Physics 361 - Machine Learning in Physics

Lecture 12 – Simulation-Based Inference

February 27th 2025

AI
∩
Universe
$\mathcal{L} = -\frac{1}{4} F_{\mu\nu} F^{\mu\nu}$
$+ i\bar{\psi}\not{D}\psi + h.c.$
$+ \bar{\psi}_i y_{ij} \psi_j \phi + h.c.$
$+ |D_\mu \phi|^2 - V(\phi)$

Moritz Münchmeyer

Simulation-Based Inference

Overview

References

- This presentation is based on the review “The frontier of simulation-based inference” (Cranmer et al 2019)
  - https://arxiv.org/abs/1911.01429

- Software packages for SBI include
  - https://sbi-dev.github.io/sbi/ simulation-based inference in python
  - https://docs.pyro.ai/en/stable/index.html Pyro - probabilistic machine learning with tensor flow

Inference with Simulations

- **Complex phenomena in physics and other sciences can often be described with simulations** (while analytic treatment is often impossible)
- Examples:
  - Formation of the universe
  - Particle shower induced by a cosmic ray in the atmosphere
  - Particle tracks generated by colliding protons at the LHC
  - Climate, Weather, Epidemics etc.
- We want to **infer physical parameters**, that go into the simulation (such as the age of the universe or the mass of the Higgs) by **“comparing” the outcome of simulations with measured data**.
- This is usually not straight forward to do. Problems include:
  - Simulations usually don’t provide explicit likelihoods.
  - We need a way to get statistical and systematic error bars.
  - It may be difficult to run enough simulations.
  - Simulations may have uncertainties themselves.

Implicit vs Explicit inference

- In most situations a simulation does not provide a probability density (likelihood) $\mathscr{L}(x \mid \theta)$ of observations given parameters. Such simulations are sometimes called **implicit models**.
- Implicit means that their **likelihood cannot be computed explicitly**, i.e. it is not computationally tractable. We only get samples of the simulation.
- On the other hand, models or simulations that do provide a likelihood are called **explicit models**. Recall for example Gaussian likelihoods.
- In this section we focus on implicit models, i.e. most simulations. Implicit inference is also called **simulation-based inference** or **likelihood-free inference** (a bit of a misnomer since we learn the likelihood from the simulation). These terms usually mean the same.

Simulators

- A simulator is a computer program that takes as input a vector of **parameters** θ, samples a series of internal states or **latent variables** $z_i \sim p_i(z_i \mid \theta, z_{<i})$, and finally produces **a data vector** $x \sim p(x \mid \theta, z)$ as output.
    - θ: the parameters of interest. **z**: Typically unobservable variables of the data generating process (including e.g. initial conditions). **x**: Observations.
- Programs that involve random samplings and are interpreted as statistical models are known as probabilistic programs, and simulators are an example.

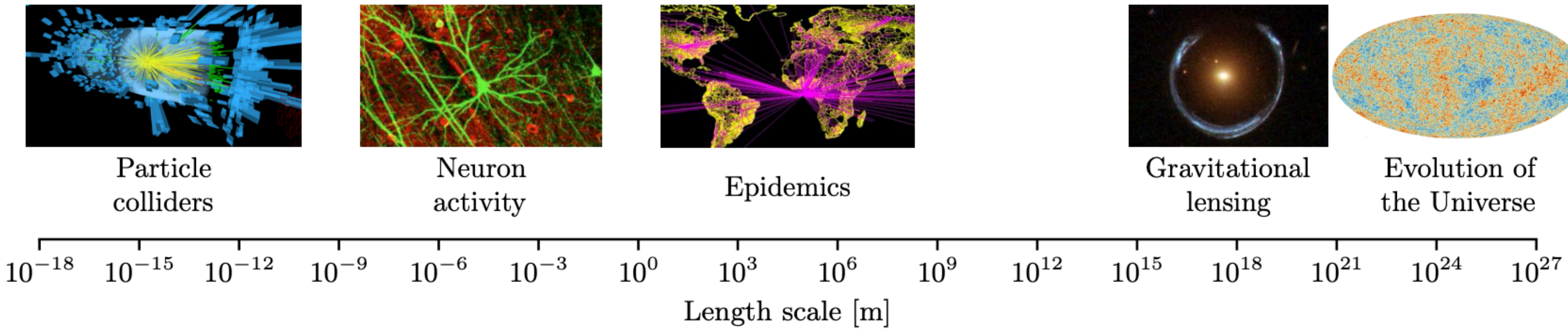
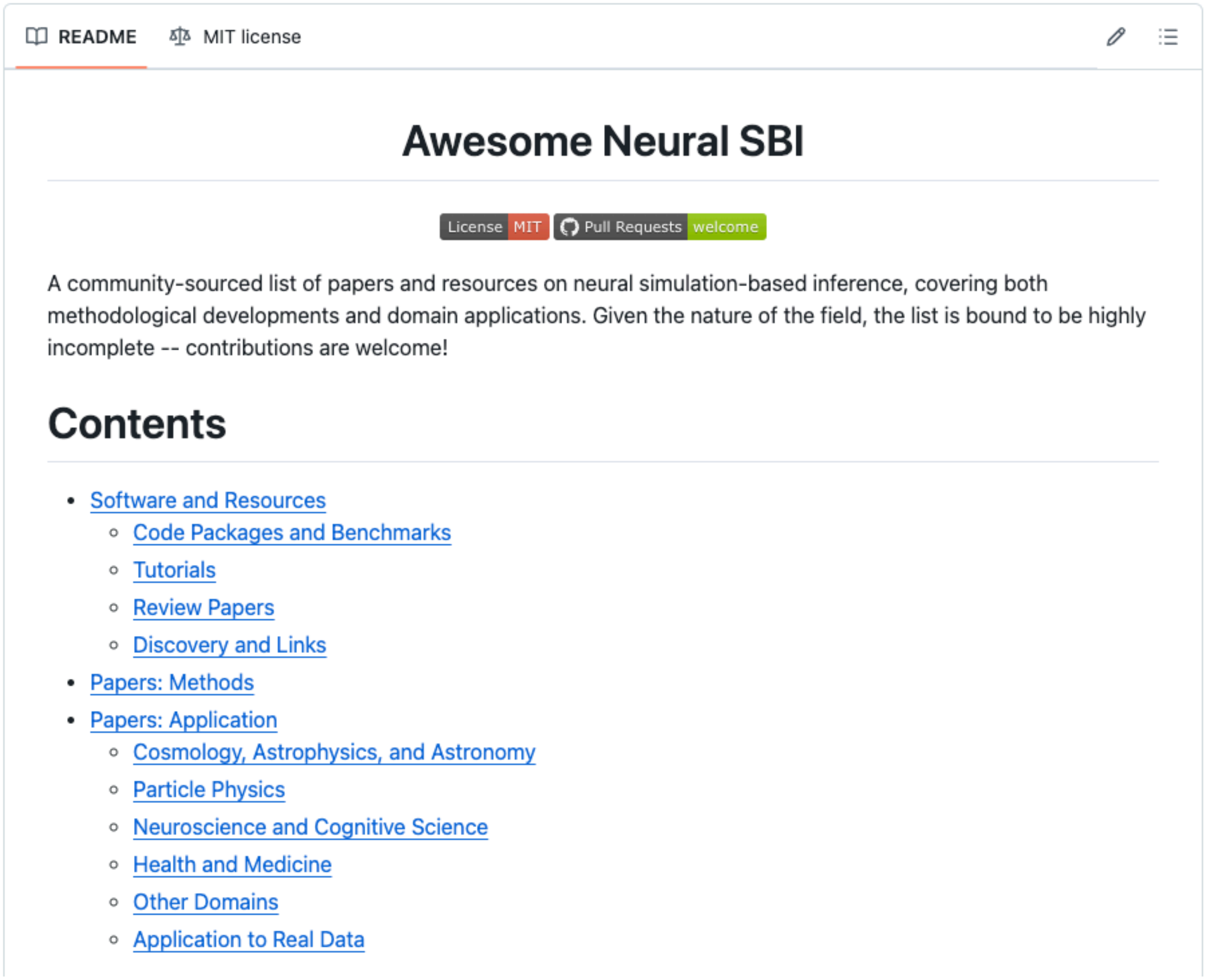


Fig. 1. Examples of phenomena at various length scales described by a diverse set of simulators, each with an intractable likelihood. Contains image material from Refs. (5–9).

List of SBI papers (method+application) all over science:

https://github.com/smsharma/awesome-neural-sbi

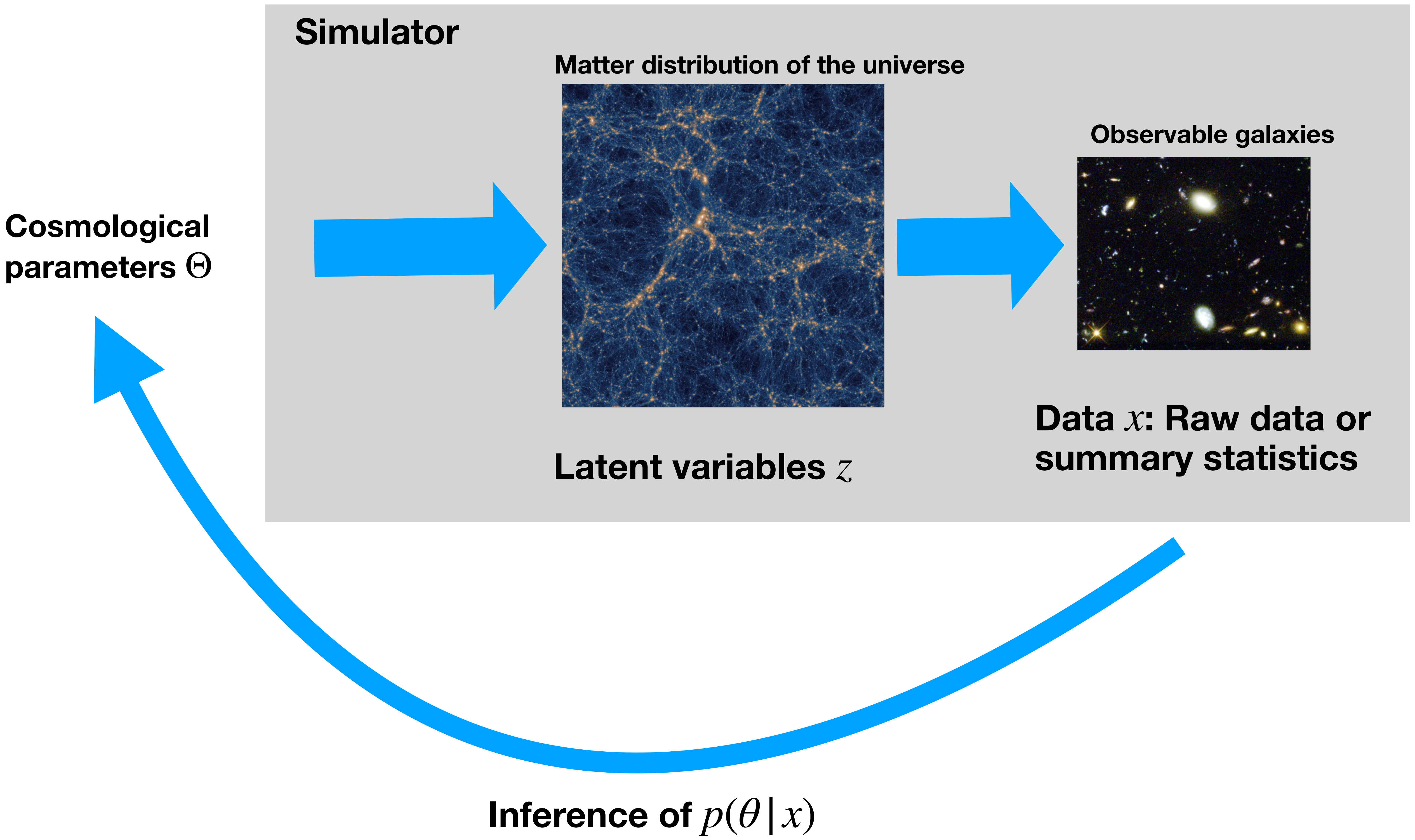


Awesome Neural SBI

A community-sourced list of papers and resources on neural simulation-based inference, covering both methodological developments and domain applications. Given the nature of the field, the list is bound to be highly incomplete -- contributions are welcome!

Contents

Example from cosmology


Simulator
Matter distribution of the universe
Observable galaxies
Cosmological parameters Θ
Latent variables z
Data x: Raw data or summary statistics
Inference of $p(\theta | x)$

Inference

- In a Bayesian data analysis, the goal is to find the posterior over the parameters given the data:

$$p(\theta|x) = \frac{p(x|\theta)\,p(\theta)}{\int \mathrm{d}\theta'\,p(x|\theta')\,p(\theta')}$$

- The likelihood is given by an intractable integral over the latent space of the simulator (e.g. we cannot simulate all possible initial conditions)

$$p(x|\theta) = \int \mathrm{d}z\; p(x, z|\theta)$$

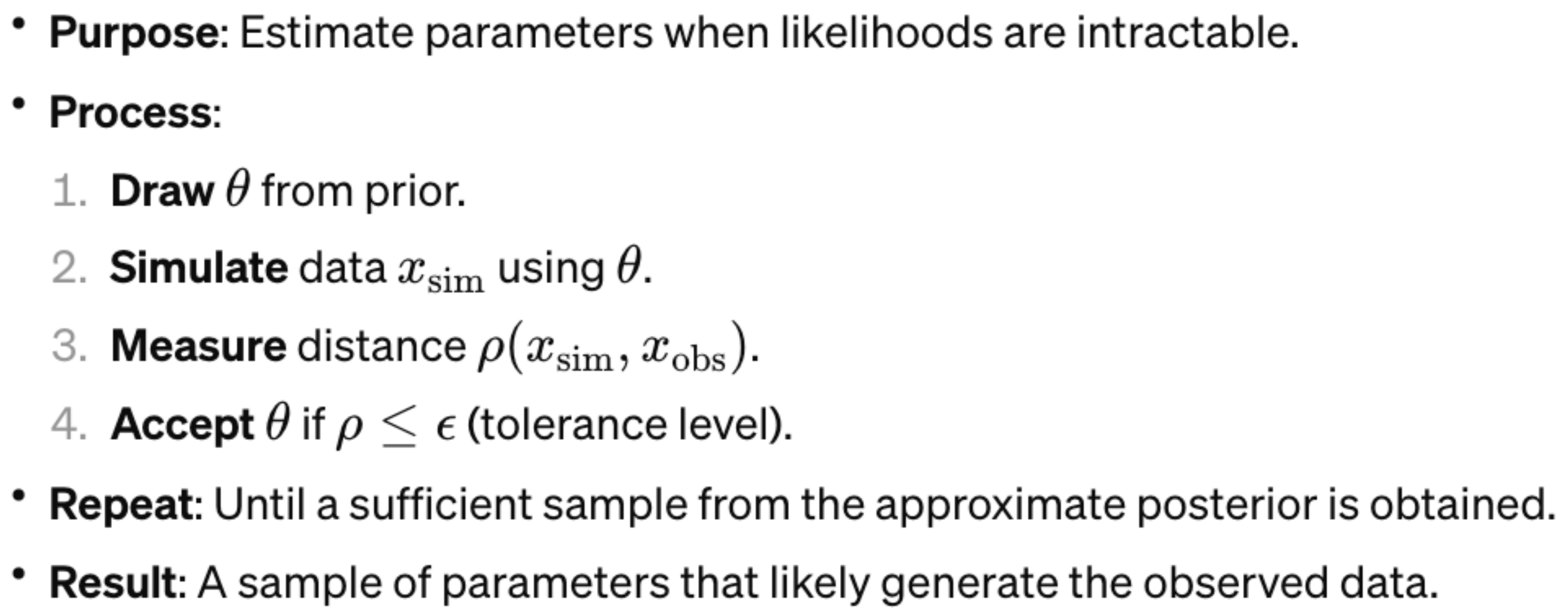
- The fundamental challenge of SBI is thus to perform Bayesian (or Frequentist) inference despite this intractability.

Summary statistics

- Often there are many possible choices what the observed data x should be.
- A **summary statistic x'=f(x)** is any compression of the raw data x.
  - Example: Instead of analyzing the raw electric field data from an antenna, we may chose to analyze the measured power spectrum.
- **Low-dimensional summary statistics are usually require**d to analyze high dimensional data to make the computation tractable.
- Traditionally the **choice of good summary statistics was left to domain experts**, which (should) know which parts of the data are important.
- However, **summary statistics are almost always lossy**, i.e. they contain less information on the parameters than the uncompressed data.
- In principle with machine learning one can **learn optimal summary** statistics directly from the simulations. SBI works both with classical and with learned summary statistics.

Approximate Bayesian Computation (ABC)

- There are two general traditional approaches of SBI, Approximate Bayesian Computation (ABC) and Density Estimation. The simplest **rejection sampling ABC algorithm** works as follows:



- **Purpose**: Estimate parameters when likelihoods are intractable.
- **Process**:
  1. **Draw** θ from prior.
  2. **Simulate** data x_{sim} using θ.
  3. **Measure** distance $\rho(x_{\mathrm{sim}}, x_{\mathrm{obs}})$.
  4. **Accept** θ if $\rho \leq \epsilon$ (tolerance level).
- **Repeat**: Until a sufficient sample from the approximate posterior is obtained.
- **Result**: A sample of parameters that likely generate the observed data.

- The tolerance ϵ controls the trade-off between the approximation quality and the computational feasibility: smaller ϵ leads to a better approximation but requires more simulations and thus more computational effort.
- **Problems**: Need a distance metric between data and simulation. No amortized inference: Need to re-run simulations when we get new data. “Throws away” simulations that are not within the tolerance level.
- There are many versions of ABC that improve its performance, e.g. Population Monte Carlo.

Classical density estimation of the likelihood

- Instead of ABC we can model the likelihood by estimating the distribution of simulated data given parameters.
- The **density estimation** algorithm works as follows:
  - Define a proposal density $\tilde{p}(\theta)$ (not necessarily the prior)
  - Draw parameters from this proposal and run many simulations to generate a data set of N pairs of parameters θ and data x: $\{(\theta_n, x_n)\}_{n=1}^{N}$
  - These are samples of the joint pdf $p(\theta, x) = p(x\,|\,\theta)\tilde{p}(\theta)$
  - From the samples we can estimate the likelihood $p(x\,|\,\theta)$ with histograms or kernel density estimation.
- **Advantage**: Amortized inference, i.e. when we take new data we can directly evaluate the likelihood without running new simulations.
- **Problem**: Potentially need more simulations than we can afford to compute (given computational limits).

Simulation-Based Inference

SBI with Neural Density Estimators

Improving SBI with machine learning

- Classical SBI has several shortcomings:
    - **Sample efficiency:** Both ABC and classical density estimation techniques suffer from the curse of dimensionality. The poor scaling means that the number of simulated samples needed to provide a good estimate of the likelihood or posterior can be prohibitively expensive.
    - **Quality of inference:** The reduction of the data to low- dimensional summary statistics invariably discards some of the information in the data about θ, which results in a loss in statistical power

- **Machine learning can improve SBI** in several ways:
    - We can **learn PDFs** with neural networks rather than using histograms or kernel density estimation. These techniques work in higher dimensions.
    - Active learning methods can systematically improve sample efficiency. Draw new simulations where they help the most.
    - Neural networks can work with very high-dimensional data. In particular we can learn optimal summary statistics of the data.

Neural density estimators (NDEs)

- Machine learning offers several methods that can be used to learn the PDF underlying a data set.
- Both unconditional and conditional PDFs can be learned.

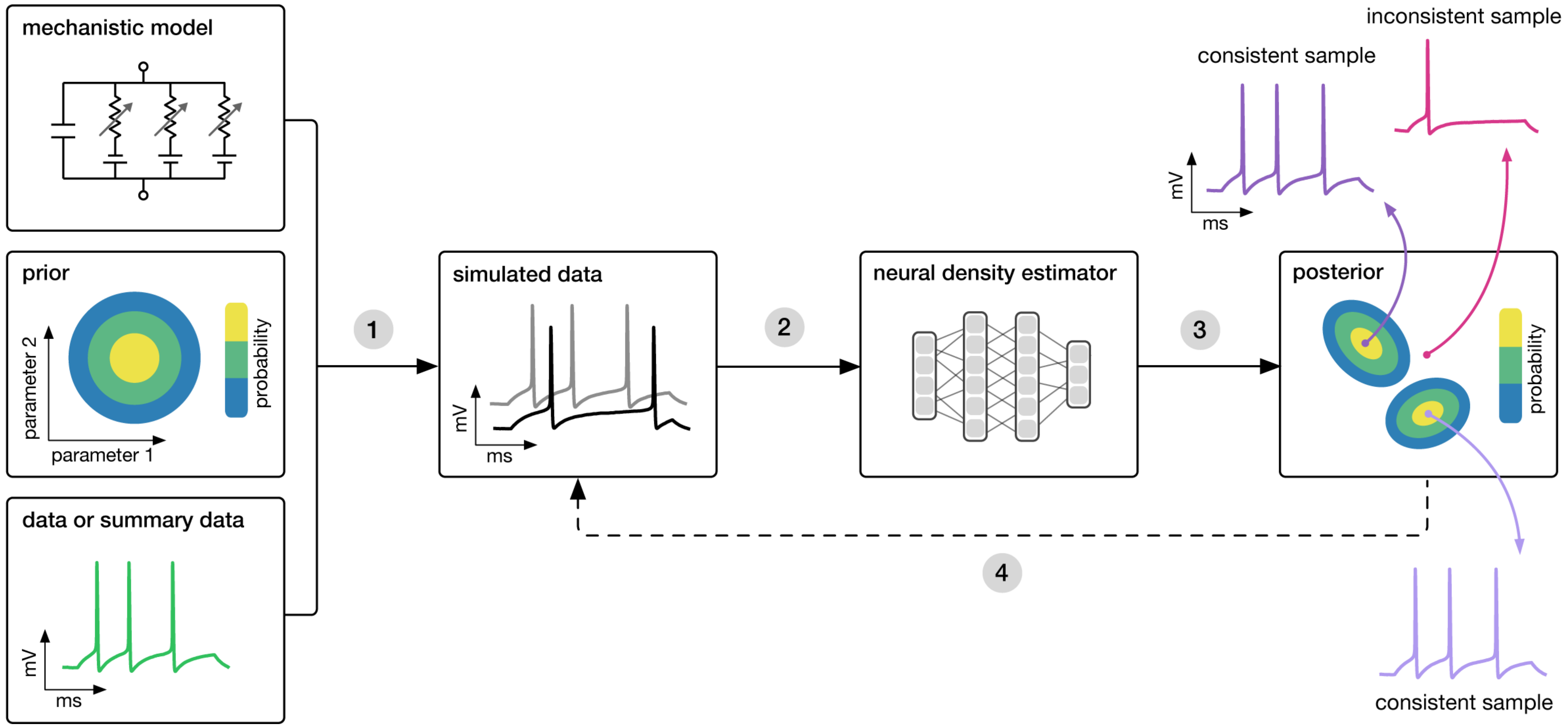
$$\{x_n\}_{n=1}^{N} \longrightarrow p(x)$$

$$\{(\theta_n, x_n)\}_{n=1}^{N} \longrightarrow p(x \mid \theta)$$

- An **NDE learns a PDF** $p(x)$ that, when sampled from, makes samples that "look like the training data". It is trained to make the training data likely under the model. Normal neural networks learn functions, NDEs learn PDFs.
- The dominant NDEs in SBI are **Normalizing Flows**. We **discussed them in detail in the last lecture**. For now just assume that we have some model that can learn PDFs from data.
- A different older NDE is a **mixture density network**. In this model a neural network outputs the parameters (means, variances, and mixture coefficients) of the mixture model (e.g. a collection of Gaussians).

Simulation-based inference with NDEs

inputs: A candidate mechanistic model (simulator), prior knowledge or constraints on model parameters, and observational data (or summary statistics thereof).



Goal: Algorithmically identify mechanistic models (simulators) which are consistent with data.

Source: https://sbi-dev.github.io/sbi/

Flavors of simulation-based inference with NDEs

- There are a number of slightly different approaches for SBI. These are (each with several variants):
  - **Neural Likelihood Estimation**
  - **Neural Posterior Estimation**
  - **Neural Ratio Estimation**

- These have different properties and in some situations one is more suitable than the other. We will discuss the first two qualitatively now.
- Mathematical details will follow after we discuss some examples.

Neural Likelihood Estimation (NLE)

- In neural likelihood estimation we learn the likelihood from simulated pairs of model parameters and data:

$$\{(\theta_n, x_n)\}_{n=1}^{N} \Longrightarrow \mathscr{L}(x\,|\,\theta)$$

- After training the NDE on our simulated data, we can then evaluate the likelihood of observed data from our measurement.

$$\mathscr{L}(x^{obs}\,|\,\theta)$$

- Now we can proceed with normal Bayesian data analysis. That usually means that we sample from the posterior with MCMC:

$$p(\theta\,|\,x^{obs}) \propto \mathscr{L}(x^{obs}\,|\,\theta)p(\theta)$$

- NLE is amortized (no new sims needed for new data). However sometimes it takes too much training data to learn the likelihood everywhere. Sequential Neural Likelihood Estimation (S-NLE) only learns the likelihood near the data and thus saves samples, at the cost of not being amortized anymore.

Neural Posterior Estimation (NPE)

- You might wonder why why learn the likelihood and not the posterior which is our ultimate goal. Learning the posterior is indeed a possibility.
- From a simulated data set

$$\{(\theta_n, x_n)\}_{n=1}^N$$

drawn from a proposal density $\tilde{p}(\theta)$ it is possible to directly learn the posterior

$$p(\theta | x)$$

- An advantage of learning the posterior directly is that we do not need to run an MCMC anymore. The model directly outputs the desired posterior, i.e. our parameter measurement. A disadvantage is that it is difficult to explore different prior distributions of the parameters.

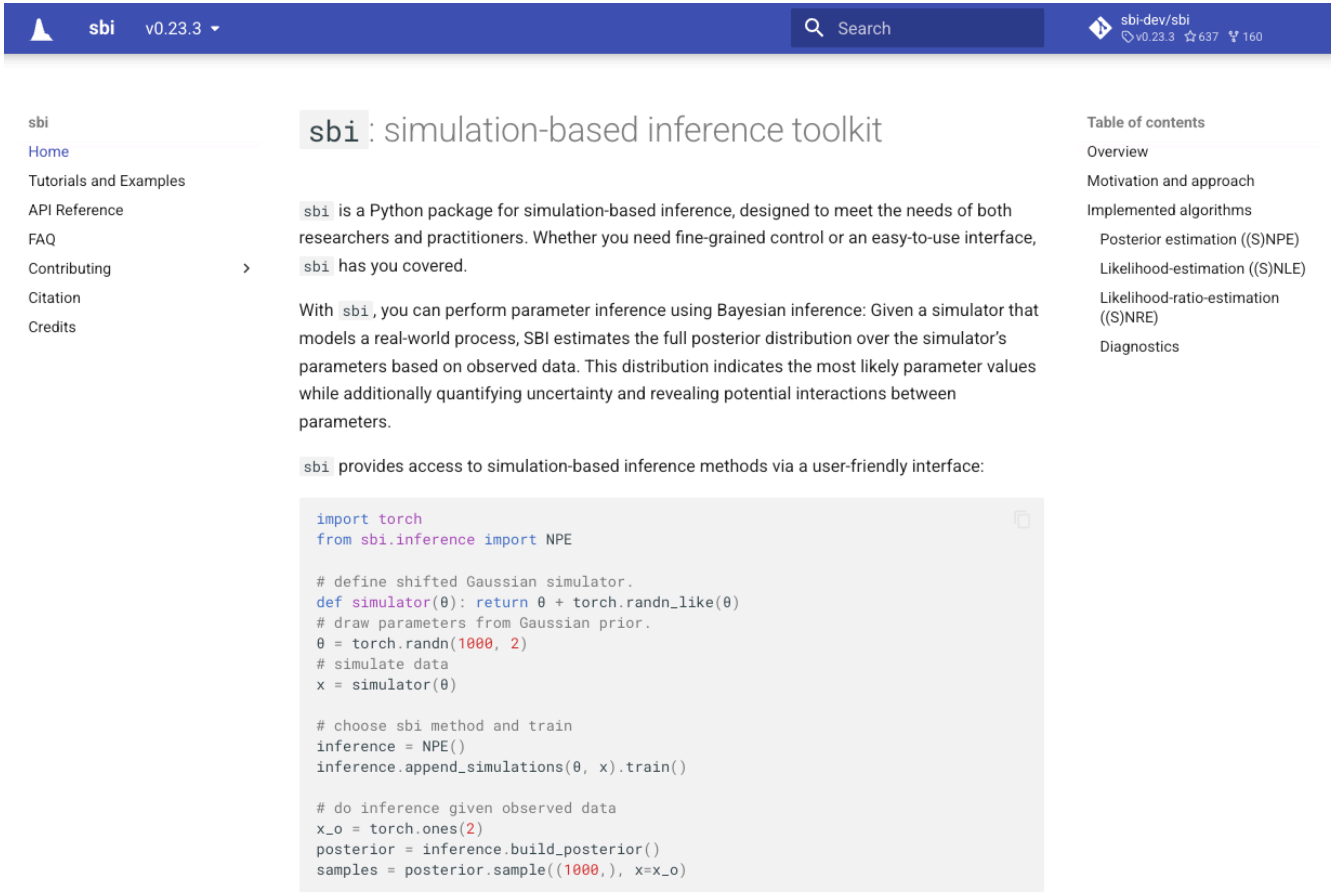
Testing the neural density estimator

- After training the NLE or NPE, it is important to **assess that the learned PDFs are correct**, and we thus do not over- or under-estimate parameter uncertainties. Most importantly, we want to be sure that our posterior is not overconfident.
- For example, if we did not train the NDE on enough simulations, we will get incorrect likelihoods and posteriors.
- It is not possible to formally guarantee that our machine learning model is correct. However, one can **run a series of tests**. In general, the more test simulations we have (independent from the training simulations), the more convincing our tests can be.
- Typical tests include in particular
  - **Posterior Predictive Checks (PPC)**
  - **Simulation-based calibration**
  - For more details, see https://sbi-dev.github.io/sbi/ diagnostics.

Simulation-Based Inference

Examples of SBI

- We will use the **“sbi” python library, a popular option.** There are other codes, for example the more general probability framework “pyro”.



`sbi`: simulation-based inference toolkit

`sbi` is a Python package for simulation-based inference, designed to meet the needs of both researchers and practitioners. Whether you need fine-grained control or an easy-to-use interface, `sbi` has you covered.

With `sbi`, you can perform parameter inference using Bayesian inference: Given a simulator that models a real-world process, SBI estimates the full posterior distribution over the simulator’s parameters based on observed data. This distribution indicates the most likely parameter values while additionally quantifying uncertainty and revealing potential interactions between parameters.

`sbi` provides access to simulation-based inference methods via a user-friendly interface:

```
import torch
from sbi.inference import NPE

# define shifted Gaussian simulator.
def simulator(θ): return θ + torch.randn_like(θ)
# draw parameters from Gaussian prior.
θ = torch.randn(1000, 2)
# simulate data
x = simulator(θ)

# choose sbi method and train
inference = NPE()
inference.append_simulations(θ, x).train()

# do inference given observed data
x_o = torch.ones(2)
posterior = inference.build_posterior()
samples = posterior.sample((1000,), x=x_o)
```

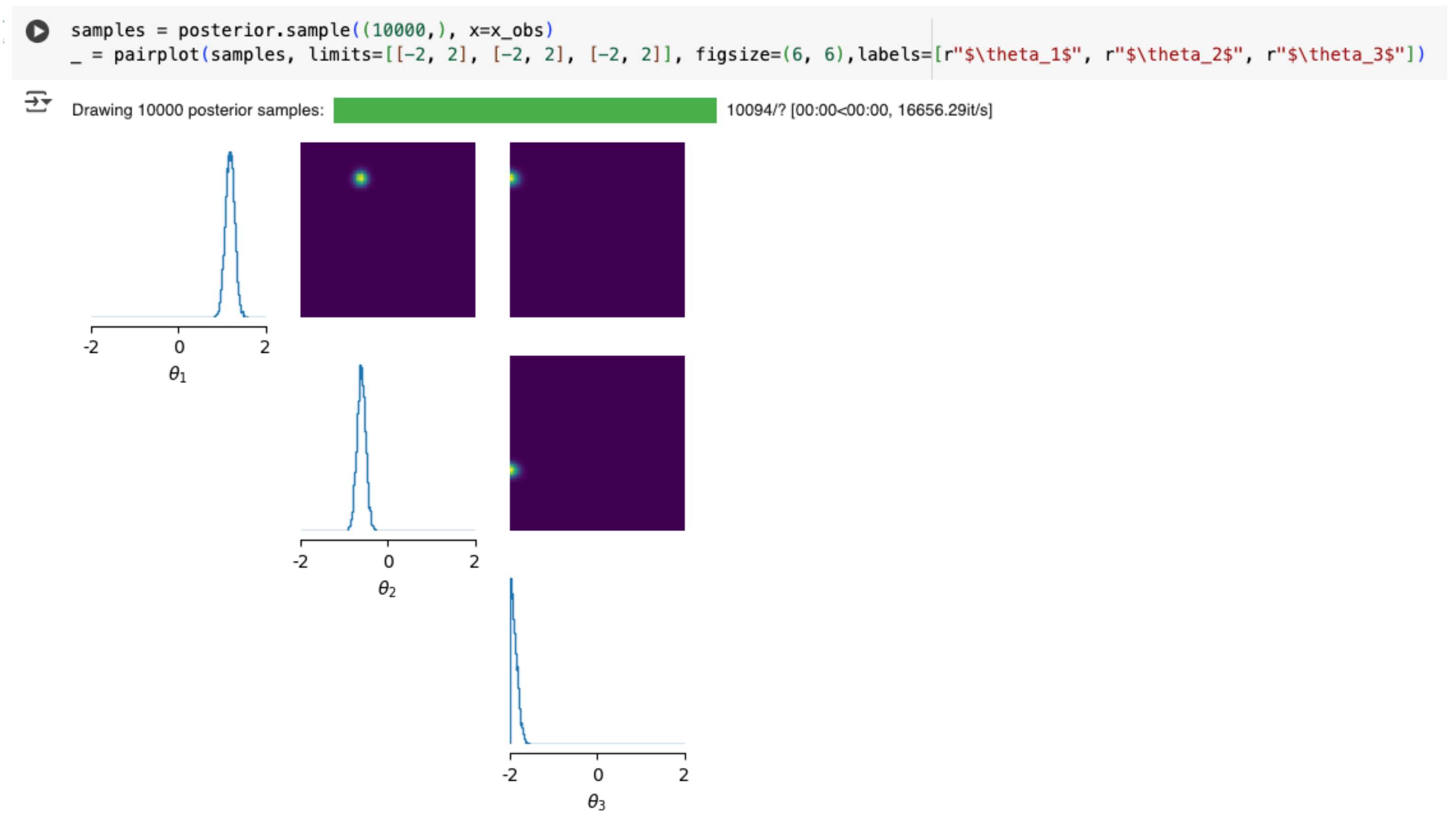
Gaussian Toy example

- We will first have a look at the Gaussian demo 00 and 01 from https://github.com/sbi-dev/sbi/tree/main/tutorials

- Model parameters θ: 3 parameters
- Output parameters x: 3 parameters

- This example uses Neural Posterior Estimation.

- (Discussion on Colab)



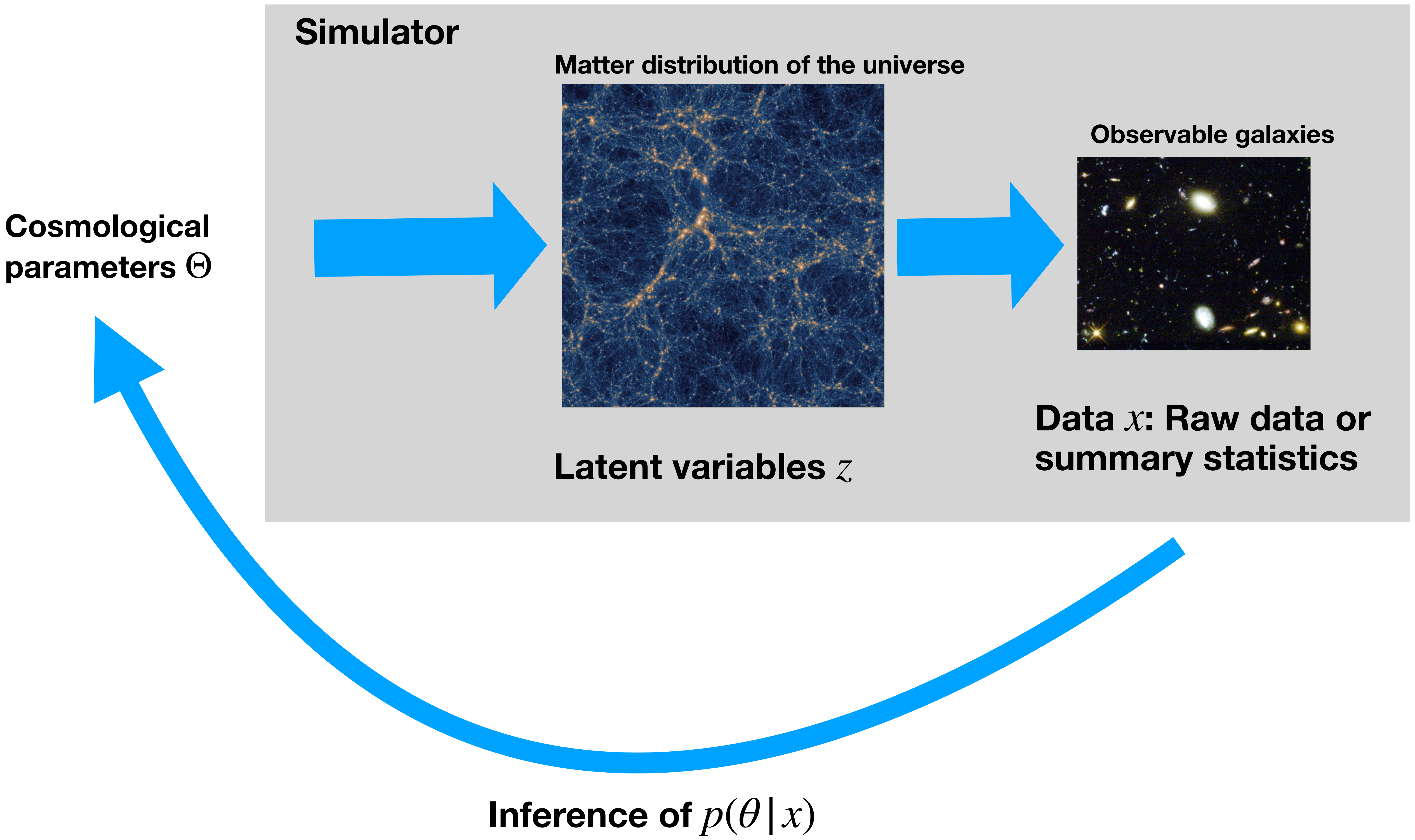
```
samples = posterior.sample((10000,), x=x_obs)
_ = pairplot(samples, limits=[[-2, 2], [-2, 2], [-2, 2]], figsize=(6, 6),labels=[r"$\theta_1$", r"$\theta_2$", r"$\theta_3$"])
```

```
Drawing 10000 posterior samples: 10094/? [00:00<00:00, 16656.29it/s]
```

SBI in Cosmology

- SBI is used in cosmology to extract more information about the fundamental parameters of the universe from the observed galaxy distribution.
- We will have quick look at these papers:
    - https://arxiv.org/pdf/2211.00723.pdf SIMBIG: A Forward Modeling Approach To Analyzing Galaxy Clustering
    - https://arxiv.org/pdf/2310.15246.pdf SIMBIG: The First Cosmological Constraints from Non-Gaussian and Non-Linear Galaxy Clustering
- The SBI method these papers use is **NPE with a Masked Autoregressive Flow**.
-

Recall example from cosmology


Simulator
Matter distribution of the universe
Observable galaxies
Cosmological parameters Θ
Latent variables z
Data x: Raw data or summary statistics
Inference of $p(\theta | x)$

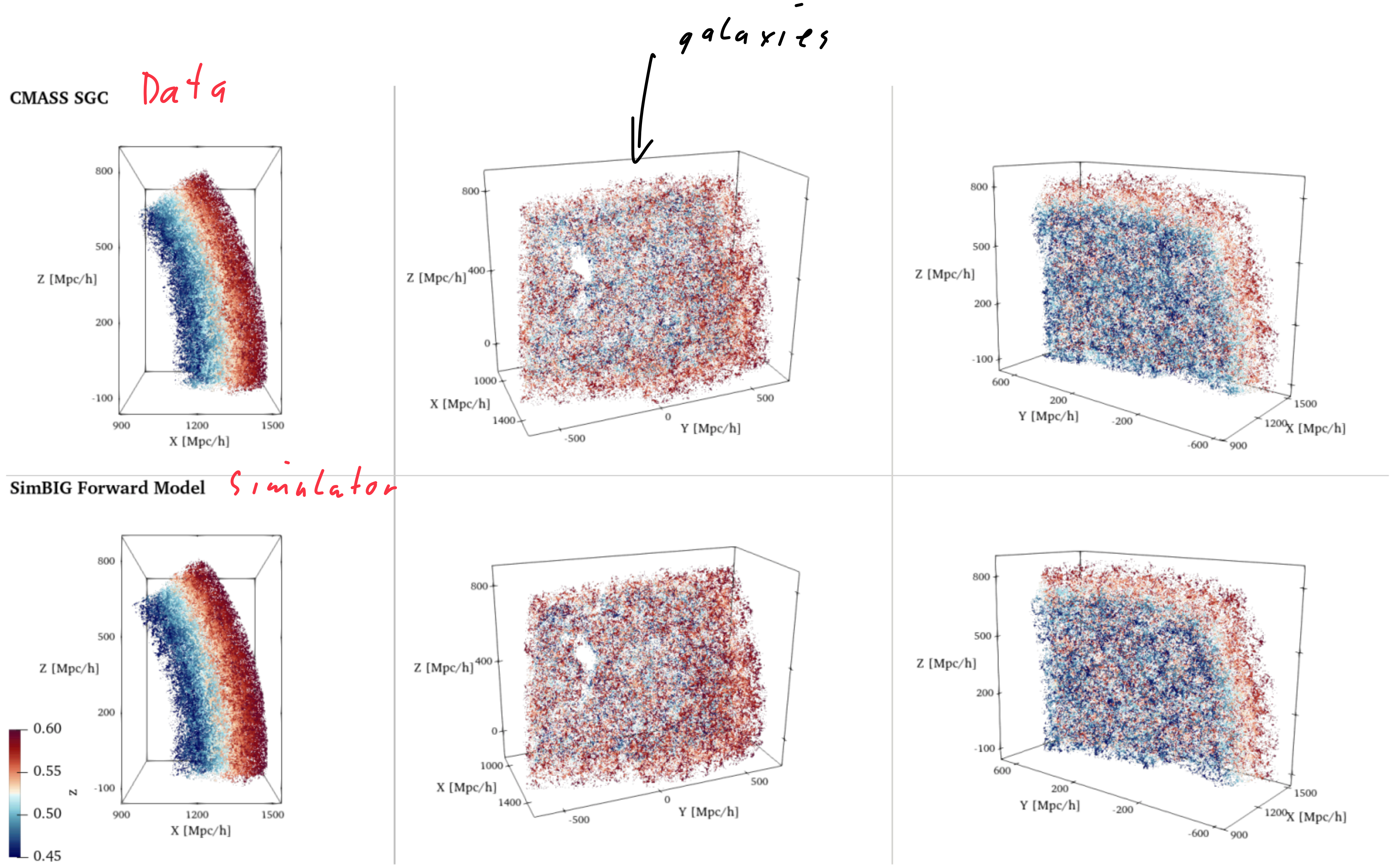


Fig. 1. The SimBIG forward model produces simulated galaxy samples with the same survey geometry and observational systematics as the observed BOSS CMASS SGC galaxy sample. We present the 3D distribution of the galaxies from three different viewing angles. The colormap represents the redshift of the galaxies. In the top set of panels, we present the distribution of galaxies in the CMASS sample. In the bottom, we present the distribution of a simulated galaxy sample, generated from our forward model. The SimBIG galaxy samples are constructed from Quijote N-body dark matter simulations using an HOD model that populates dark matter halos identified using the Rockstar algorithm. The 3D distributions illustrate that our forward model is able to generate galaxy distributions that are difficult to statistically distinguish from observations. For more comparisons of the 3D distributions, we refer readers to .

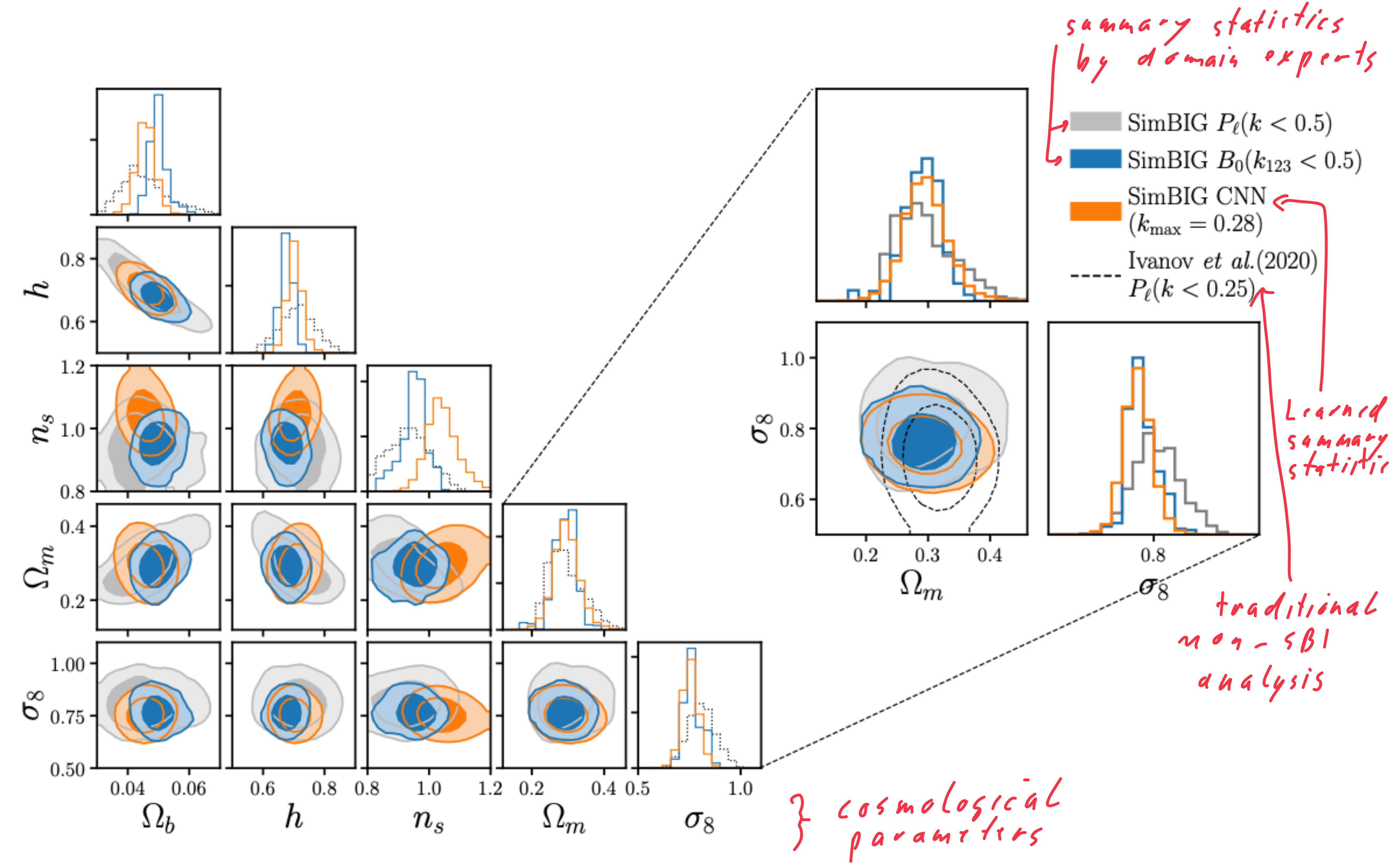


Figure 1. *Left*: Posteriors of cosmological parameters inferred from B_0 (blue) and CNN (orange) using SimBIG. All posteriors include a ω_b prior from BBN studies. The contours mark the 68 and 95 percentiles. For comparison, we include the posterior from the SimBIG P_ℓ analysis (gray). *Right*: We focus on the posteriors of Ω_m and σ_8, the parameters that can be most significantly constrained by galaxy clustering. We include the posterior from the Ivanov et al. (2020) PT-based $P_\ell(k < 0.25\,h/\text{Mpc})$ analysis for reference (black dashed). The SimBIG B_0 and CNN constraints are significantly tighter yet consistent with P_ℓ constraints.

Course logistics

- **Reading for this lecture:**
    - This lecture was based mostly on https://arxiv.org/abs/1911.01429