

# Simulation-based inference

Cyrille Doux

LPSC Grenoble (CNRS/IN2P3)



Copilot prompt: particle physics and cosmology simulations flowing into neural networks like Van Gogh

IN2P3 School of statistics - June 5<sup>th</sup> 2026



# Prerequisites

1. **Continuous random variables:** probability density functions, marginal and conditional distributions

# Prerequisites

1. **Continuous random variables**: probability density functions, marginal and conditional distributions
2. **Likelihood** functions and standard forms: Poisson, (multivariate) Gaussian, etc.

*Romain, Lydia, Nicolas*

# Prerequisites

1. **Continuous random variables**: probability density functions, marginal and conditional distributions
2. **Likelihood** functions and standard forms: Poisson, (multivariate) Gaussian, etc. *Romain, Lydia, Nicolas*
3. Bayesian inference: **prior** and **posterior** distributions, **Bayes' theorem** *Florian and I*

# Prerequisites

1. **Continuous random variables**: probability density functions, marginal and conditional distributions
2. **Likelihood** functions and standard forms: Poisson, (multivariate) Gaussian, etc. *Romain, Lydia, Nicolas*
3. Bayesian inference: **prior** and **posterior** distributions, **Bayes' theorem** *Florian and I*
4. **Sampling techniques**: rejection sampling, Markov chain Monte-Carlo (MCMC)

# Prerequisites

1. **Continuous random variables**: probability density functions, marginal and conditional distributions
2. **Likelihood** functions and standard forms: Poisson, (multivariate) Gaussian, etc. *Romain, Lydia, Nicolas*
3. Bayesian inference: **prior** and **posterior** distributions, **Bayes' theorem** *Florian and I*
4. **Sampling techniques**: rejection sampling, Markov chain Monte-Carlo (MCMC)
5. **Neural networks**: dense neural networks, classification/regression, training *Florian*

# Prerequisites

1. **Continuous random variables**: probability density functions, marginal and conditional distributions
2. **Likelihood** functions and standard forms: Poisson, (multivariate) Gaussian, etc. *Romain, Lydia, Nicolas*
3. Bayesian inference: **prior** and **posterior** distributions, **Bayes' theorem** *Florian and I*
4. **Sampling techniques**: rejection sampling, Markov chain Monte-Carlo (MCMC)
5. **Neural networks**: dense neural networks, classification/regression, training *Florian*
6. (optional) **Generative** models: generative adversarial networks, normalizing flows, etc.

# Learning objectives

We will learn:

# Learning objectives

We will learn:

1. What problem(s) SBI tries to solve

# Learning objectives

We will learn:

1. What problem(s) SBI tries to solve
2. How SBI uses simulated and observed data to constrain models

# Learning objectives

We will learn:

1. What problem(s) SBI tries to solve
2. How SBI uses simulated and observed data to constrain models
3. How to implement a very simple SBI workflow in python ([notebook](#))

# Learning objectives

We will learn:

1. What problem(s) SBI tries to solve
2. How SBI uses simulated and observed data to constrain models
3. How to implement a very simple SBI workflow in python ([notebook](#))
4. Which questions to ask before applying SBI techniques to your work

# Learning objectives

We will learn:

1. What problem(s) SBI tries to solve
2. How SBI uses simulated and observed data to constrain models
3. How to implement a very simple SBI workflow in python ([notebook](#))
4. Which questions to ask before applying SBI techniques to your work
5. What are refinements and research directions

# Introduction

Motivations and basic idea



# Starting point: the intractable likelihood problem

# Starting point: the intractable likelihood problem

- ▶ Two teams

# Starting point: the intractable likelihood problem

- ▶ Two teams
  - ▶ Frequentist

# Starting point: the intractable likelihood problem

- ▶ Two teams
  - ▶ Frequentist
    - ▶ Assume there exists some true  $\theta^*$  value

# Starting point: the intractable likelihood problem

- ▶ Two teams

- ▶ Frequentist

- ▶ Assume there exists some true  $\theta^*$  value

- ▶ Use likelihood (ratios) to compute **confidence intervals**  $I(\text{data}, \alpha)$ , such that  $P(\theta^* \in I(\text{data}, \alpha)) = \alpha$

# Starting point: the intractable likelihood problem

- ▶ Two teams

- ▶ Frequentist

- ▶ Assume there exists some true  $\theta^*$  value

- ▶ Use likelihood (ratios) to compute **confidence intervals**  $I(\text{data}, \alpha)$ , such that  $P(\theta^* \in I(\text{data}, \alpha)) = \alpha$

- ▶ Bayesian

# Starting point: the intractable likelihood problem

- ▶ Two teams

- ▶ Frequentist

- ▶ Assume there exists some true  $\theta^*$  value

- ▶ Use likelihood (ratios) to compute **confidence intervals**  $I(\text{data}, \alpha)$ , such that  $P(\theta^* \in I(\text{data}, \alpha)) = \alpha$

- ▶ Bayesian

- ▶  $\theta$  are themselves random variables, with marginal distribution  $P(\theta)$  called the **prior**

# Starting point: the intractable likelihood problem

- ▶ Two teams

- ▶ Frequentist

- ▶ Assume there exists some true  $\theta^*$  value
    - ▶ Use likelihood (ratios) to compute **confidence intervals**  $I(\text{data}, \alpha)$ , such that  $P(\theta^* \in I(\text{data}, \alpha)) = \alpha$

- ▶ Bayesian

- ▶  $\theta$  are themselves random variables, with marginal distribution  $P(\theta)$  called the **prior**
    - ▶ Bayes' theorem uses data to update the distribution,  $P(\theta|\text{data}) \propto \mathcal{L}(\text{data}|\theta) \times P(\theta)$ ,\* ie the **posterior**

\* Btw this is only  $P(A,B)=P(A|B)P(B)=P(B|A)P(A)$  applied to the joint probability space of (params, data) — nothing deep, really.

# Starting point: the intractable likelihood problem

- ▶ **Two teams**

- ▶ **Frequentist**

- ▶ Assume there exists some true  $\theta^*$  value

- ▶ Use likelihood (ratios) to compute **confidence intervals**  $I(\text{data}, \alpha)$ , such that  $P(\theta^* \in I(\text{data}, \alpha)) = \alpha$

- ▶ **Bayesian**

- ▶  $\theta$  are themselves random variables, with marginal distribution  $P(\theta)$  called the **prior**

- ▶ Bayes' theorem uses data to update the distribution,  $P(\theta|\text{data}) \propto \mathcal{L}(\text{data}|\theta) \times P(\theta)$ ,\* ie the **posterior**

- ▶ **Same key ingredient: likelihood  $\mathcal{L}(\text{data}|\theta)$**

\* Btw this is only  $P(A,B)=P(A|B)P(B)=P(B|A)P(A)$  applied to the joint probability space of (params, data) — nothing deep, really.

# Starting point: the intractable likelihood problem

- ▶ **Two teams**

- ▶ **Frequentist**

- ▶ Assume there exists some true  $\theta^*$  value

- ▶ Use likelihood (ratios) to compute **confidence intervals**  $I(\text{data}, \alpha)$ , such that  $P(\theta^* \in I(\text{data}, \alpha)) = \alpha$

- ▶ **Bayesian**

- ▶  $\theta$  are themselves random variables, with marginal distribution  $P(\theta)$  called the **prior**

- ▶ Bayes' theorem uses data to update the distribution,  $P(\theta|\text{data}) \propto \mathcal{L}(\text{data}|\theta) \times P(\theta)$ ,\* ie the **posterior**

- ▶ **Same key ingredient: likelihood  $\mathcal{L}(\text{data}|\theta)$**

**What if the likelihood is unknown or intractable? 🤯**

\* Btw this is only  $P(A,B)=P(A|B)P(B)=P(B|A)P(A)$  applied to the joint probability space of (params, data) — nothing deep, really.

# Why would the likelihood be “intractable”?

# Why would the likelihood be “intractable”?

- ▶ Likelihood of observables
  - ▶ Elementary stochastic processes described *via analytic* theory (QFT, GR), *but...*
  - ▶ Final observables involves *many such processes* (potentially non-linear) with *instrumental signatures*, observational effects, selection functions, pre-processing, etc.
  - ▶ So **it becomes impossible to write an analytical likelihood function** in closed form 🌀

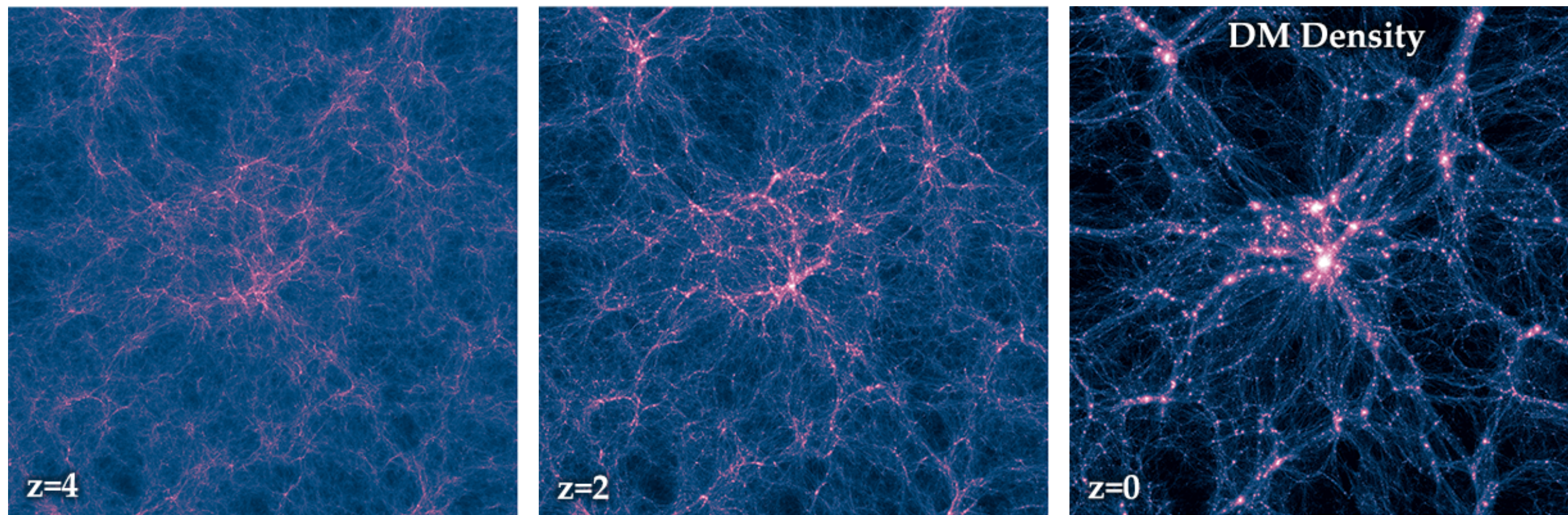
# Why would the likelihood be “intractable”?

## ▶ Likelihood of observables

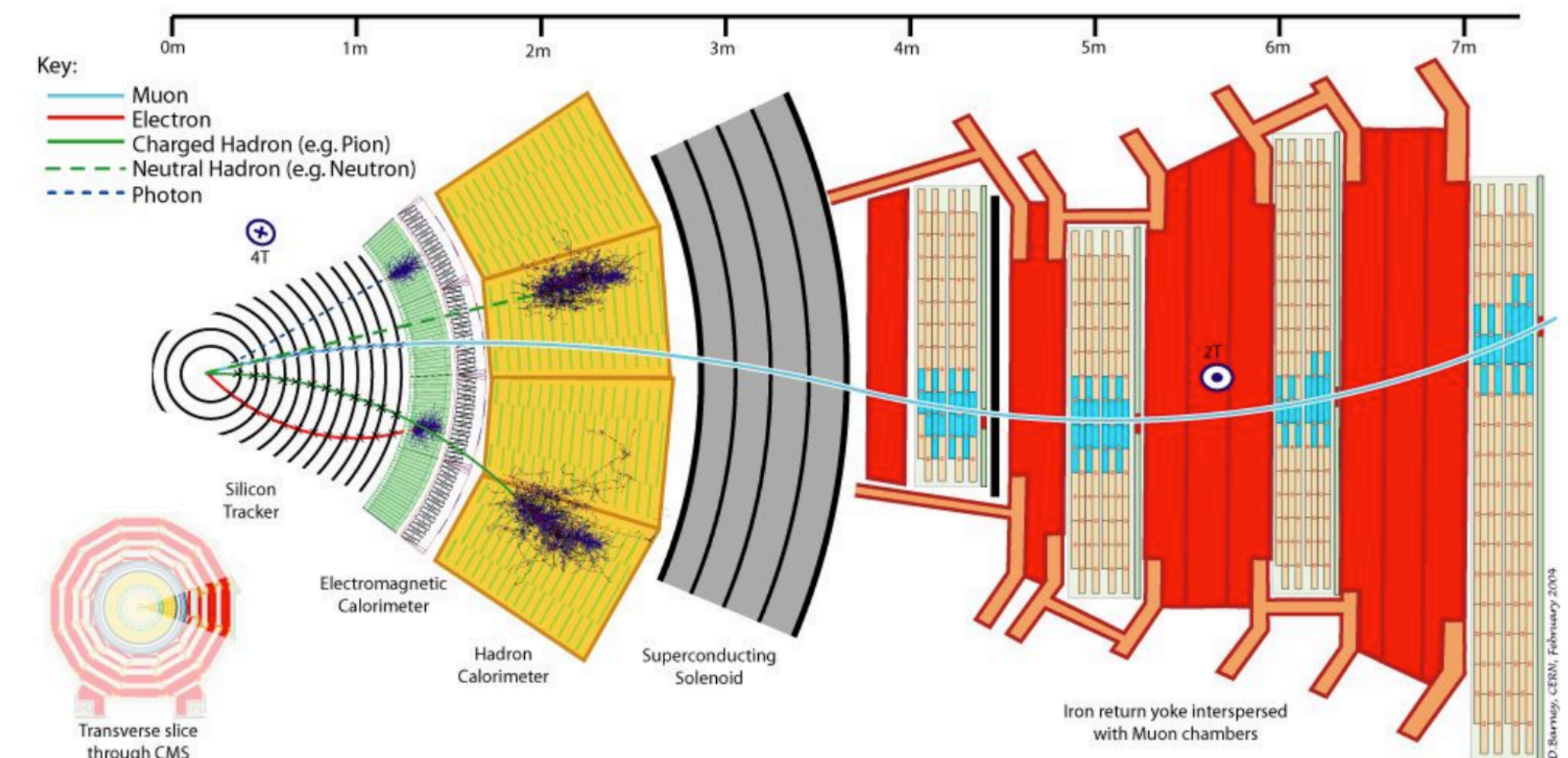
- ▶ Elementary stochastic processes described *via analytic* theory (QFT, GR), *but...*
- ▶ Final observables involves *many such processes* (potentially non-linear) with *instrumental signatures*, observational effects, selection functions, pre-processing, etc.
- ▶ So *it becomes impossible to write an analytical likelihood function* in closed form 🌀

## ▶ Examples

Cosmological structure evolution (Illustris simulation)

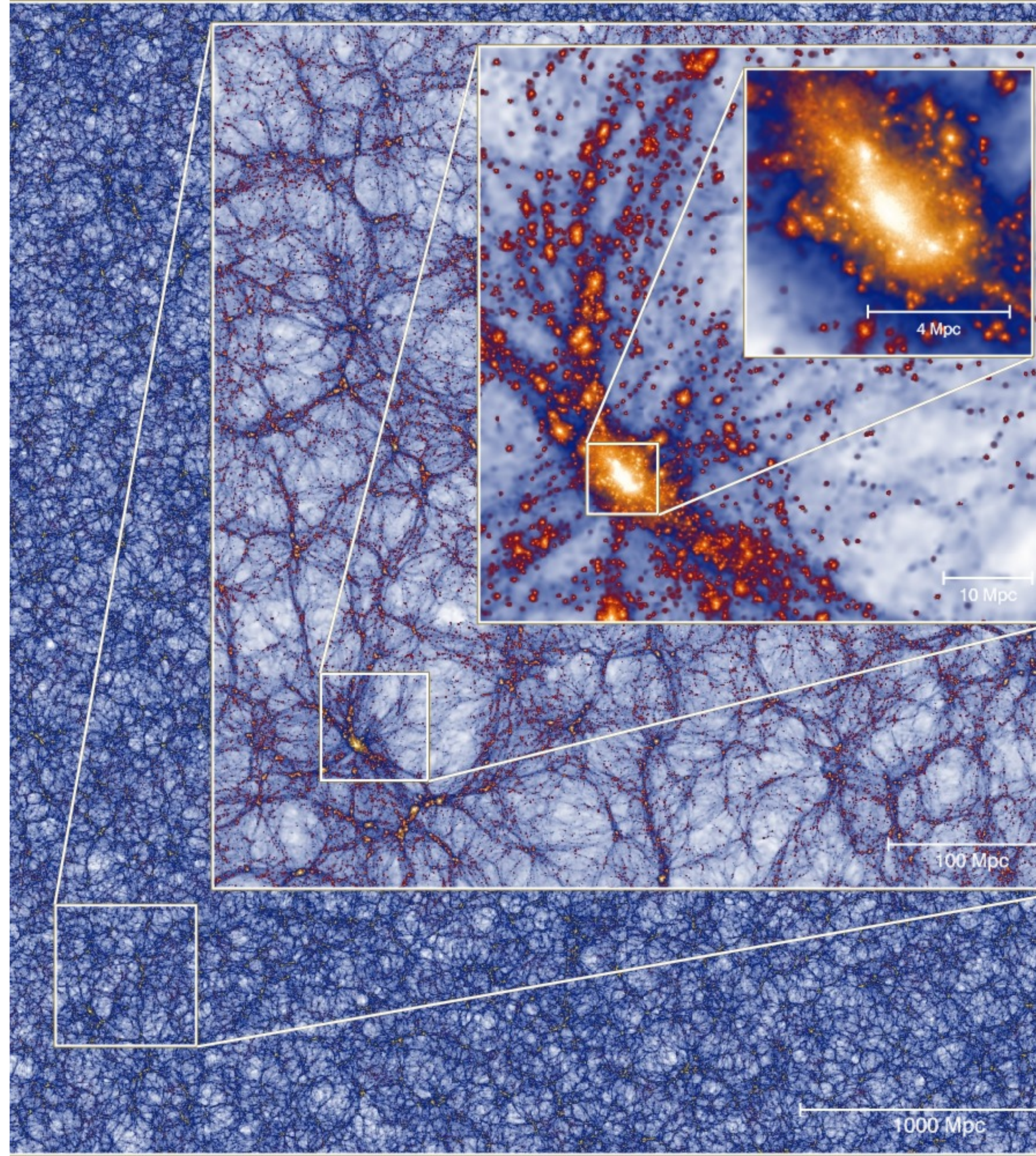


Event decay through CMS detector at CERN



# Simulations

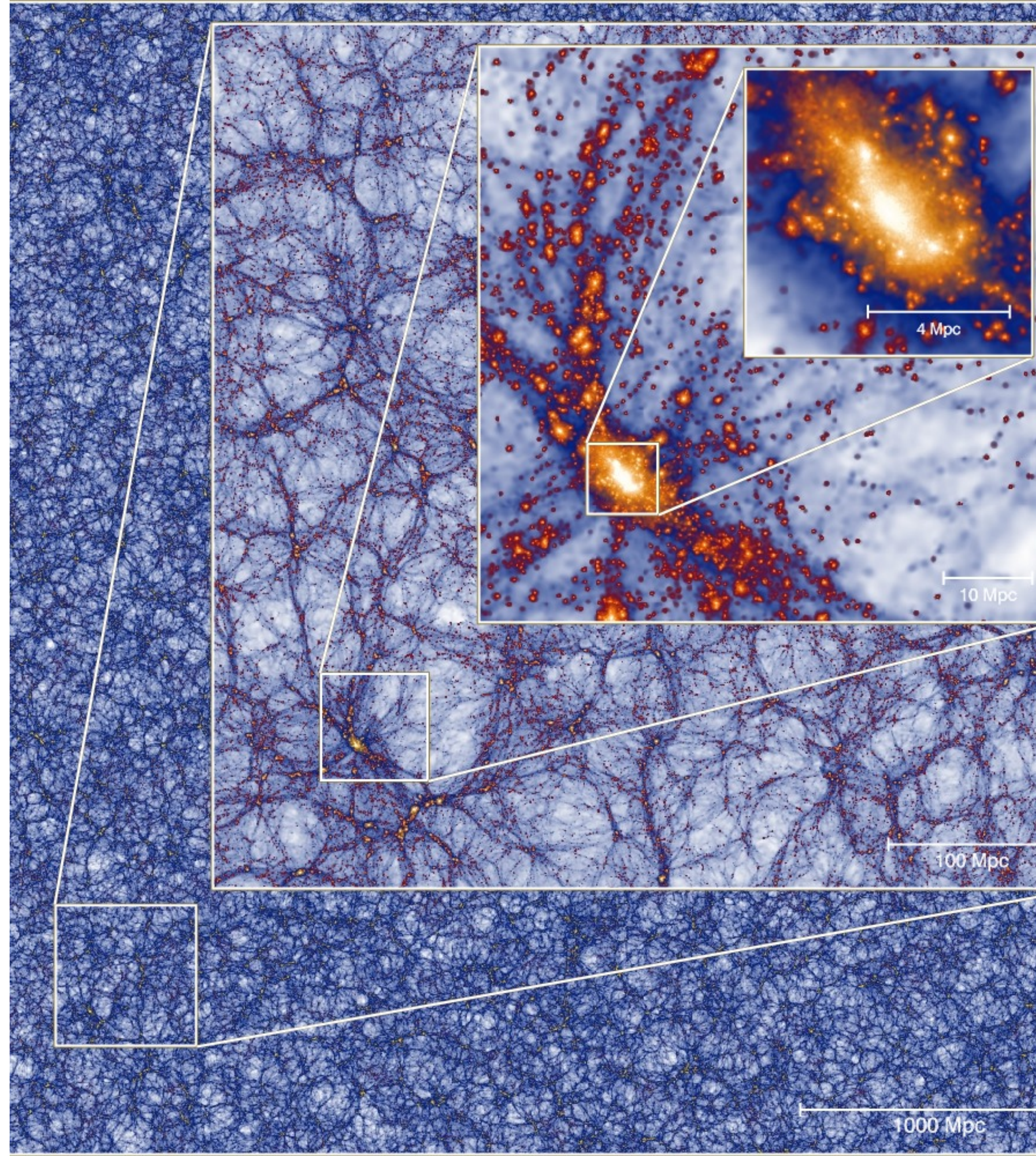
We can *generate* data! 😎



# Simulations

We can *generate* data! 😎

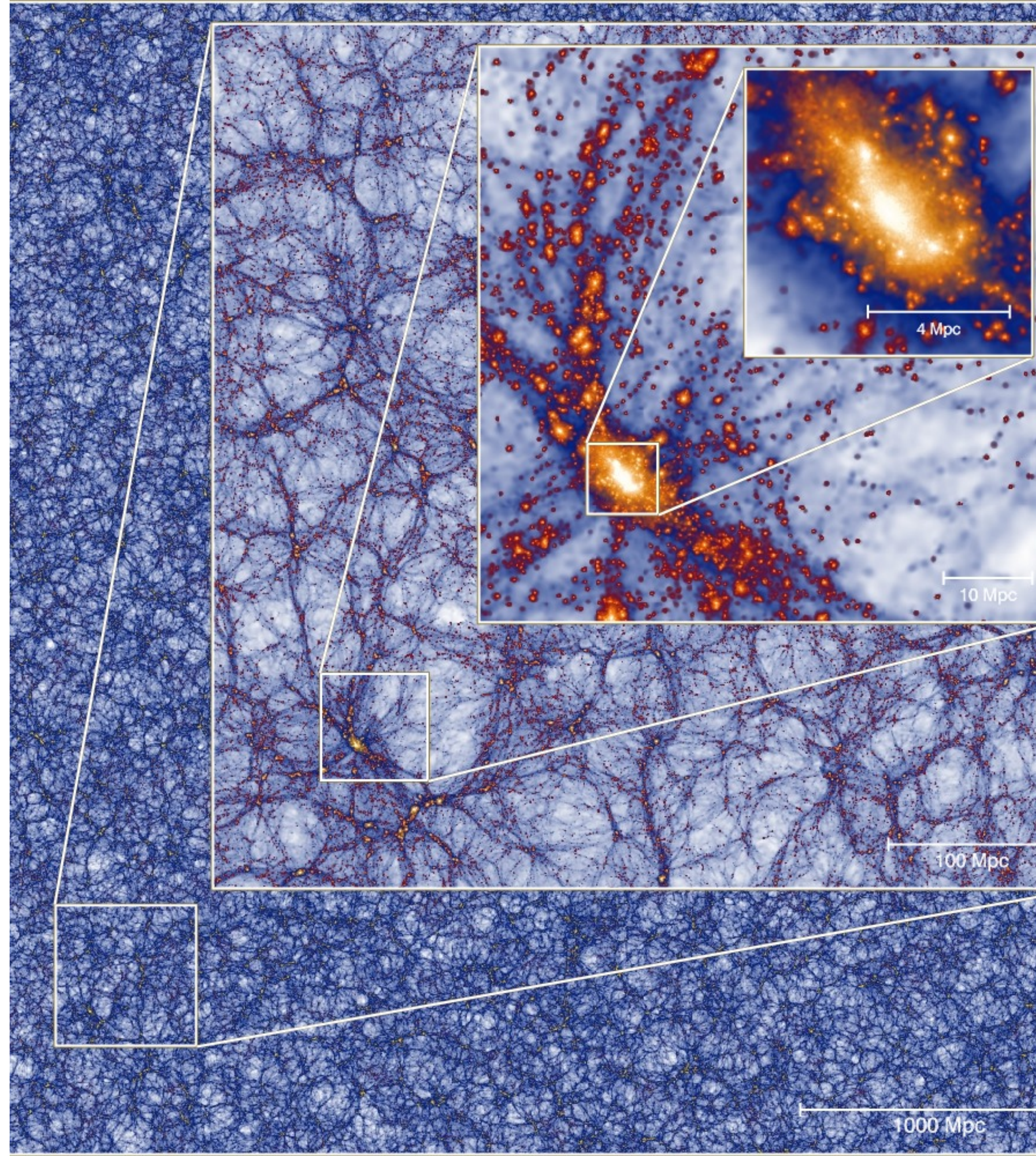
- ▶ **Why** we generate sims
  1. Theoretical predictions
  2. Systematics calibration
  3. End-to-end validation



# Simulations

We can *generate* data! 😎

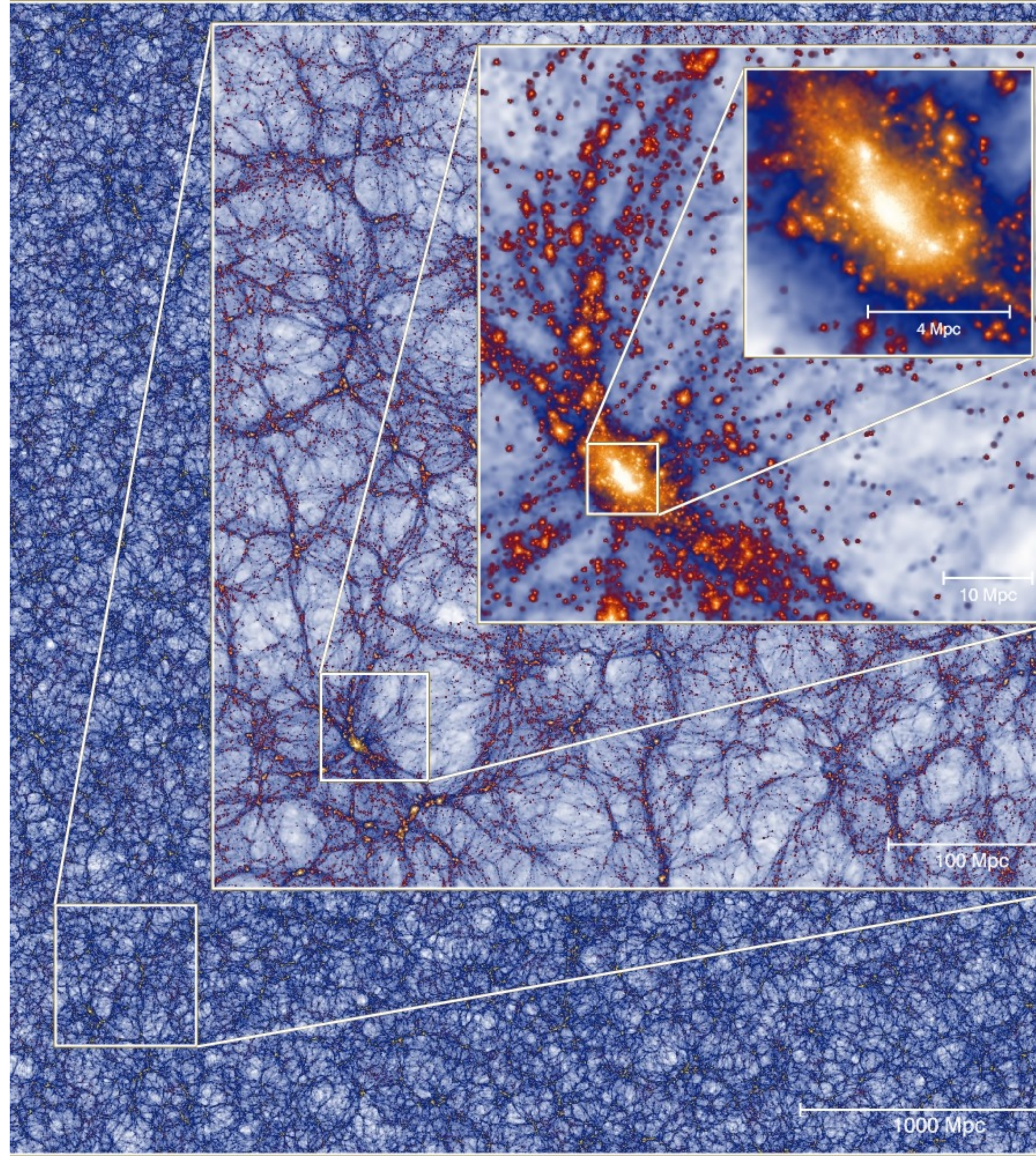
- ▶ **Why** we generate sims
  1. Theoretical predictions
  2. Systematics calibration
  3. End-to-end validation
- ▶ **Simulations:** *bottom-up* approach from elementary processes to complex observables



# Simulations

We can *generate* data! 😎

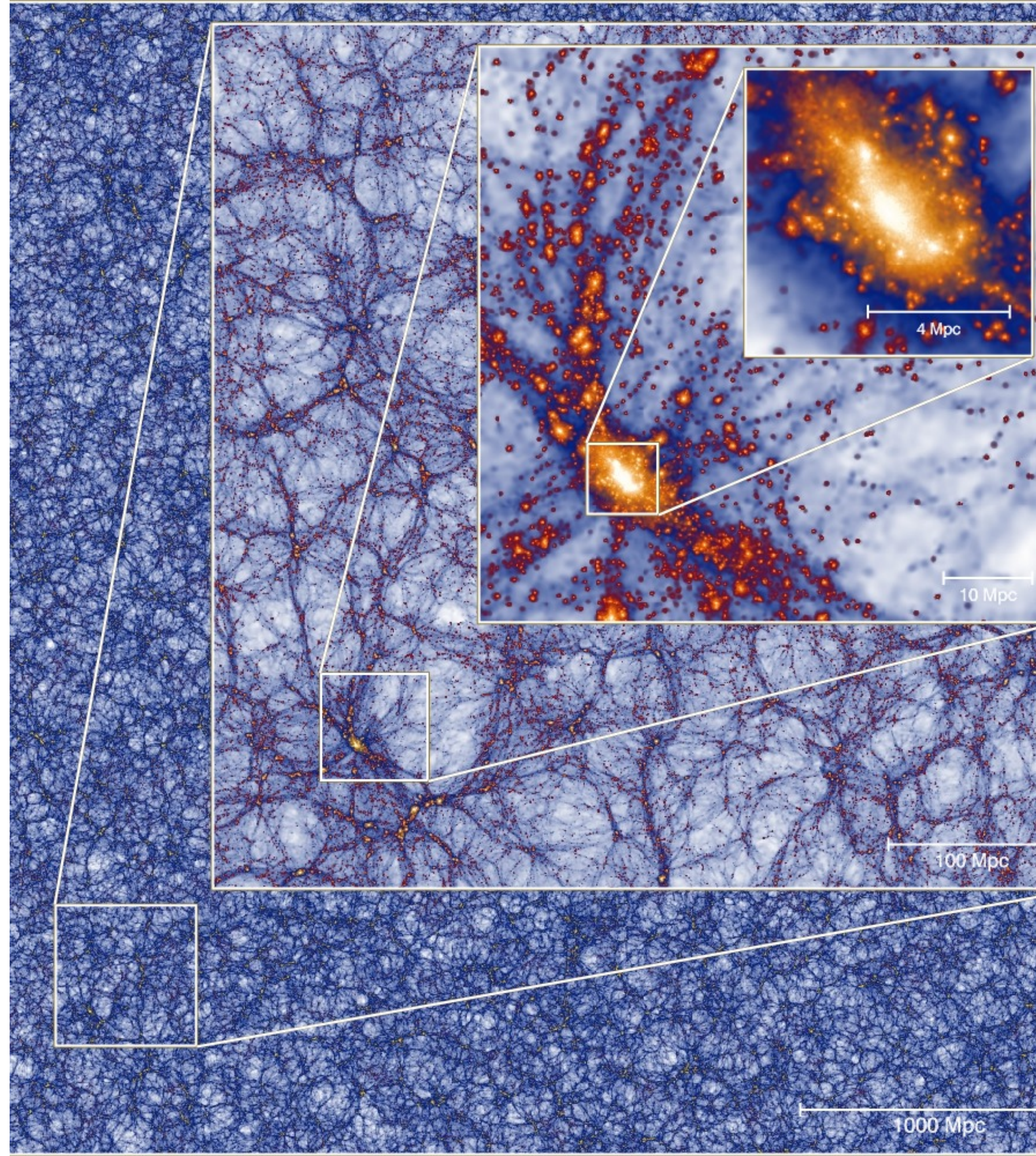
- ▶ **Why** we generate sims
  1. Theoretical predictions
  2. Systematics calibration
  3. End-to-end validation
- ▶ **Simulations:** *bottom-up* approach from elementary processes to complex observables
- ▶ **Observational/instrumental signature/selection functions** on top of fundamental physics 🙌



# Simulations

We can *generate* data! 😎

- ▶ **Why** we generate sims
  1. Theoretical predictions
  2. Systematics calibration
  3. End-to-end validation
- ▶ **Simulations:** *bottom-up* approach from elementary processes to complex observables
- ▶ **Observational/instrumental signature/selection functions** on top of fundamental physics 🙌
- ▶ **Accuracy/resolution/amount** increasing with computing resources/techniques (for now)



# Simulation-based inference

So what are we talking about?

# Simulation-based inference

So what are we talking about?

- ▶ **Definitions and vocabulary**

- ▶ SBI = methods to infer the parameters  $\theta$  of a statistical model described by a likelihood,  $\mathcal{L}(\text{data}|\theta)$ , that may only be *sampled* through simulations, *but not evaluated*
- ▶ SBI = *likelihood-free* inference = *implicit likelihood* inference

# Simulation-based inference

So what are we talking about?

- ▶ **Definitions and vocabulary**

- ▶ SBI = methods to infer the parameters  $\theta$  of a statistical model described by a likelihood,  $\mathcal{L}(\text{data}|\theta)$ , that may only be *sampled* through simulations, *but not evaluated*
- ▶ SBI = *likelihood-free* inference = *implicit likelihood* inference

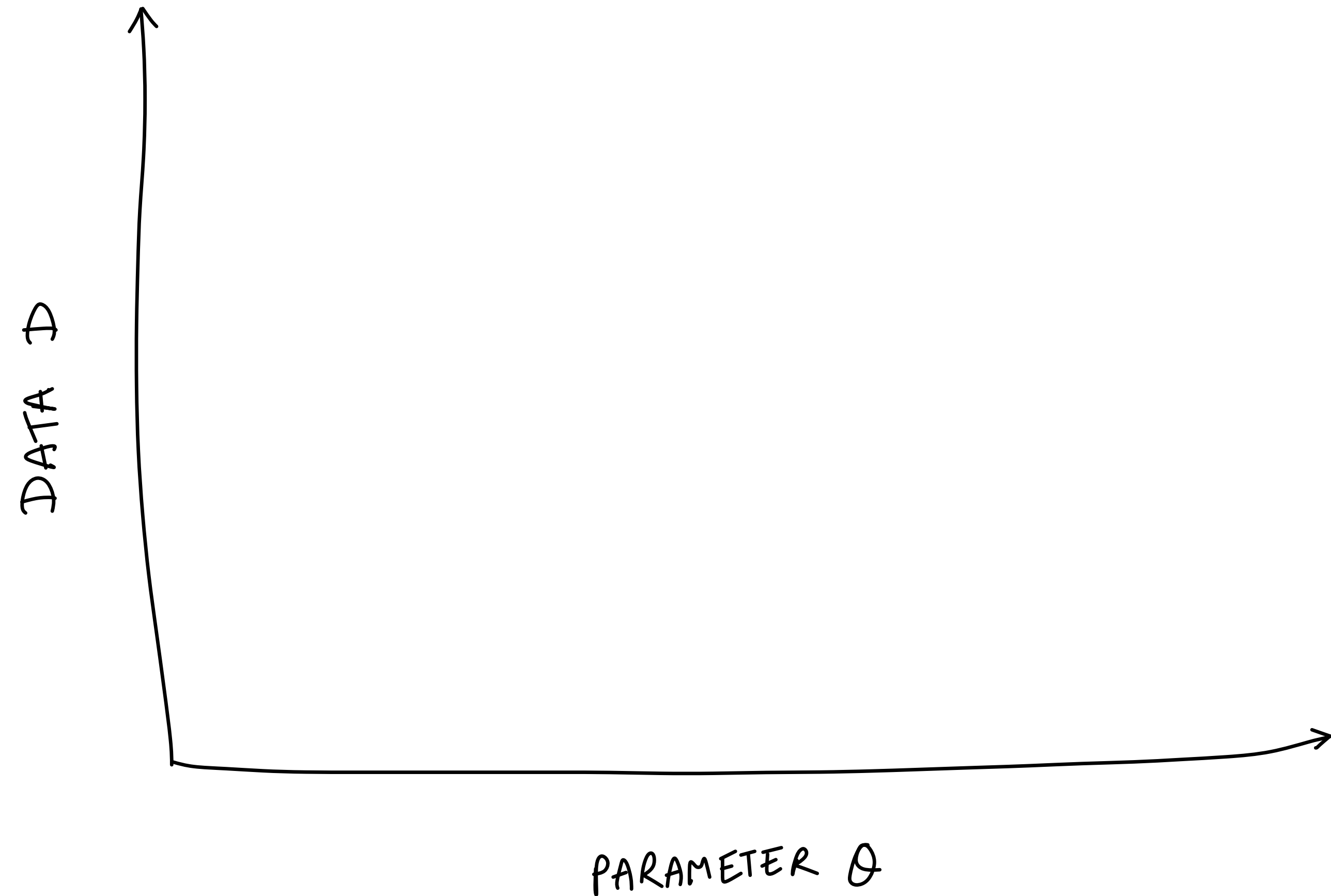
- ▶ **How can SBI accomplish such a feat?** 🧙

- ▶ **Observed data** compared to **simulated data** *generated* at various parameter values
- ▶ Mathematically sound approach with deep generative models

# Basic idea

First, without the maths

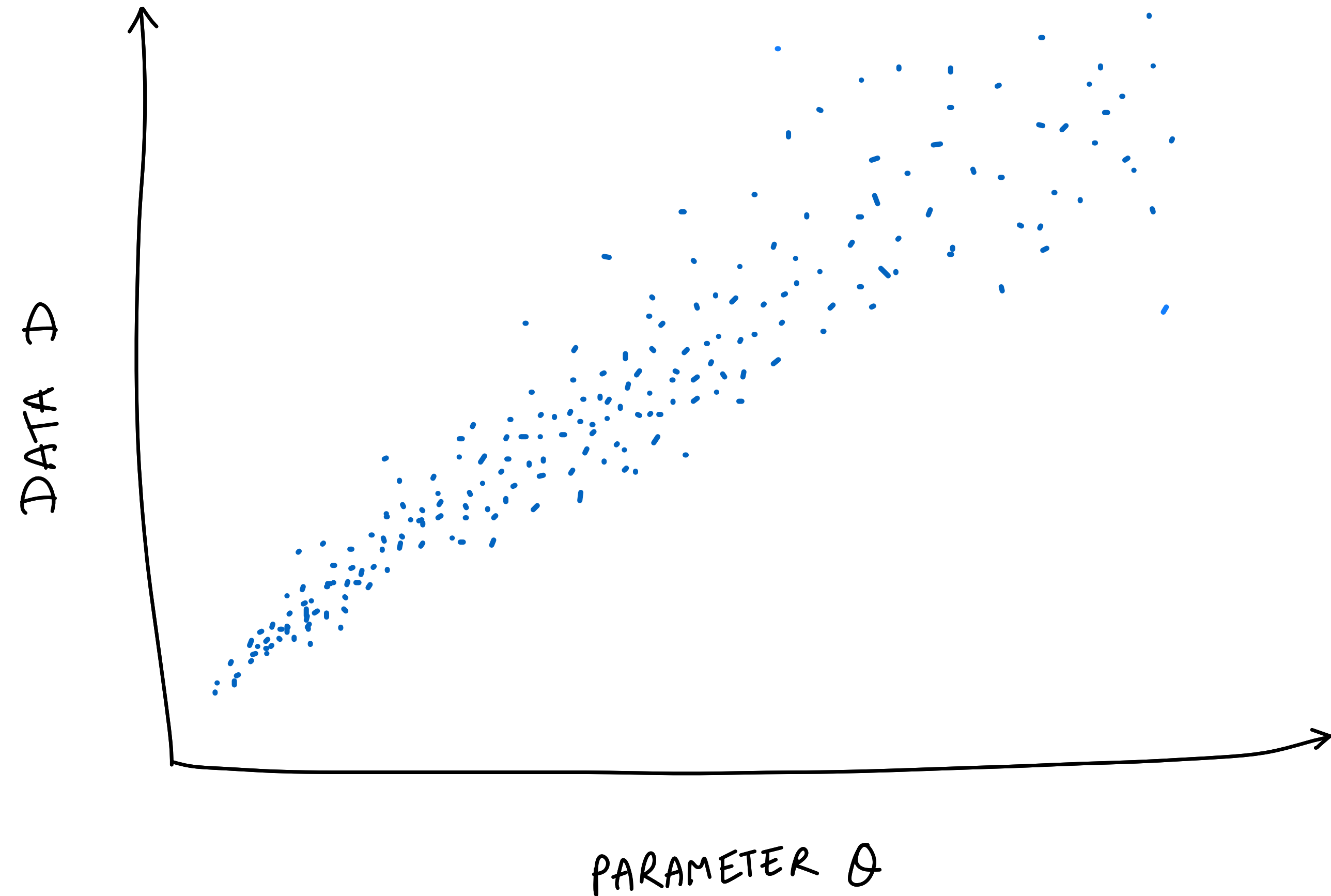
- Assume
  - We have a simulator to generate data  $D$  at any model parameter value  $\theta$
- Recipe



# Basic idea

First, without the maths

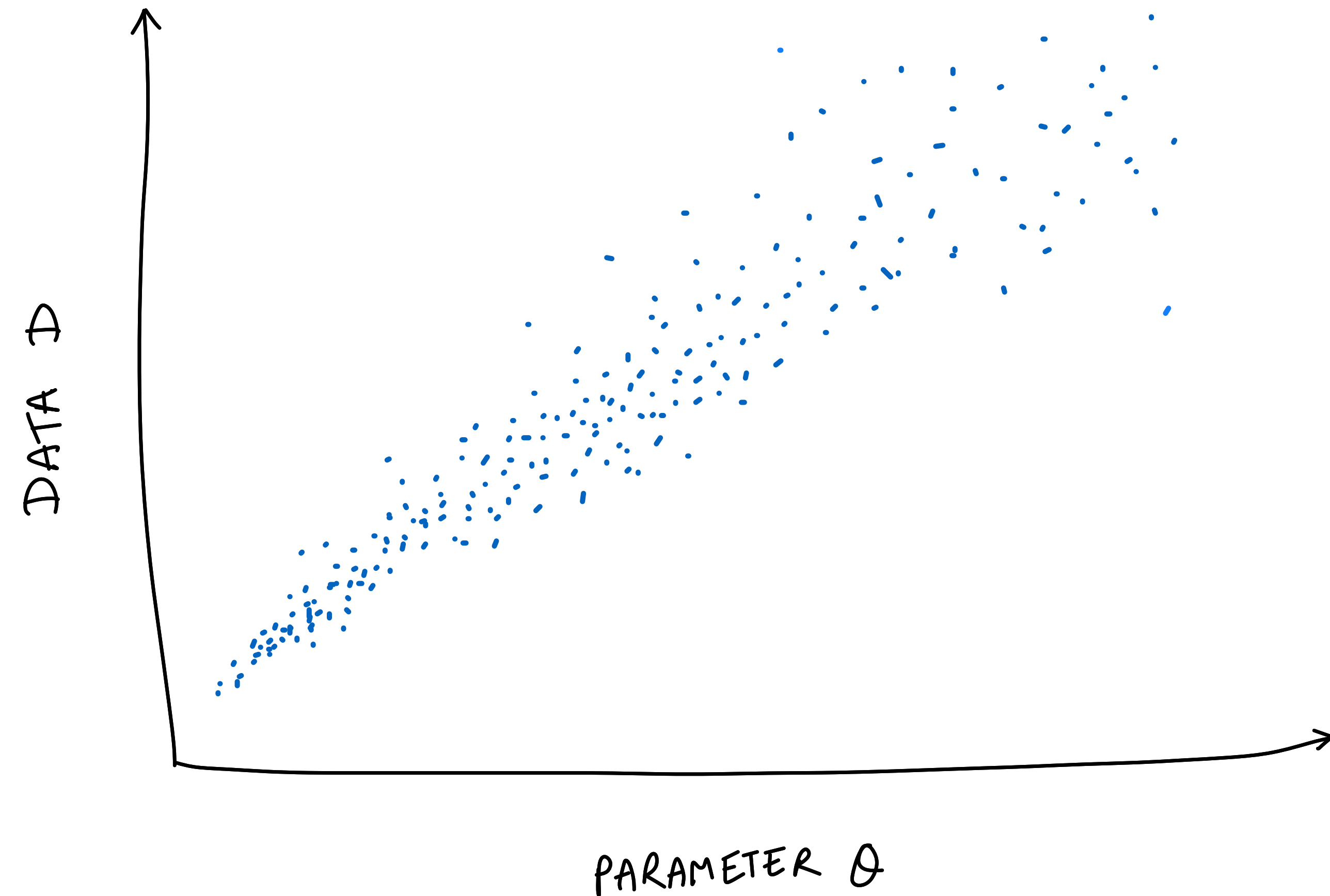
- Assume
  - We have a simulator to generate data  $D$  at any model parameter value  $\theta$
- Recipe
  1. **Run simulator** at various parameter  $\theta$



# Basic idea

First, without the maths

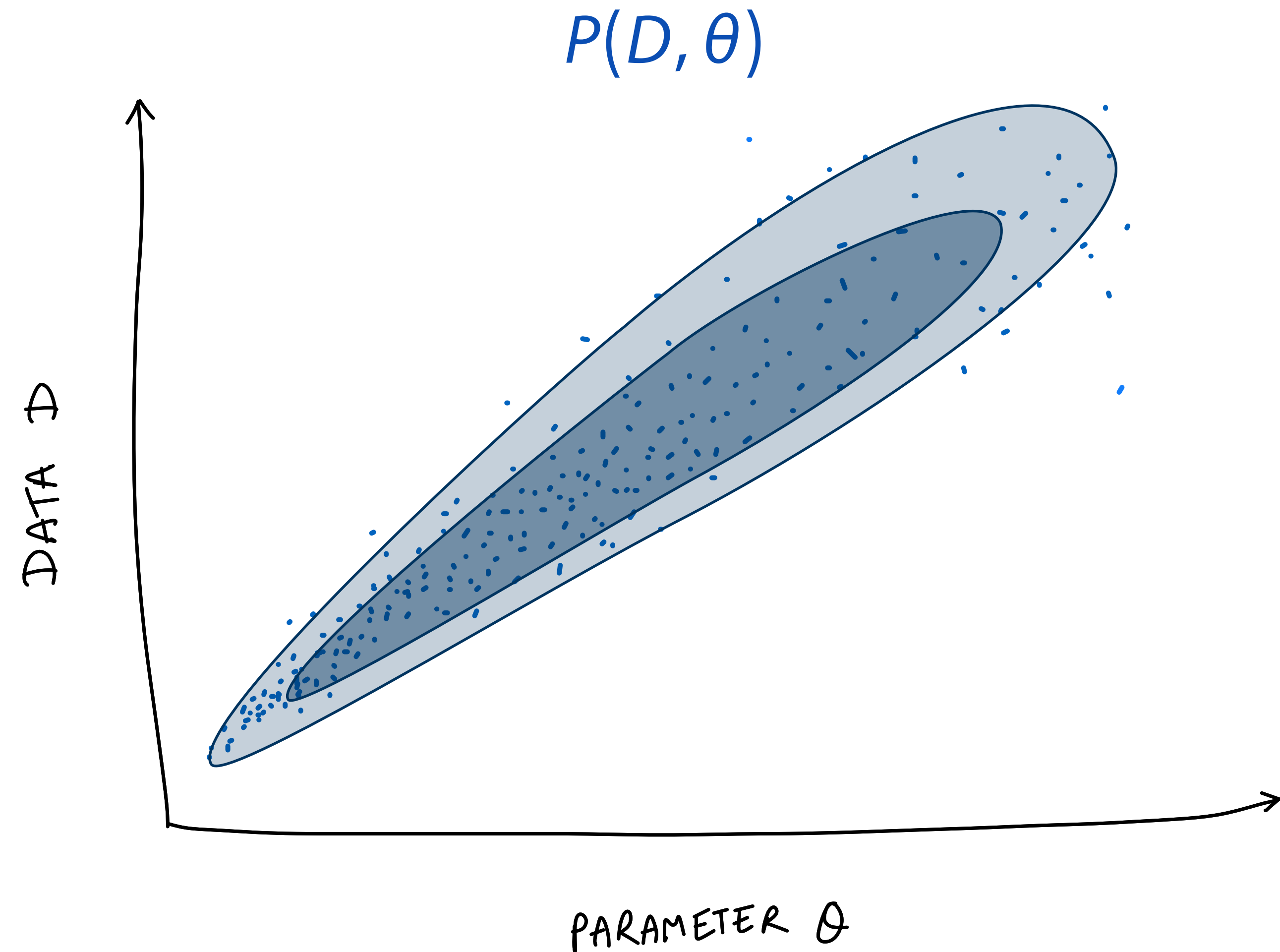
- Assume
  - We have a simulator to generate data  $D$  at any model parameter value  $\theta$
- Recipe
  1. **Run simulator** at various parameter  $\theta$
  2. *Learn* one of:



# Basic idea

First, without the maths

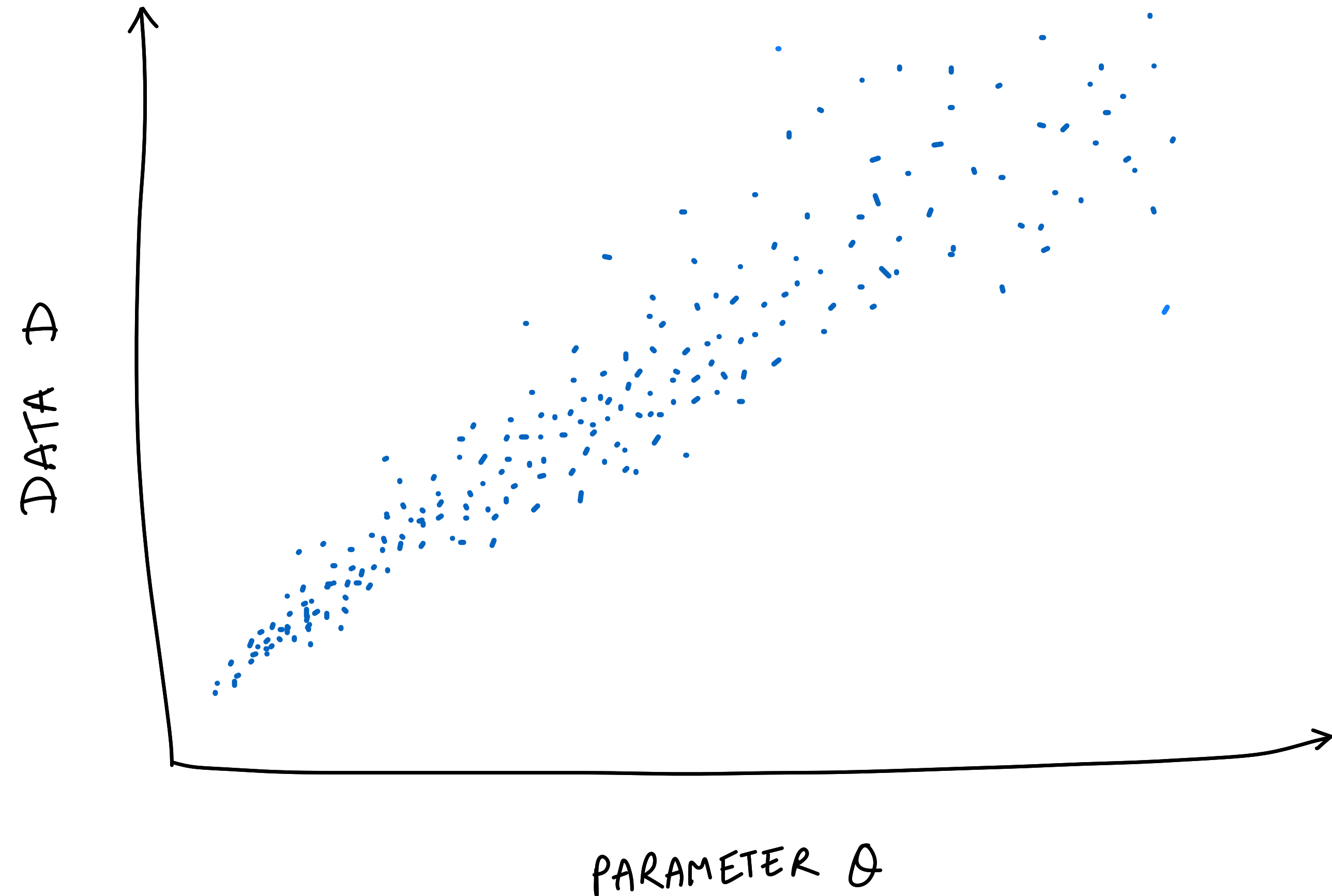
- ▶ Assume
  - ▶ We have a simulator to generate data  $D$  at any model parameter value  $\theta$
- ▶ Recipe
  1. **Run simulator** at various parameter  $\theta$
  2. *Learn* one of:
    1. Joint distribution  $P(D, \theta)$



# Basic idea

First, without the maths

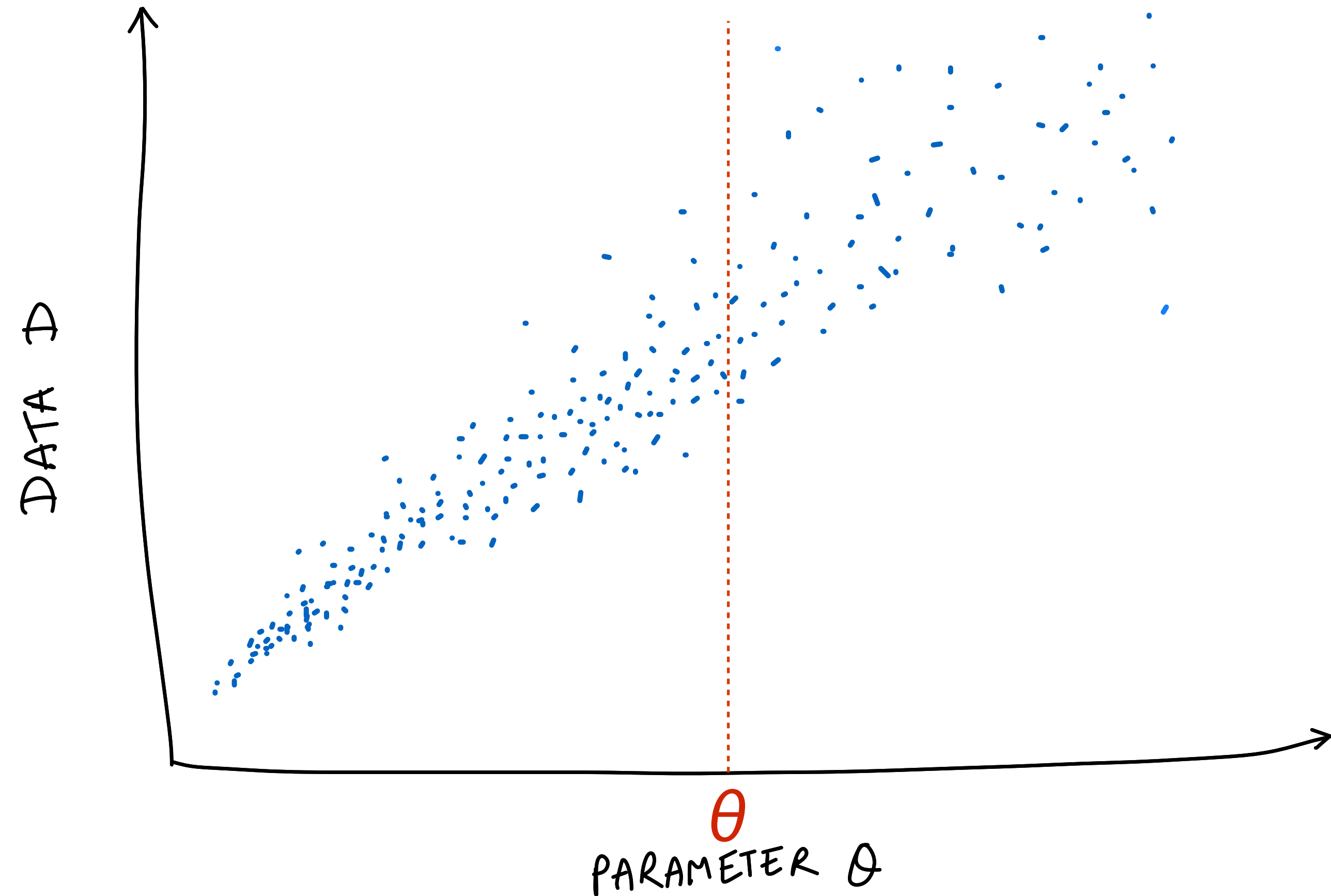
- Assume
  - We have a simulator to generate data  $D$  at any model parameter value  $\theta$
- Recipe
  1. Run simulator at various parameter  $\theta$
  2. Learn one of:
    1. Joint distribution  $P(D, \theta)$
    2. Likelihood  $\mathcal{L}(D|\theta)$



# Basic idea

First, without the maths

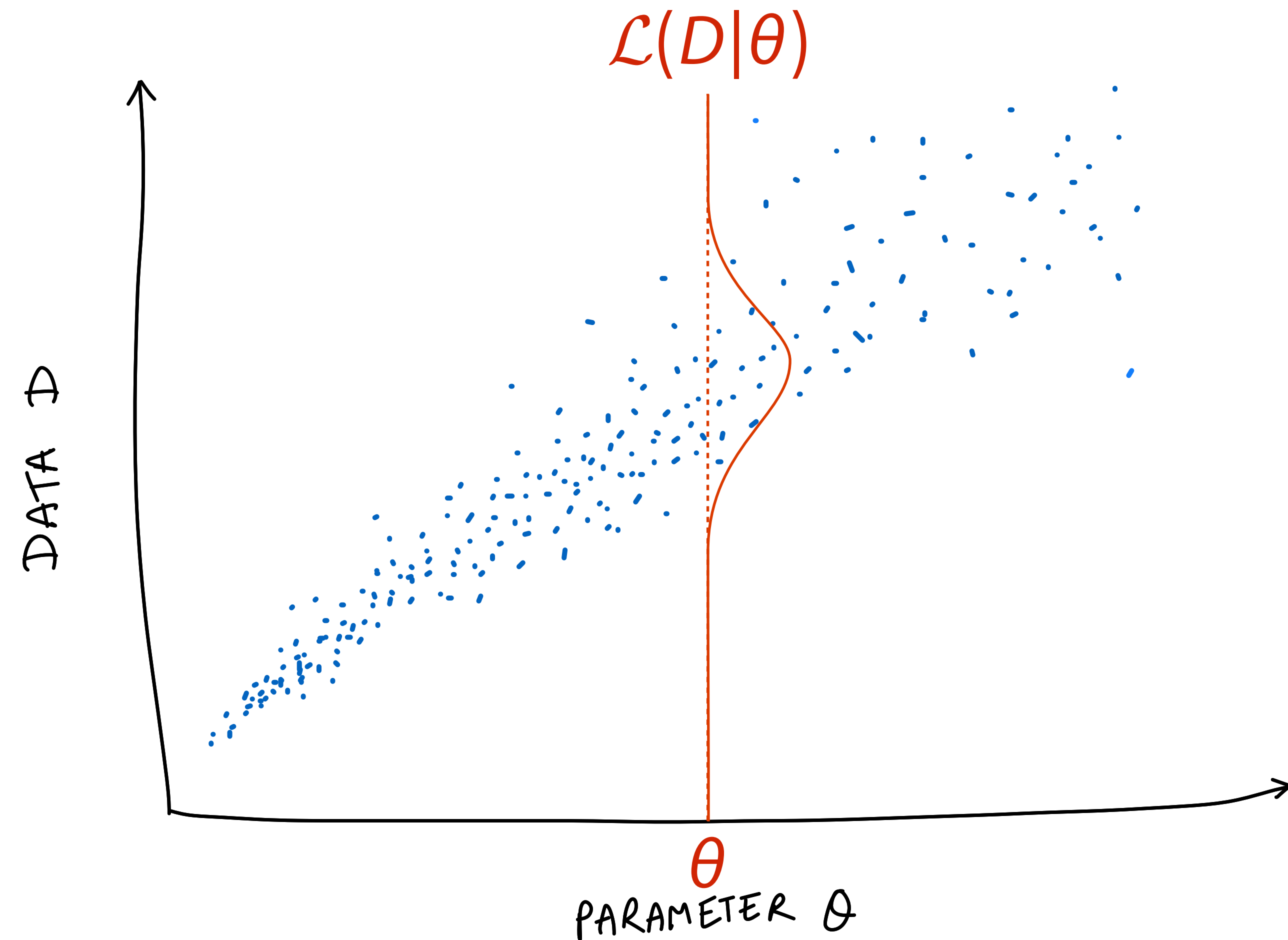
- Assume
  - We have a simulator to generate data  $D$  at any model parameter value  $\theta$
- Recipe
  1. Run simulator at various parameter  $\theta$
  2. Learn one of:
    1. Joint distribution  $P(D, \theta)$
    2. Likelihood  $\mathcal{L}(D|\theta)$



# Basic idea

First, without the maths

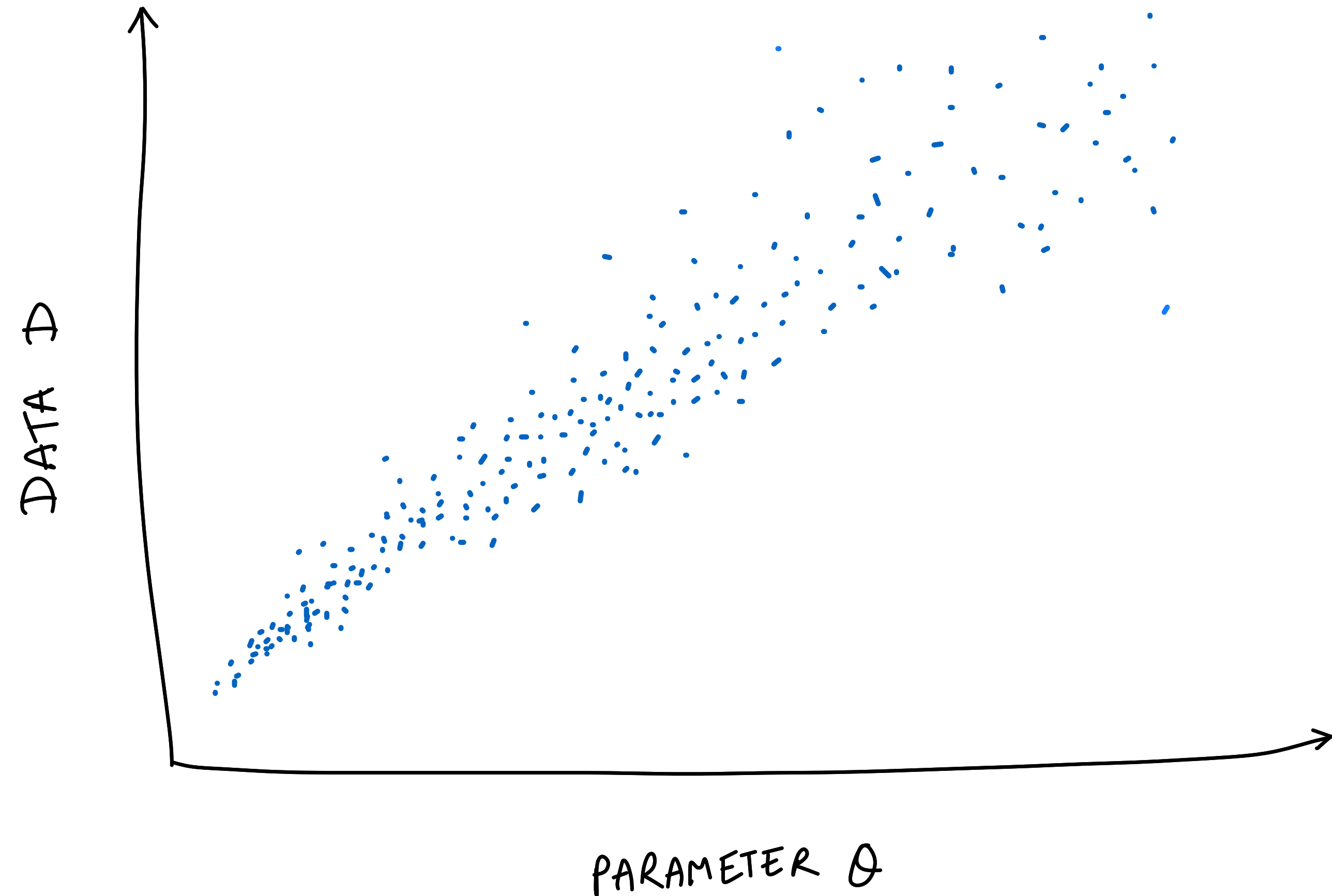
- Assume
  - We have a simulator to generate data  $D$  at any model parameter value  $\theta$
- Recipe
  1. Run simulator at various parameter  $\theta$
  2. Learn one of:
    1. Joint distribution  $P(D, \theta)$
    2. Likelihood  $\mathcal{L}(D|\theta)$



# Basic idea

First, without the maths

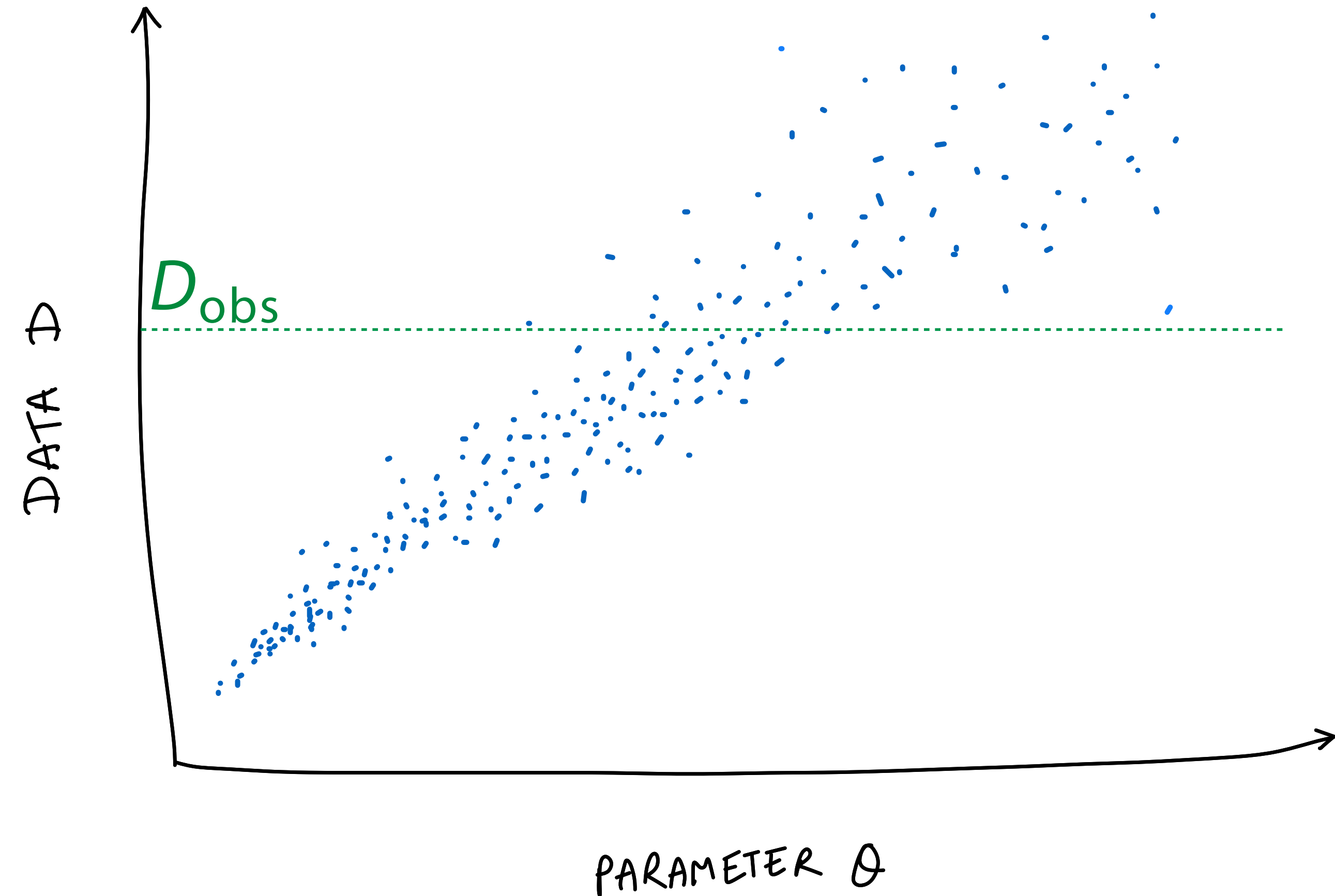
- Assume
  - We have a simulator to generate data  $D$  at any model parameter value  $\theta$
- Recipe
  1. Run simulator at various parameter  $\theta$
  2. Learn one of:
    1. Joint distribution  $P(D, \theta)$
    2. Likelihood  $\mathcal{L}(D|\theta)$
    3. Posterior  $P(D|\theta)$



# Basic idea

First, without the maths

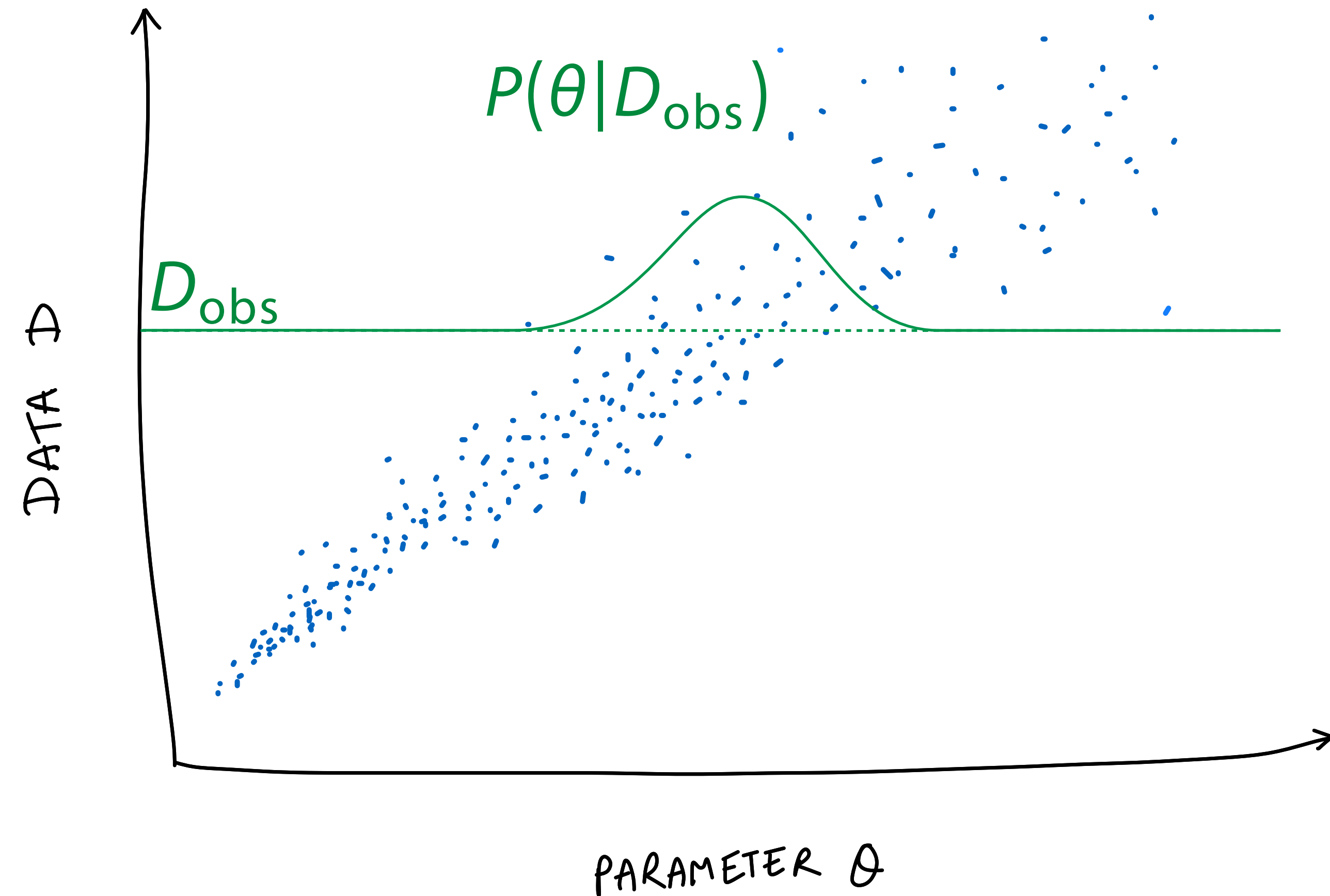
- ▶ Assume
  - ▶ We have a simulator to generate data  $D$  at any model parameter value  $\theta$
- ▶ Recipe
  1. Run simulator at various parameter  $\theta$
  2. Learn one of:
    1. Joint distribution  $P(D, \theta)$
    2. Likelihood  $\mathcal{L}(D|\theta)$
    3. Posterior  $P(D|\theta)$



# Basic idea

First, without the maths

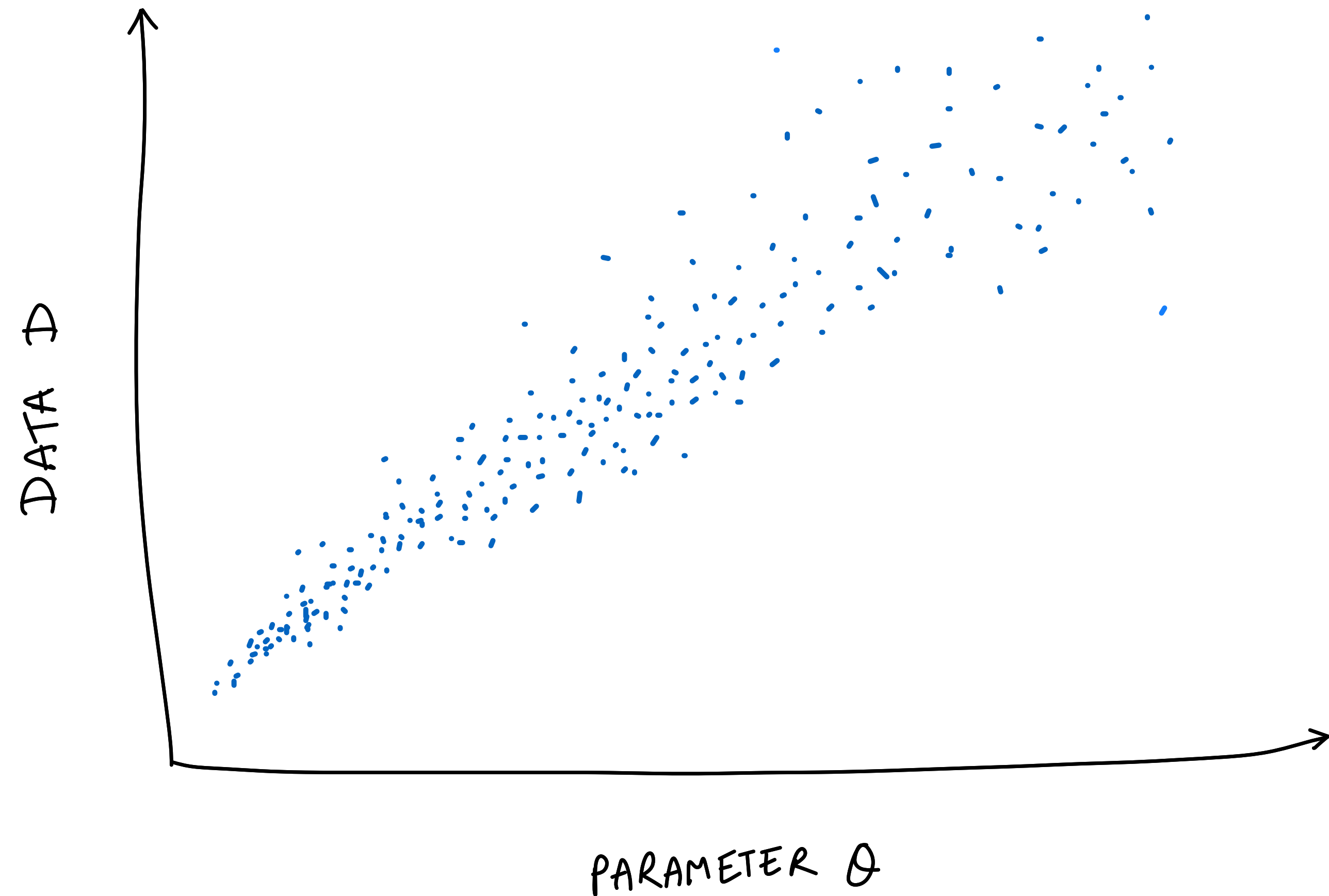
- ▶ Assume
  - ▶ We have a simulator to generate data  $D$  at any model parameter value  $\theta$
- ▶ Recipe
  1. **Run simulator** at various parameter  $\theta$
  2. *Learn* one of:
    1. Joint distribution  $P(D, \theta)$
    2. Likelihood  $\mathcal{L}(D|\theta)$
    3. Posterior  $P(D|\theta)$



# Basic idea

First, without the maths

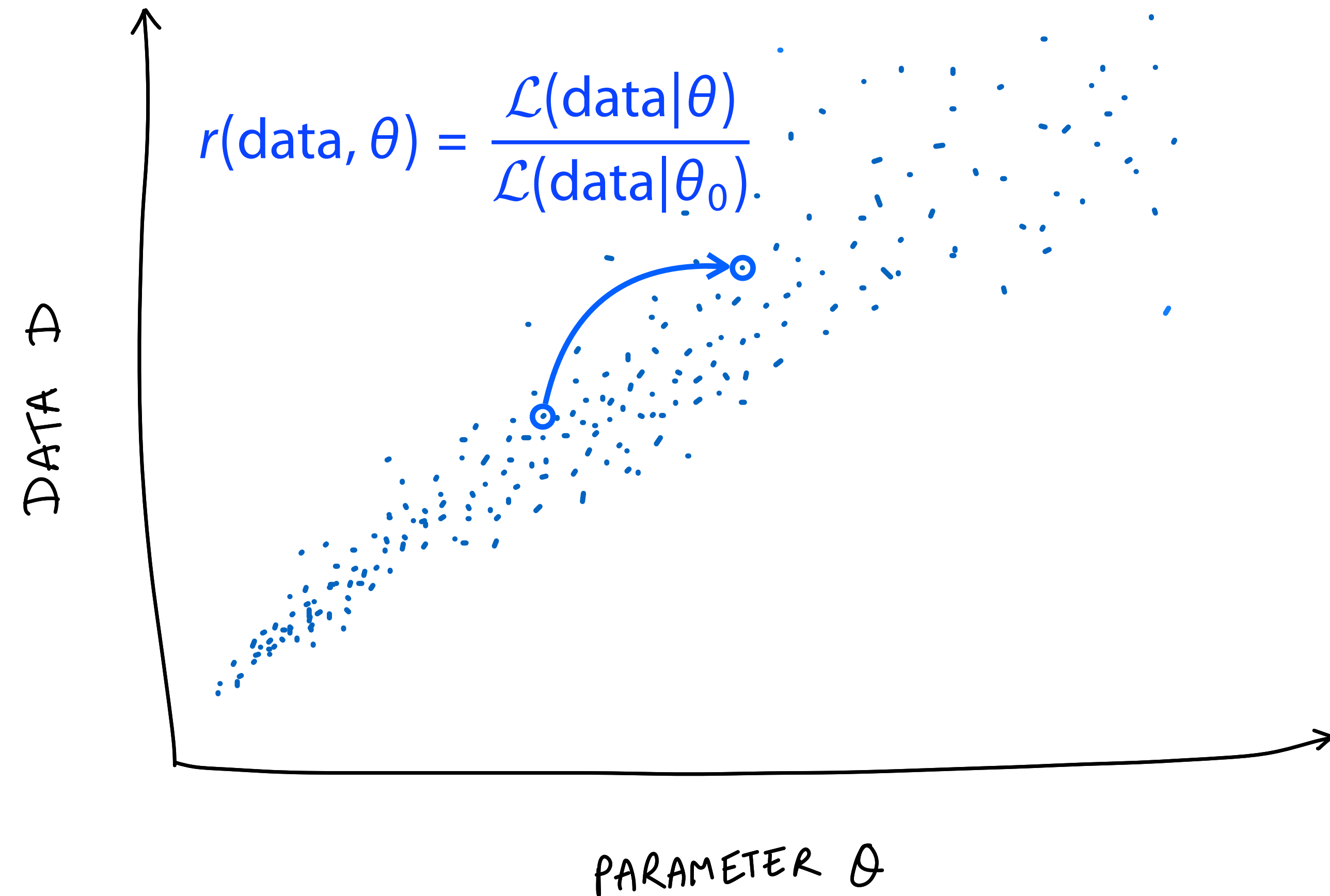
- Assume
  - We have a simulator to generate data  $D$  at any model parameter value  $\theta$
- Recipe
  1. Run simulator at various parameter  $\theta$
  2. Learn one of:
    1. Joint distribution  $P(D, \theta)$
    2. Likelihood  $\mathcal{L}(D|\theta)$
    3. Posterior  $P(D|\theta)$
    4. Likelihood ratios  $r(D, \theta)$



# Basic idea

First, without the maths

- ▶ Assume
  - ▶ We have a simulator to generate data  $D$  at any model parameter value  $\theta$
- ▶ Recipe
  1. Run simulator at various parameter  $\theta$
  2. Learn one of:
    1. Joint distribution  $P(D, \theta)$
    2. Likelihood  $\mathcal{L}(D|\theta)$
    3. Posterior  $P(D|\theta)$
    4. Likelihood ratios  $r(D, \theta)$

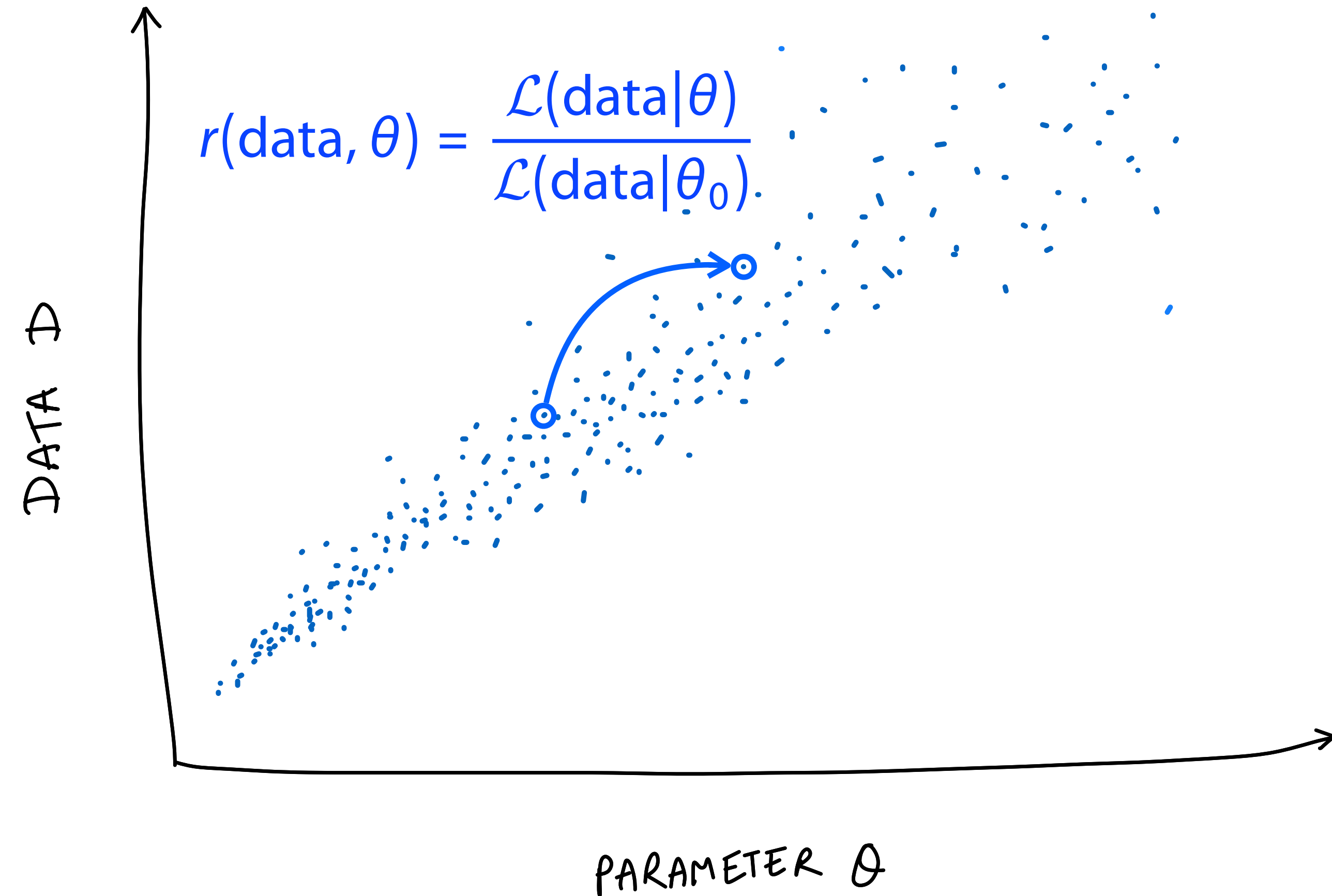


# Basic idea

First, without the maths

- Assume
  - We have a simulator to generate data  $D$  at any model parameter value  $\theta$
- Recipe
  1. Run simulator at various parameter  $\theta$
  2. Learn one of:
    1. Joint distribution  $P(D, \theta)$
    2. Likelihood  $\mathcal{L}(D|\theta)$
    3. Posterior  $P(D|\theta)$
    4. Likelihood ratios  $r(D, \theta)$

That's it!\*



# Pros and cons

SBI vs likelihood-based approach

# Pros and cons

SBI vs likelihood-based approach

- ▶ **Simulations vs analytical modelling trade-off**
  - ▶ Analytic modelling involves theoretical **approximations**
  - ▶ Simulations model elementary processes, but have their own issues: **resolution, convergence**, etc.

# Pros and cons

## SBI vs likelihood-based approach

- ▶ **Simulations vs analytical modelling trade-off**
  - ▶ Analytic modelling involves theoretical **approximations**
  - ▶ Simulations model elementary processes, but have their own issues: **resolution, convergence**, etc.
- ▶ **SBI does not *approximate* the likelihood**
  - ▶ No need to **assume functional form** (e.g., Gaussian or Poisson)
  - ▶ Bonus: no need to evaluate costly covariance matrices!

# Pros and cons

## SBI vs likelihood-based approach

- ▶ **Simulations vs analytical modelling trade-off**
  - ▶ Analytic modelling involves theoretical **approximations**
  - ▶ Simulations model elementary processes, but have their own issues: **resolution, convergence**, etc.
- ▶ **SBI does not *approximate* the likelihood**
  - ▶ No need to **assume functional form** (e.g., Gaussian or Poisson)
  - ▶ Bonus: no need to evaluate costly covariance matrices!
- ▶ **New issues** 😊
  - ▶ **How many** simulations? **Where** in parameter space?
  - ▶ **Learning distributions?** Kernel Density Estimation (KDE) only work in very low dimension... 😞

# SBI starter pack

Simulators, emulators and ABC



**Basic idea, now with the maths** 🤔

# Basic idea, now with the maths 🤔

- ▶ Notations

# Basic idea, now with the maths 🤔

- ▶ Notations

- ▶ Model parameters  $\theta$ : physical and nuisance parameters

# Basic idea, now with the maths 🤔

## ▸ Notations

- Model parameters  $\theta$ : physical and nuisance parameters
- Latent variables  $z$ : internal, unobservable state of the system (see examples below)

# Basic idea, now with the maths 🤪

## ▸ Notations

- Model parameters  $\theta$ : physical and nuisance parameters
- Latent variables  $z$ : internal, unobservable state of the system (see examples below)
- Data  $x$ : simulated or observed

# Basic idea, now with the maths 🤔

## ▸ Notations

- Model parameters  $\theta$ : physical and nuisance parameters
- Latent variables  $z$ : internal, unobservable state of the system (see examples below)
- Data  $x$ : simulated or observed

## ▸ Simulator

# Basic idea, now with the maths 🤪

## ▸ Notations

- Model parameters  $\theta$ : physical and nuisance parameters
- Latent variables  $z$ : internal, unobservable state of the system (see examples below)
- Data  $x$ : simulated or observed

## ▸ Simulator

- Given params, the simulator samples  $z \sim P(z|\theta)$  and generates data  $x \sim P(x|\theta, z)$

# Basic idea, now with the maths 🤪

## ▶ Notations

- ▶ Model parameters  $\theta$ : physical and nuisance parameters
- ▶ Latent variables  $z$ : internal, unobservable state of the system (see examples below)
- ▶ Data  $x$ : simulated or observed

## ▶ Simulator

- ▶ Given params, the simulator samples  $z \sim P(z|\theta)$  and generates data  $x \sim P(x|\theta, z)$
- ▶ May involve both **stochastic** and **deterministic** steps

# Basic idea, now with the maths 🤪

## ▶ Notations

- ▶ Model parameters  $\theta$ : physical and nuisance parameters
- ▶ Latent variables  $z$ : internal, unobservable state of the system (see examples below)
- ▶ Data  $x$ : simulated or observed

## ▶ Simulator

- ▶ Given params, the simulator samples  $z \sim P(z|\theta)$  and generates data  $x \sim P(x|\theta, z)$
- ▶ May involve both **stochastic** and **deterministic** steps
- ▶ Varies a lot between fields/experiments, so **no one-size-fits-all method**

# Basic idea, now with the maths 🤪

## ▶ Notations

- ▶ Model parameters  $\theta$ : physical and nuisance parameters
- ▶ Latent variables  $z$ : internal, unobservable state of the system (see examples below)
- ▶ Data  $x$ : simulated or observed

## ▶ Simulator

- ▶ Given params, the simulator samples  $z \sim P(z|\theta)$  and generates data  $x \sim P(x|\theta, z)$
- ▶ May involve both **stochastic** and **deterministic** steps
- ▶ Varies a lot between fields/experiments, so **no one-size-fits-all method**

## ▶ Goal

# Basic idea, now with the maths 🤪

## ▶ Notations

- ▶ Model parameters  $\theta$ : physical and nuisance parameters
- ▶ Latent variables  $z$ : internal, unobservable state of the system (see examples below)
- ▶ Data  $x$ : simulated or observed

## ▶ Simulator

- ▶ Given params, the simulator samples  $z \sim P(z|\theta)$  and generates data  $x \sim P(x|\theta, z)$
- ▶ May involve both **stochastic** and **deterministic** steps
- ▶ Varies a lot between fields/experiments, so **no one-size-fits-all method**

## ▶ Goal

- ▶ Infer parameter constraints, e.g., the posterior distribution  $P(\theta|x_{\text{obs}})$

# Basic idea, now with the maths 🤔

## ▶ Notations

- ▶ Model parameters  $\theta$ : physical and nuisance parameters
- ▶ Latent variables  $z$ : internal, unobservable state of the system (see examples below)
- ▶ Data  $x$ : simulated or observed

## ▶ Simulator

- ▶ Given params, the simulator samples  $z \sim P(z|\theta)$  and generates data  $x \sim P(x|\theta, z)$
- ▶ May involve both **stochastic** and **deterministic** steps
- ▶ Varies a lot between fields/experiments, so **no one-size-fits-all method**

## ▶ Goal

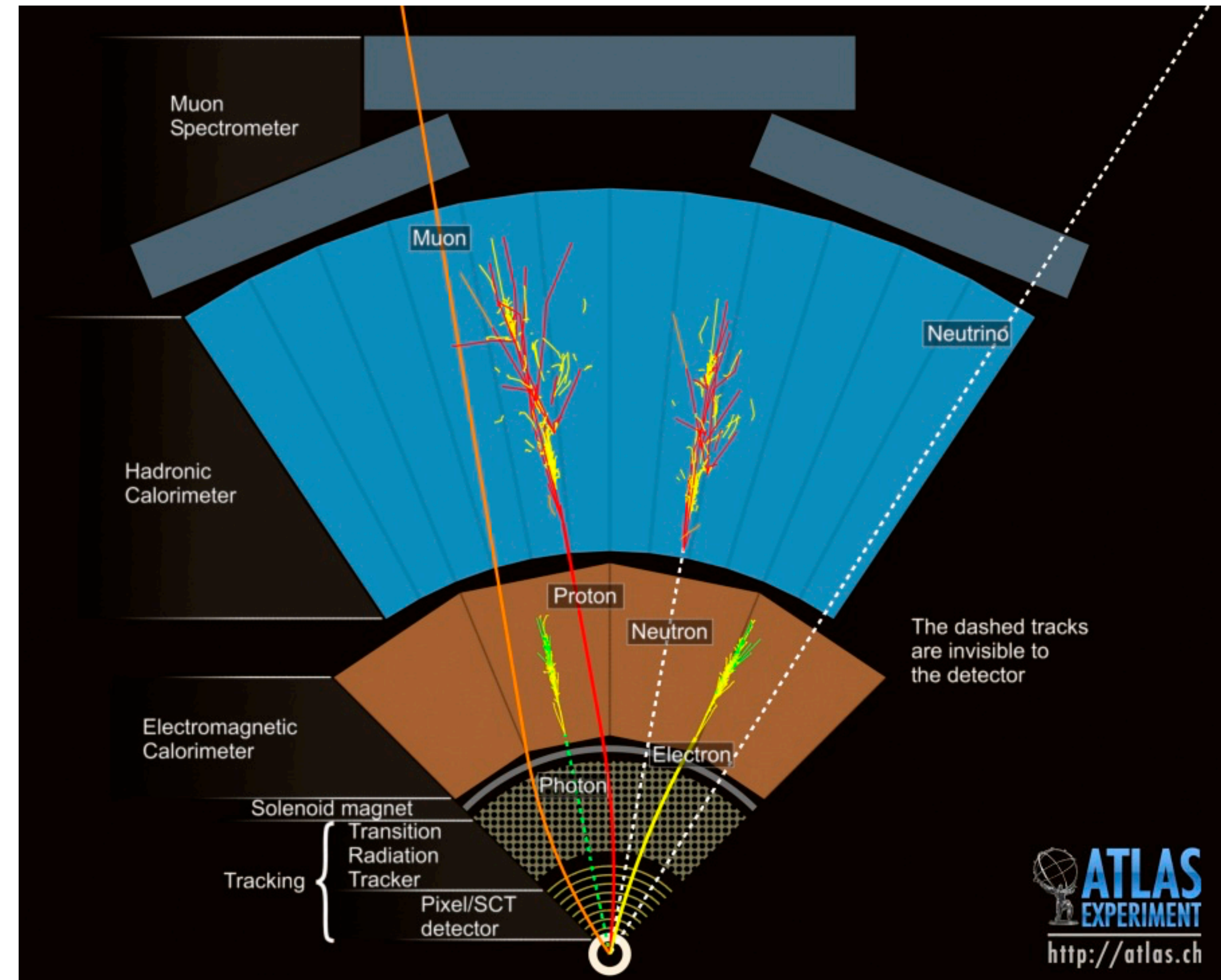
- ▶ Infer parameter constraints, e.g., the posterior distribution  $P(\theta|x_{\text{obs}})$
- ▶ The simulator likelihood is  $P(x|\theta) = \int dz P(x, z|\theta)$ , which is intractable in general.

# Simulator examples

## Particle physics

Parameters	masses, couplings, normalization factors and nuisance parameters, ie $\mathcal{O}(10)$
Latent variables	invariant mass, parton momenta, shower splitting, interactions with $10^8$ detectors = $\mathcal{O}(10^8/\text{event})$
Data	i.i.d. events with various cuts, typically binned by invariant mass
Simulation properties	fast, stochastic simulations of many independent events, $\mathcal{O}(1\text{ min/event})$

Monte-Carlo simulation of an event in the ATLAS detector at CERN



# Simulator examples

## Cosmology

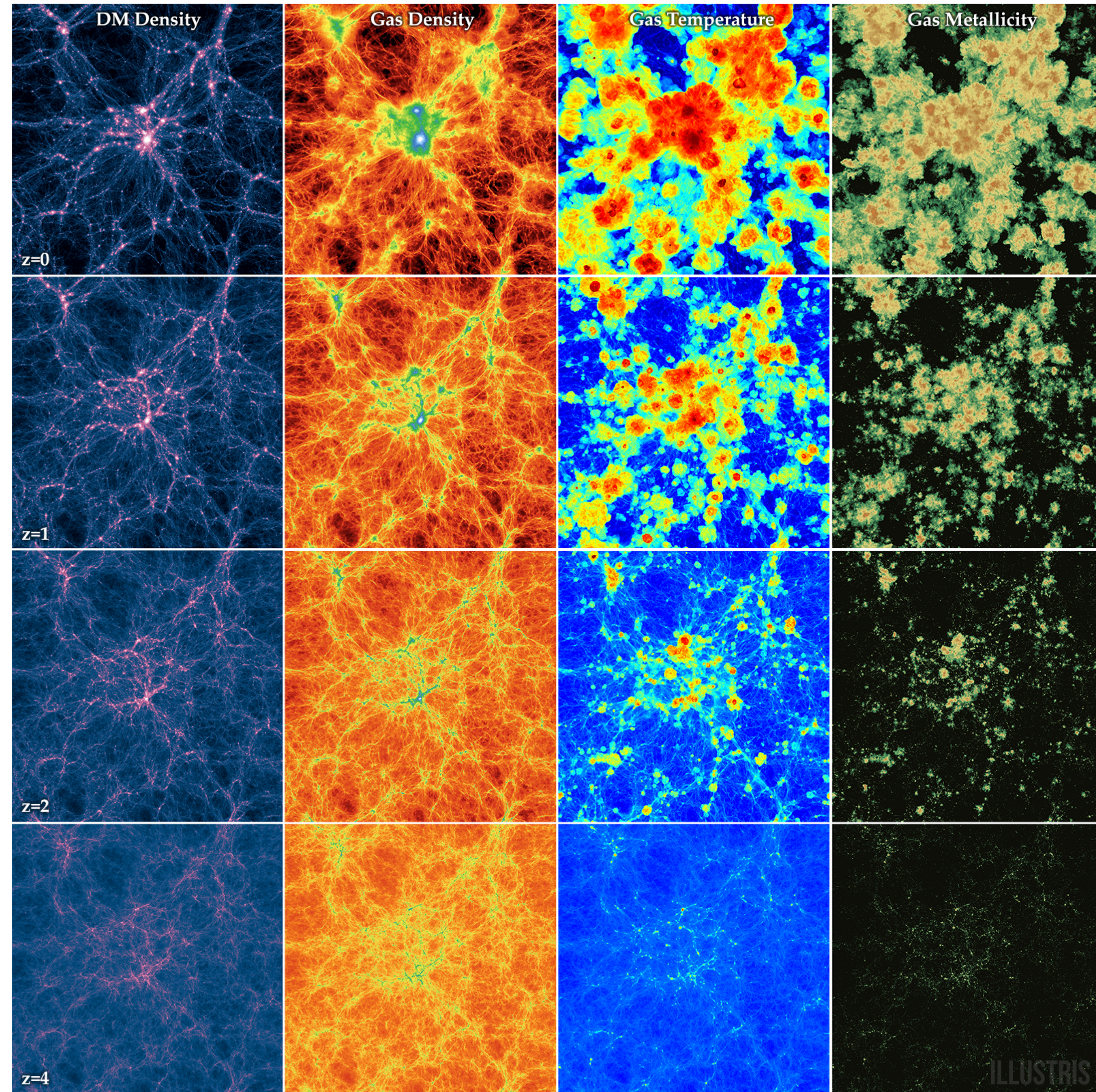
**Parameters** cosmology (matter density, expansion, etc) and nuisance parameters, ie  $\mathcal{O}(10)$

**Latent variables** initial matter distribution (density or particle  $x, v$ ) =  $\mathcal{O}(10^{10})$

**Data** single observation of cosmic fields or their summary statistics

**Simulation properties** slow, deterministic evolution of cosmological fields via particles,  $\mathcal{O}(10^8\text{s})$

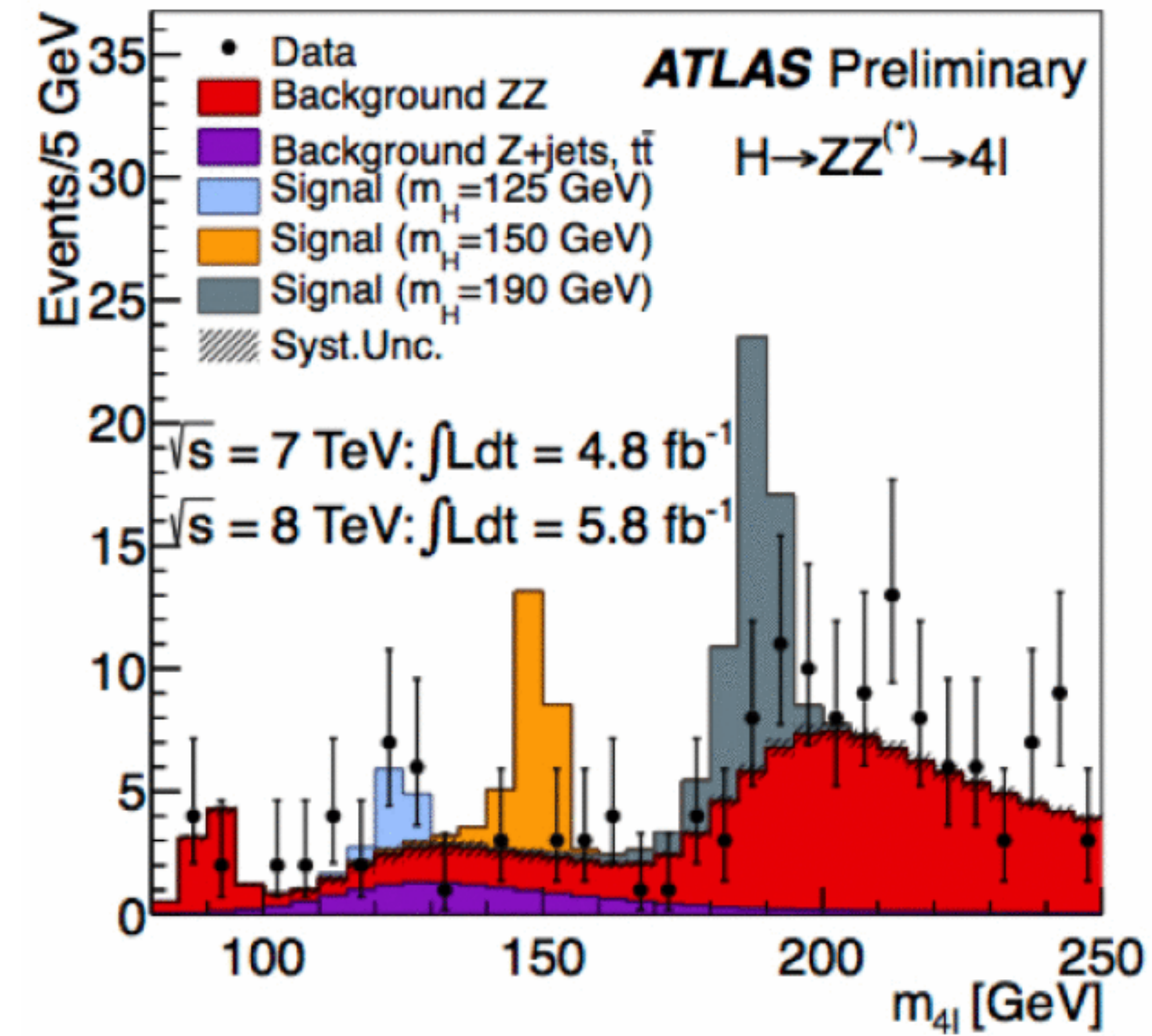
Illustris simulation at redshifts  $z=0, 1, 2$  and  $4$



# Emulators

Not really what we mean by SBI 🤔

ATLAS Higgs search



# Emulators

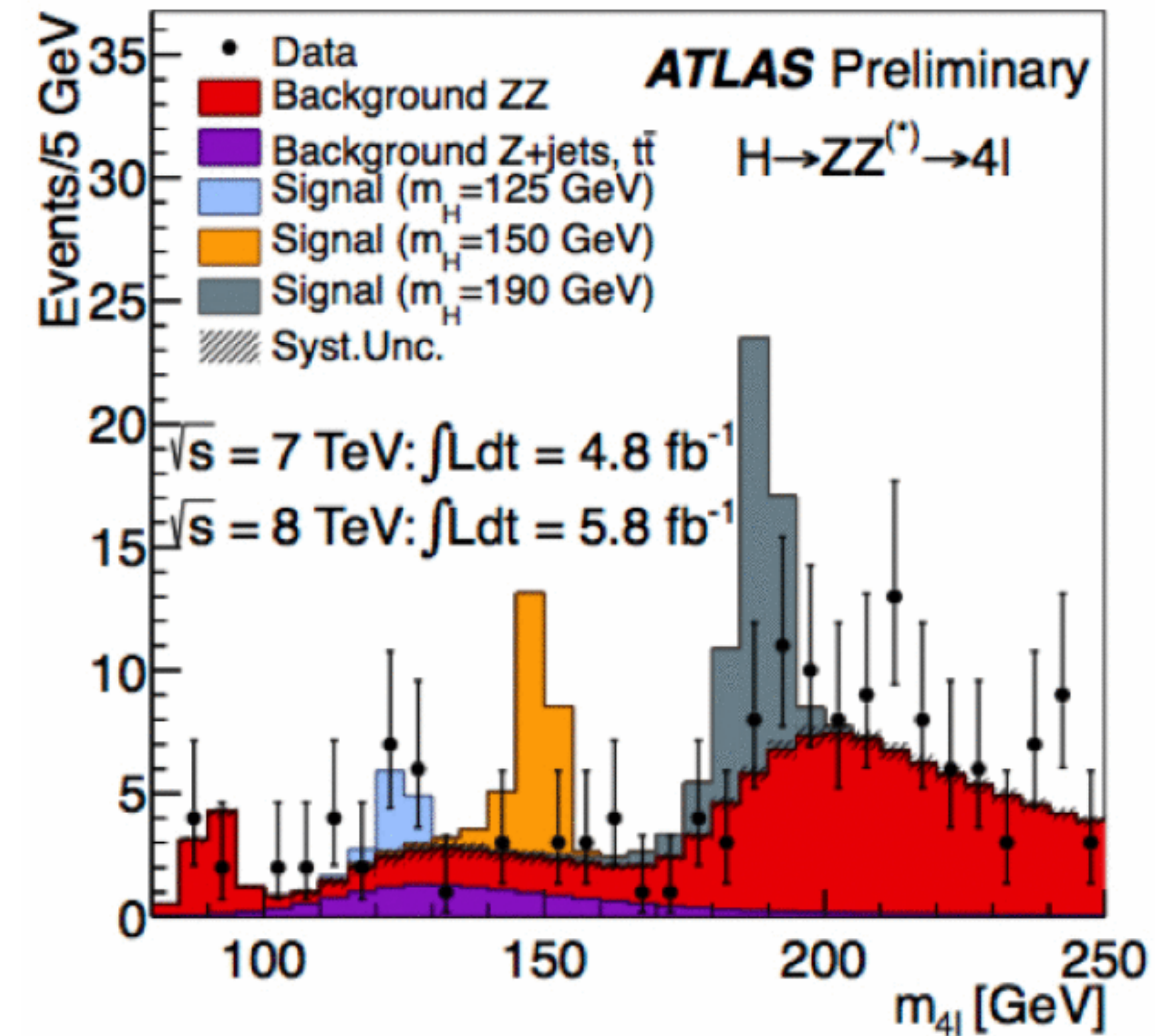
Not really what we mean by SBI 🤔

## ► Likelihood *approximation*

- Standard form = mix of Poisson + Gaussian, eg

$$\mathcal{L}(n_{\text{events}}|\mu, \theta) = \prod_{i \in \text{bins}} \mathcal{P}(n_i; \lambda = \mu S_i(\theta_{\text{phys}}) + B_i(\theta_{\text{syst}})) \times \mathcal{N}(\theta)$$

ATLAS Higgs search



# Emulators

Not really what we mean by SBI 🤔

## ► Likelihood *approximation*

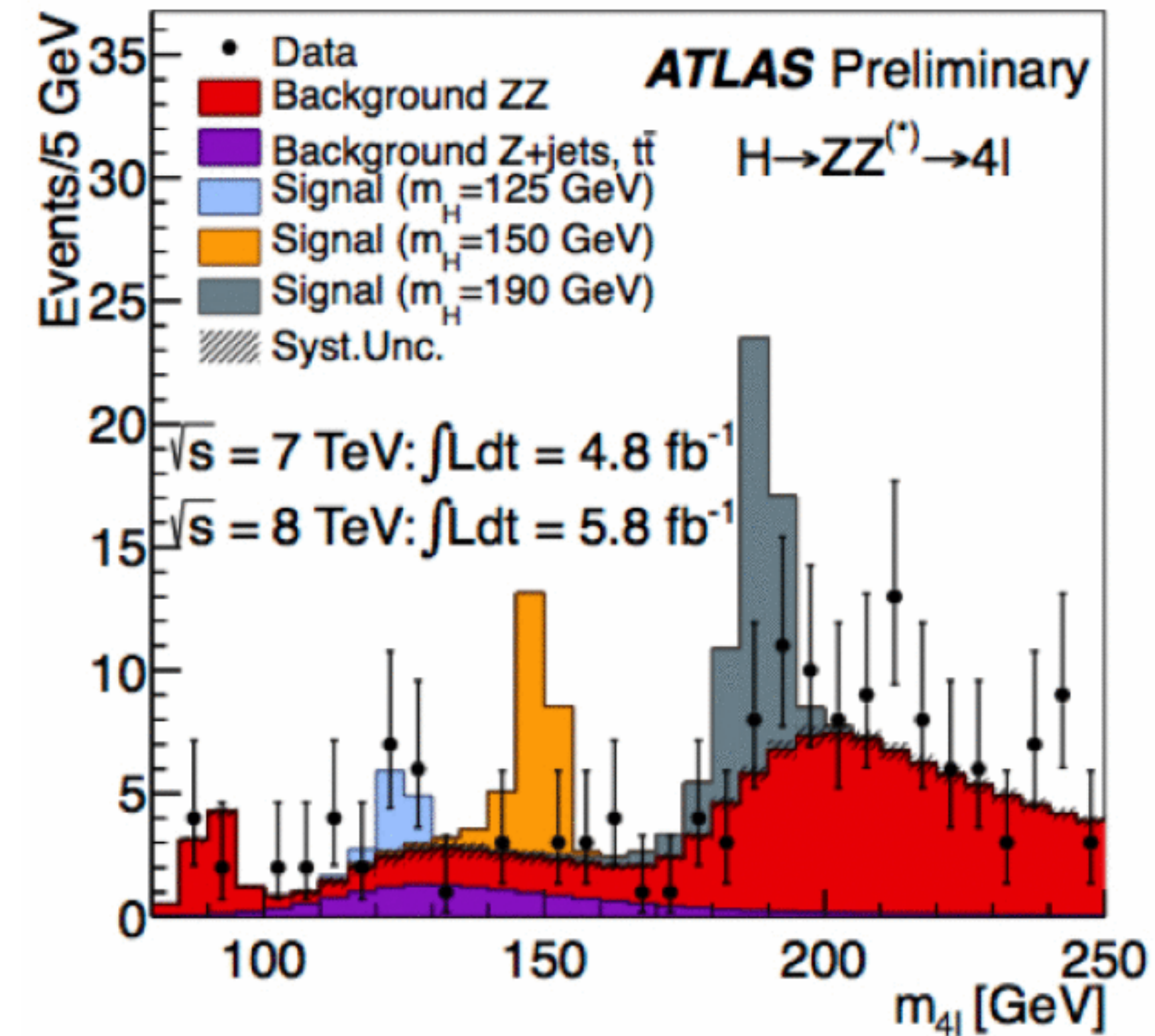
- Standard form = mix of Poisson + Gaussian, eg

$$\mathcal{L}(n_{\text{events}}|\mu, \theta) = \prod_{i \in \text{bins}} \mathcal{P}(n_i; \lambda = \mu S_i(\theta_{\text{phys}}) + B_i(\theta_{\text{syst}})) \times \mathcal{N}(\theta)$$

## ► Expected signal/background estimated from **templates**

1. Sample parameter space ~ uniformly (e.g. latin hypercube)
2. At each parameter value  $\theta$ :
  - a. Generate multiple simulations
  - b. Measure summary statistics ( $S(\theta)$  or  $B(\theta)$ ) on each simulation
  - c. Compute average measurements (decrease theory noise)
3. **Interpolate** averaged  $S(\theta)$  between  $\theta$ 's (e.g., Gaussian processes)

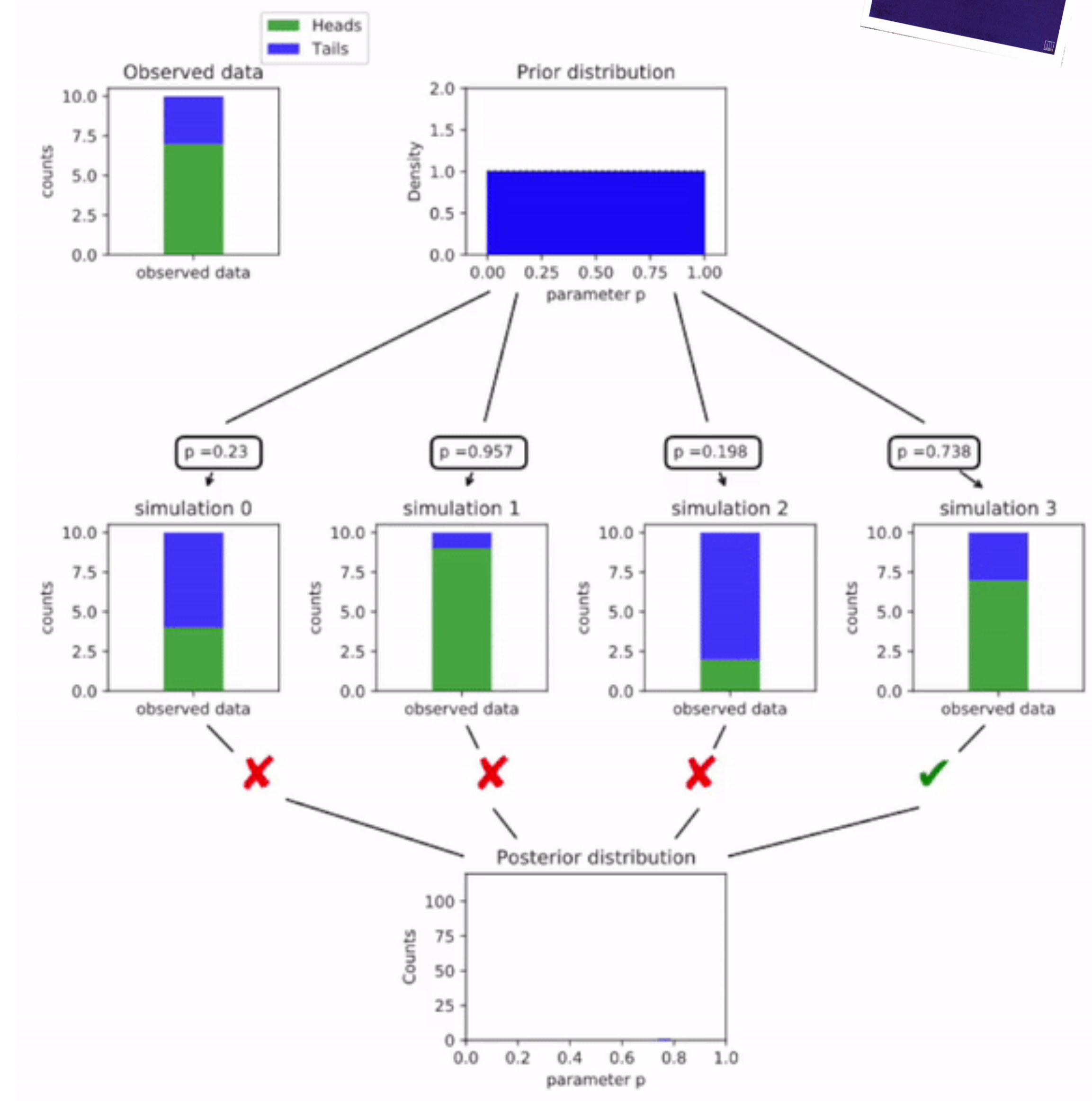
ATLAS Higgs search



# Approximate Bayesian Computation



A-B-C, it's easy as 1-2-3 🧑



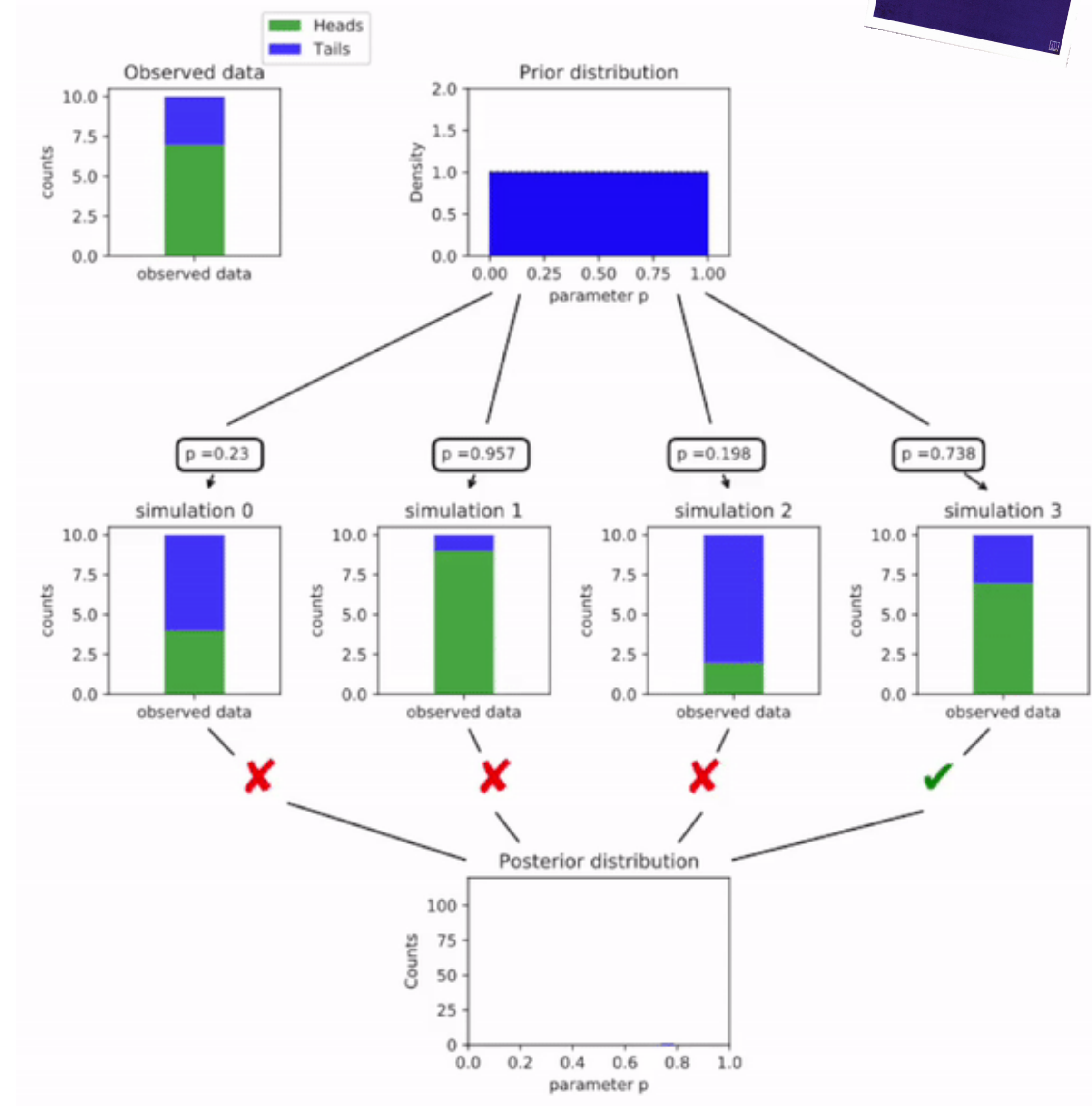
# Approximate Bayesian Computation



A-B-C, it's easy as 1-2-3 

## ► Historical approach

- ABC has existed since the 1980's
- Many successful applications: epidemiology, paleontology, cosmology, you name it!



# Approximate Bayesian Computation



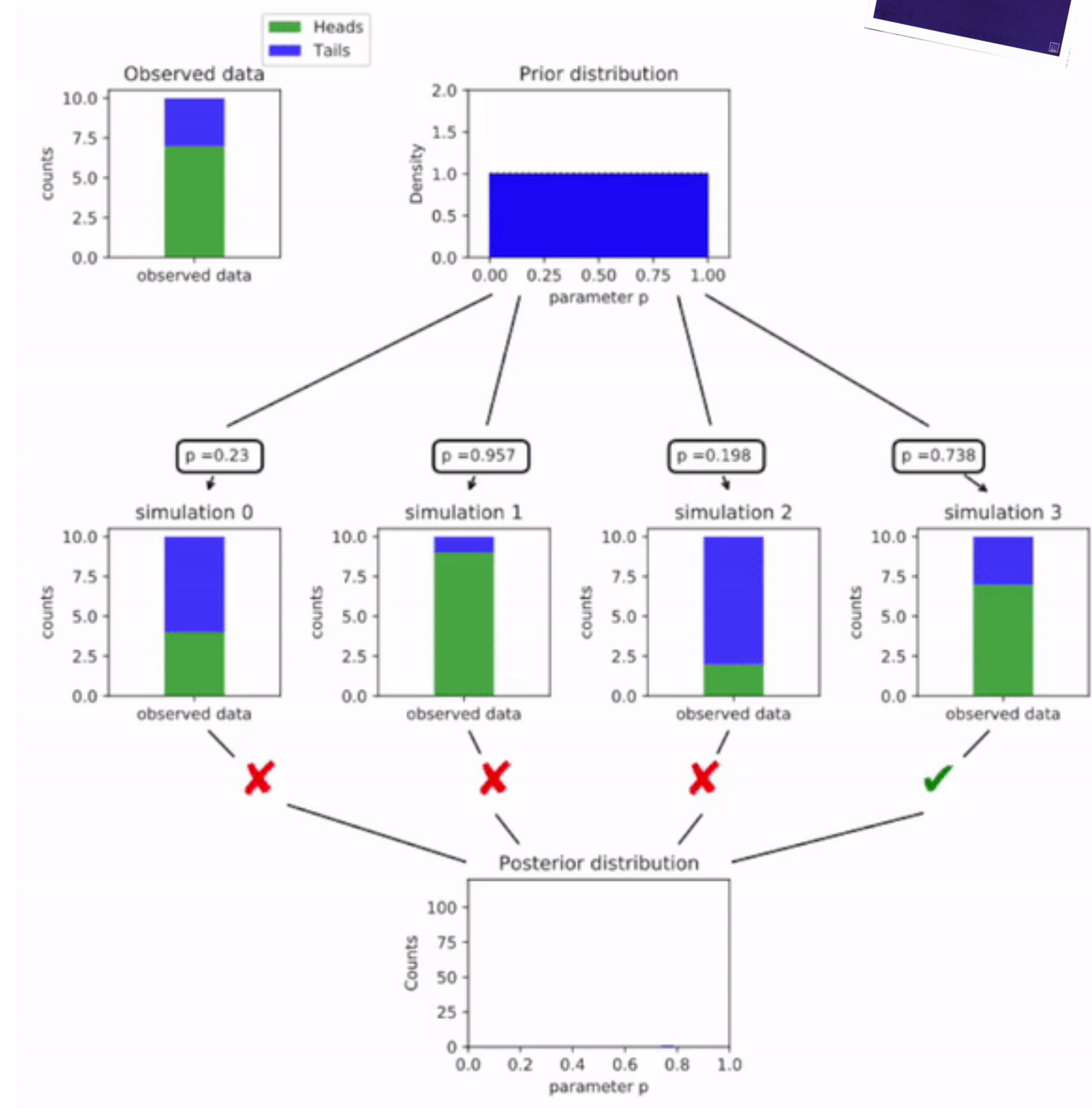
A-B-C, it's easy as 1-2-3 🧑

## ▶ Historical approach

- ▶ ABC has existed since the 1980's
- ▶ Many successful applications: epidemiology, paleontology, cosmology, you name it!

## ▶ Algorithm based on **rejection sampling** (no maths)

1. Sample parameter space
2. Generate simulations at these params values
3. Keep only params where  $\text{sim} \sim \text{data}$



# Approximate Bayesian Computation



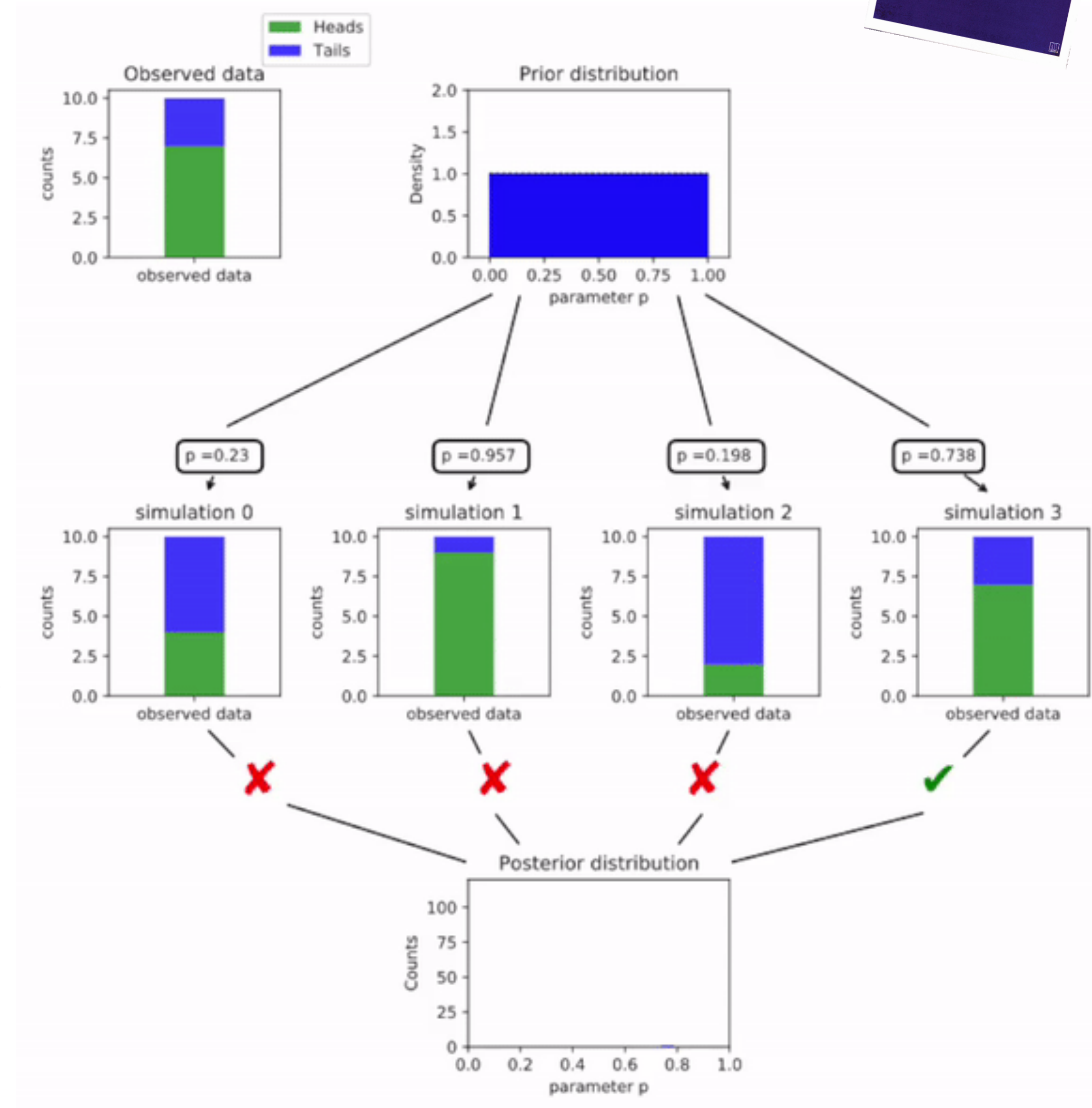
A-B-C, it's easy as 1-2-3 🧑

## ▶ Historical approach

- ▶ ABC has existed since the 1980's
- ▶ Many successful applications: epidemiology, paleontology, cosmology, you name it!

## ▶ Algorithm based on **rejection sampling** (no maths)

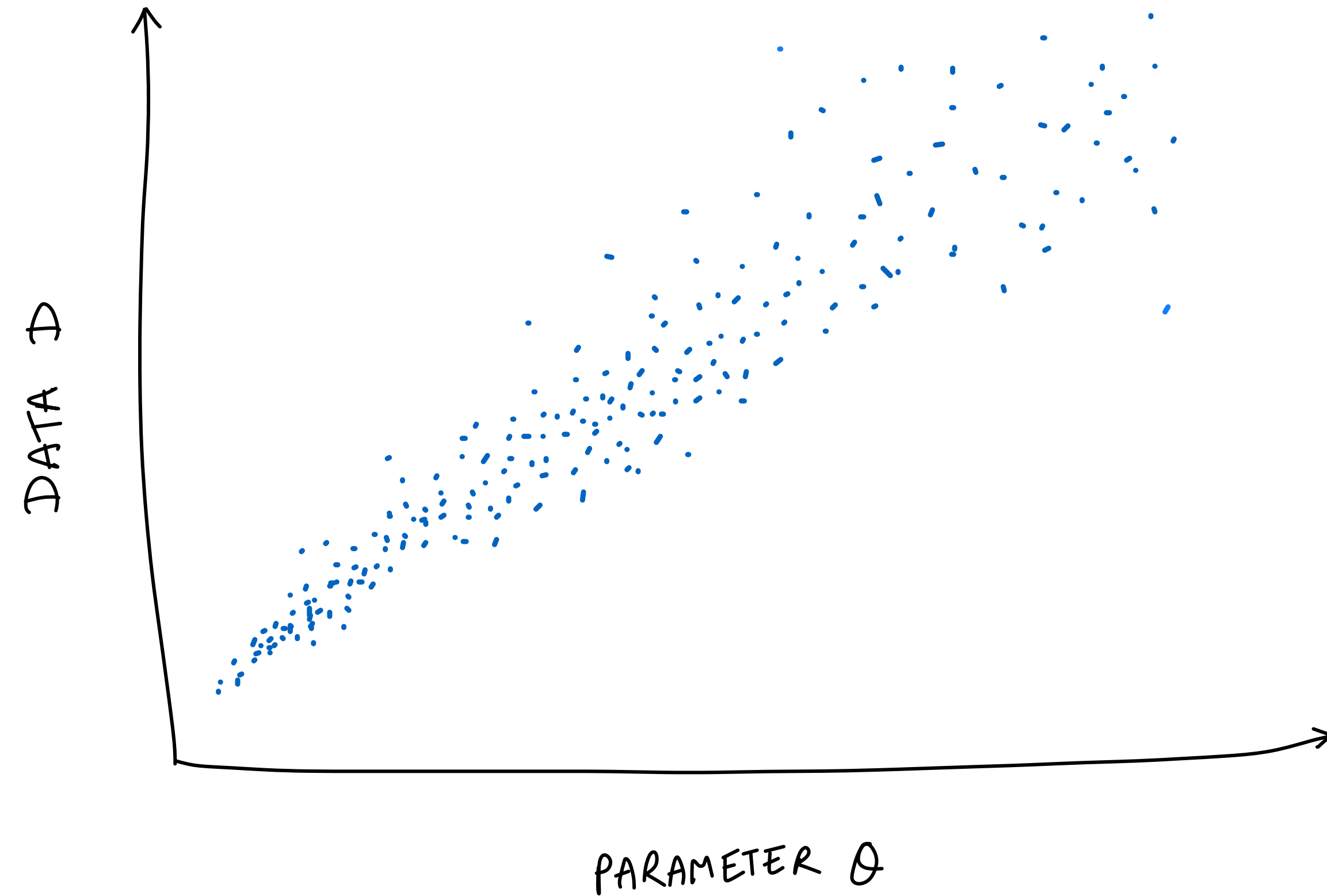
1. Sample parameter space
2. Generate simulations at these params values
3. Keep only params where  $\text{sim} \sim \text{data}$



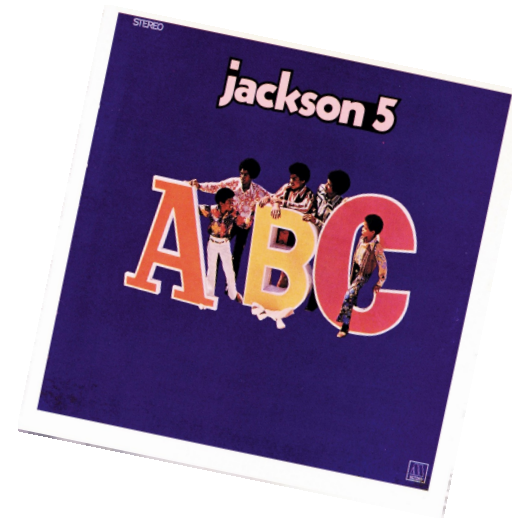
# Approximate Bayesian Computation

A-B-C, it's easy as 1-2-3 🧑

- ▶ Algorithm based on rejection sampling (with maths)

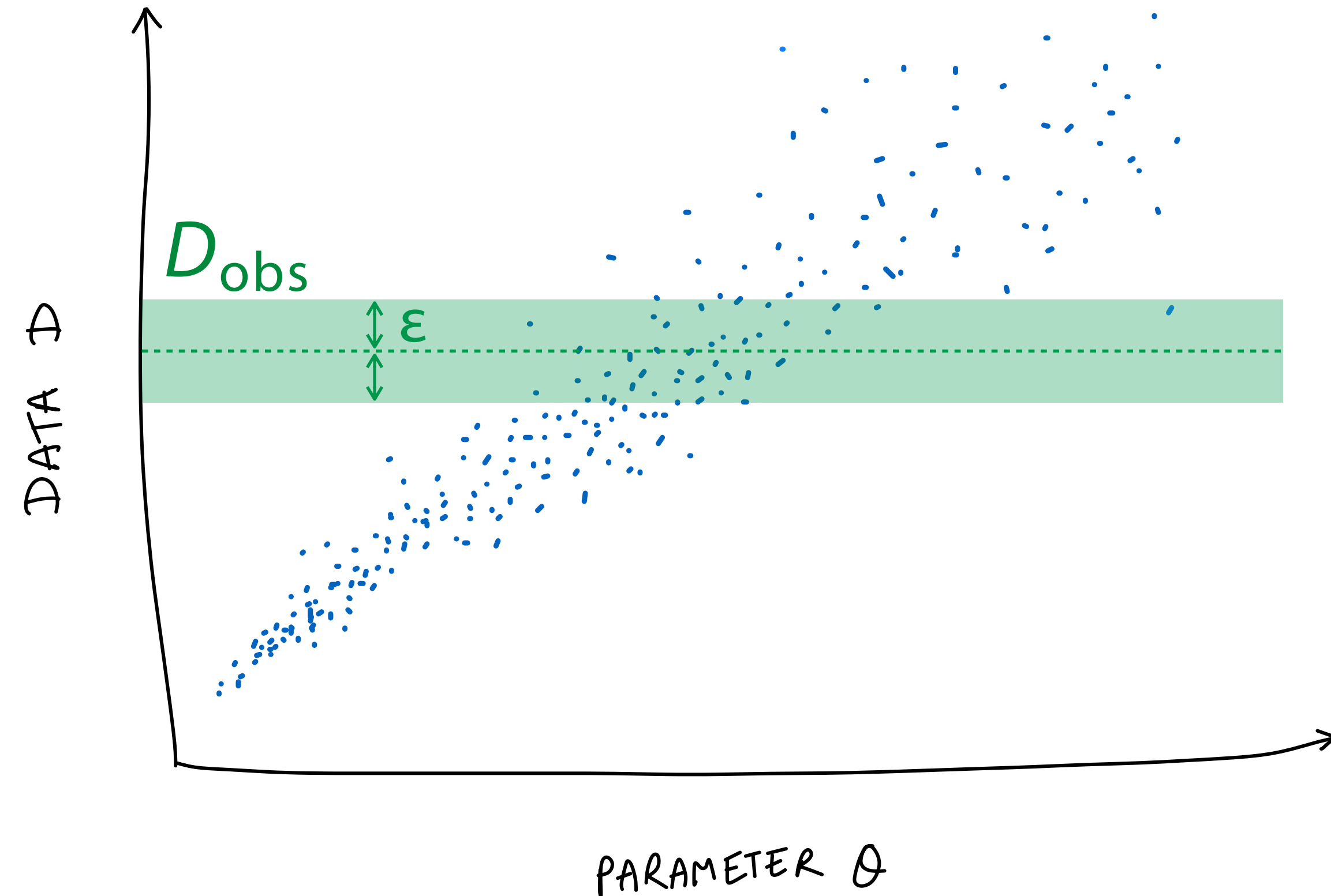


# Approximate Bayesian Computation



A-B-C, it's easy as 1-2-3 🧑

- ▶ Algorithm based on rejection sampling (with maths)
  - ▶ Given a **distance**  $\rho$  in data space, a **threshold**  $\varepsilon > 0$ , and observed data  $d_{\text{obs}}$

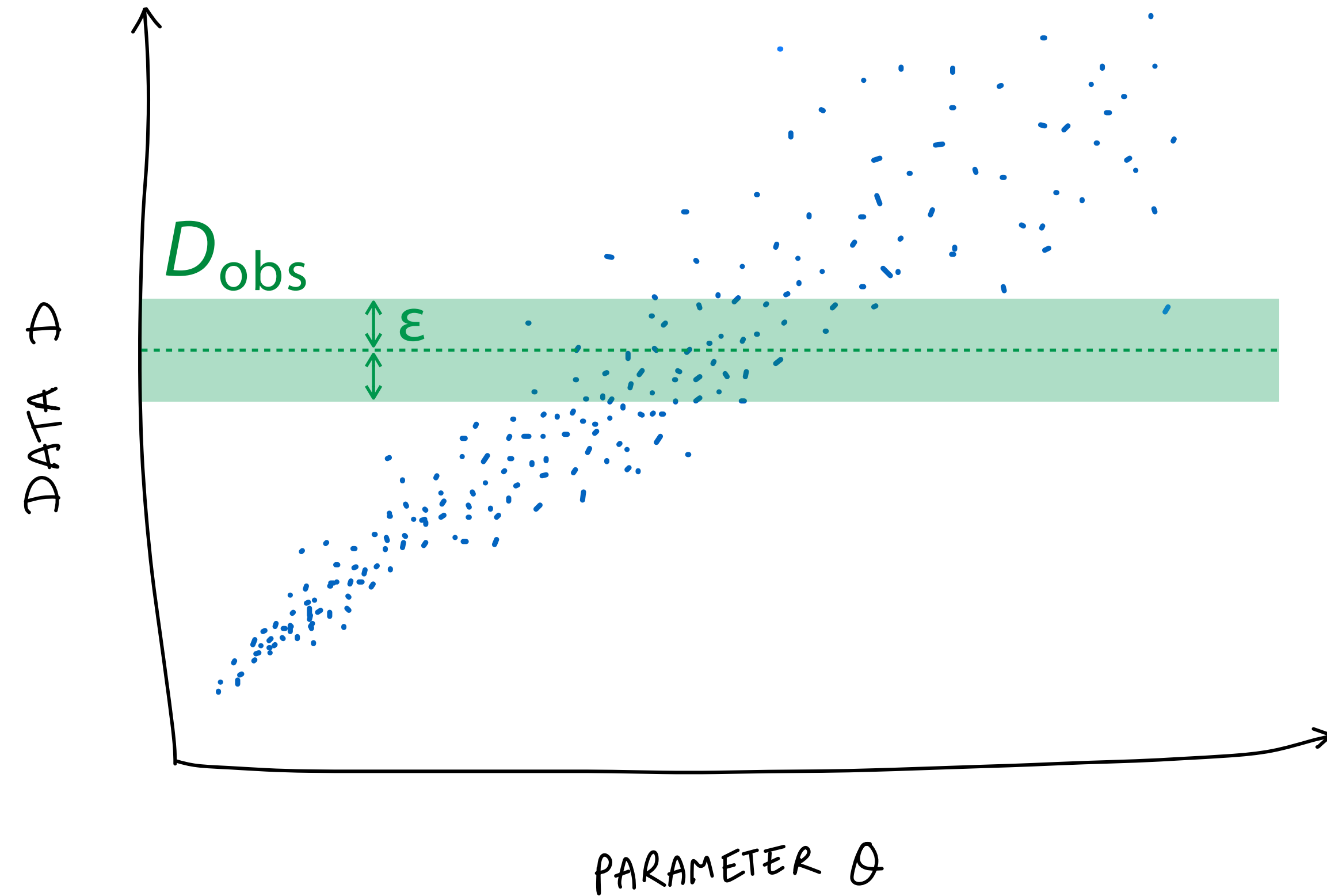


# Approximate Bayesian Computation

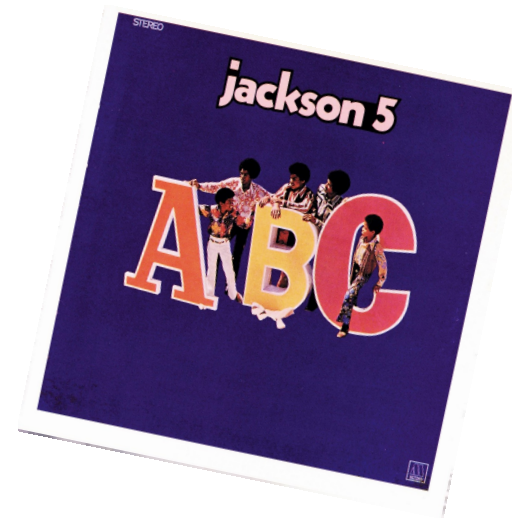


A-B-C, it's easy as 1-2-3 🧑

- ▶ Algorithm based on rejection sampling (with maths)
  - ▶ Given a **distance**  $\rho$  in data space, a **threshold**  $\varepsilon > 0$ , and observed data  $d_{\text{obs}}$
  - ▶ For  $i=1 \dots N_{\text{step}}$

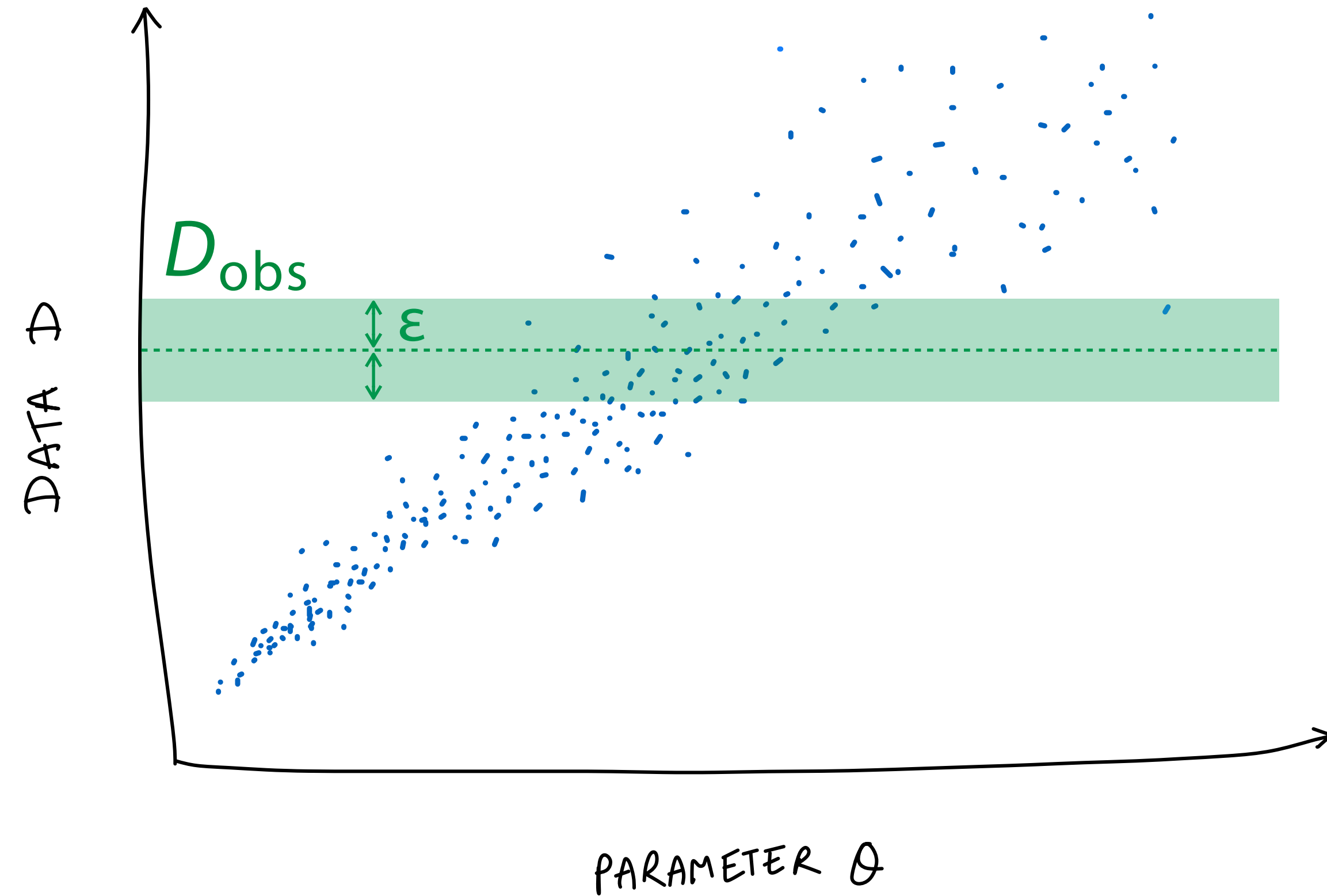


# Approximate Bayesian Computation

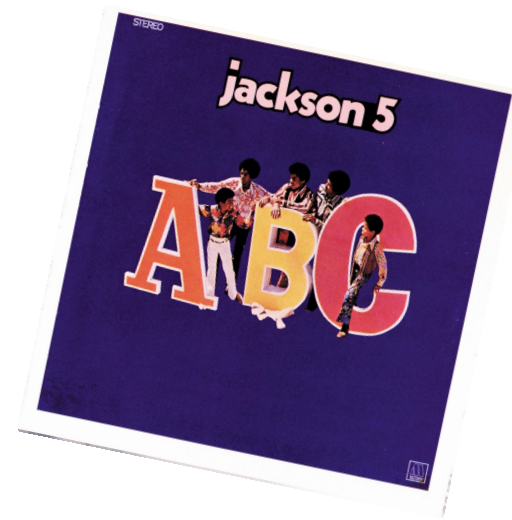


A-B-C, it's easy as 1-2-3 🧑

- ▶ Algorithm based on rejection sampling (with maths)
  - ▶ Given a **distance**  $\rho$  in data space, a **threshold**  $\varepsilon > 0$ , and observed data  $d_{\text{obs}}$
  - ▶ For  $i=1 \dots N_{\text{step}}$ 
    1. Draw  $\theta_i \sim P(\theta)$



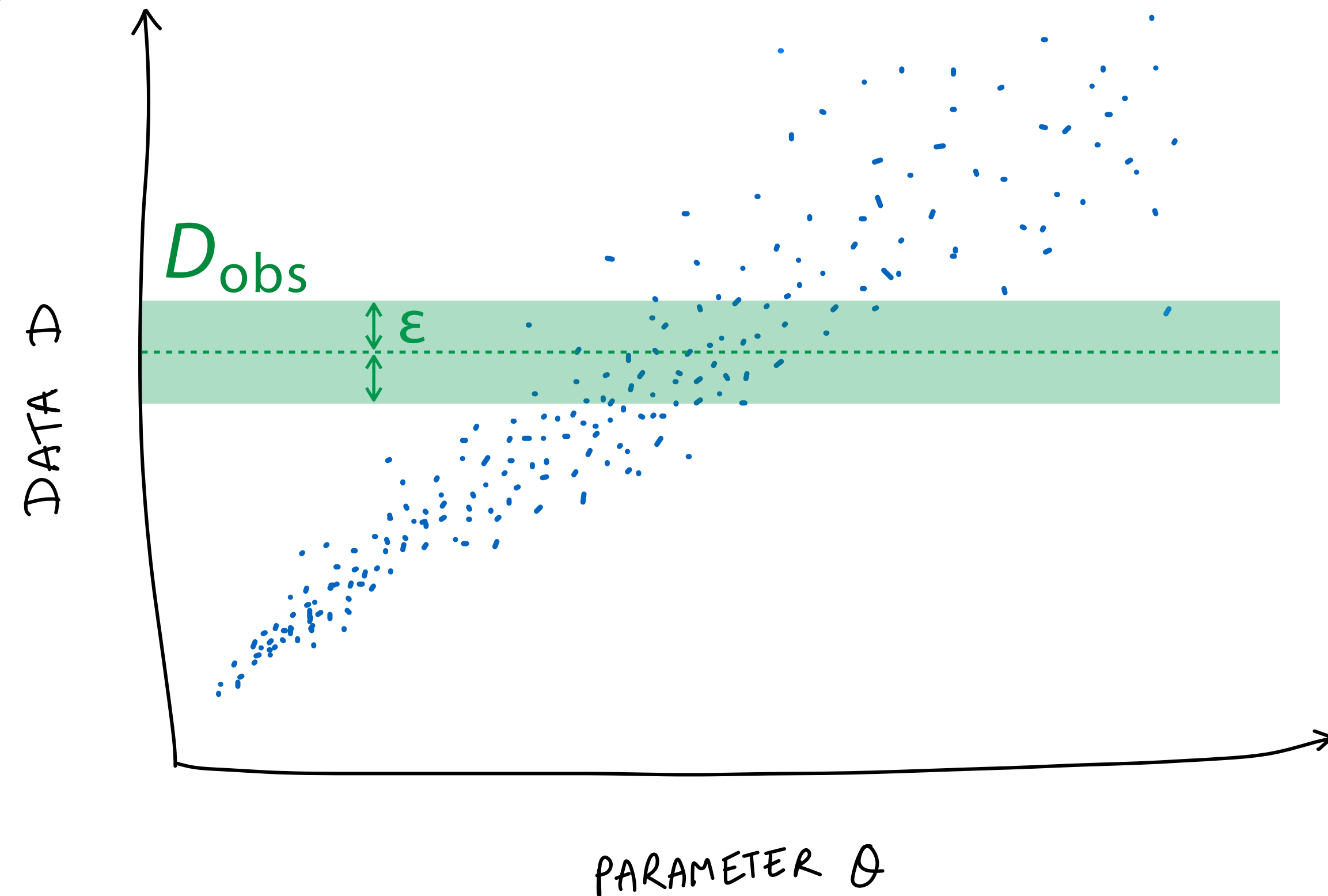
# Approximate Bayesian Computation



A-B-C, it's easy as 1-2-3 🧑

- ▶ Algorithm based on rejection sampling (with maths)

- ▶ Given a **distance**  $\rho$  in data space, a **threshold**  $\varepsilon > 0$ , and observed data  $d_{\text{obs}}$
- ▶ For  $i=1 \dots N_{\text{step}}$ 
  1. Draw  $\theta_i \sim P(\theta)$
  2. Run simulator at  $\theta_i$  to generate data  $d_i \sim P(d|\theta)$



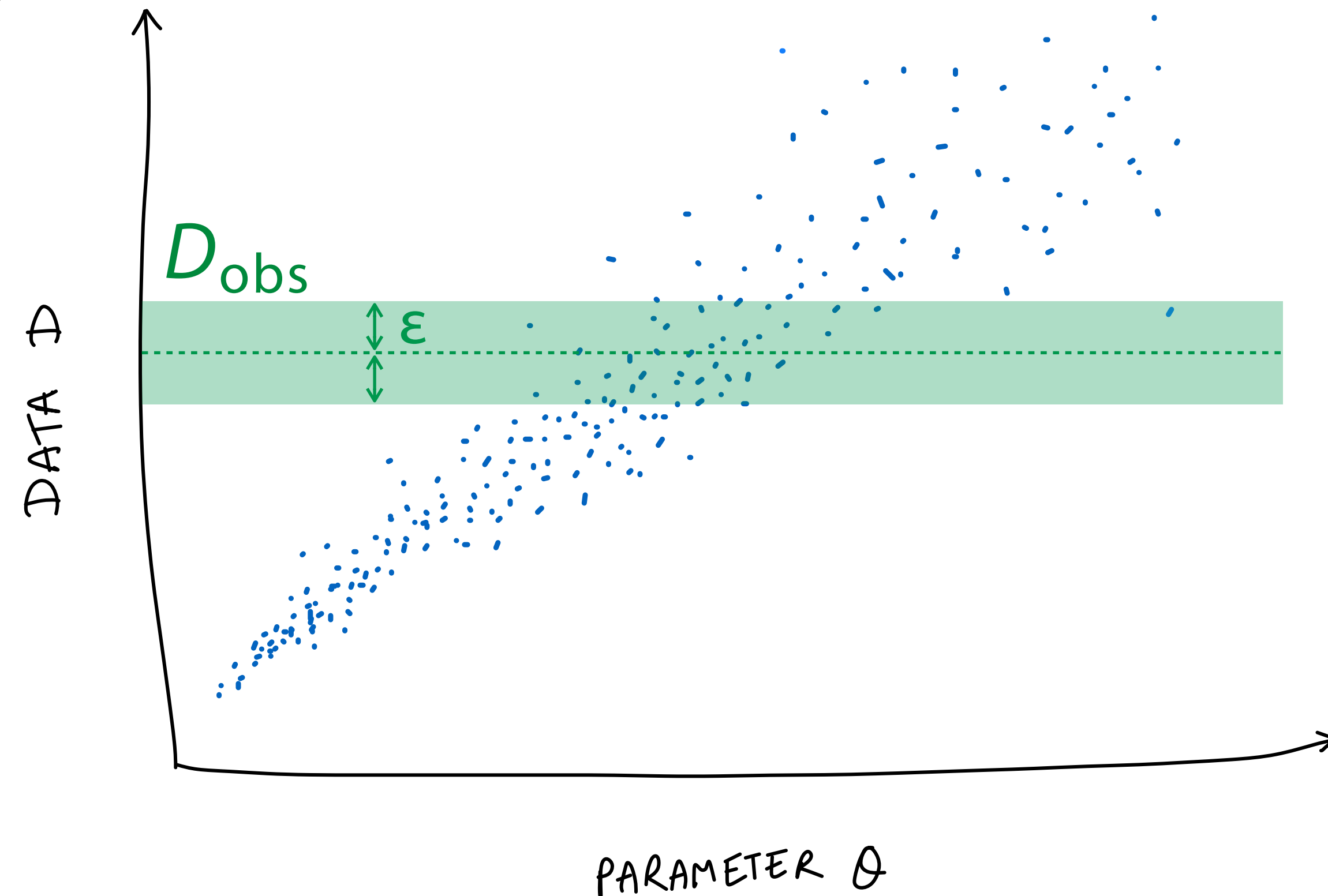
# Approximate Bayesian Computation



A-B-C, it's easy as 1-2-3 🧑

## ▶ Algorithm based on rejection sampling (with maths)

- ▶ Given a **distance**  $\rho$  in data space, a **threshold**  $\varepsilon > 0$ , and observed data  $d_{\text{obs}}$
- ▶ For  $i=1 \dots N_{\text{step}}$ 
  1. Draw  $\theta_i \sim P(\theta)$
  2. Run simulator at  $\theta_i$  to generate data  $d_i \sim P(d|\theta)$
  3. If  $\rho(d_i, d_{\text{obs}}) < \varepsilon$ , **accept** sample  $\theta_i$ , else **reject**



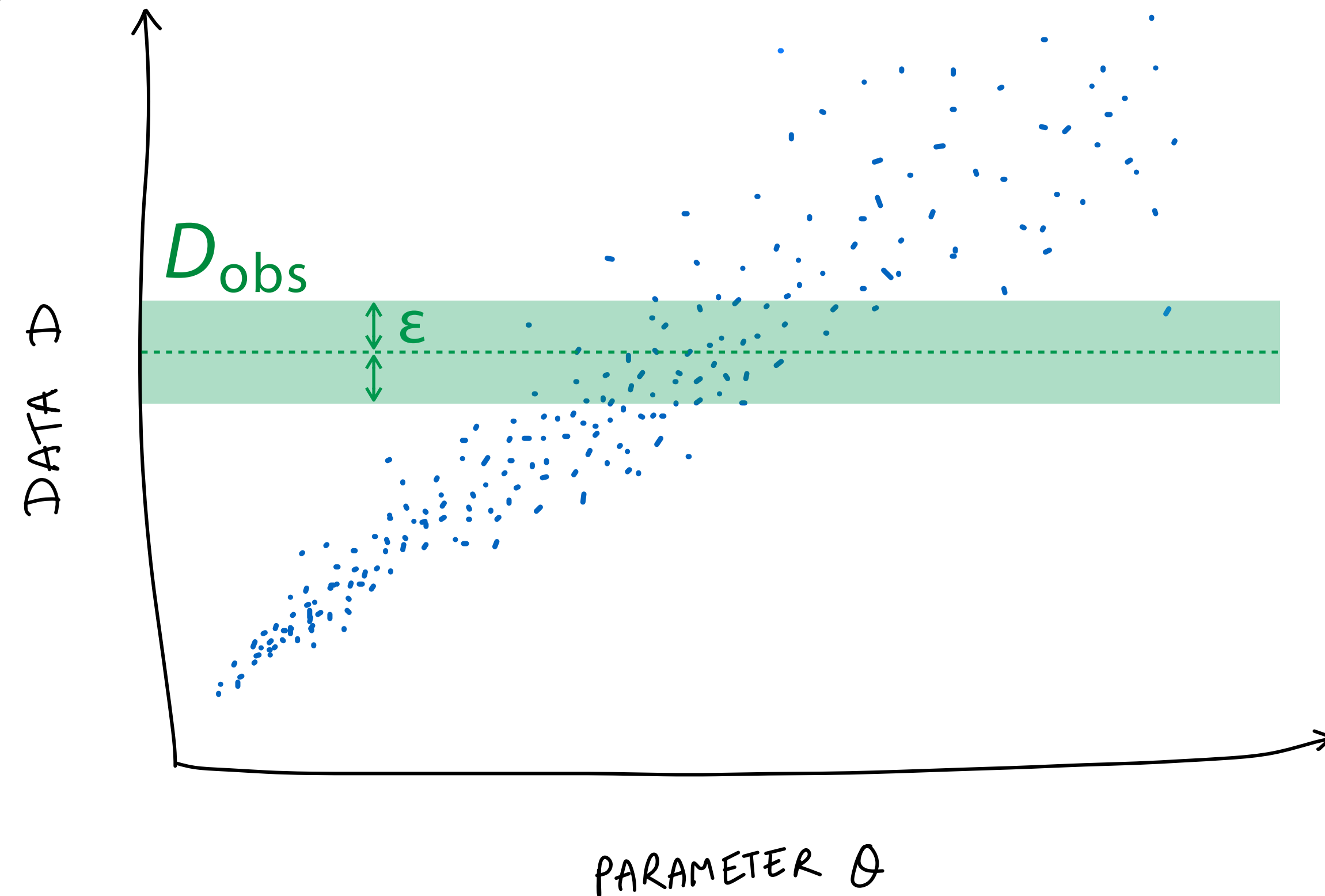
# Approximate Bayesian Computation



A-B-C, it's easy as 1-2-3 🧑

## ▶ Algorithm based on rejection sampling (with maths)

- ▶ Given a **distance**  $\rho$  in data space, a **threshold**  $\varepsilon > 0$ , and observed data  $d_{\text{obs}}$
- ▶ For  $i=1 \dots N_{\text{step}}$ 
  1. Draw  $\theta_i \sim P(\theta)$
  2. Run simulator at  $\theta_i$  to generate data  $d_i \sim P(d|\theta)$
  3. If  $\rho(d_i, d_{\text{obs}}) < \varepsilon$ , **accept** sample  $\theta_i$ , else **reject**
- ▶ Result: histogram of accepted samples  $\sim P(\theta|d_{\text{obs}})$



# Approximate Bayesian Computation

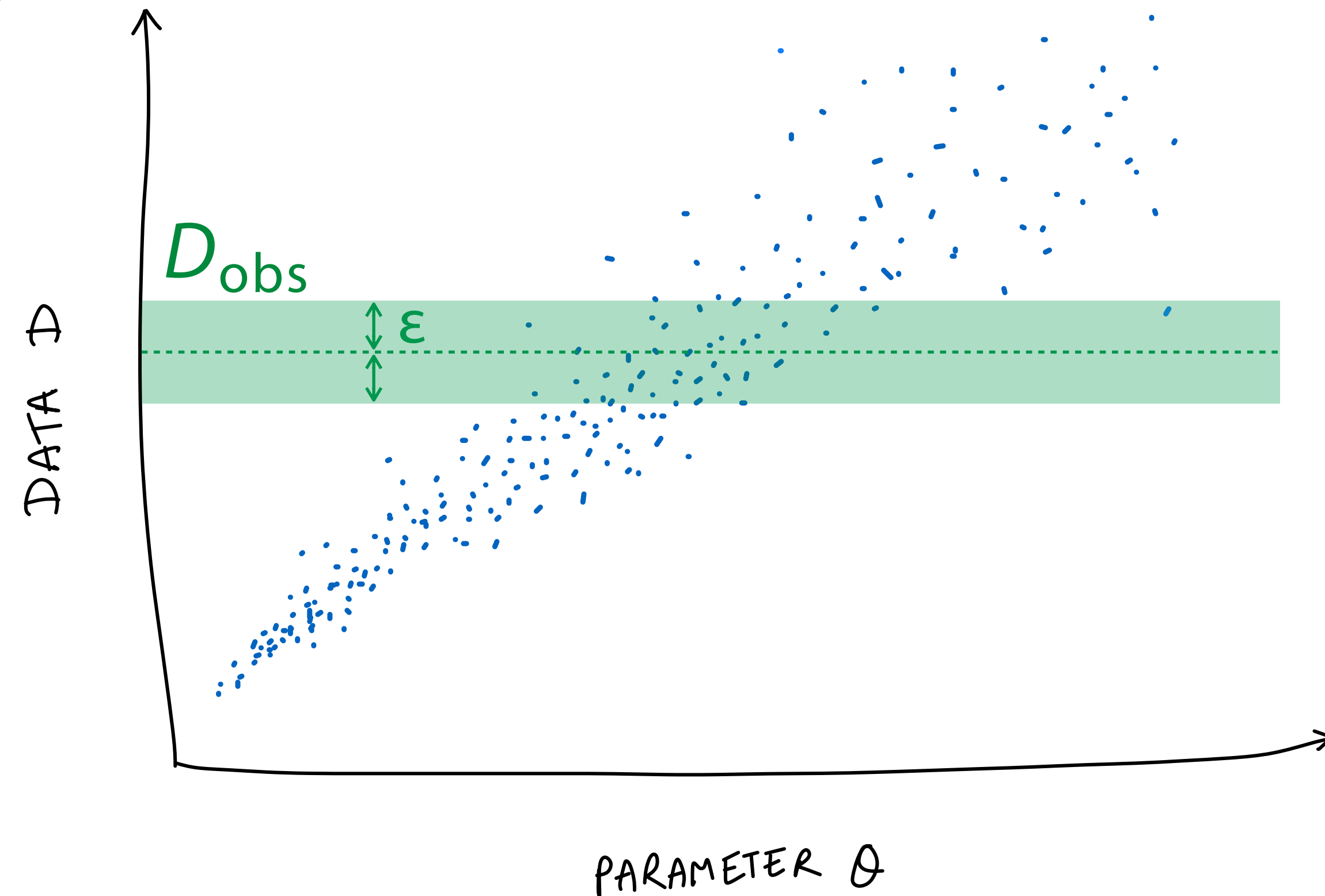


A-B-C, it's easy as 1-2-3 🧑

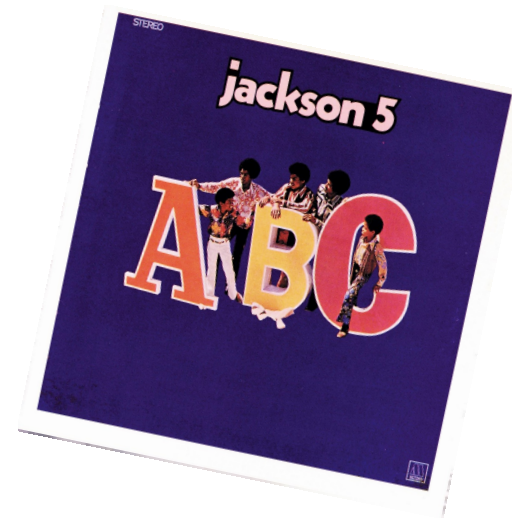
## ▶ Algorithm based on rejection sampling (with maths)

- ▶ Given a **distance**  $\rho$  in data space, a **threshold**  $\varepsilon > 0$ , and observed data  $d_{\text{obs}}$
- ▶ For  $i=1 \dots N_{\text{step}}$ 
  1. Draw  $\theta_i \sim P(\theta)$
  2. Run simulator at  $\theta_i$  to generate data  $d_i \sim P(d|\theta)$
  3. If  $\rho(d_i, d_{\text{obs}}) < \varepsilon$ , **accept** sample  $\theta_i$ , else **reject**
- ▶ Result: histogram of accepted samples  $\sim P(\theta|d_{\text{obs}})$

## ▶ **Major caveat:** choice of distance $\rho$ and threshold $\varepsilon$



# Approximate Bayesian Computation



A-B-C, it's easy as 1-2-3 🧑

## ▶ Algorithm based on rejection sampling (with maths)

▶ Given a **distance**  $\rho$  in data space, a **threshold**  $\varepsilon > 0$ , and observed data  $d_{\text{obs}}$

▶ For  $i=1 \dots N_{\text{step}}$

1. Draw  $\theta_i \sim P(\theta)$

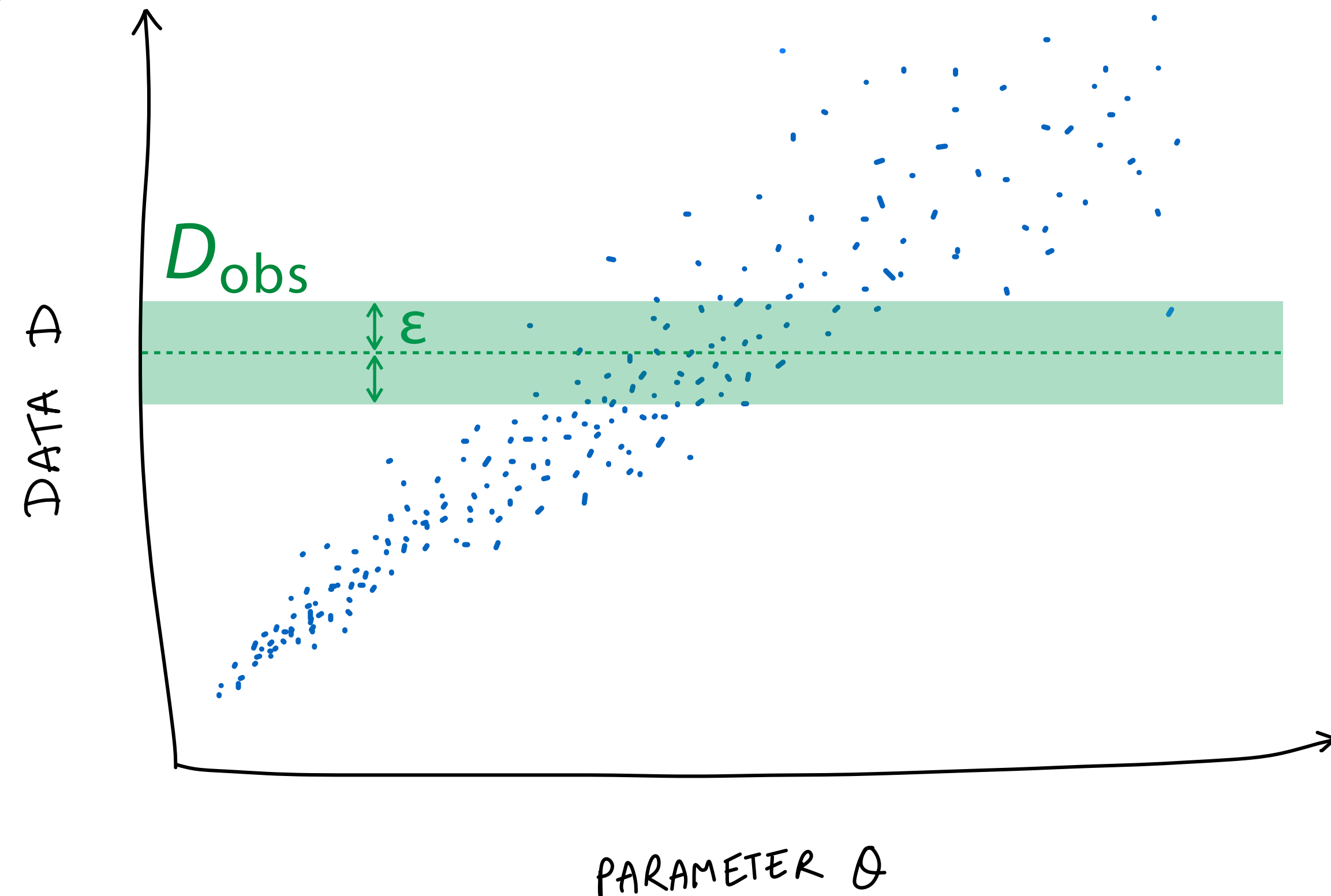
2. Run simulator at  $\theta_i$  to generate data  $d_i \sim P(d|\theta)$

3. If  $\rho(d_i, d_{\text{obs}}) < \varepsilon$ , **accept** sample  $\theta_i$ , else **reject**

▶ Result: histogram of accepted samples  $\sim P(\theta|d_{\text{obs}})$

## ▶ **Major caveat:** choice of distance $\rho$ and threshold $\varepsilon$

▶  $\varepsilon$  too small  $\rightarrow$  reject most sims  $\rightarrow$  inefficient



# Approximate Bayesian Computation



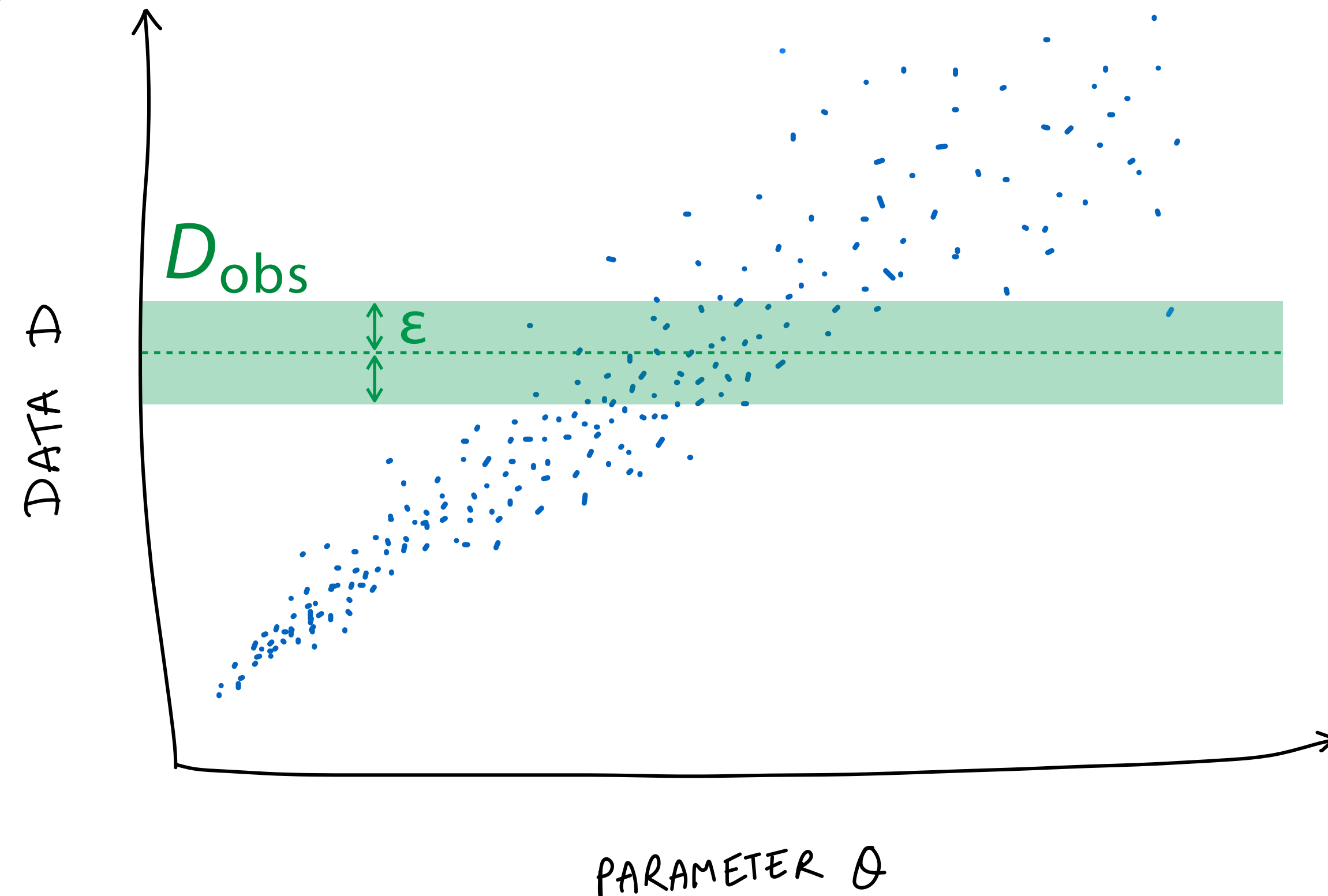
A-B-C, it's easy as 1-2-3 🧑

## ▶ Algorithm based on rejection sampling (with maths)

- ▶ Given a **distance**  $\rho$  in data space, a **threshold**  $\varepsilon > 0$ , and observed data  $d_{\text{obs}}$
- ▶ For  $i=1 \dots N_{\text{step}}$ 
  1. Draw  $\theta_i \sim P(\theta)$
  2. Run simulator at  $\theta_i$  to generate data  $d_i \sim P(d|\theta)$
  3. If  $\rho(d_i, d_{\text{obs}}) < \varepsilon$ , **accept** sample  $\theta_i$ , else **reject**
- ▶ Result: histogram of accepted samples  $\sim P(\theta|d_{\text{obs}})$

## ▶ **Major caveat:** choice of distance $\rho$ and threshold $\varepsilon$

- ▶  $\varepsilon$  too small  $\rightarrow$  reject most sims  $\rightarrow$  inefficient
- ▶  $\varepsilon$  too large  $\rightarrow$  distorted posterior  $\rightarrow$  inaccurate



# Neural SBI

SBI with generative models



Copilot prompt: particle physics simulations in the style of Basquiat paintings

# SBI with neural classifiers

Neural likelihood ratio estimation

# SBI with neural classifiers

## Neural likelihood ratio estimation

- Likelihood ratio
  - Neyman-Pearson lemma: likelihood ratio is the **optimal discriminator** between hypotheses
  - Amortized likelihood ratio

$$r(x|\theta) = \frac{\mathcal{L}(x|\theta)}{\mathcal{L}(x|\theta_0)}$$

# SBI with neural classifiers

## Neural likelihood ratio estimation

- ▶ Likelihood ratio
  - ▶ Neyman-Pearson lemma: likelihood ratio is the **optimal discriminator** between hypotheses
  - ▶ Amortized likelihood ratio

$$r(x|\theta) = \frac{\mathcal{L}(x|\theta)}{\mathcal{L}(x|\theta_0)}$$

- ▶ Train **NN classifier** between two sets of sims

- ▶ class=1  $(\theta, x) \sim \pi(\theta)\mathcal{L}(x|\theta)$

- ▶ class=0  $(\theta_0, x) \sim \delta(\theta_0)\mathcal{L}(x|\theta_0)$

then  $\text{NN}(x, \theta) \xrightarrow{\text{training}} \frac{1}{1 + \frac{1}{r(x, \theta)}}$  🙌

# SBI with neural classifiers

## Neural likelihood ratio estimation

- ▶ Likelihood ratio
  - ▶ Neyman-Pearson lemma: likelihood ratio is the **optimal discriminator** between hypotheses
  - ▶ Amortized likelihood ratio

$$r(x|\theta) = \frac{\mathcal{L}(x|\theta)}{\mathcal{L}(x|\theta_0)}$$

- ▶ Train **NN classifier** between two sets of sims

- ▶ class=1  $(\theta, x) \sim \pi(\theta)\mathcal{L}(x|\theta)$

then  $\text{NN}(x, \theta) \xrightarrow{\text{training}} \frac{1}{1 + \frac{1}{r(x, \theta)}}$  🙌

- ▶ class=0  $(\theta_0, x) \sim \delta(\theta_0)\mathcal{L}(x|\theta_0)$

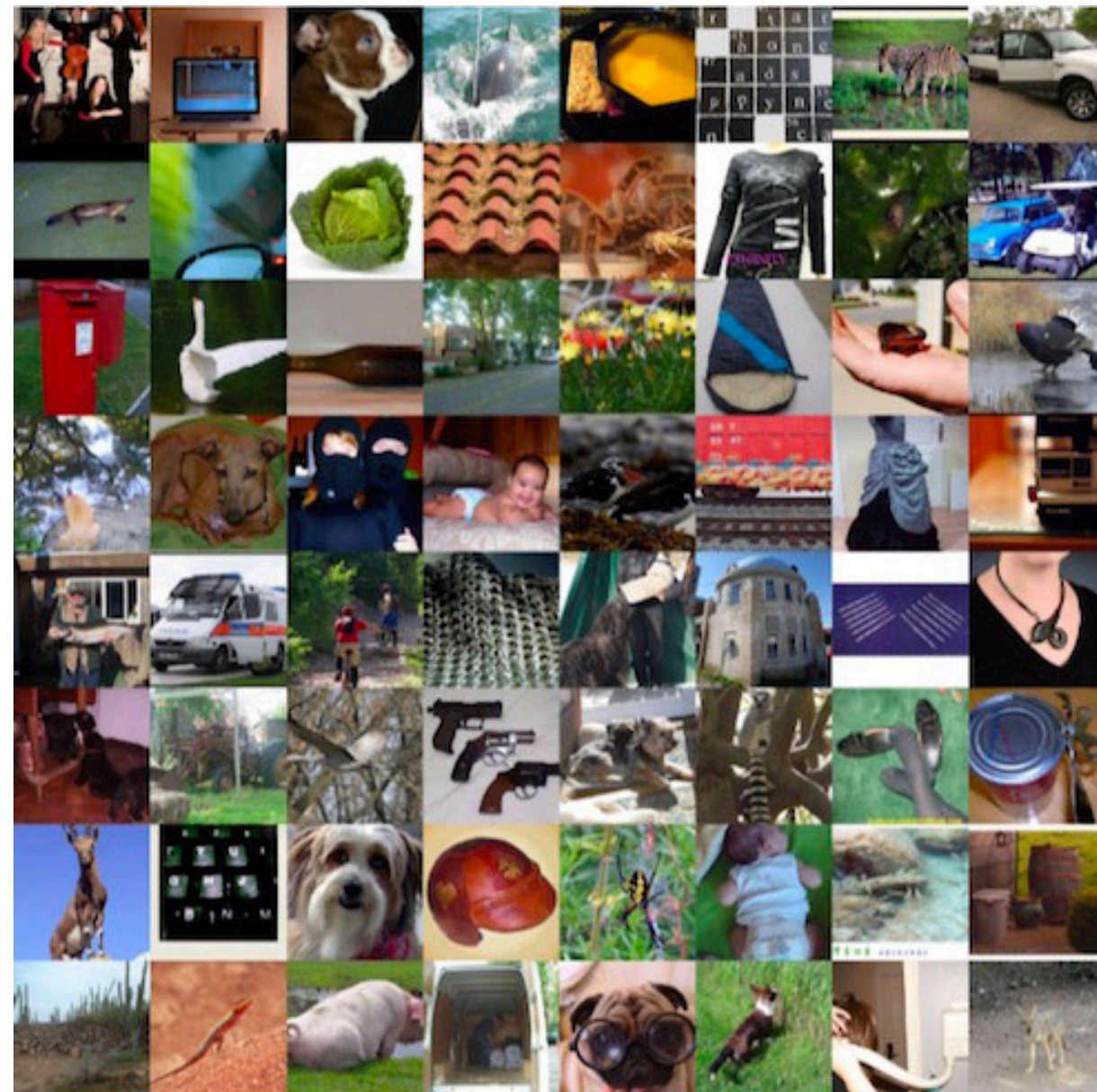
- ▶ Allows **frequentists and Bayesian approaches**, without actually knowing the likelihood!

- ▶ For MCMC chains, use  $p(\theta|x) \approx r(x, \theta)\pi(\theta)$

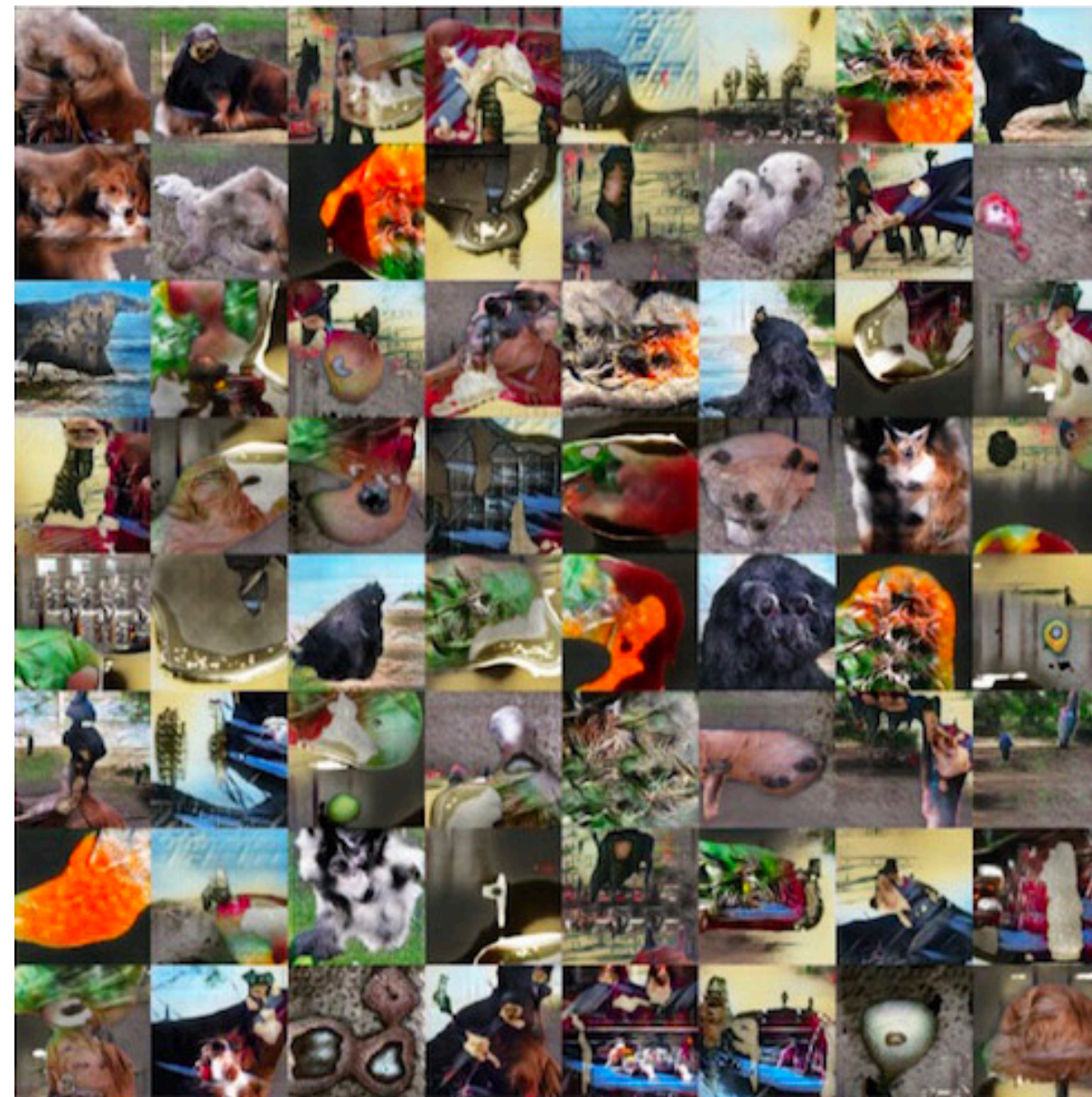
# Generative models

## History of models

Real images (ImageNet)

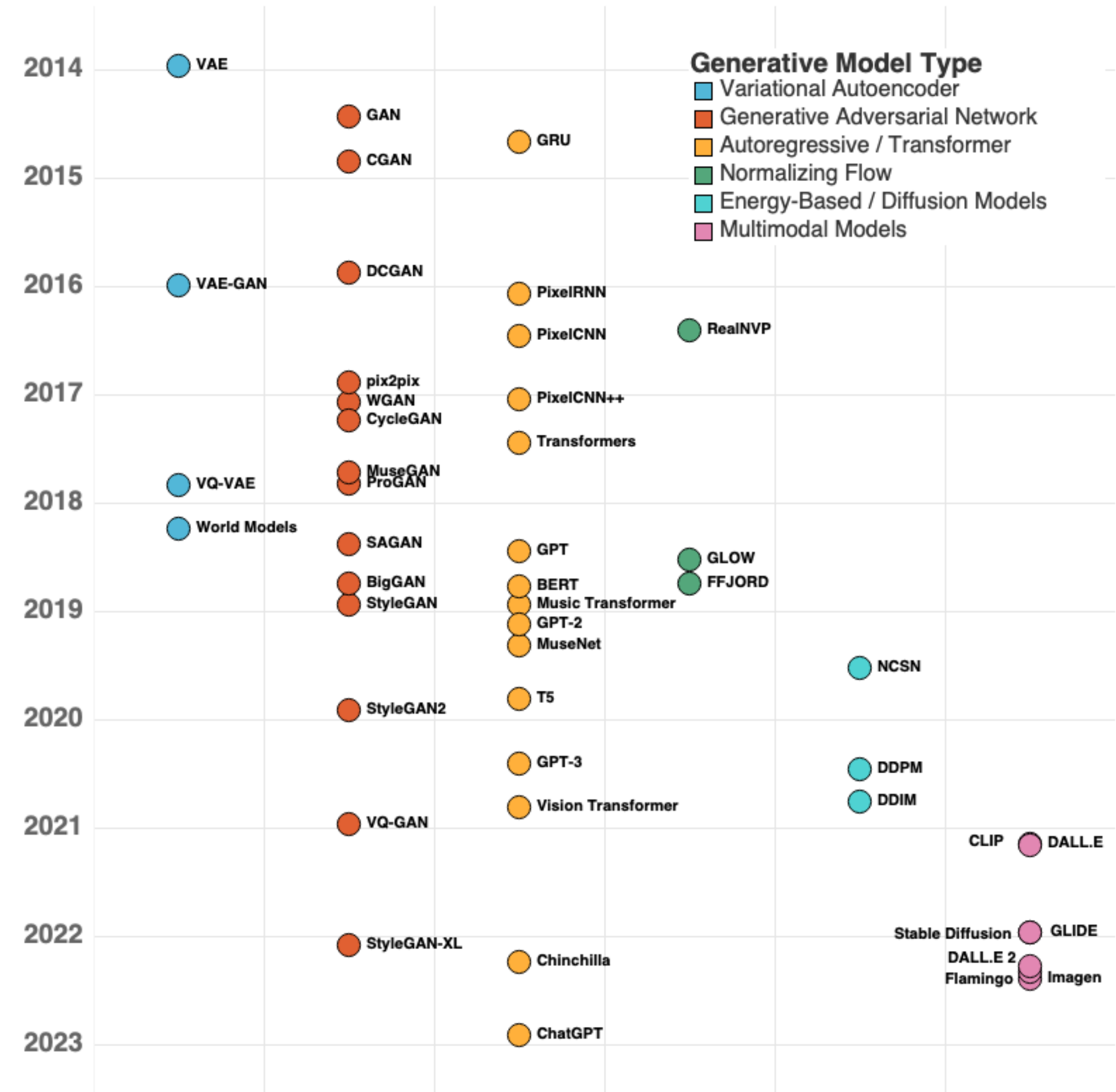


Generated images



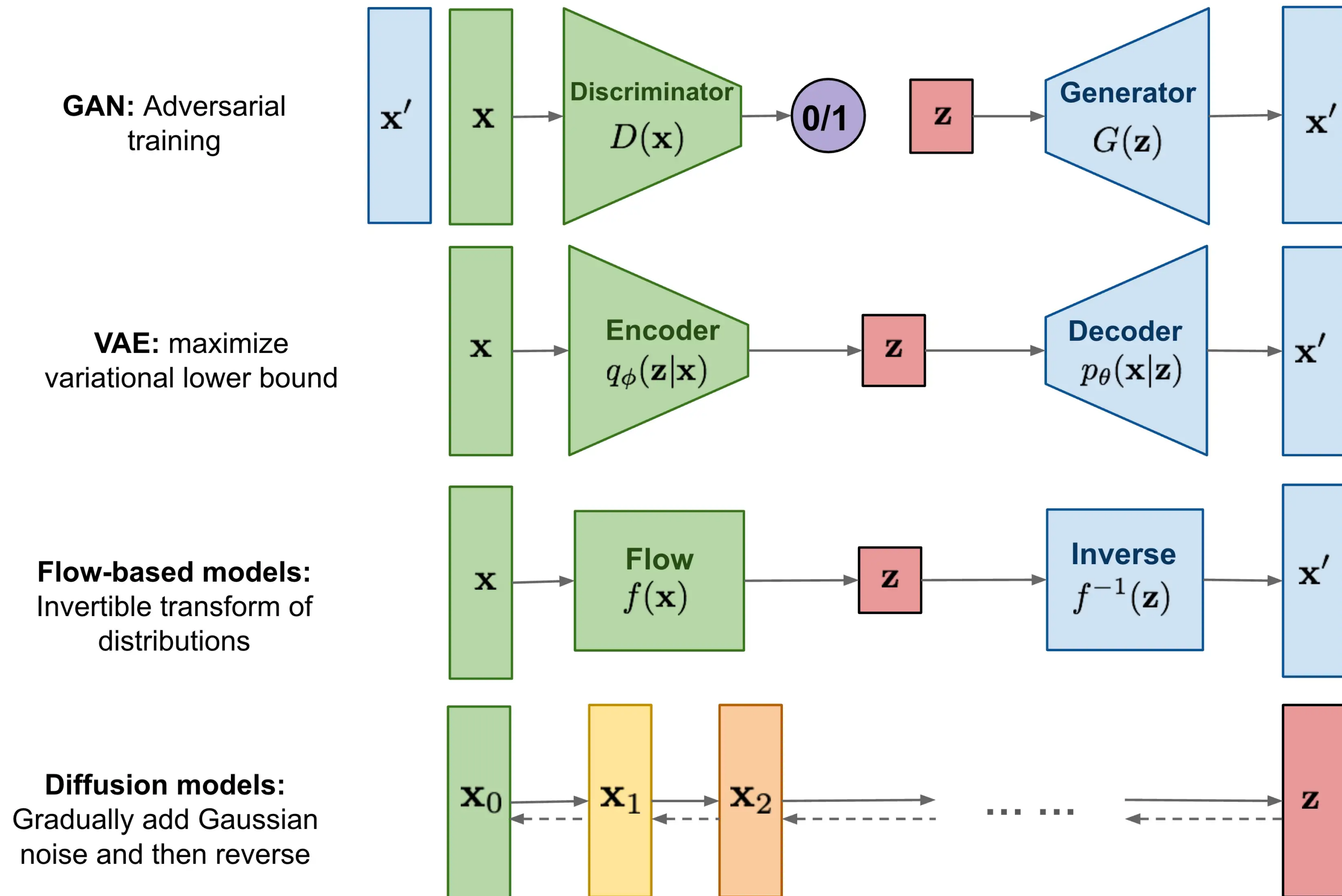
[openai.com](https://openai.com)

## Generative AI Timeline



David Foster

# Generative models



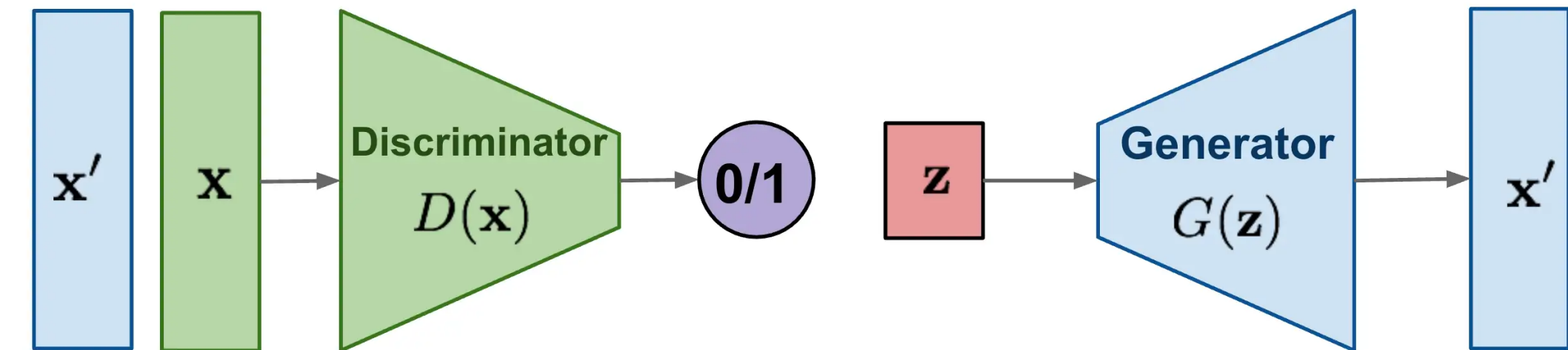
*Lilian Weng*

# Generative models

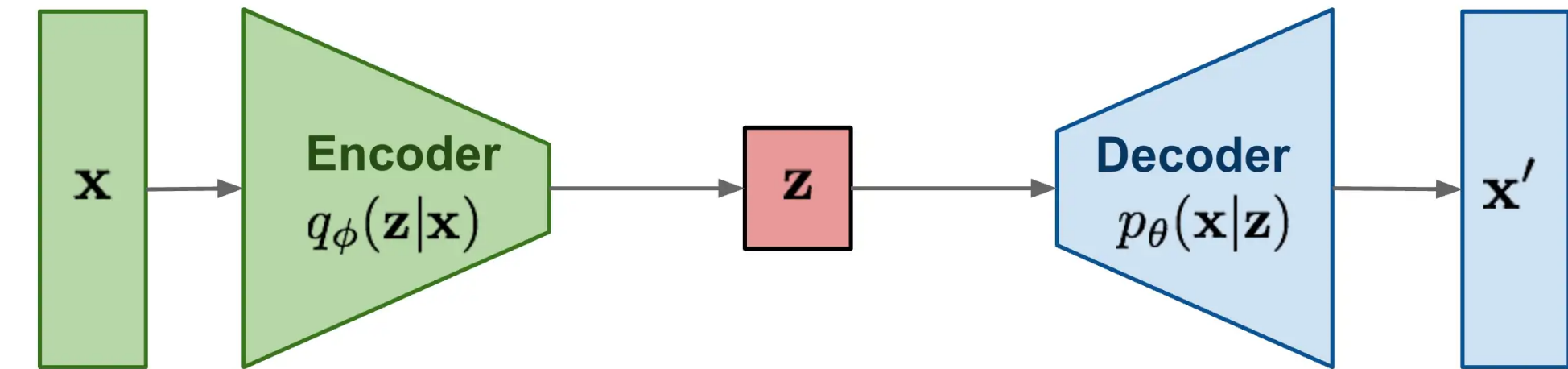
## ▶ Latent variable models

- ▶ Learn distribution of data from samples
- ▶ Data mapped to (meaningless) latent variables with **known distributions**

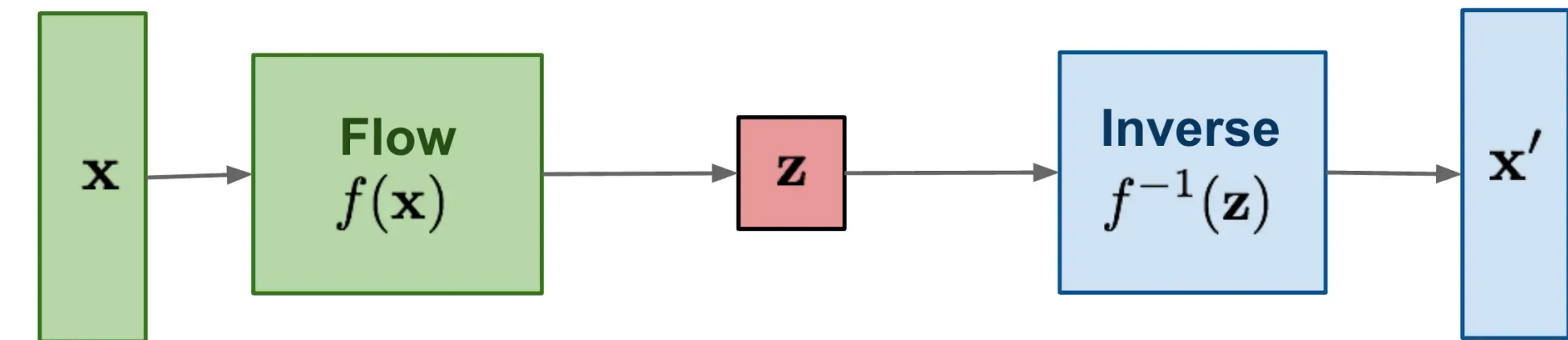
**GAN:** Adversarial training



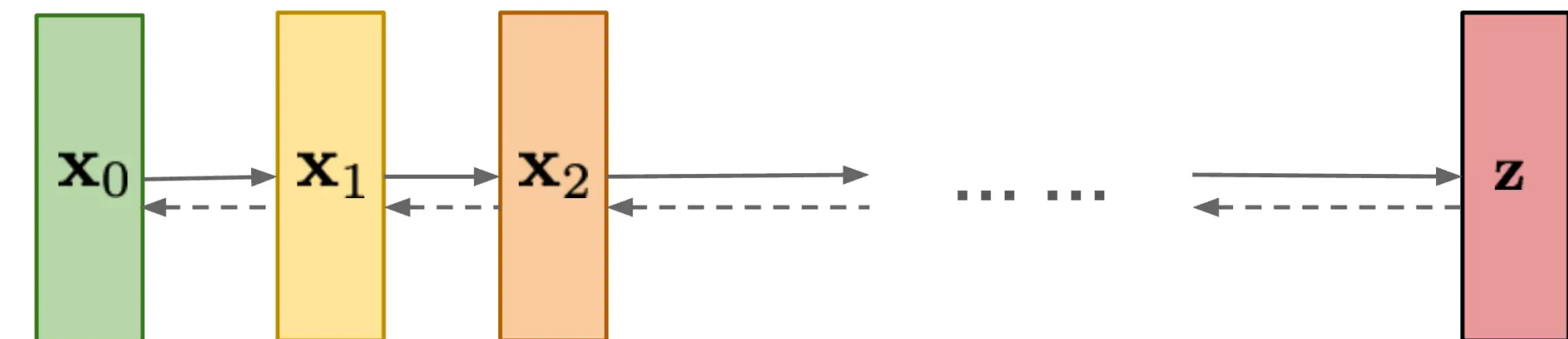
**VAE:** maximize variational lower bound



**Flow-based models:** Invertible transform of distributions



**Diffusion models:** Gradually add Gaussian noise and then reverse



*Lilian Weng*

# Generative models

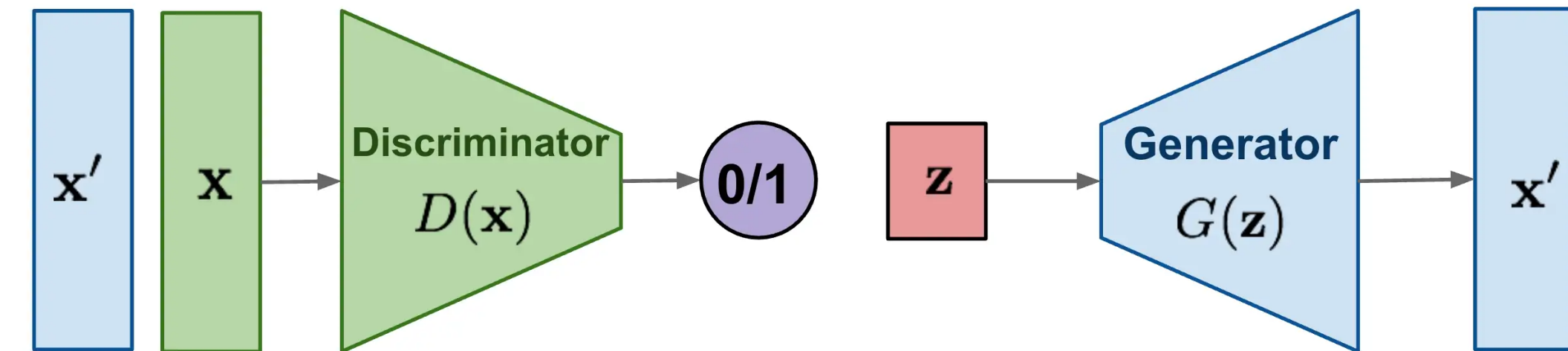
## ▶ Latent variable models

- ▶ Learn distribution of data from samples
- ▶ Data mapped to (meaningless) latent variables with **known distributions**

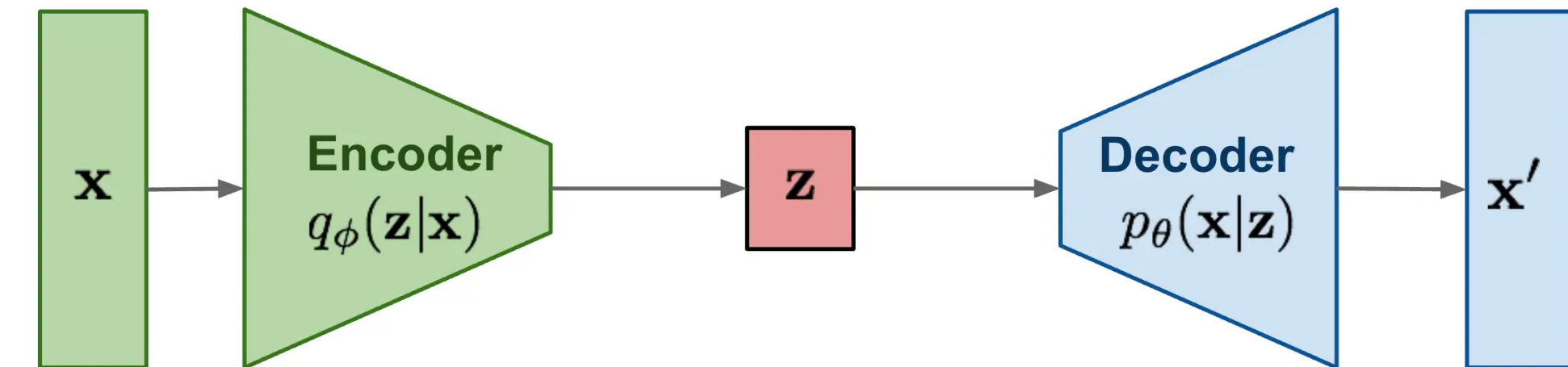
## ▶ Functionalities

- ▶ **Sampling**, ie generate new data
- ▶ Probability **density evaluation** 🙌

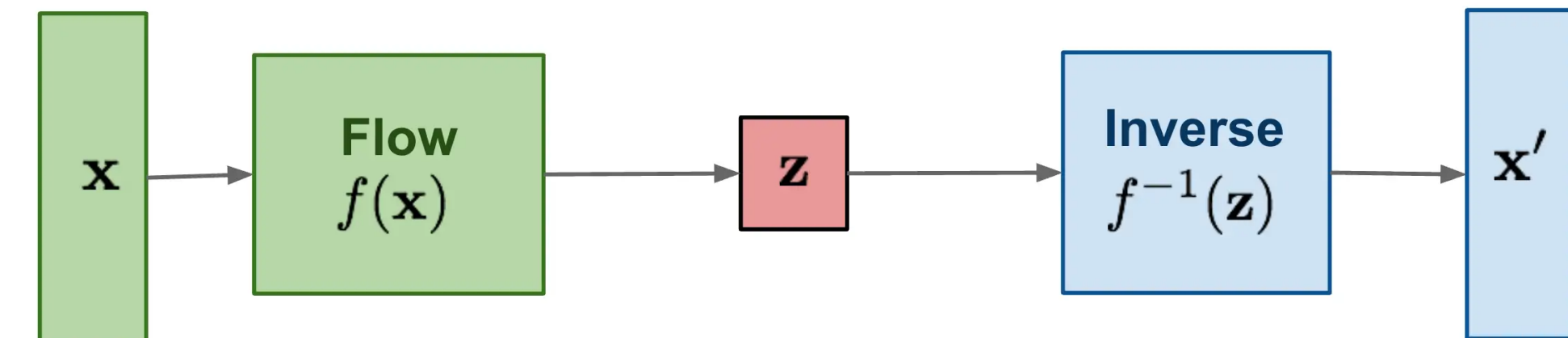
**GAN:** Adversarial training



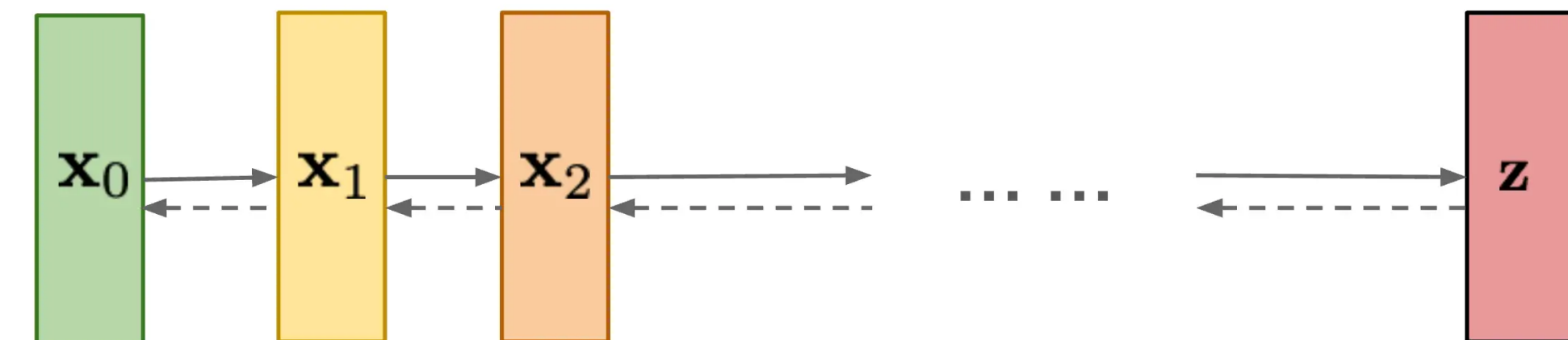
**VAE:** maximize variational lower bound



**Flow-based models:** Invertible transform of distributions



**Diffusion models:** Gradually add Gaussian noise and then reverse



*Lilian Weng*

# Generative models

## ▶ Latent variable models

- ▶ Learn distribution of data from samples
- ▶ Data mapped to (meaningless) latent variables with **known distributions**

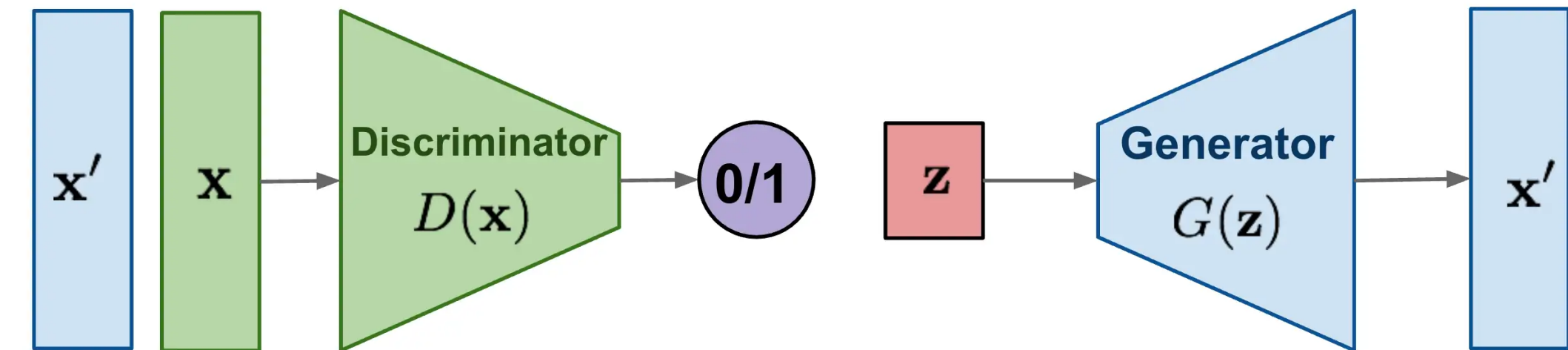
## ▶ Functionalities

- ▶ **Sampling**, ie generate new data
- ▶ Probability **density evaluation** 🙌

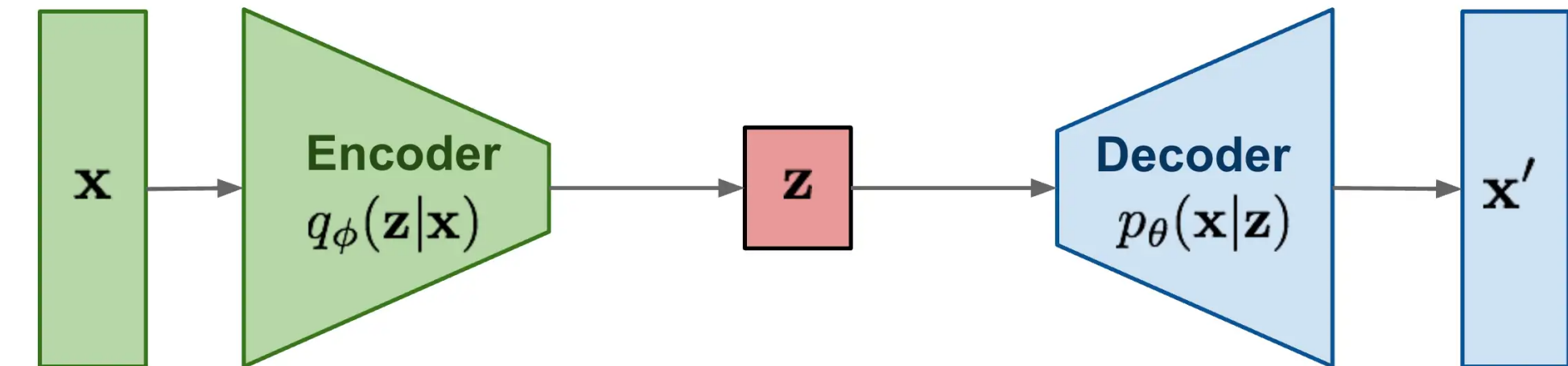
## ▶ Pros and cons

- ▶ Sampling/evaluation speed
- ▶ Sample quality
- ▶ Mode coverage

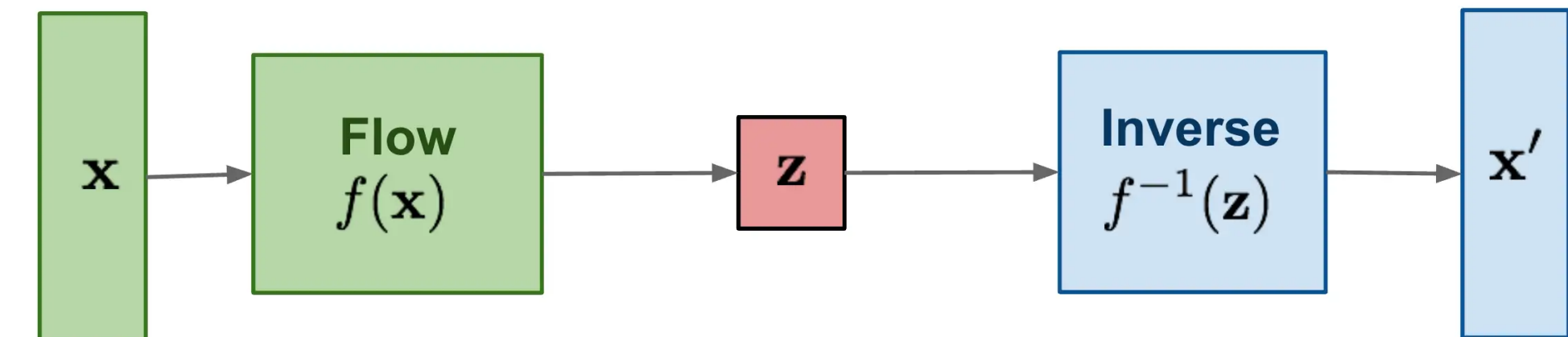
**GAN:** Adversarial training



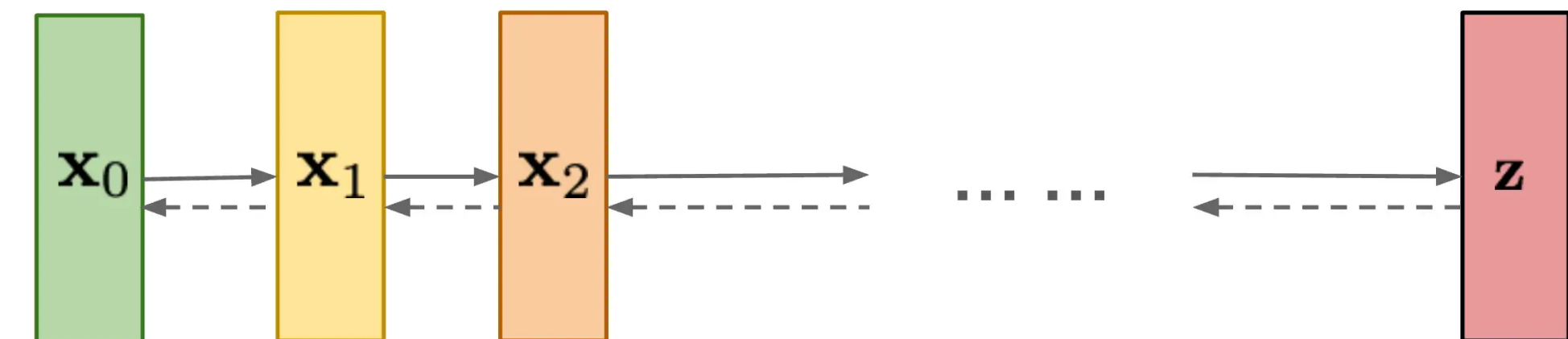
**VAE:** maximize variational lower bound



**Flow-based models:** Invertible transform of distributions



**Diffusion models:** Gradually add Gaussian noise and then reverse



*Lilian Weng*

# Normalizing flows

## Principle

1. Given training data  $x \sim p_x(x)$ ,

distribution we are  
trying to *learn*



# Normalizing flows

## Principle

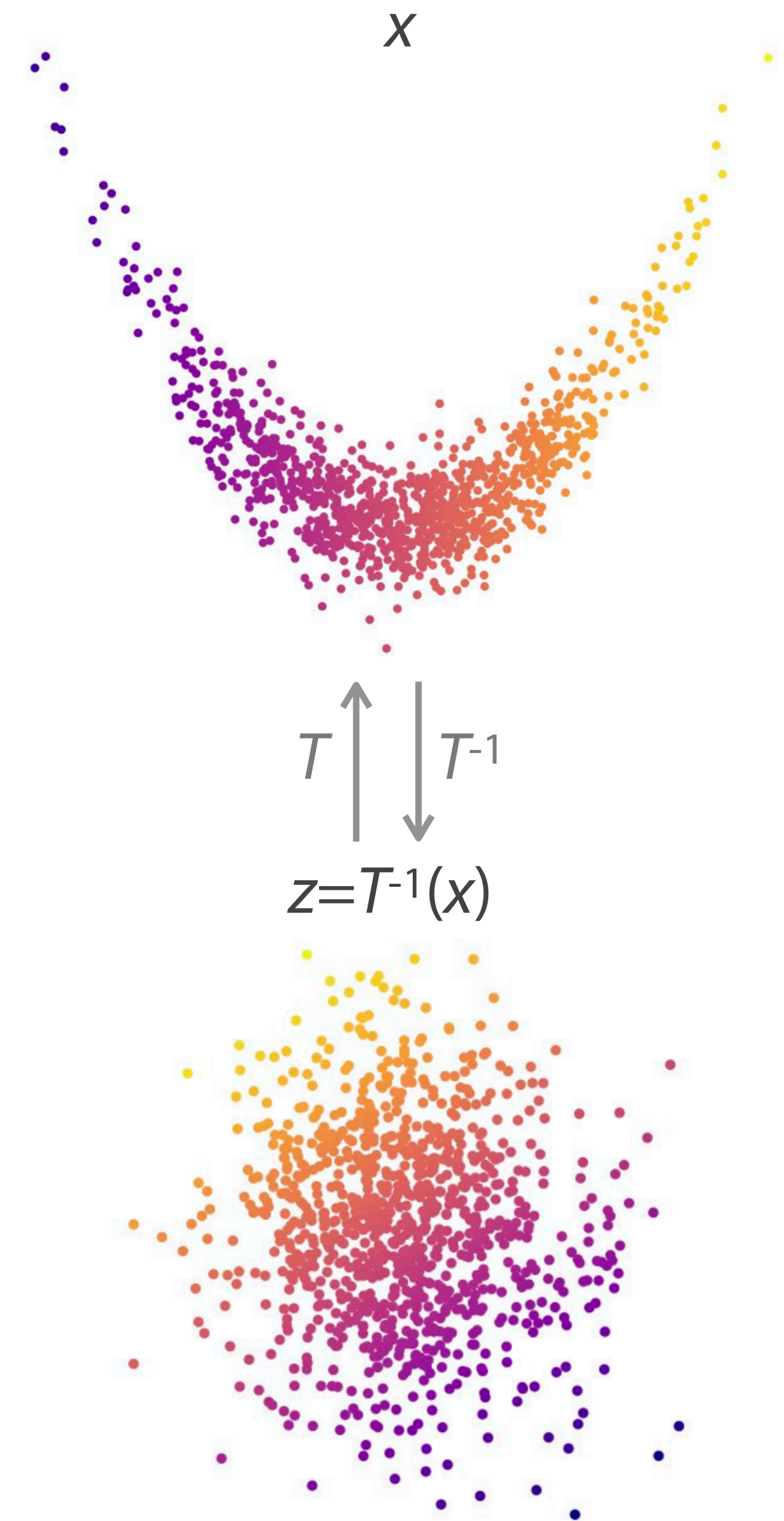
1. Given training data  $x \sim p_x(x)$ ,
  2. Map data  $x$  to variable  $z = T^{-1}(x)$  with **invertible mapping  $T$** , parametrized by neural networks
- distribution we are trying to learn



# Normalizing flows

## Principle

1. Given training data  $x \sim p_x(x)$ ,
  2. Map data  $x$  to variable  $z = T^{-1}(x)$  with **invertible mapping  $T$** , parametrized by neural networks
- distribution we are trying to learn

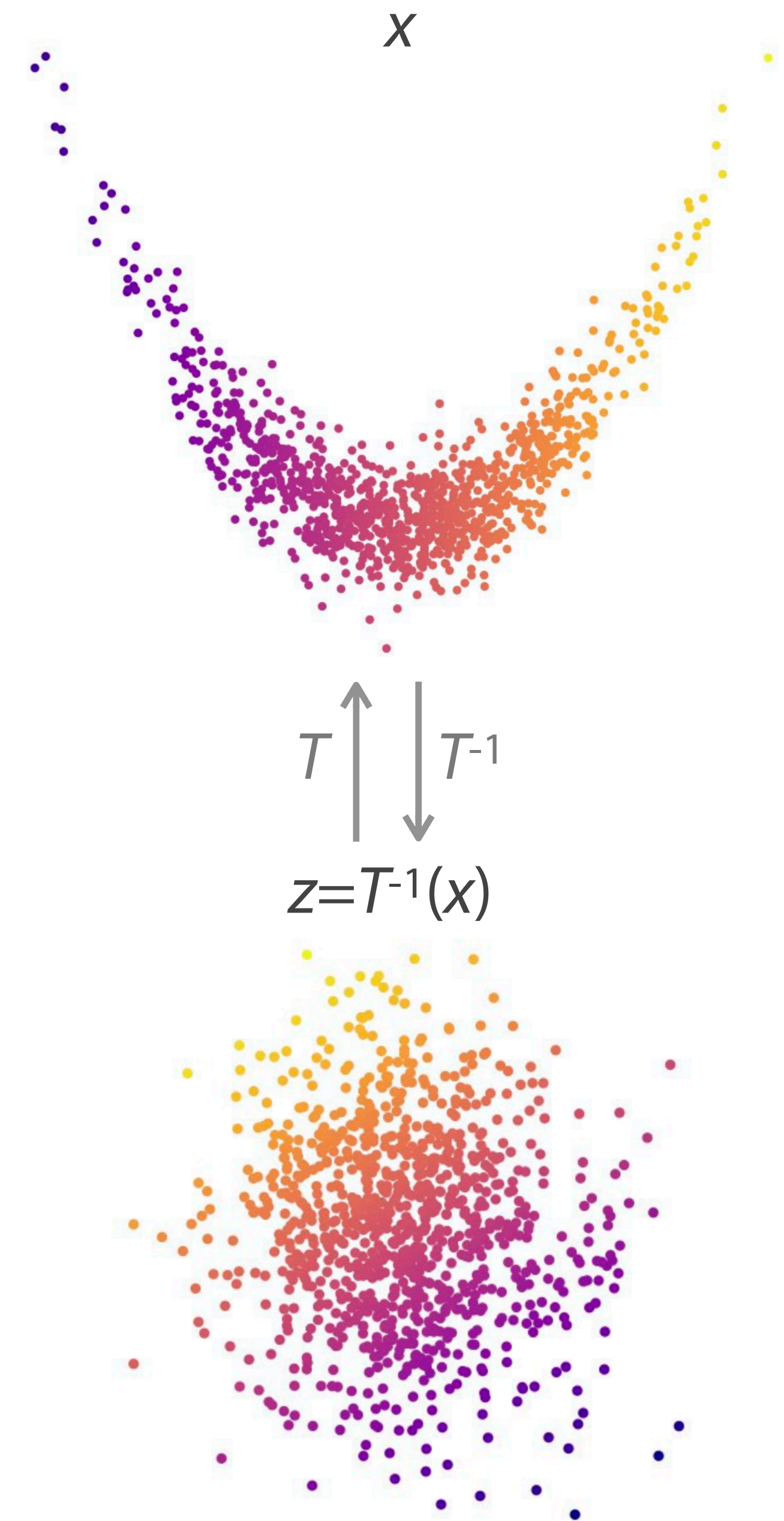


# Normalizing flows

## Principle

1. Given training data  $x \sim p_x(x)$ ,
2. Map data  $x$  to variable  $z = T^{-1}(x)$  with **invertible mapping  $T$** , parametrized by neural networks
3. Train neural network to make  $z \sim p_z(z)$  unit Gaussian

distribution we are trying to learn

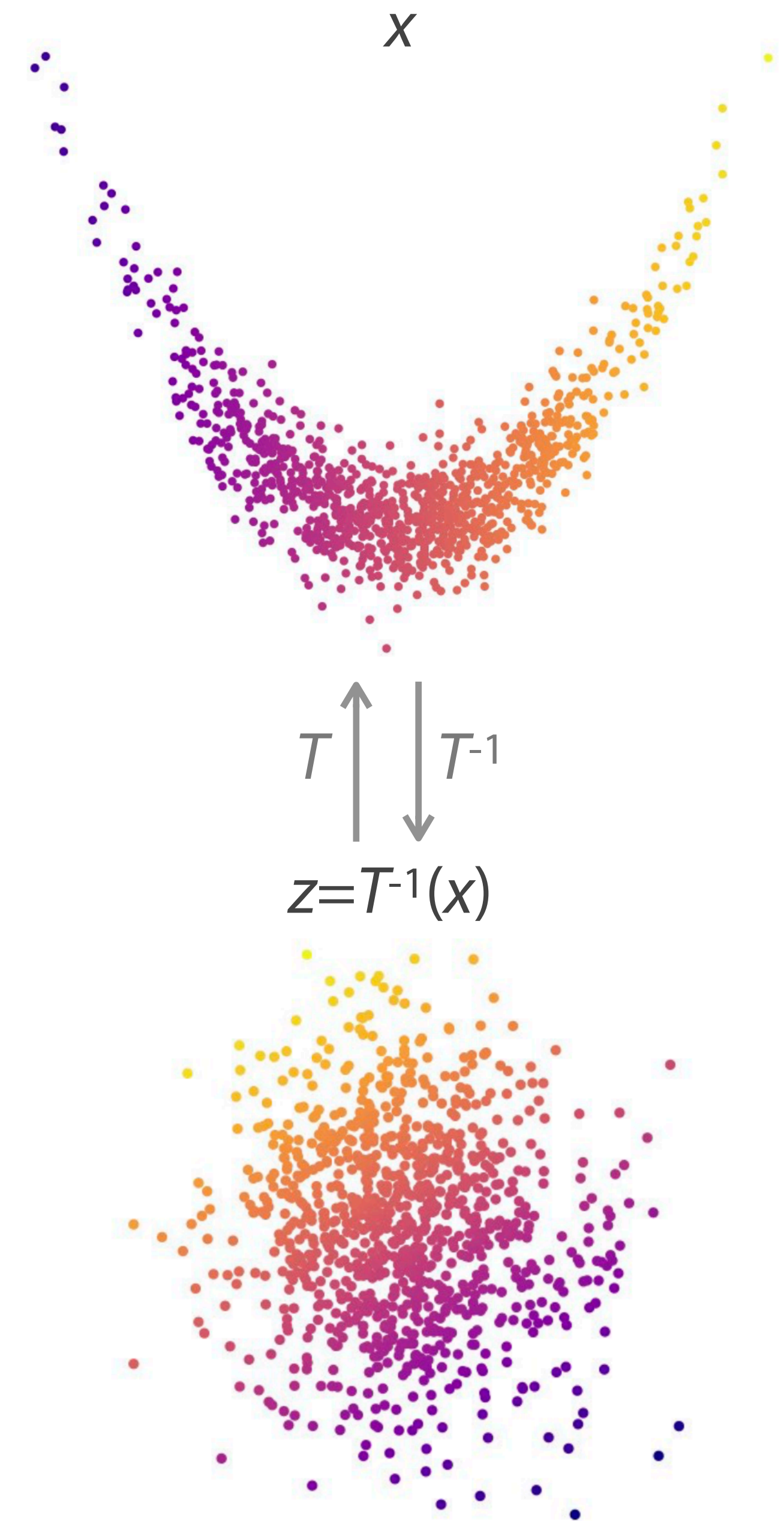


# Normalizing flows

## Principle

1. Given training data  $x \sim p_x(x)$ ,
2. Map data  $x$  to variable  $z = T^{-1}(x)$  with **invertible mapping  $T$** , parametrized by neural networks
3. Train neural network to make  $z \sim p_z(z)$  unit Gaussian
4. Then

distribution we are trying to learn



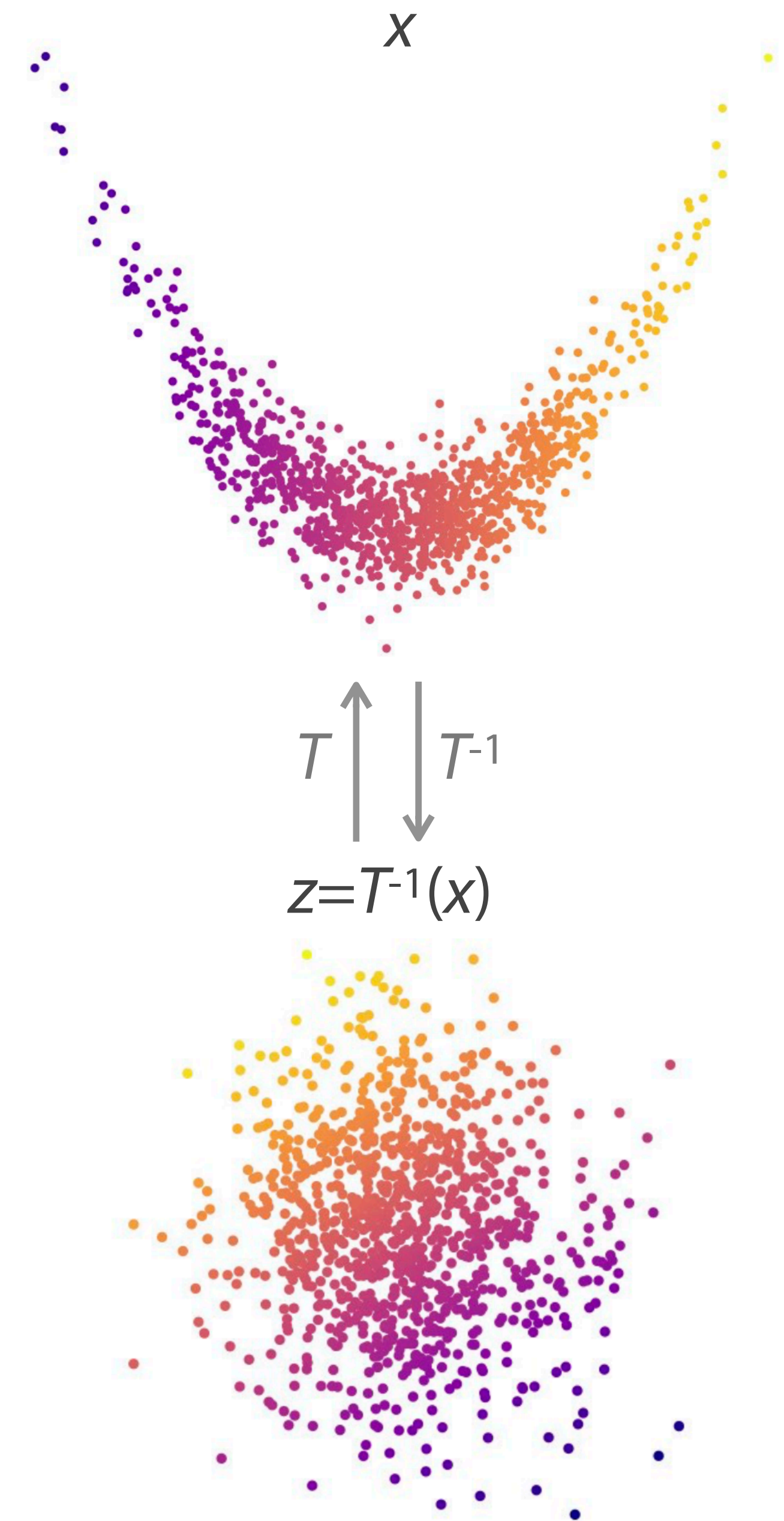
# Normalizing flows

## Principle

1. Given training data  $x \sim p_x(x)$ ,
2. Map data  $x$  to variable  $z = T^{-1}(x)$  with **invertible mapping  $T$** , parametrized by neural networks
3. Train neural network to make  $z \sim p_z(z)$  unit Gaussian
4. Then

$$p_x(x) = p_z(z) |\nabla_x T(x)|^{-1}$$

distribution we are trying to learn



# Normalizing flows

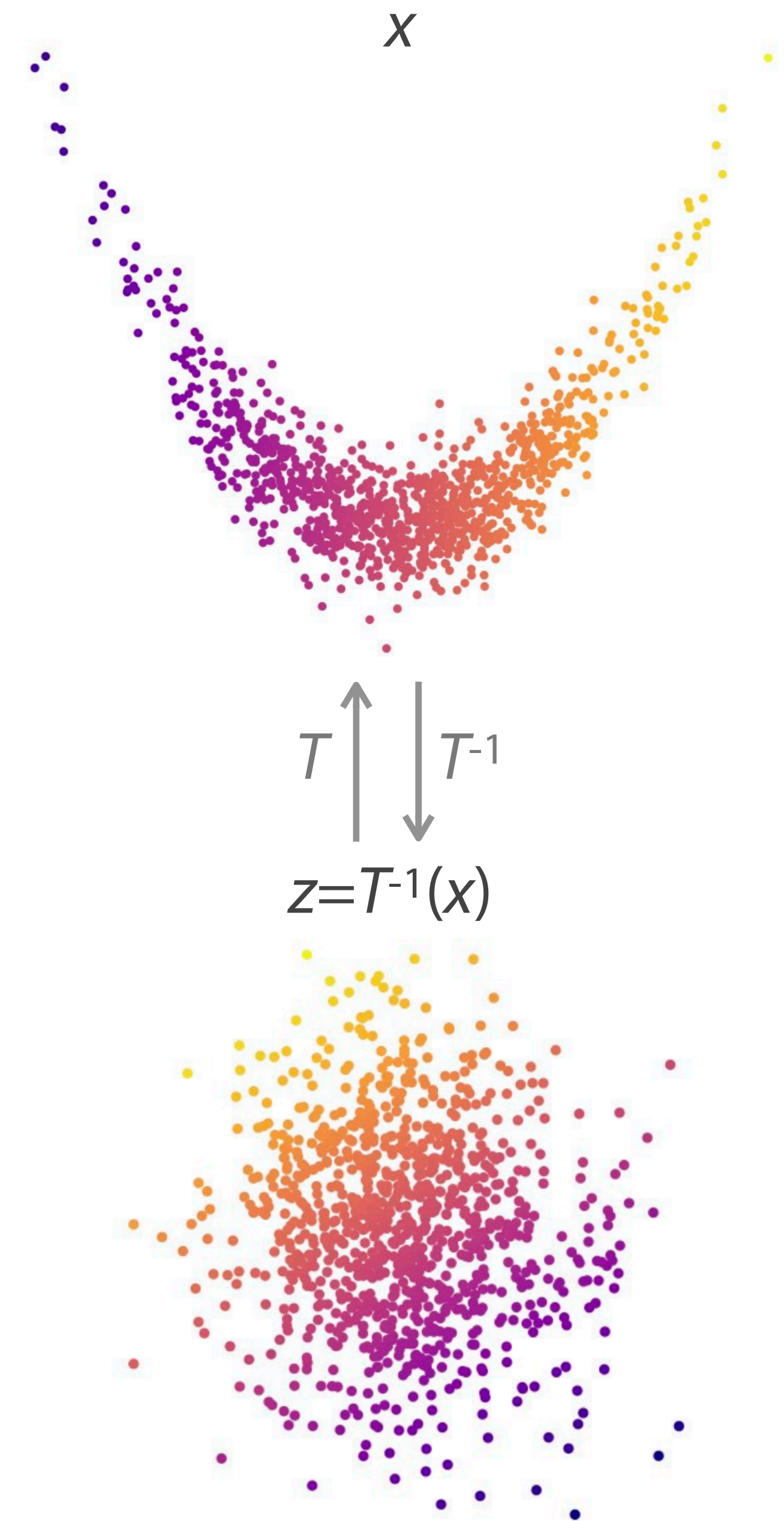
## Principle

1. Given training data  $x \sim p_x(x)$ ,
2. Map data  $x$  to variable  $z = T^{-1}(x)$  with **invertible mapping**  $T$ , parametrized by neural networks
3. Train neural network to make  $z \sim p_z(z)$  unit Gaussian
4. Then

$$p_x(x) = p_z(z) |\nabla_x T(x)|^{-1}$$

- Fast **evaluation** of  $p_x$

distribution we are trying to learn



# Normalizing flows

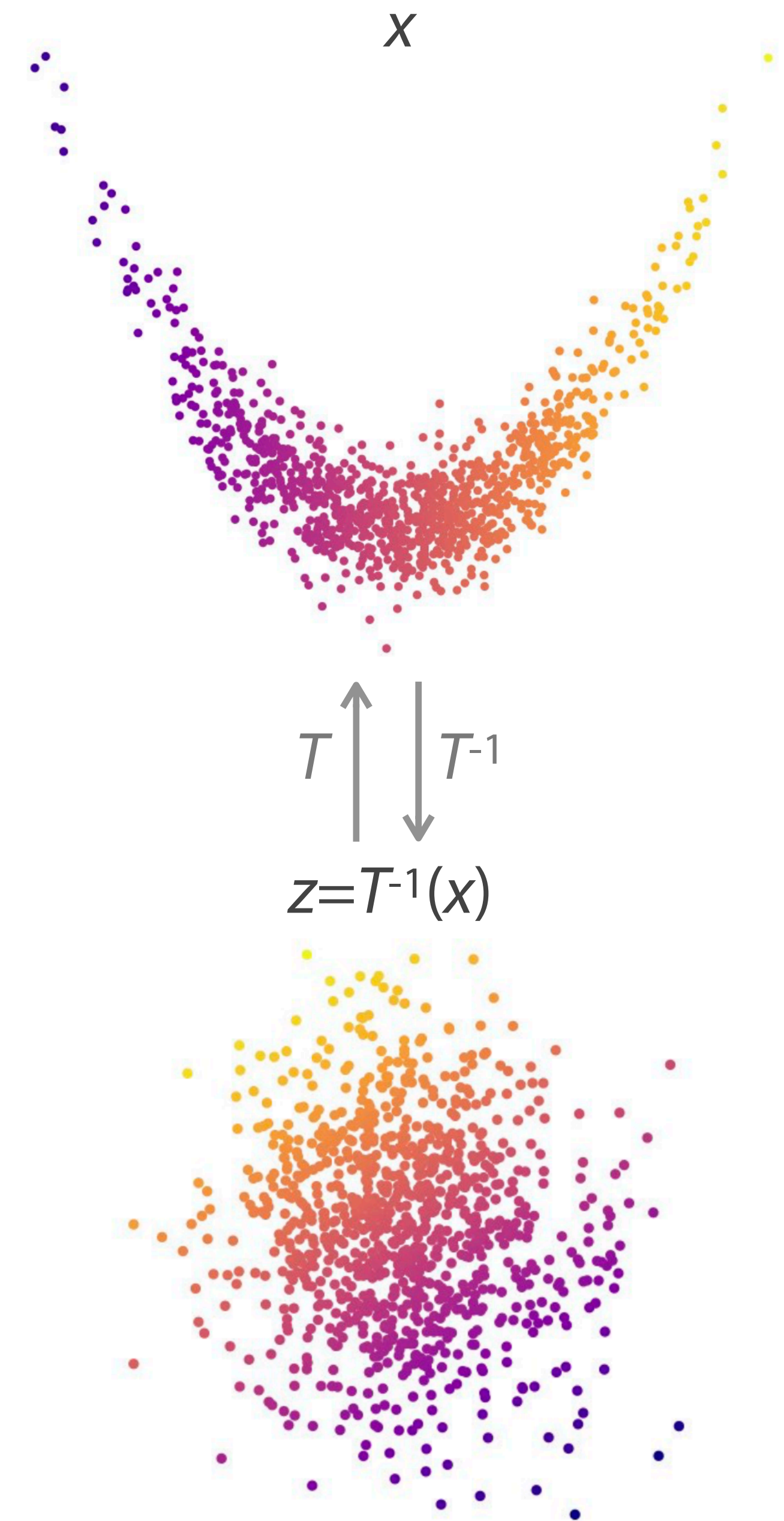
## Principle

1. Given training data  $x \sim p_x(x)$ ,
2. Map data  $x$  to variable  $z = T^{-1}(x)$  with **invertible mapping**  $T$ , parametrized by neural networks
3. Train neural network to make  $z \sim p_z(z)$  unit Gaussian
4. Then

$$p_x(x) = p_z(z) |\nabla_x T(x)|^{-1}$$

- Fast **evaluation** of  $p_x$
- **Sampling** of  $p_x$  : sample  $z \sim p_z(z)$ , transform to  $x = T(z)$

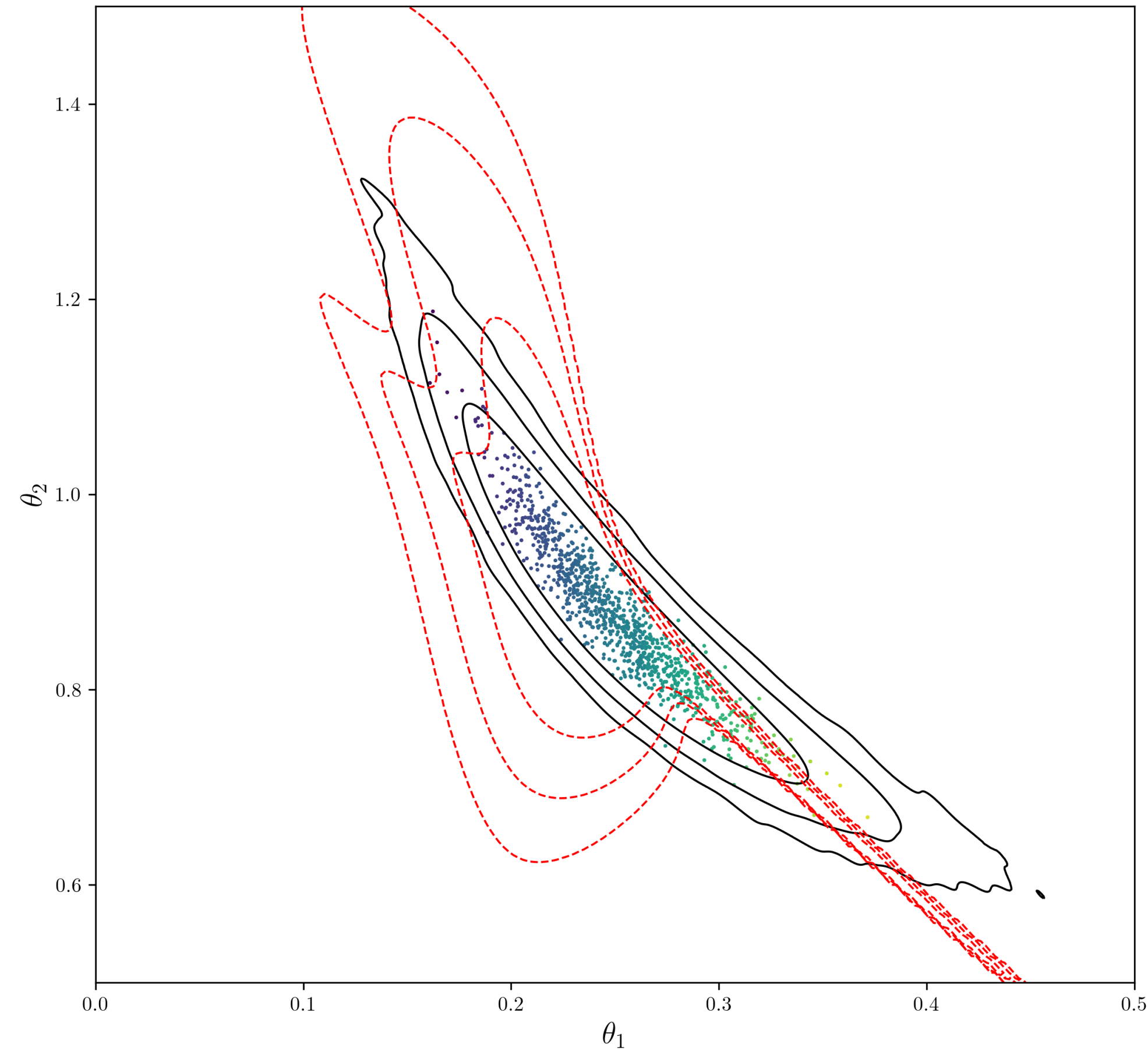
distribution we are trying to learn



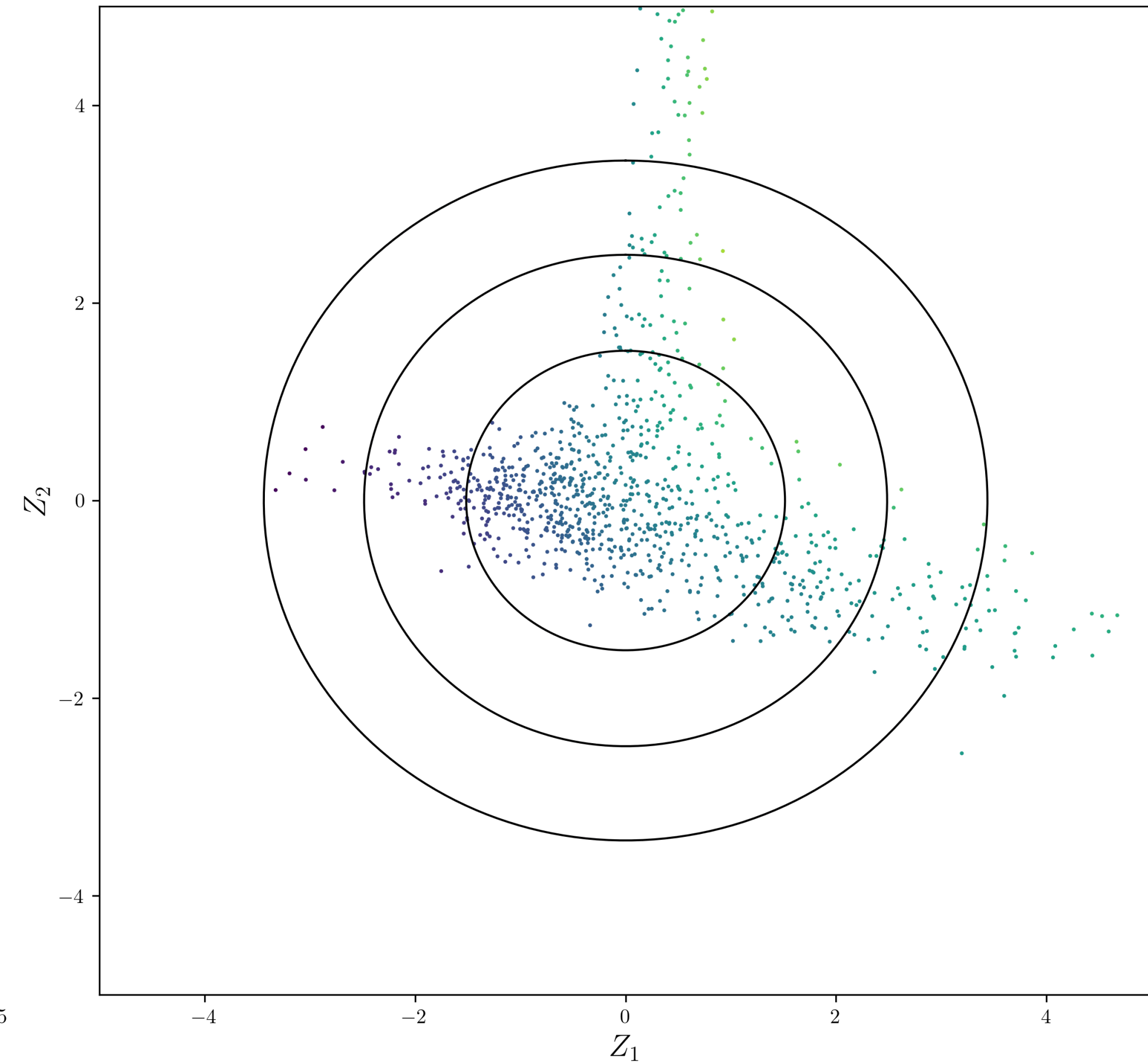
# Normalizing flows

What happens during training? 🤩

Parameter space  $p_x$



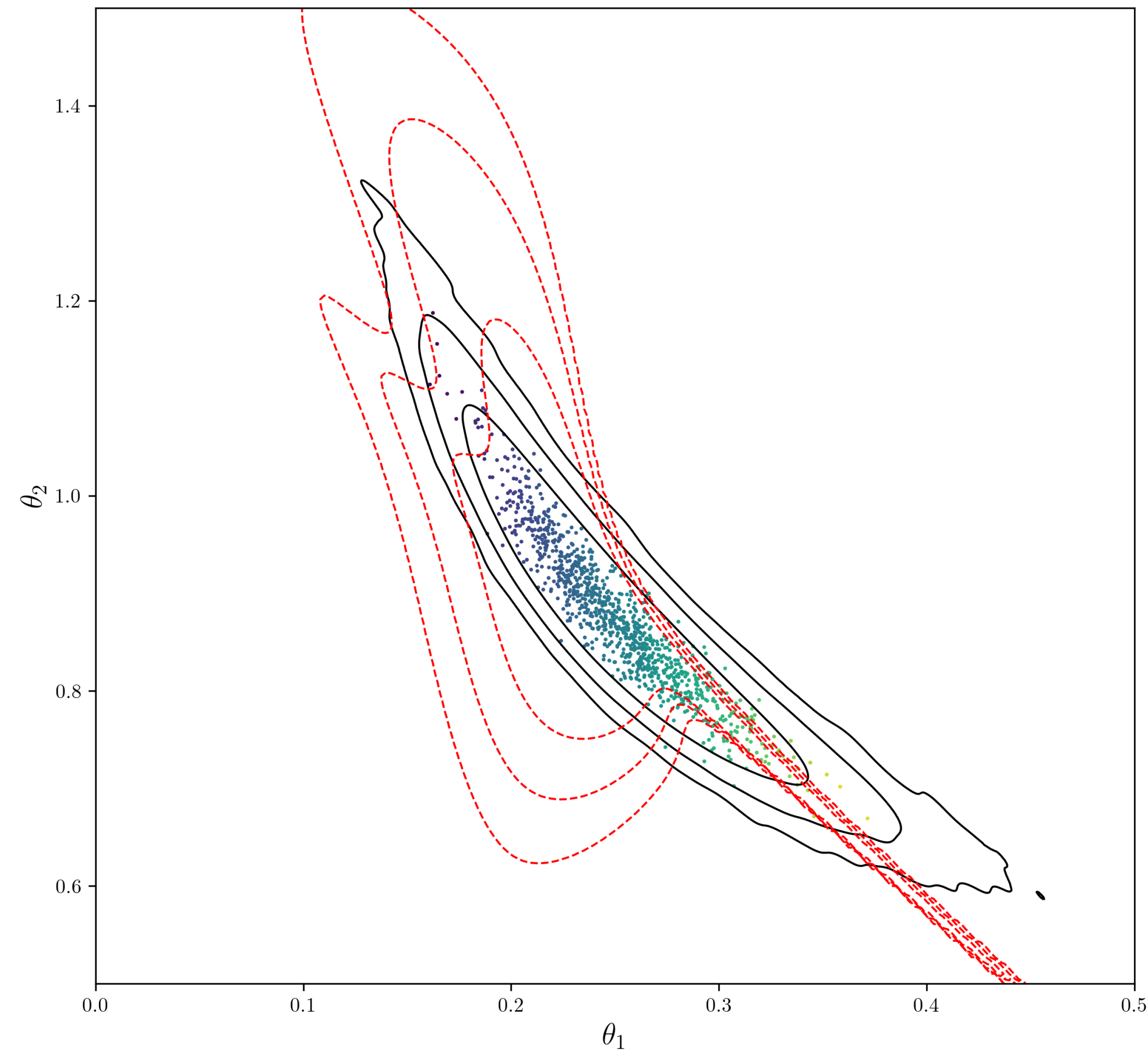
Latent space  $p_z$



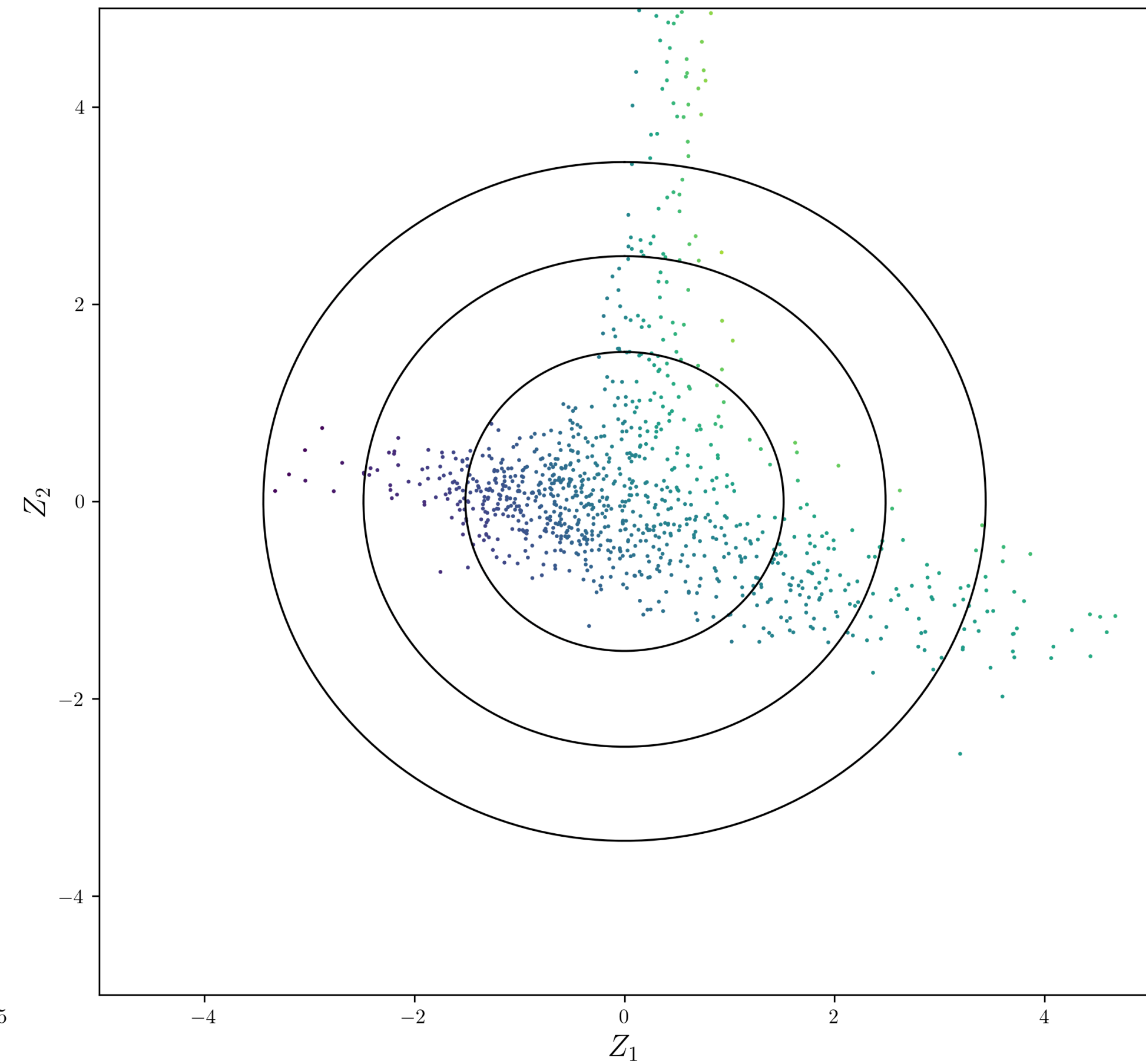
# Normalizing flows

What happens during training? 🤩

Parameter space  $p_x$



Latent space  $p_z$



# Normalizing flows

In practice

# Normalizing flows

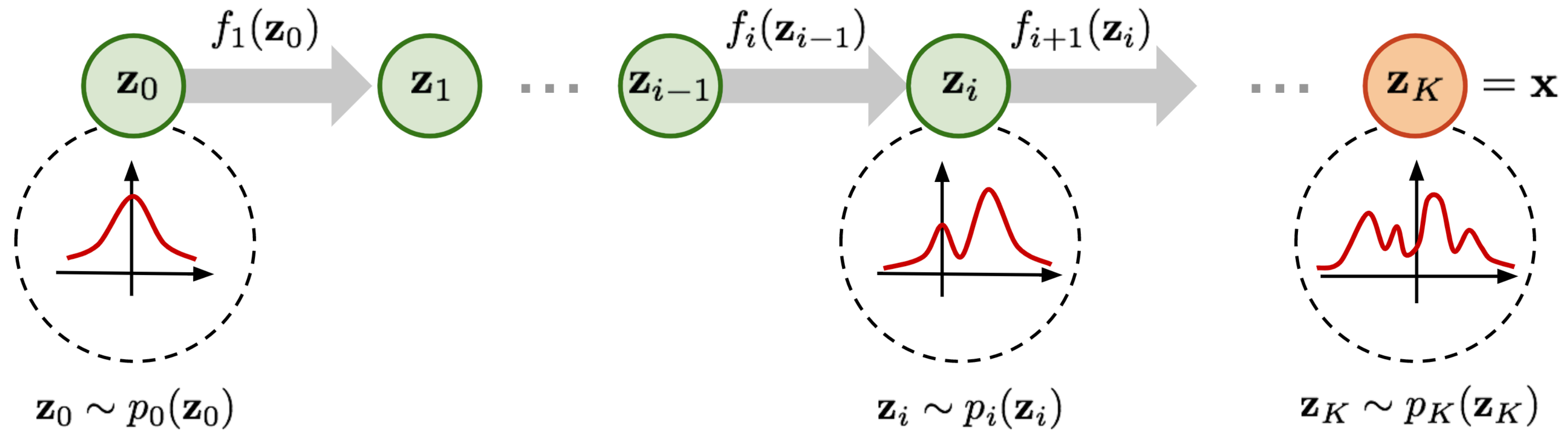
In practice

- **Stack** “simple” transformations to learn complex distributions

# Normalizing flows

In practice

- ▶ **Stack** “simple” transformations to learn complex distributions

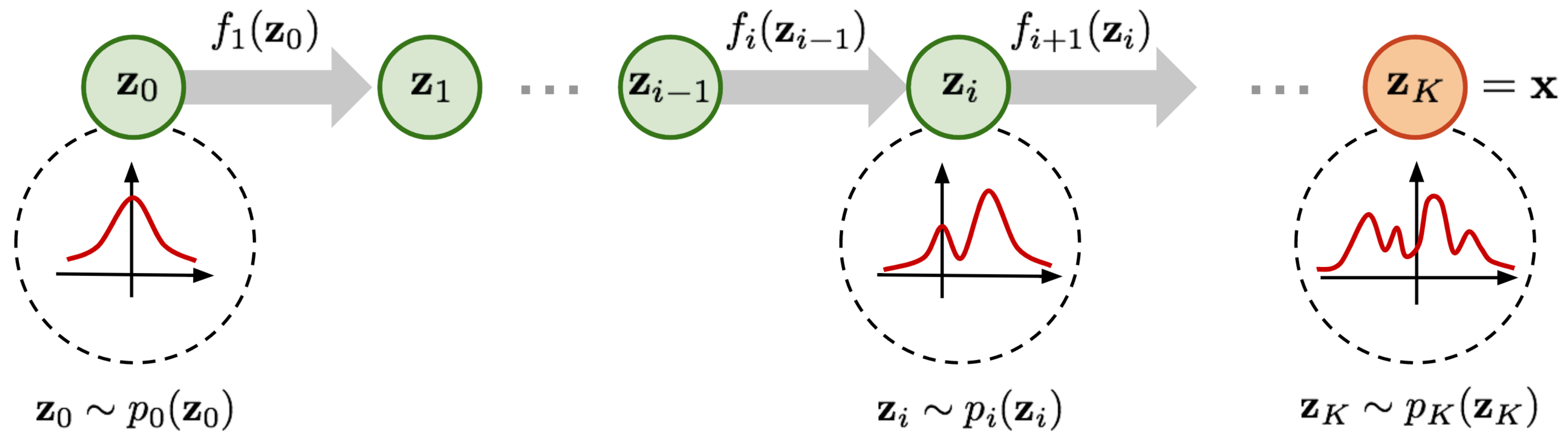


*Lilian Wang*

# Normalizing flows

In practice

- ▶ **Stack** “simple” transformations to learn complex distributions



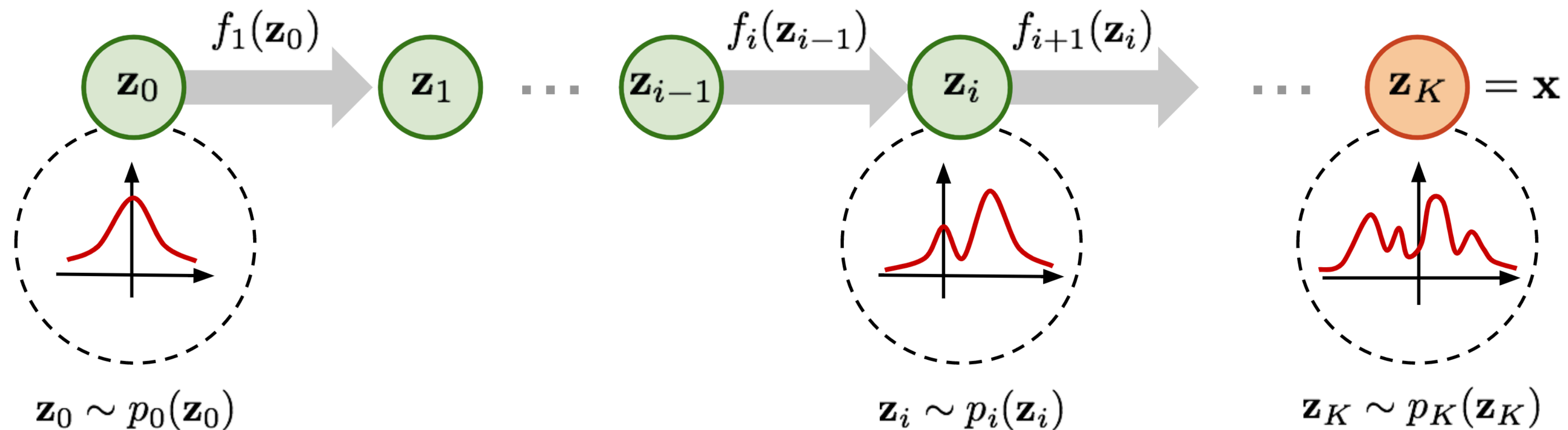
*Lilian Wang*

- ▶ NF flavors vary by parametrization of invertible mappings:  $\pm$  flexible,  $\pm$  fast

# Normalizing flows

In practice

- ▶ **Stack** “simple” transformations to learn complex distributions



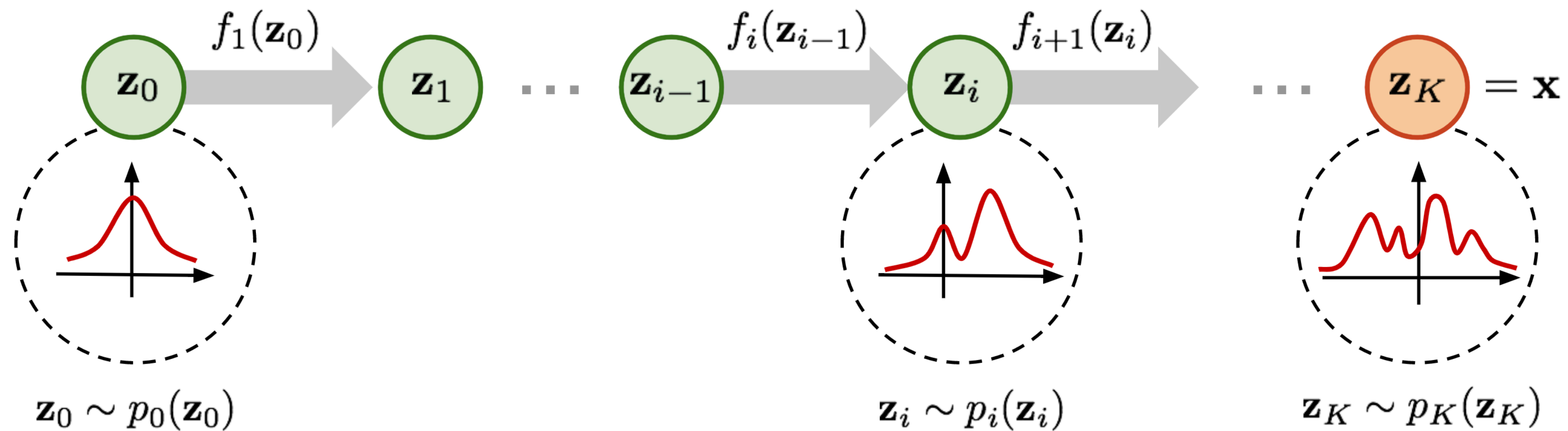
*Lilian Wang*

- ▶ NF flavors vary by parametrization of invertible mappings:  $\pm$  flexible,  $\pm$  fast
- ▶ Popular ones: Masked Autoregressive Flows, FFJORD, Glow (1x1 conv), Neural Splines

# Normalizing flows

In practice

- ▶ **Stack** “simple” transformations to learn complex distributions

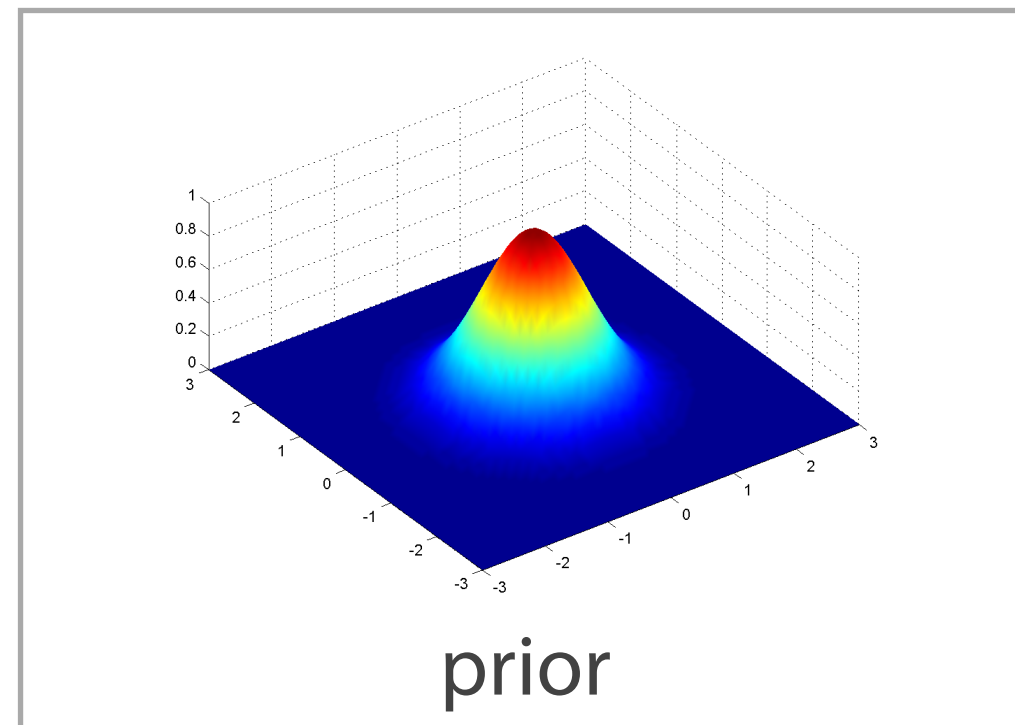


*Lilian Wang*

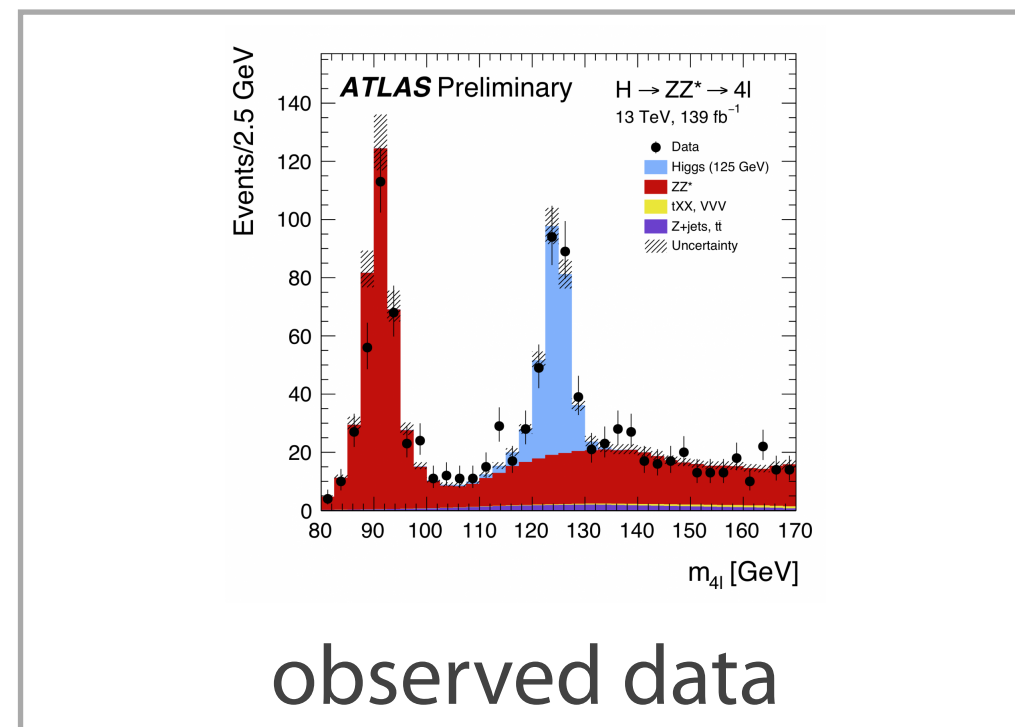
- ▶ NF flavors vary by parametrization of invertible mappings:  $\pm$  flexible,  $\pm$  fast
- ▶ Popular ones: Masked Autoregressive Flows, FFJORD, Glow (1x1 conv), Neural Splines
- ▶ **Conditional NFs**: mapping  $x=T(z|\theta)$  depends on conditioning variable  $\theta$  (just another NN input)

# SBI with cNF's

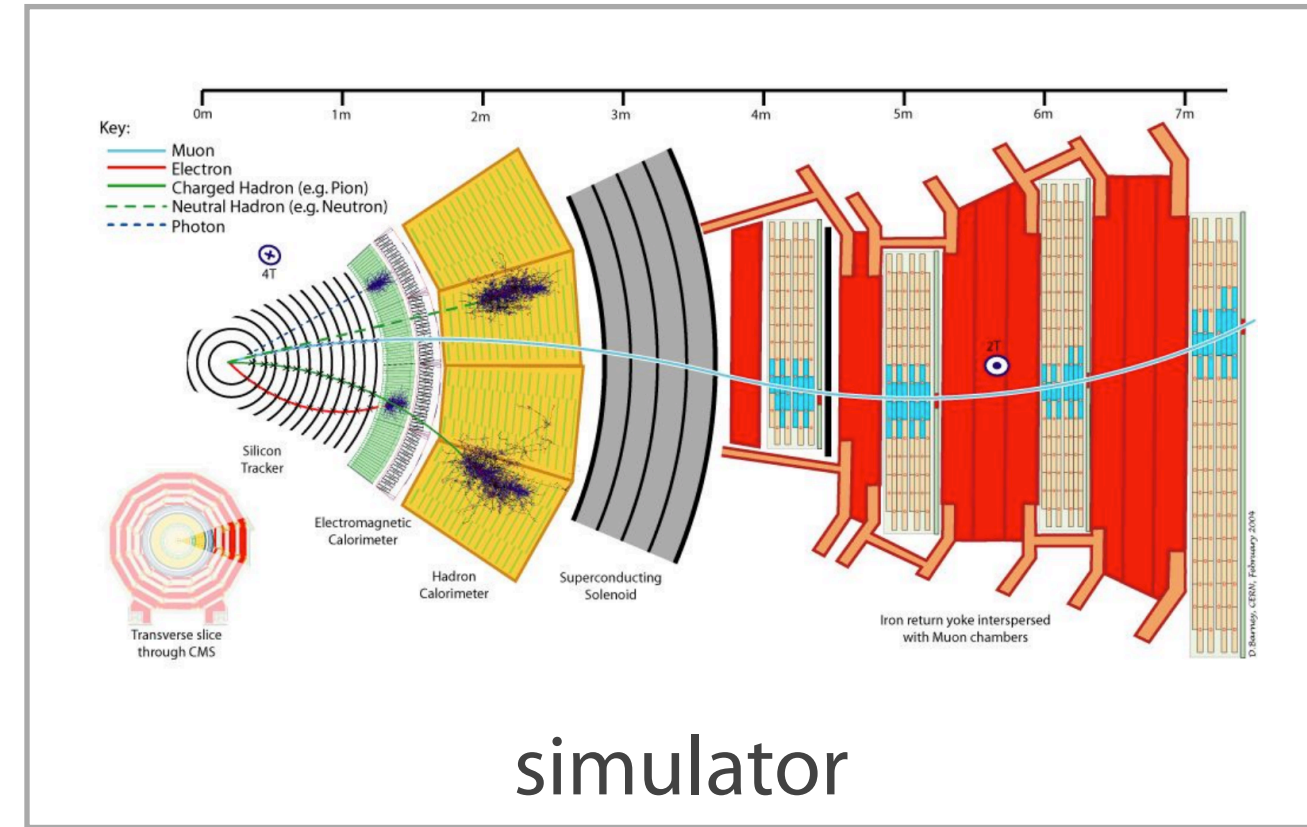
$p(\theta)$



$X_{\text{obs}}$



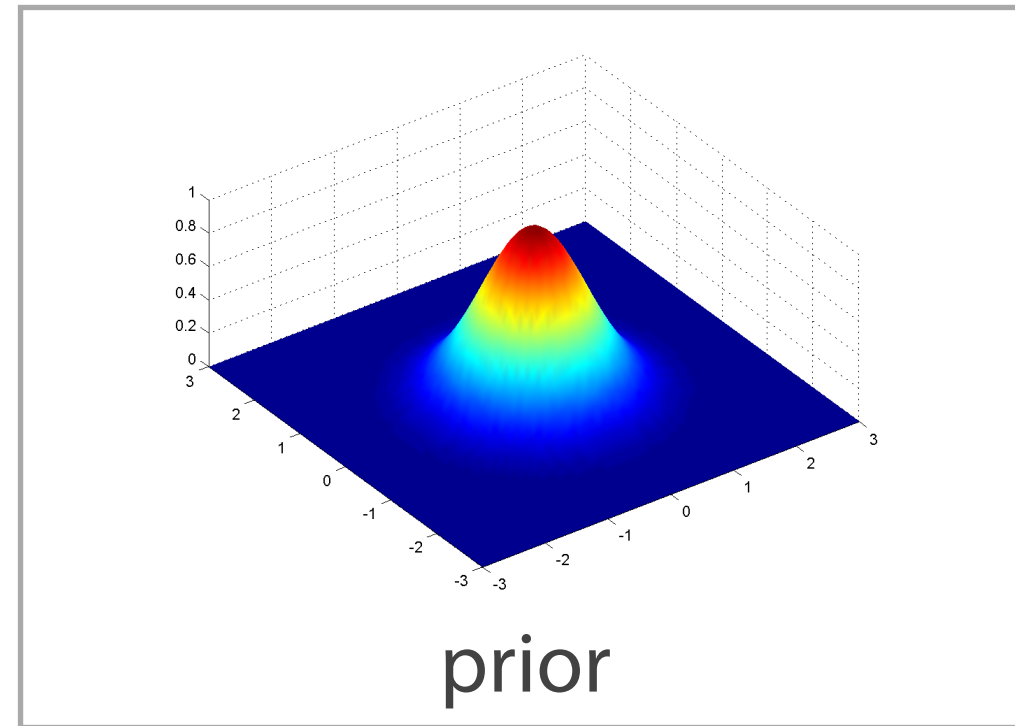
# SBI with cNF's



simulator

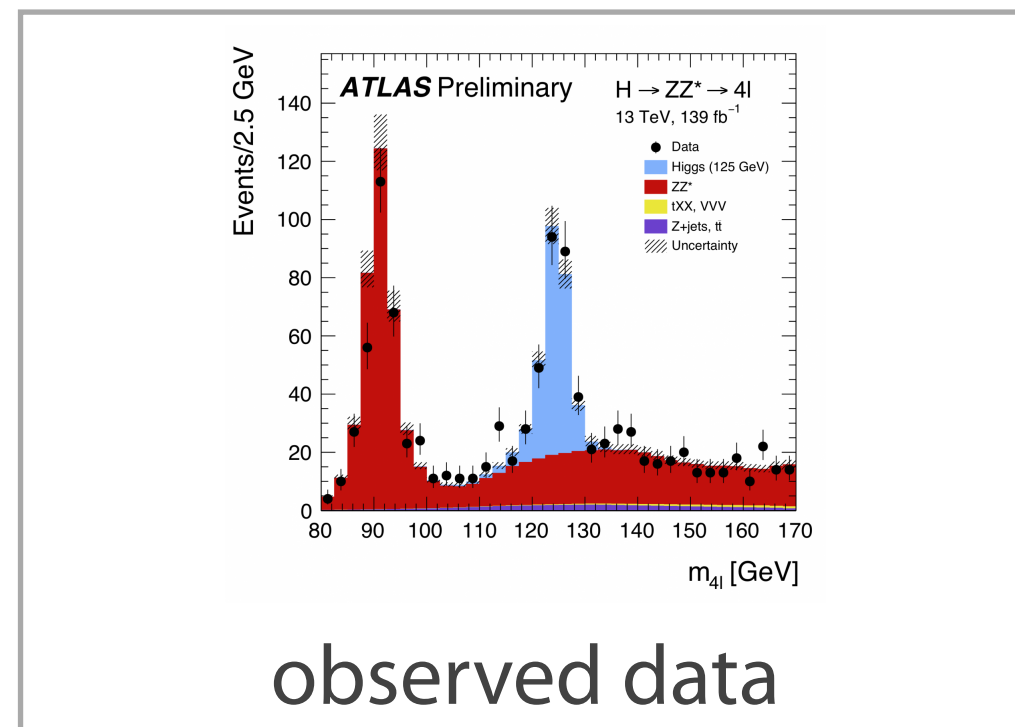
$$\theta \sim p(\theta)$$

$p(\theta)$



prior

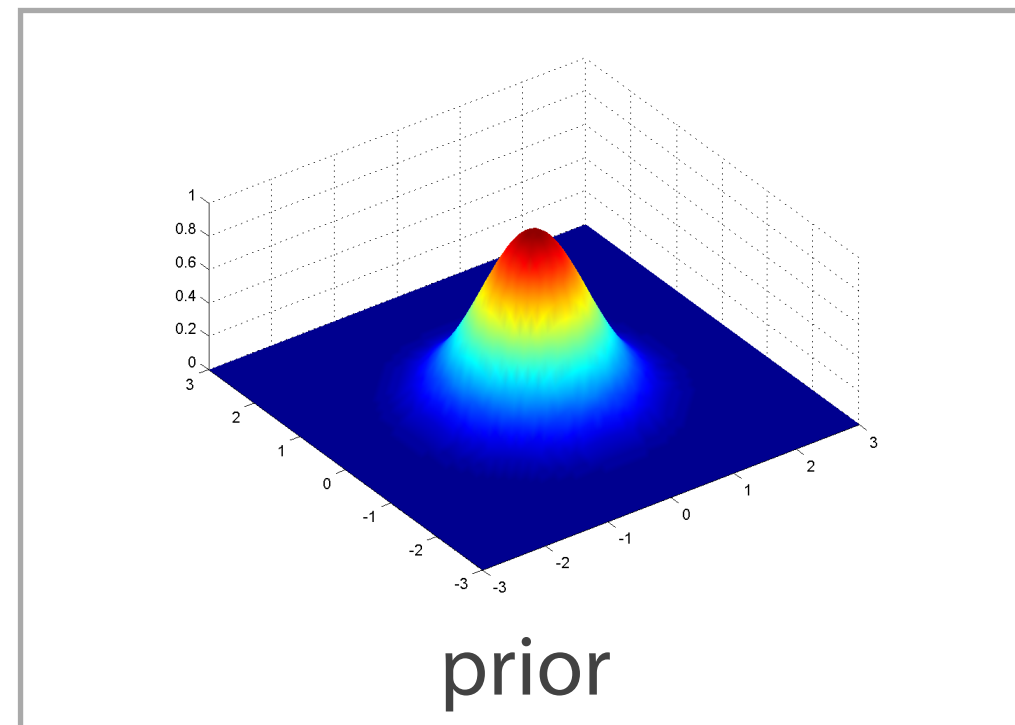
$X_{obs}$



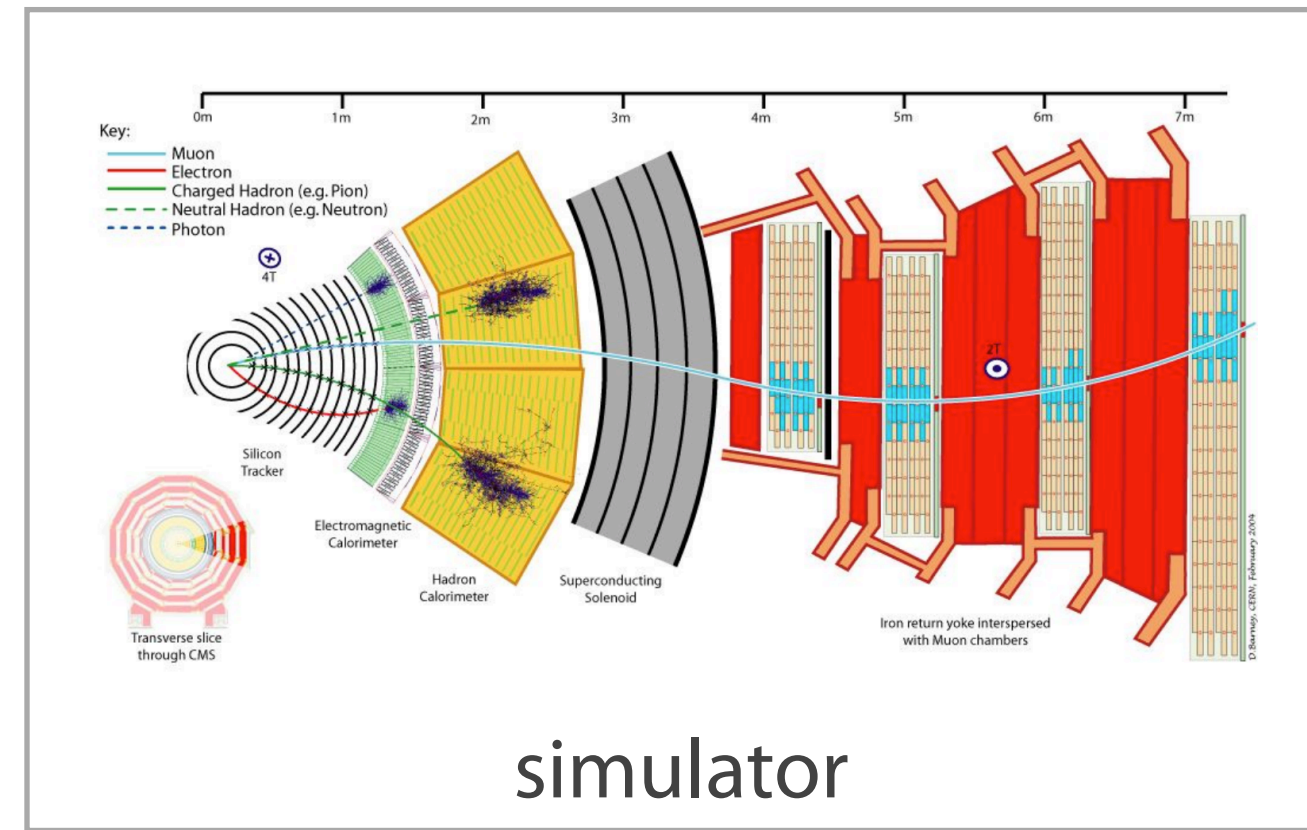
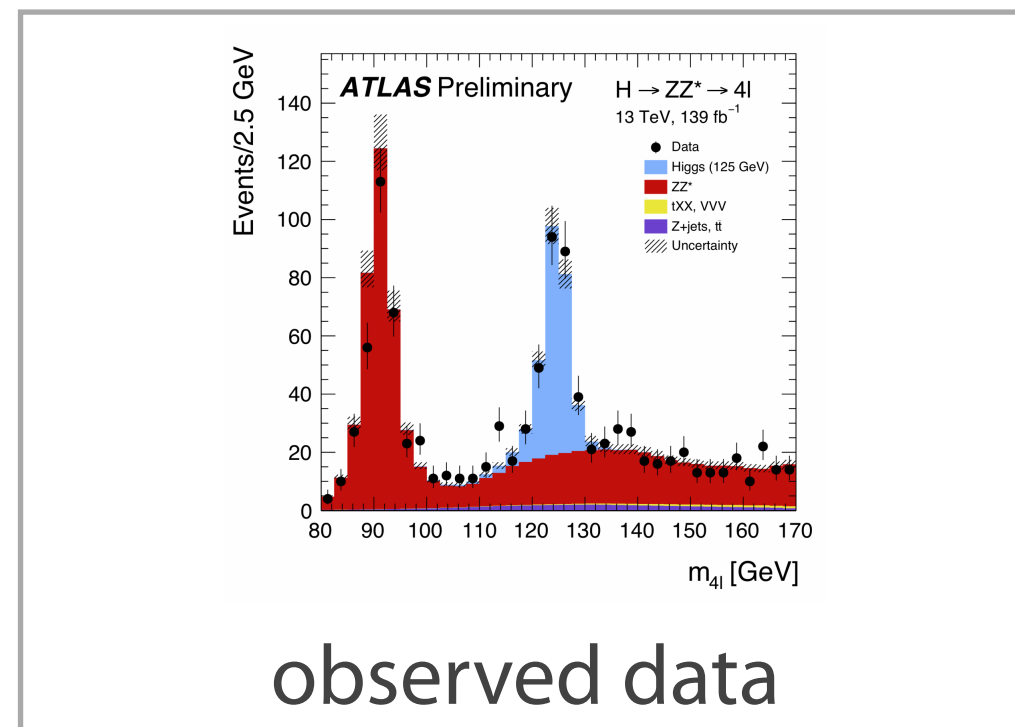
observed data

# SBI with cNF's

$p(\theta)$

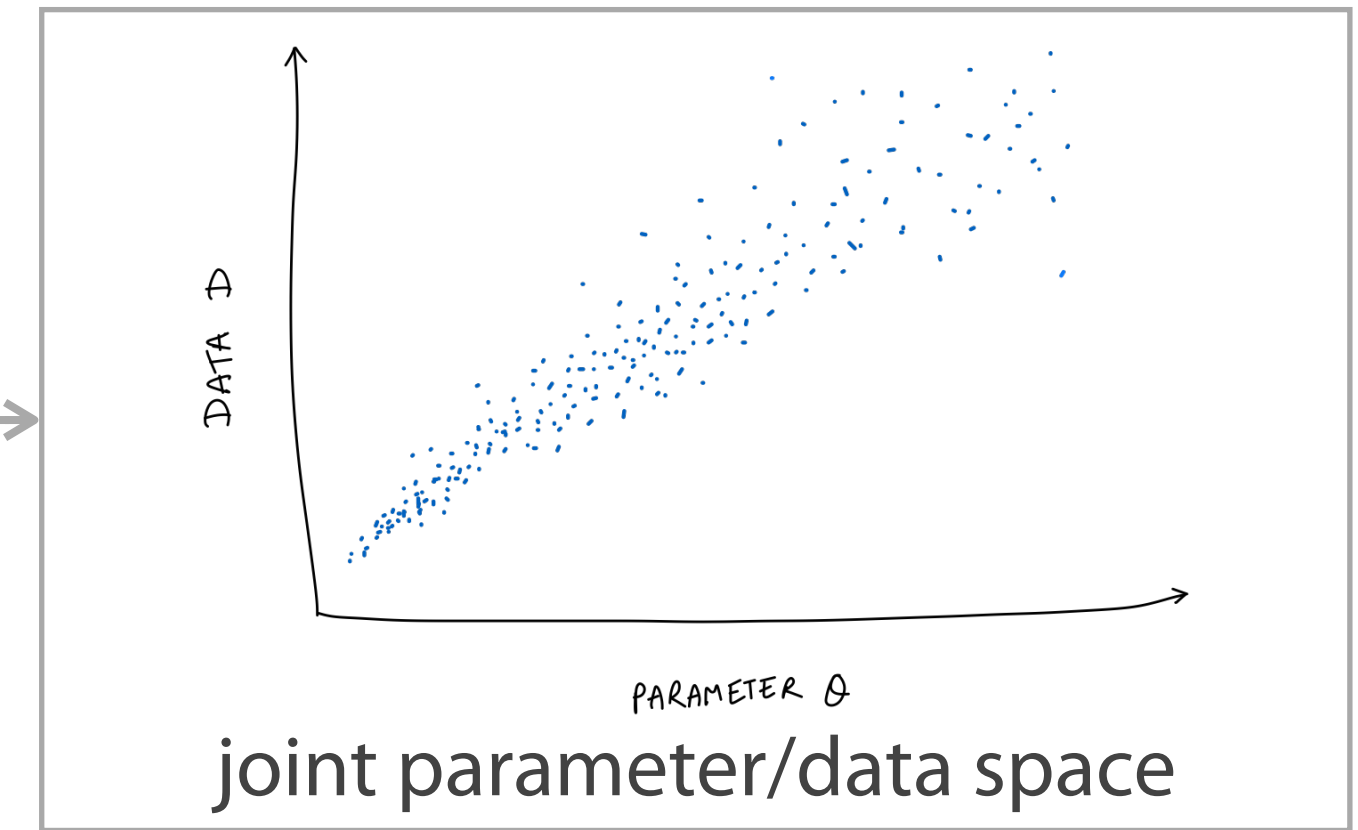


$X_{\text{obs}}$

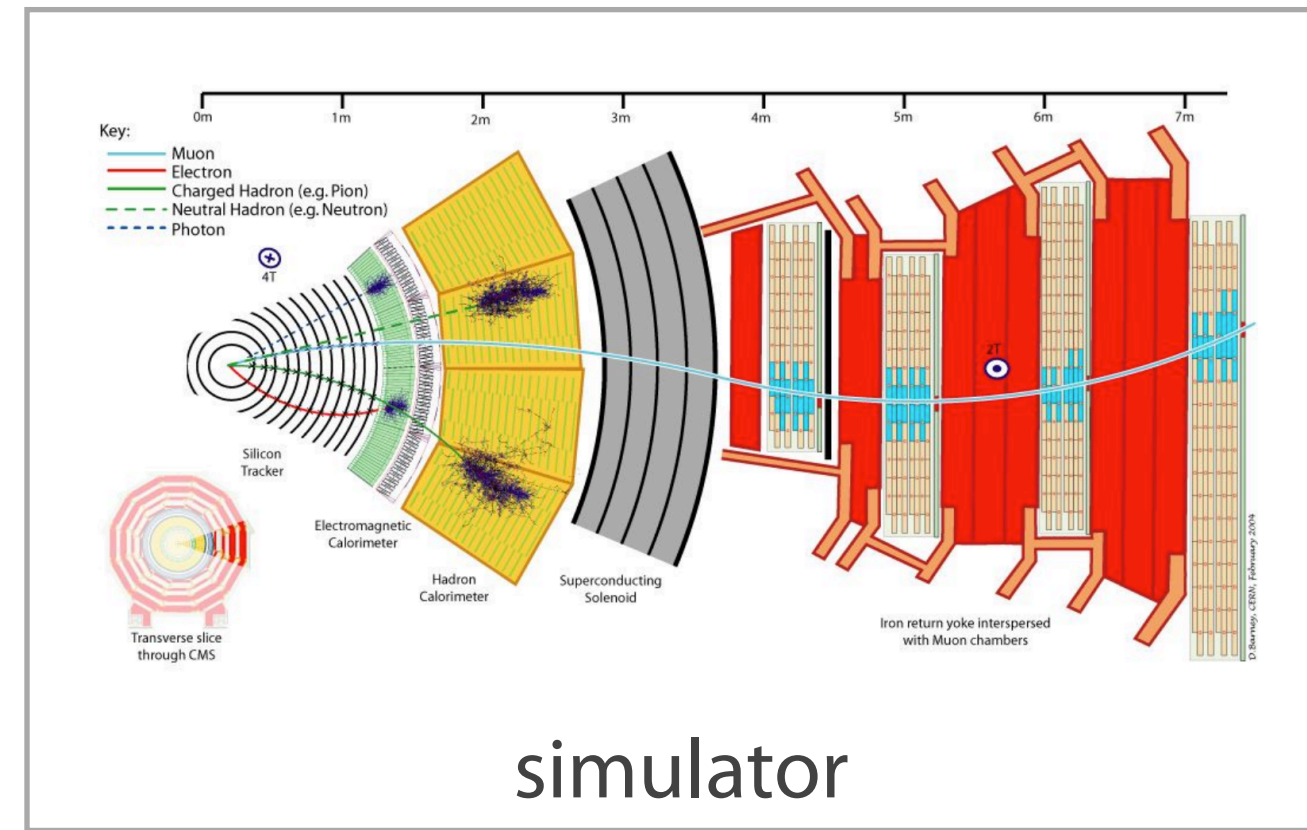
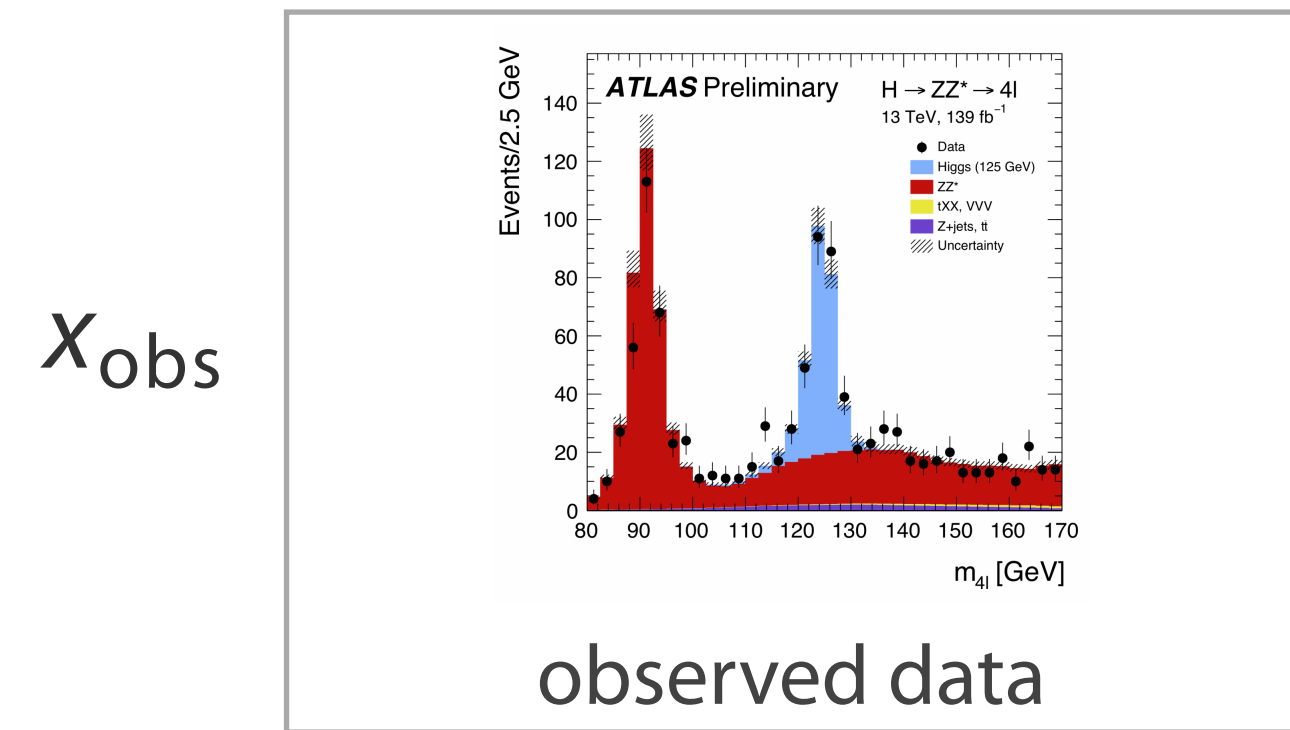
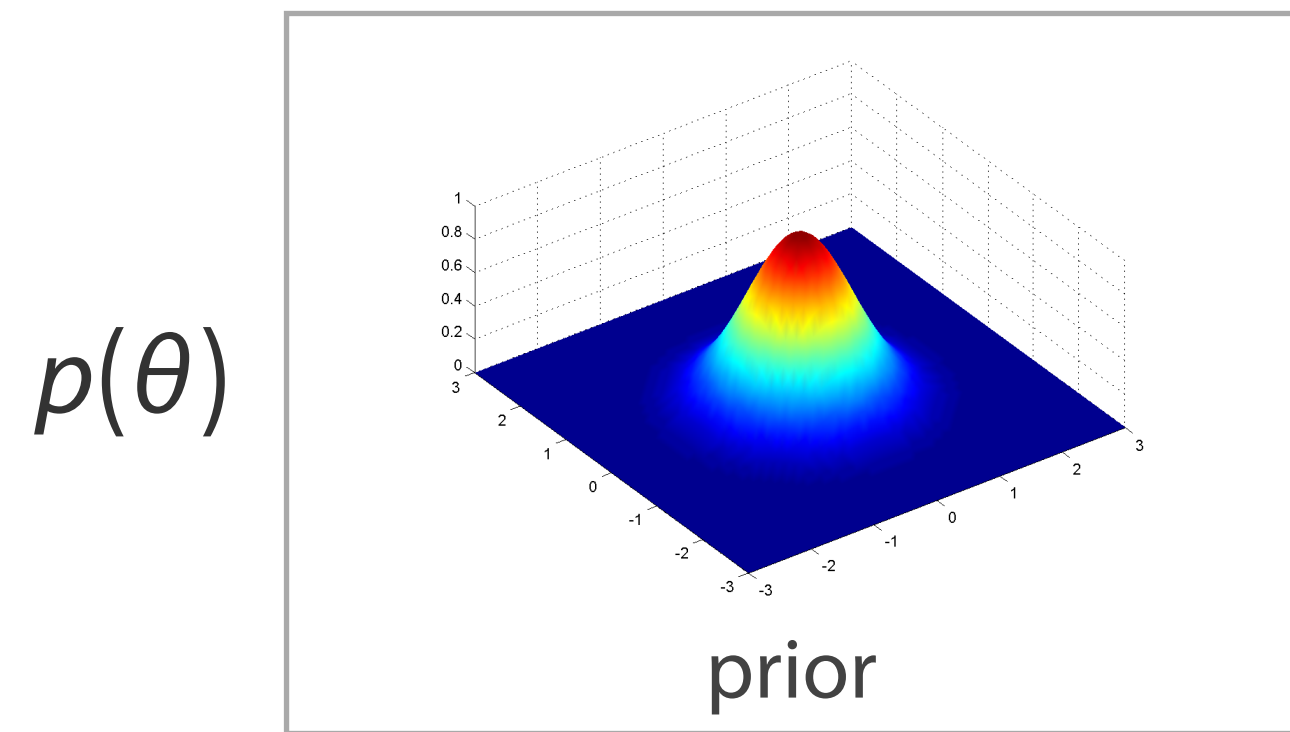


$\theta \sim p(\theta)$

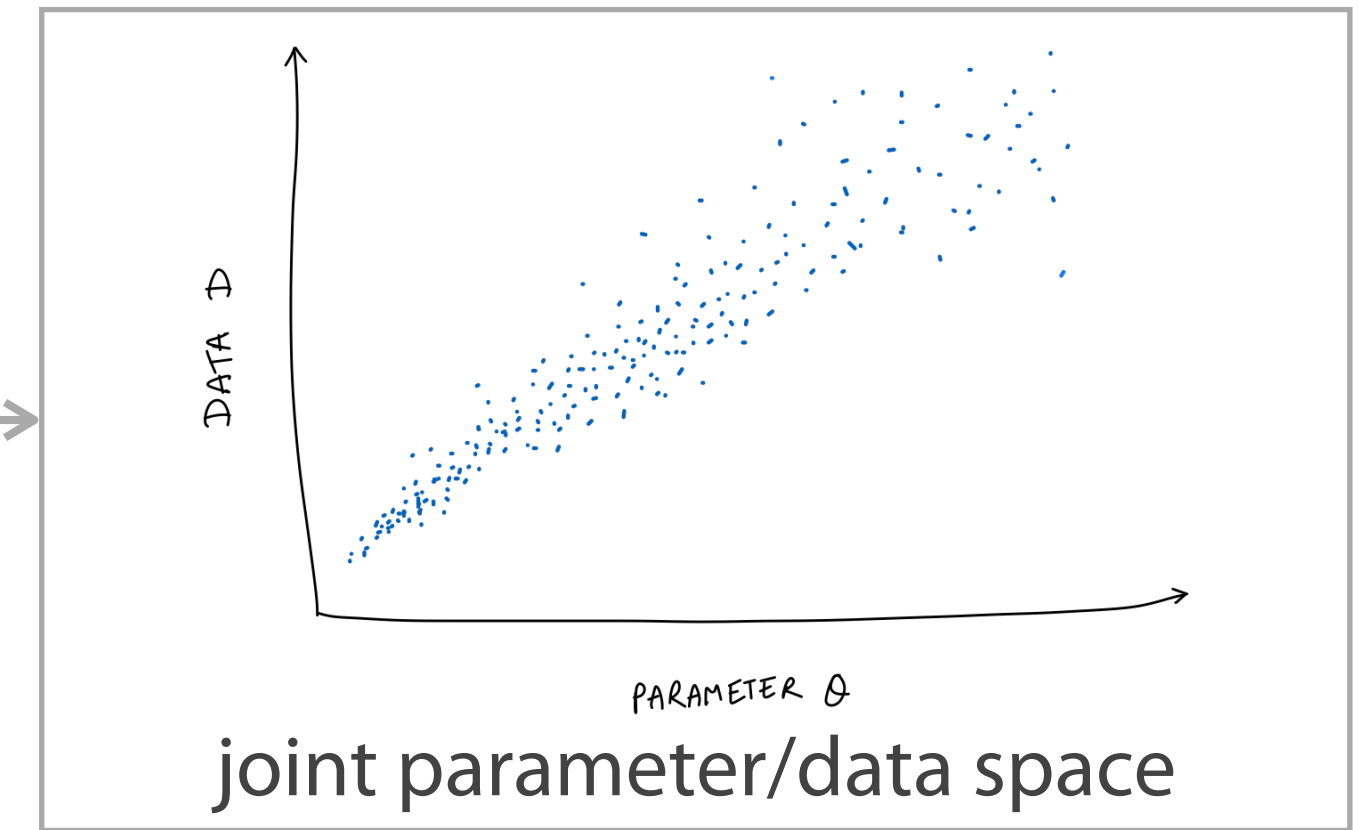
$x \sim p(x|\theta)$



# SBI with cNF's

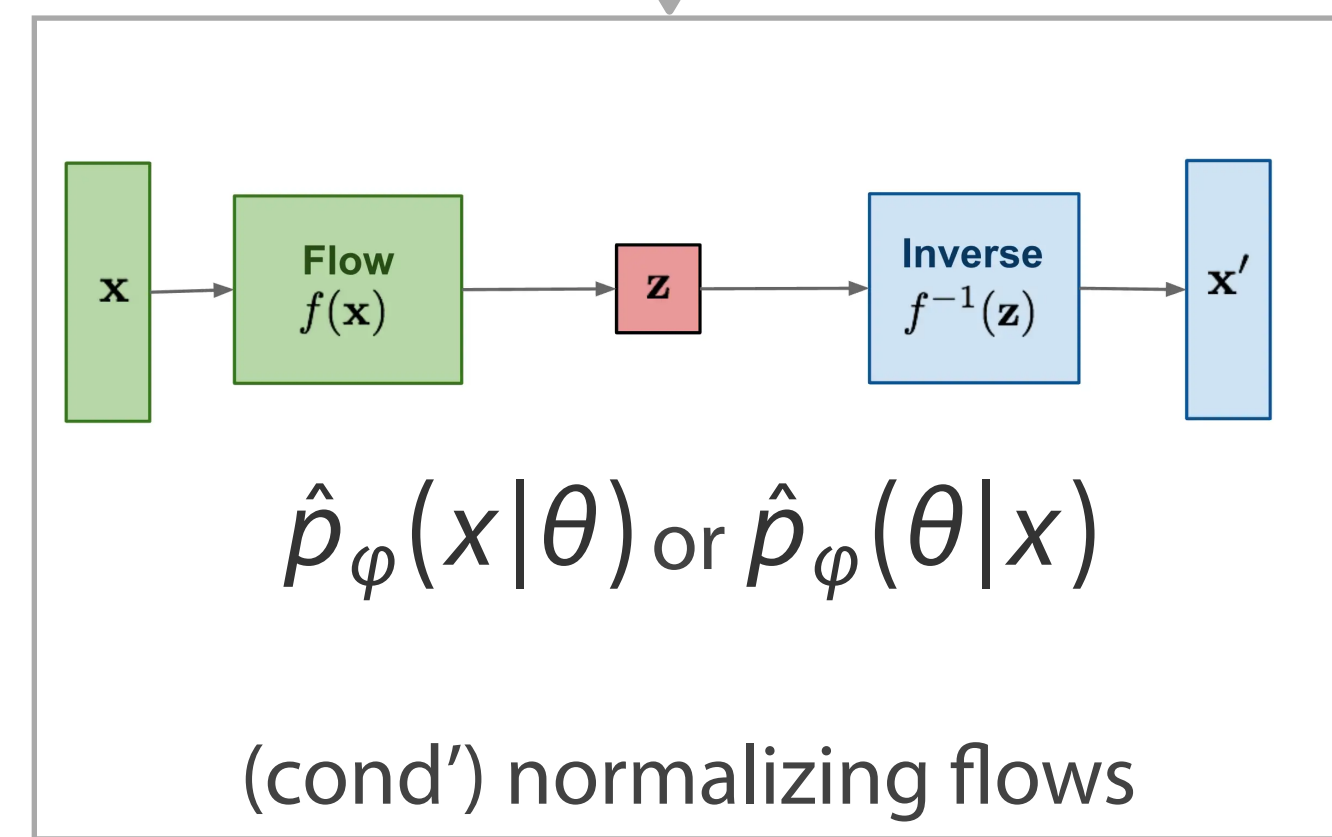


$$x \sim p(x|\theta)$$

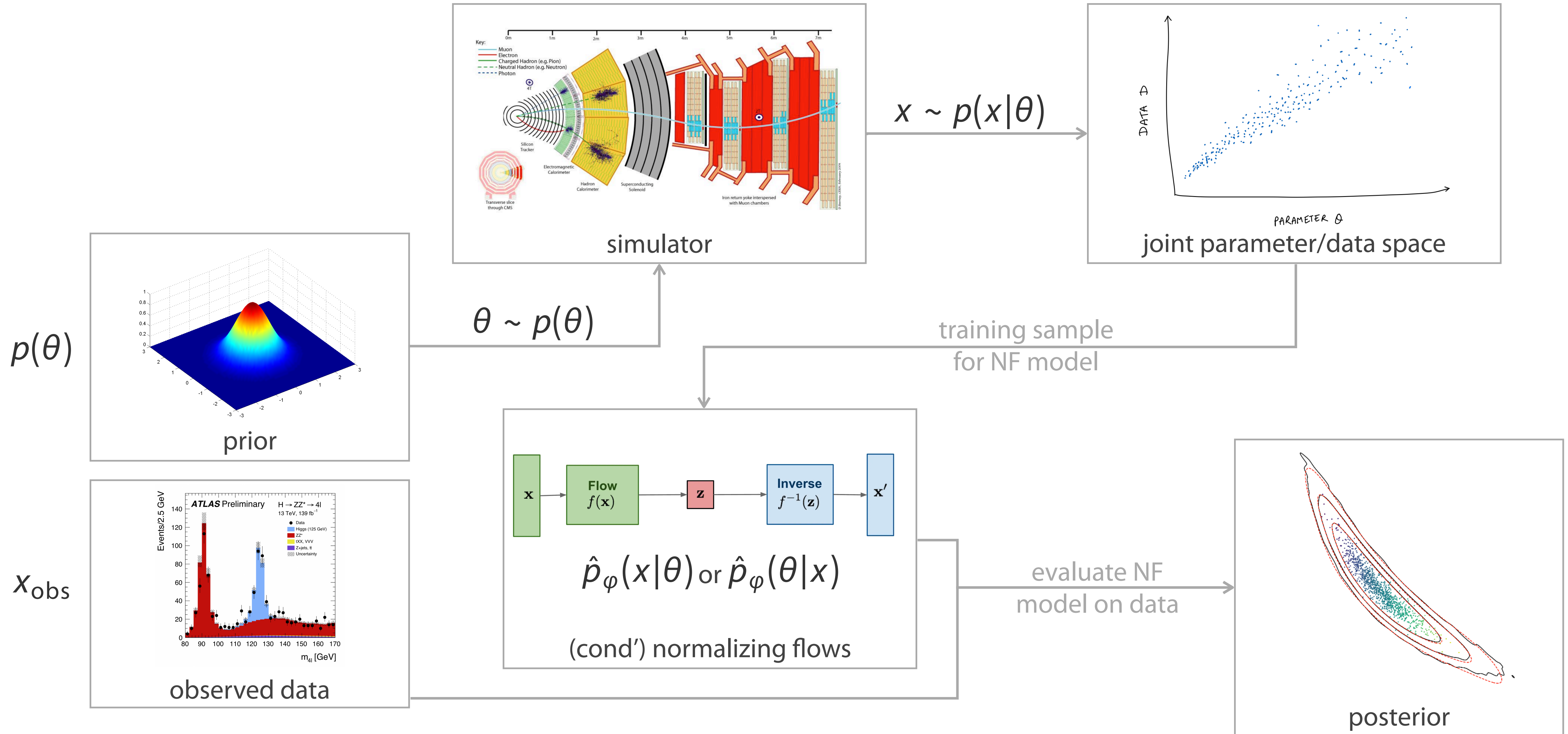


$$\theta \sim p(\theta)$$

training sample  
for NF model



# SBI with cNF's



# SBI with cNF's

Two main flavours

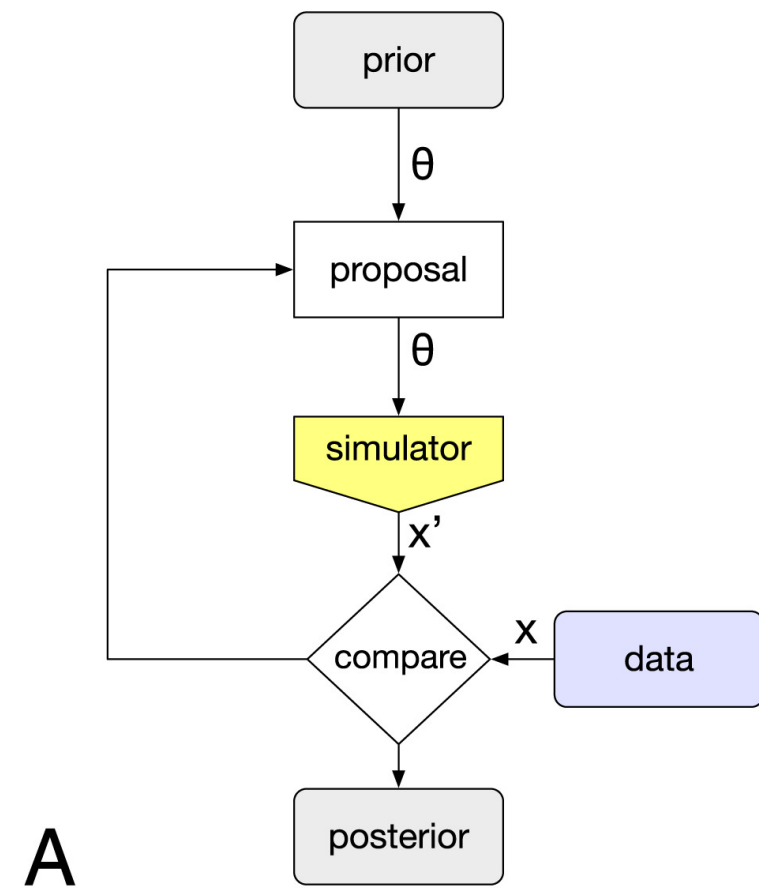
	neural POSTERIOR estimation (NPE)	neural LIKELIHOOD estimation (NLE)
Conditioning	condition on data to learn posterior $\hat{p}_\varphi(\theta x)$	condition on parameter to learn likelihood $\hat{p}_\varphi(x \theta)$
Posterior	fast evaluation on new data (amortised)	sampling (MCMC) to obtain posterior
Prior	depends on prior	no dependence on prior

▸ **Sequential version: SNPE/SNLE**

- Alternate sampling/training to target simulations in region of interest

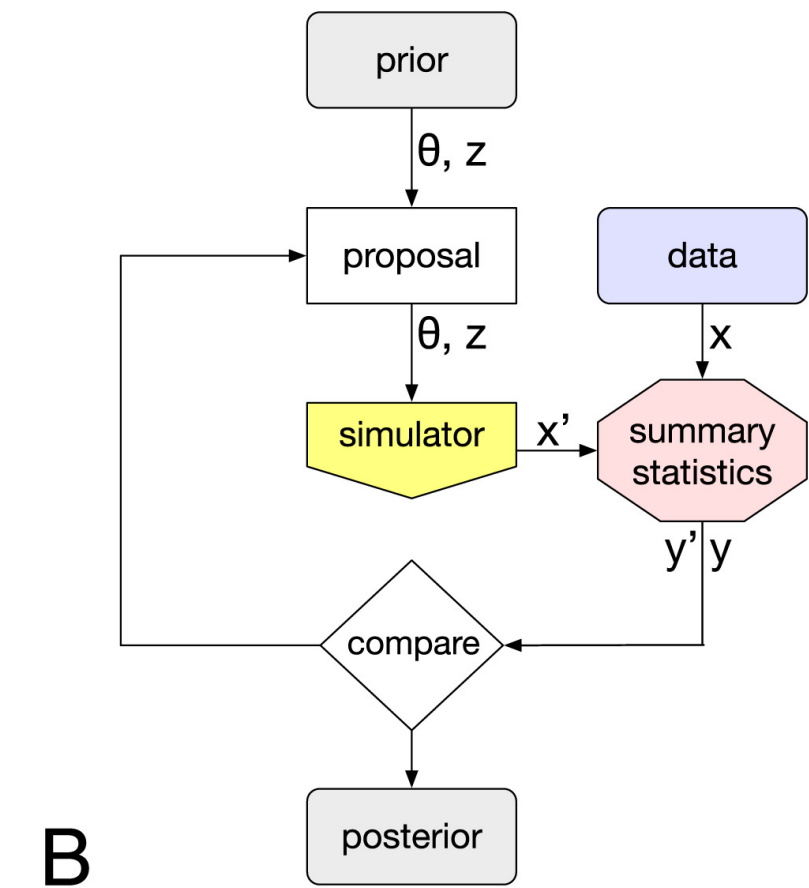
# Approaches to SBI

**Approximate Bayesian Computation with Monte Carlo sampling**



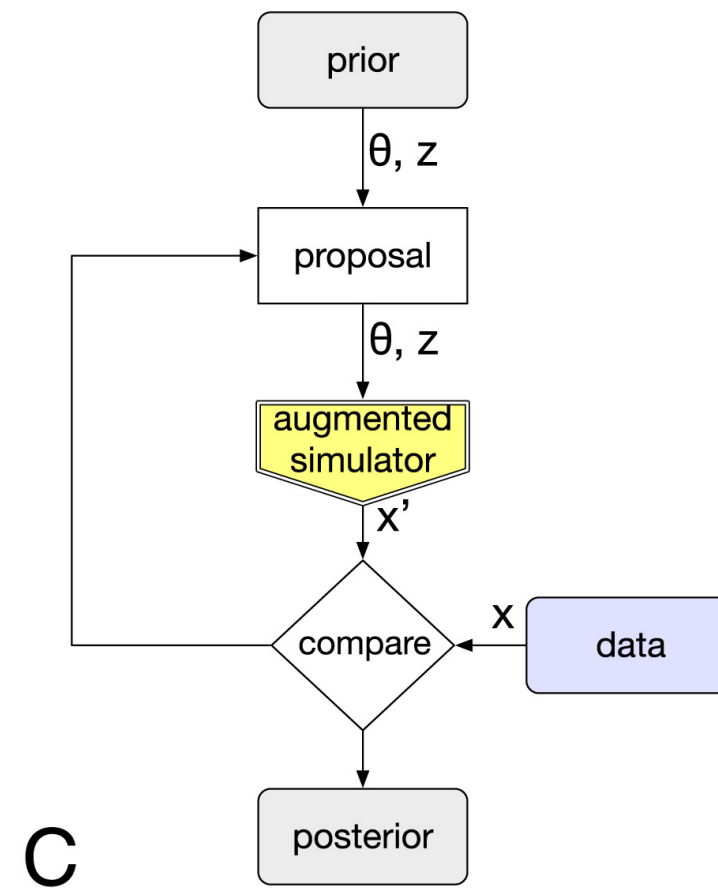
A

**Approximate Bayesian Computation with learned summary statistics**



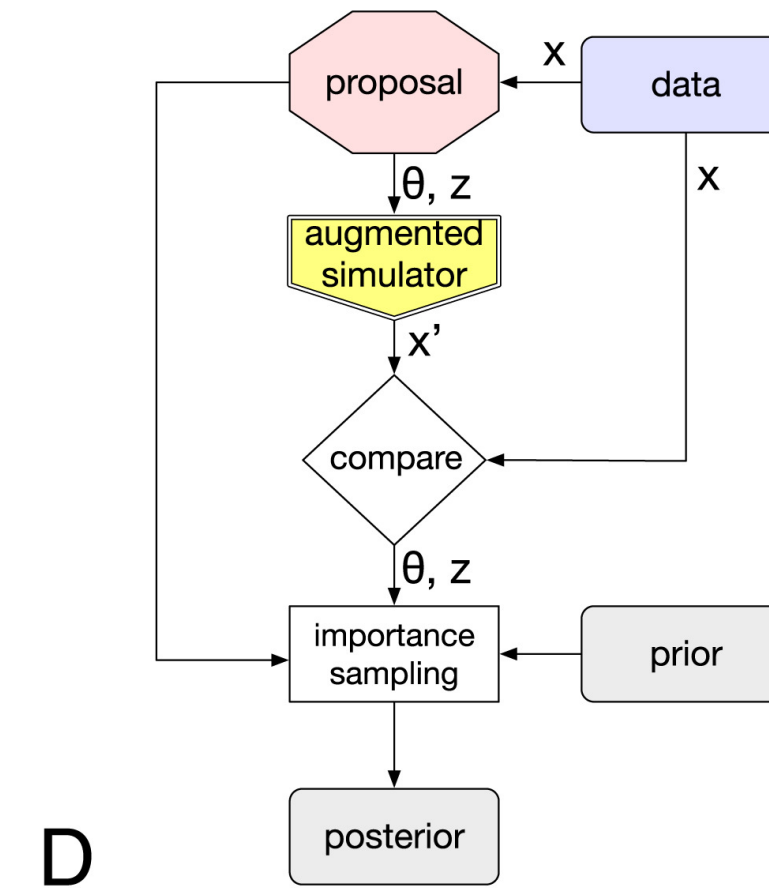
B

**Probabilistic Programming with Monte Carlo sampling**



C

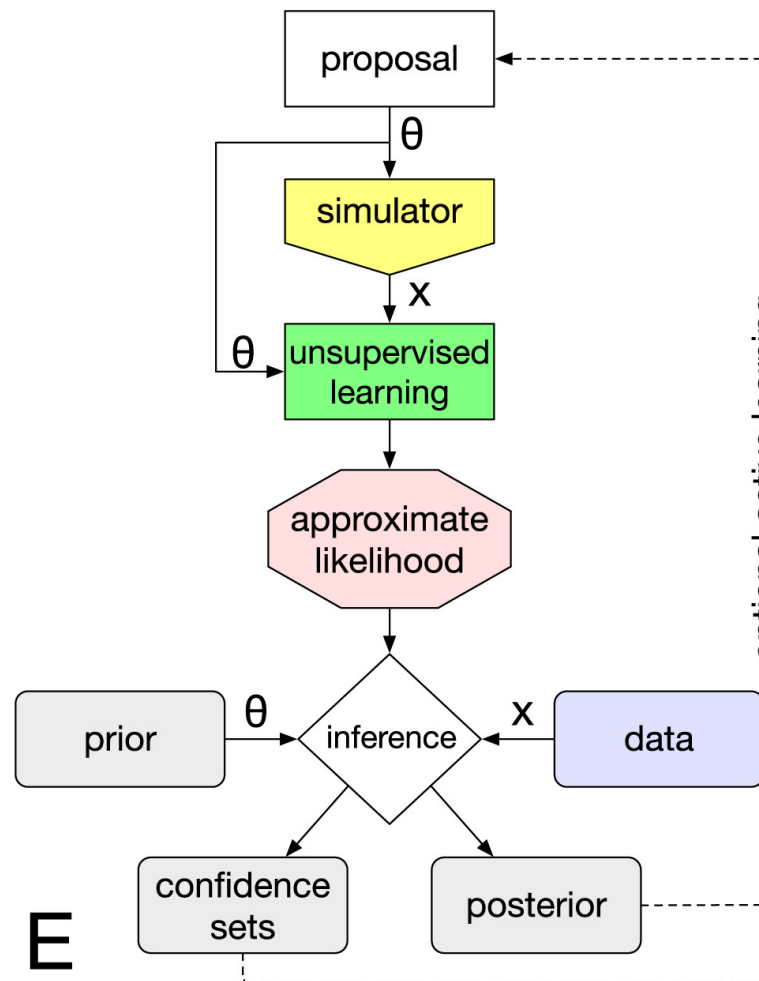
**Probabilistic Programming with Inference Compilation**



D

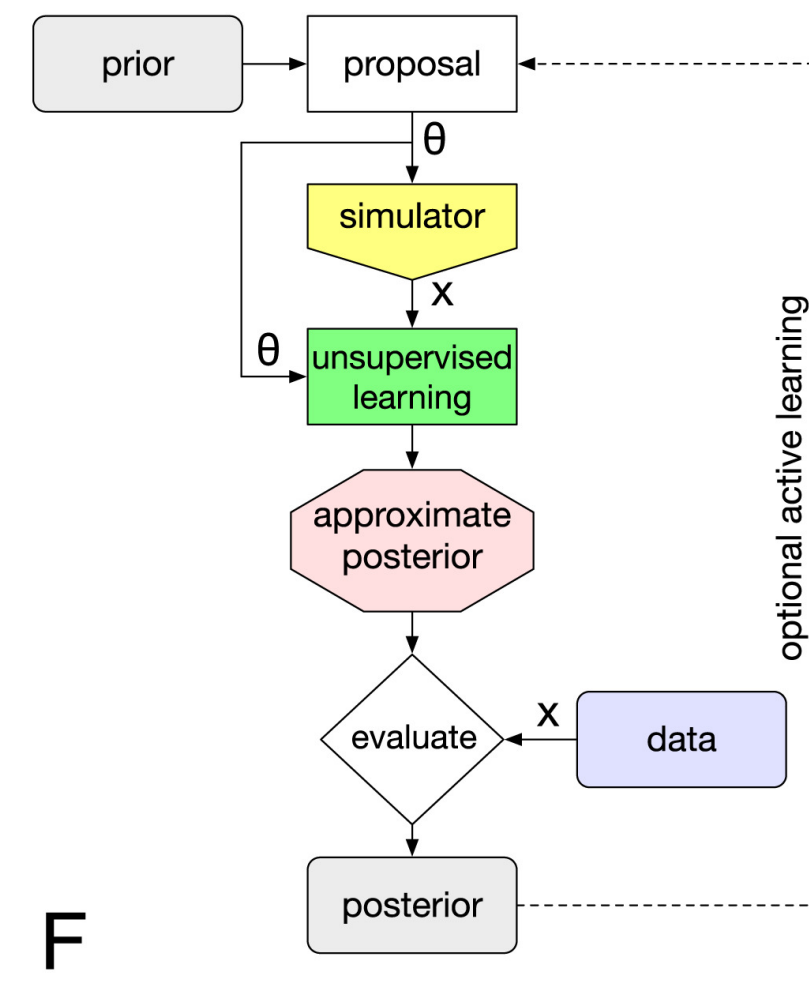
← Simulator directly used during inference

**Amortized likelihood**



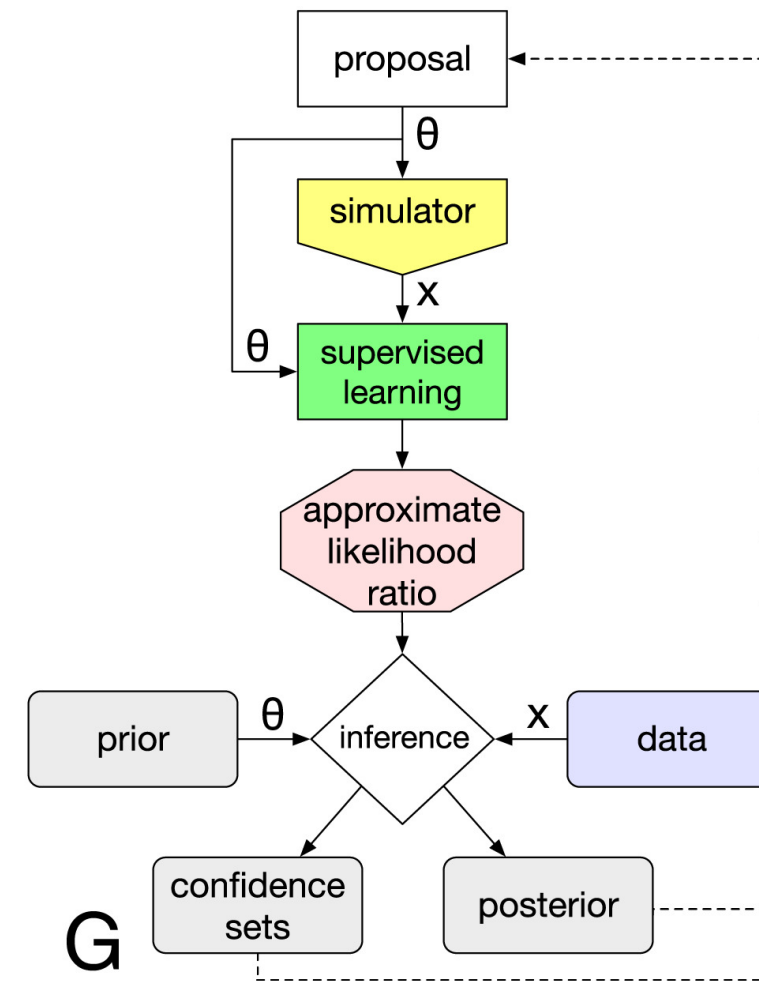
E

**Amortized posterior**



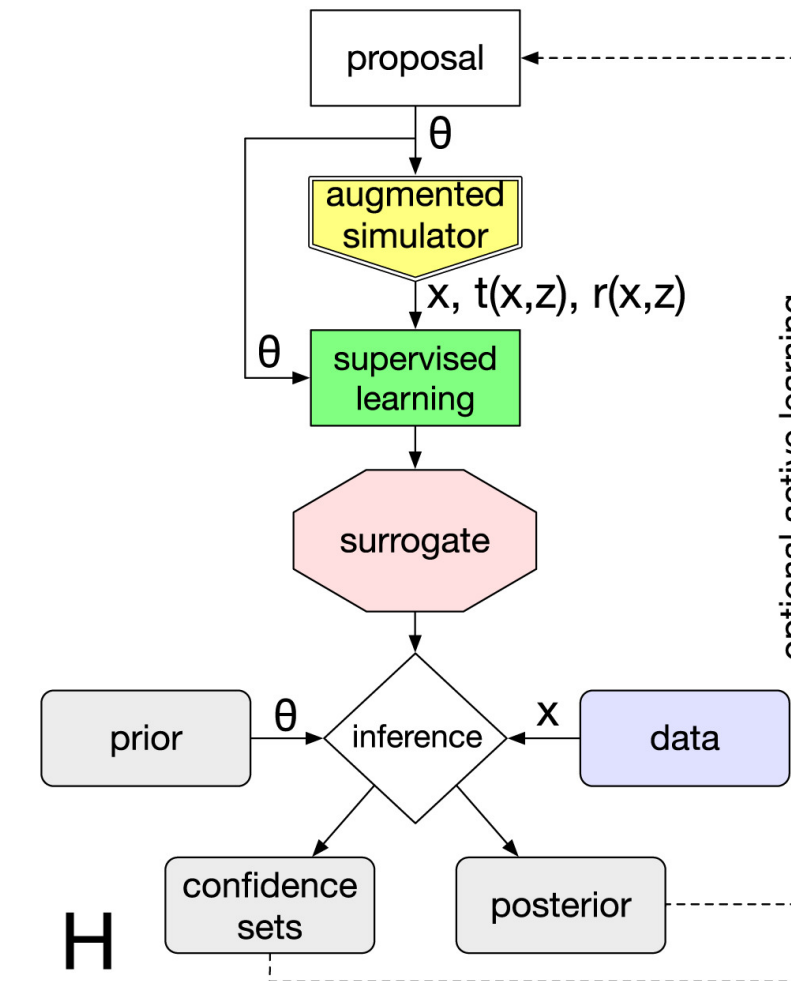
F

**Amortized likelihood ratio**



G

**Amortized surrogates trained with augmented data**



H

← Simulator used before inference (*amortized*)

Different approaches to simulation-based inference (review [Cranmer+20](#))

# Tutorial

Freely adapted from Peter Melchior's [excellent tutorial](#)

```
chi2 = (y - y_model)**2 / (yerr**2)
return np.sum(-chi2 / 2)

def log_prior(theta):
    if all(theta > theta_low) and all(theta < theta_high):
        return 0
    return -np.inf

def log_posterior(theta, x, y, yerr):
    lp = log_prior(theta)
    if np.isfinite(lp):
        lp += log_likelihood(theta, x, y, yerr)
    return lp

# create a small ball around the MLE the initialize each walker
nwalkers, ndim = 30, 5
theta_guess = [0.5, 0.6, 0.2, -0.2, 0.1]
pos = theta_guess + 1e-4 * np.random.randn(nwalkers, ndim)

# run emcee
sampler = emcee.EnsembleSampler(nwalkers, ndim, log_posterior, args=(x, y, y_err))
sampler.run_mcmc(pos, 10000, progress=True);
```

```
100%|██████████| 10000/10000 [00:45<00:00, 220.62it/s]
```

```
fig, axes = plt.subplots(ndim, sharex=True)
mcmc_samples = sampler.get_chain()
labels = ["l", "m", "s", "a", "b"]
for i in range(ndim):
    ax = axes[i]
    ax.plot(mcmc_samples[:, :, i], "k", alpha=0.3, rasterized=True)
    ax.set_xlim(0, 1000)
    ax.set_ylabel(labels[i])

axes[-1].set_xlabel("step number");
```



# Setup

[Google Colab notebook](#)

## Model and goal

### ▸ Packages

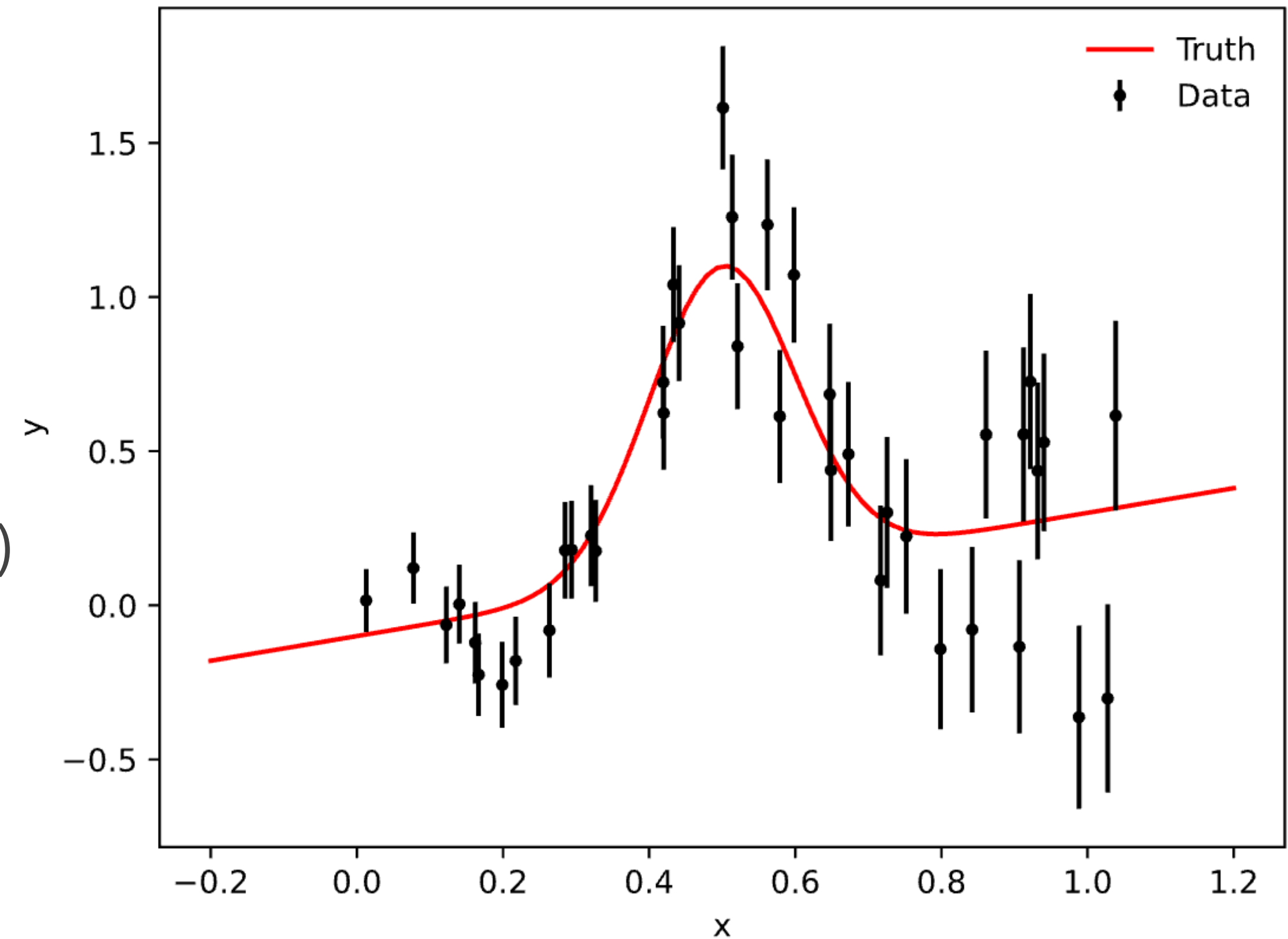
- [sbi](#): SBI package implementing SNPE, SNLE, NLRE among others
- [PyTorch](#): facebook's machine-learning library

### ▸ Simulator

- Model = background (affine) + signal bump (gaussian)
- Gaussian noise growing with  $x$  (why not?)

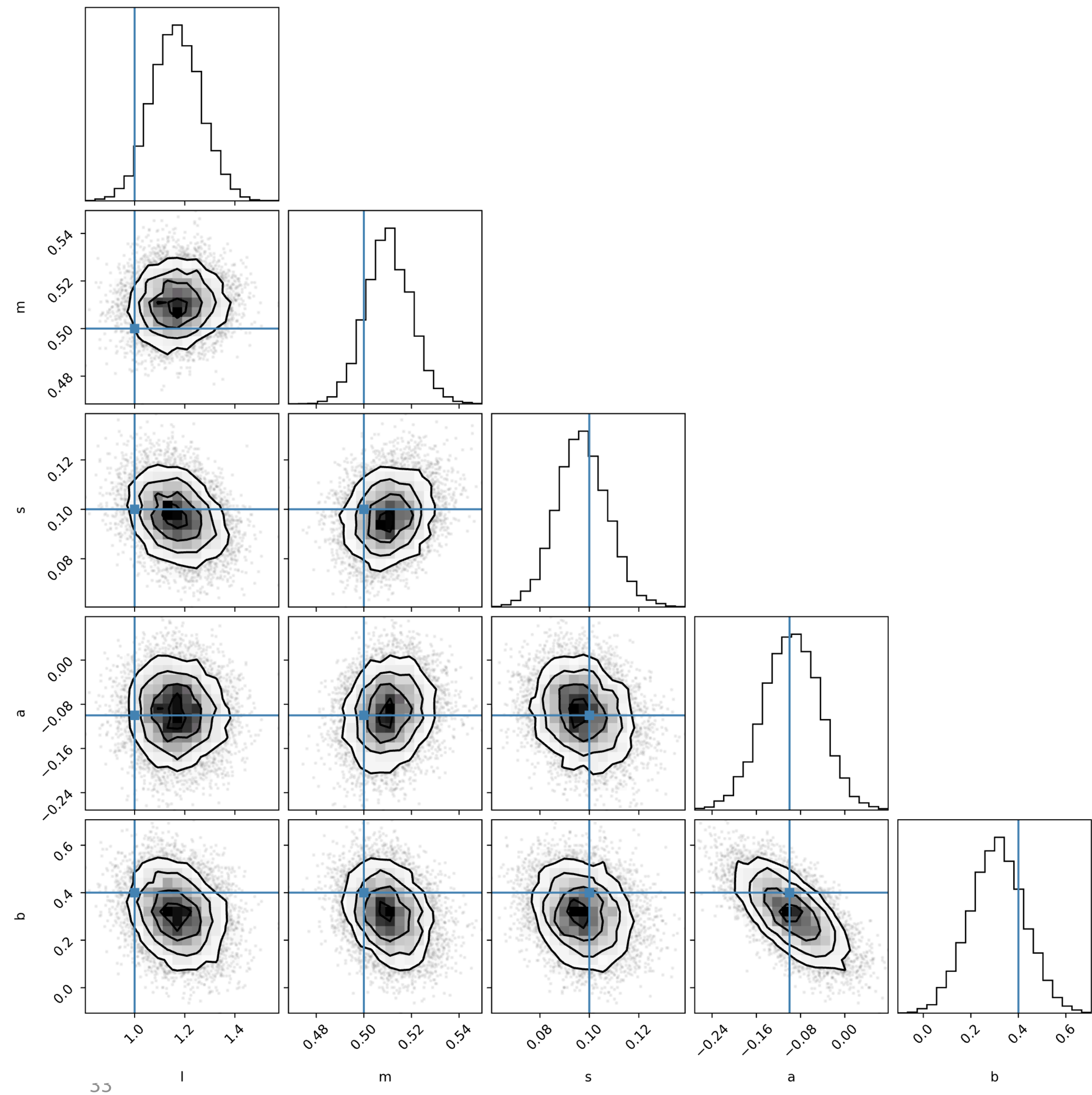
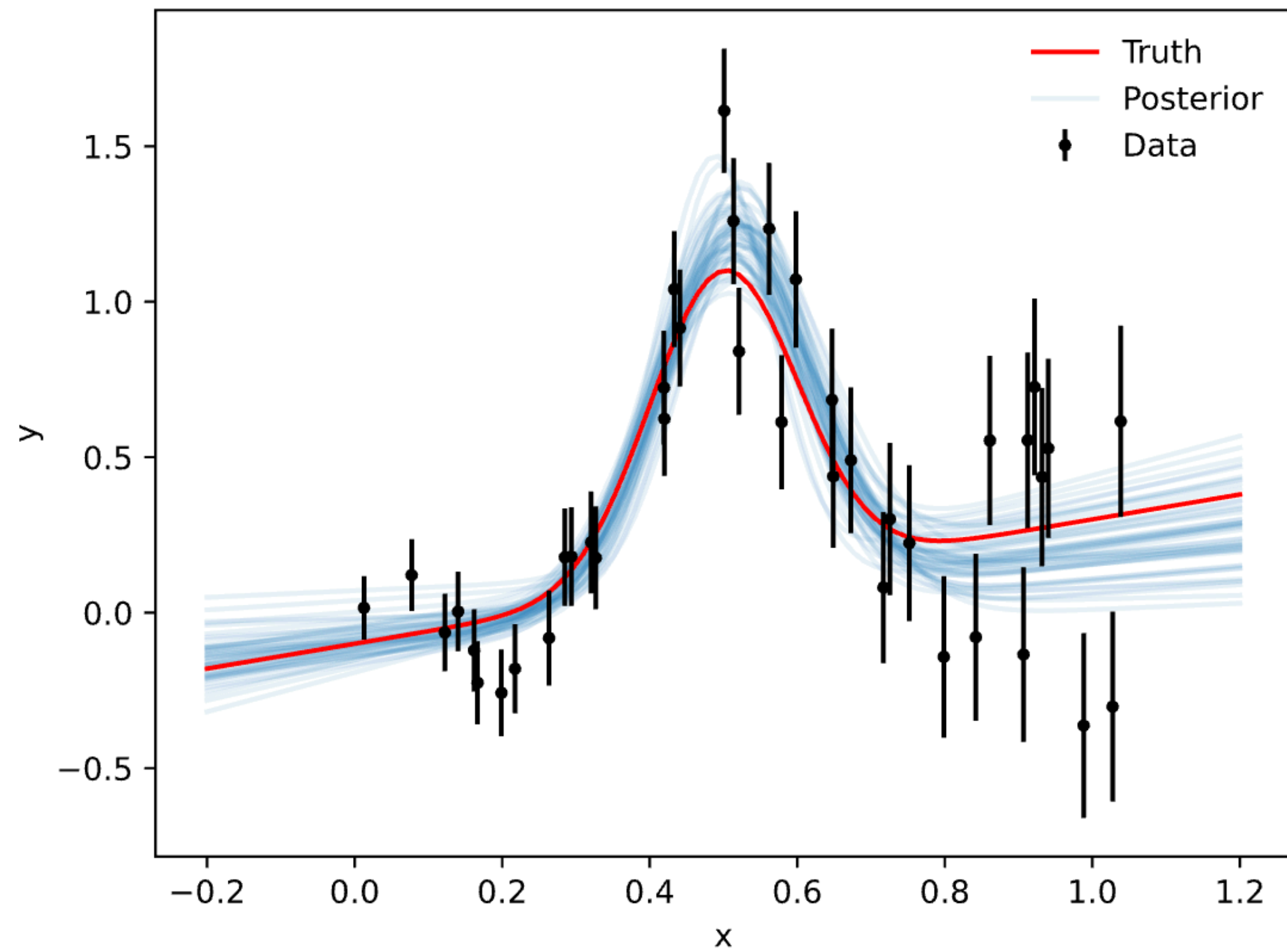
### ▸ Goals

1. Run simple SBI
2. Illustrate differences between likelihood-based and SBI



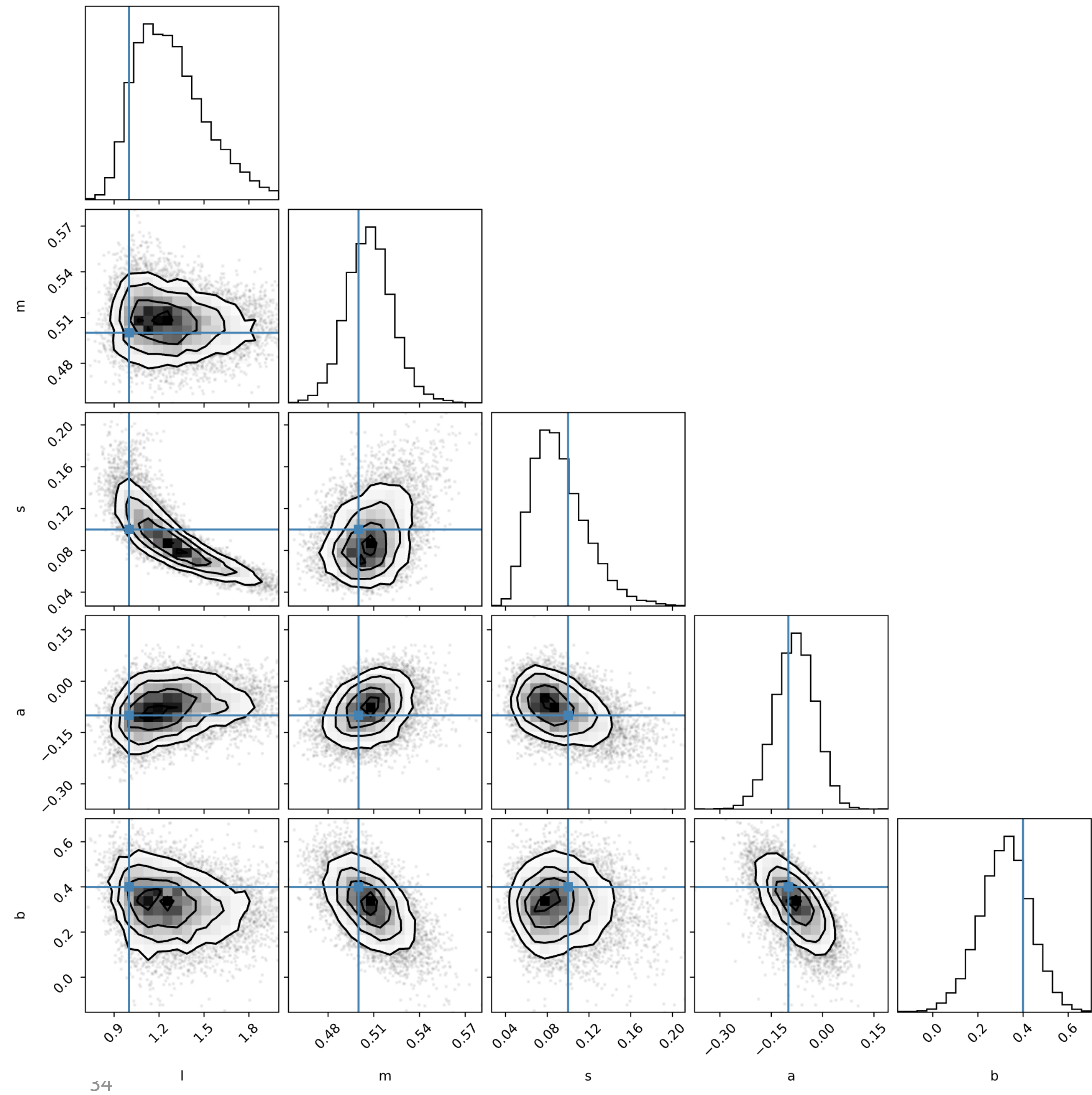
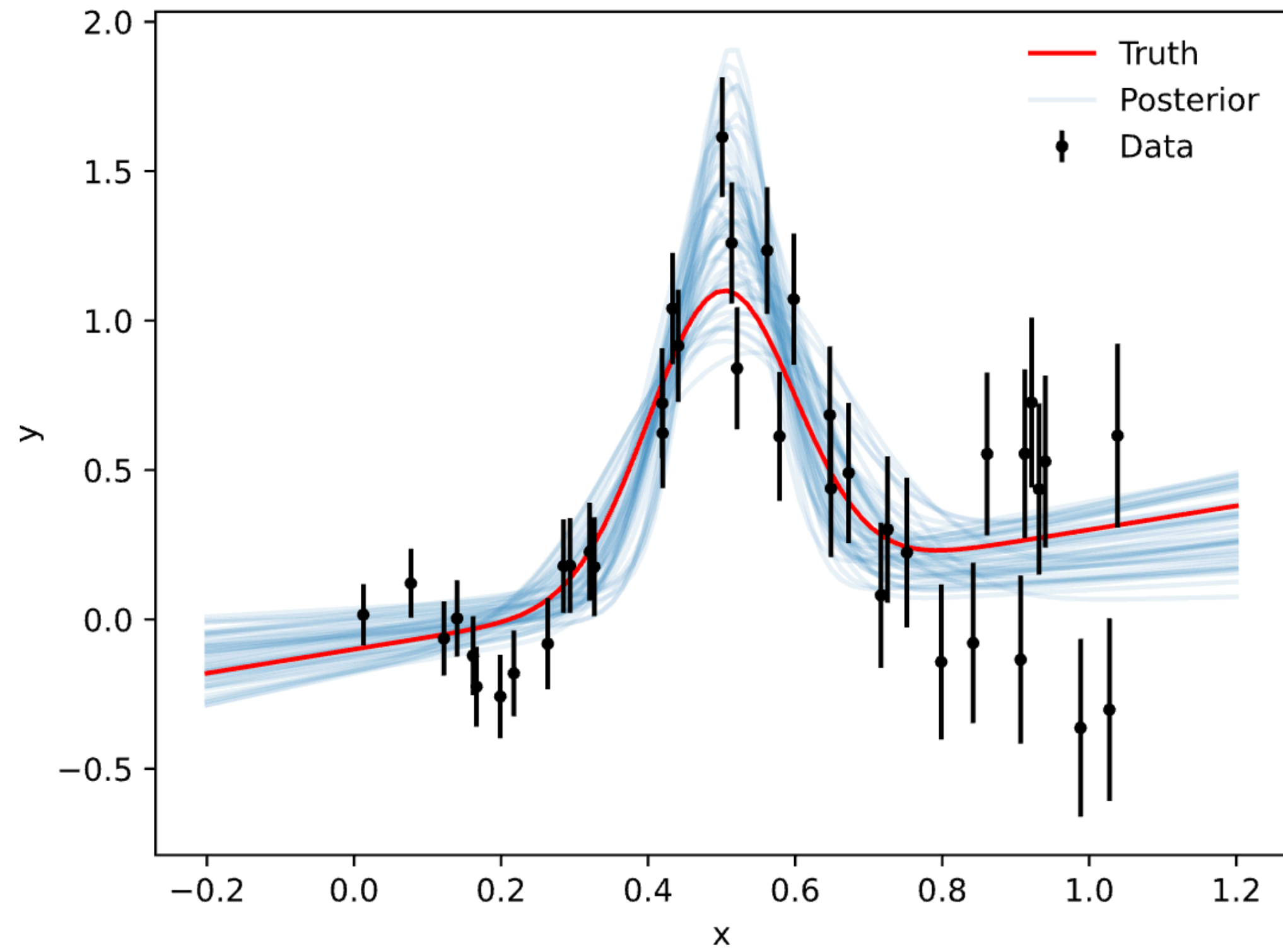
# Results

## Likelihood-based inference



# Results

## Simulation-based inference





# Beyond vanilla SBI

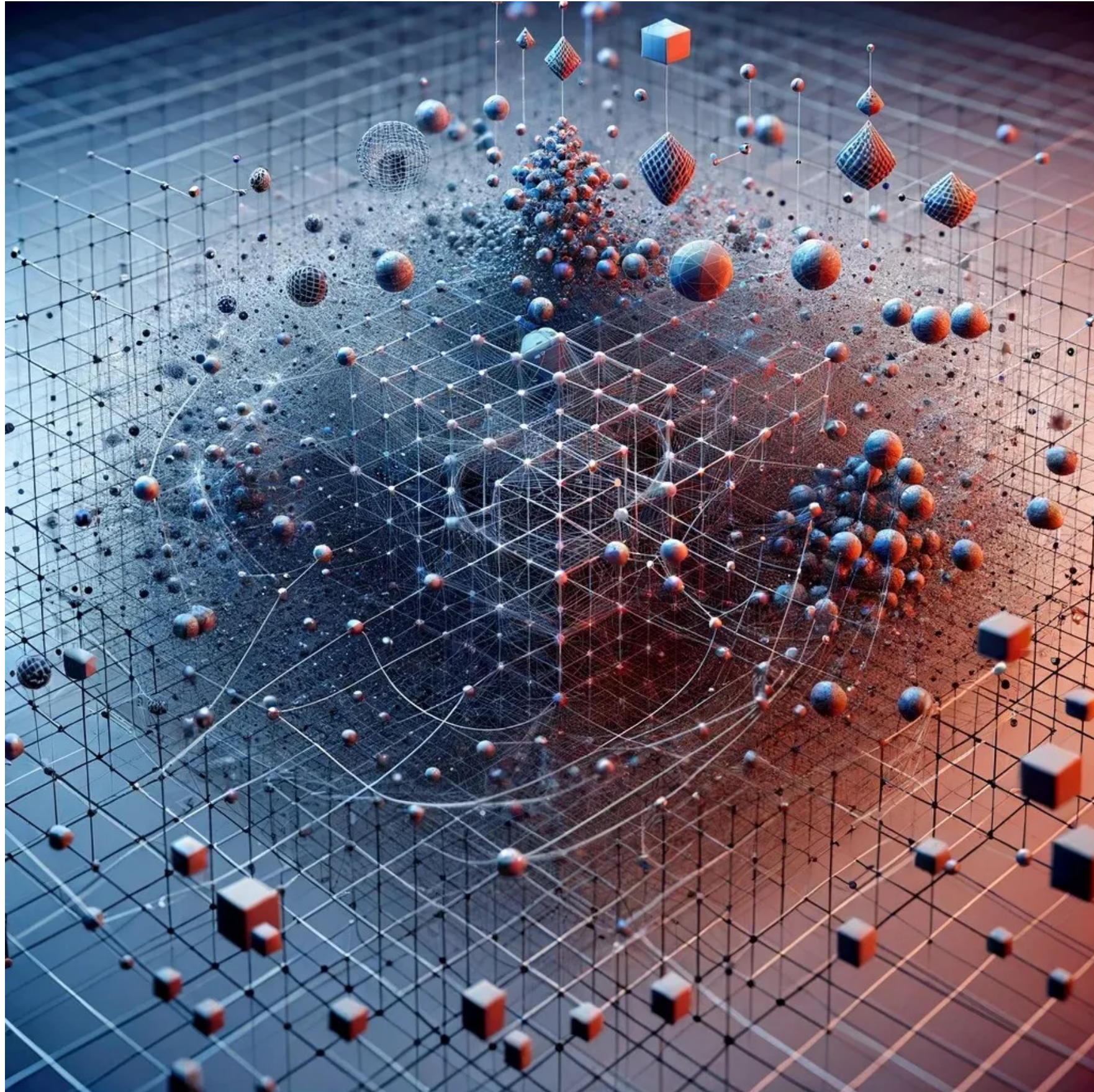
Improvements and research directions



Copilot prompt: cosmology simulations in the style of Basquiat paintings

# Challenges of SBI

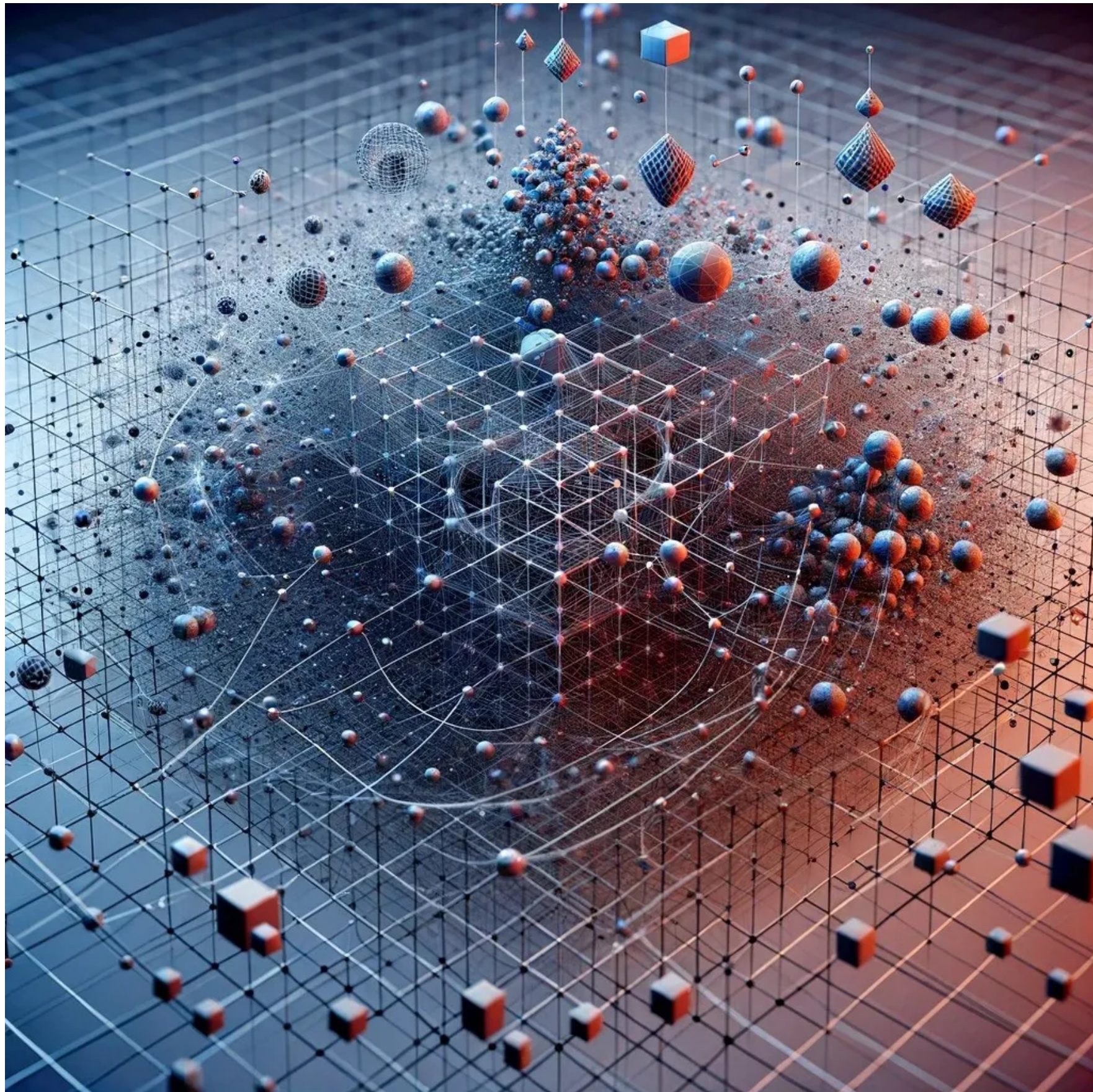
Spoiler alert: SBI is not magic



\*let's assume fitting NFs is negligible

# Challenges of SBI

Spoiler alert: SBI is not magic

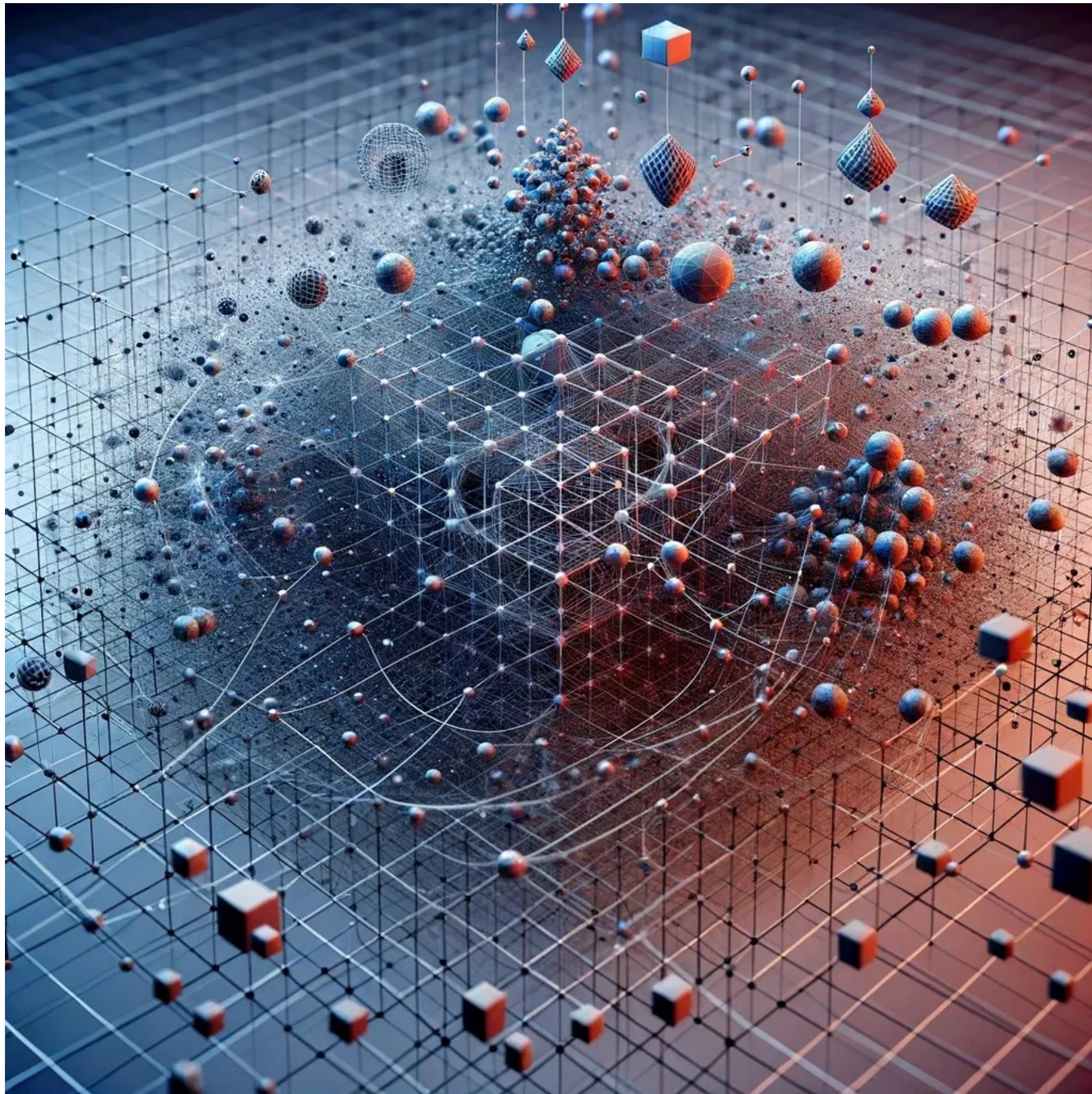


- ▶ **Curse of dimensionality** 🌀
  - ▶ Larger dimension = much harder problem
  - ▶ *Both* data and params dimensions matter

\*let's assume fitting NFs is negligible

# Challenges of SBI

Spoiler alert: SBI is not magic

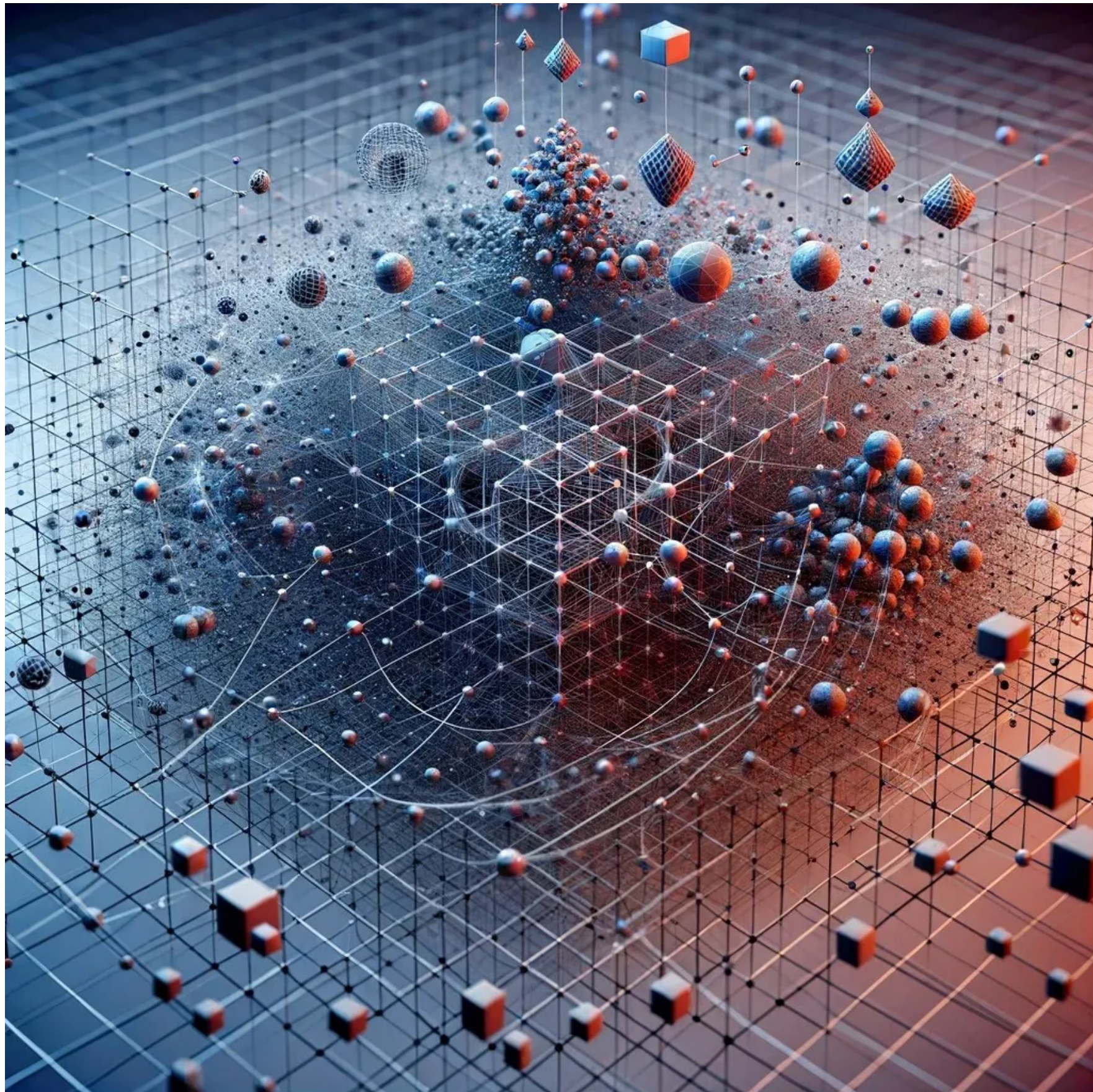


- ▶ **Curse of dimensionality** 🌀
  - ▶ Larger dimension = much harder problem
  - ▶ *Both* data and params dimensions matter
- ▶ **How many simulations?** 🔥
  - ▶ Limited computing resources\*
  - ▶ Depends on dimensions, model complexity, priors, etc

\*let's assume fitting NFs is negligible

# Challenges of SBI

Spoiler alert: SBI is not magic

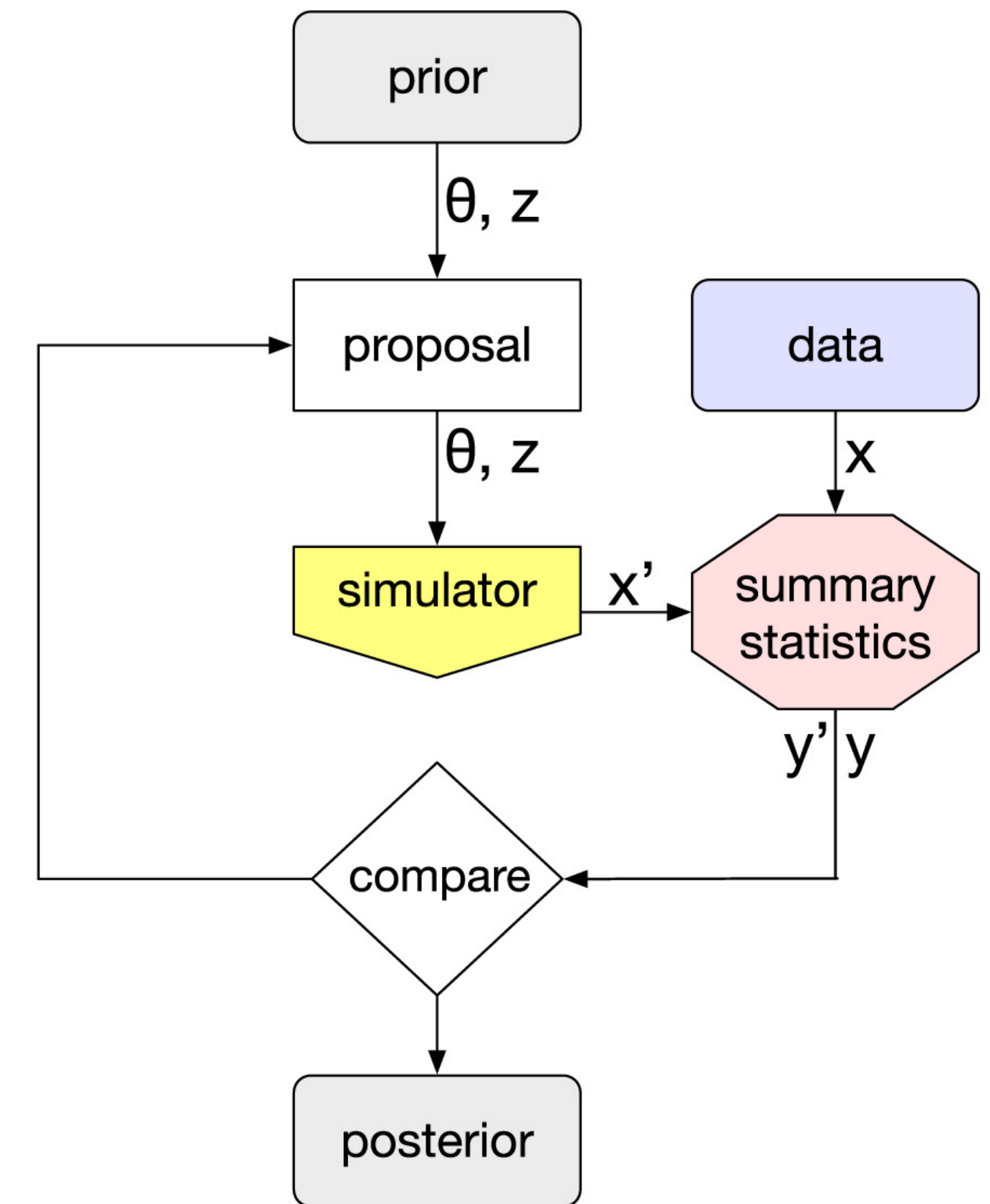


- ▶ **Curse of dimensionality** 🌀
  - ▶ Larger dimension = much harder problem
  - ▶ *Both* data and params dimensions matter
- ▶ **How many simulations?** 🔥
  - ▶ Limited computing resources\*
  - ▶ Depends on dimensions, model complexity, priors, etc
- ▶ **Ideas to circumvent these issues?** 😎
  1. Reduce dimension of data
  2. Target simulations
  3. Better/faster simulator

\*let's assume fitting NFs is negligible

# (Learned) summary statistics

## Approximate Bayesian Computation with learned summary statistics



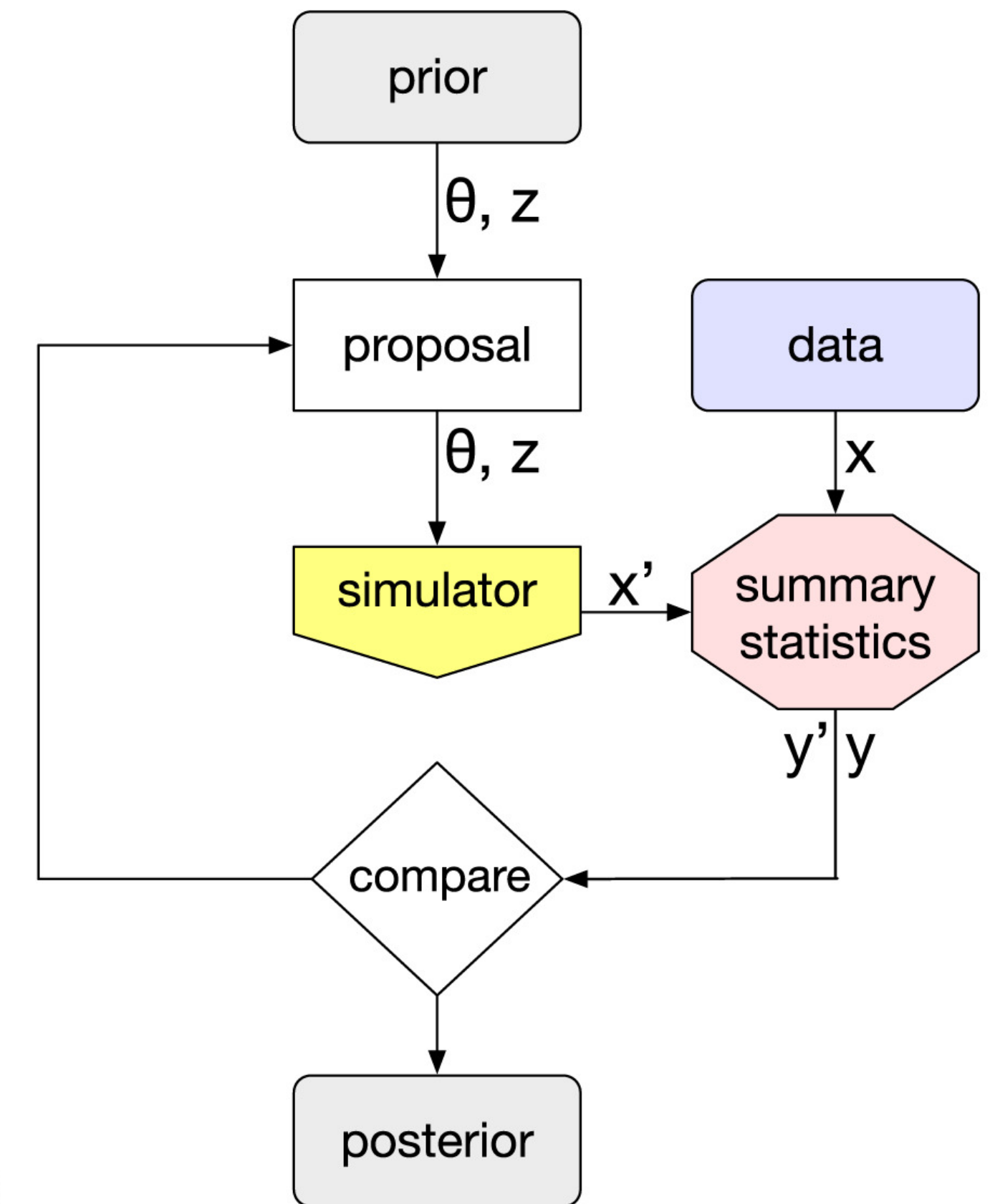
**B**

Cranmer+20

# (Learned) summary statistics

- ▶ Compress data with **minimal information loss**
  - ▶ Physicists are *experts* at it! 😎
  - ▶ Mathematically, aim to approach *sufficient* statistic  $f(X)$ , such that  $\mathbb{E}[\theta|f(X)] = \mathbb{E}[\theta|X]$  (not a definition)
  - ▶ Examples: binned histogram for Poisson, correlation function for Gaussian fields, etc.

## Approximate Bayesian Computation with learned summary statistics



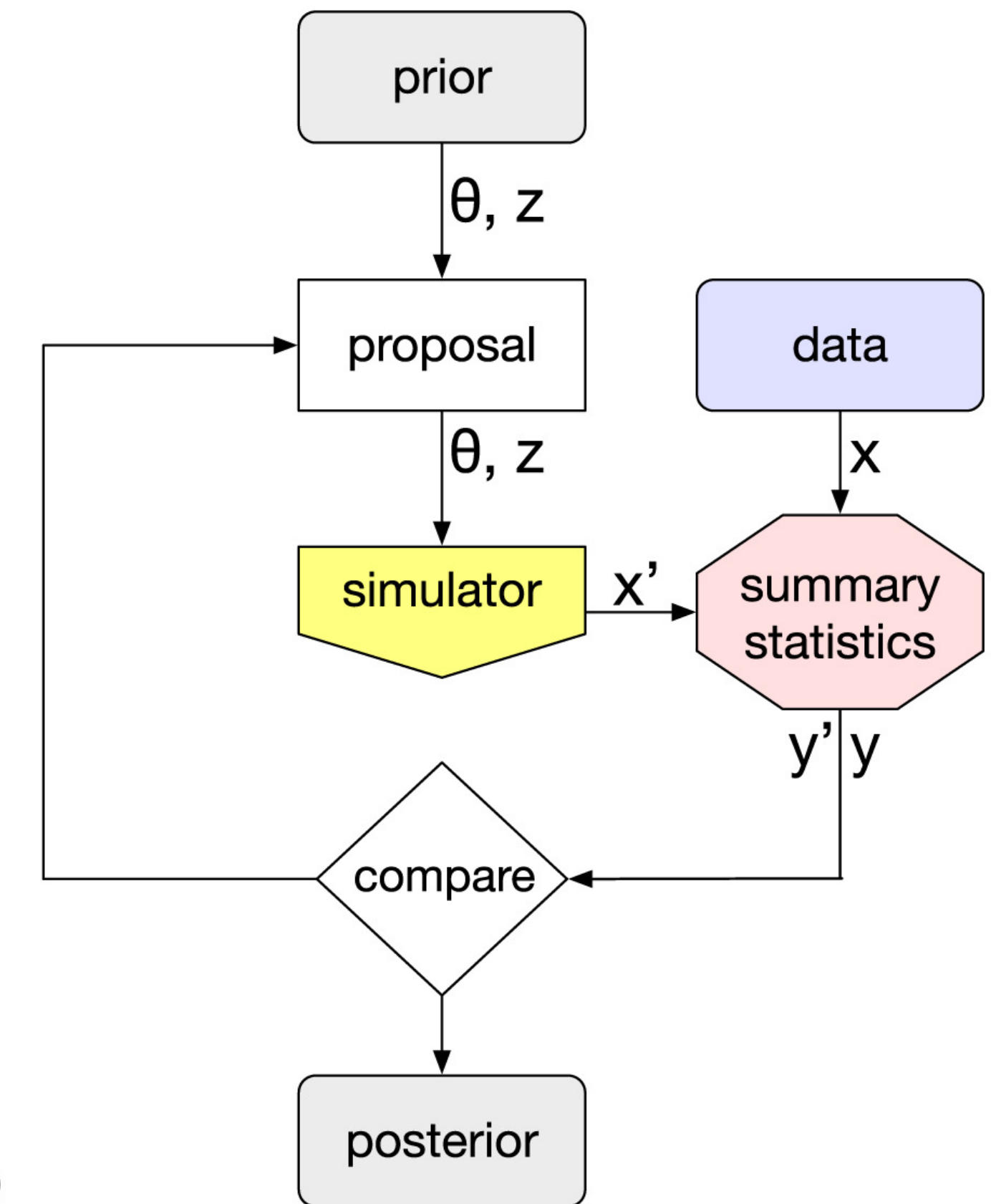
**B**

Cranmer+20

# (Learned) summary statistics

- ▶ Compress data with **minimal information loss**
  - ▶ Physicists are *experts* at it! 😎
  - ▶ Mathematically, aim to approach *sufficient* statistic  $f(X)$ , such that  $\mathbb{E}[\theta|f(X)] = \mathbb{E}[\theta|X]$  (not a definition)
  - ▶ Examples: binned histogram for Poisson, correlation function for Gaussian fields, etc.
- ▶ **Learned** summary statistics
  - ▶ PCA-like compression, e.g., MOPED (Heavens+15)
  - ▶ Neural network compression, auto-encoders, etc.

## Approximate Bayesian Computation with learned summary statistics



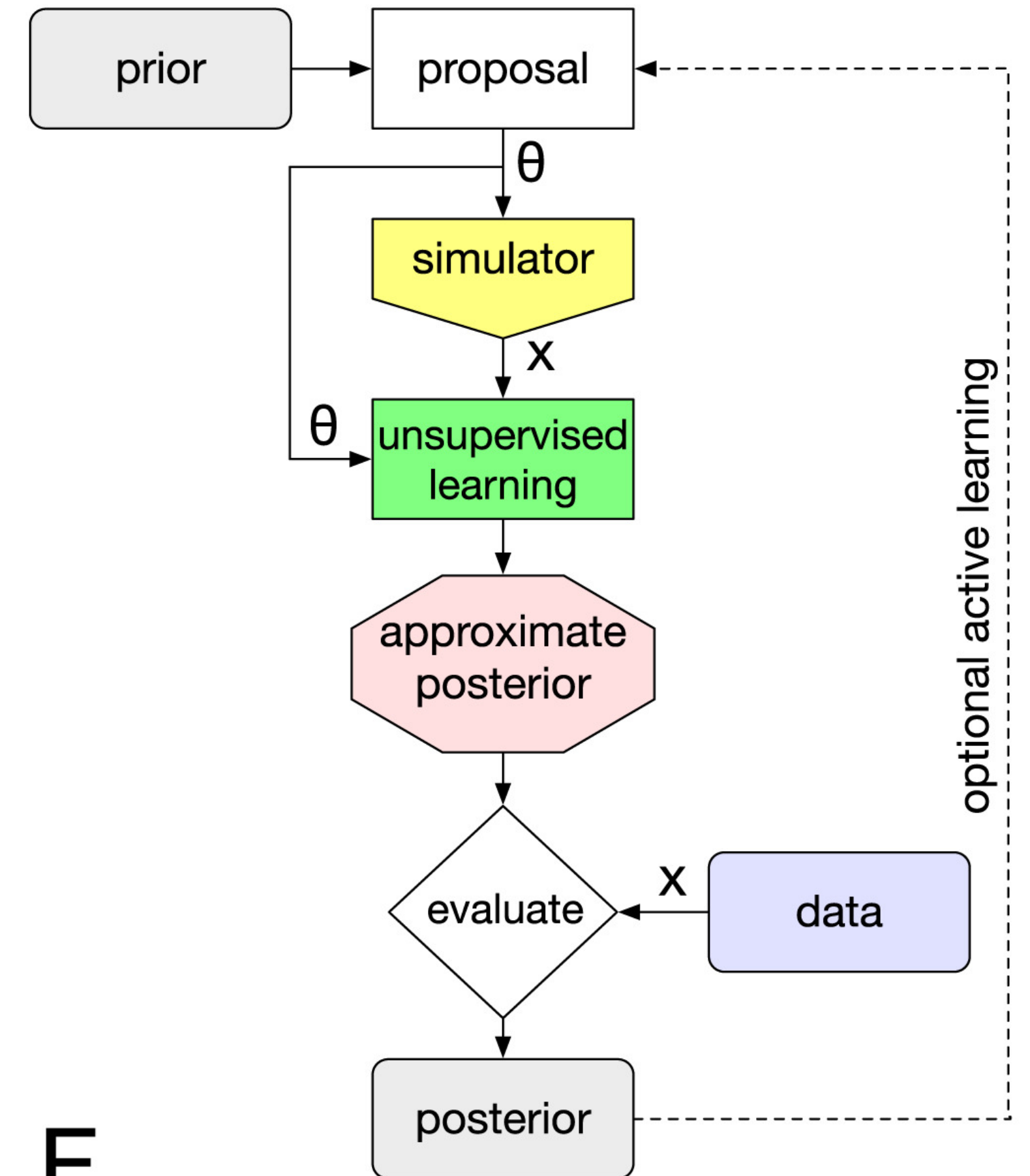
**B**

Cranmer+20

# Active learning

Sampling where it matters

## Amortized posterior



**F**

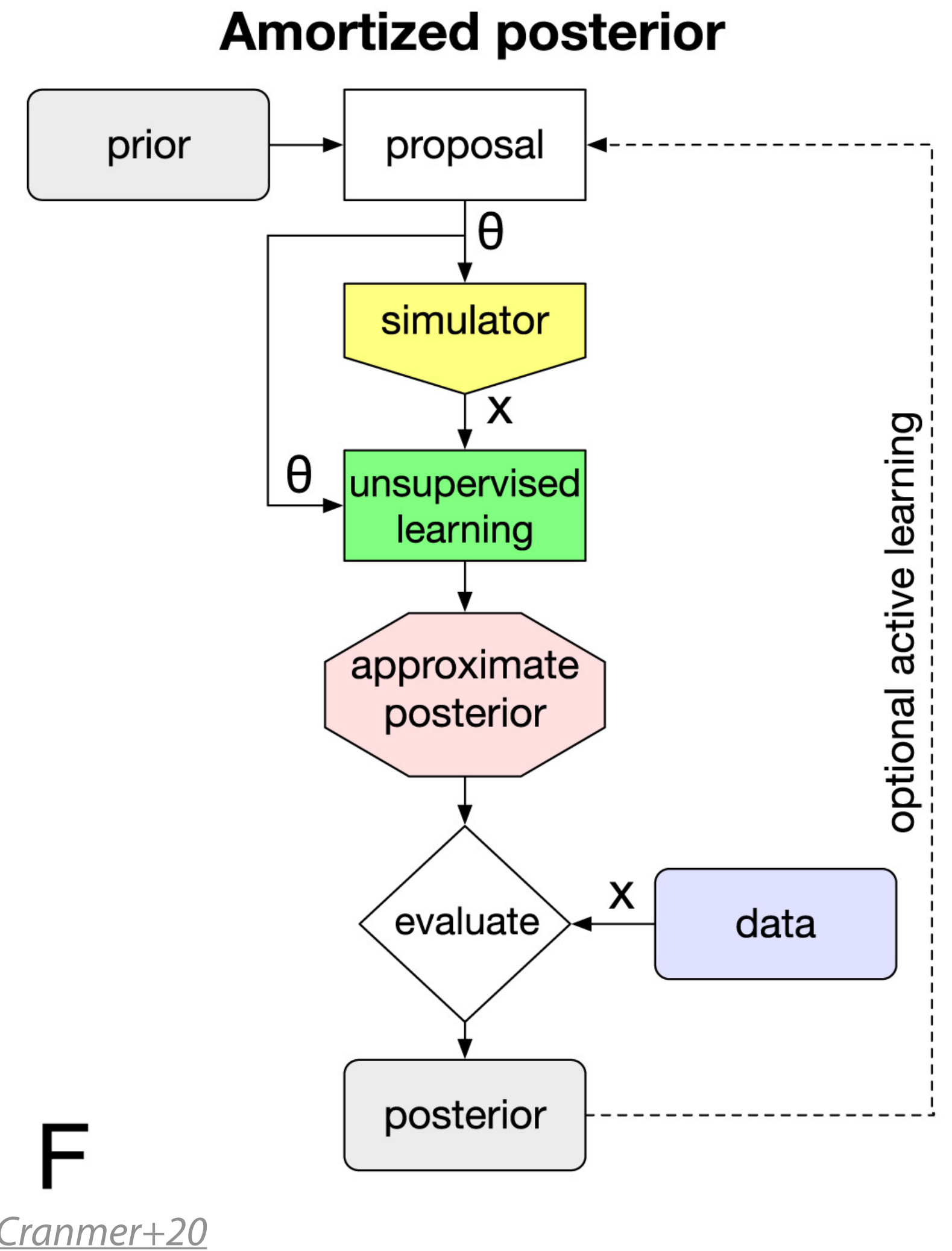
*Cranmer+20*

# Active learning

Sampling where it matters

## ▸ General idea

- Acquire new data (ie simulations for SBI) where NF model errors matter more, ie
  - where NF model is most *uncertain*
  - where NF model is most *likely*



# Active learning

Sampling where it matters

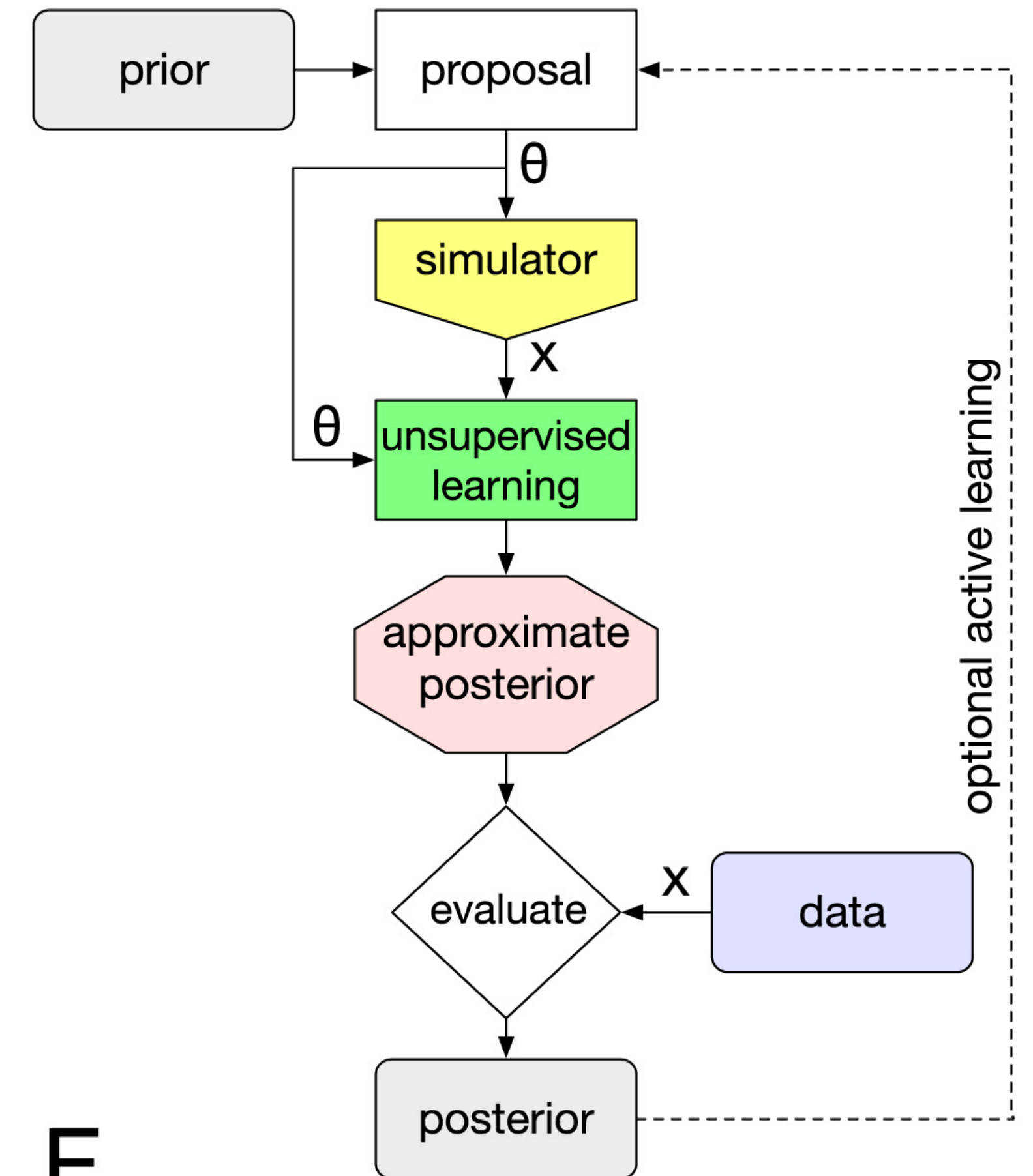
## ► General idea

- **Acquire new data** (ie simulations for SBI) **where NF model errors matter more**, ie
  - where NF model is most *uncertain*
  - where NF model is most *likely*

## ► In practice

- **Many strategies**: uncertainty sampling, adversarial sampling, discriminative sampling
- Depends on your sims...

## Amortized posterior



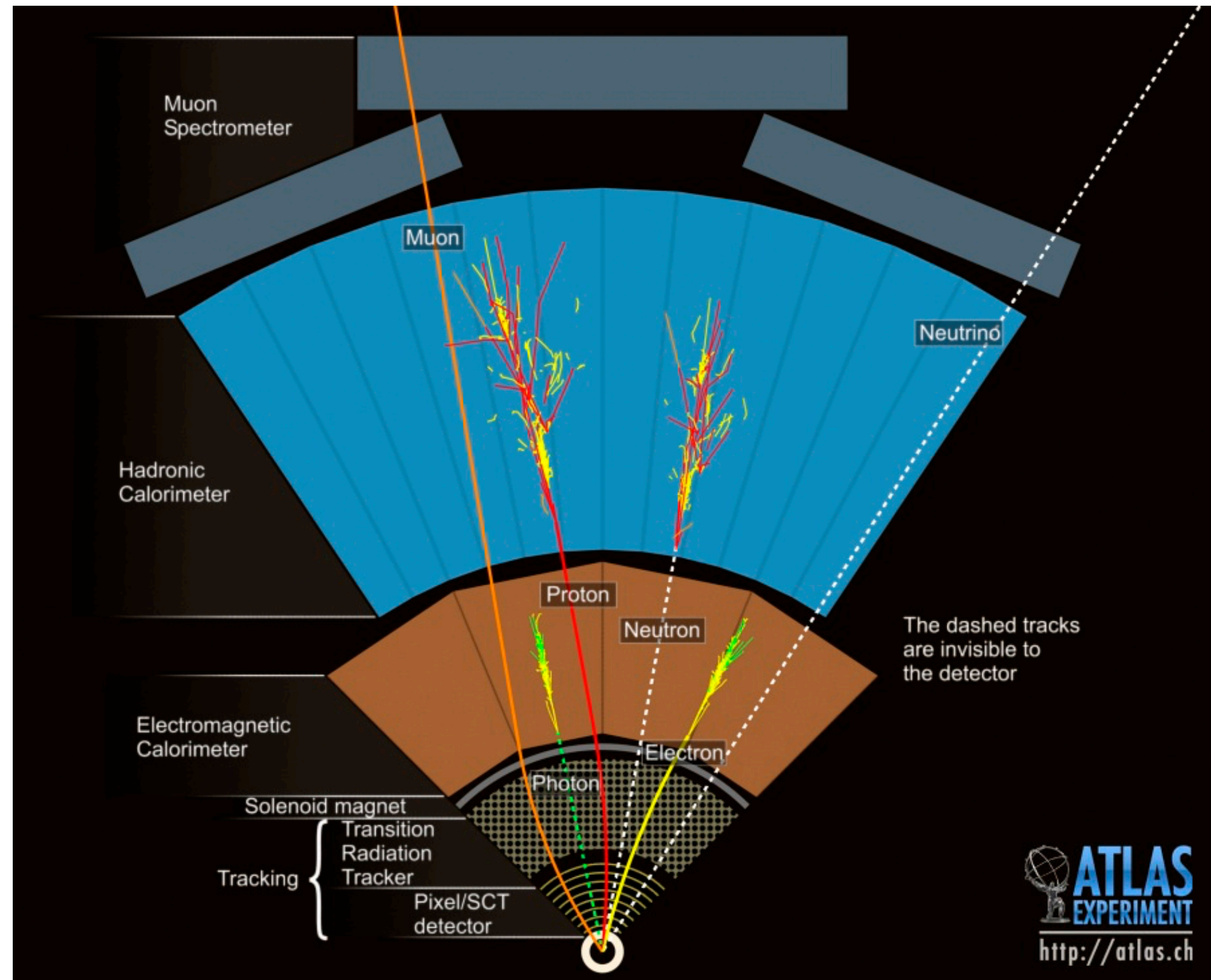
F

Cranmer+20

# Fiddling with the simulator

Faster, better, stronger

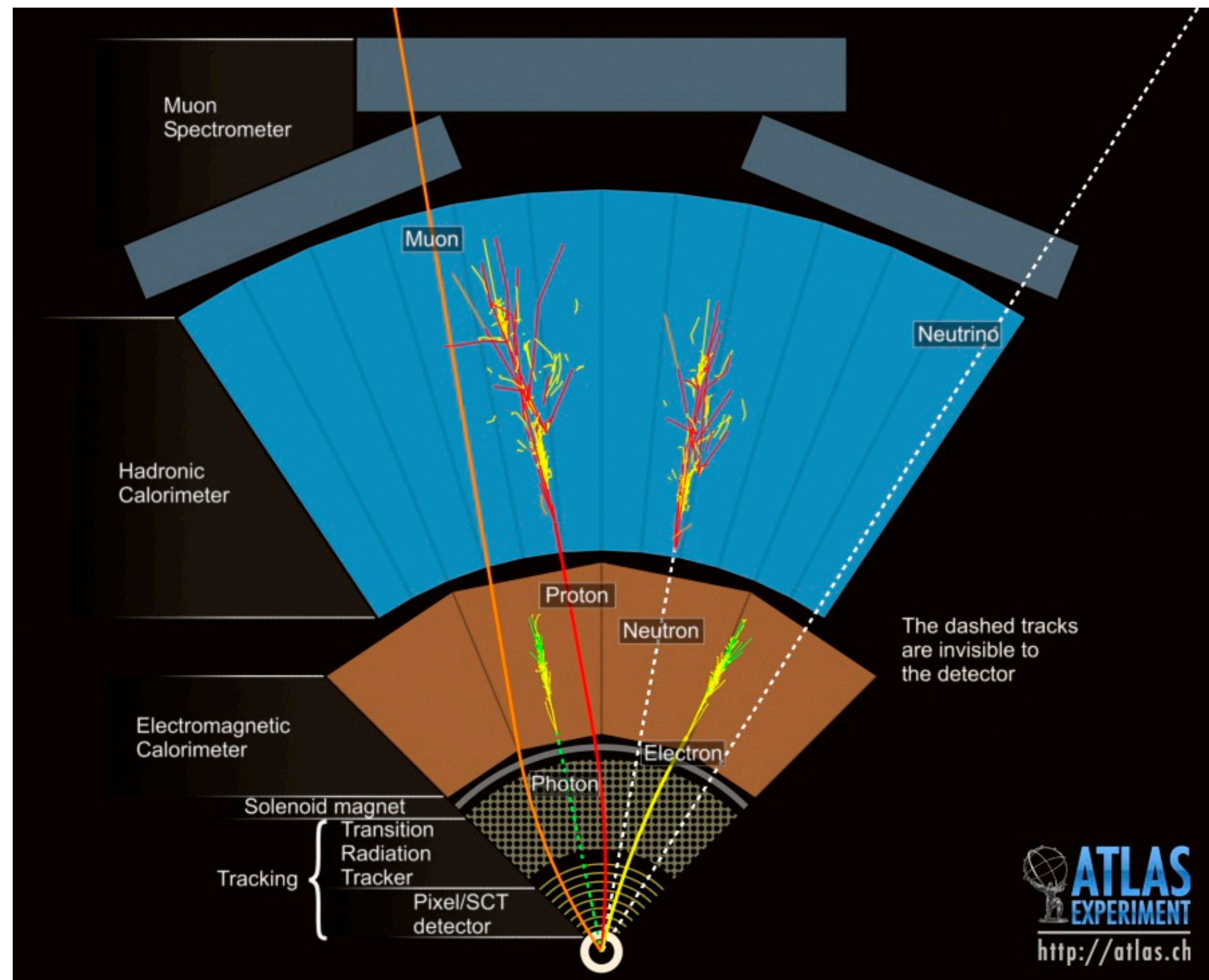
Monte-Carlo simulation of an event in the ATLAS detector



# Fiddling with the simulator

Faster, better, stronger

Monte-Carlo simulation of an event in the ATLAS detector



## ▸ Probabilistic programming

- Alter simulator sampling distributions ( $\theta, z, x$ )

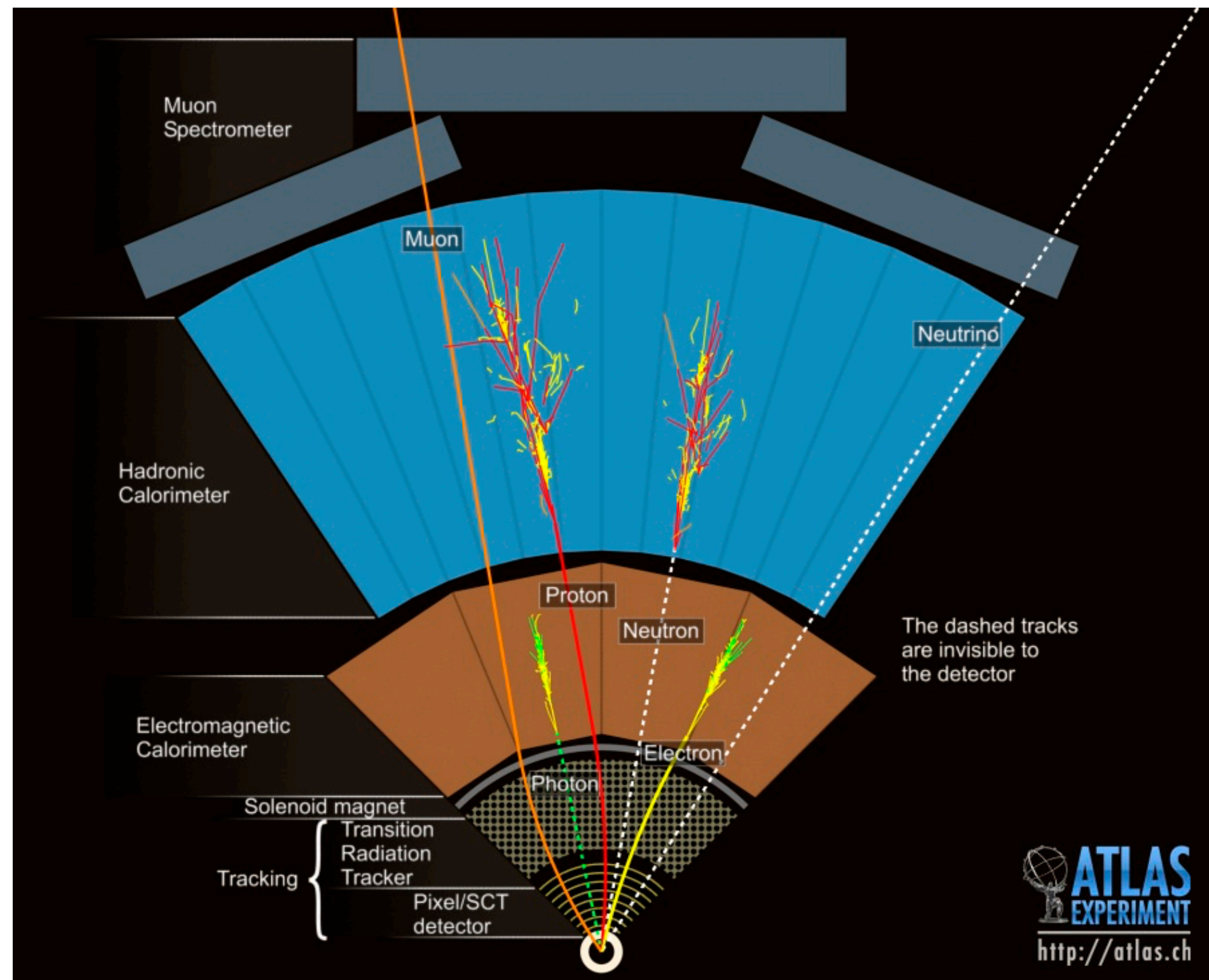
- Example: infer latent variables  $z$  to target simulations even better (akin to active learning)

- **Data augmentation**: simple transformations to generate more *pseudo*-independent data

# Fiddling with the simulator

Faster, better, stronger

Monte-Carlo simulation of an event in the ATLAS detector



## ▸ Probabilistic programming

- Alter simulator sampling distributions  $(\theta, z, x)$ 
  - Example: infer latent variables  $z$  to target simulations even better (akin to active learning)
- **Data augmentation**: simple transformations to generate more *pseudo*-independent data

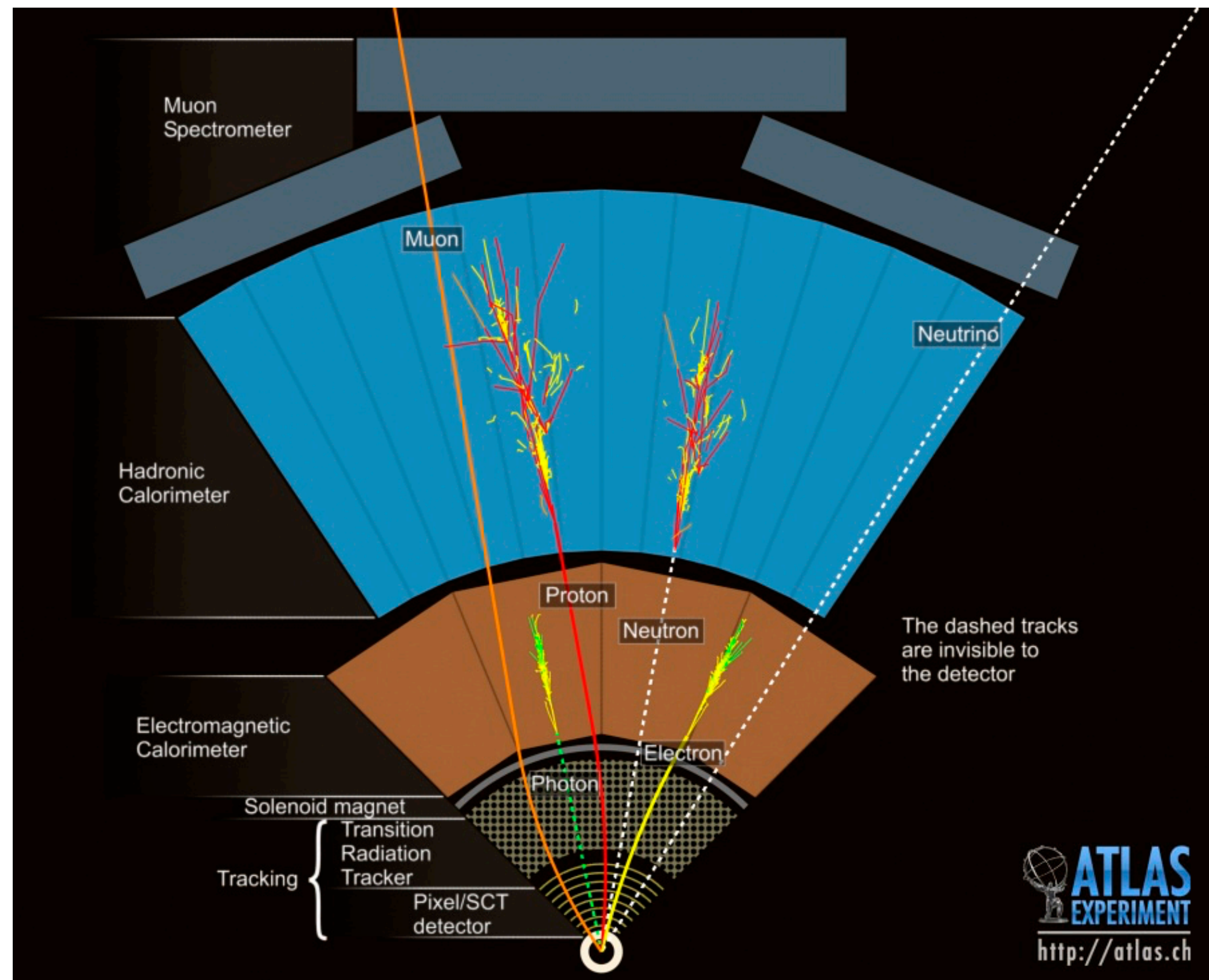
## ▸ Differentiable simulations

- Provides **gradients**, e.g., the score  $\nabla_{\theta} \log \mathcal{L}(X|\theta)$
- Allows faster sampling, e.g. HMC, and improved NF fit

# Fiddling with the simulator

Faster, better, stronger

Monte-Carlo simulation of an event in the ATLAS detector



## ▸ Probabilistic programming

- Alter simulator sampling distributions  $(\theta, z, x)$ 
  - Example: infer latent variables  $z$  to target simulations even better (akin to active learning)
- **Data augmentation**: simple transformations to generate more *pseudo*-independent data

## ▸ Differentiable simulations

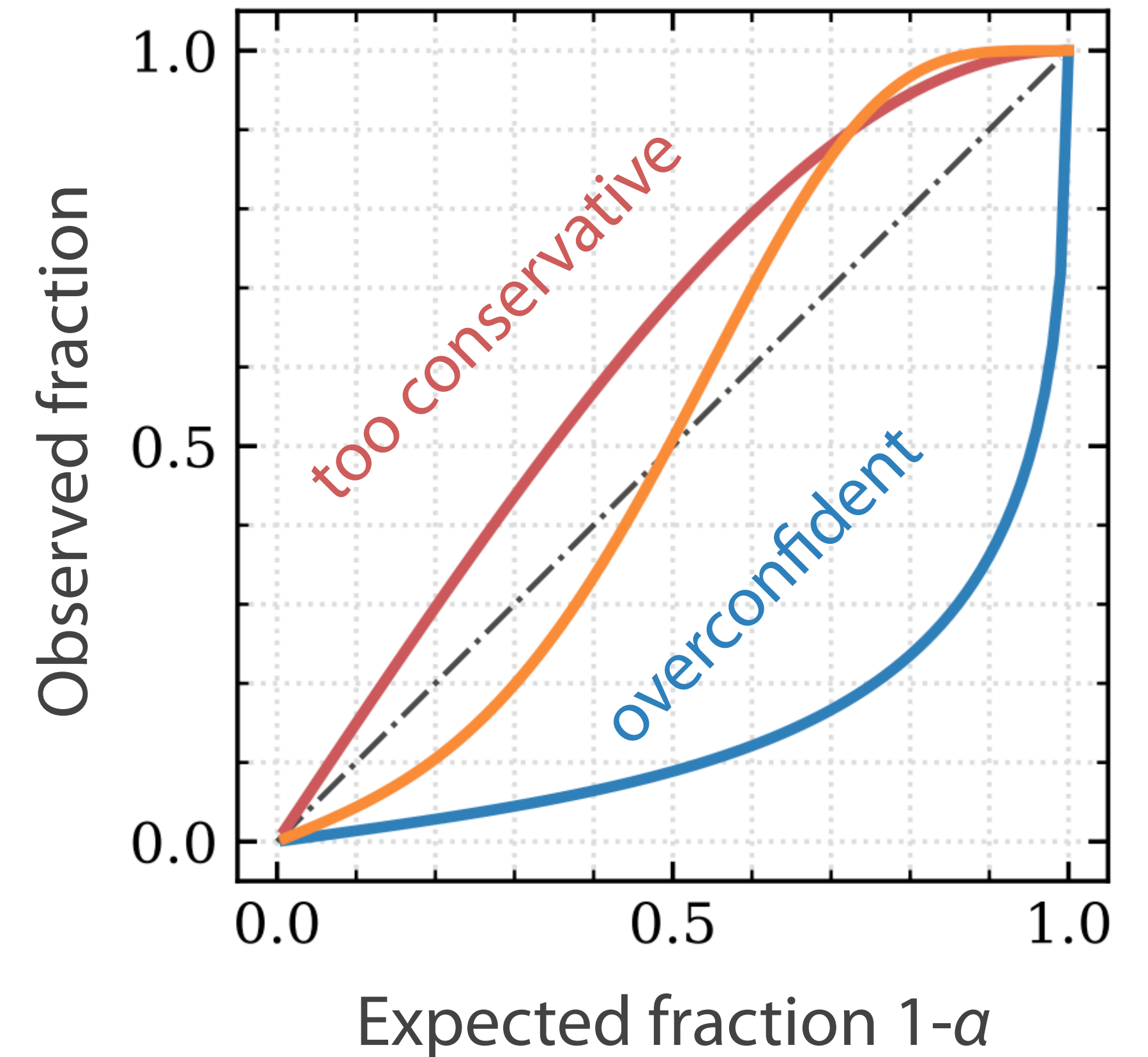
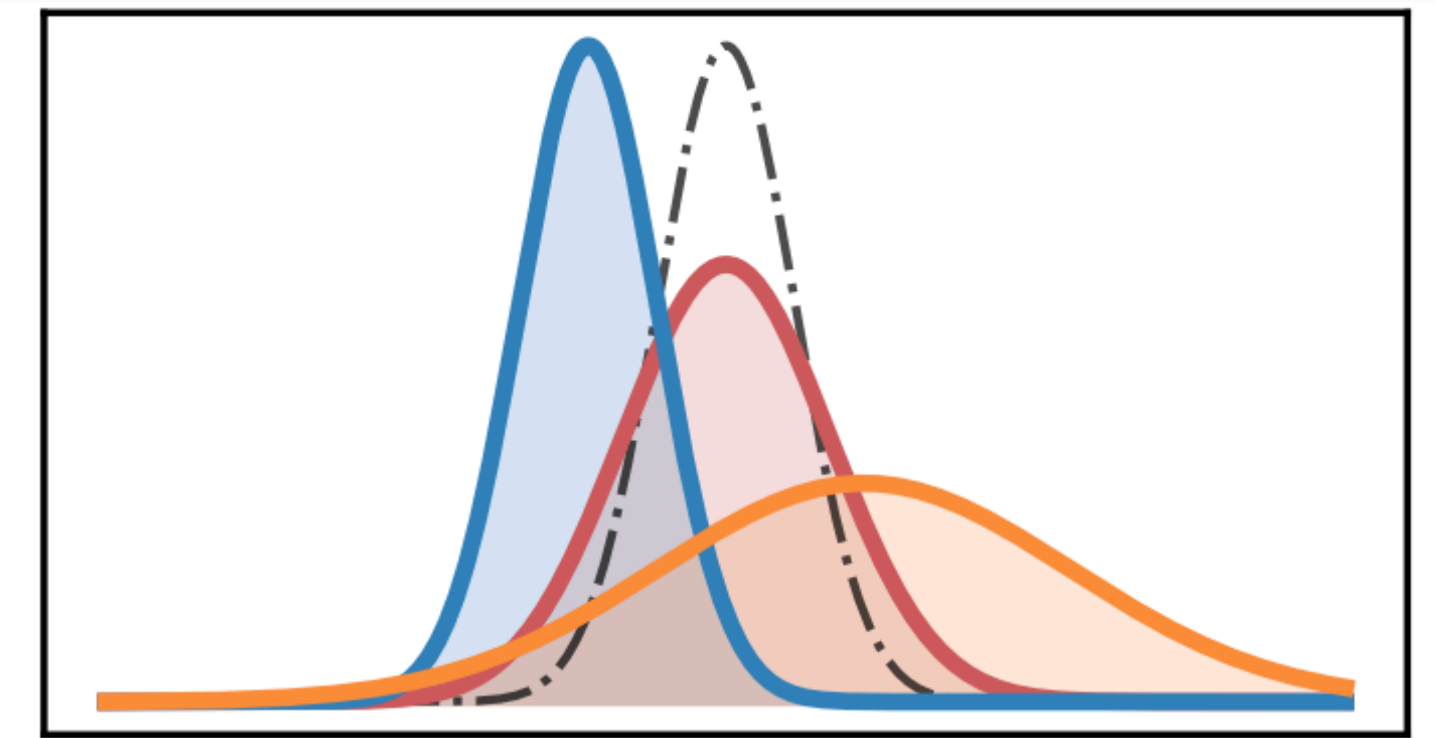
- Provides **gradients**, e.g., the score  $\nabla_{\theta} \log \mathcal{L}(X|\theta)$
- Allows faster sampling, e.g. HMC, and improved NF fit

## ▸ Multi-resolution simulators

- Combine many fast low-res sims with few high-res sims...

# Coverage diagnostic

How accurate are posterior distributions?



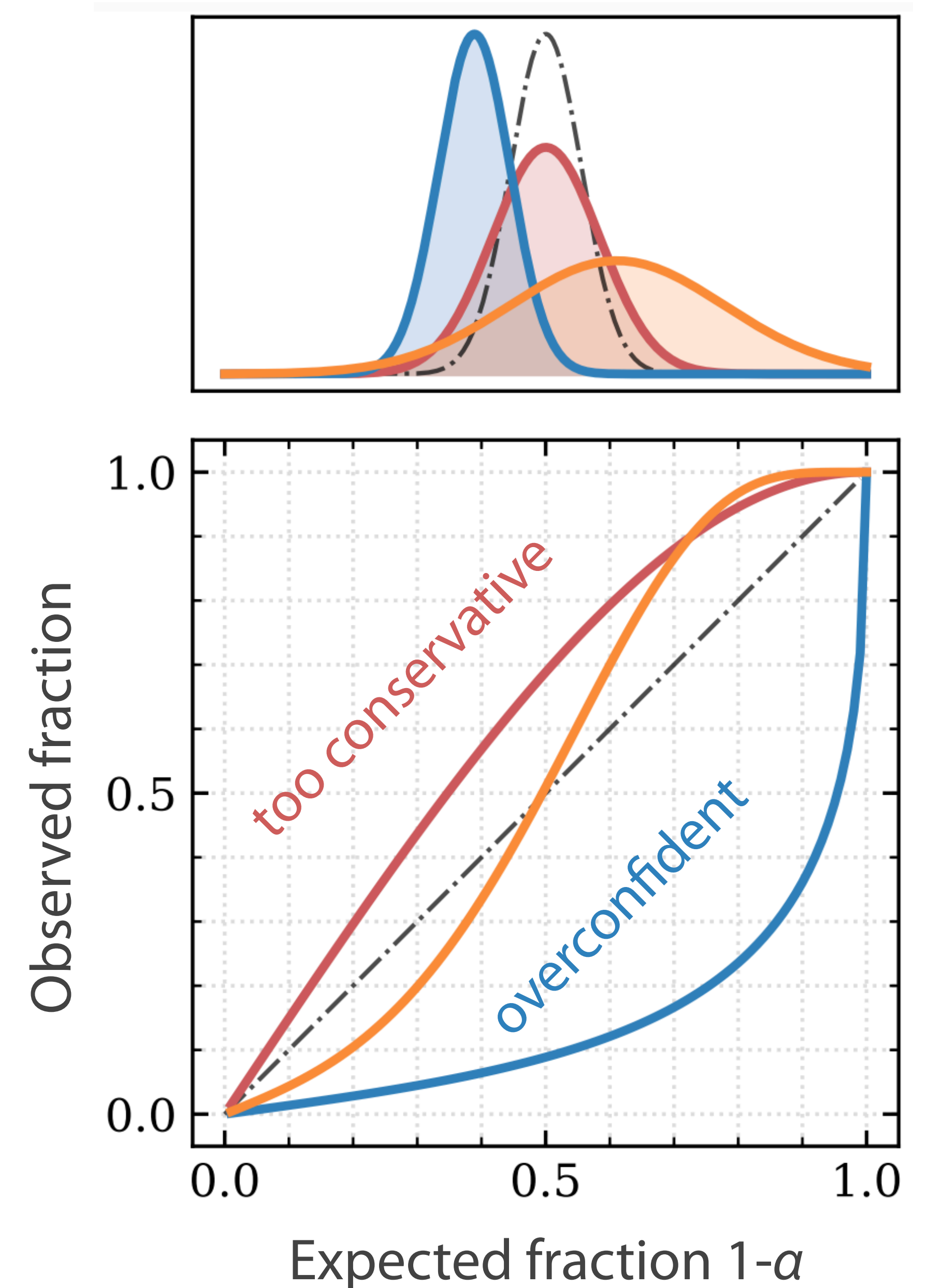
*Cannon+22*

# Coverage diagnostic

How accurate are posterior distributions?

## ► Empirical coverage test

1. Given a test set  $(\theta_i, x_i) \sim \pi(\theta)\mathcal{L}(x|\theta)$ , confidence level  $\alpha$  and neural posterior estimator  $q(\theta|x)$
2. Compute fraction of  $(\theta_i, x_i)$  that lie in the  $1-\alpha$  highest confidence region of  $q$
3. Should be  $1-\alpha$  !



*Cannon+22*

# Coverage diagnostic

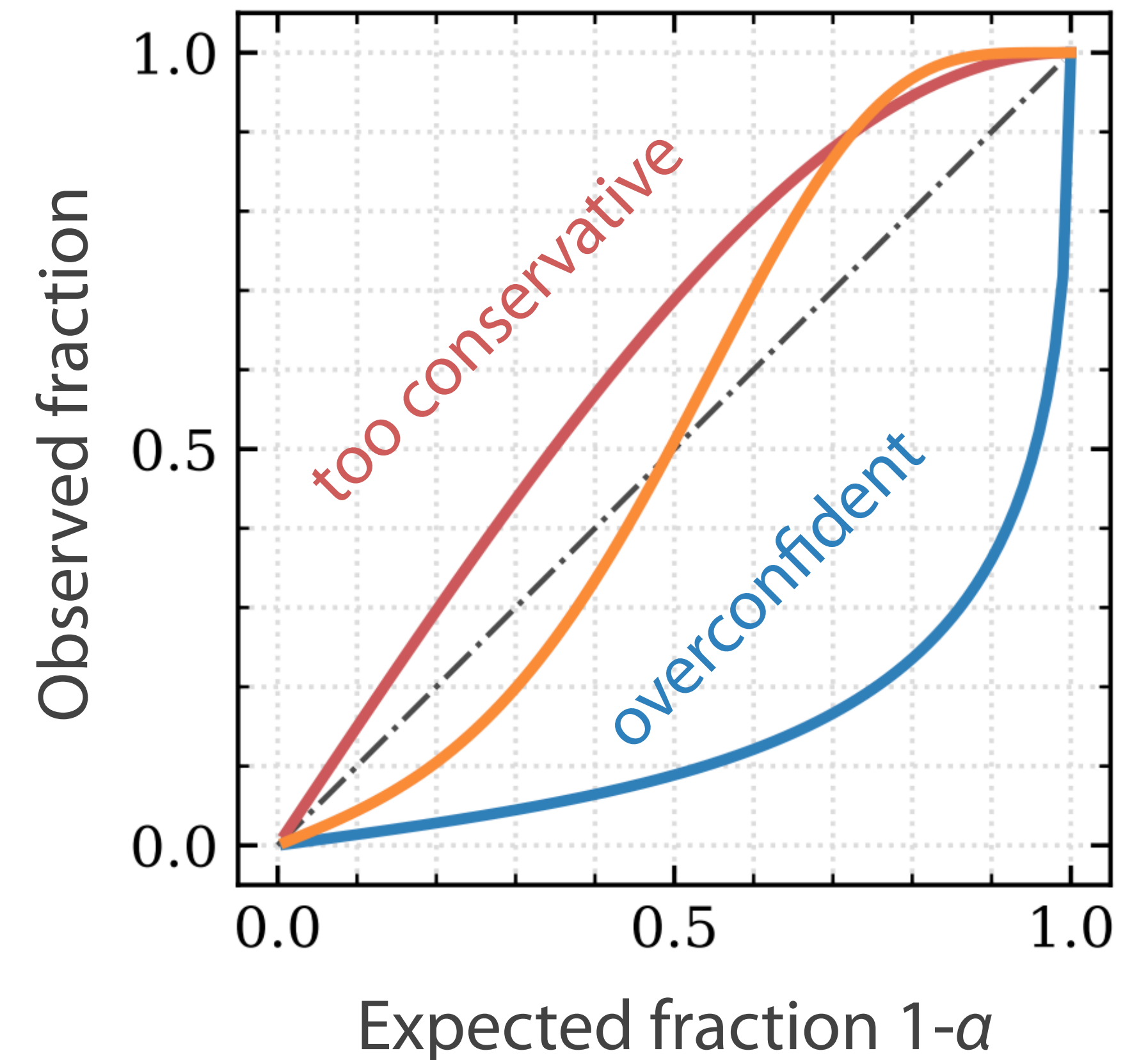
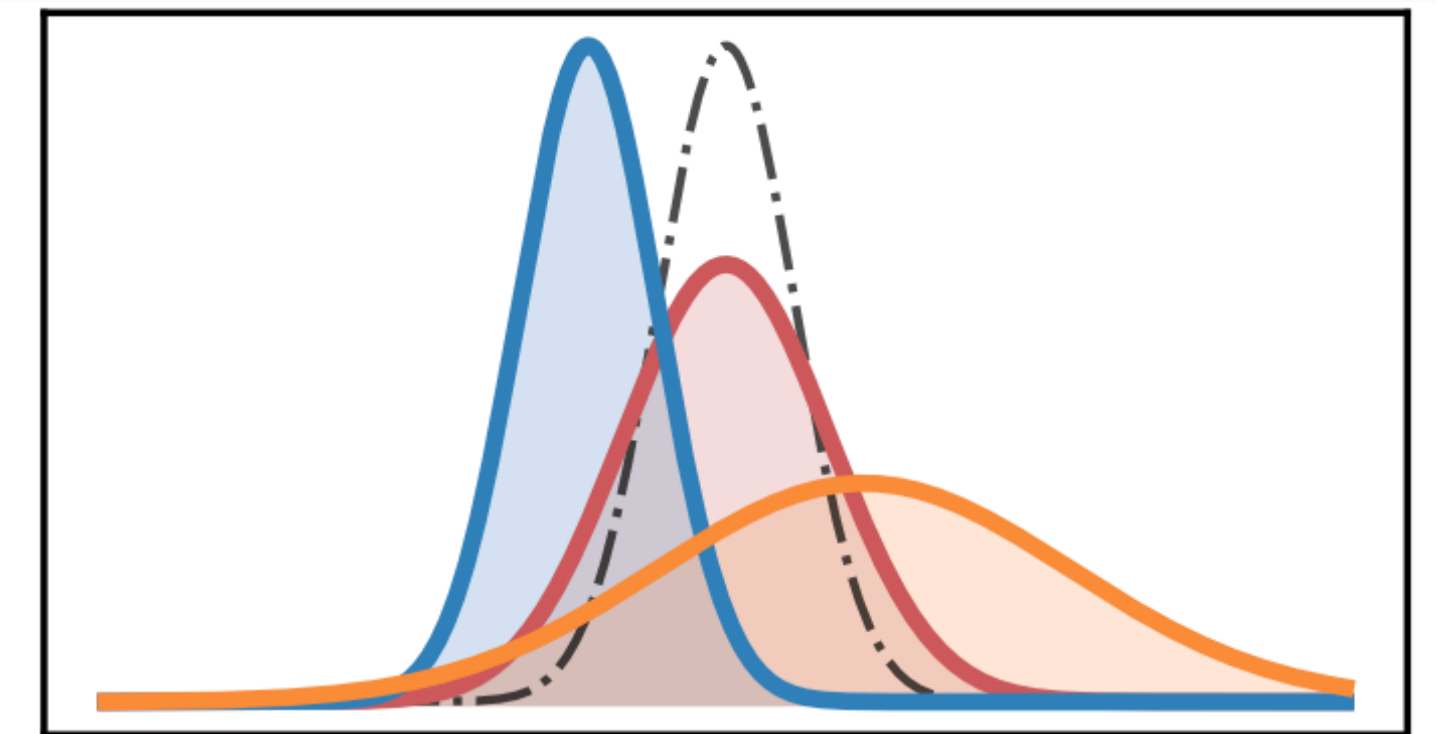
How accurate are posterior distributions?

## ► Empirical coverage test

1. Given a test set  $(\theta_i, x_i) \sim \pi(\theta)\mathcal{L}(x|\theta)$ , confidence level  $\alpha$  and neural posterior estimator  $q(\theta|x)$
2. Compute fraction of  $(\theta_i, x_i)$  that lie in the  $1-\alpha$  highest confidence region of  $q$
3. Should be  $1-\alpha$ !

## ► What to change if test fails?

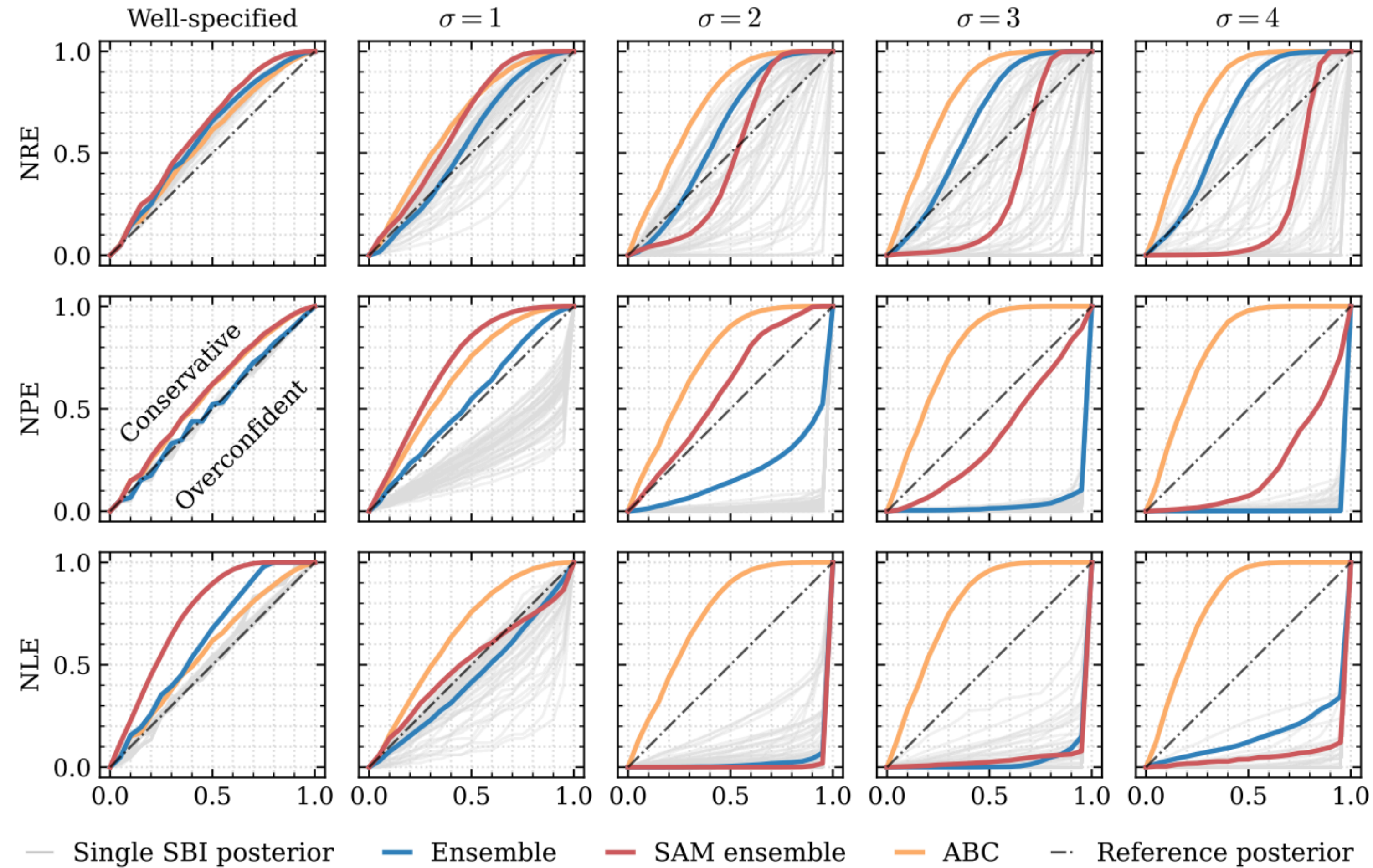
- Number of simulations
- Sampling of sims in params space
- Model architecture, flexibility (over/under-fitting or biases)



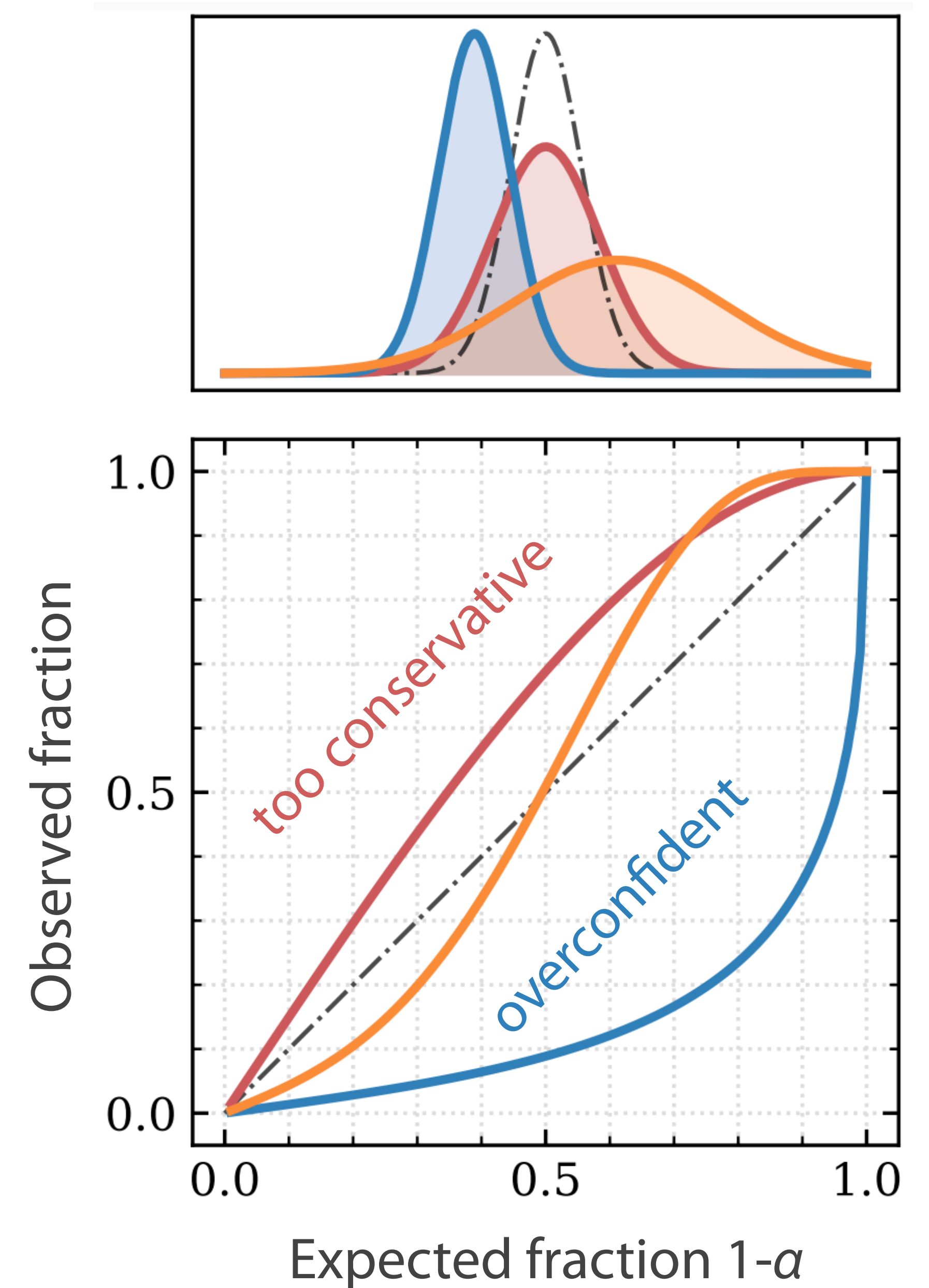
*Cannon+22*

# Coverage diagnostic

How accurate are posterior distributions?



*Cannon+22*



*Cannon+22*

# Summary

1. SBI = techniques to infer model parameters when the likelihood may only be sampled by simulator
2. SBI is not new: ABC has been there for 40 years
3. What's new is applying machine-learning to
  - learn (amortized) distributions or likelihood ratios
  - summarize data
  - generalize to high-dimensional and complex models
4. SBI is an active field of research: diagnostics/calibration, active learning, tight integration of simulator/inference, etc.

# References

- Websites

- <https://simulation-based-inference.org/>: applications of SBI in various scientific fields
- <https://lilianweng.github.io/posts/2018-10-13-flow-models/>

- Review papers

- Cranmer, K., Brehmer, J. & Louppe, G. The frontier of simulation-based inference. *Proc. Natl. Acad. Sci.* **117**, 30055–30062 (2020).
- Johann Brehmer, Kyle Cranmer *Simulation-based inference methods for particle physics* [arXiv:2010.06439](https://arxiv.org/abs/2010.06439)
- Links in the slides...