lsbi: linear simulation based inference

Will Handley
<wh260@cam.ac.uk>

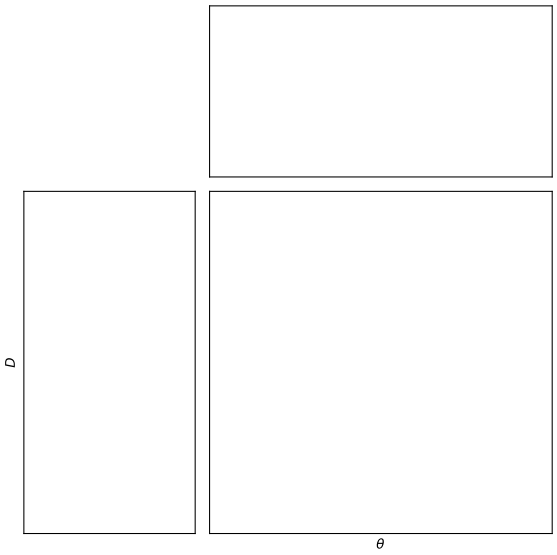
Royal Society University Research Fellow
Institute of Astronomy, University of Cambridge
Kavli Institute for Cosmology, Cambridge
Gonville & Caius College
willhandley.co.uk/talks

June 19, 2025


UNIVERSITY OF CAMBRIDGE

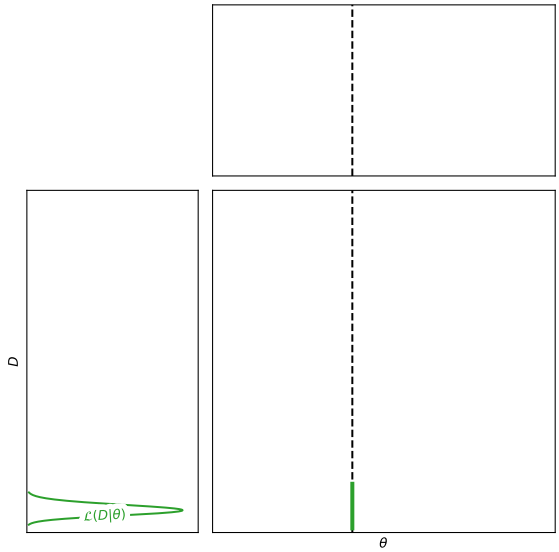

SBI: Simulation-based inference

- What do you do if you don't know $\mathcal{L}(D|\theta)$?
- If you have a simulator/forward model $\theta \to D$ 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 $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$ (also called the *unnormalised posterior*)
- Task of SBI is take joint $\mathcal{J}$ samples and learn posterior $\mathcal{P}(\theta|D)$
- Present state of the art achieves this using *neural networks*



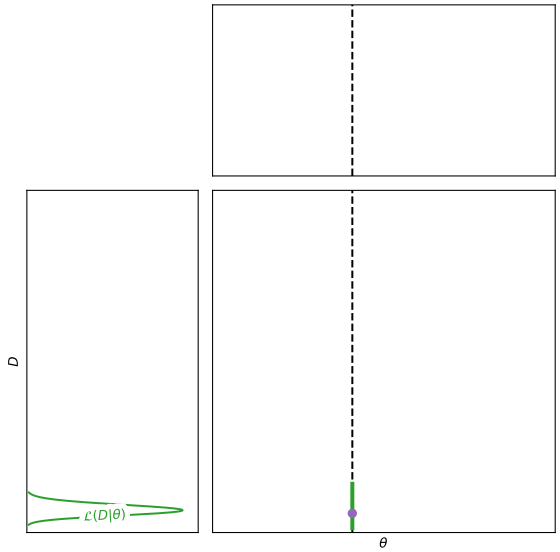
SBI: Simulation-based inference

- What do you do if you don't know $\mathcal{L}(D|\theta)$?
- If you have a simulator/forward model $\theta \to D$ 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 $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$ (also called the *unnormalised posterior*)
- Task of SBI is take joint $\mathcal{J}$ samples and learn posterior $\mathcal{P}(\theta|D)$
- Present state of the art achieves this using *neural networks*



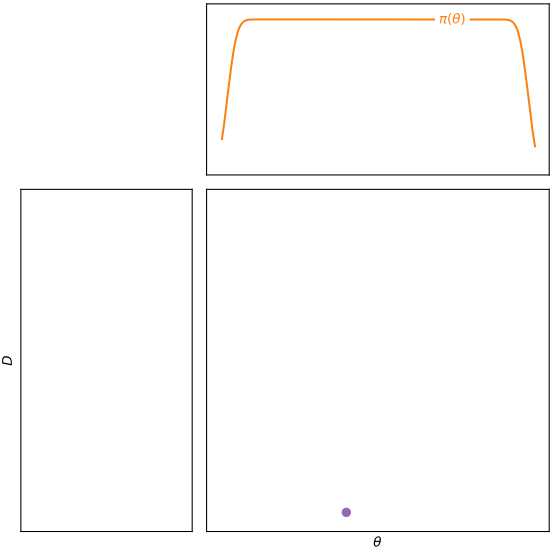
SBI: Simulation-based inference

- What do you do if you don't know $\mathcal{L}(D|\theta)$?
- If you have a simulator/forward model $\theta \to D$ 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 $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$ (also called the *unnormalised posterior*)
- Task of SBI is take joint $\mathcal{J}$ samples and learn posterior $\mathcal{P}(\theta|D)$
- Present state of the art achieves this using *neural networks*



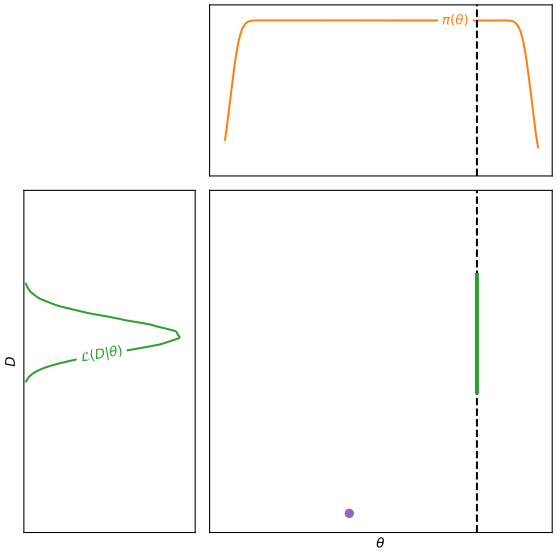
SBI: Simulation-based inference

- What do you do if you don't know $\mathcal{L}(D|\theta)$?
- If you have a simulator/forward model $\theta \to D$ 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 $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$ (also called the *unnormalised posterior*)
- Task of SBI is take joint $\mathcal{J}$ samples and learn posterior $\mathcal{P}(\theta|D)$
- Present state of the art achieves this using *neural networks*



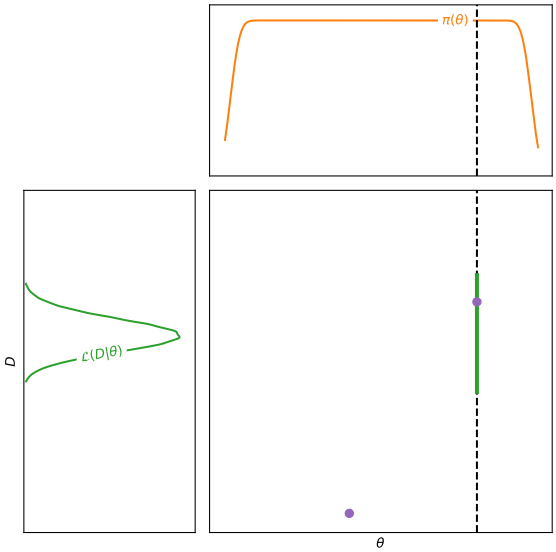
SBI: Simulation-based inference

- What do you do if you don't know $\mathcal{L}(D|\theta)$?
- If you have a simulator/forward model $\theta \to D$ 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 $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$ (also called the *unnormalised posterior*)
- Task of SBI is take joint $\mathcal{J}$ samples and learn posterior $\mathcal{P}(\theta|D)$
- Present state of the art achieves this using *neural networks*



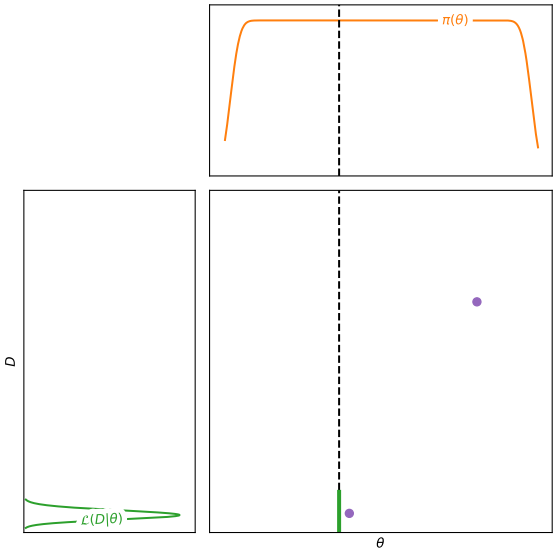
SBI: Simulation-based inference

- What do you do if you don't know $\mathcal{L}(D|\theta)$?
- If you have a simulator/forward model $\theta \to D$ 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 $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$ (also called the *unnormalised posterior*)
- Task of SBI is take joint $\mathcal{J}$ samples and learn posterior $\mathcal{P}(\theta|D)$
- Present state of the art achieves this using *neural networks*



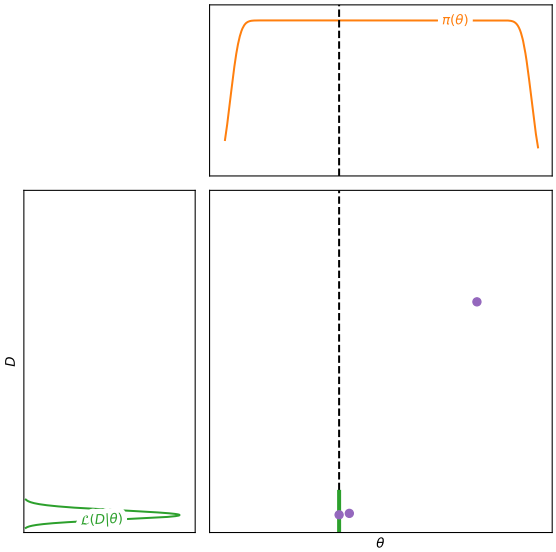
SBI: Simulation-based inference

- What do you do if you don't know $\mathcal{L}(D|\theta)$?
- If you have a simulator/forward model $\theta \to D$ 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 $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$ (also called the *unnormalised posterior*)
- Task of SBI is take joint $\mathcal{J}$ samples and learn posterior $\mathcal{P}(\theta|D)$
- Present state of the art achieves this using *neural networks*



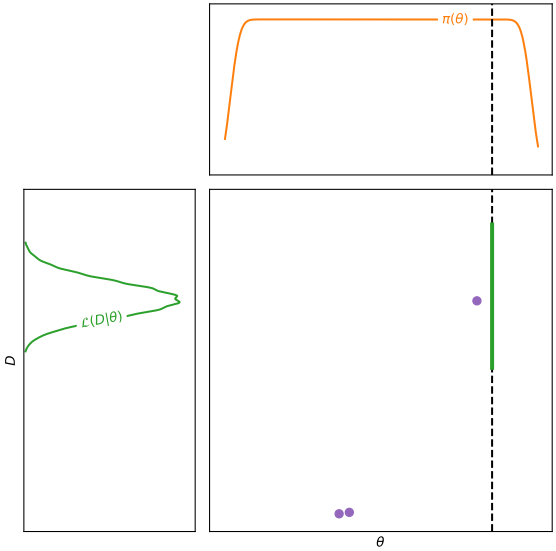
SBI: Simulation-based inference

- What do you do if you don't know $\mathcal{L}(D|\theta)$?
- If you have a simulator/forward model $\theta \to D$ 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 $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$ (also called the *unnormalised posterior*)
- Task of SBI is take joint $\mathcal{J}$ samples and learn posterior $\mathcal{P}(\theta|D)$
- Present state of the art achieves this using *neural networks*



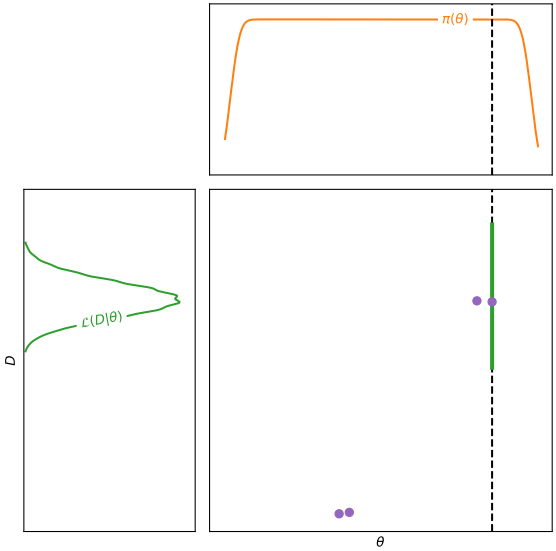
SBI: Simulation-based inference

- What do you do if you don't know $\mathcal{L}(D|\theta)$?
- If you have a simulator/forward model $\theta \to D$ 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 $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$ (also called the *unnormalised posterior*)
- Task of SBI is take joint $\mathcal{J}$ samples and learn posterior $\mathcal{P}(\theta|D)$
- Present state of the art achieves this using *neural networks*



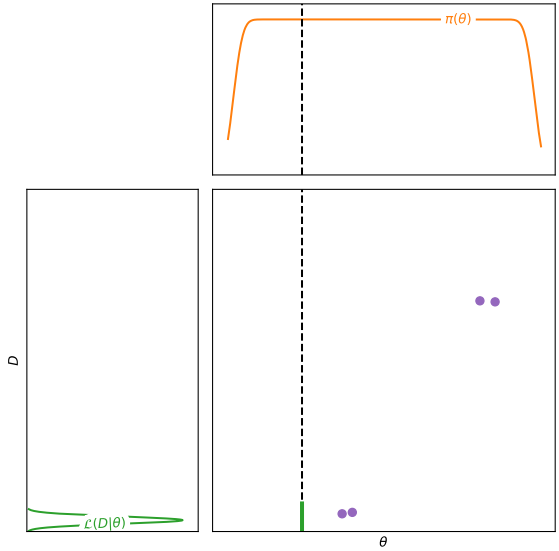
SBI: Simulation-based inference

- What do you do if you don't know $\mathcal{L}(D|\theta)$?
- If you have a simulator/forward model $\theta \to D$ 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 $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$ (also called the *unnormalised posterior*)
- Task of SBI is take joint $\mathcal{J}$ samples and learn posterior $\mathcal{P}(\theta|D)$
- Present state of the art achieves this using *neural networks*



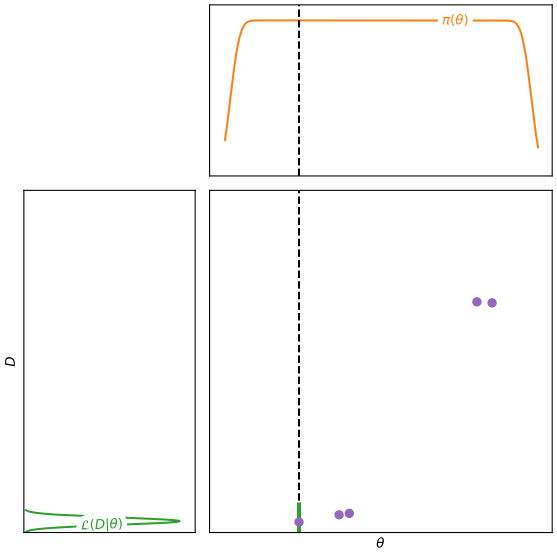
SBI: Simulation-based inference

- What do you do if you don't know $\mathcal{L}(D|\theta)$?
- If you have a simulator/forward model $\theta \to D$ 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 $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$ (also called the *unnormalised posterior*)
- Task of SBI is take joint $\mathcal{J}$ samples and learn posterior $\mathcal{P}(\theta|D)$
- Present state of the art achieves this using *neural networks*



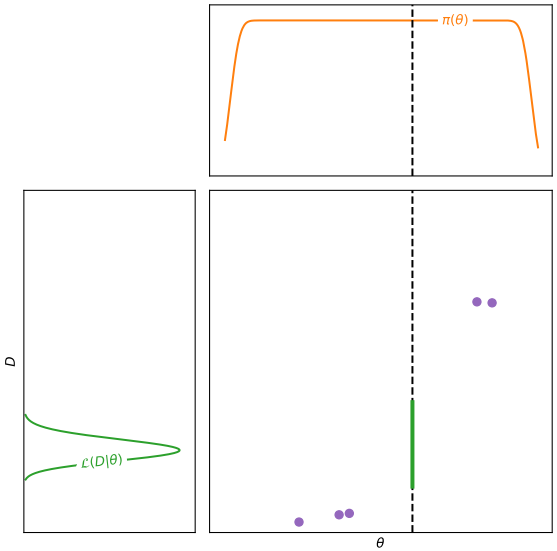
SBI: Simulation-based inference

- What do you do if you don't know $\mathcal{L}(D|\theta)$?
- If you have a simulator/forward model $\theta \to D$ 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 $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$ (also called the *unnormalised posterior*)
- Task of SBI is take joint $\mathcal{J}$ samples and learn posterior $\mathcal{P}(\theta|D)$
- Present state of the art achieves this using *neural networks*



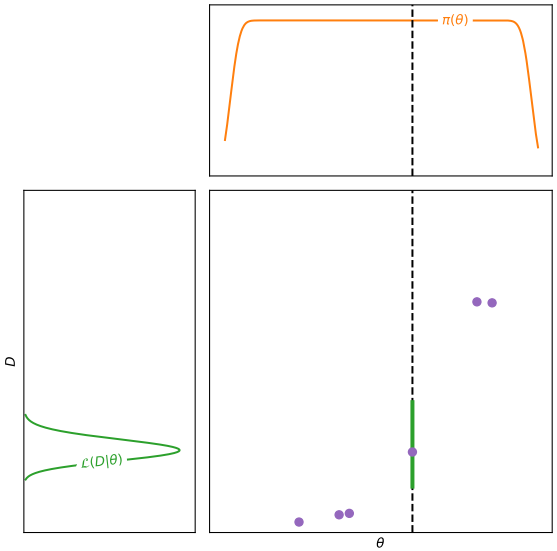
SBI: Simulation-based inference

- What do you do if you don't know $\mathcal{L}(D|\theta)$?
- If you have a simulator/forward model $\theta \to D$ 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 $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$ (also called the *unnormalised posterior*)
- Task of SBI is take joint $\mathcal{J}$ samples and learn posterior $\mathcal{P}(\theta|D)$
- Present state of the art achieves this using *neural networks*



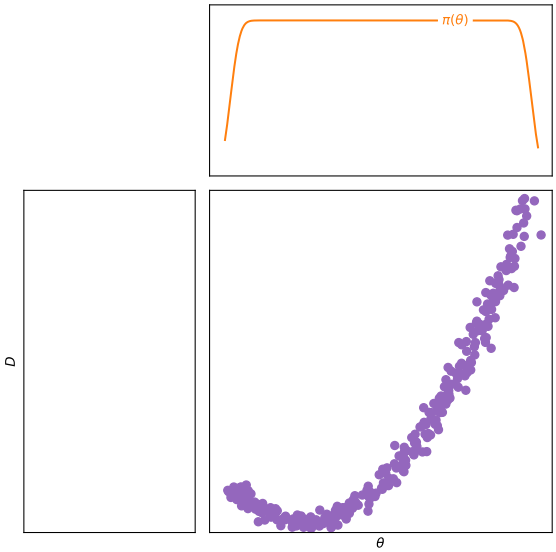
SBI: Simulation-based inference

- What do you do if you don't know $\mathcal{L}(D|\theta)$?
- If you have a simulator/forward model $\theta \to D$ 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 $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$ (also called the *unnormalised posterior*)
- Task of SBI is take joint $\mathcal{J}$ samples and learn posterior $\mathcal{P}(\theta|D)$
- Present state of the art achieves this using *neural networks*



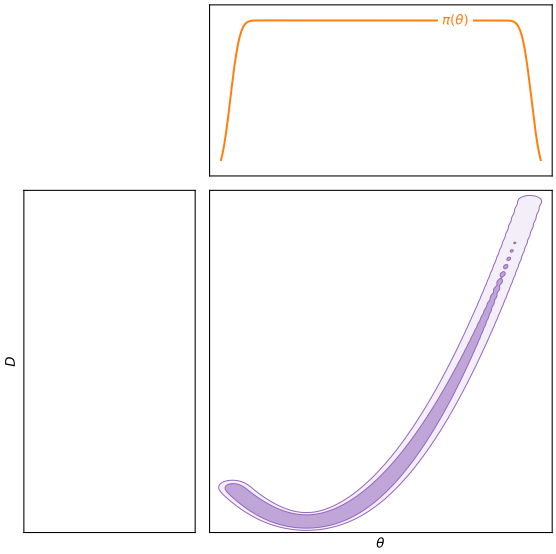
SBI: Simulation-based inference

- What do you do if you don't know $\mathcal{L}(D|\theta)$?
- If you have a simulator/forward model $\theta \to D$ 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 $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$ (also called the *unnormalised posterior*)
- Task of SBI is take joint $\mathcal{J}$ samples and learn posterior $\mathcal{P}(\theta|D)$
- Present state of the art achieves this using *neural networks*



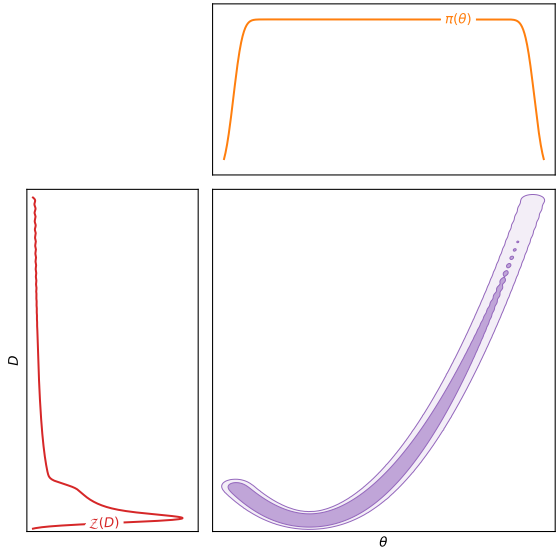
SBI: Simulation-based inference

- What do you do if you don't know $\mathcal{L}(D|\theta)$?
- If you have a simulator/forward model $\theta \to D$ 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 $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$ (also called the *unnormalised posterior*)
- Task of SBI is take joint $\mathcal{J}$ samples and learn posterior $\mathcal{P}(\theta|D)$
- Present state of the art achieves this using *neural networks*



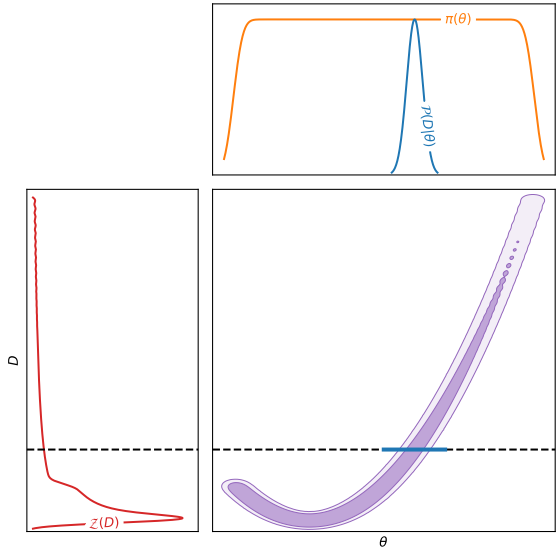
SBI: Simulation-based inference

- What do you do if you don't know $\mathcal{L}(D|\theta)$?
- If you have a simulator/forward model $\theta \to D$ 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 $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$ (also called the *unnormalised posterior*)
- Task of SBI is take joint $\mathcal{J}$ samples and learn posterior $\mathcal{P}(\theta|D)$
- Present state of the art achieves this using *neural networks*



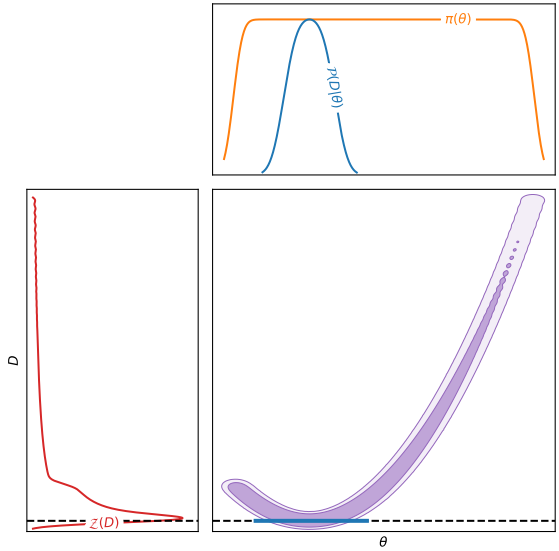
SBI: Simulation-based inference

- What do you do if you don't know $\mathcal{L}(D|\theta)$?
- If you have a simulator/forward model $\theta \to D$ 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 $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$ (also called the *unnormalised posterior*)
- Task of SBI is take joint $\mathcal{J}$ samples and learn posterior $\mathcal{P}(\theta|D)$
- Present state of the art achieves this using *neural networks*



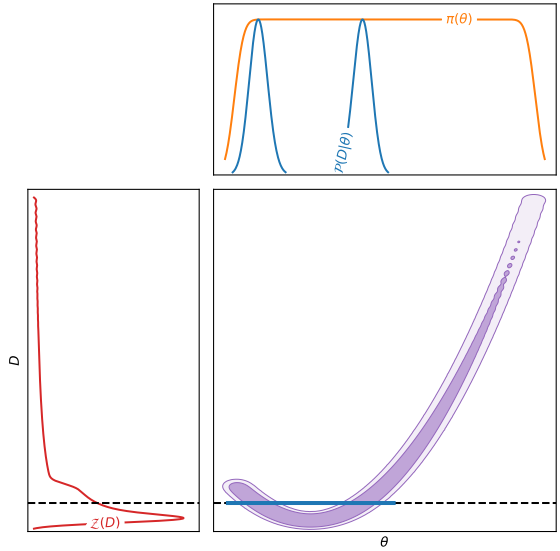
SBI: Simulation-based inference

- What do you do if you don't know $\mathcal{L}(D|\theta)$?
- If you have a simulator/forward model $\theta \to D$ 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 $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$ (also called the *unnormalised posterior*)
- Task of SBI is take joint $\mathcal{J}$ samples and learn posterior $\mathcal{P}(\theta|D)$
- Present state of the art achieves this using *neural networks*



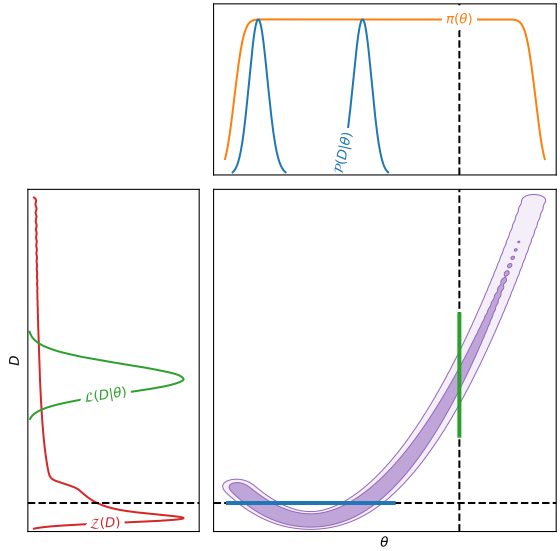
SBI: Simulation-based inference

- What do you do if you don't know $\mathcal{L}(D|\theta)$?
- If you have a simulator/forward model $\theta \to D$ 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 $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$ (also called the *unnormalised posterior*)
- Task of SBI is take joint $\mathcal{J}$ samples and learn posterior $\mathcal{P}(\theta|D)$
- Present state of the art achieves this using *neural networks*



SBI: Simulation-based inference

- What do you do if you don't know $\mathcal{L}(D|\theta)$?
- If you have a simulator/forward model $\theta \to D$ 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 $\mathcal{J}(\theta, D) = \mathcal{L}(D|\theta)\pi(\theta)$ (also called the *unnormalised posterior*)
- Task of SBI is take joint $\mathcal{J}$ samples and learn posterior $\mathcal{P}(\theta|D)$
- Present state of the art achieves this using *neural networks*



Why linear SBI?

If neural networks are all that, why should we consider the regressive step of going back to linear versions?

- It is **pedagogically** helpful
  - separates general principles of SBI from the details of neural networks
  - (particularly for ML skeptics)
- It is **practically** useful
  - for producing expressive examples with known ground truths
- It is **pragmatically** useful
  - competitive with neural approaches in terms of accuracy
  - faster and more interpretable

- Linear generative model (m, M, C)

$$D \sim \mathcal{N}(m + M\theta, C)$$

where:

θ : n dimensional parameters
D : d dimensional data
M : $d \times n$ transfer matrix
m : d-dimensional shift
C : $d \times d$ data covariance

- Linear generative model (m, M, C)

$$D \sim \mathcal{N}(m + M\theta, C)$$

where:

θ : n dimensional parameters
D : d dimensional data
M : $d \times n$ transfer matrix
m : d-dimensional shift
C : $d \times d$ data covariance

- k Simulations

$$S = \{(\theta_i, D_i) : i = 1, \ldots, k\}$$

Linear Simulation Based Inference

Mathematical setup and Bayesian Inference

- Linear generative model (m, M, C)

$$D \sim \mathcal{N}(m + M\theta, C)$$

where:

θ : n dimensional parameters
D : d dimensional data
M : $d \times n$ transfer matrix
m : d-dimensional shift
C : $d \times d$ data covariance

- k Simulations

$$S = \{(\theta_i, D_i) : i = 1, \ldots, k\}$$

- We infer (m, M, C) from simulations S. The likelihood is:

$$P(\{D_i\}|\{\theta_i\}|m, M, C) = \prod_i \mathcal{N}(D_i|m+M\theta_i, C)$$

- Linear generative model (m, M, C)

$$D \sim \mathcal{N}(m + M\theta, C)$$

where:

θ : n dimensional parameters
D : d dimensional data
M : $d \times n$ transfer matrix
m : d-dimensional shift
C : $d \times d$ data covariance

- k Simulations

$$S = \{(\theta_i, D_i) : i = 1, \ldots, k\}$$

- We infer (m, M, C) from simulations S. The likelihood is:

$$P(\{D_i\}|\{\theta_i\}|m, M, C) = \prod_i \mathcal{N}(D_i|m+M\theta_i, C)$$

- We use a conjugate prior π on (m, M, C) (with hyperparameters $\lambda_0, D_0, \theta_0, M_0, \Omega_0, \nu_0, \Psi_0$):

$$\pi : \begin{cases} m|M, C, \sim \mathcal{N}(D_0 - M\theta_0, \frac{1}{\lambda_0}C), \\ M|C, \sim \mathcal{MN}(M_0, C, \Omega_0^{-1}), \\ C \sim \mathcal{W}_{\nu_0}^{-1}(\Psi_0) \end{cases}$$

- Linear generative model (m, M, C)

$$D \sim \mathcal{N}(m + M\theta, C)$$

where:

θ : n dimensional parameters
D : d dimensional data
M : $d \times n$ transfer matrix
m : d-dimensional shift
C : $d \times d$ data covariance

- k Simulations

$$S = \{(\theta_i, D_i) : i = 1, \ldots, k\}$$

- We infer (m, M, C) from simulations S. The likelihood is:

$$P(\{D_i\}|\{\theta_i\}|m, M, C) = \prod_i \mathcal{N}(D_i|m+M\theta_i, C)$$

- We use a conjugate prior π on (m, M, C) (with hyperparameters $\lambda_0, D_0, \theta_0, M_0, \Omega_0, \nu_0, \Psi_0$):

$$\pi : \begin{cases} m|M, C, \sim \mathcal{N}(D_0 - M\theta_0, \frac{1}{\lambda_0}C), \\ M|C, \sim \mathcal{MN}(M_0, C, \Omega_0^{-1}), \\ C \sim \mathcal{W}_{\nu_0}^{-1}(\Psi_0) \end{cases}$$

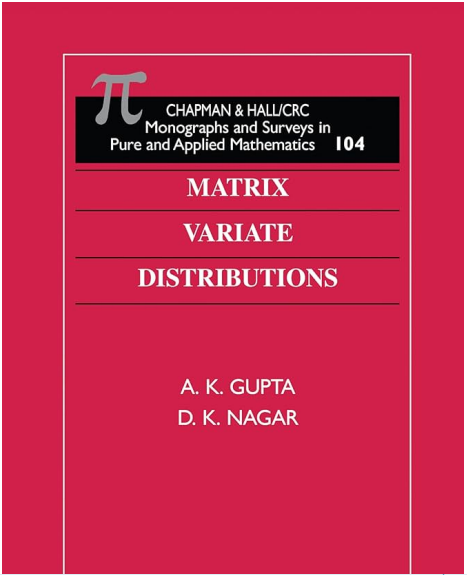
LSBI vs Neural SBI: Explainability

Neural SBI challenges:

- Black-box approximations
- Hyperparameter tuning required
- Saturation at $-5 < \log r < 5$
- Training instability
- Difficult convergence diagnostics

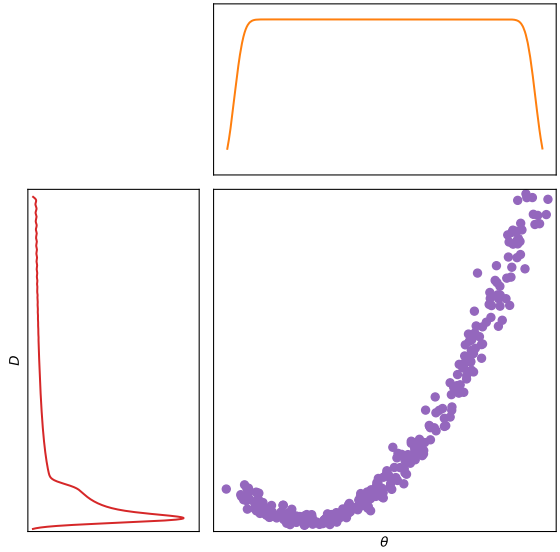
LSBI advantages:

- Transparent mathematical foundation
- Self-tuning (no hyperparameters)



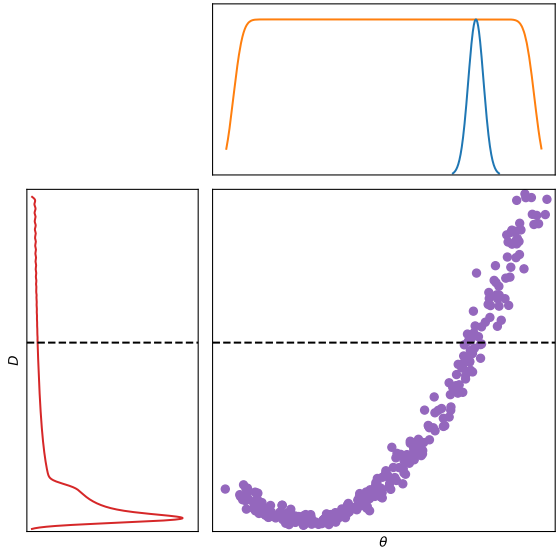
Sequential LSBI

- Same model as before
- Mark the observed data D_{obs}
- Fit a model using `lsbi`
- Evaluate the posterior (cheap as linear)
- Now use this posterior to pick $\{\theta_i\}$
- Generate $\{D_i\}$ from original simulator
- Fit `lsbi` to these
- Evaluate the new posterior
- Iterate until convergence



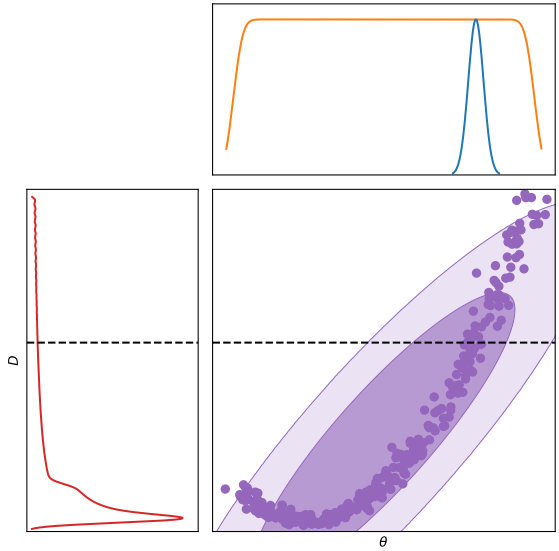
Sequential LSBI

- Same model as before
- Mark the observed data D_{obs}
- Fit a model using `lsbi`
- Evaluate the posterior (cheap as linear)
- Now use this posterior to pick $\{\theta_i\}$
- Generate $\{D_i\}$ from original simulator
- Fit `lsbi` to these
- Evaluate the new posterior
- Iterate until convergence



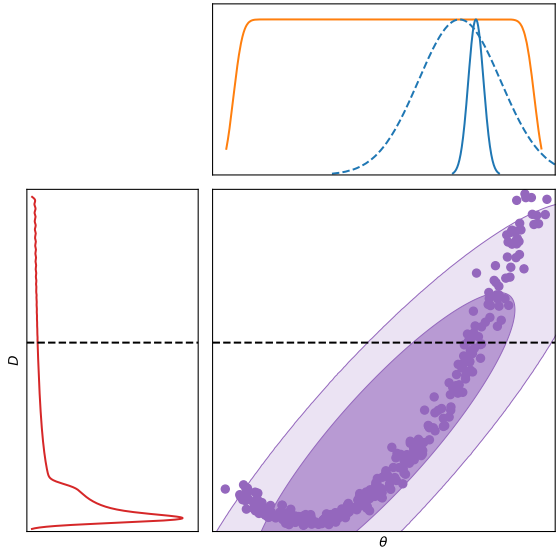
Sequential LSBI

- Same model as before
- Mark the observed data D_{obs}
- Fit a model using `lsbi`
- Evaluate the posterior (cheap as linear)
- Now use this posterior to pick $\{\theta_i\}$
- Generate $\{D_i\}$ from original simulator
- Fit `lsbi` to these
- Evaluate the new posterior
- Iterate until convergence



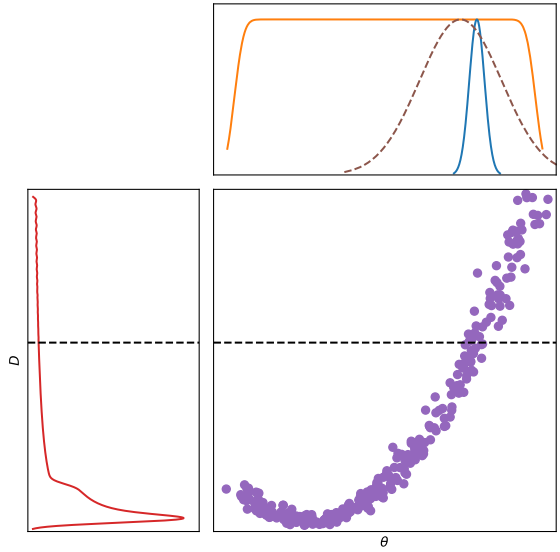
Sequential LSBI

- Same model as before
- Mark the observed data D_{obs}
- Fit a model using `lsbi`
- Evaluate the posterior (cheap as linear)
- Now use this posterior to pick $\{\theta_i\}$
- Generate $\{D_i\}$ from original simulator
- Fit `lsbi` to these
- Evaluate the new posterior
- Iterate until convergence



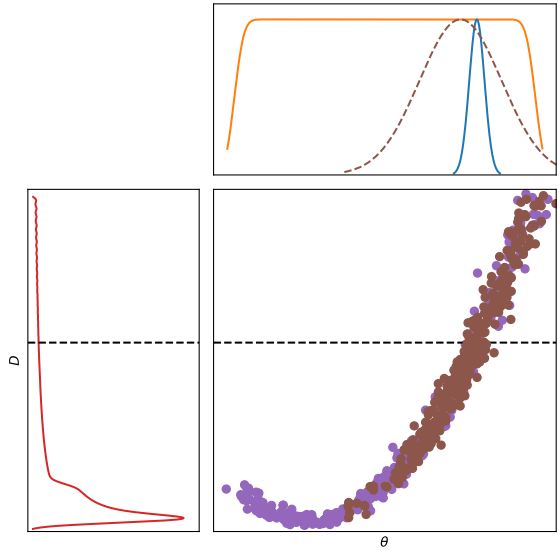
Sequential LSBI

- Same model as before
- Mark the observed data D_{obs}
- Fit a model using `lsbi`
- Evaluate the posterior (cheap as linear)
- Now use this posterior to pick $\{\theta_i\}$
- Generate $\{D_i\}$ from original simulator
- Fit `lsbi` to these
- Evaluate the new posterior
- Iterate until convergence



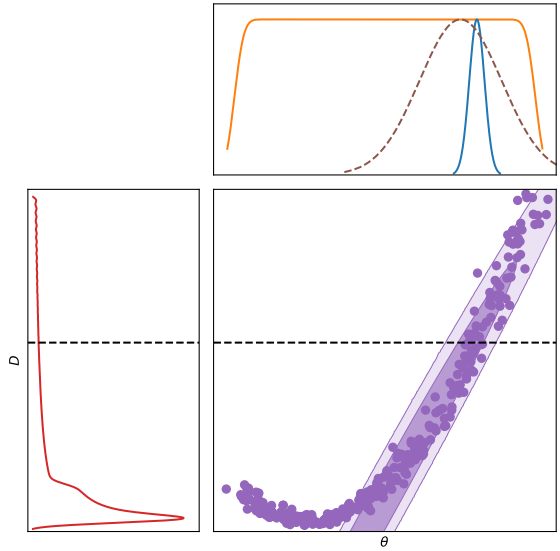
Sequential LSBI

- Same model as before
- Mark the observed data D_{obs}
- Fit a model using `lsbi`
- Evaluate the posterior (cheap as linear)
- Now use this posterior to pick $\{\theta_i\}$
- Generate $\{D_i\}$ from original simulator
- Fit `lsbi` to these
- Evaluate the new posterior
- Iterate until convergence



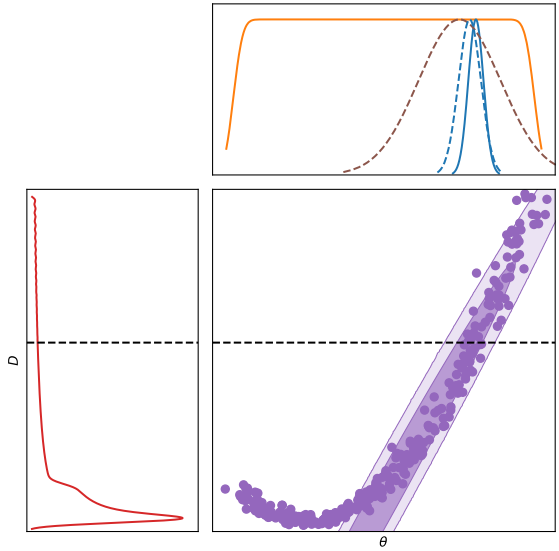
Sequential LSBI

- Same model as before
- Mark the observed data D_{obs}
- Fit a model using `lsbi`
- Evaluate the posterior (cheap as linear)
- Now use this posterior to pick $\{\theta_i\}$
- Generate $\{D_i\}$ from original simulator
- Fit `lsbi` to these
- Evaluate the new posterior
- Iterate until convergence



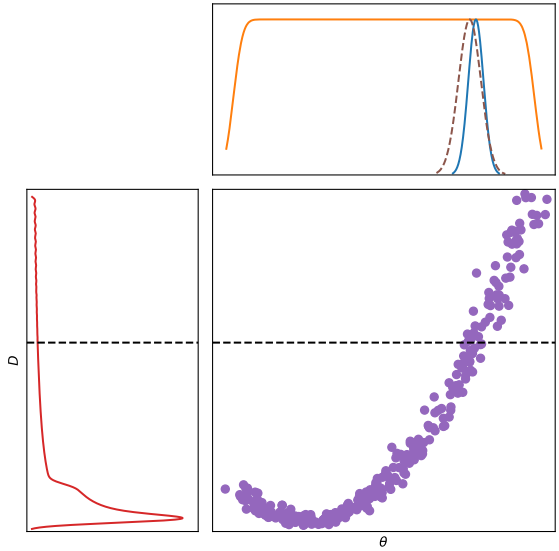
Sequential LSBI

- Same model as before
- Mark the observed data D_{obs}
- Fit a model using `lsbi`
- Evaluate the posterior (cheap as linear)
- Now use this posterior to pick $\{\theta_i\}$
- Generate $\{D_i\}$ from original simulator
- Fit `lsbi` to these
- Evaluate the new posterior
- Iterate until convergence



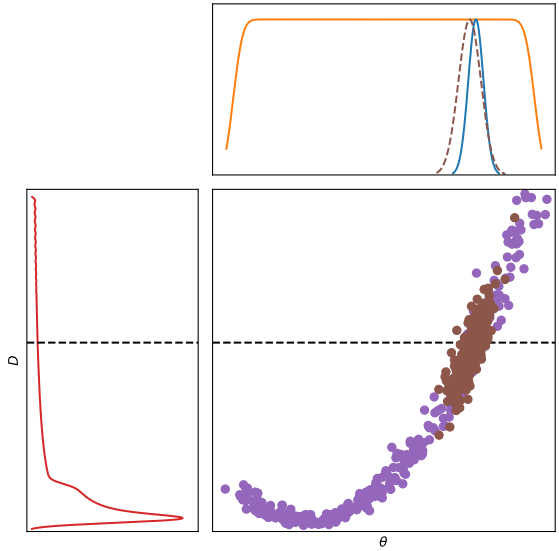
Sequential LSBI

- Same model as before
- Mark the observed data D_{obs}
- Fit a model using `lsbi`
- Evaluate the posterior (cheap as linear)
- Now use this posterior to pick $\{\theta_i\}$
- Generate $\{D_i\}$ from original simulator
- Fit `lsbi` to these
- Evaluate the new posterior
- Iterate until convergence



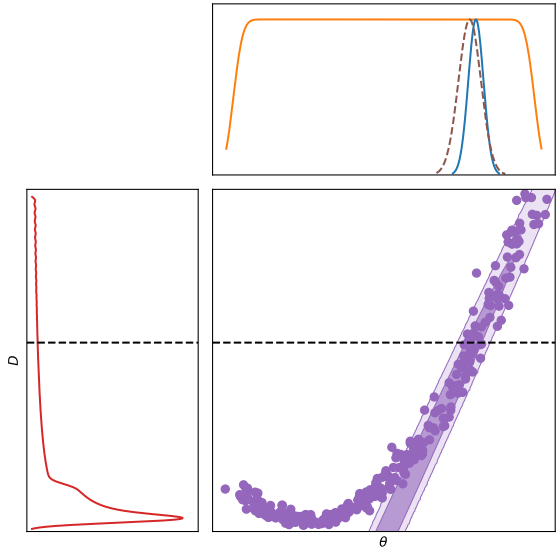
Sequential LSBI

- Same model as before
- Mark the observed data D_{obs}
- Fit a model using `lsbi`
- Evaluate the posterior (cheap as linear)
- Now use this posterior to pick $\{\theta_i\}$
- Generate $\{D_i\}$ from original simulator
- Fit `lsbi` to these
- Evaluate the new posterior
- Iterate until convergence



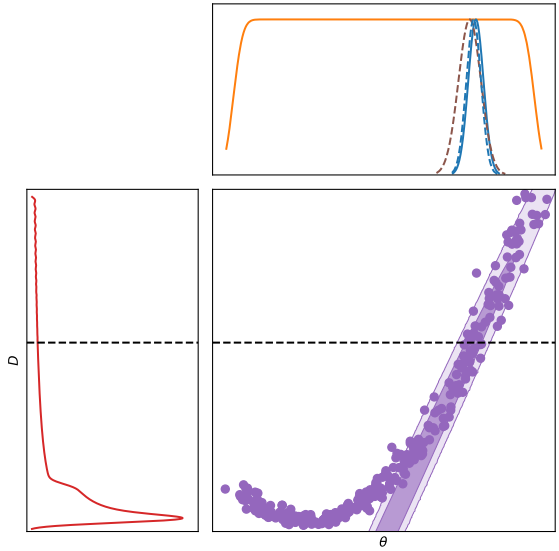
Sequential LSBI

- Same model as before
- Mark the observed data D_{obs}
- Fit a model using `lsbi`
- Evaluate the posterior (cheap as linear)
- Now use this posterior to pick $\{\theta_i\}$
- Generate $\{D_i\}$ from original simulator
- Fit `lsbi` to these
- Evaluate the new posterior
- Iterate until convergence



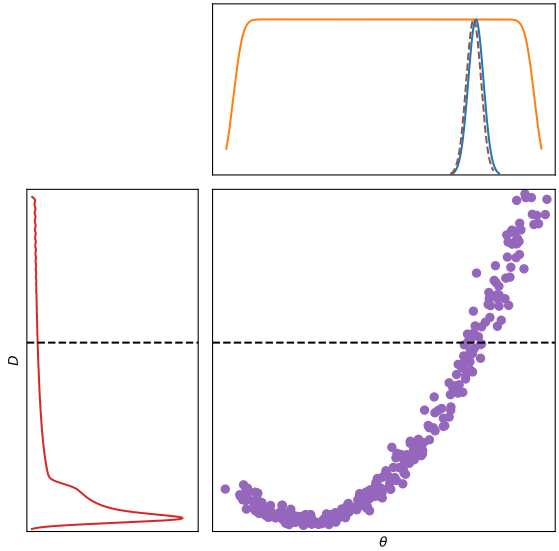
Sequential LSBI

- Same model as before
- Mark the observed data D_{obs}
- Fit a model using `lsbi`
- Evaluate the posterior (cheap as linear)
- Now use this posterior to pick $\{\theta_i\}$
- Generate $\{D_i\}$ from original simulator
- Fit `lsbi` to these
- Evaluate the new posterior
- Iterate until convergence



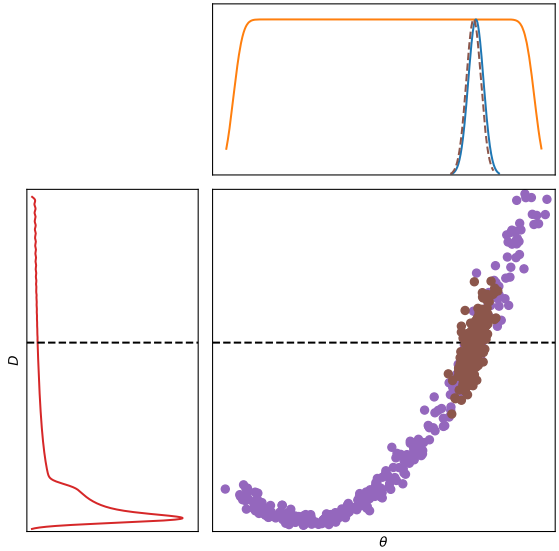
Sequential LSBI

- Same model as before
- Mark the observed data D_{obs}
- Fit a model using `lsbi`
- Evaluate the posterior (cheap as linear)
- Now use this posterior to pick $\{\theta_i\}$
- Generate $\{D_i\}$ from original simulator
- Fit `lsbi` to these
- Evaluate the new posterior
- Iterate until convergence



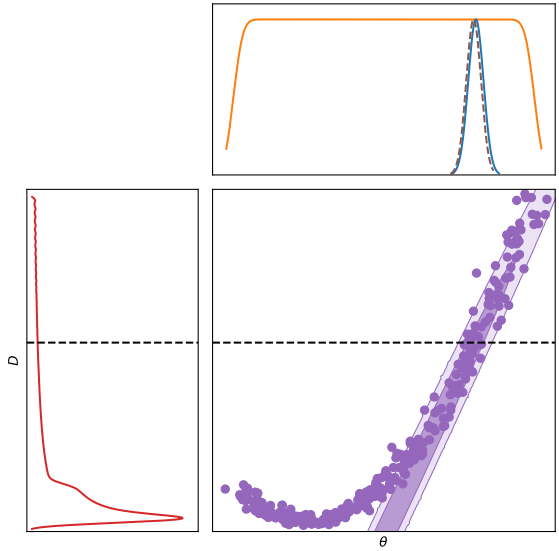
Sequential LSBI

- Same model as before
- Mark the observed data D_{obs}
- Fit a model using `lsbi`
- Evaluate the posterior (cheap as linear)
- Now use this posterior to pick $\{\theta_i\}$
- Generate $\{D_i\}$ from original simulator
- Fit `lsbi` to these
- Evaluate the new posterior
- Iterate until convergence



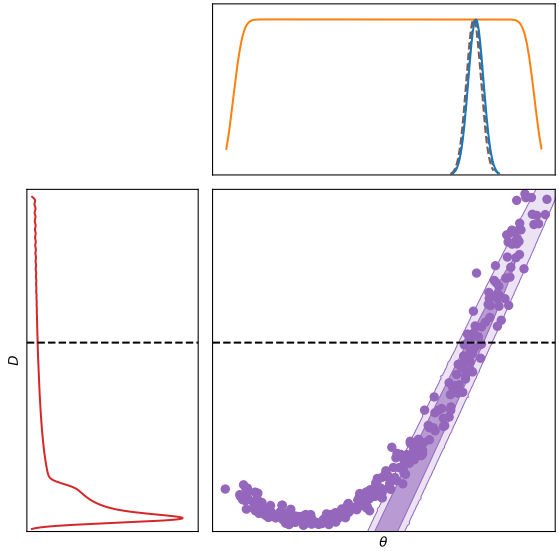
Sequential LSBI

- Same model as before
- Mark the observed data D_{obs}
- Fit a model using `lsbi`
- Evaluate the posterior (cheap as linear)
- Now use this posterior to pick $\{\theta_i\}$
- Generate $\{D_i\}$ from original simulator
- Fit `lsbi` to these
- Evaluate the new posterior
- Iterate until convergence



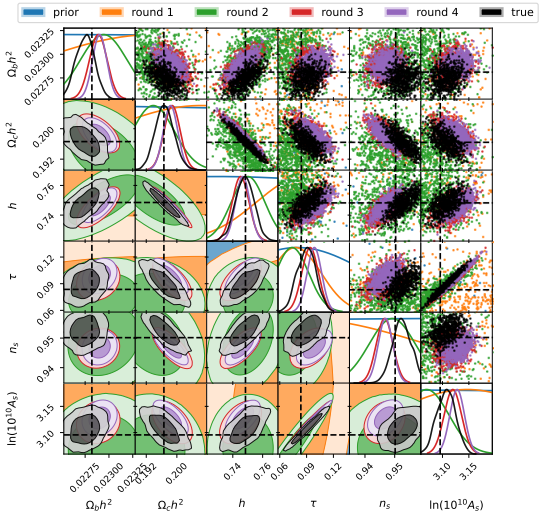
Sequential LSBI

- Same model as before
- Mark the observed data D_{obs}
- Fit a model using `lsbi`
- Evaluate the posterior (cheap as linear)
- Now use this posterior to pick $\{\theta_i\}$
- Generate $\{D_i\}$ from original simulator
- Fit `lsbi` to these
- Evaluate the new posterior
- Iterate until convergence



Cosmology Example: ΛCDM from CMB

- Apply LSBI to cosmic microwave background
- Cosmic-variance limited, temperature-only, full sky
- $n = 6$ parameters, $d = 2500$ data dimensions
- Sequential improvement around observed data
- Competitive with traditional methods



`lsbi`: Code & Implementation

- `lsbi` is a pip-installable python package
- Extends `scipy.stats.multivariate_normal`
  - Vectorised distributions with broadcastable arrays
  - `.marginalise(...)` and `.condition(...)` methods
  - Built-in plotting functionality
- Implements `LinearModel` class with interpretable methods:
  - `.prior()`, `.likelihood(theta)`, `.posterior(D)`
  - `.evidence()` - all return distributions
- Also implements `MixtureModel` for nonlinear effects
- [2501.03921] and `github.com/handley-lab/lsbi`

Results & Recent Work: [2501.03921]

Performance, scaling, and key contributions

Key advantages:

- Analytical solutions (no iterative training)
- Transparent mathematical operations
- Competitive with neural methods
- Faster than neural training, no GPU requirements
- Self-tuning (no hyperparameter search)
- Handles $n \approx 6$, $d \approx 2500$ problems

Key contributions ([2501.03921]):

- Mathematical foundation using matrix-variate distributions
- Sequential refinement algorithm for nonlinear problems

Results demonstrate:

- Competitive accuracy on standard benchmarks
- Significant computational speedup
- Enhanced interpretability
- Pedagogical value for understanding SBI principles

Available at:

- [2501.03921]
- `github.com/handley-lab/lsbi`

Master's student project led to

Conclusions

github.com/handley-lab/group

- **lsbi** provides explainable alternative to neural SBI methods
- Separates SBI principles from ML implementation details
- Competitive accuracy with interpretable mathematical foundation
- Available: [2501.03921] and github.com/handley-lab/lsbi
- Future: realistic simulations, model comparison, importance sampling