posterior  $\mathcal{P}(\theta|D)$  and evidence  $\mathcal{Z}(D)$  and possibly likelihood  $\mathcal{L}(D|\theta)$ .
- ▶ Present state of the art achieves this using *machine learning* (neural networks).
  - ▶ My group's research tries to remove machine learning [github.com/handley-lab/lstbi](https://github.com/handley-lab/lstbi).



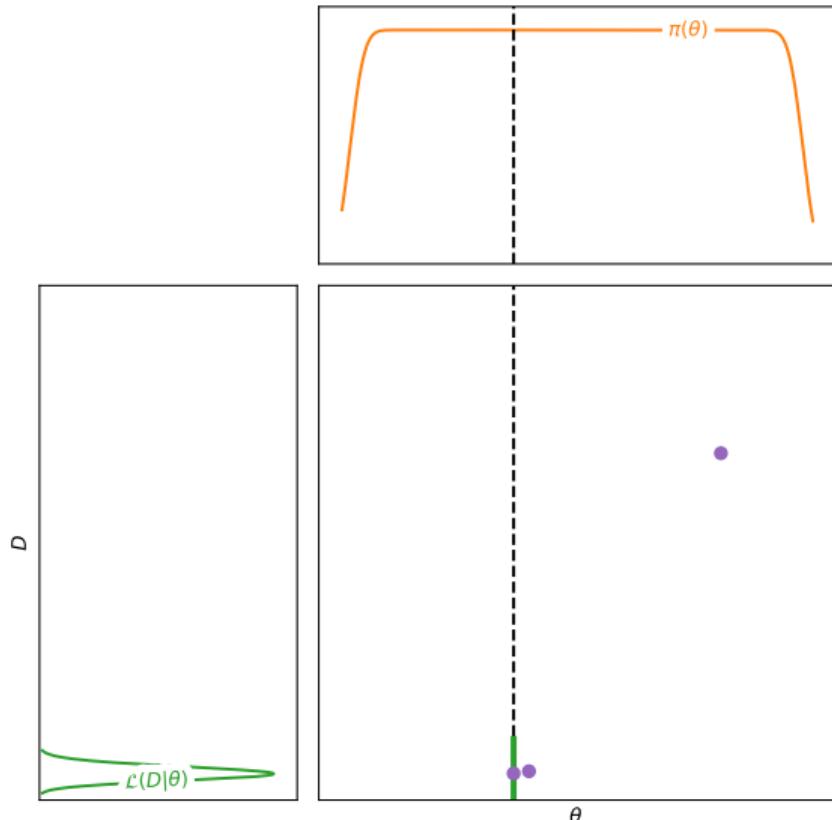
# SBI: Simulation-based inference

- ▶ What do you do if you don't know  $\mathcal{L}(D|\theta)$ ?
- ▶ If you have a simulator/forward model  
 $\theta \rightarrow D$  defines an *implicit likelihood*  $\mathcal{L}$ .
- ▶ Simulator generates samples from  $\mathcal{L}(\cdot|\theta)$ .
- ▶ With a prior  $\pi(\theta)$  can generate samples from joint distribution  $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$   
the “*probability of everything*”.
- ▶ Task of SBI is take joint  $\mathcal{J}$  samples and learn posterior  $\mathcal{P}(\theta|D)$  and evidence  $\mathcal{Z}(D)$  and possibly likelihood  $\mathcal{L}(D|\theta)$ .
- ▶ Present state of the art achieves this using *machine learning* (neural networks).
  - ▶ My group's research tries to removes machine learning [github.com/handley-lab/lsbi](https://github.com/handley-lab/lsbi).



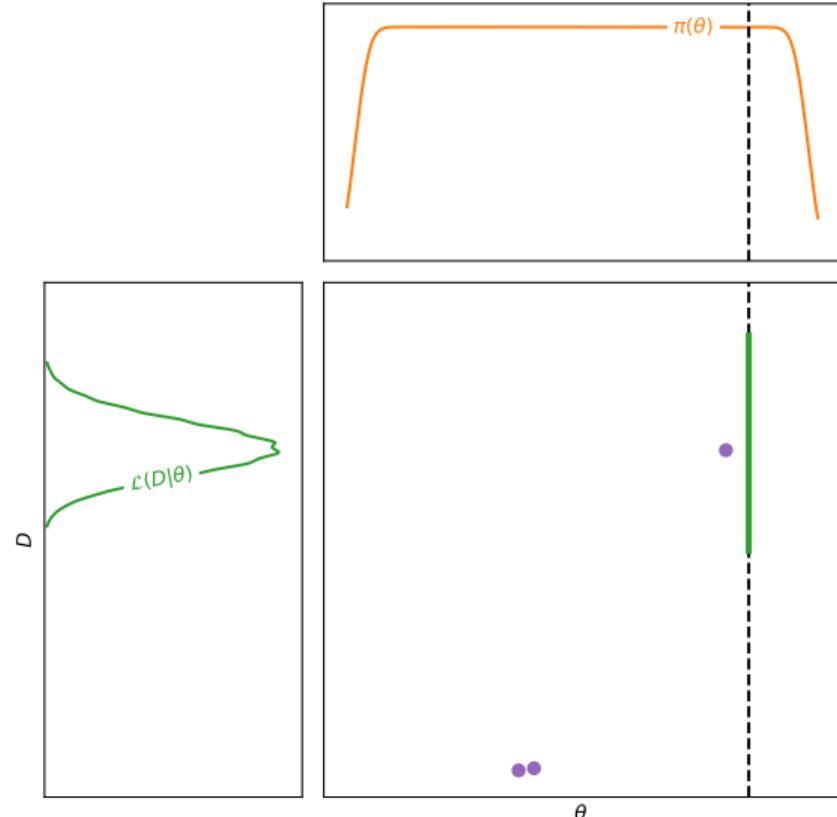
# SBI: Simulation-based inference

- ▶ What do you do if you don't know  $\mathcal{L}(D|\theta)$ ?
- ▶ If you have a simulator/forward model  
 $\theta \rightarrow D$  defines an *implicit likelihood*  $\mathcal{L}$ .
- ▶ Simulator generates samples from  $\mathcal{L}(\cdot|\theta)$ .
- ▶ With a prior  $\pi(\theta)$  can generate samples from joint distribution  $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$   
the “*probability of everything*”.
- ▶ Task of SBI is take joint  $\mathcal{J}$  samples and learn posterior  $\mathcal{P}(\theta|D)$  and evidence  $\mathcal{Z}(D)$  and possibly likelihood  $\mathcal{L}(D|\theta)$ .
- ▶ Present state of the art achieves this using *machine learning* (neural networks).
  - ▶ My group's research tries to removes machine learning [github.com/handley-lab/lsbi](https://github.com/handley-lab/lsbi).



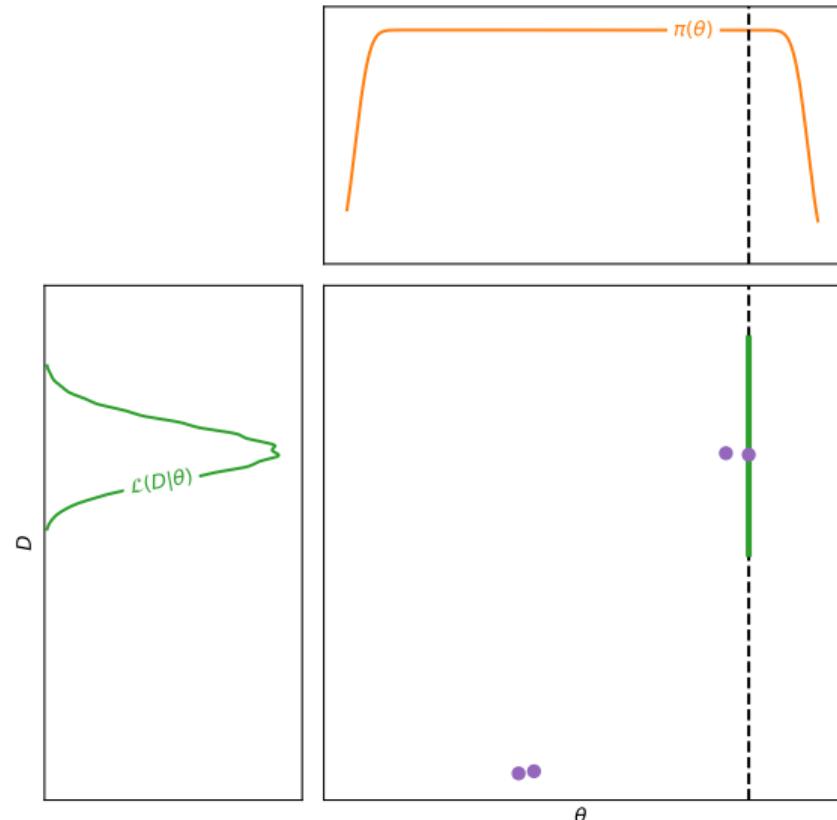
# SBI: Simulation-based inference

- ▶ What do you do if you don't know  $\mathcal{L}(D|\theta)$ ?
- ▶ If you have a simulator/forward model  
 $\theta \rightarrow D$  defines an *implicit likelihood*  $\mathcal{L}$ .
- ▶ Simulator generates samples from  $\mathcal{L}(\cdot|\theta)$ .
- ▶ With a prior  $\pi(\theta)$  can generate samples from joint distribution  $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$   
the “probability of everything”.
- ▶ Task of SBI is take joint  $\mathcal{J}$  samples and learn posterior  $\mathcal{P}(\theta|D)$  and evidence  $\mathcal{Z}(D)$  and possibly likelihood  $\mathcal{L}(D|\theta)$ .
- ▶ Present state of the art achieves this using *machine learning* (neural networks).
  - ▶ My group's research tries to removes machine learning [github.com/handley-lab/lstbi](https://github.com/handley-lab/lstbi).



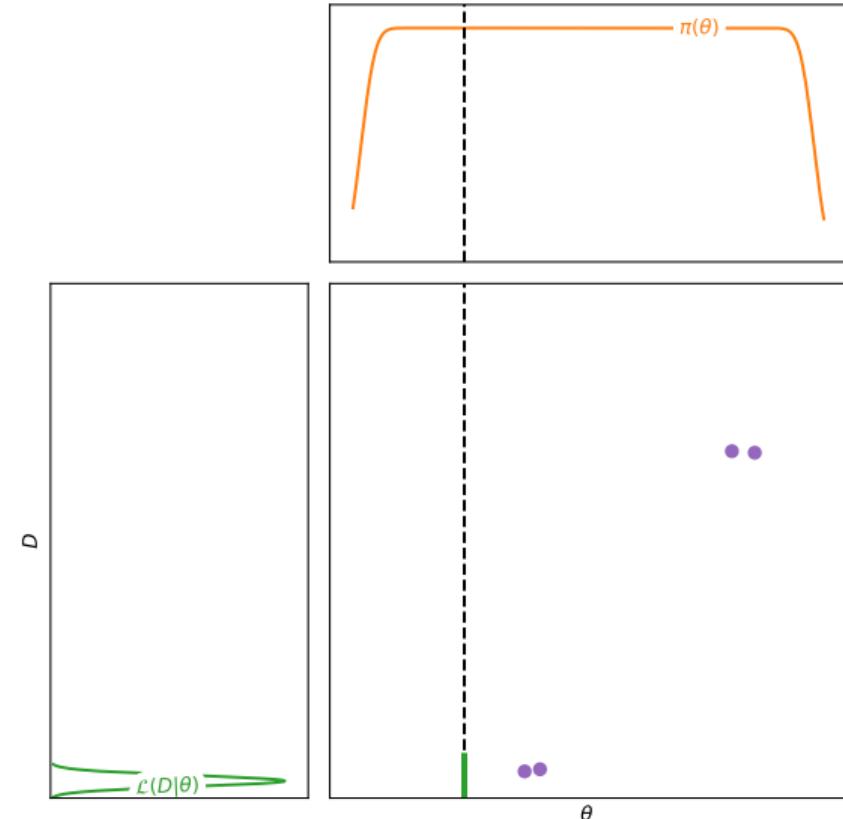
# SBI: Simulation-based inference

- ▶ What do you do if you don't know  $\mathcal{L}(D|\theta)$ ?
- ▶ If you have a simulator/forward model  
 $\theta \rightarrow D$  defines an *implicit likelihood*  $\mathcal{L}$ .
- ▶ Simulator generates samples from  $\mathcal{L}(\cdot|\theta)$ .
- ▶ With a prior  $\pi(\theta)$  can generate samples from joint distribution  $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$   
the “probability of everything”.
- ▶ Task of SBI is take joint  $\mathcal{J}$  samples and learn posterior  $\mathcal{P}(\theta|D)$  and evidence  $\mathcal{Z}(D)$  and possibly likelihood  $\mathcal{L}(D|\theta)$ .
- ▶ Present state of the art achieves this using *machine learning* (neural networks).
  - ▶ My group's research tries to remove machine learning [github.com/handley-lab/lsbi](https://github.com/handley-lab/lsbi).



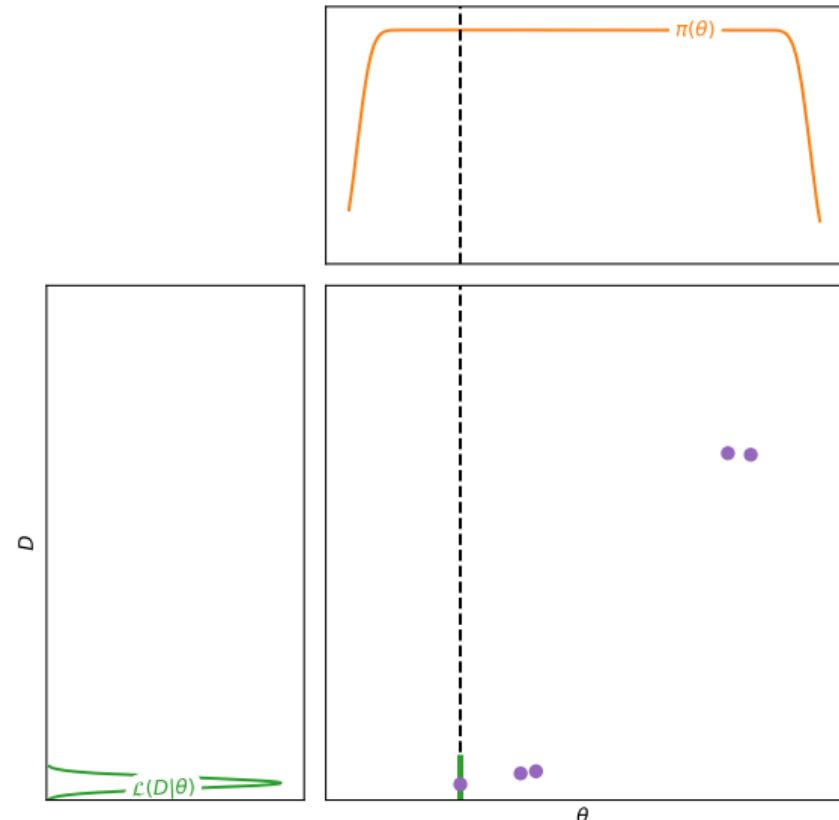
# SBI: Simulation-based inference

- ▶ What do you do if you don't know  $\mathcal{L}(D|\theta)$ ?
- ▶ If you have a simulator/forward model  
 $\theta \rightarrow D$  defines an *implicit likelihood*  $\mathcal{L}$ .
- ▶ Simulator generates samples from  $\mathcal{L}(\cdot|\theta)$ .
- ▶ With a prior  $\pi(\theta)$  can generate samples from joint distribution  $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$   
the “*probability of everything*”.
- ▶ Task of SBI is take joint  $\mathcal{J}$  samples and learn posterior  $\mathcal{P}(\theta|D)$  and evidence  $\mathcal{Z}(D)$  and possibly likelihood  $\mathcal{L}(D|\theta)$ .
- ▶ Present state of the art achieves this using *machine learning* (neural networks).
  - ▶ My group's research tries to removes machine learning [github.com/handley-lab/lstbi](https://github.com/handley-lab/lstbi).



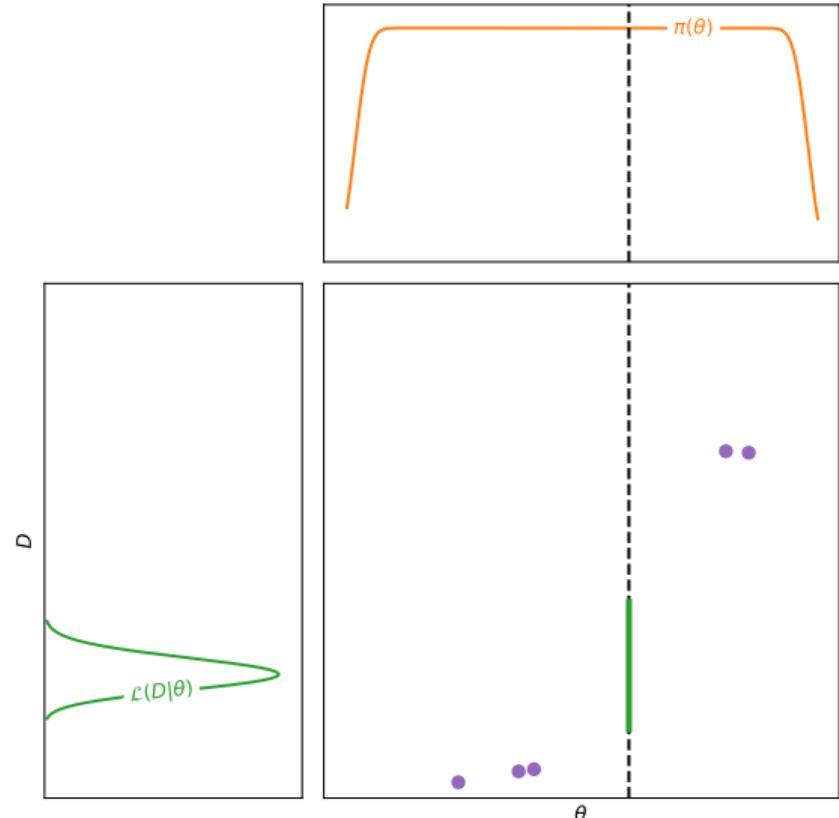
# SBI: Simulation-based inference

- ▶ What do you do if you don't know  $\mathcal{L}(D|\theta)$ ?
- ▶ If you have a simulator/forward model  
 $\theta \rightarrow D$  defines an *implicit likelihood*  $\mathcal{L}$ .
- ▶ Simulator generates samples from  $\mathcal{L}(\cdot|\theta)$ .
- ▶ With a prior  $\pi(\theta)$  can generate samples from joint distribution  $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$   
the “*probability of everything*”.
- ▶ Task of SBI is take joint  $\mathcal{J}$  samples and learn posterior  $\mathcal{P}(\theta|D)$  and evidence  $\mathcal{Z}(D)$  and possibly likelihood  $\mathcal{L}(D|\theta)$ .
- ▶ Present state of the art achieves this using *machine learning* (neural networks).
  - ▶ My group's research tries to removes machine learning [github.com/handley-lab/lstbi](https://github.com/handley-lab/lstbi).



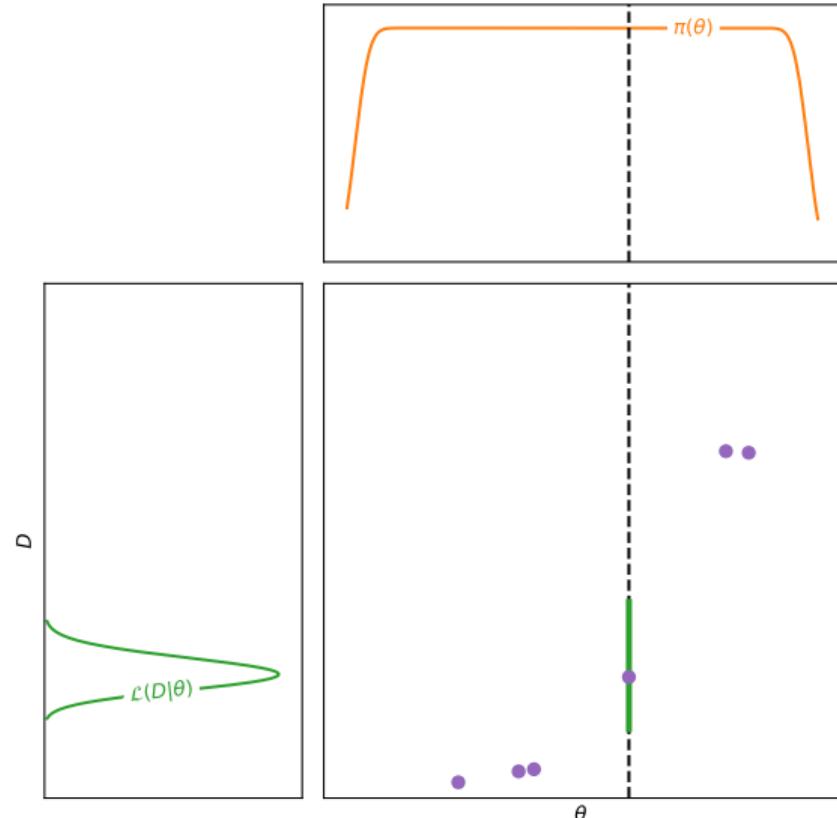
# SBI: Simulation-based inference

- ▶ What do you do if you don't know  $\mathcal{L}(D|\theta)$ ?
- ▶ If you have a simulator/forward model  
 $\theta \rightarrow D$  defines an *implicit likelihood*  $\mathcal{L}$ .
- ▶ Simulator generates samples from  $\mathcal{L}(\cdot|\theta)$ .
- ▶ With a prior  $\pi(\theta)$  can generate samples from joint distribution  $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$   
the “probability of everything”.
- ▶ Task of SBI is take joint  $\mathcal{J}$  samples and learn posterior  $\mathcal{P}(\theta|D)$  and evidence  $\mathcal{Z}(D)$  and possibly likelihood  $\mathcal{L}(D|\theta)$ .
- ▶ Present state of the art achieves this using *machine learning* (neural networks).
  - ▶ My group's research tries to removes machine learning [github.com/handley-lab/lsbi](https://github.com/handley-lab/lsbi).



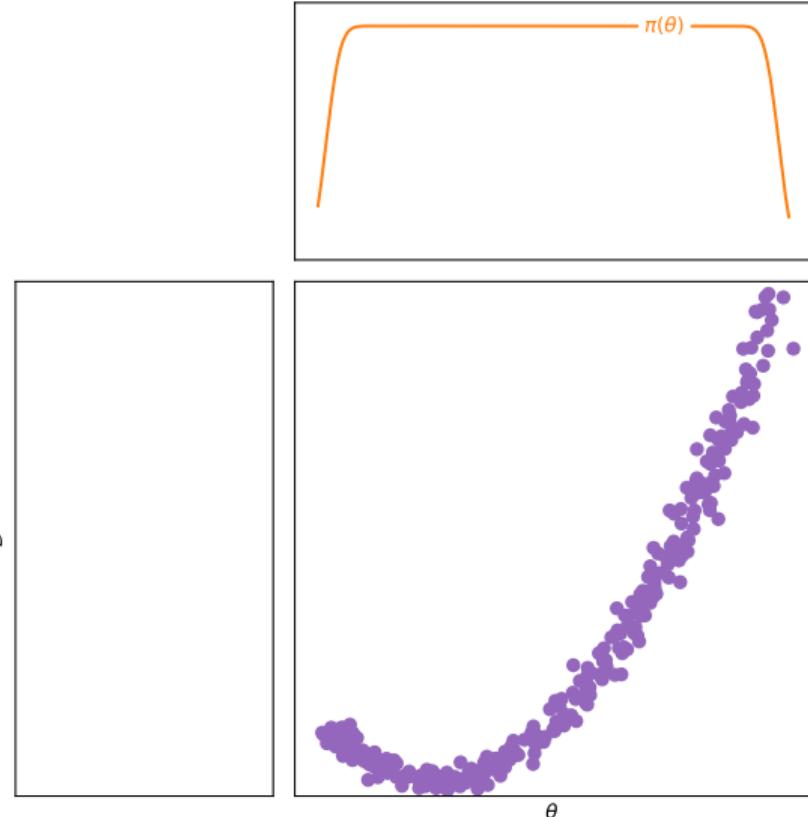
# SBI: Simulation-based inference

- ▶ What do you do if you don't know  $\mathcal{L}(D|\theta)$ ?
- ▶ If you have a simulator/forward model  
 $\theta \rightarrow D$  defines an *implicit likelihood*  $\mathcal{L}$ .
- ▶ Simulator generates samples from  $\mathcal{L}(\cdot|\theta)$ .
- ▶ With a prior  $\pi(\theta)$  can generate samples from joint distribution  $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$   
the “*probability of everything*”.
- ▶ Task of SBI is take joint  $\mathcal{J}$  samples and learn posterior  $\mathcal{P}(\theta|D)$  and evidence  $\mathcal{Z}(D)$  and possibly likelihood  $\mathcal{L}(D|\theta)$ .
- ▶ Present state of the art achieves this using *machine learning* (neural networks).
  - ▶ My group's research tries to removes machine learning [github.com/handley-lab/lsbi](https://github.com/handley-lab/lsbi).



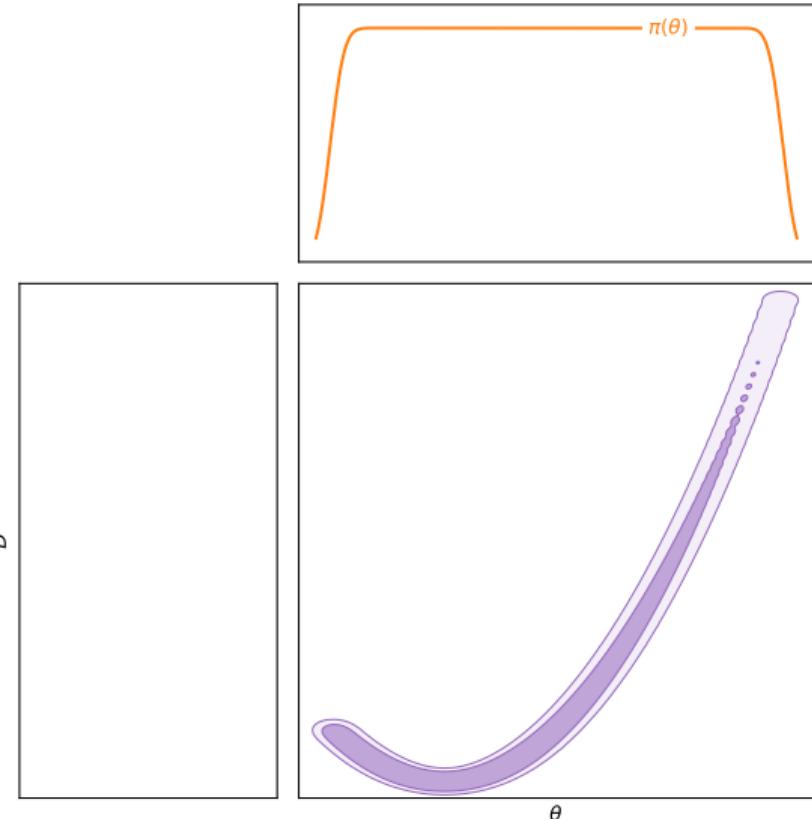
# SBI: Simulation-based inference

- ▶ What do you do if you don't know  $\mathcal{L}(D|\theta)$ ?
- ▶ If you have a simulator/forward model  
 $\theta \rightarrow D$  defines an *implicit likelihood*  $\mathcal{L}$ .
- ▶ Simulator generates samples from  $\mathcal{L}(\cdot|\theta)$ .
- ▶ With a prior  $\pi(\theta)$  can generate samples from joint distribution  $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$   
the “*probability of everything*”.
- ▶ Task of SBI is take joint  $\mathcal{J}$  samples and learn posterior  $\mathcal{P}(\theta|D)$  and evidence  $\mathcal{Z}(D)$   
and possibly likelihood  $\mathcal{L}(D|\theta)$ .
- ▶ Present state of the art achieves this using *machine learning* (neural networks).
  - ▶ My group's research tries to removes machine learning [github.com/handley-lab/lsbi](https://github.com/handley-lab/lsbi).



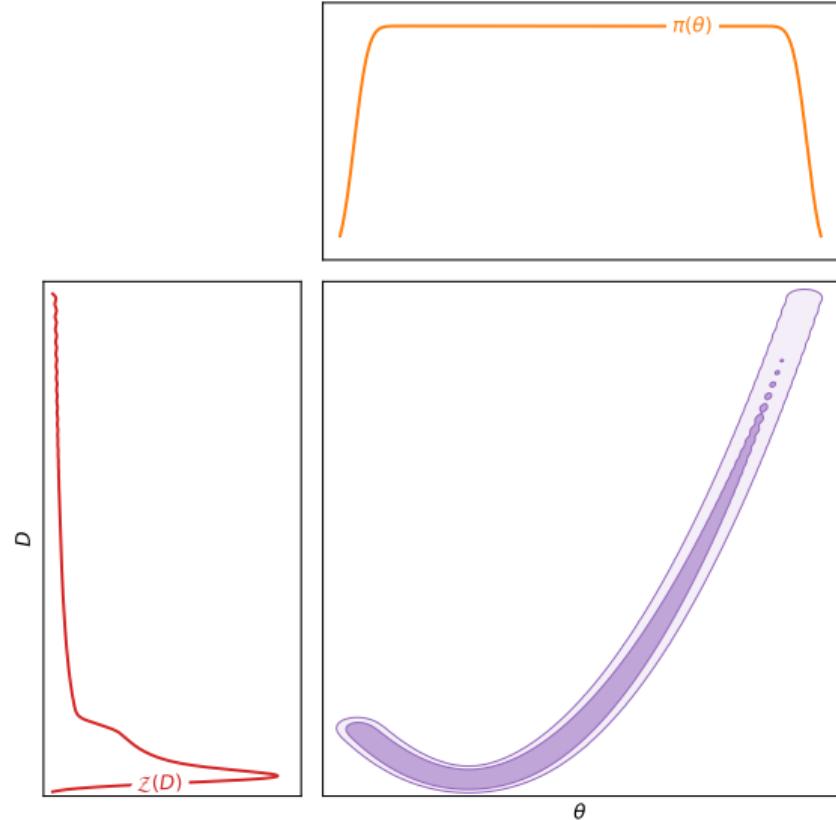
# SBI: Simulation-based inference

- ▶ What do you do if you don't know  $\mathcal{L}(D|\theta)$ ?
- ▶ If you have a simulator/forward model  
 $\theta \rightarrow D$  defines an *implicit likelihood*  $\mathcal{L}$ .
- ▶ Simulator generates samples from  $\mathcal{L}(\cdot|\theta)$ .
- ▶ With a prior  $\pi(\theta)$  can generate samples from joint distribution  $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$   
the “probability of everything”.
- ▶ Task of SBI is take joint  $\mathcal{J}$  samples and learn posterior  $\mathcal{P}(\theta|D)$  and evidence  $\mathcal{Z}(D)$   
and possibly likelihood  $\mathcal{L}(D|\theta)$ .
- ▶ Present state of the art achieves this using *machine learning* (neural networks).
  - ▶ My group's research tries to remove machine learning [github.com/handley-lab/lsbi](https://github.com/handley-lab/lsbi).



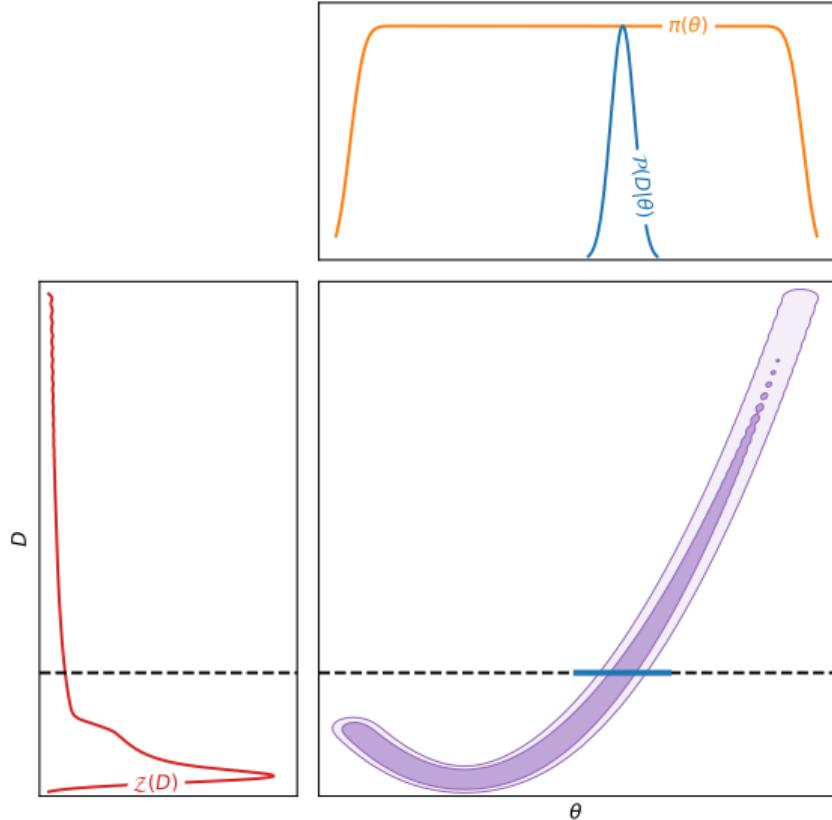
# SBI: Simulation-based inference

- ▶ What do you do if you don't know  $\mathcal{L}(D|\theta)$ ?
- ▶ If you have a simulator/forward model  
 $\theta \rightarrow D$  defines an *implicit likelihood*  $\mathcal{L}$ .
- ▶ Simulator generates samples from  $\mathcal{L}(\cdot|\theta)$ .
- ▶ With a prior  $\pi(\theta)$  can generate samples from joint distribution  $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$   
the “probability of everything”.
- ▶ Task of SBI is take joint  $\mathcal{J}$  samples and learn posterior  $\mathcal{P}(\theta|D)$  and evidence  $\mathcal{Z}(D)$  and possibly likelihood  $\mathcal{L}(D|\theta)$ .
- ▶ Present state of the art achieves this using *machine learning* (neural networks).
  - ▶ My group's research tries to removes machine learning [github.com/handley-lab/lsbi](https://github.com/handley-lab/lsbi).



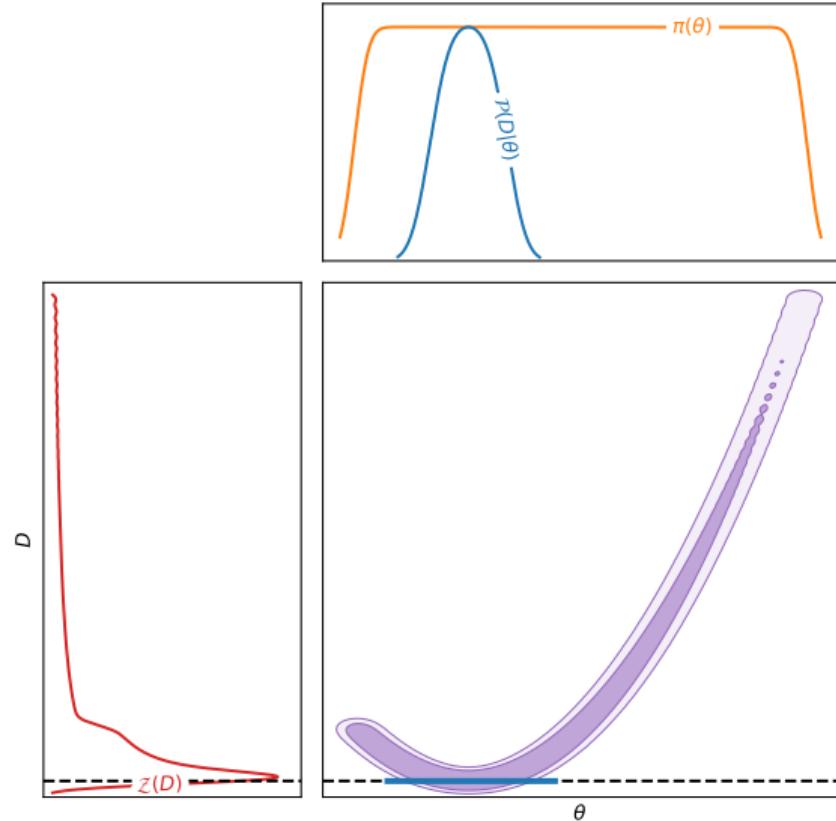
# SBI: Simulation-based inference

- ▶ What do you do if you don't know  $\mathcal{L}(D|\theta)$ ?
- ▶ If you have a simulator/forward model  
 $\theta \rightarrow D$  defines an *implicit likelihood*  $\mathcal{L}$ .
- ▶ Simulator generates samples from  $\mathcal{L}(\cdot|\theta)$ .
- ▶ With a prior  $\pi(\theta)$  can generate samples from joint distribution  $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$   
the “probability of everything”.
- ▶ Task of SBI is take joint  $\mathcal{J}$  samples and learn posterior  $\mathcal{P}(\theta|D)$  and evidence  $\mathcal{Z}(D)$  and possibly likelihood  $\mathcal{L}(D|\theta)$ .
- ▶ Present state of the art achieves this using *machine learning* (neural networks).
  - ▶ My group's research tries to remove machine learning [github.com/handley-lab/lstbi](https://github.com/handley-lab/lstbi).



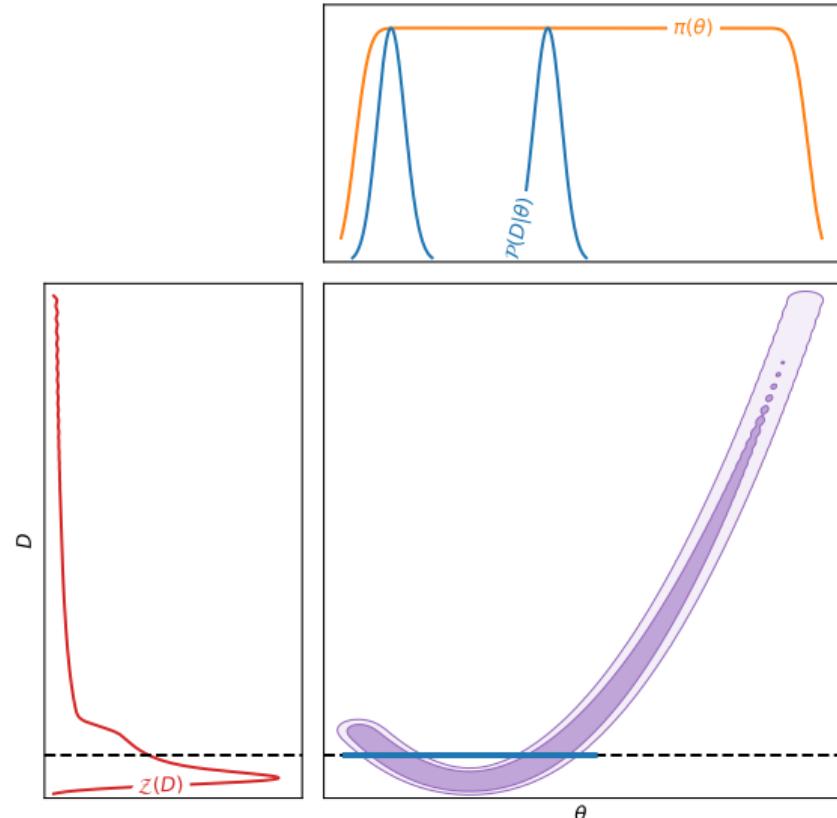
# SBI: Simulation-based inference

- ▶ What do you do if you don't know  $\mathcal{L}(D|\theta)$ ?
- ▶ If you have a simulator/forward model  
 $\theta \rightarrow D$  defines an *implicit likelihood*  $\mathcal{L}$ .
- ▶ Simulator generates samples from  $\mathcal{L}(\cdot|\theta)$ .
- ▶ With a prior  $\pi(\theta)$  can generate samples from joint distribution  $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$   
the “probability of everything”.
- ▶ Task of SBI is take joint  $\mathcal{J}$  samples and learn posterior  $\mathcal{P}(\theta|D)$  and evidence  $\mathcal{Z}(D)$  and possibly likelihood  $\mathcal{L}(D|\theta)$ .
- ▶ Present state of the art achieves this using *machine learning* (neural networks).
  - ▶ My group's research tries to remove machine learning [github.com/handley-lab/lsbi](https://github.com/handley-lab/lsbi).



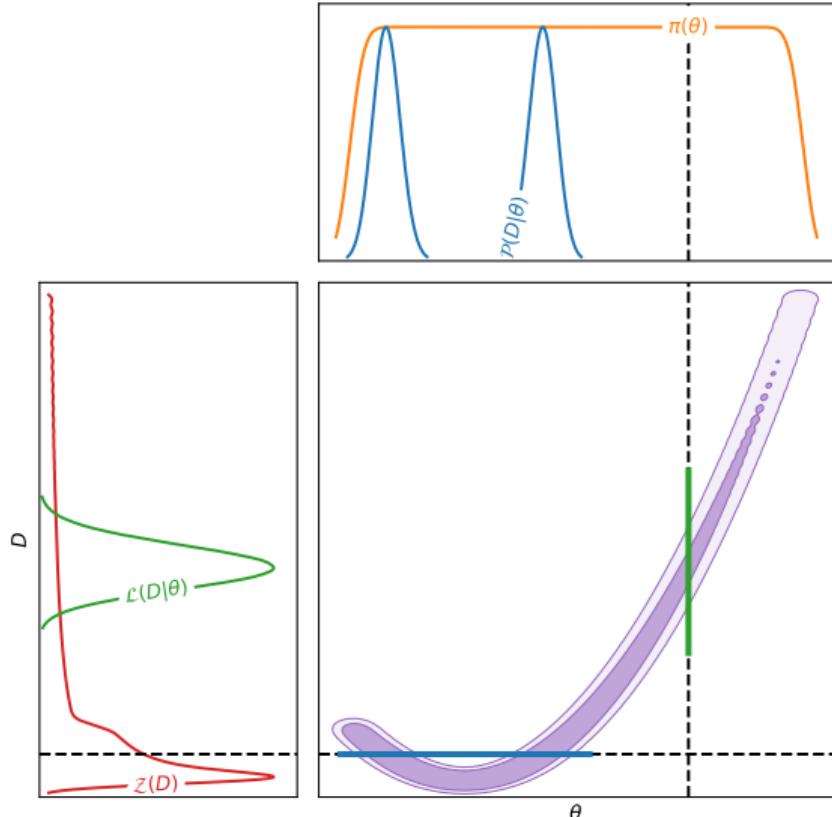
# SBI: Simulation-based inference

- ▶ What do you do if you don't know  $\mathcal{L}(D|\theta)$ ?
- ▶ If you have a simulator/forward model  
 $\theta \rightarrow D$  defines an *implicit likelihood*  $\mathcal{L}$ .
- ▶ Simulator generates samples from  $\mathcal{L}(\cdot|\theta)$ .
- ▶ With a prior  $\pi(\theta)$  can generate samples from joint distribution  $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$   
the “probability of everything”.
- ▶ Task of SBI is take joint  $\mathcal{J}$  samples and learn posterior  $\mathcal{P}(\theta|D)$  and evidence  $\mathcal{Z}(D)$  and possibly likelihood  $\mathcal{L}(D|\theta)$ .
- ▶ Present state of the art achieves this using *machine learning* (neural networks).
  - ▶ My group's research tries to removes machine learning [github.com/handley-lab/lsbi](https://github.com/handley-lab/lsbi).



# SBI: Simulation-based inference

- ▶ What do you do if you don't know  $\mathcal{L}(D|\theta)$ ?
- ▶ If you have a simulator/forward model  
 $\theta \rightarrow D$  defines an *implicit likelihood*  $\mathcal{L}$ .
- ▶ Simulator generates samples from  $\mathcal{L}(\cdot|\theta)$ .
- ▶ With a prior  $\pi(\theta)$  can generate samples from joint distribution  $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$   
the “probability of everything”.
- ▶ Task of SBI is take joint  $\mathcal{J}$  samples and learn posterior  $\mathcal{P}(\theta|D)$  and evidence  $\mathcal{Z}(D)$  and possibly likelihood  $\mathcal{L}(D|\theta)$ .
- ▶ Present state of the art achieves this using *machine learning* (neural networks).
  - ▶ My group's research tries to remove machine learning [github.com/handley-lab/lsbi](https://github.com/handley-lab/lsbi).



# Why SBI?

SBI is useful because:

1. If you don't have a likelihood, you can still do inference
  - ▶ This is the usual case beyond CMB cosmology
2. Faster than LBI
  - ▶ emulation – also applies to LBI in principle
3. No need to pragmatically encode fiducial cosmologies
  - ▶ Covariance computation implicitly encoded in simulations
  - ▶ Highly relevant for disentangling tensions & systematics
4. Equips AI/ML with Bayesian interpretability
5. Lower barrier to entry than LBI
  - ▶ Much easier to forward model a systematic
  - ▶ Emerging set of plug-and-play packages
  - ▶ For this reason alone, it will come to dominate scientific inference

A screenshot of the GitHub repository page for 'sbi' (simulation-based inference). The page shows the README file, which contains Python code for a simulation-based inference example. The code includes imports, a function definition, and a call to the function with specific parameters. Below the code, there are sections for 'Likelihood estimation' and 'Likelihood ratio estimation'.

[github.com/sbi-dev](https://github.com/sbi-dev)

A screenshot of the GitHub repository page for 'Swyft'. The page features a large logo at the top. Below the logo, there is a navigation menu with links like 'Home', 'About', 'Installation', 'Documentation', 'Examples', and 'Project'. The main content area displays the project's purpose: "Swyft is a system for scientific simulation-based inference at scale".

[github.com/undark-lab/swyft](https://github.com/undark-lab/swyft)

A screenshot of the GitHub repository page for 'pyselfi'. The page has a dark header with the project name 'pyselfi'. Below the header, there is a navigation menu with links for 'HOME', 'SCIENTIFIC', 'ACADEMIC', 'BLOG', and 'ABOUT ME'. The main content area features a large logo and some descriptive text about the package.

[github.com/florent-leclercq/pyselfi](https://github.com/florent-leclercq/pyselfi)

A screenshot of the GitHub repository page for 'pydelfi'. The page has a dark header with the project name 'pydelfi'. Below the header, there is a navigation menu with links for 'HOME', 'ABOUT', 'INSTALLATION', 'PYDELFI', and 'PAPER'. The main content area features a large logo and some descriptive text about the package.

[github.com/justinalsing/pydelfi](https://github.com/justinalsing/pydelfi)

# SBI in astrophysics

- ▶ 2024 has been the year it has started to be applied to real data.
- ▶ Mostly for weak lensing
- ▶ However: SBI requires mock data generation code
- ▶ Most data analysis codes were built before the generative paradigm.
- ▶ It's still a lot of work to upgrade cosmological likelihoods to be able to do this (e.g. plik & camspec).

## Investigating the turbulent hot gas in X-COP galaxy clusters

S. Dupourqué<sup>1</sup>, N. Clerc<sup>1</sup>, E. Pointecouteau<sup>1</sup>, D. Eckert<sup>2</sup>, S. Ettori<sup>3</sup>, and F. Vazza<sup>4,5,6</sup>

Dark Energy Survey Year 3 results: simulation-based cosmological inference with wavelet harmonics, scattering transforms, and moments of weak lensing mass maps II. Cosmological results

M. Gatti,<sup>1,\*</sup> G. Campailla,<sup>2</sup> N. Jeffrey,<sup>3</sup> L. Whitney,<sup>3</sup> A. Paredon,<sup>4</sup> J. Prat,<sup>5</sup> J. Williamson,<sup>3</sup> M. Ravera,<sup>2</sup> B.

## Neural Posterior Estimation with guaranteed exact coverage: the ringdown of GW150914

Marco Crisostomi<sup>1,2</sup>, Kallol Dey<sup>3</sup>, Enrico Barausse<sup>1,2</sup>, Roberto Trotta<sup>1,2,4,5</sup>

## Applying Simulation-Based Inference to Spectral and Spatial Information from the Galactic Center Gamma-Ray Excess

Katharena Christy,<sup>a</sup> Eric J. Baxter,<sup>b</sup> Jason Kumar<sup>a</sup>

## KIDS-1000 and DES-Y1 combined: Cosmology from peak count statistics

Joachim Harnois-Déraps<sup>1\*</sup>, Sven Heydenreich<sup>2</sup>, Benjamin Giblin<sup>3</sup>, Nicolas Martinet<sup>4</sup>, Tilman Tröster<sup>5</sup>, Marika Asgari<sup>1,6,7</sup>, Pierre Burger<sup>8,9,10</sup>, Tiago Castro<sup>1,12,13,14</sup>, Klaus Dolag<sup>15</sup>, Catherine Heymans<sup>3,16</sup>, Hendrik Hildebrandt<sup>16</sup>, Benjamin Joachimi<sup>17</sup> & Angus H. Wright<sup>16</sup>

## KiDS-SBI: Simulation-Based Inference Analysis of KiDS-1000 Cosmic Shear

Maximilian von Wietersheim-Kramsta<sup>1,2,3</sup>, Kiyam Lin<sup>4</sup>, Nicolas Tessore<sup>1</sup>, Benjamin Joachimi<sup>1</sup>, Arthur Loureiro<sup>4,5</sup>, Robert Reischke<sup>6,7</sup>, and Angus H. Wright<sup>1</sup>

## Simulation-based inference of deep fields: galaxy population model and redshift distributions

Beatrice Moser,<sup>a,1</sup> Tomasz Kacprzak,<sup>a,b</sup> Silvan Fischbacher,<sup>a</sup> Alexandre Refregier,<sup>a</sup> Dominic Grimm,<sup>a</sup> Luca Tortorelli<sup>c</sup>

## SmBiG: Cosmological Constraints using Simulation-Based Inference of Galaxy Clustering with Marked Power Spectra

ELENA MASSARA  <sup>1,2,\*</sup>, CHANGHOON HAN  <sup>2</sup>, MICHAEL EICKENBERG <sup>2</sup>, SHERELY HO <sup>3</sup>, JIAMIN HOU <sup>2</sup>, PABLO LEMOS <sup>4,5</sup>, CHIRAG MODI <sup>4,6</sup>, AZADEH MORADNEZHAD DEHGHAN  <sup>7,8,11</sup>, LIAM PARKER <sup>9,12</sup> AND BRUNO RÉGALDO-SAINT BLANCARD 

# Neural Ratio Estimation

- SBI flavours: [github.com/sbi-dev/sbi](https://github.com/sbi-dev/sbi)

NPE Neural posterior estimation

NLE Neural likelihood estimation

NJE Neural joint estimation

NRE Neural ratio estimation

- NRE recap:

1. Generate joint samples  $(\theta, D) \sim \mathcal{J}$

- straightforward if you have a simulator:

$\theta \sim \pi(\cdot)$ ,  $D \sim \mathcal{L}(\cdot | \theta)$

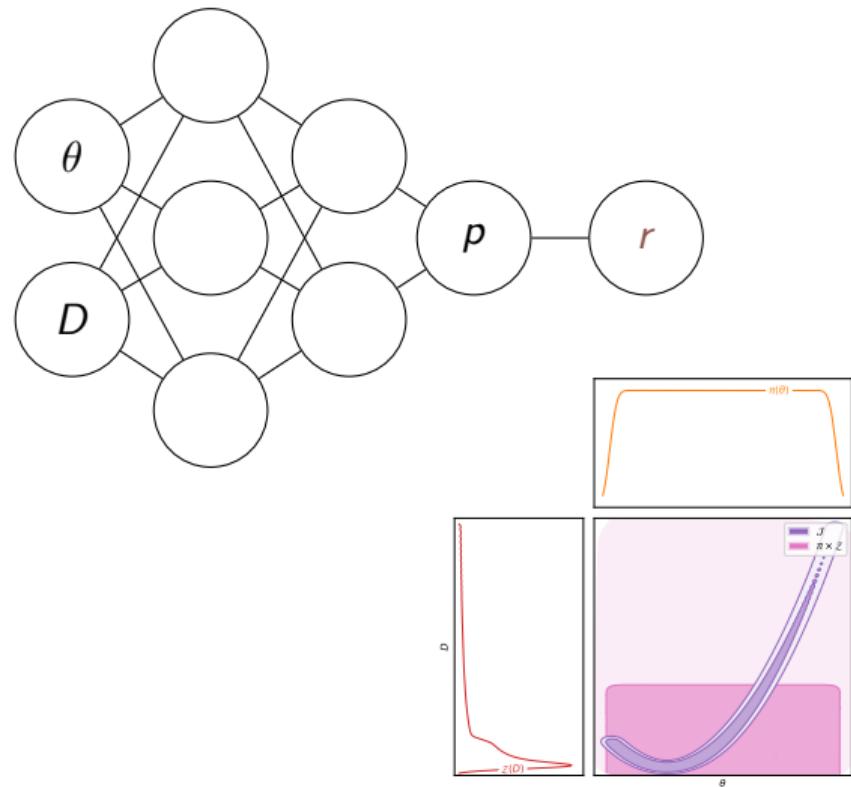
2. Generate separated samples  $\theta \sim \pi$ ,  $D \sim \mathcal{Z}$

- aside: can shortcut step 2 by scrambling the  $(\theta, D)$  pairings from step 1

3. Train probabilistic classifier  $p$  to distinguish whether  $(\theta, D)$  came from  $\mathcal{J}$  or  $\pi \times \mathcal{Z}$ .

$$4. \frac{p}{1-p} = r = \frac{P(\theta, D)}{P(\theta)P(D)} = \frac{\mathcal{J}}{\pi \times \mathcal{Z}} = \frac{\mathcal{L}}{\mathcal{Z}} = \frac{\mathcal{P}}{\pi}.$$

5. Use ratio  $r$  for parameter estimation  $\mathcal{P} = r \times \pi$



# Neural Ratio Estimation

- ▶ SBI flavours: [github.com/sbi-dev/sbi](https://github.com/sbi-dev/sbi)

NPE Neural posterior estimation

NLE Neural likelihood estimation

NJE Neural joint estimation

NRE Neural ratio estimation

- ▶ NRE recap:

1. Generate joint samples  $(\theta, D) \sim \mathcal{J}$

- ▶ *straightforward if you have a simulator:*  
 $\theta \sim \pi(\cdot)$ ,  $D \sim \mathcal{L}(\cdot | \theta)$

2. Generate separated samples  $\theta \sim \pi$ ,  $D \sim \mathcal{Z}$

- ▶ *aside: can shortcut step 2 by scrambling the  $(\theta, D)$  pairings from step 1*

3. Train probabilistic classifier  $p$  to distinguish whether  $(\theta, D)$  came from  $\mathcal{J}$  or  $\pi \times \mathcal{Z}$ .

4.  $\frac{p}{1-p} = r = \frac{P(\theta, D)}{P(\theta)P(D)} = \frac{\mathcal{J}}{\pi \times \mathcal{Z}} = \frac{\mathcal{L}}{\mathcal{Z}} = \frac{\mathcal{P}}{\pi}$ .

5. Use ratio  $r$  for parameter estimation  $\mathcal{P} = r \times \pi$

## Bayesian proof

- ▶ Let  $M_{\mathcal{J}}$ :  $(\theta, D) \sim \mathcal{J}$ ,  $M_{\pi \mathcal{Z}}$ :  $(\theta, D) \sim \pi \times \mathcal{Z}$

- ▶ Classifier gives

$$p(\theta, D) = P(M_{\mathcal{J}} | \theta, D) = 1 - P(M_{\pi \mathcal{Z}} | \theta, D)$$

- ▶ Bayes theorem then shows

$$\frac{p}{1-p} = \frac{P(M_{\mathcal{J}} | \theta, D)}{P(M_{\pi \mathcal{Z}} | \theta, D)} = \frac{P(\theta, D | M_{\mathcal{J}})P(M_{\mathcal{J}})}{P(\theta, D | M_{\pi \mathcal{Z}})P(M_{\pi \mathcal{Z}})} = \frac{\mathcal{J}}{\pi \mathcal{Z}},$$

where we have assumed

- ▶  $P(M_{\mathcal{J}}) = P(M_{\pi \mathcal{Z}})$ ,

and by definition

- ▶  $\mathcal{J}(\theta, D) = P(\theta, D | M_{\mathcal{J}})$

- ▶  $\pi(\theta)\mathcal{Z}(D) = P(\theta, D | M_{\pi \mathcal{Z}})$ .

# Neural Ratio Estimation

- SBI flavours: [github.com/sbi-dev/sbi](https://github.com/sbi-dev/sbi)

NPE Neural posterior estimation

NLE Neural likelihood estimation

NJE Neural joint estimation

NRE Neural ratio estimation

- NRE recap:

1. Generate joint samples  $(\theta, D) \sim \mathcal{J}$

- ▶ *straightforward if you have a simulator:*

$$\theta \sim \pi(\cdot), D \sim \mathcal{L}(\cdot|\theta)$$

2. Generate separated samples  $\theta \sim \pi, D \sim \mathcal{Z}$

- ▶ *aside: can shortcut step 2 by scrambling the  $(\theta, D)$  pairings from step 1*

3. Train probabilistic classifier  $p$  to distinguish whether  $(\theta, D)$  came from  $\mathcal{J}$  or  $\pi \times \mathcal{Z}$ .

4.  $\frac{p}{1-p} = r = \frac{P(\theta, D)}{P(\theta)P(D)} = \frac{\mathcal{J}}{\pi \times \mathcal{Z}} = \frac{\mathcal{L}}{\mathcal{Z}} = \frac{P}{\pi}$ .

5. Use ratio  $r$  for parameter estimation  $\mathcal{P} = r \times \pi$

## Why I like NRE

- ▶ The link between classification and inference is profound.
- ▶ Density estimation is hard – Dimensionless  $r$  divides out the hard-to-calculate parts.

## Why I don't like NRE

- ▶ Practical implementations require marginalisation [[2107.01214](#)], or autoregression [[2308.08597](#)].
- ▶ Model comparison and parameter estimation are separate [[2305.11241](#)].

# Marginal inference

Harry Bevins



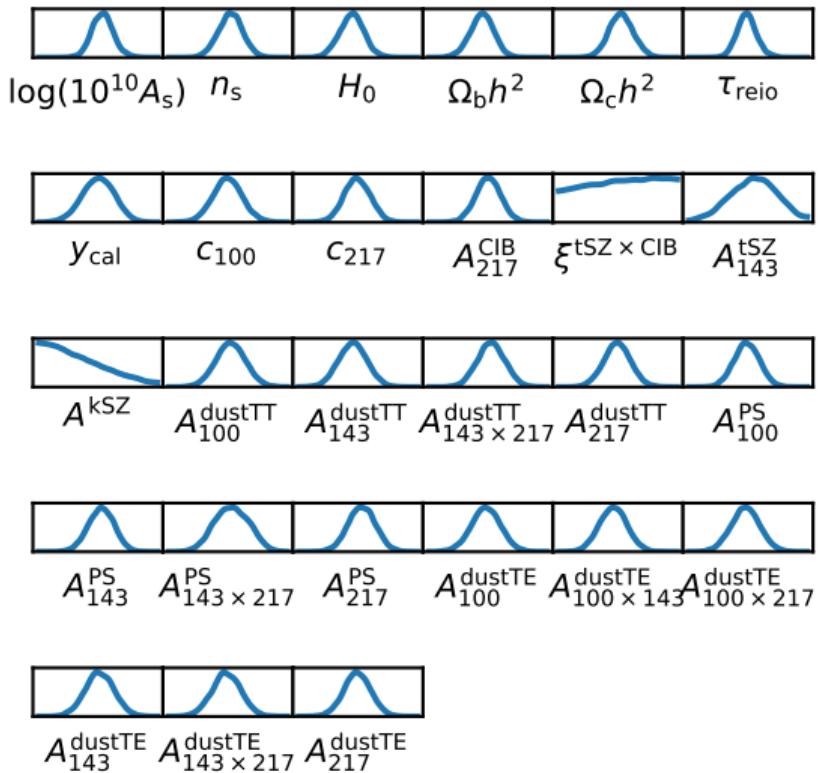
PhD → JRF

- ▶ Many cosmological likelihoods come with nuisance parameters that have limited relevance for onward inference.

- ▶ Notation: CMB cosmology

$\mathcal{L}$	Likelihood	(e.g. plik),
$D$	Data	(e.g. CMB),
$\theta$	Cosmological parameters	(e.g. $\Omega_m$ , $H_0 \dots$ ),
$\alpha$	Nuisance parameters	(e.g. $A_{\text{planck}} \dots$ ),
$M$	Model	(e.g. $\Lambda\text{CDM}$ ).

- ▶ Some marginal statistics (e.g. marginal means, posteriors...) are easy to compute.
- ▶ More machinery is needed for e.g. nuisance marginalised likelihoods and marginal KL divergences  $\mathcal{D}_{\text{KL}}$ .



# Marginal inference

Harry Bevins

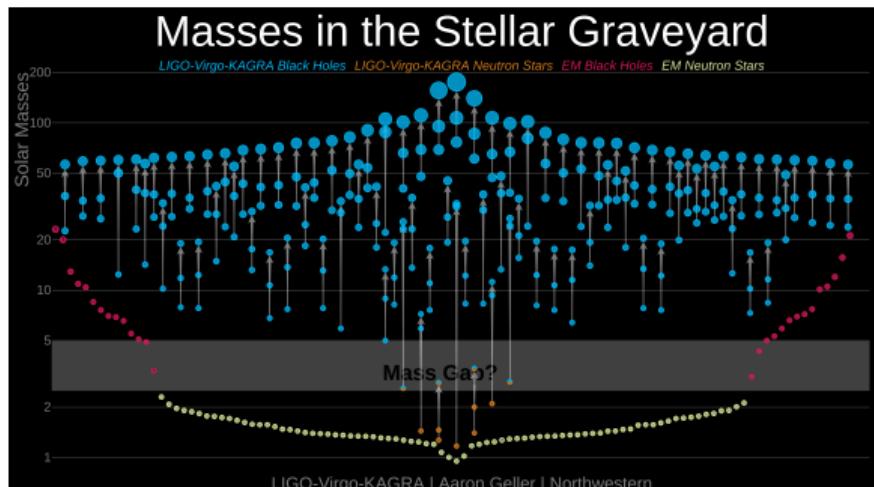
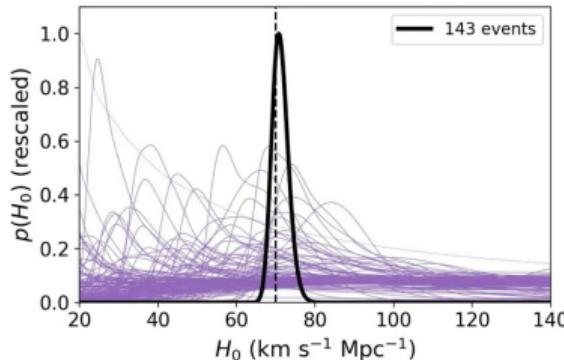


PhD→JRF

- ▶ Many cosmological likelihoods come with nuisance parameters that have limited relevance for onward inference.
- ▶ Notation: GW cosmology

$\mathcal{L}$	Likelihood	(e.g. LAL),
$D$	Data	(e.g. GW170817),
$\theta$	Cosmological parameters	(e.g. $H_0 \dots$ ),
$\alpha$	Nuisance parameters	(e.g. $m_1, m_2 \dots$ ),
$M$	Model	(e.g. $\Lambda$ CDM).

- ▶ Some marginal statistics (e.g. marginal means, posteriors...) are easy to compute.
- ▶ More machinery is needed for e.g. nuisance marginalised likelihoods and marginal KL divergences  $\mathcal{D}_{KL}$ .





- ▶ Bayes theorem

$$\mathcal{L}(\theta, \alpha) \times \pi(\theta, \alpha) = \mathcal{P}(\theta, \alpha) \times \mathcal{Z} \quad (1)$$

**Likelihood** × **Prior** = **Posterior** × **Evidence**

$\alpha$ : nuisance parameters,  $\theta$ : cosmo parameters.

- ▶ Marginal Bayes theorem

$$\mathcal{L}(\theta) \times \pi(\theta) = \mathcal{P}(\theta) \times \mathcal{Z} \quad (2)$$

- ▶ Non-trivially gives **nuisance-free likelihood**

$$\boxed{\mathcal{L}(\theta) = \frac{\mathcal{P}(\theta)\mathcal{Z}}{\pi(\theta)} = \frac{\int \mathcal{L}(\theta, \alpha)\pi(\theta, \alpha)d\alpha}{\int \pi(\theta, \alpha)d\alpha}} \quad (3)$$

## Key properties

- ▶ Given datasets  $A$  and  $B$ , each with own nuisance parameters  $\alpha_A$  and  $\alpha_B$ :
- ▶ If you use  $\mathcal{L}_A(\theta)$ , you get the same (marginal) posterior and evidence if you had run with nuisance parameters  $\alpha_A$  (ditto  $B$ ).
- ▶ If you run inference on  $\mathcal{L}_A(\theta) \times \mathcal{L}_B(\theta)$ , you get the same (marginal) posterior and evidence if you had run with all nuisance parameters  $\alpha_A, \alpha_B$  on.  
*(weak marginal consistency requirements on joint  $\pi(\theta, \alpha_A, \alpha_B)$  and marginal priors)*



$$\mathcal{L}(\theta) = \frac{\mathcal{P}(\theta)\mathcal{Z}}{\pi(\theta)}$$

- ▶ To compute the nuisance marginalised likelihood, need:

 $\mathcal{L}(\theta, \alpha)$  $\pi(\theta, \alpha)$ 

1. Bayesian evidence  $\mathcal{Z}$
2. Marginal prior and posterior **densities**

## 1. Bayesian evidence $\mathcal{Z}$ : g

- ▶ Nested sampling
- ▶ Parallel tempering (pocomc, ptmcmc)
- ▶ Sequential Monte Carlo (SMC)
- ▶ MCEvidence

## 2. Marginal prior $\pi(\theta)$ and posterior $\mathcal{P}(\theta)$ densities:

- ▶ Histograms of samples
- ▶ Kernel density estimation
- ▶ Normalising flows / Diffusion models
- ▶ ...
- ▶ Emulators usually much faster than original likelihoods
- ▶ `margarine`: PyPI, [github.com/htjb/margarine](https://github.com/htjb/margarine)

# Nuisance marginalised likelihoods: Practice [2205.12841]

Harry Bevins



PhD→JRF

$$\mathcal{L}(\theta) = \frac{\mathcal{P}(\theta)\mathcal{Z}}{\pi(\theta)}$$

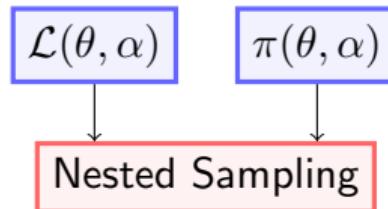
- ▶ To compute the nuisance marginalised likelihood, need:
  1. Bayesian evidence  $\mathcal{Z}$
  2. Marginal prior and posterior **densities**

## 1. Bayesian evidence $\mathcal{Z}$ : g

- ▶ Nested sampling
- ▶ Parallel tempering (pocomc, ptmcmc)
- ▶ Sequential Monte Carlo (SMC)
- ▶ MCEvidence

## 2. Marginal prior $\pi(\theta)$ and posterior $\mathcal{P}(\theta)$ densities:

- ▶ Histograms of samples
- ▶ Kernel density estimation
- ▶ Normalising flows / Diffusion models
- ▶ ...
- ▶ Emulators usually much faster than original likelihoods
- ▶ `margarine`: PyPI, [github.com/htjb/margarine](https://github.com/htjb/margarine)



# Nuisance marginalised likelihoods: Practice [2205.12841]

Harry Bevins



PhD→JRF

$$\mathcal{L}(\theta) = \frac{\mathcal{P}(\theta)\mathcal{Z}}{\pi(\theta)}$$

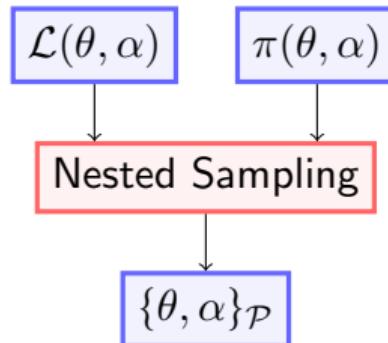
- ▶ To compute the nuisance marginalised likelihood, need:
  1. Bayesian evidence  $\mathcal{Z}$
  2. Marginal prior and posterior **densities**

## 1. Bayesian evidence $\mathcal{Z}$ : g

- ▶ Nested sampling
- ▶ Parallel tempering (pocomc, ptmcmc)
- ▶ Sequential Monte Carlo (SMC)
- ▶ MCEvidence

## 2. Marginal prior $\pi(\theta)$ and posterior $\mathcal{P}(\theta)$ densities:

- ▶ Histograms of samples
- ▶ Kernel density estimation
- ▶ Normalising flows / Diffusion models
- ▶ ...
- ▶ Emulators usually much faster than original likelihoods
- ▶ `margarine`: PyPI, [github.com/htjb/margarine](https://github.com/htjb/margarine)



# Nuisance marginalised likelihoods: Practice [2205.12841]

Harry Bevins



PhD→JRF

$$\mathcal{L}(\theta) = \frac{\mathcal{P}(\theta)\mathcal{Z}}{\pi(\theta)}$$

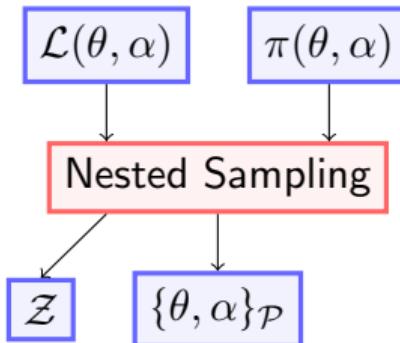
- ▶ To compute the nuisance marginalised likelihood, need:
  1. Bayesian evidence  $\mathcal{Z}$
  2. Marginal prior and posterior **densities**

## 1. Bayesian evidence $\mathcal{Z}$ : g

- ▶ Nested sampling
- ▶ Parallel tempering (pocomc, ptmcmc)
- ▶ Sequential Monte Carlo (SMC)
- ▶ MCEvidence

## 2. Marginal prior $\pi(\theta)$ and posterior $\mathcal{P}(\theta)$ densities:

- ▶ Histograms of samples
- ▶ Kernel density estimation
- ▶ Normalising flows / Diffusion models
- ▶ ...
- ▶ Emulators usually much faster than original likelihoods
- ▶ `margarine`: PyPI, [github.com/htjb/margarine](https://github.com/htjb/margarine)



# Nuisance marginalised likelihoods: Practice [2205.12841]

Harry Bevins



PhD→JRF

$$\mathcal{L}(\theta) = \frac{\mathcal{P}(\theta)\mathcal{Z}}{\pi(\theta)}$$

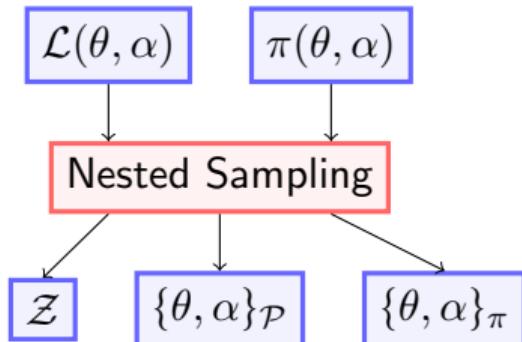
- ▶ To compute the nuisance marginalised likelihood, need:
  1. Bayesian evidence  $\mathcal{Z}$
  2. Marginal prior and posterior **densities**

## 1. Bayesian evidence $\mathcal{Z}$ : g

- ▶ Nested sampling
- ▶ Parallel tempering (pocomc, ptmcmc)
- ▶ Sequential Monte Carlo (SMC)
- ▶ MCEvidence

## 2. Marginal prior $\pi(\theta)$ and posterior $\mathcal{P}(\theta)$ densities:

- ▶ Histograms of samples
- ▶ Kernel density estimation
- ▶ Normalising flows / Diffusion models
- ▶ ...
- ▶ Emulators usually much faster than original likelihoods
- ▶ `margarine`: PyPI, [github.com/htjb/margarine](https://github.com/htjb/margarine)



# Nuisance marginalised likelihoods: Practice [2205.12841]

Harry Bevins



PhD→JRF

$$\mathcal{L}(\theta) = \frac{\mathcal{P}(\theta)\mathcal{Z}}{\pi(\theta)}$$

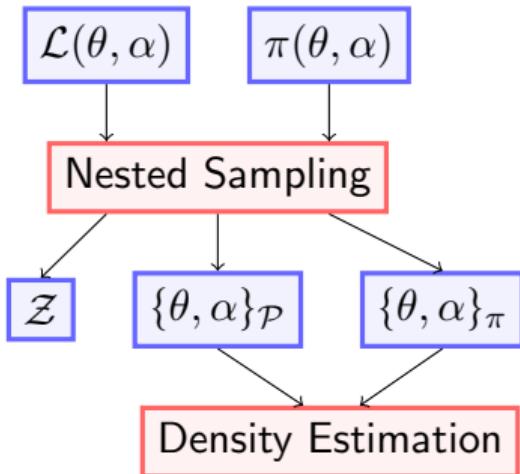
- ▶ To compute the nuisance marginalised likelihood, need:
  1. Bayesian evidence  $\mathcal{Z}$
  2. Marginal prior and posterior **densities**

## 1. Bayesian evidence $\mathcal{Z}$ : g

- ▶ Nested sampling
- ▶ Parallel tempering (pocomc, ptmcmc)
- ▶ Sequential Monte Carlo (SMC)
- ▶ MCEvidence

## 2. Marginal prior $\pi(\theta)$ and posterior $\mathcal{P}(\theta)$ densities:

- ▶ Histograms of samples
- ▶ Kernel density estimation
- ▶ Normalising flows / Diffusion models
- ▶ ...
- ▶ Emulators usually much faster than original likelihoods
- ▶ `margarine`: PyPI, [github.com/htjb/margarine](https://github.com/htjb/margarine)



# Nuisance marginalised likelihoods: Practice [2205.12841]

Harry Bevins



PhD→JRF

$$\mathcal{L}(\theta) = \frac{\mathcal{P}(\theta)\mathcal{Z}}{\pi(\theta)}$$

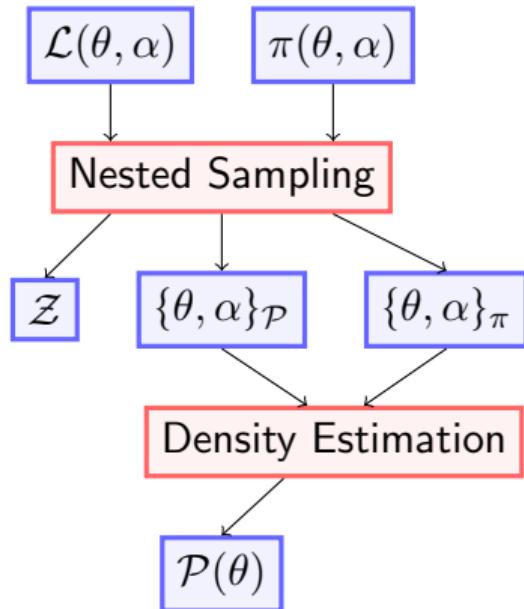
- ▶ To compute the nuisance marginalised likelihood, need:
  1. Bayesian evidence  $\mathcal{Z}$
  2. Marginal prior and posterior **densities**

## 1. Bayesian evidence $\mathcal{Z}$ : g

- ▶ Nested sampling
- ▶ Parallel tempering (pocomc, ptmcmc)
- ▶ Sequential Monte Carlo (SMC)
- ▶ MCEvidence

## 2. Marginal prior $\pi(\theta)$ and posterior $\mathcal{P}(\theta)$ densities:

- ▶ Histograms of samples
- ▶ Kernel density estimation
- ▶ Normalising flows / Diffusion models
- ▶ ...
- ▶ Emulators usually much faster than original likelihoods
- ▶ `margarine`: PyPI, [github.com/htjb/margarine](https://github.com/htjb/margarine)



# Nuisance marginalised likelihoods: Practice [2205.12841]

Harry Bevins



PhD→JRF

$$\mathcal{L}(\theta) = \frac{\mathcal{P}(\theta)\mathcal{Z}}{\pi(\theta)}$$

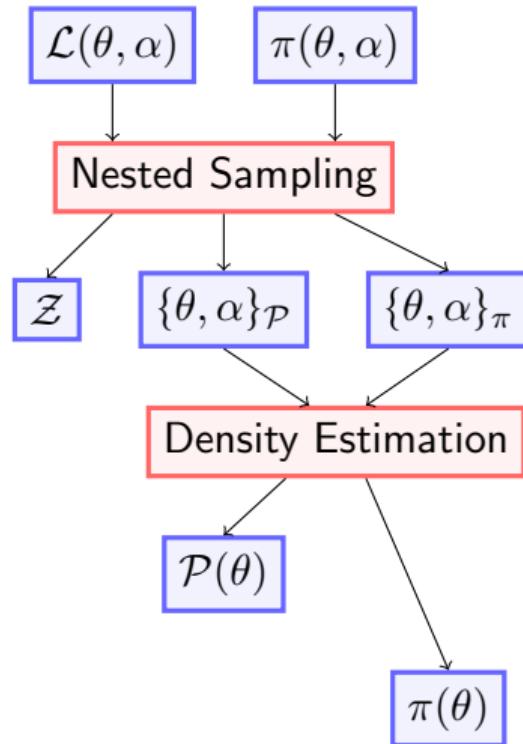
- ▶ To compute the nuisance marginalised likelihood, need:
  1. Bayesian evidence  $\mathcal{Z}$
  2. Marginal prior and posterior **densities**

## 1. Bayesian evidence $\mathcal{Z}$ : g

- ▶ Nested sampling
- ▶ Parallel tempering (pocomc, ptmcmc)
- ▶ Sequential Monte Carlo (SMC)
- ▶ MCEvidence

## 2. Marginal prior $\pi(\theta)$ and posterior $\mathcal{P}(\theta)$ densities:

- ▶ Histograms of samples
- ▶ Kernel density estimation
- ▶ Normalising flows / Diffusion models
- ▶ ...
- ▶ Emulators usually much faster than original likelihoods
- ▶ `margarine`: PyPI, [github.com/htjb/margarine](https://github.com/htjb/margarine)



# Nuisance marginalised likelihoods: Practice [2205.12841]

Harry Bevins



PhD→JRF

$$\mathcal{L}(\theta) = \frac{\mathcal{P}(\theta)\mathcal{Z}}{\pi(\theta)}$$

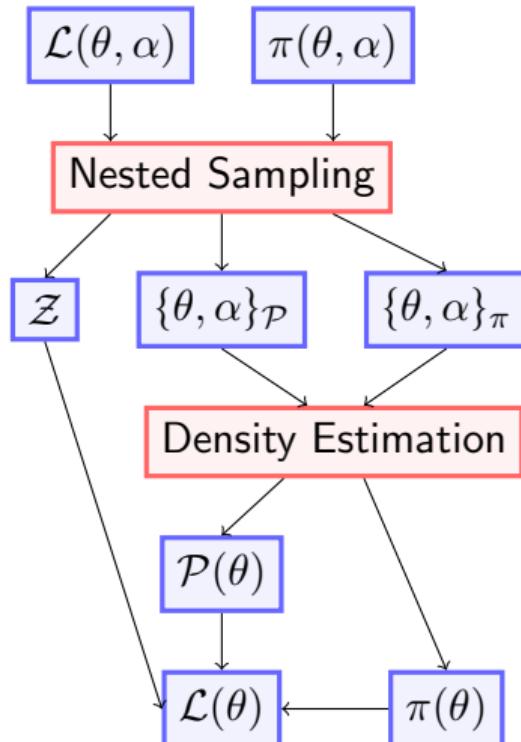
- ▶ To compute the nuisance marginalised likelihood, need:
  1. Bayesian evidence  $\mathcal{Z}$
  2. Marginal prior and posterior **densities**

## 1. Bayesian evidence $\mathcal{Z}$ : g

- ▶ Nested sampling
- ▶ Parallel tempering (pocomc, ptmcmc)
- ▶ Sequential Monte Carlo (SMC)
- ▶ MCEvidence

## 2. Marginal prior $\pi(\theta)$ and posterior $\mathcal{P}(\theta)$ densities:

- ▶ Histograms of samples
- ▶ Kernel density estimation
- ▶ Normalising flows / Diffusion models
- ▶ ...
- ▶ Emulators usually much faster than original likelihoods
- ▶ `margarine`: PyPI, [github.com/htjb/margarine](https://github.com/htjb/margarine)



# Nuisance marginalised likelihoods: Practice [2205.12841]

Harry Bevins



PhD → JRF

$$\mathcal{L}(\theta) = \frac{\mathcal{P}(\theta)\mathcal{Z}}{\pi(\theta)}$$

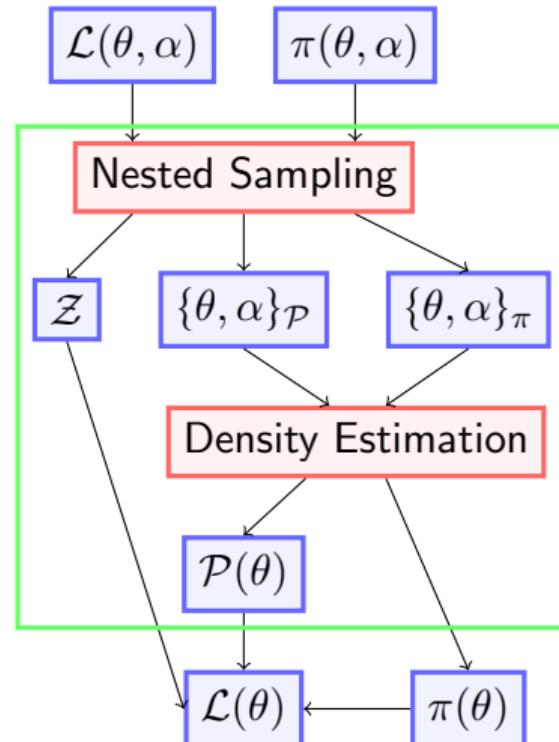
- ▶ To compute the nuisance marginalised likelihood, need:
  1. Bayesian evidence  $\mathcal{Z}$
  2. Marginal prior and posterior **densities**

## 1. Bayesian evidence $\mathcal{Z}$ : g

- ▶ Nested sampling
- ▶ Parallel tempering (pocomc, ptmcmc)
- ▶ Sequential Monte Carlo (SMC)
- ▶ MCEvidence

## 2. Marginal prior $\pi(\theta)$ and posterior $\mathcal{P}(\theta)$ densities:

- ▶ Histograms of samples
- ▶ Kernel density estimation
- ▶ Normalising flows / Diffusion models
- ▶ ...
- ▶ Emulators usually much faster than original likelihoods
- ▶ `margarine`: PyPI, [github.com/htjb/margarine](https://github.com/htjb/margarine)



- ▶ Library of pre-trained bijectors to be used as priors/emulators/nuisance marginalised likelihoods (DiRAC allocation unimpeded)
- ▶ e.g. easy to apply a *Planck*/DES/HERA/JWST prior or likelihood to your existing MCMC chains without needing to install the whole cosmology machinery.
- ▶ Hierarchical modelling:
  - ▶ Usually, have  $N$  objects, each with nuisance parameters  $\alpha_i$ , and shared parameters of interest  $\theta$ .
  - ▶ Likelihood  $\mathcal{L}(\{D_i\}|\theta, \{\alpha_i\}) = \prod_i^N \mathcal{L}_i(D_i|\theta, \alpha_i)$  has  $N \times \text{len}(\alpha_i) + \text{len}(\theta)$  parameters
  - ▶ Instead, break problem down into  $N$  runs on  $\text{len}(\theta) + \text{len}(\alpha_i)$  parameters, and one final one on  $\text{len}(\theta)$  parameters, using nuisance marginal likelihoods  $\mathcal{L}_i(\theta)$ .
  - ▶ In addition to computational tractability, also can perform model comparison with nuisance marginalised likelihoods.



## How fast in nested sampling?

$$T = T_{\mathcal{L}} \times n_{\text{live}} \times \mathcal{D}_{\text{KL}} \times f_{\text{sampler}}$$

## How accurate is nested sampling?

$$\sigma(\log \mathcal{Z}) \approx \sqrt{\mathcal{D}_{\text{KL}} / n_{\text{live}}}$$

in  $d$  dimensional parameter space:

$T_{\mathcal{L}}$ : likelihood eval time  $\sim \mathcal{O}(d)$

$n_{\text{live}}$ : number of live points  $\sim \mathcal{O}(d)$

$\mathcal{D}_{\text{KL}}$ : KL divergence from prior to posterior  $\approx \log V_{\pi} / V_{\mathcal{P}} \sim \mathcal{O}(d)$

$f_{\text{sampler}}$ : efficiency of point generation region  $\sim \mathcal{O}(e^{d/d_0})$  or path  $\sim \mathcal{O}(d)$

- ▶ Algorithmically improving  $f_{\text{sampler}}$  is only a fraction of the story!
- ▶  $\mathcal{D}_{\text{KL}}$  appears twice, so improvements here are quadratically important.
- ▶ Gradients give you  $d$  more information.

- ▶  $T \sim \mathcal{O}(d^4)$  whilst polynomial is far from ideal, although progress can be made on all fronts.

# Accelerated nested sampling with beta flows

Metha Prathaban

PhD



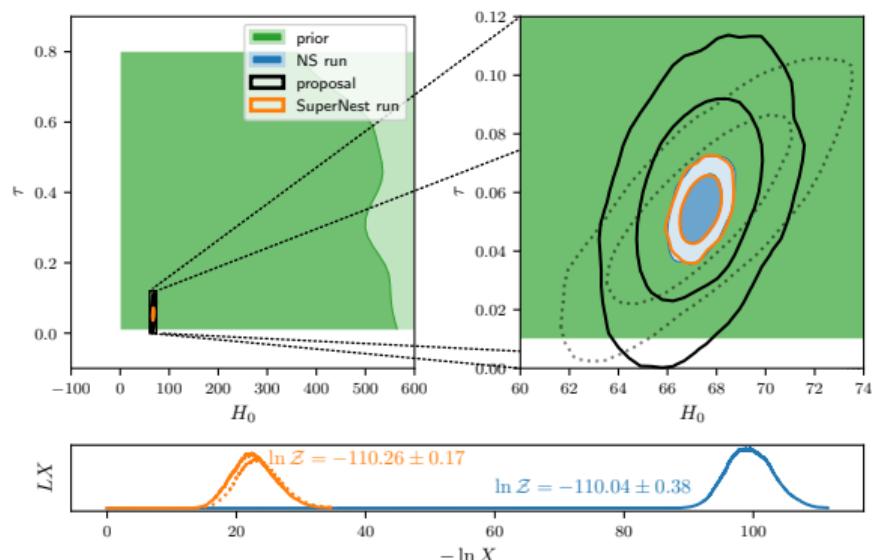
- ▶ For constant “run quality”  $\sigma$ ,

$$T = T_{\mathcal{L}} \times n_{\text{live}} \times \mathcal{D}_{\text{KL}} \times f_{\text{sampler}}, \quad \sigma \approx \sqrt{\mathcal{D}_{\text{KL}} / n_{\text{live}}}$$

$$\Rightarrow T = T_{\mathcal{L}} \times \sigma \times \mathcal{D}_{\text{KL}}^2 \times f_{\text{sampler}}$$

so if you can reduce the KL divergence, then quadratic gains to be made

- ▶ This can be crudely achieved by choosing a narrower prior  $\pi^*$  and then correcting the evidence  $\mathcal{Z} = \mathcal{Z}^* \frac{V_\pi^*}{V_\pi}$  (REACH [2210.07409])
- ▶ This can be made more sophisticated with SuperNest [2212.01760] & posterior repartitioning
- ▶ Recent application to gravitational waves by Metha Prathaban Ongoing work





- ▶ **very** recent work over the past month
- ▶ Have implemented a nested slice sampler in blackjax [[#755](#)]

```
1 pip install git+https://github.com/handley-lab/blackjax@nested_sampling
2 import blackjax.ns.adaptive
```

- ▶ parallelised over vmapped likelihood & prior evaluations
- ▶ Plugs into jim [[kazewong/jim](#)] and ripple [[2302.05329](#)]
- ▶ interested in finding use-cases for such a sampler this week
- ▶ Also interested in understanding current limitations/strengths of jax/GPU GW programming

# Conclusions



[github.com/handley-lab](https://github.com/handley-lab)