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 - May 17th 2024

Prerequisites

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

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

Copilot prompt:particle physics and cosmology simulations flowing into neural networks like Basquiat with white background

Starting point: the intractable likelihood problem

- **Two main roads to statistical inference**
  - Frequentist
    - Assume there exists some true θ^* value
    - Use likelihood (ratios) to compute confidence intervals $I(\text{data}, \alpha)$, such that $P(\theta^* \in I(\text{data}, \alpha)) = \alpha$
  - Bayesian
    - θ are themselves random variables, with marginal distribution $P(\theta)$ called the prior
    - Bayes' theorem uses data to update the distribution, $P(\theta|\text{data}) \propto \mathcal{L}(\text{data}|\theta) \times \mathrm{P}(\theta)$,* *ie* the posterior
- **Same key ingredient: the likelihood $\mathcal{L}(\text{data}|\theta)$**

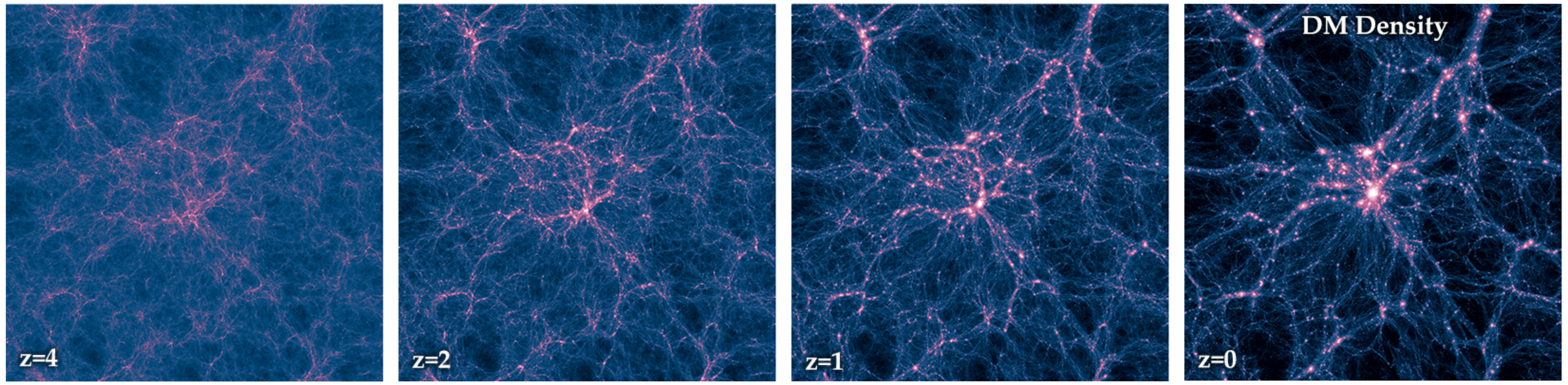
What if the likelihood is unknown? Or intractable? Or with no good approximation? 🤯

* 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.

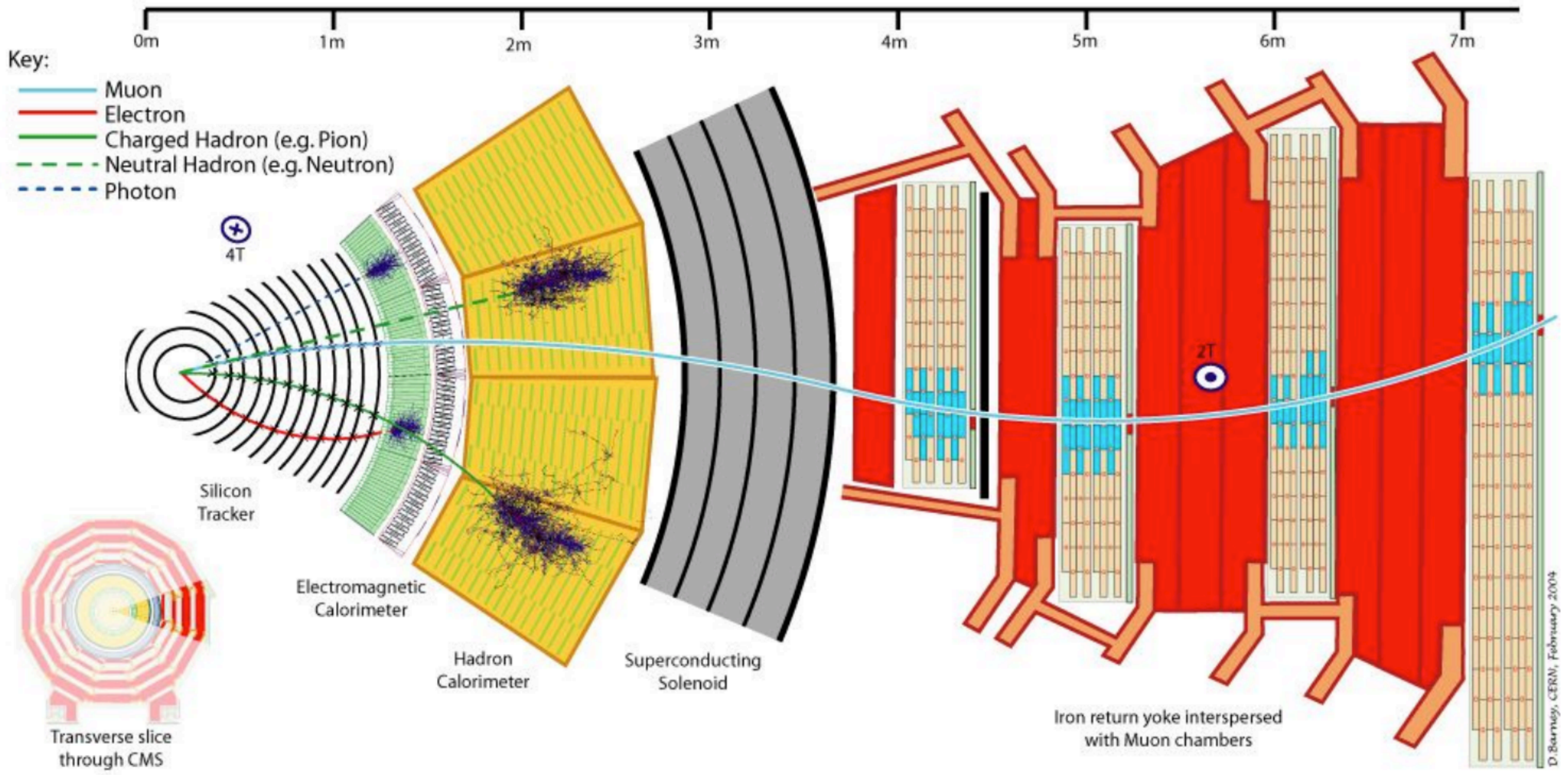
Why would the likelihood not be "tractable"?

- **Likelihood of observables from complex (stochastic) processes**
  - Elementary processes may be described *via* analytic theory (QFT, GR), but…
  - Final observables involves many processes, potentially non-linear, and convolved with instrumental signature
  - So they may not be modeled (semi-)analytically, ie **one cannot *write* a likelihood function** 😵‍💫
- **Examples**

Cosmological structure evolution (Illustris simulation)



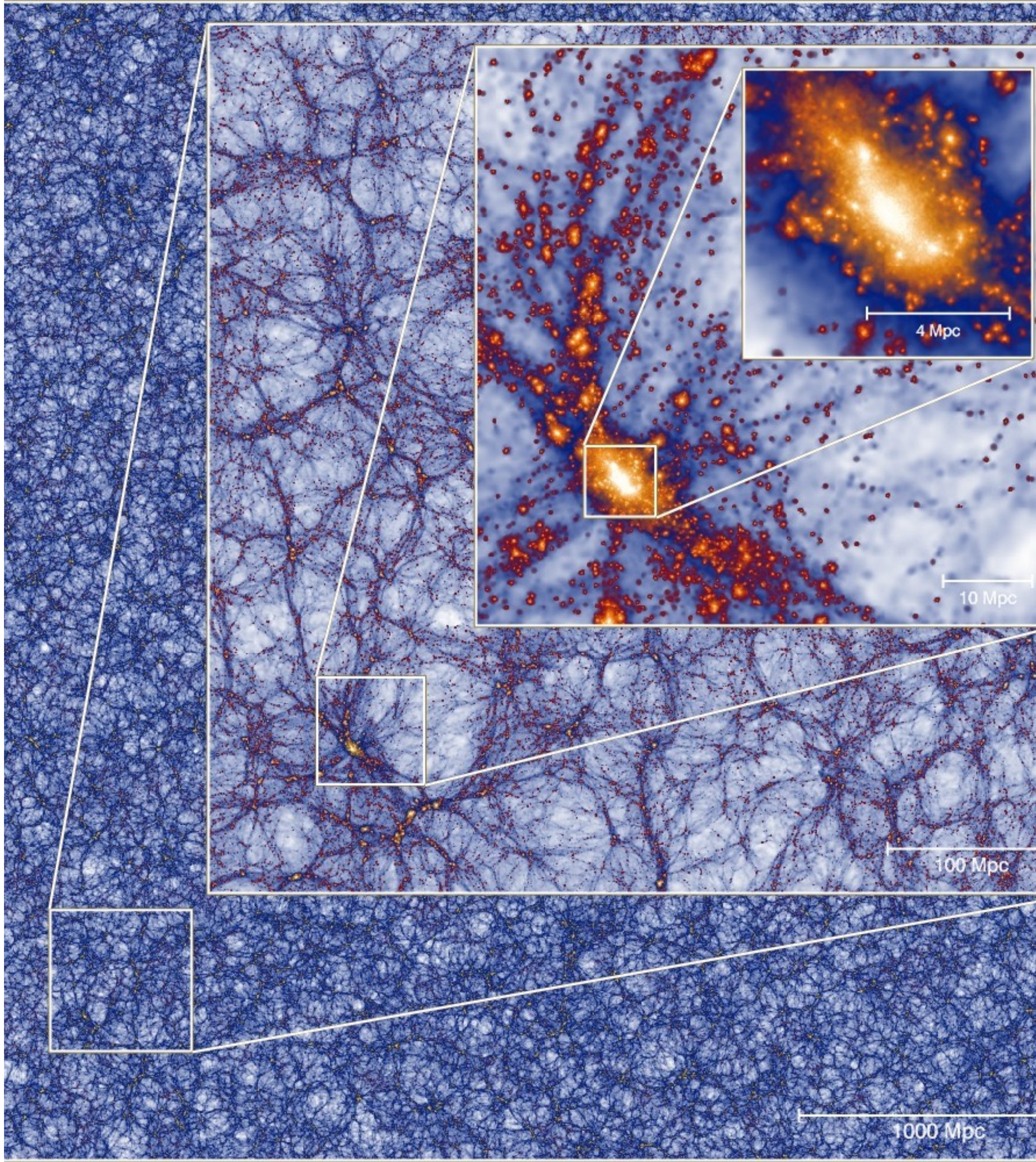
Event decay through CMS detector at CERN



Simulations

We can generate data! 😎

- Why we generate sims
  1. Theoretical predictions testing
  2. Calibration of systematics
  3. End-to-end pipeline validation
- Simulations model complex physics with a *bottom-up* approach from well-understood elementary (stochastic) processes
- Observational/instrumental signature can be simulated on top of fundamental physics 🤘
- Accuracy/resolution/amount is increasing with computing resources/techniques (for now)



Simulation-based inference

So what are we talking about?

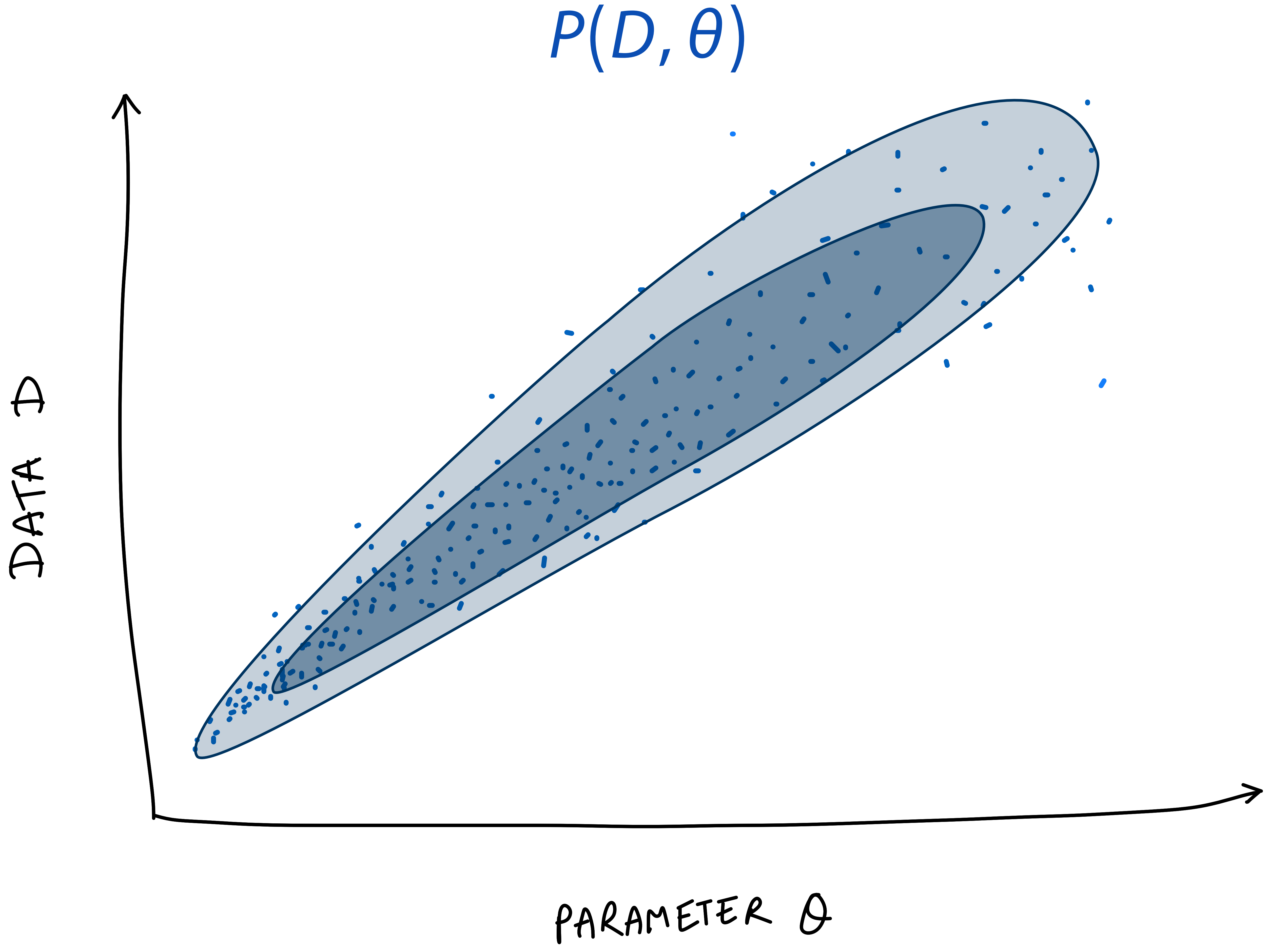
- **Definitions and vocabulary**
  - SBI = methods constraining the parameters of a statistical model described by a likelihood $\mathcal{L}(\text{data}|\theta)$ *that may only be sampled through simulations*
  - SBI = *likelihood-free* inference = *implicit likelihood* inference
- **In SBI, simulations replace (semi-)analytic models**
  - In some fields (e.g., LHC stuff), inference is already driven by simulations
  - Pros and cons: do we trust simulations? More later.
- **How can SBI accomplish such a feat?** 🧙
  - Observed data will be compared to simulated data *generated* at various parameter values
  - Mathematically sound approach, now using generative, deep-learning models

Basic idea

First, without the maths

- Assume
  - We only have a simulator to generate data at any model parameter value
- Goal
  - Infer constraints on model parameters
- Recipe

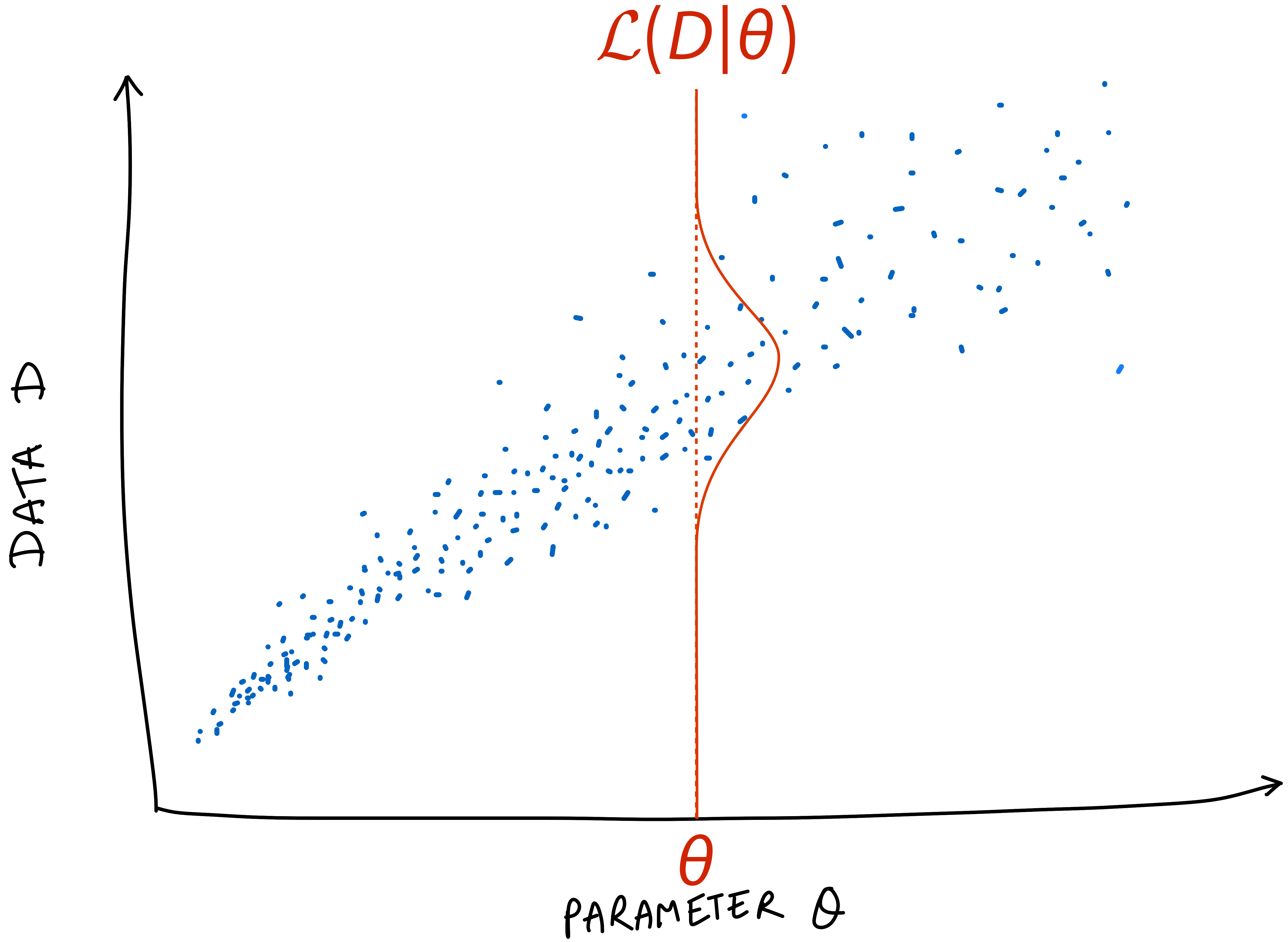
1. Run simulator at various parameter θ



Basic idea

First, without the maths

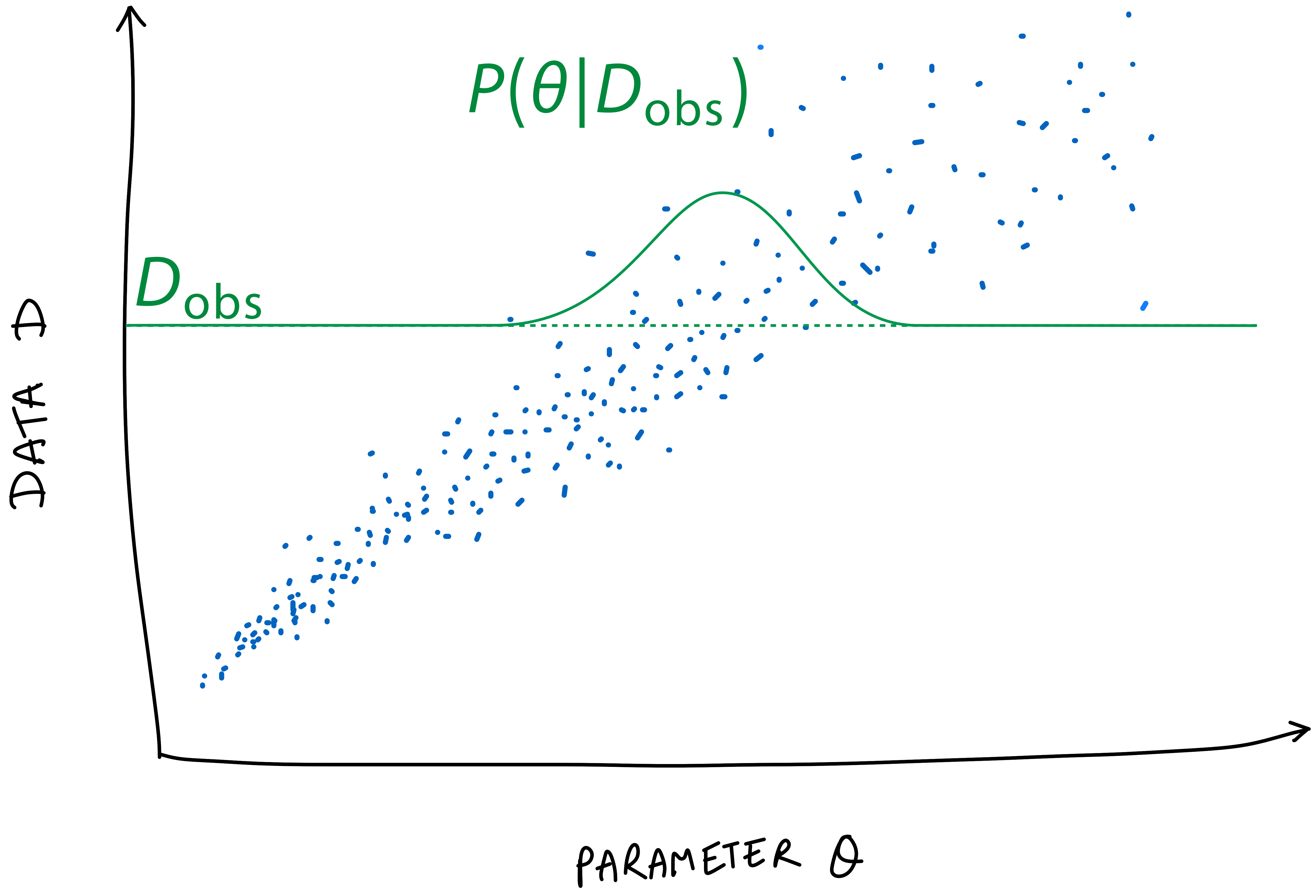
- Assume
  - We only have a simulator to generate data at any model parameter value
- Goal
  - Infer constraints on model parameters
- Recipe
  1. Run simulator at various parameter θ
  2. Learn the likelihood



Basic idea

First, without the maths

- Assume
  - We only have a simulator to generate data at any model parameter value
- Goal
  - Infer constraints on model parameters
- Recipe
  1. Run simulator at various parameter θ
  2. Learn the likelihood…
  3. or the posterior…

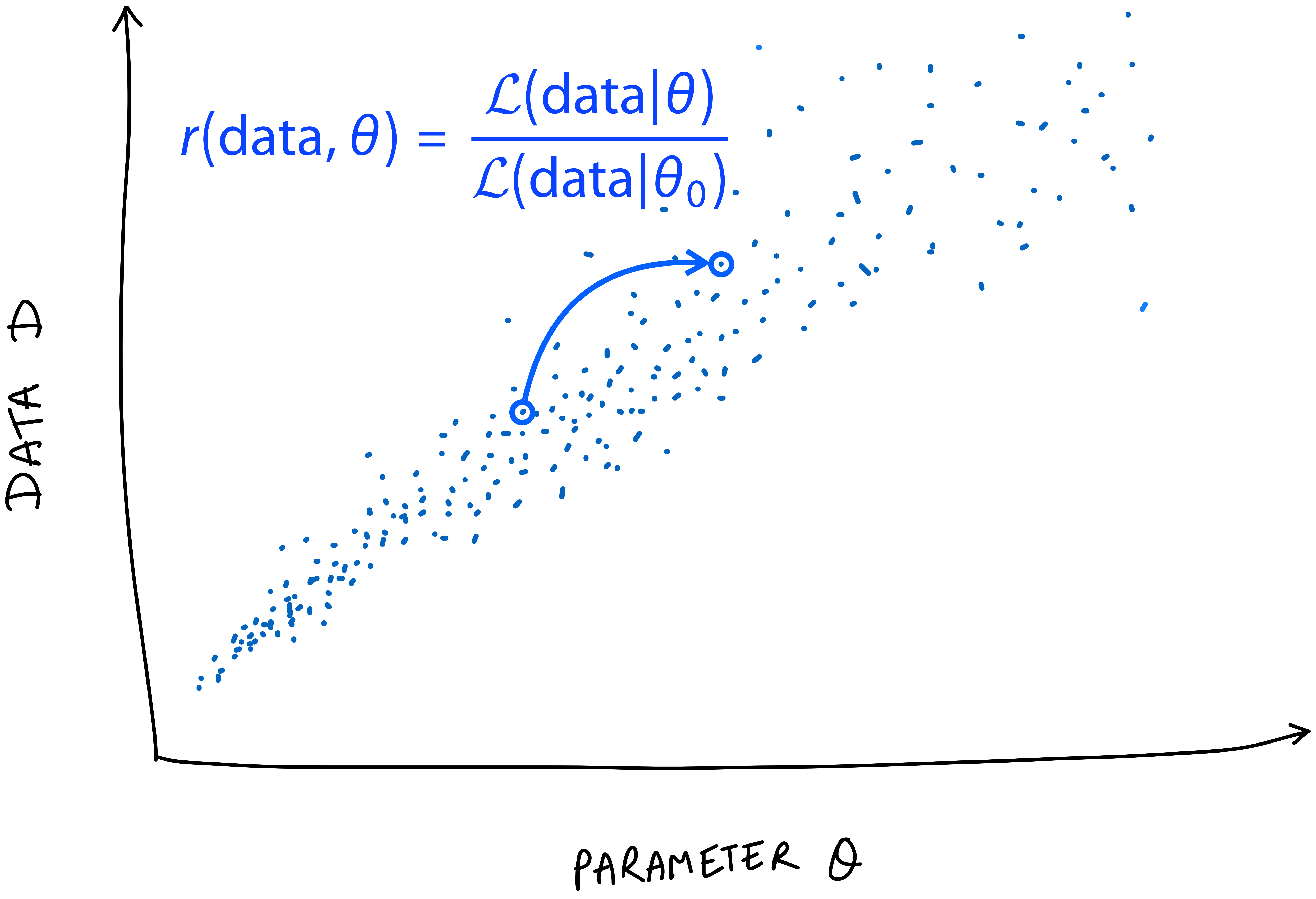


Basic idea

First, without the maths

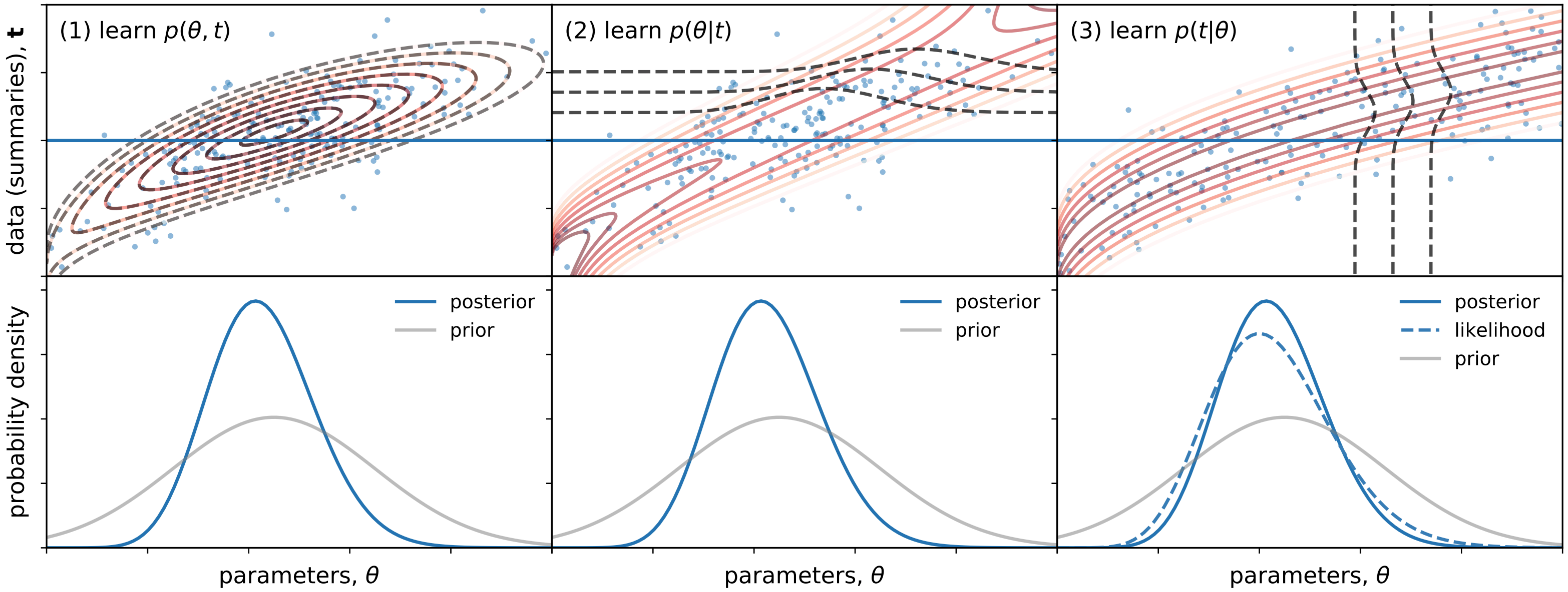
- Assume
  - We only have a simulator to generate data at any model parameter value
- Goal
  - Infer constraints on model parameters
- Recipe
  1. Run simulator at various parameter θ
  2. Learn the likelihood…
  3. or the posterior…
  4. or likelihood ratios.

That's it!*



*well, not really

Joint *vs* posterior *vs* likelihood



Alsing+19

Pros and cons

SBI *vs* likelihood-based approach

- **Simulations *vs* analytical modelling trade-off**
  - Analytic modelling usually involves some theoretical approximations
  - Simulations model elementary processes, but have problems of their own: resolution, convergence, etc.
- **SBI does not approximate the likelihood**
  - No need to assume functional form, e.g., Gaussian or Poisson: very difficult to validate in practice
  - 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 in practice

Simulators, emulators and ABC

Copilot prompt: student learning approximate bayesian computation by the mediterranean sea, impressionism

Basic idea, now with the maths 😵

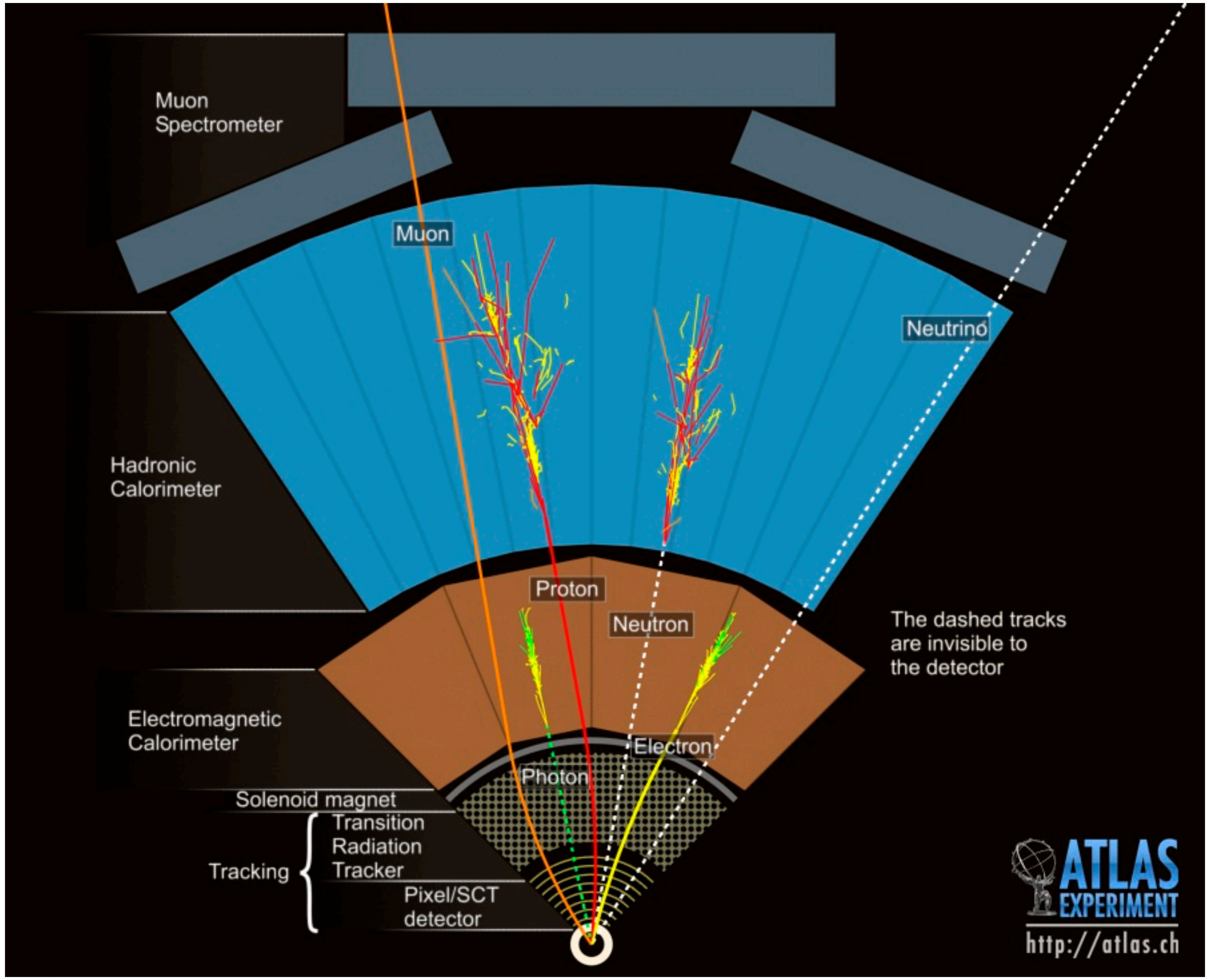
- **Notations**
  - Model parameters θ : physical and nuisance parameters
  - Latent variables z : internal, unobservable state of the system (see examples below)
  - Data x : simulated or observed
- **Simulator**
  - Given params, the simulator samples $z \sim P(z|\theta)$ and generates data $x \sim P(x|\theta, z)$
  - 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 $P(x|\theta) = \int \mathrm{d}z\, P(x, z|\theta)$, 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

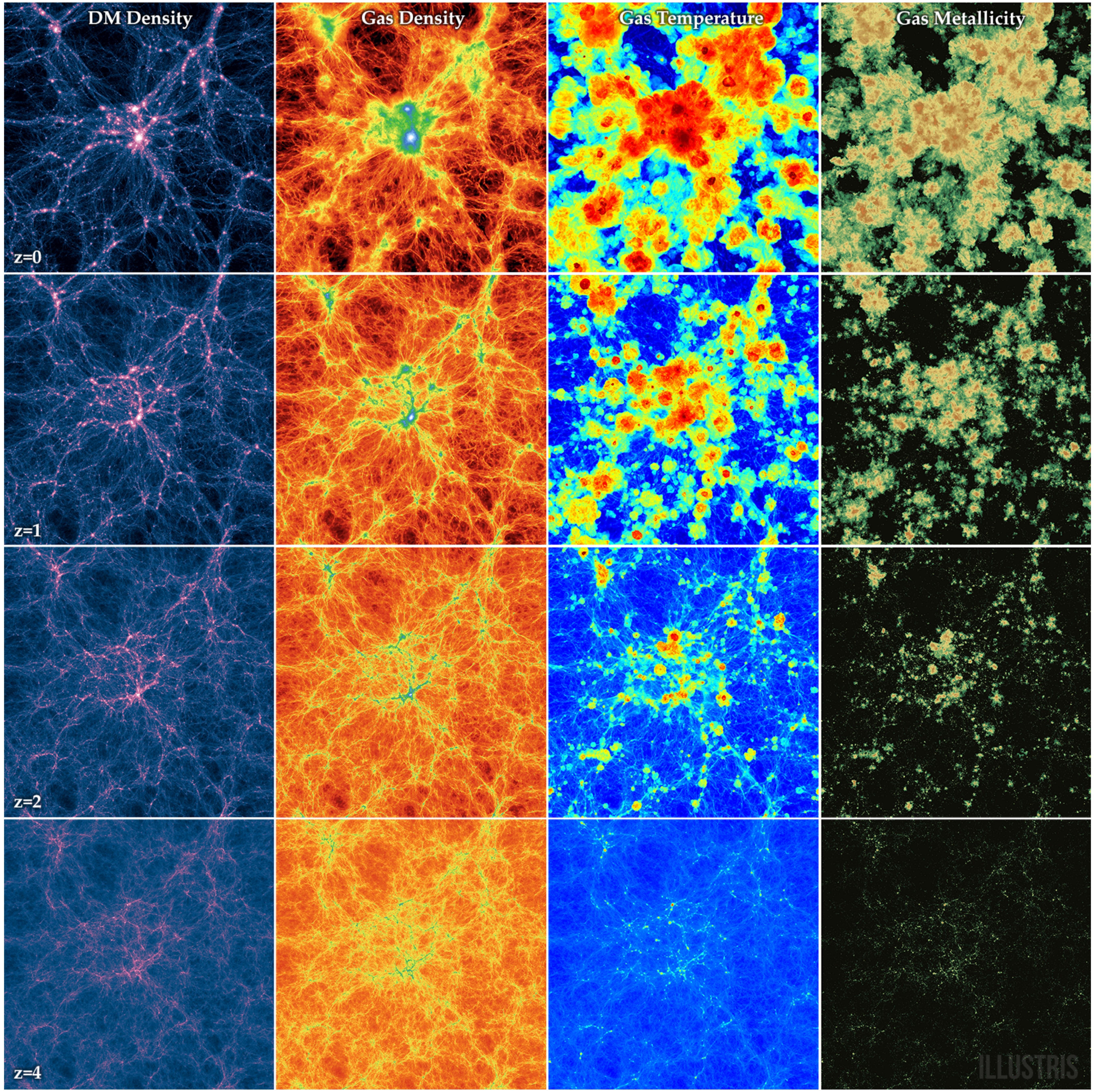


Simulator examples

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, $\mathcal{O}(10^8 s)$

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



Emulators/templates

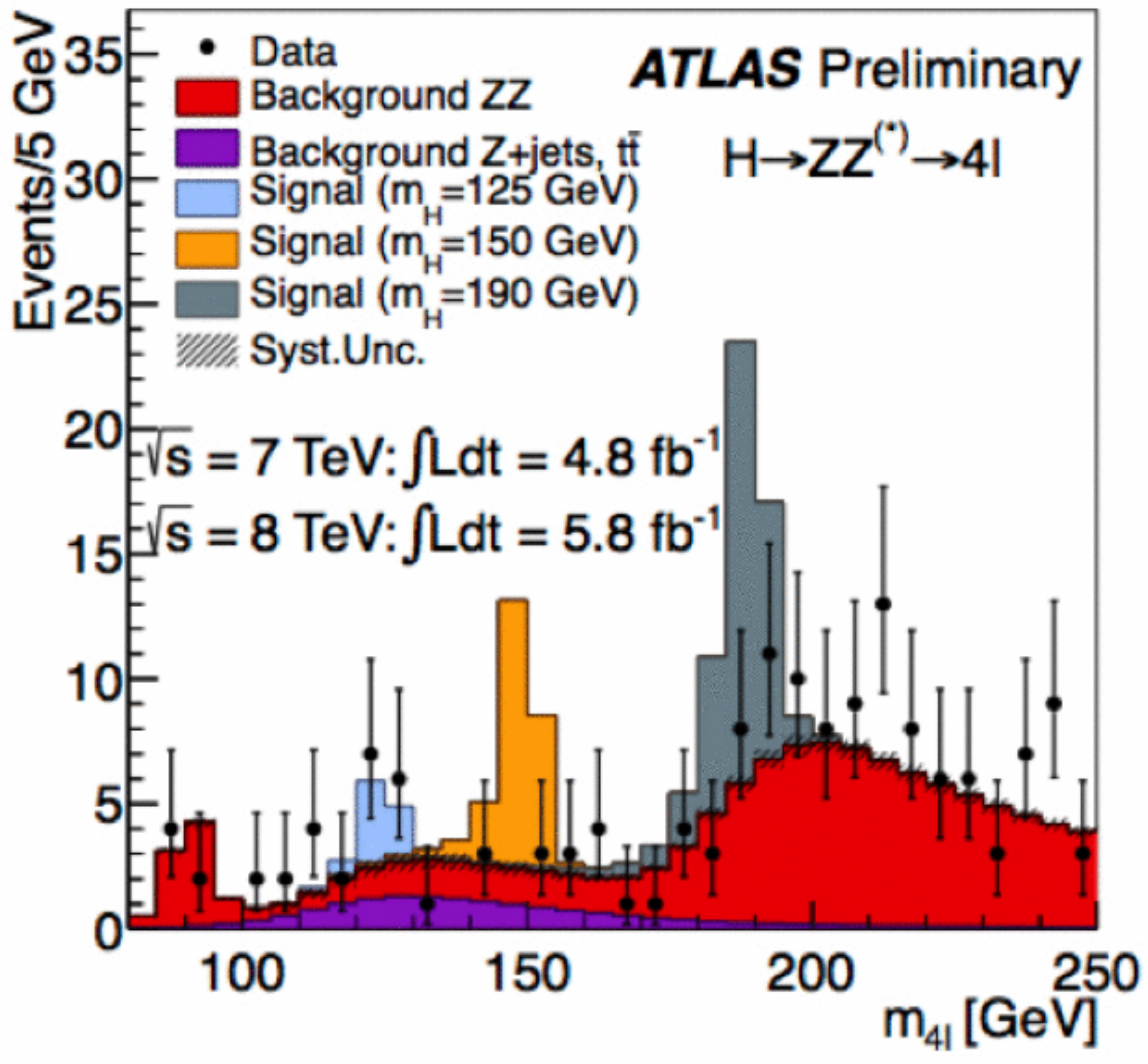
Not really what we mean by SBI 🤔

- **Likelihood approximation**
  - Standard form = mix of Poisson + Gaussian, eg

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

- **Emulators**
  - Signal and background estimated from templates generated at various θ
  - Average of simulations to decrease "theoretical" uncertainty and/or low-dimensional summary statistics
  - Interpolation between finite number of simulations (e.g., with Gaussian processes)

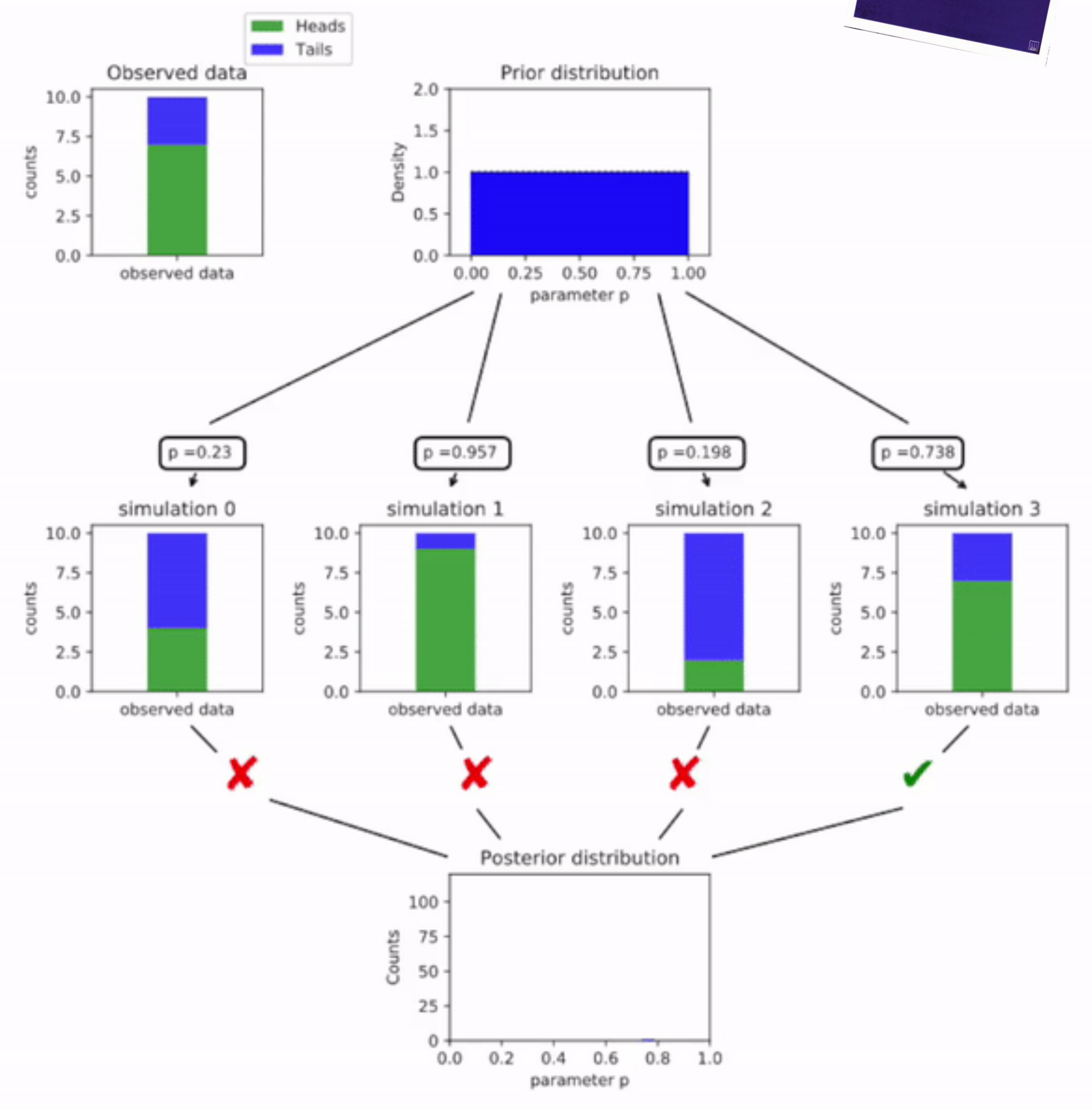
ATLAS Higgs search



Approximate Bayesian Computation

A-B-C, it's easy as 1-2-3 🕺

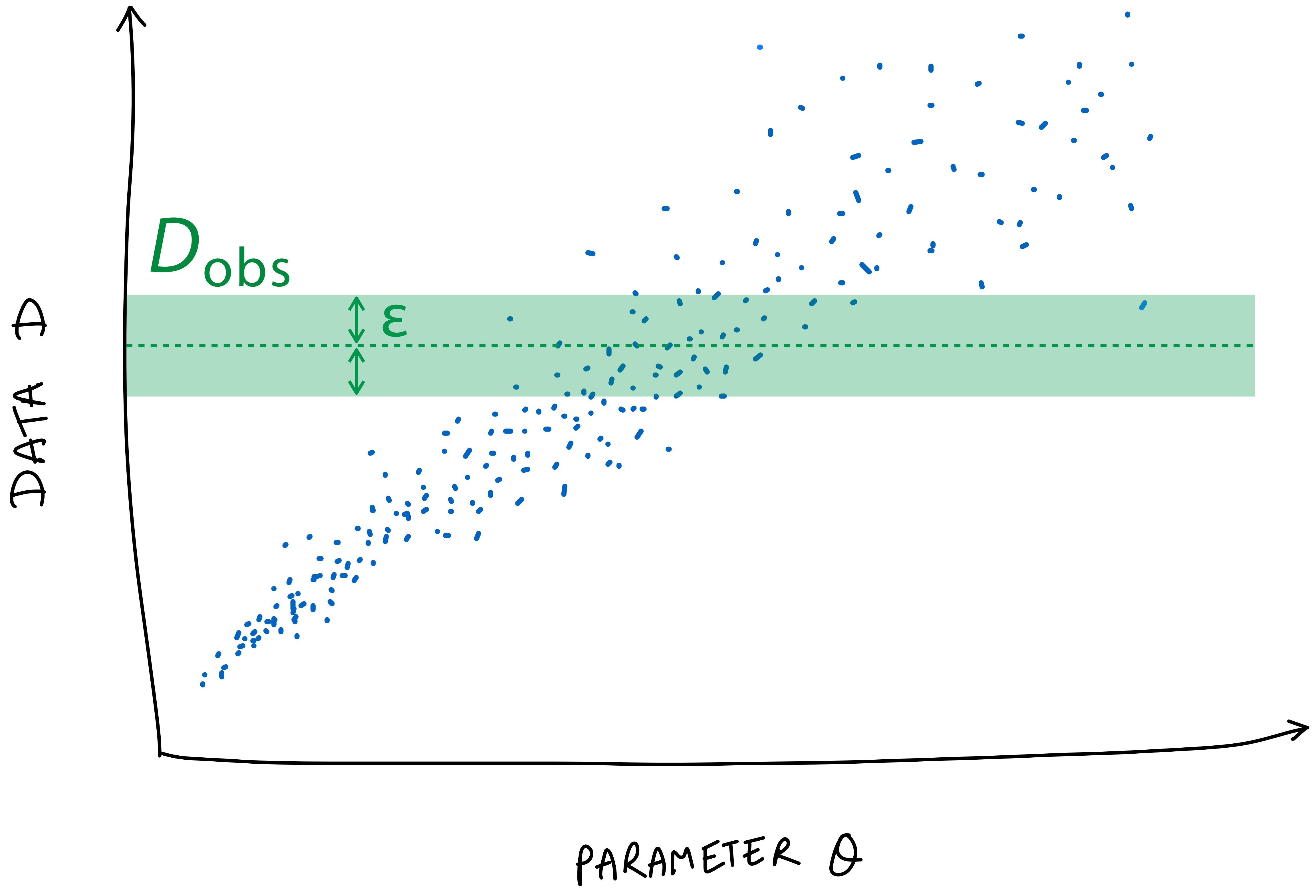
- **Historical approach**
  - SBI is not new: ABC has existed since the 1980s
  - Many successful applications in a wide variety of fields: epidemiology, paleontology, cosmology, you name it!
- **Algorithm based on rejection sampling (no maths)**
  1. Sample parameter space
  2. Generate simulations at these params values
  3. Keep only params where sim ~ data



Approximate Bayesian Computation

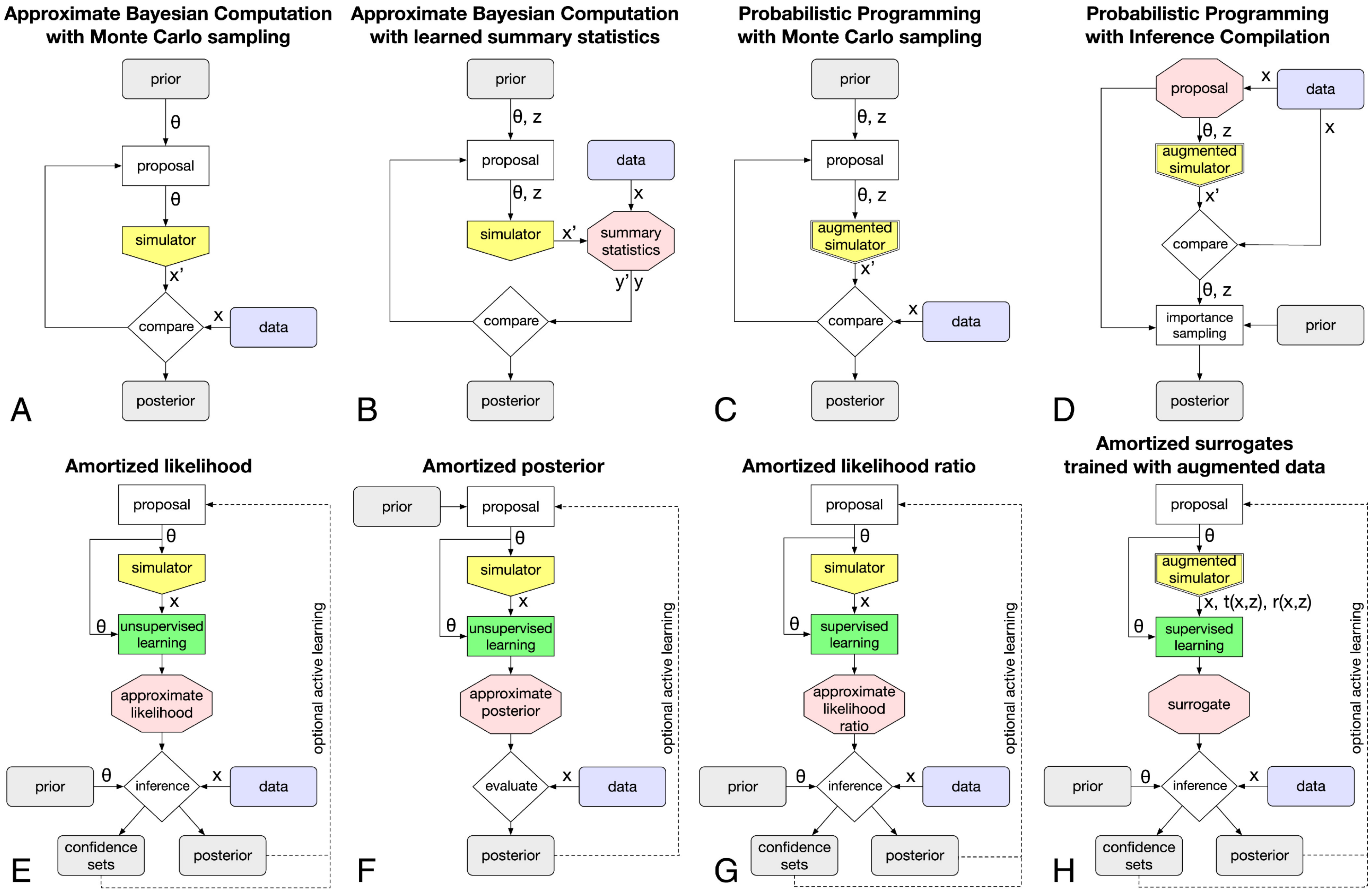
A-B-C, it's easy as 1-2-3 🕺

- **Algorithm based on rejection sampling (with maths)**
  - Given a distance ρ in data space, a threshold $\varepsilon>0$, and observed data d_{obs}
  - For $i=1\ldots N_{step}$
    1. Draw $\theta_i \sim P(\theta)$
    2. Use simulator at θ_i to generate data $d_i \sim P(d|\theta)$
    3. If $\rho(d_i,d_{obs})<\varepsilon$, accept sample θ_i, else reject
  - Result: histogram of accepted samples ~ $P(\theta|d_{obs})$



- **Major caveat: choice of distance ρ and threshold ε**
  - ε too small → reject most sims → inefficient
  - ε too large → distorted posterior → inaccurate

Approaches to SBI



Simulator directly used during inference

Simulator used before inference (*amortized*)

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

Neural SBI

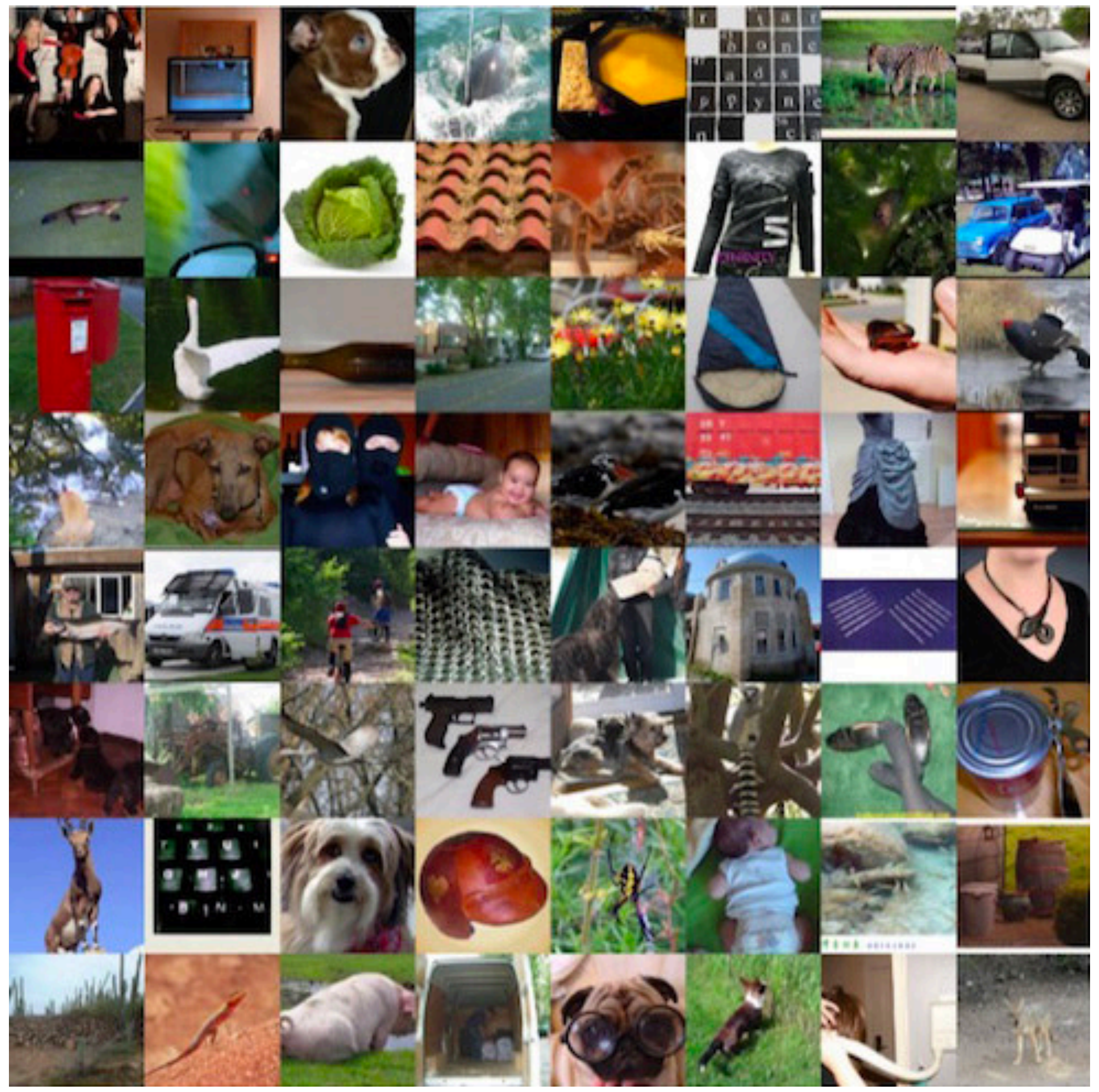
SBI with generative models

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

Generative models

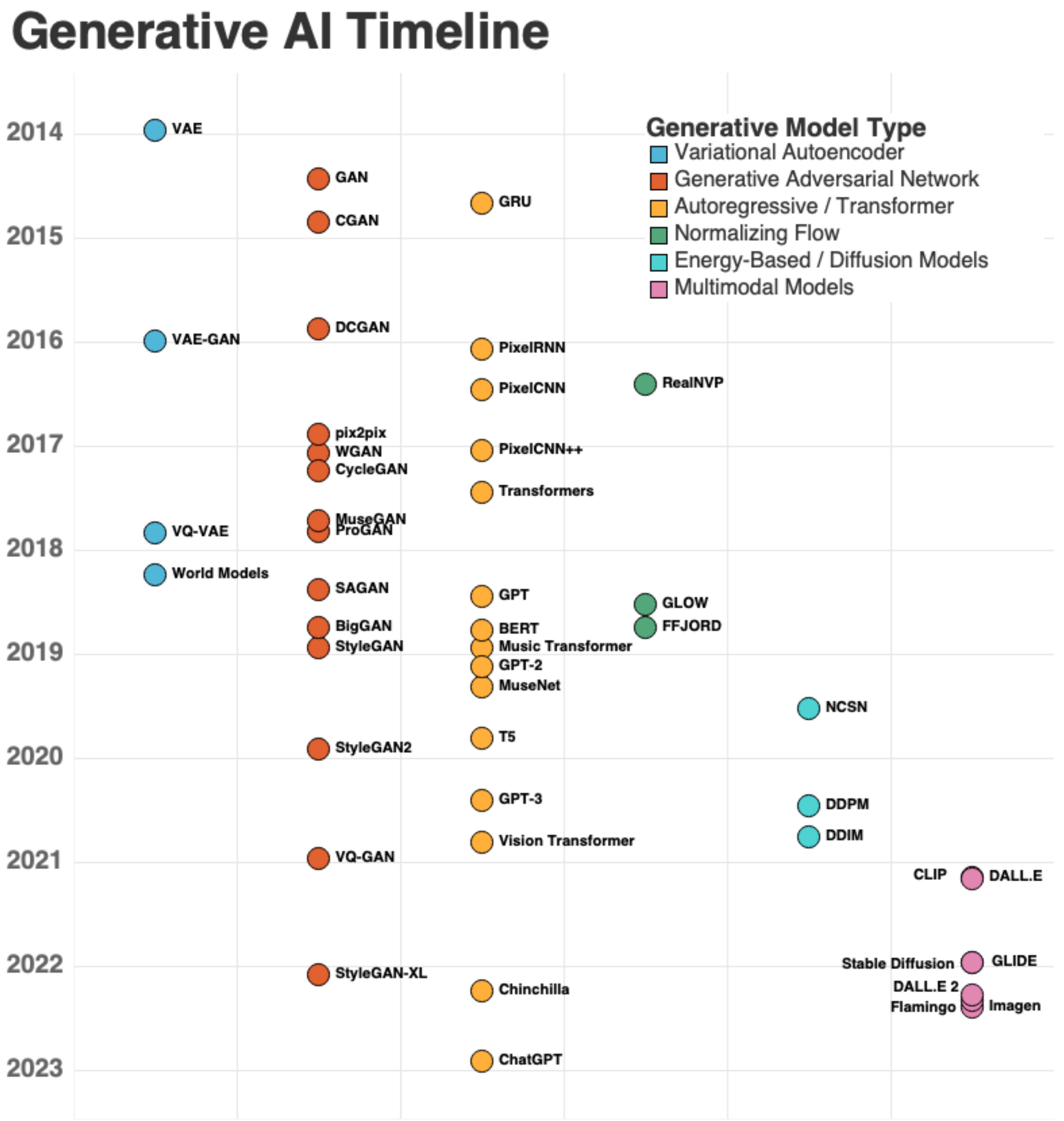
History of models

Real images (ImageNet)



Generated images

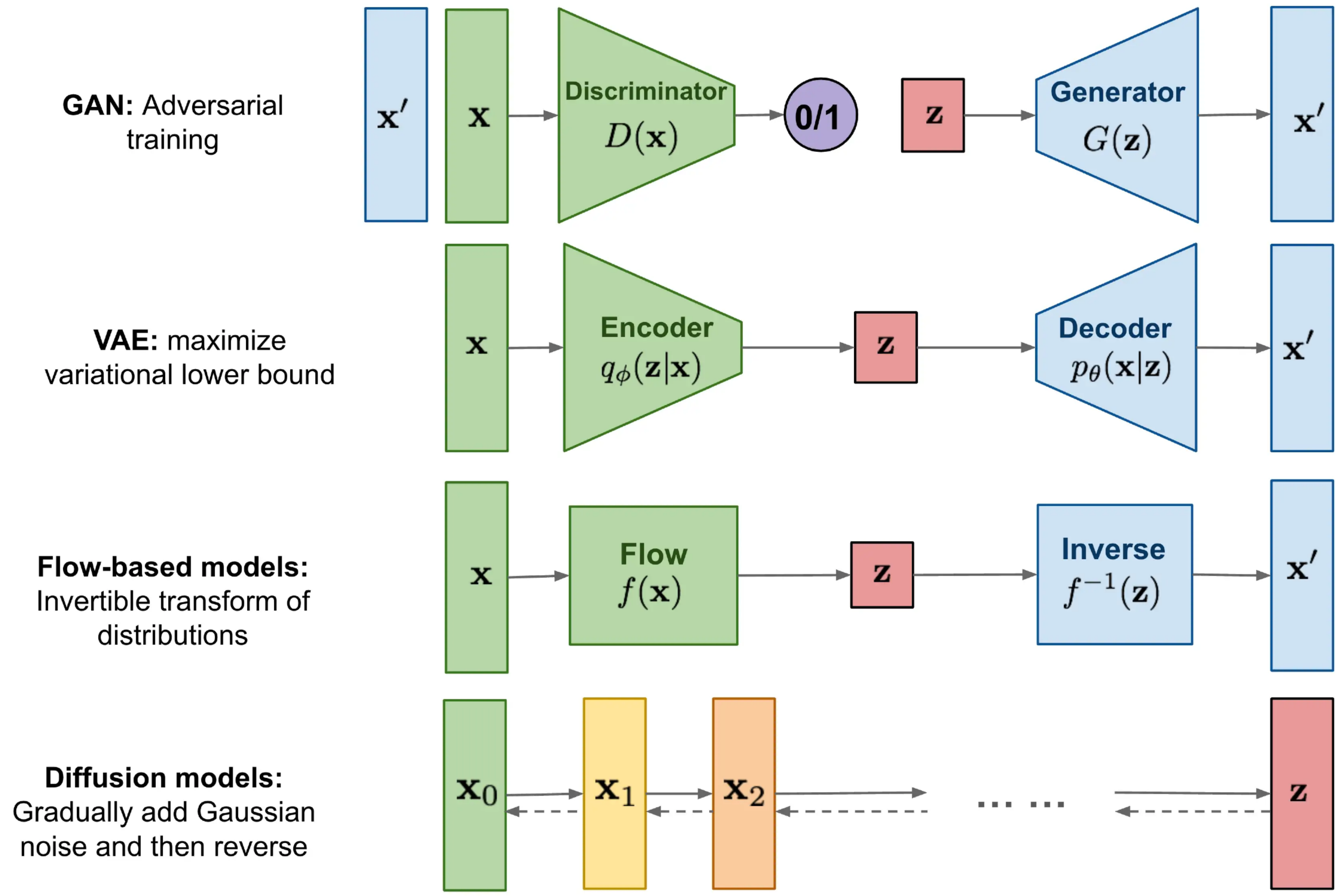
openai.com



David Foster

Generative models

- **Latent variable models**
  - Data mapped to (meaningless) latent variables with known distributions
  - With/without compression (e.g., VAE)
- **Functionalities**
  - Sampling, ie generate new data
  - Flows allow probability density evaluation 🤟
- **Pros and cons**
  - Sampling/evaluation speed
  - Sample quality
  - Mode coverage



Lilian Weng

Normalizing flows

Principle

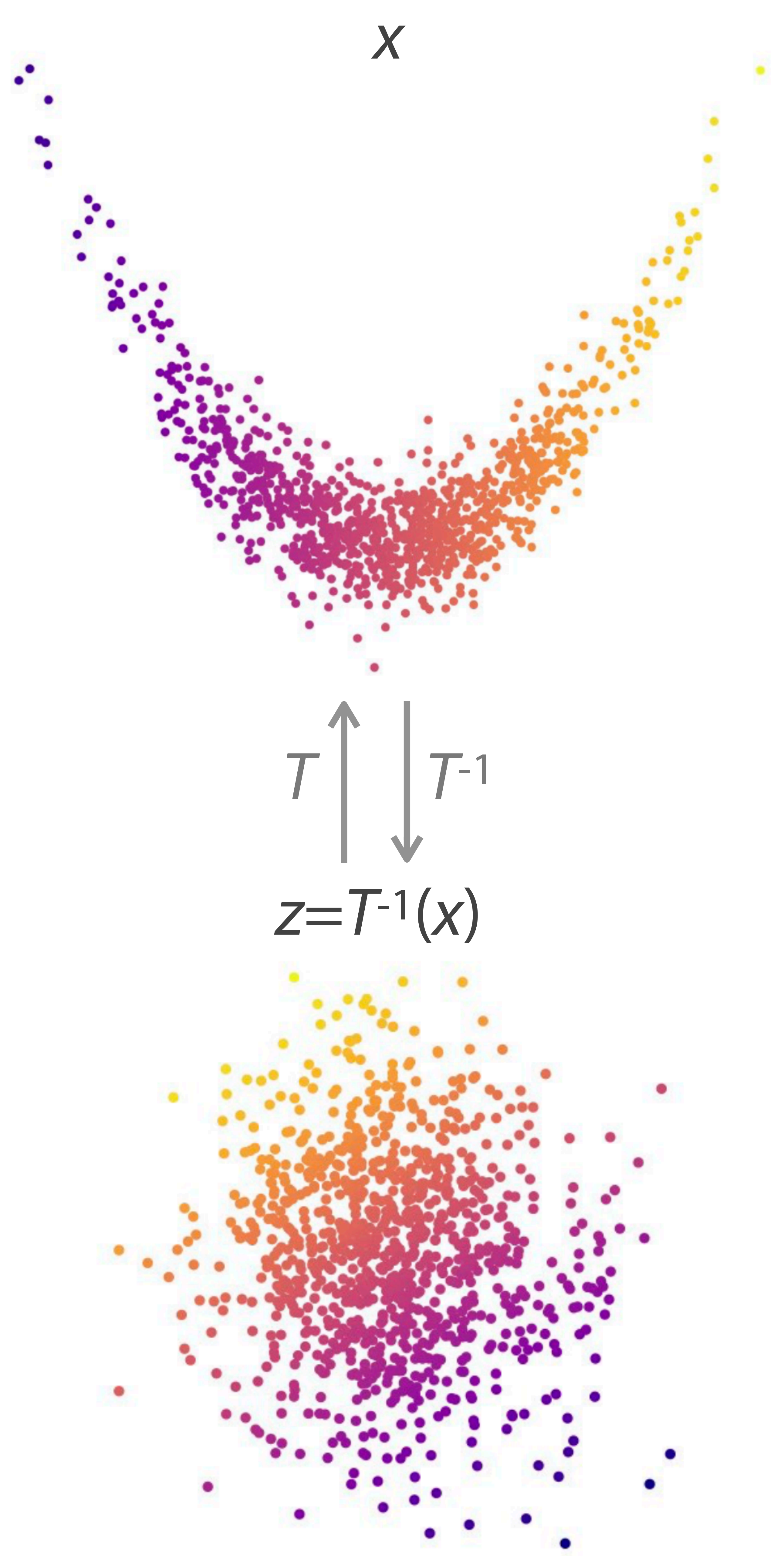
1. Start with data $x \sim p_x(x)$, whose distribution p_x we are trying to learn

2. We map it to some variable $z = T^{-1}(x)$ with an invertible mapping T parametrized by neural networks

3. Imagine we can transform x such that $z \sim p_z(z)$ is unit Gaussian, then

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

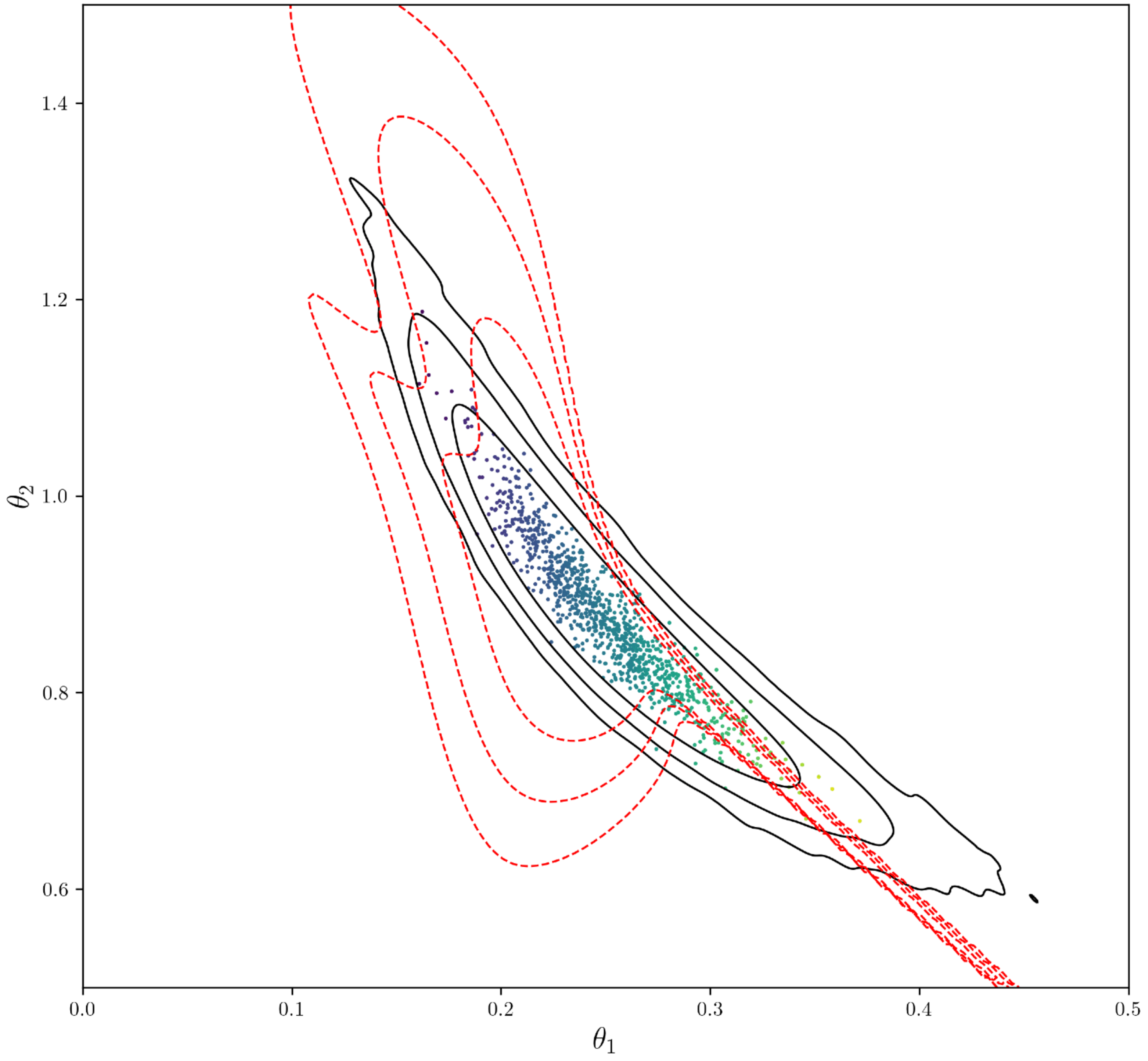
- Fast evaluation of p_x
- Sampling of p_x : sample z from unit Gaussian and transform to $x = T(z)$



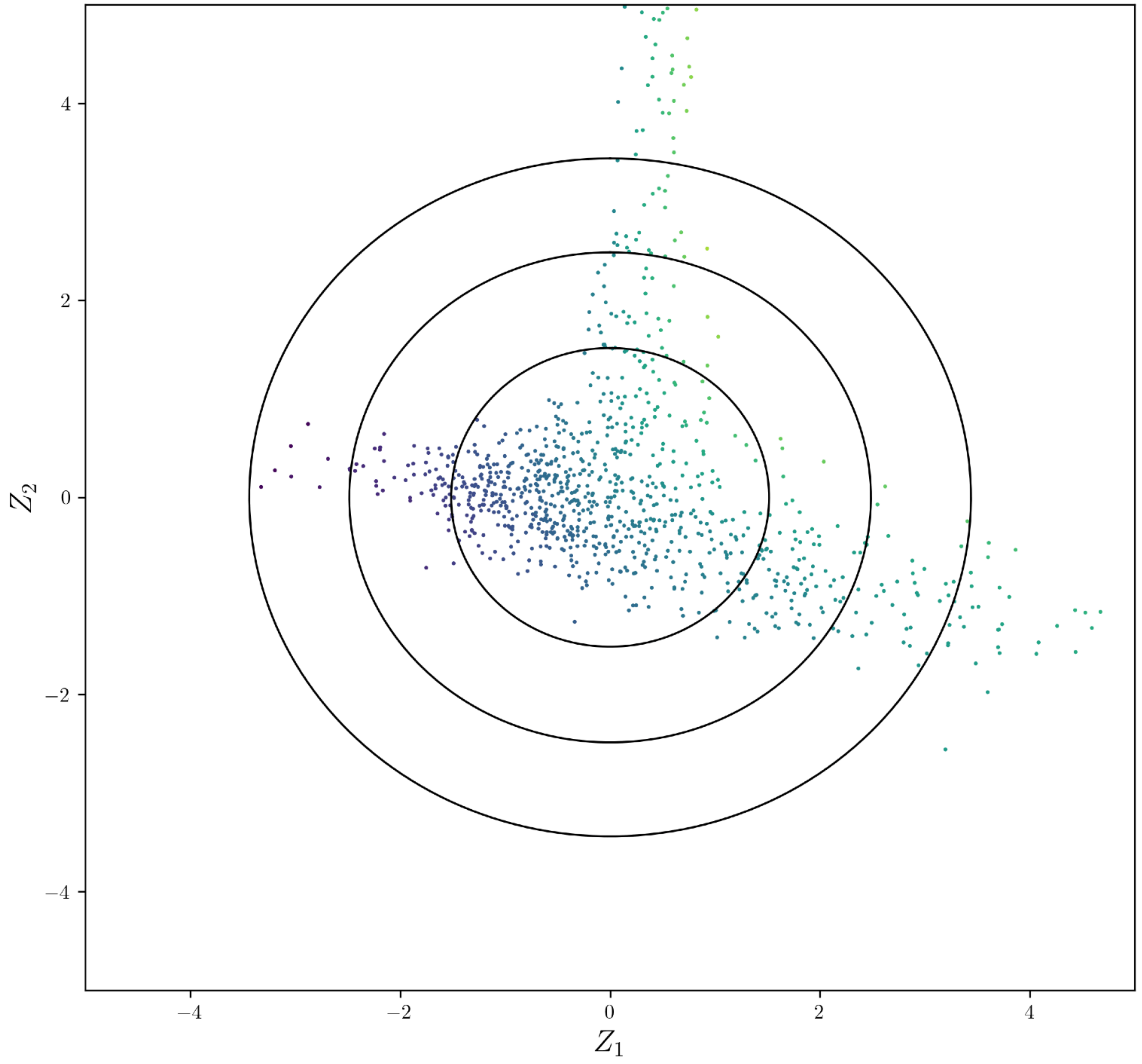
Normalizing flows

What happens during training? 🤩

Parameter space p_x



Latent space p_z

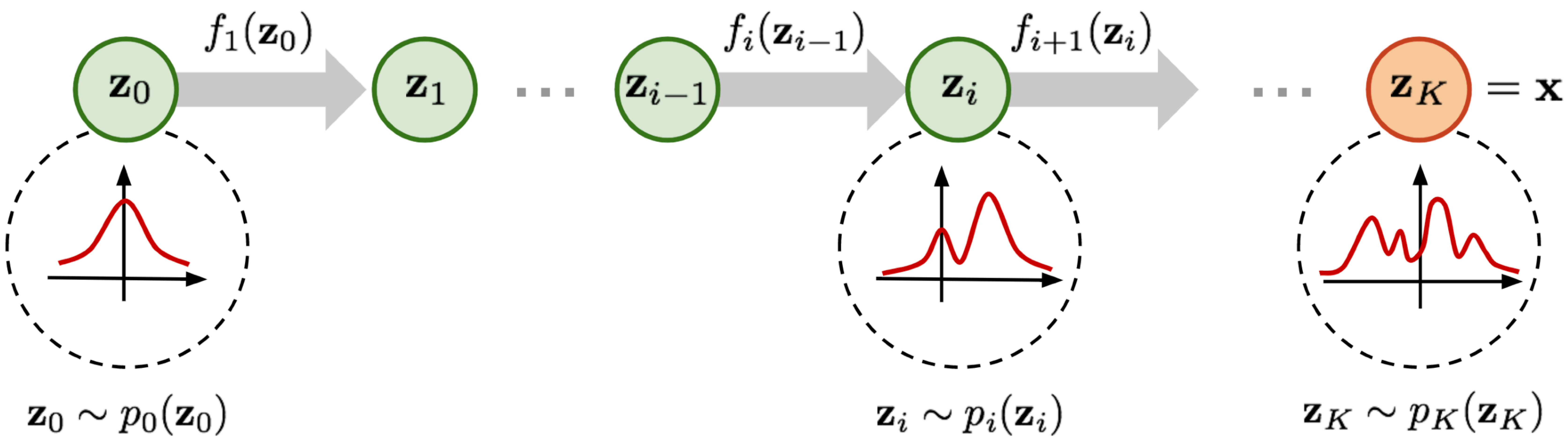


Raveri & Doux 22

Normalizing flows

In practice

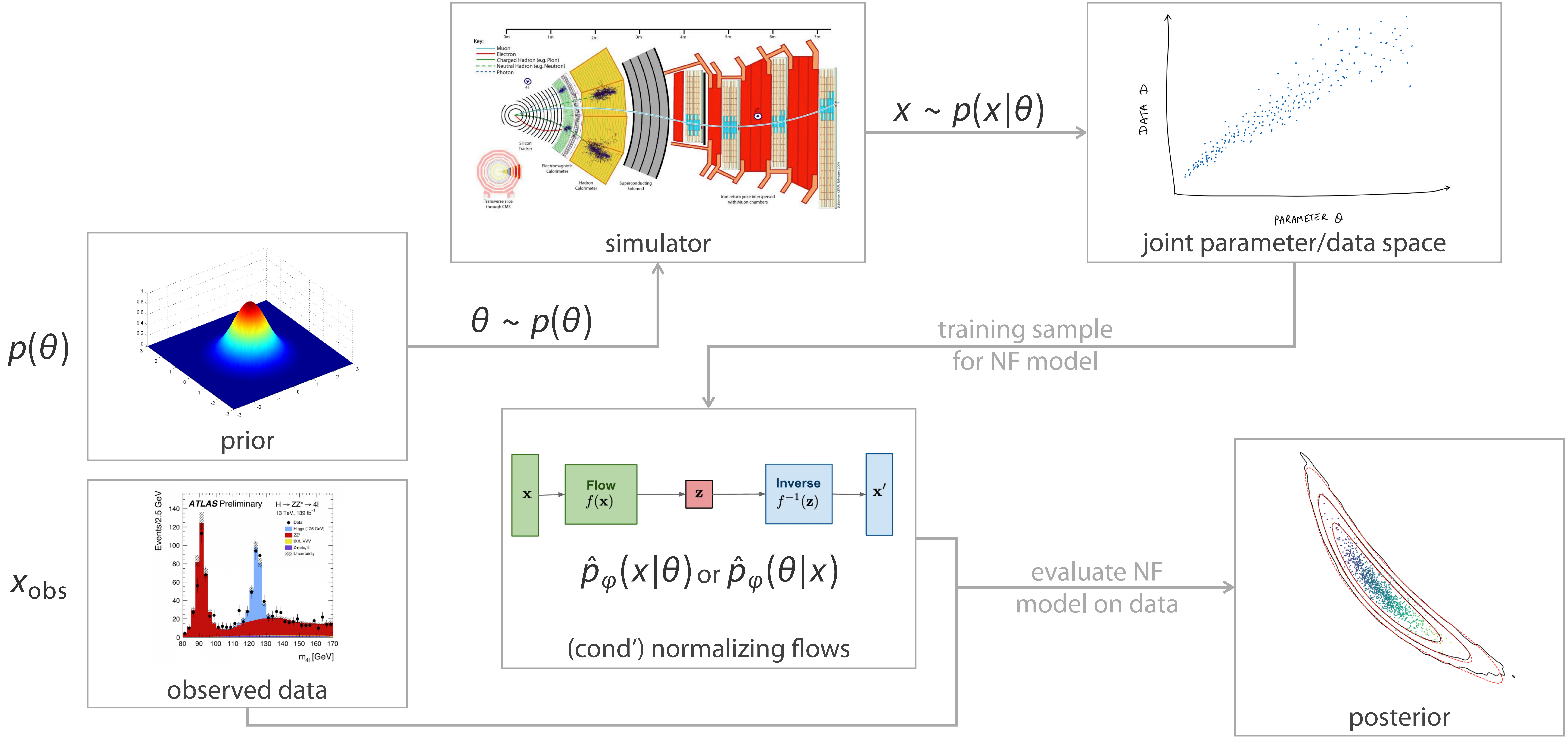
- Stack a bunch of “simple” transformations to learn more complex distributions



Lilian Wang

  - Popular NF flavors: Masked Autoregressive Flows, FFJORD, Glow (1x1conv), Neural Splines
  - NFs vary by parametrization of invertible mappings: ± flexible, ± fast
- Conditional NFs: mapping $x=T(z|\theta)$ depends on conditioning variable θ (just another NN input)

SBI with cNFs



Adapted from Michael Kagan

SBI with cNFs

Learn the posterior or the likelihood

	Neural posterior estimation (NPE)	Neural likelihood estimation (NLE)
Conditioning	condition on data to learn posterior	condition on data to learn likelihood
	$\hat{p}_\varphi(\theta\|x)$	$\hat{p}_\varphi(x\|\theta)$
Pros and cons	‣ fast evaluation on (new) observed data ‣ depends on prior	‣ involves sampling to obtain posterior ‣ no dependence on prior

- **Sequential version: SNPE/SNLE**
  - Alternate sampling/training to target simulations in region of interest

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");
```

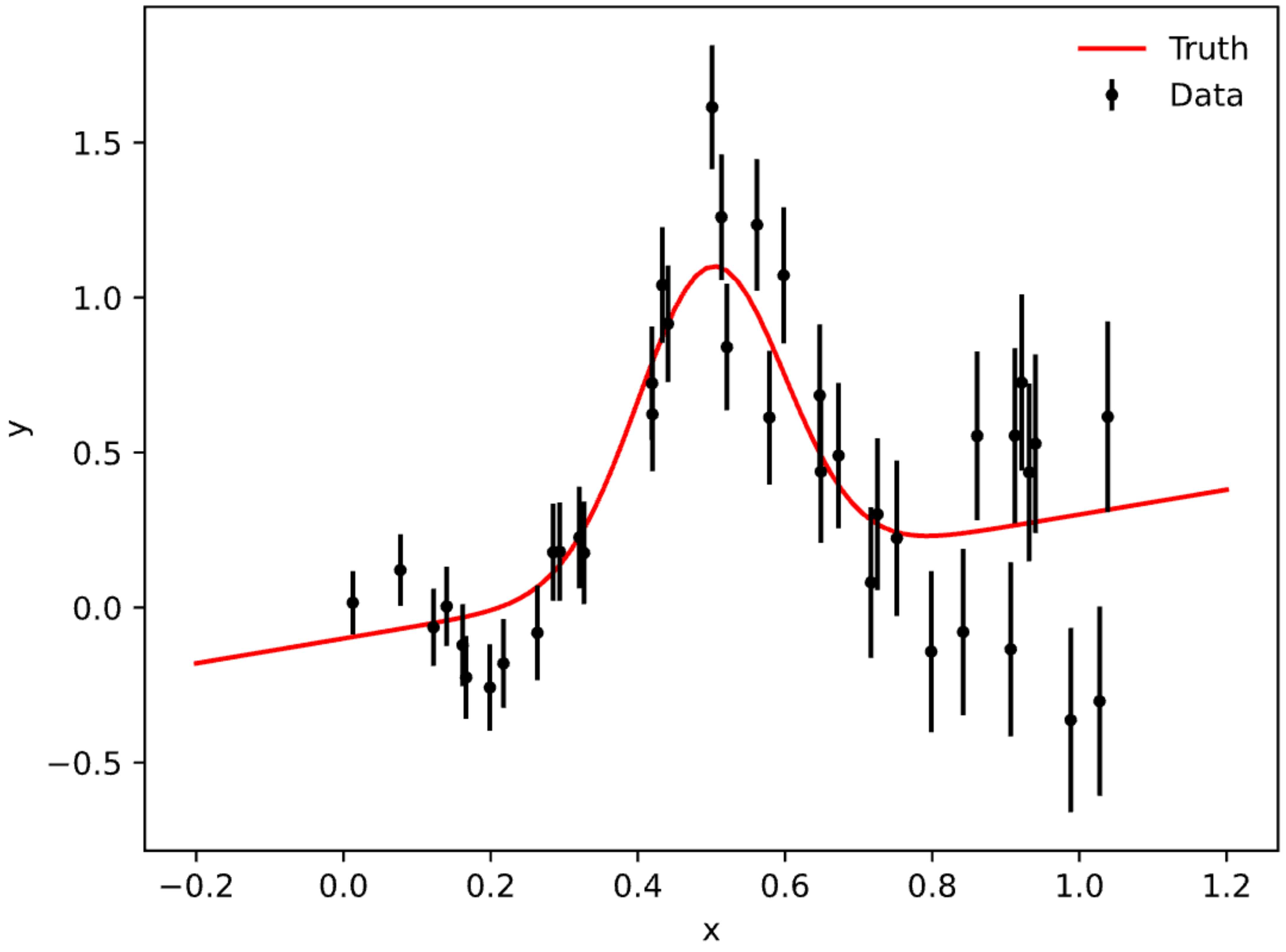
Setup

Google Colab notebook

Model and goal

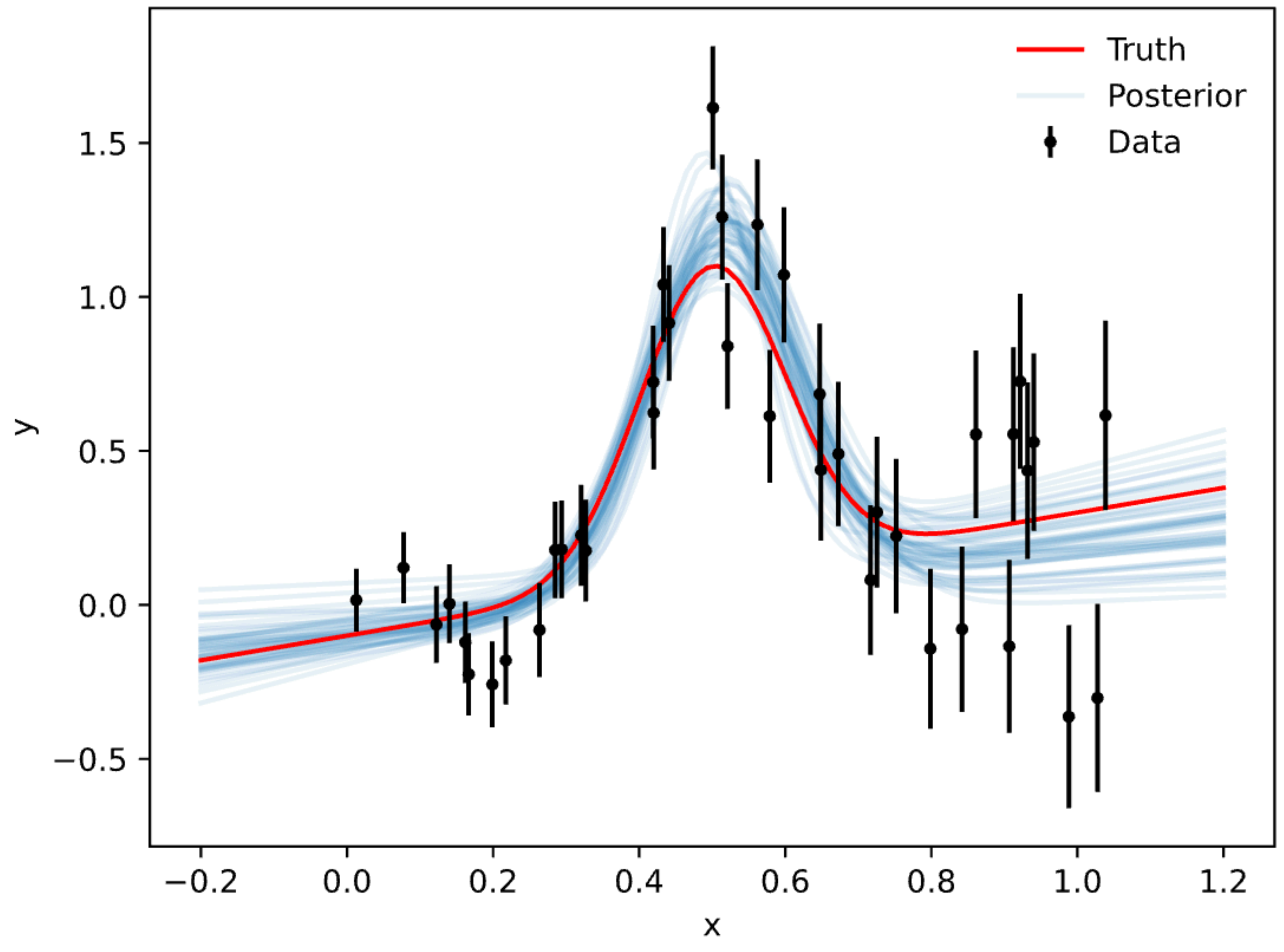
- **Packages**
  - sbi: SBI package implementing SNPE, SNLE, NLRE among others
  - PyTorch: facebook's machine-learning library
- **Simulator**
  - Model = background (affine) + signal bump (gaussian)
  - Gaussian noise growing with x (why not?)
- **Goals**

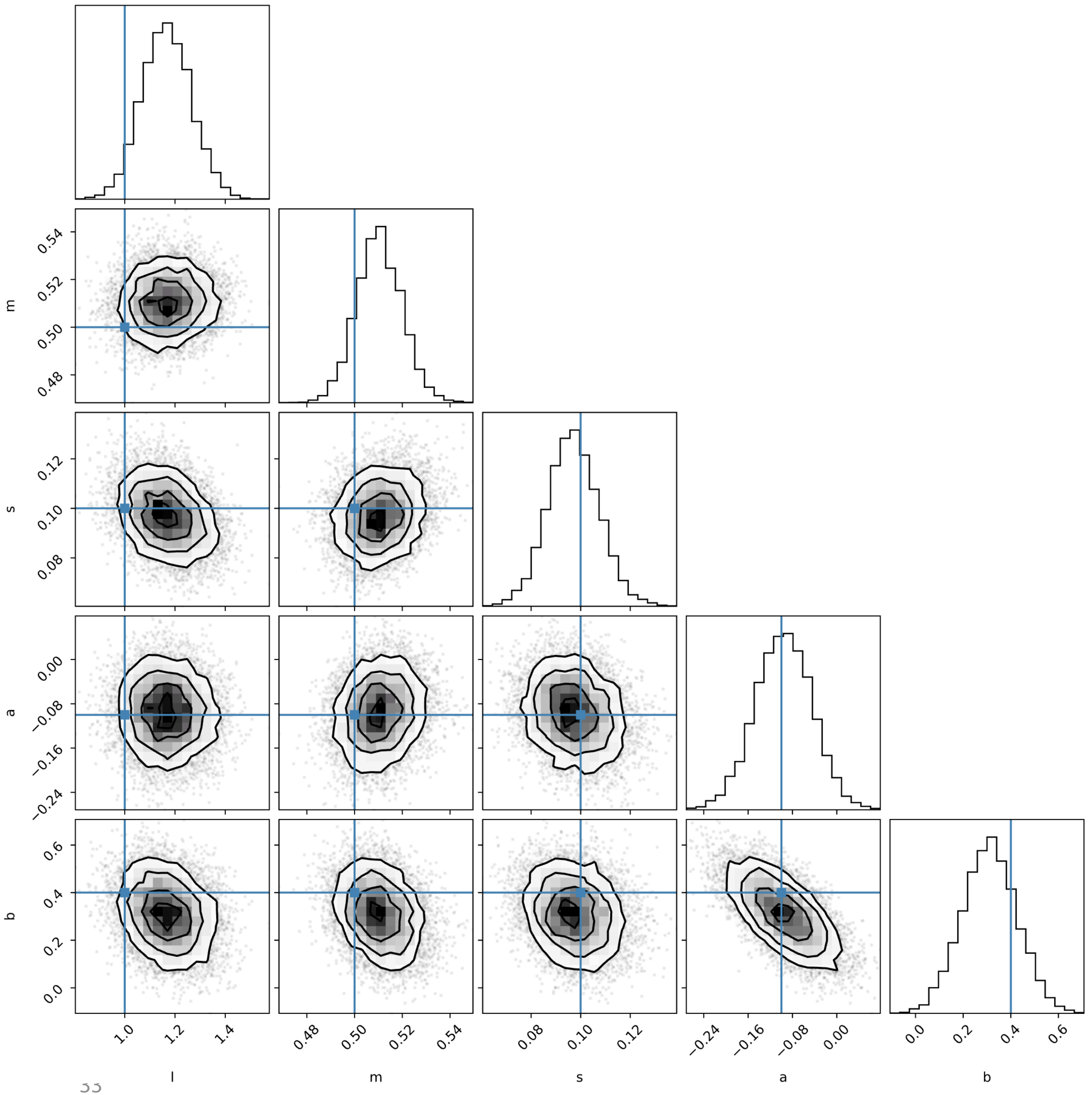
1. Run simple SBI
2. Illustrate differences between likelihood-based and SBI



Results

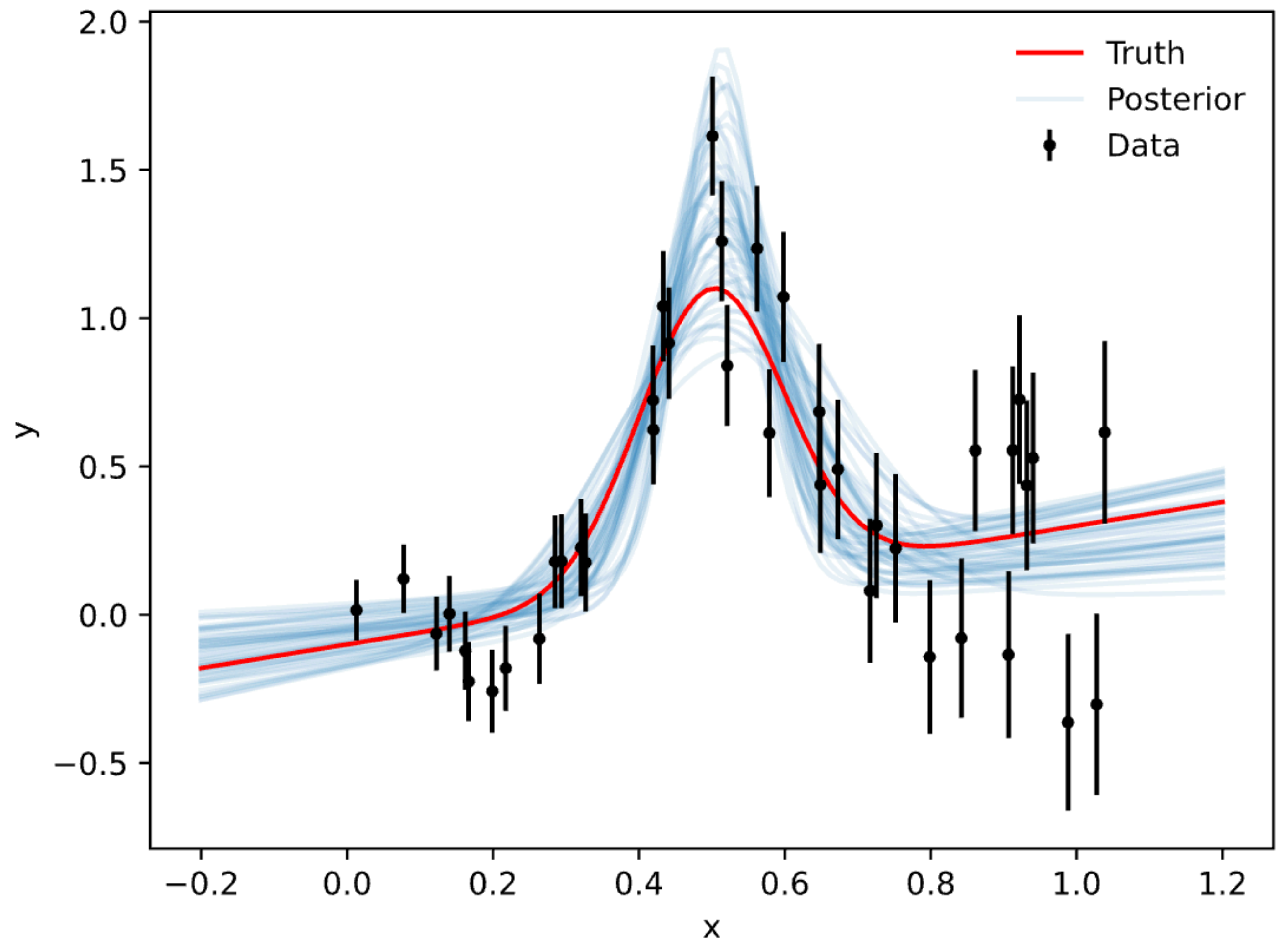
Likelihood-based inference

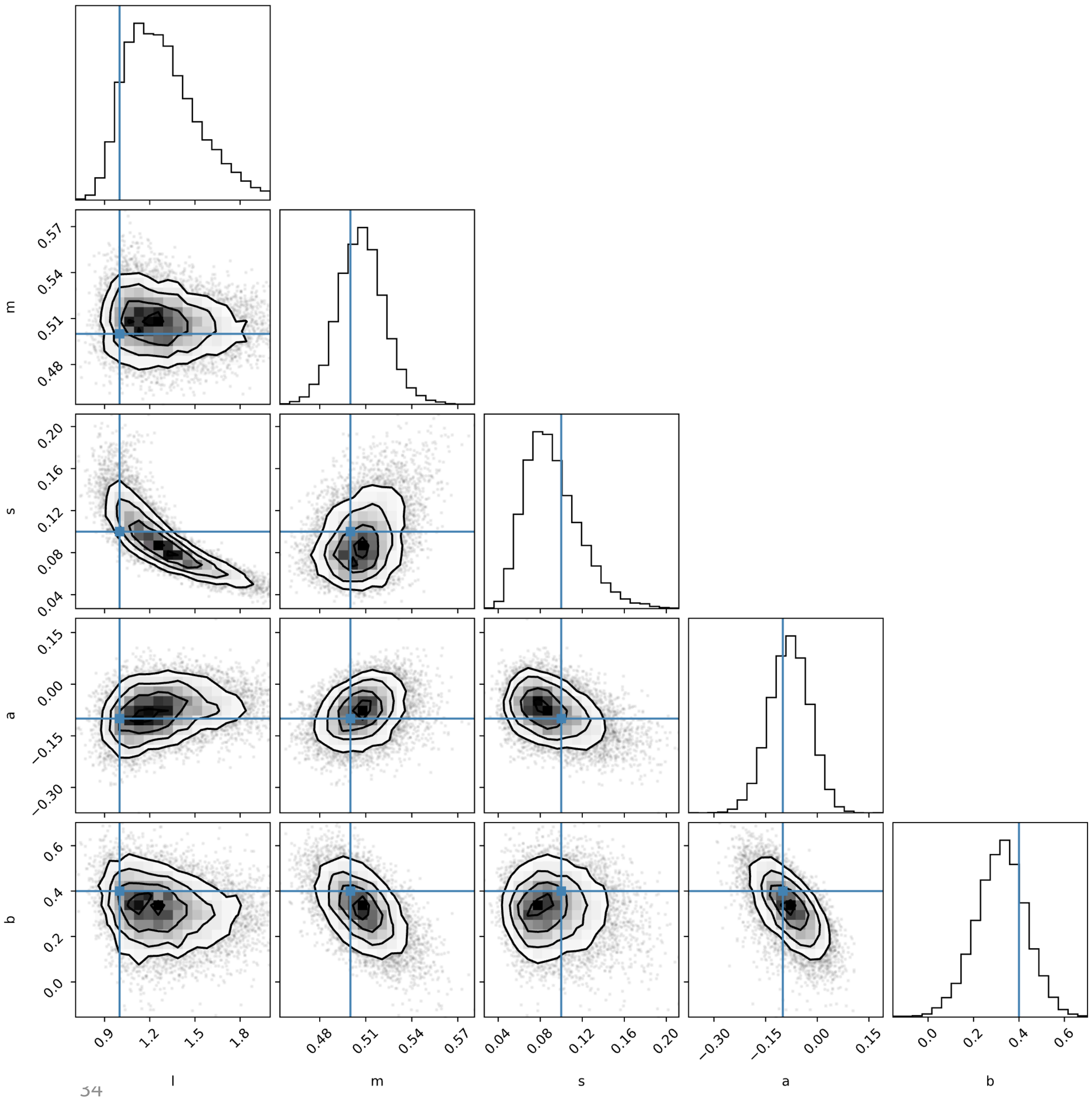




Results

Simulation-based inference





Results

Likelihood-based vs SBI

- What did you find out?

- What did we hide when using the sbi package?

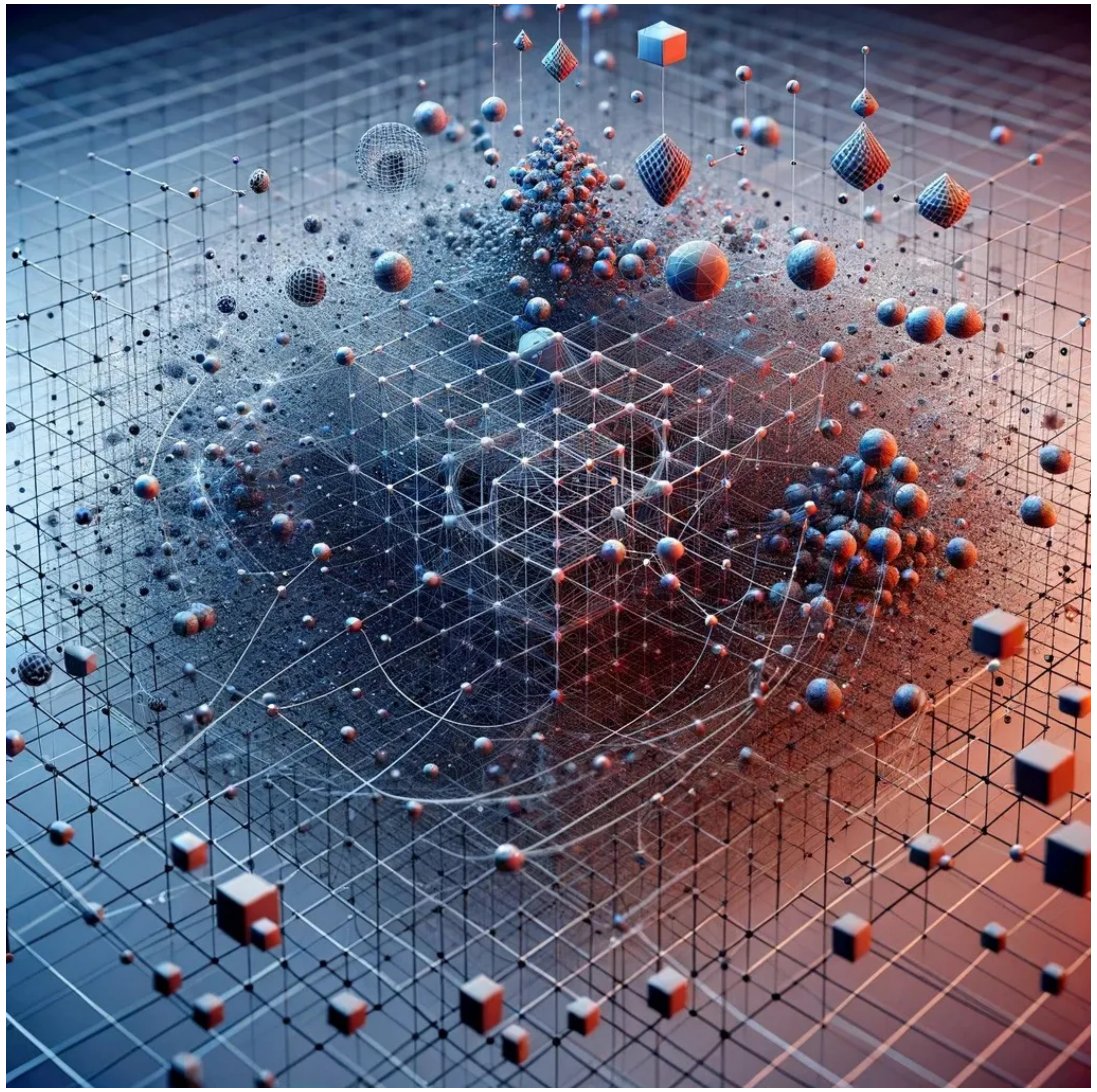
Beyond vanilla SBI

Improvements and research directions

Copilot prompt: cosmology simulations in the style of Basquiat paintings

Challenges of SBI

Spoiler alert: SBI is not magic



- Curse of dimensionality 😵‍💫
  - Larger dimension = harder problem
  - Both data and params dimensions matter
- **Running many simulations** 🔥
  - Limited computing resources = finite number of sims*
  - Number of sims needed depends on dimensions, model complexity, params priors
- **Ideas to circumvent these issues?** 😎
  1. Reduce dimension of data
  2. Target simulations
  3. Better/faster simulator

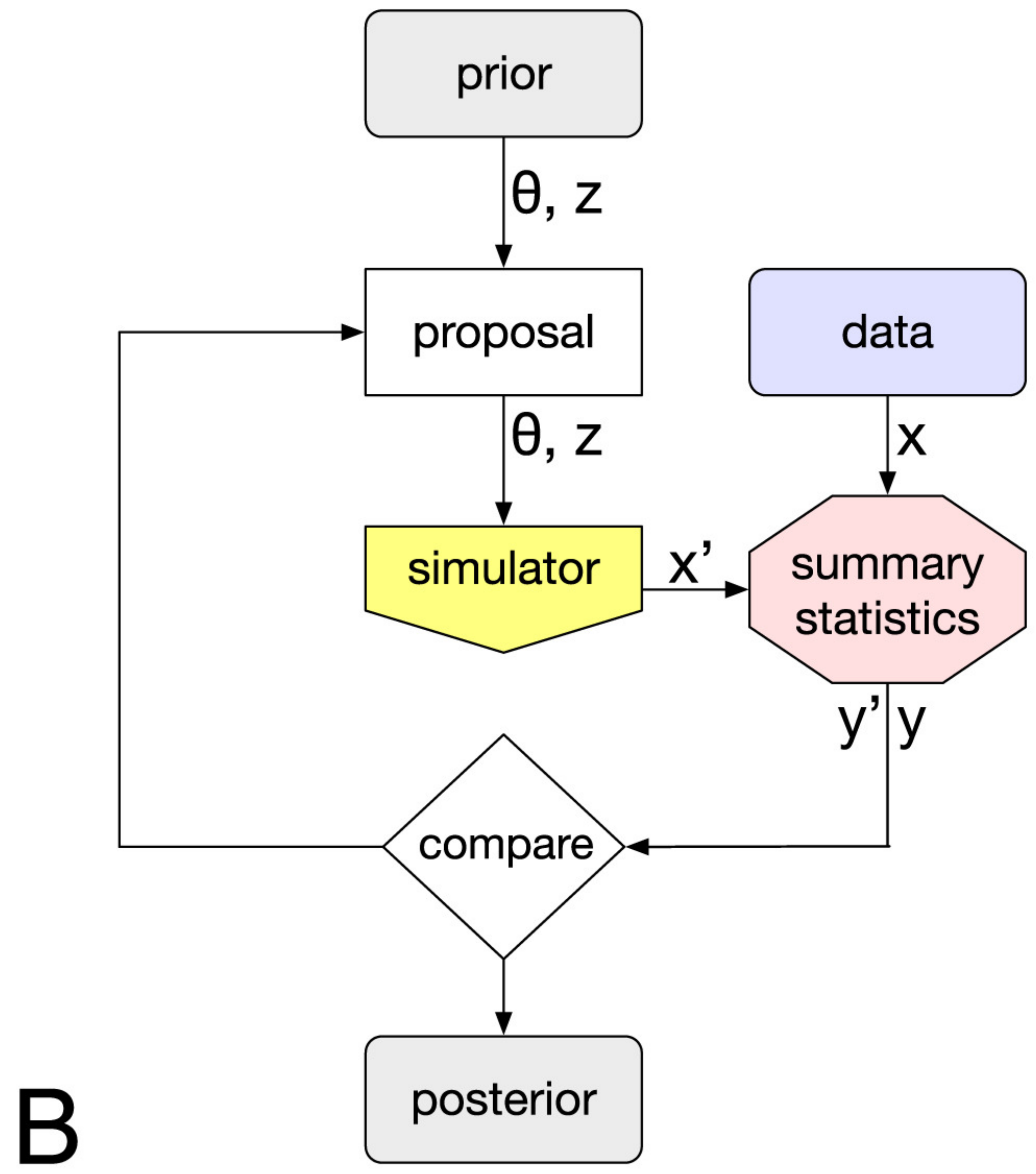
*let's assume fitting NFs is negligible

(learned) summary statistics

Compressing data

- **Summarize data with minimal information loss**
  - Physicists are *experts* at it! 😎
  - Mathematically, 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, the likelihood itself (!), etc.
- **Learned summary statistics**
  - PCA-like compression, e.g., MOPED (Heavens+15)
  - Neural network compression, e.g., information maximizing NN (Charnock+18), auto-encoders, etc.

Approximate Bayesian Computation with learned summary statistics

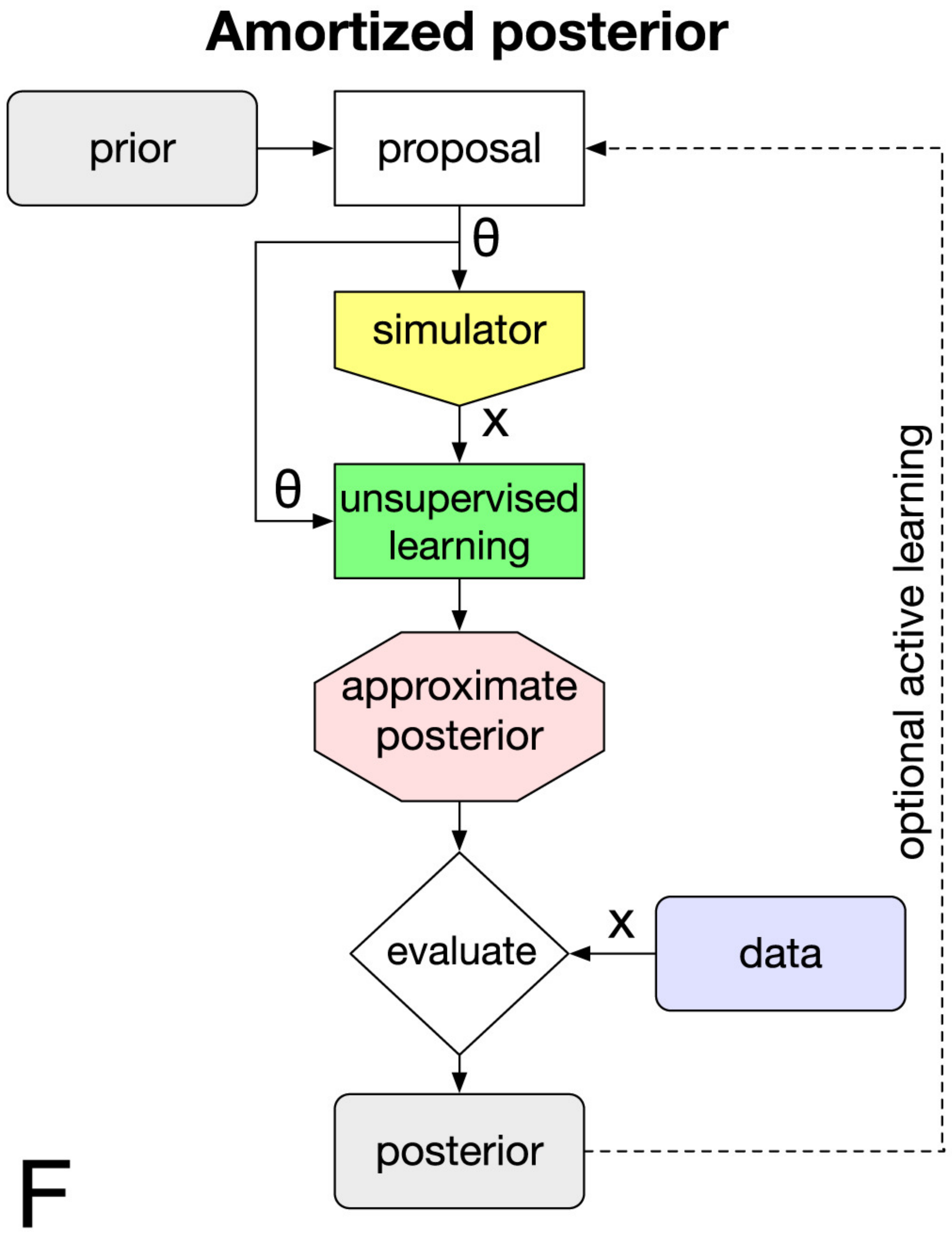


Cranmer+20

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*
- **In practice**
  - Many strategies: uncertainty sampling, adversarial sampling, discriminative sampling
  - Depends on your sims…

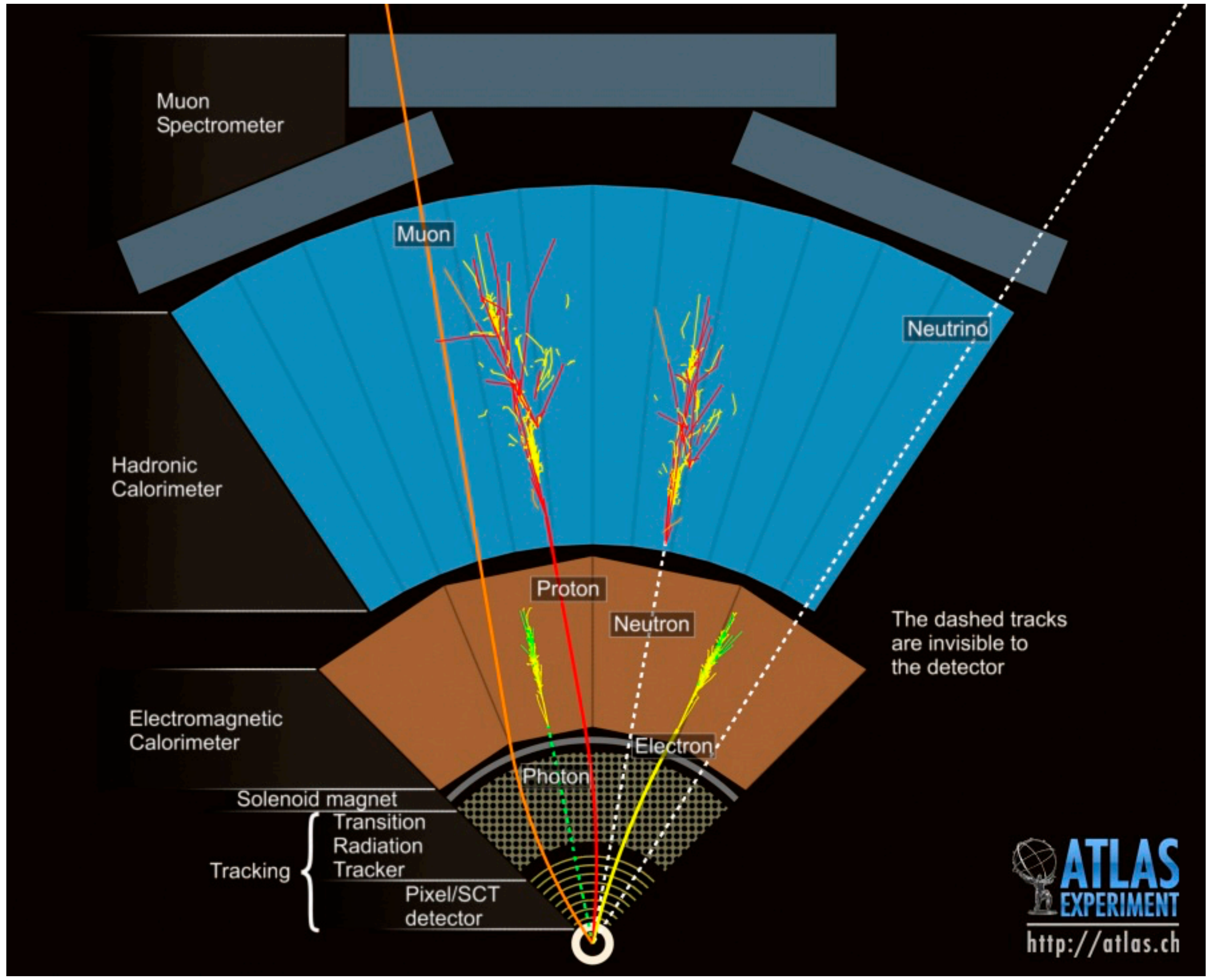


Cranmer+20

Fiddling with the simulator

Faster, better, stronger

Monte-Carlo simulation of an event in the ATLAS detector



- **Probabilistic programming**
  - Alter simulator sampling distributions (θ, 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…

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 $(\theta_0, x) \sim \delta(\theta_0)\mathcal{L}(x|\theta_0)$

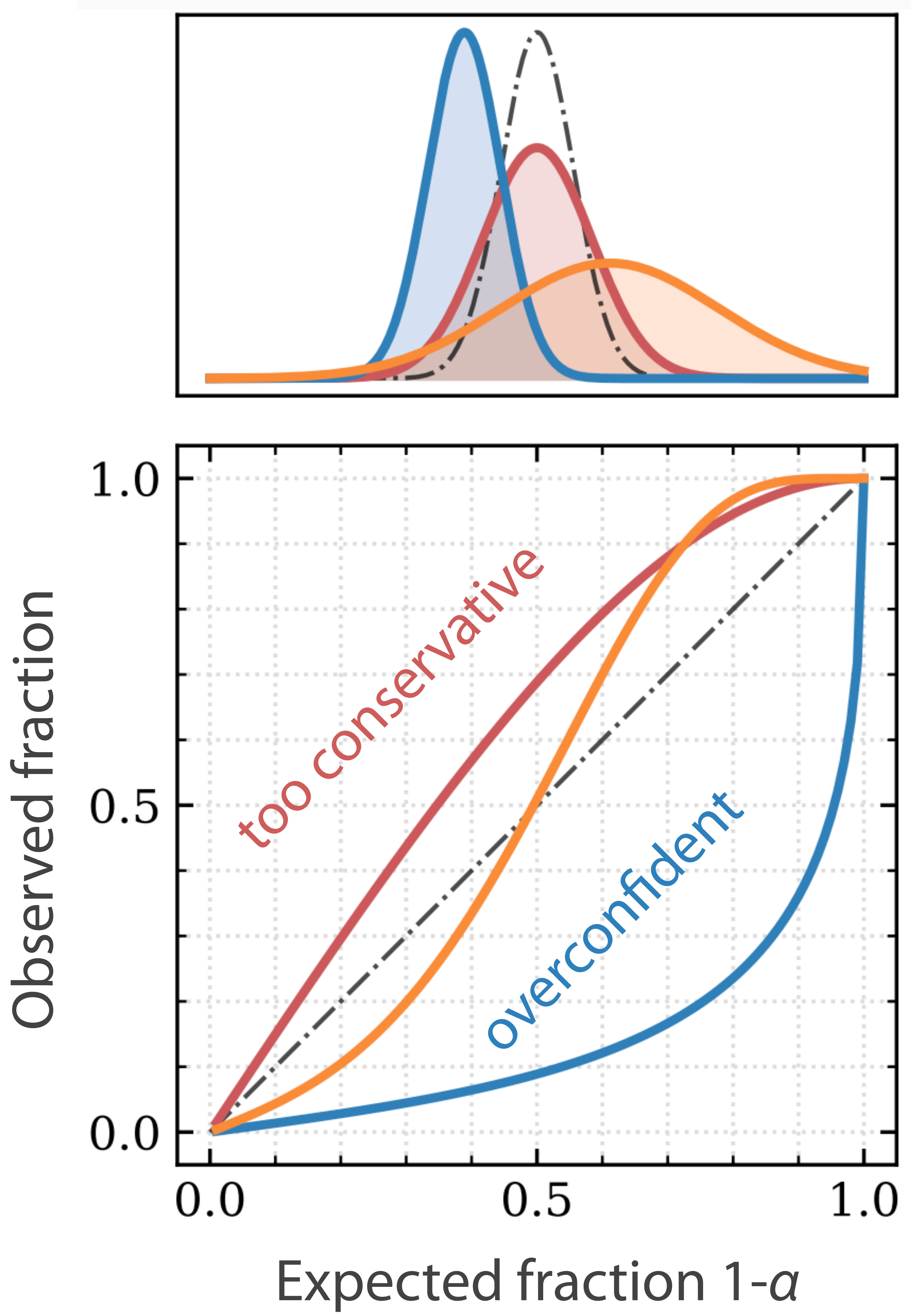
then $\mathrm{NN}(x, \theta) \xrightarrow{\text{training}} \frac{1}{1 + \frac{1}{r(x,\theta)}}$ ✌️

- Allows frequentists and Bayesian approaches, without actually knowing the likelihood!
  - For MCMC chains, use $p(\theta|x) \approx r(x, \theta)\pi(\theta)$

Coverage diagnostic

How accurate are posterior distributions?

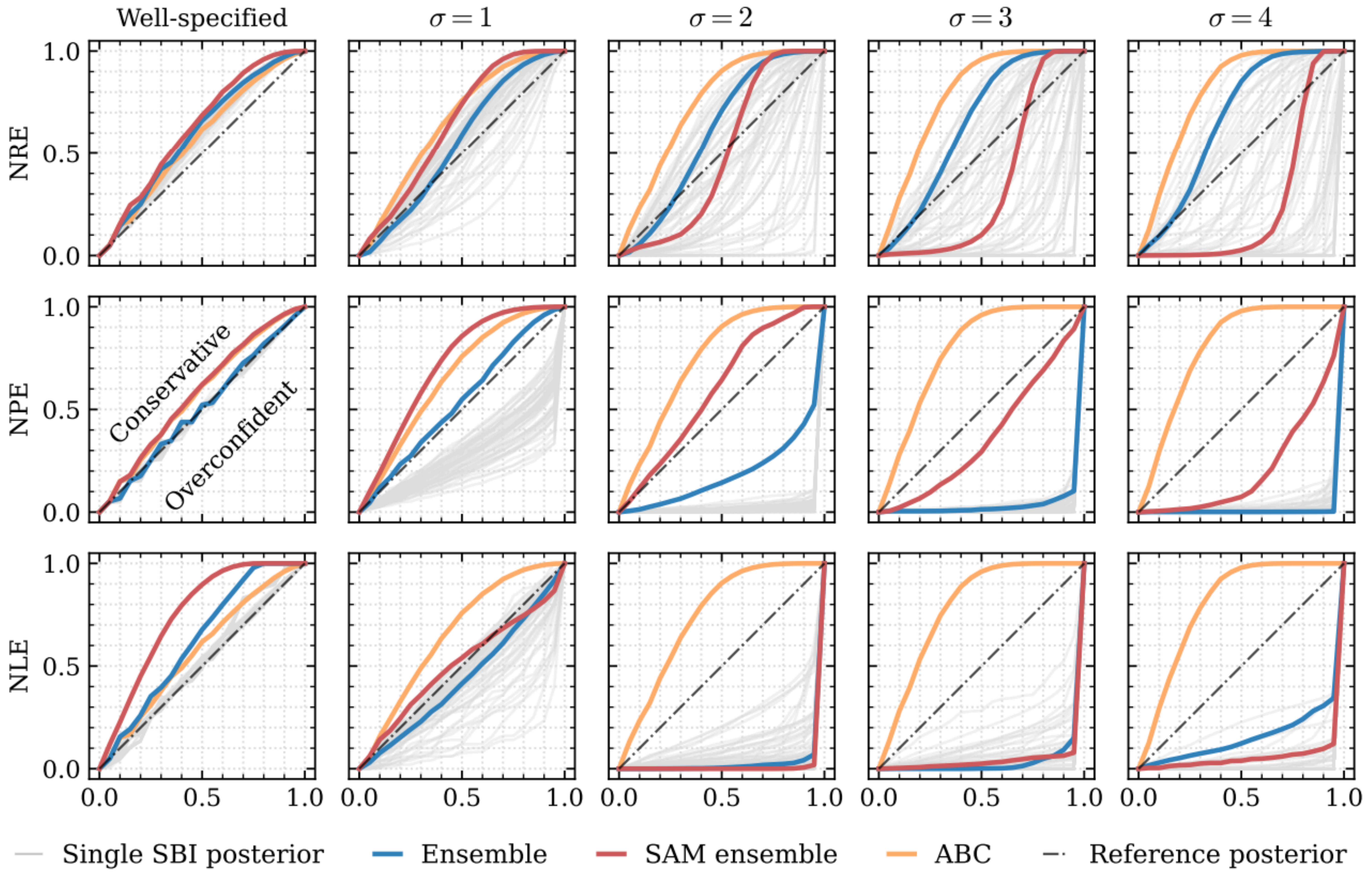
- **Empirical coverage test**
  1. Given a test set $(\theta_i, x_i) \sim \pi(\theta)\mathcal{L}(x|\theta)$, confidence level α and neural posterior estimator $q(\theta|x)$
  2. Compute fraction of (θ_i, x_i) that lie in the 1-α highest confidence region of q
  3. Should be 1-α !
- **What to change if test fails?**
  - Number of simulations
  - Sampling of sims in params space
  - Model architecture, flexibility (over/under-fitting or biases)



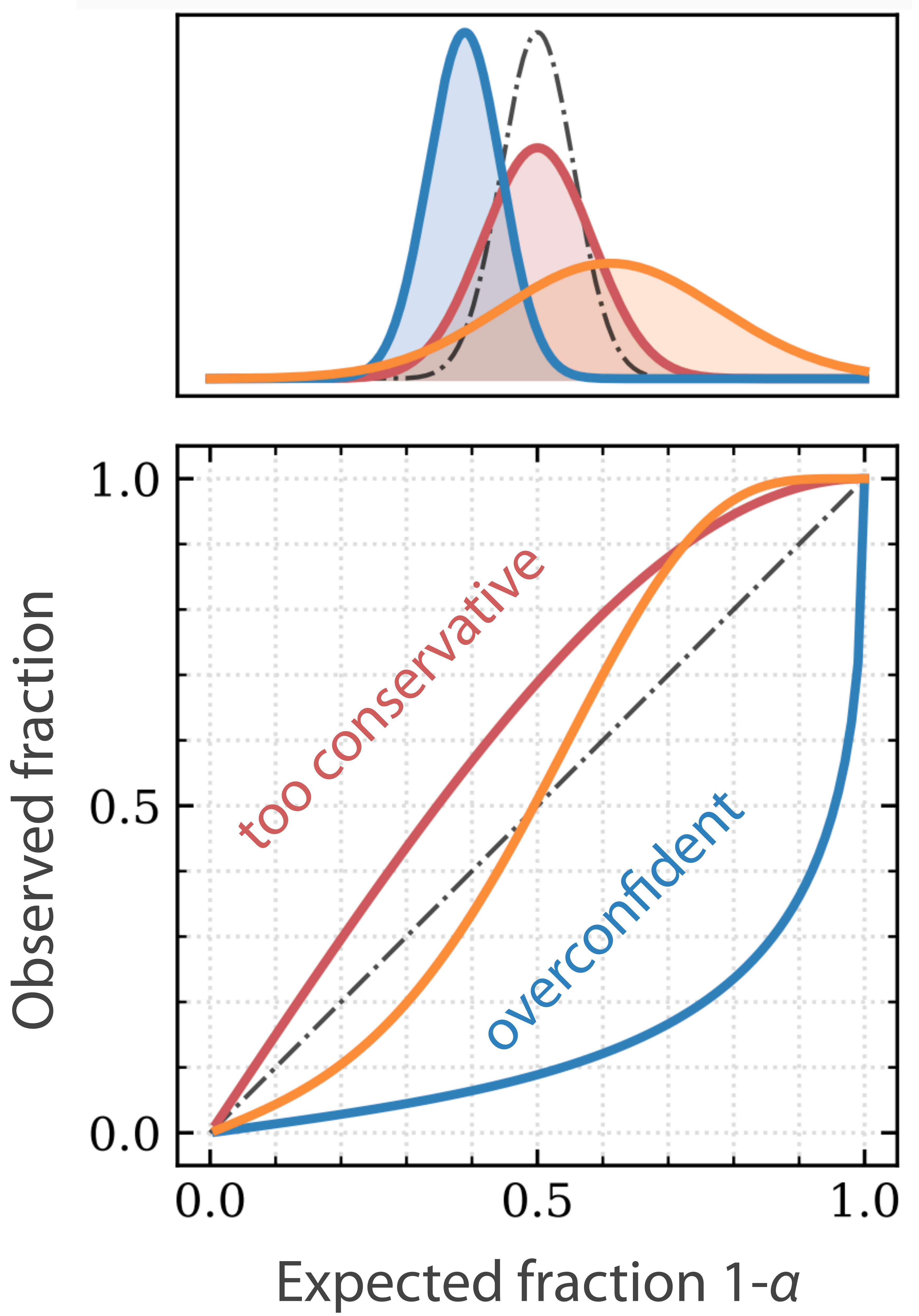
Cannon+22

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

  - Links in the slides…