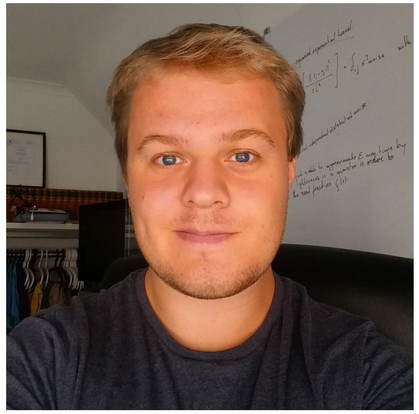
Mining for Dark Matter substructure: Learning from lenses without a likelihood

Johann Brehmer

New York University

DMWG seminar, RIKEN
February 17, 2020

Siddharth Mishra-Sharma



Joeri Hermans

Gilles Louppe

Kyle Cranmer

NYU

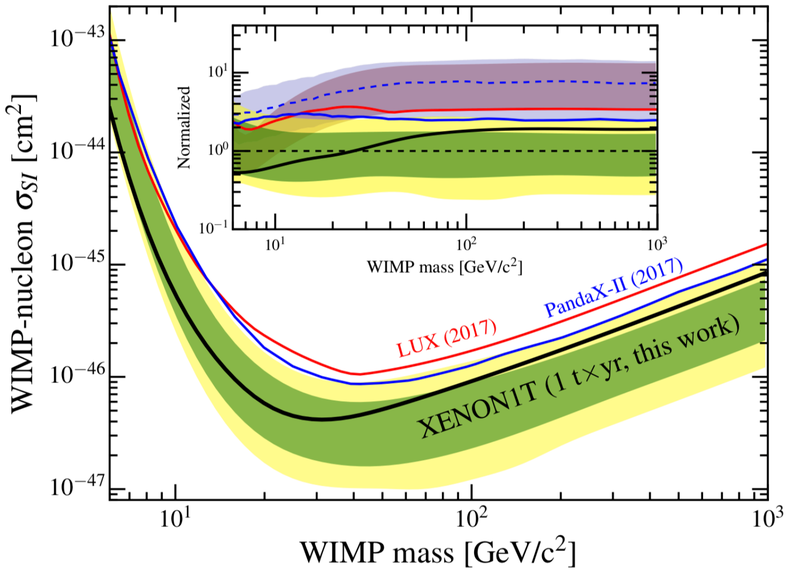
MSDSE
Moore - Sloan
Data Science Environments

diana hep

iris hep
Institute for Research and Innovation
in Software for High Energy Physics

The SCAILFIN Project
scailfin.github.io

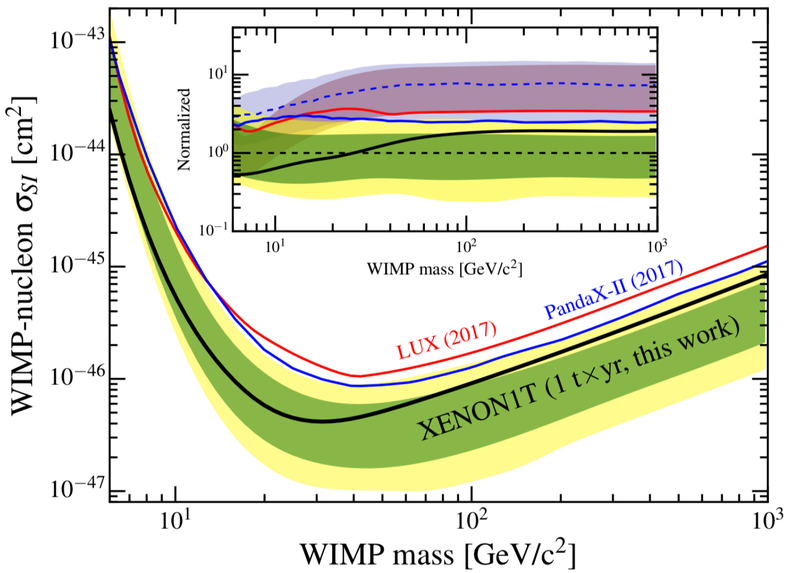
Dark matter



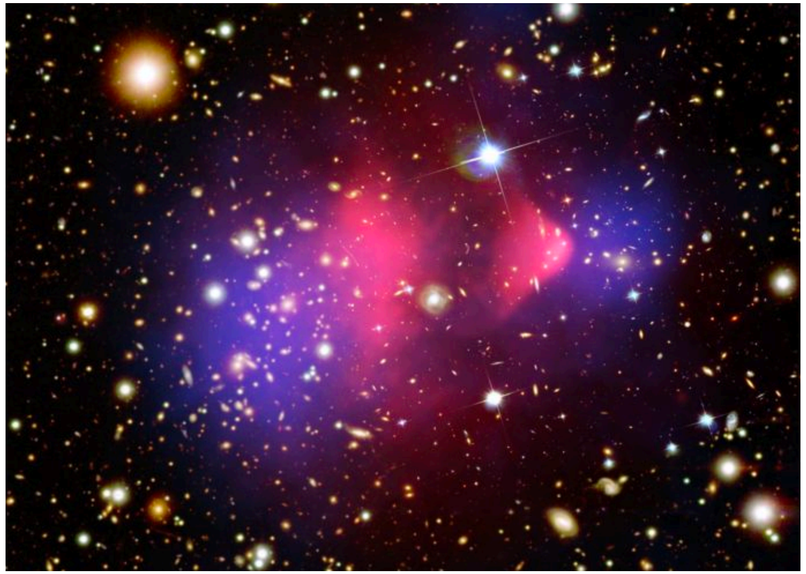
[XENON 1805.12562]

Non-gravitational interactions:
no evidence so far

Dark matter



[XENON 1805.12562]

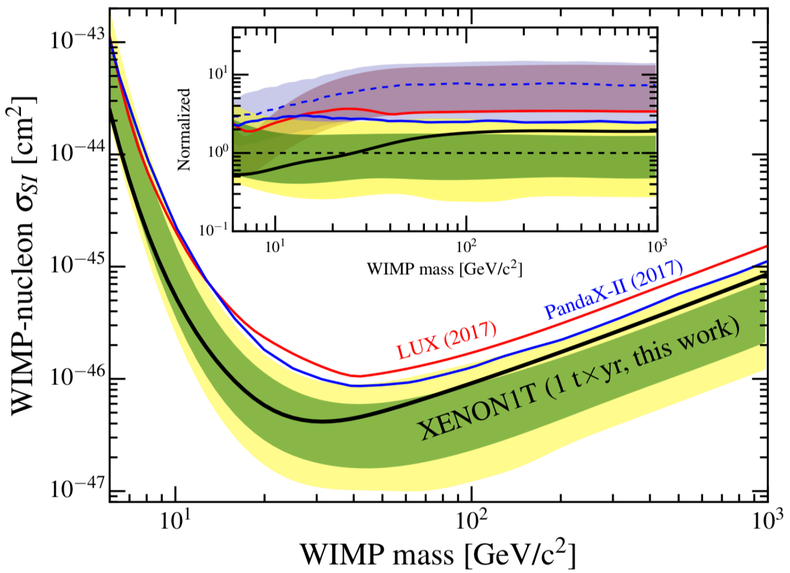


[NASA]

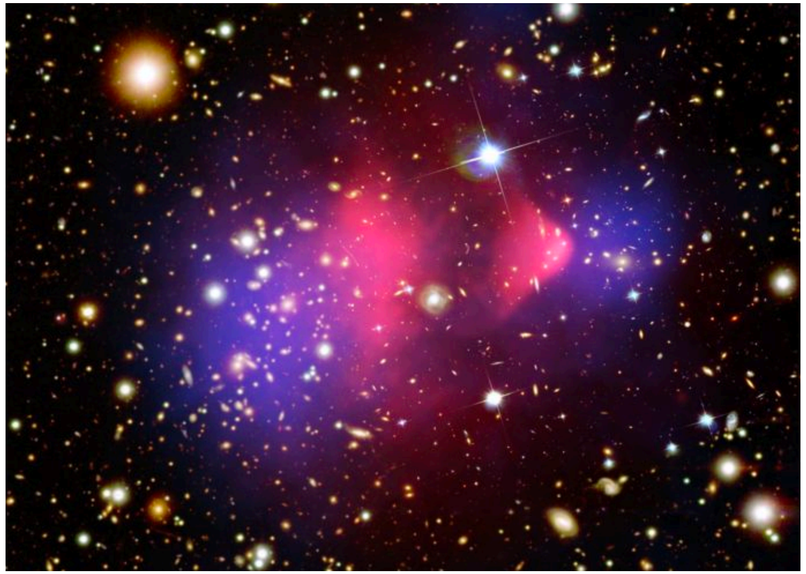
Non-gravitational interactions: no evidence so far

Gravitational interactions at large length scales: plenty of evidence, consistent with ΛCDM

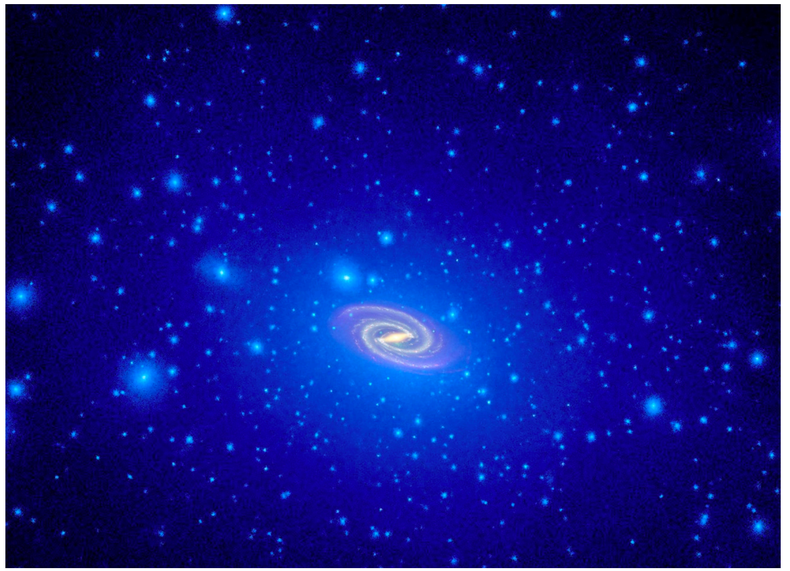
Dark matter



[XENON 1805.12562]



[NASA]



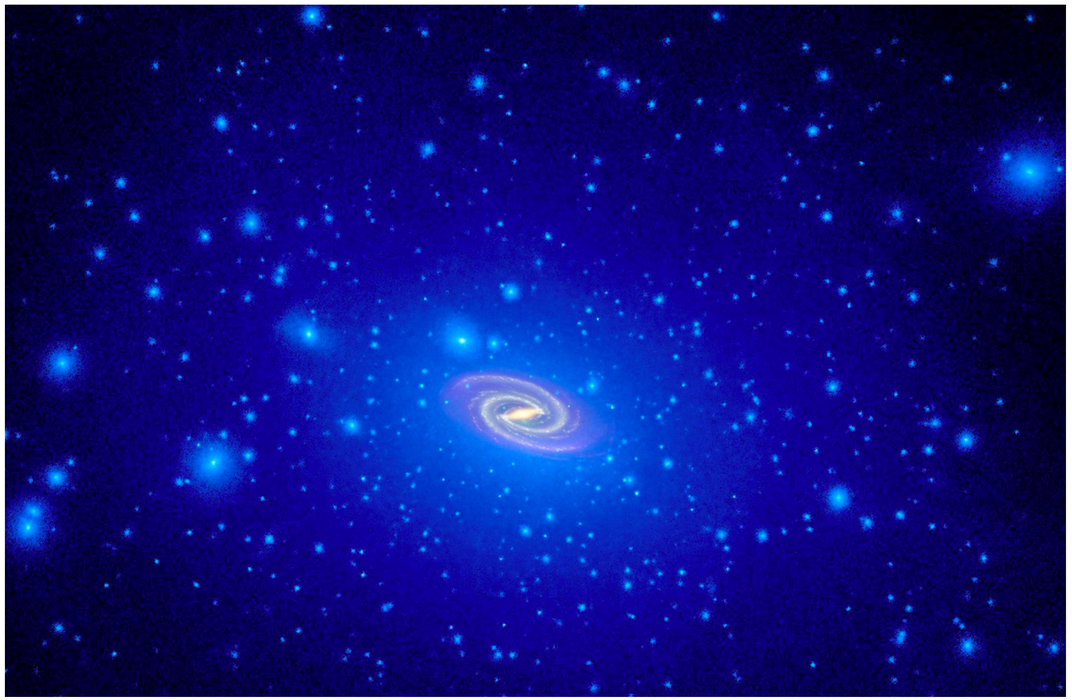
[T. Brown, J.Tumlinson]

Non-gravitational interactions: no evidence so far

Gravitational interactions at large length scales: plenty of evidence, consistent with ΛCDM

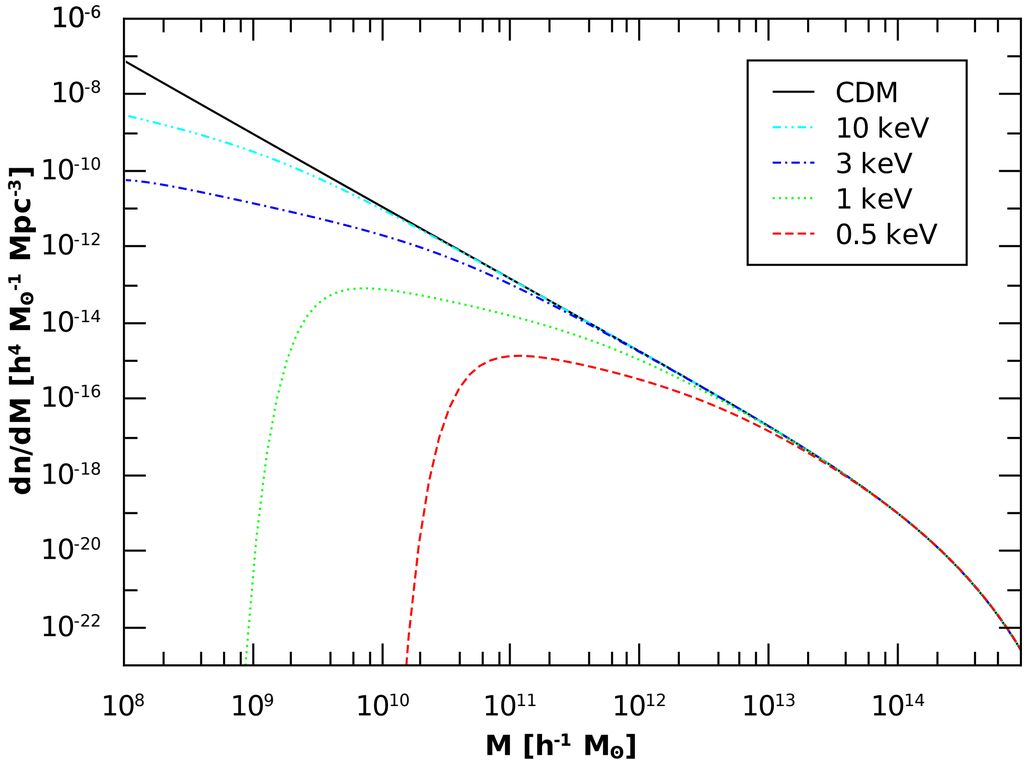
Gravitational interactions at small scales (DM substructure): beginning to be probed, many models predict deviations from ΛCDM

DM small-scale structure is a probe of its particle nature



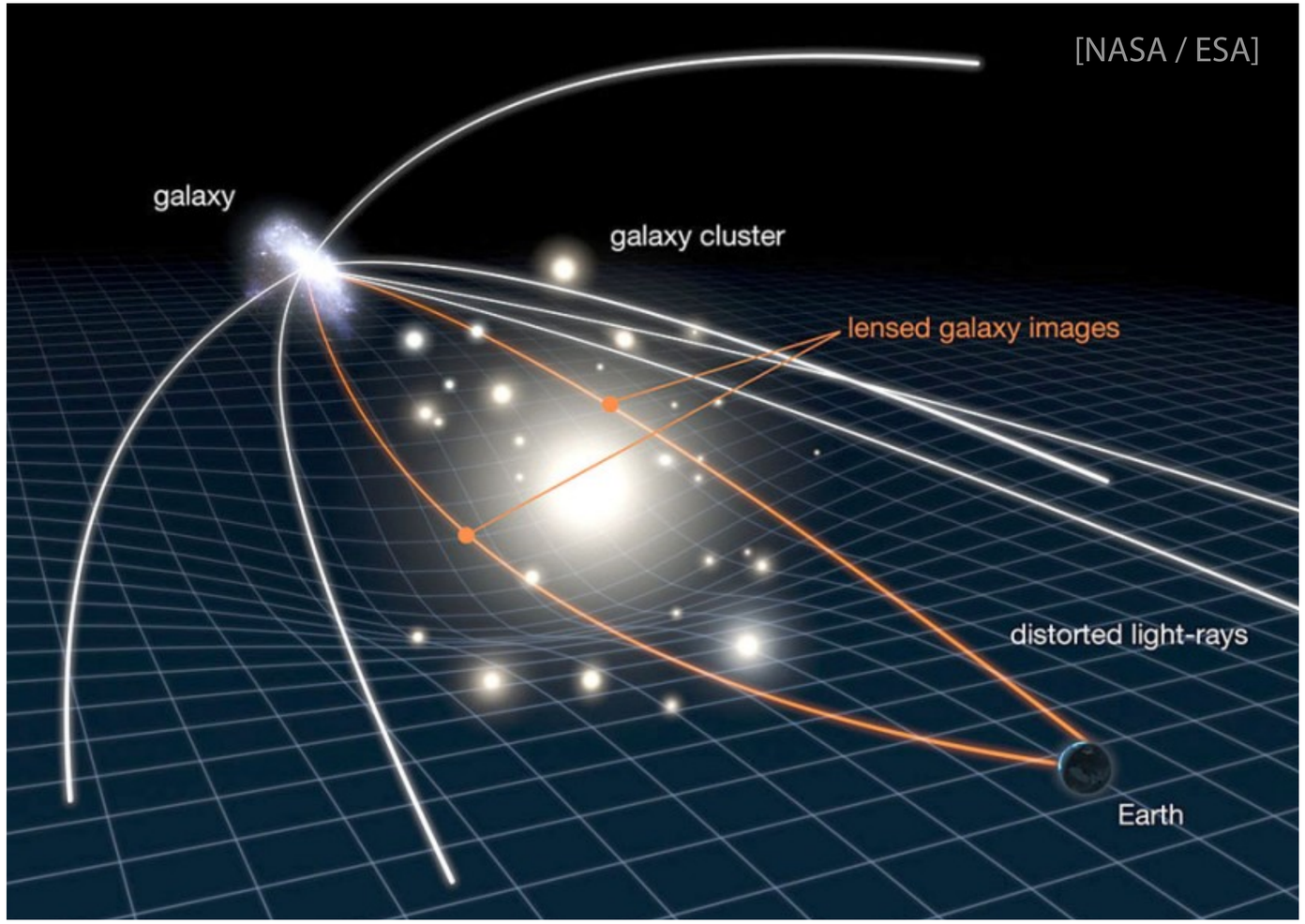
[T. Brown, J.Tumlinson]

Abundance of DM subhalos vs mass:



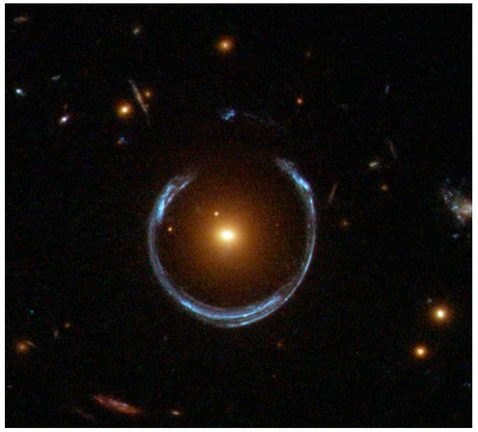
[R. Dunstan et al 1109.6291]

Strong gravitational lensing

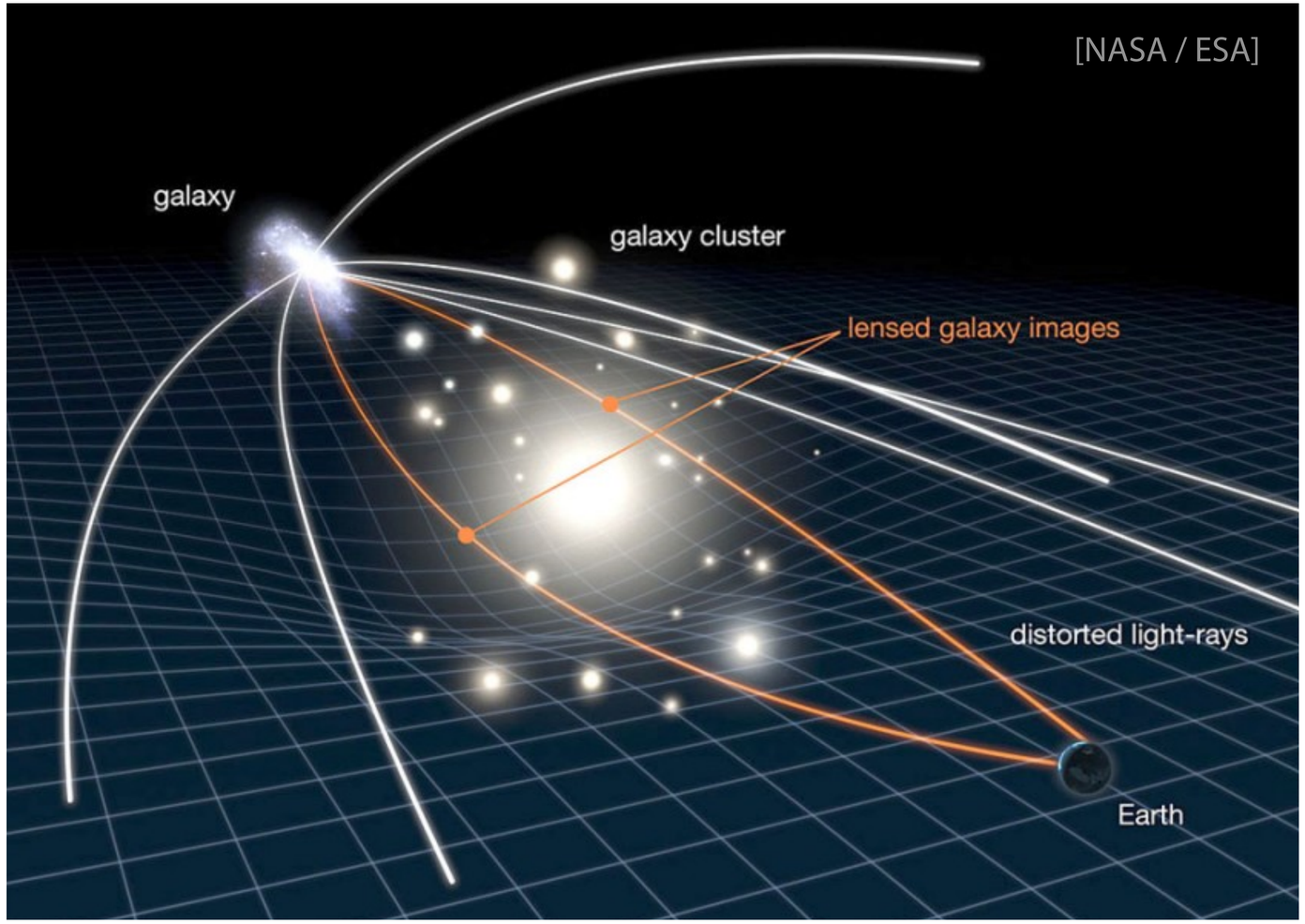


Multiple images of quasars

Extended arcs from galaxies

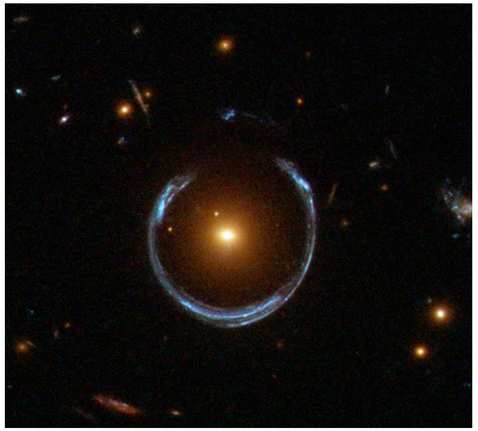


Strong gravitational lensing

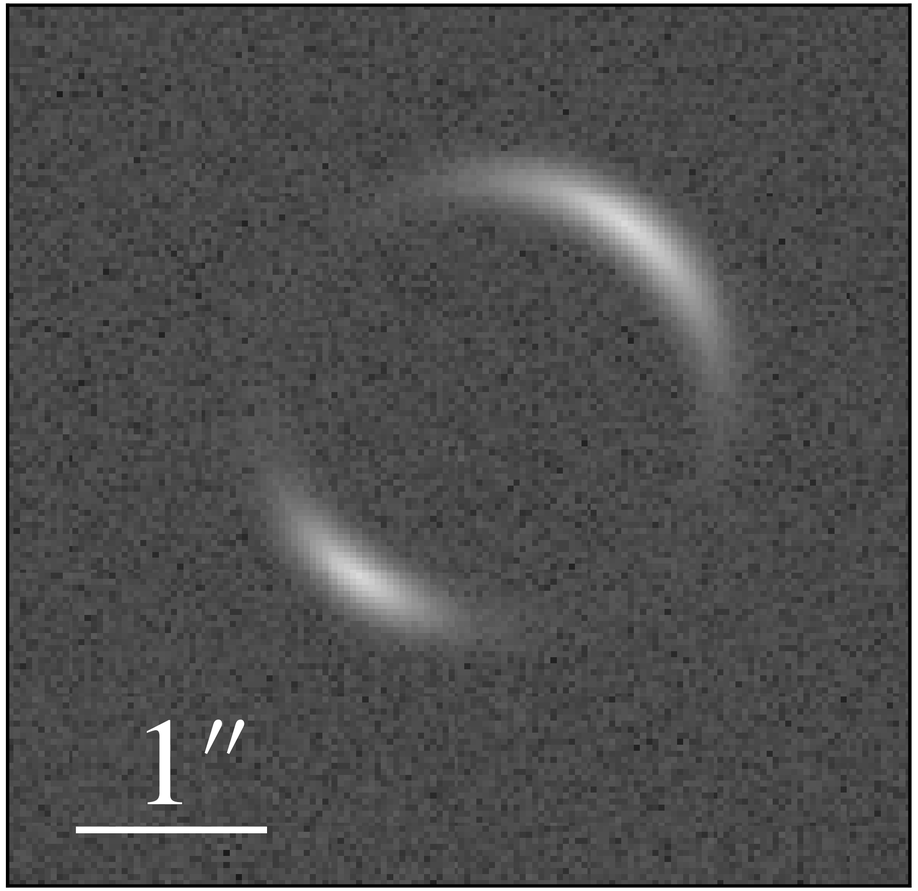


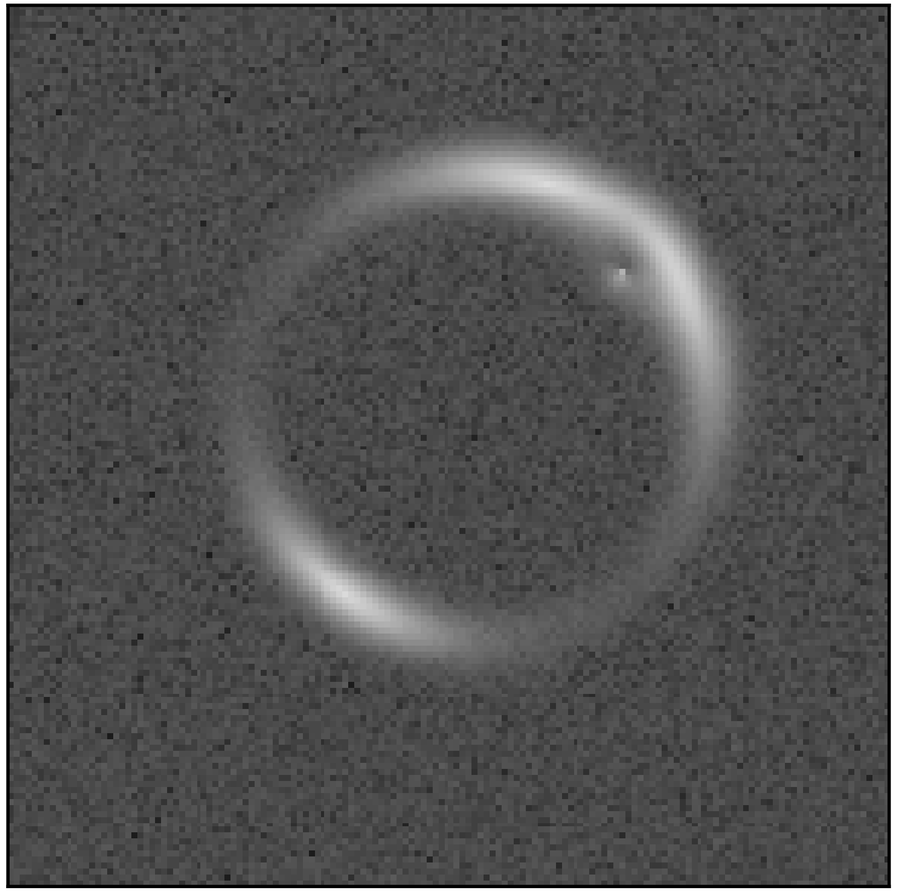
Multiple images of quasars

Extended arcs from galaxies



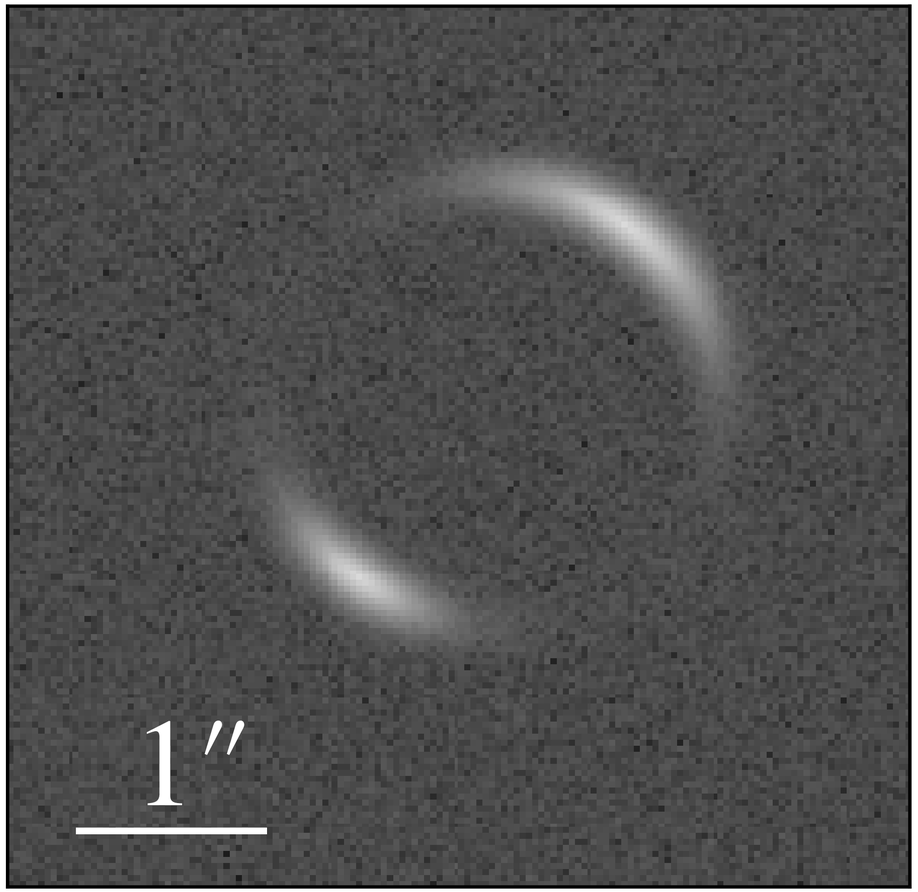
Subhalos affect strong lensing



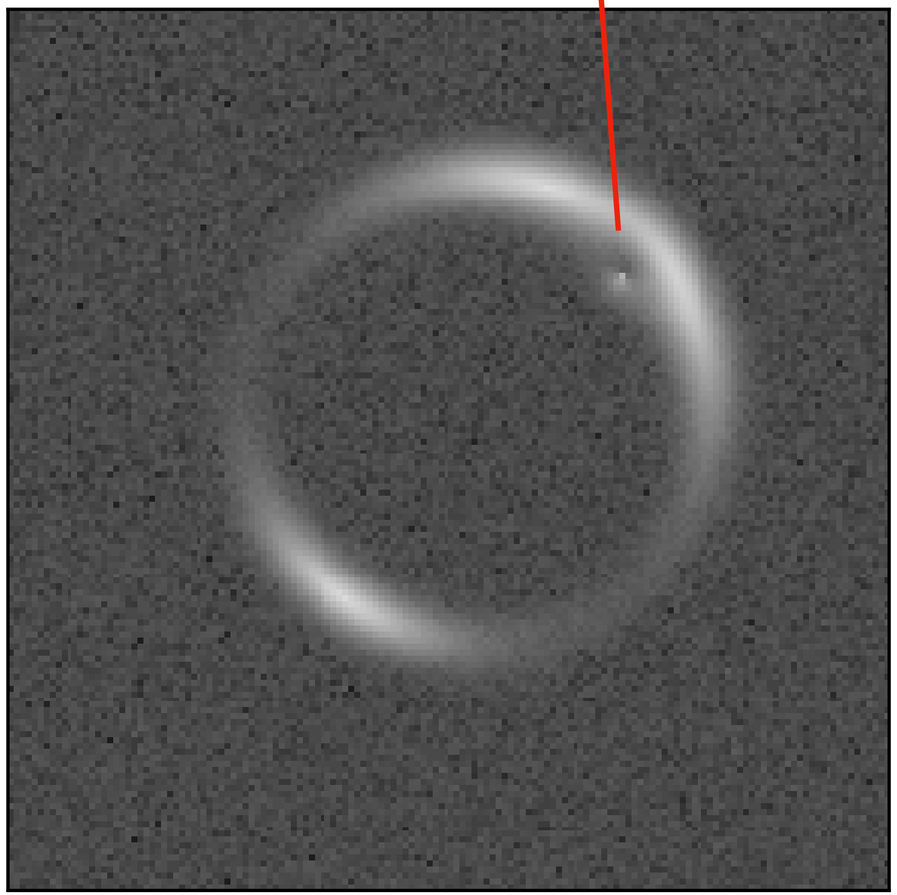


Subhalos affect strong lensing

Smooth halo only

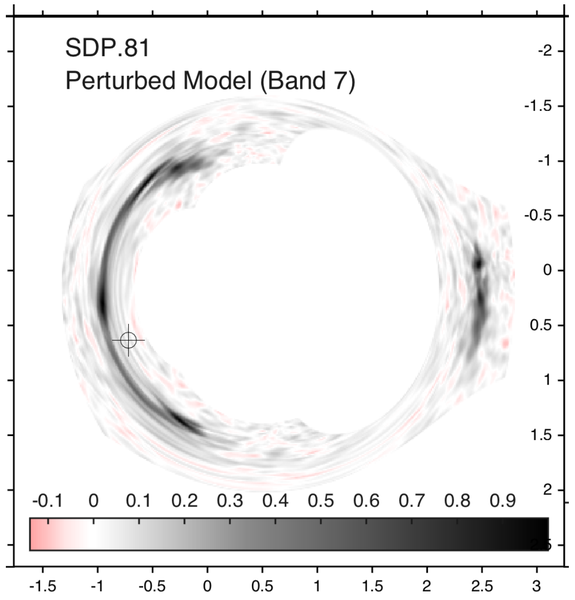


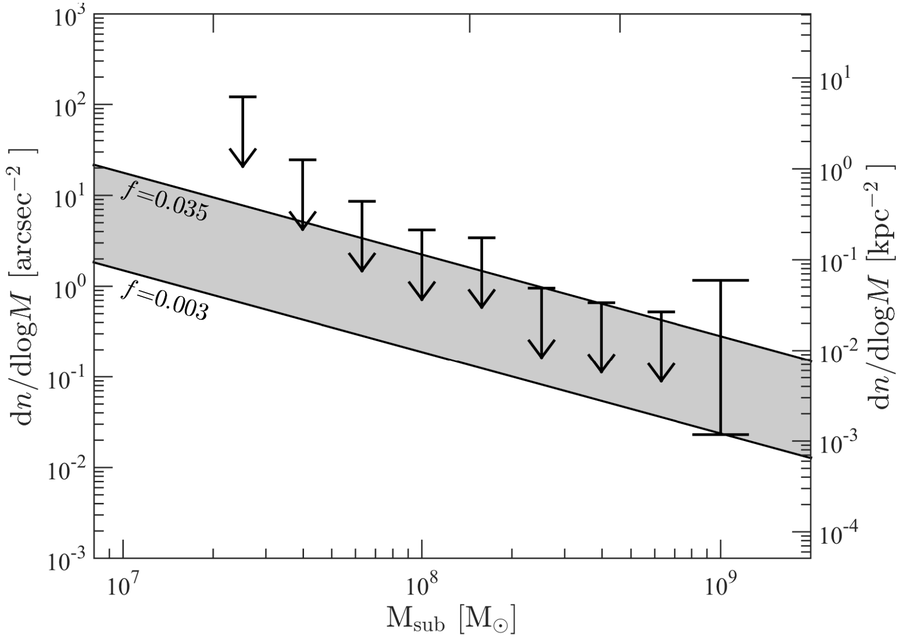
Smooth halo + subhalo



Inference clump by clump

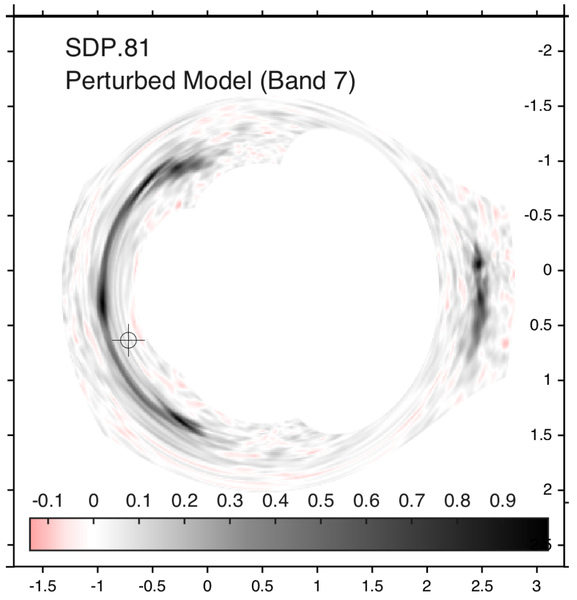
- Individual heavy subhalos leave a relatively clear signal and can be identified [Hezaveh et al 1601.01388]

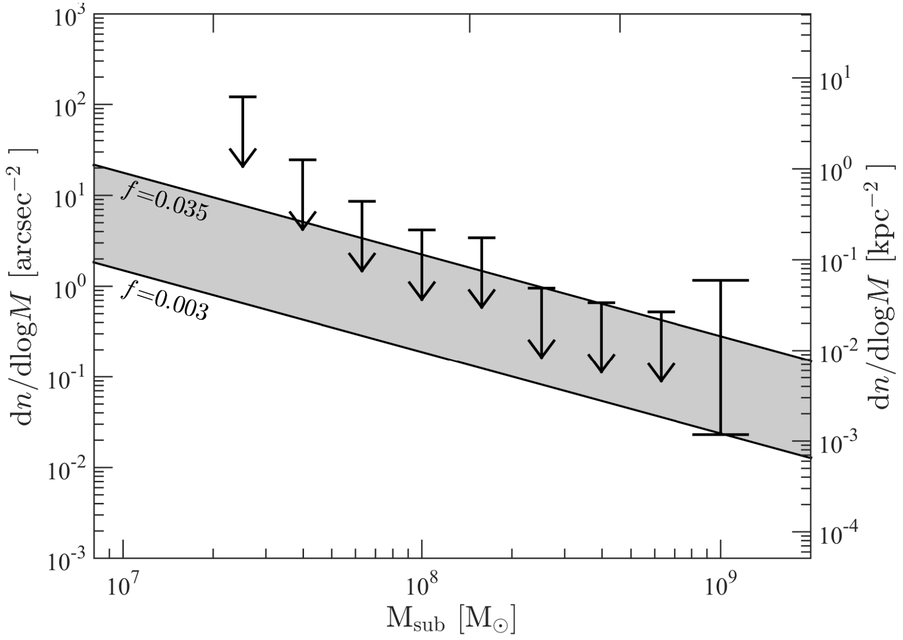




Inference clump by clump

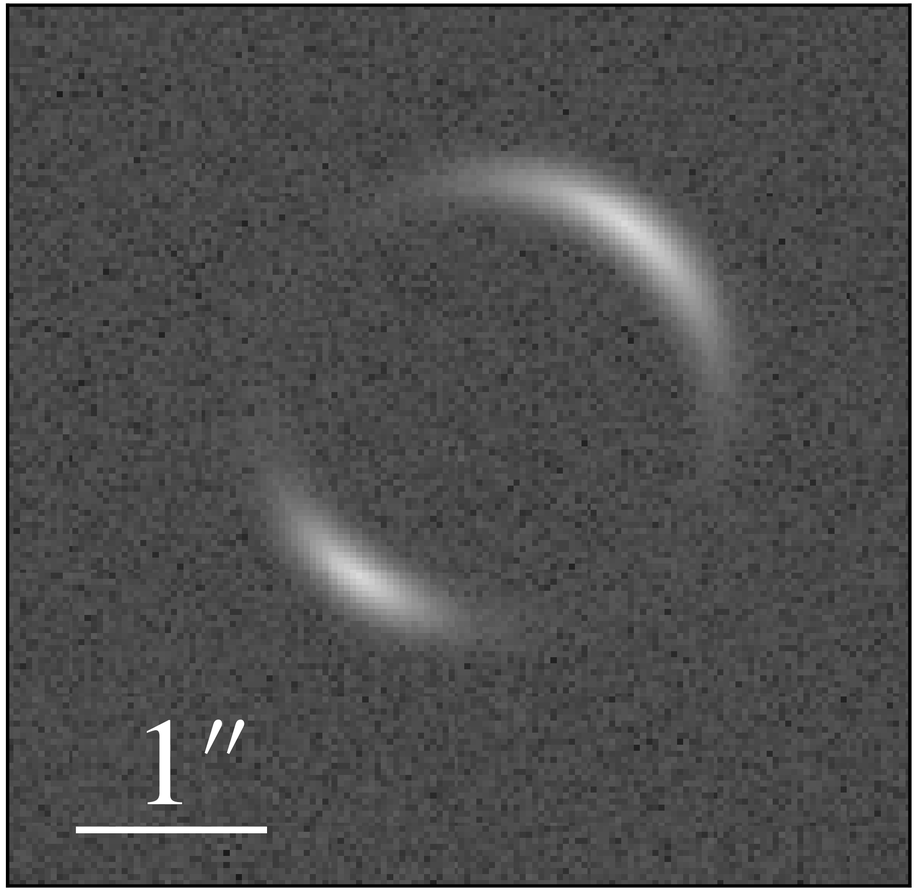
- Individual heavy subhalos leave a relatively clear signal and can be identified [Hezaveh et al 1601.01388]

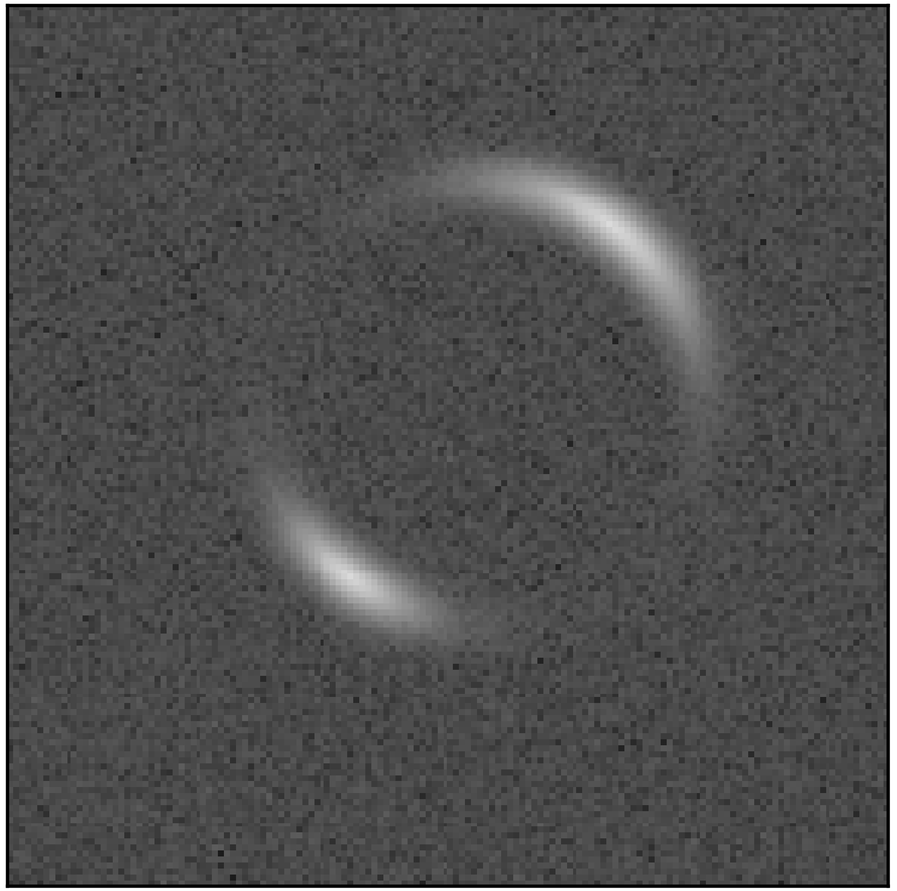




- Transdimensional inference allows simultaneously inferring multiple individual subhalos and population-level parameters [B. Brewer et al 1508.00662; T. Daylan et al 1706.06111]
- Works for subhalos $\sim 10^9 \mathrm{M}_\odot$, scales poorly to many lighter subhalos (or many observed lenses)

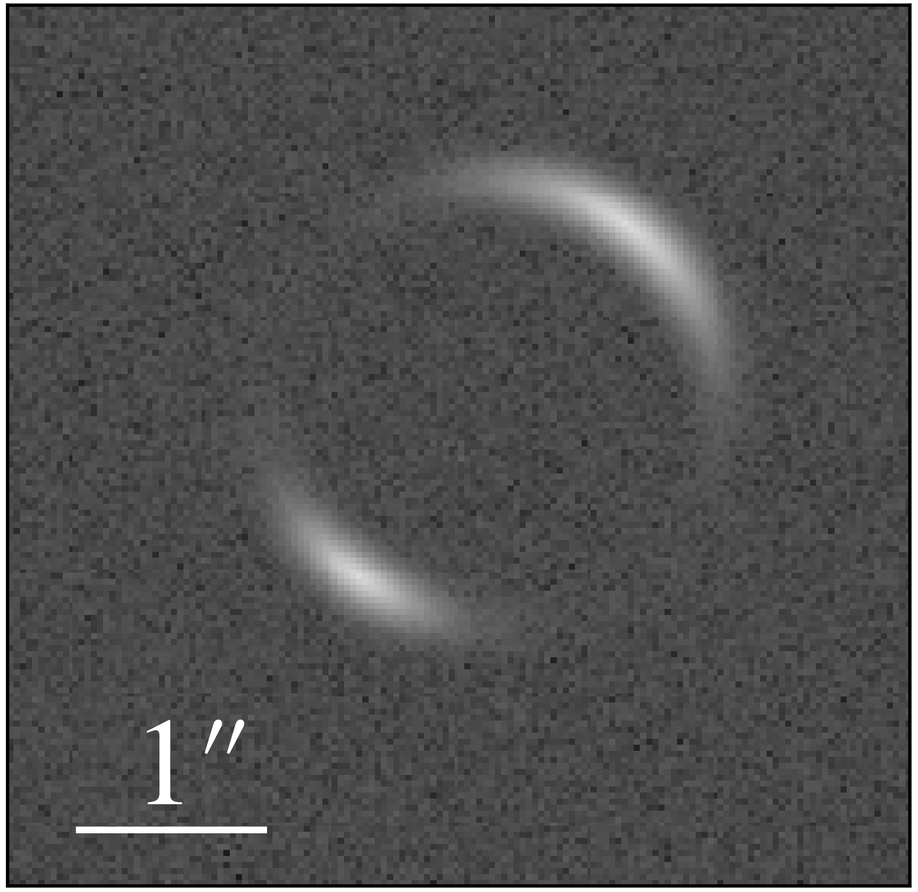
Many light subhalos collectively affect strong lensing



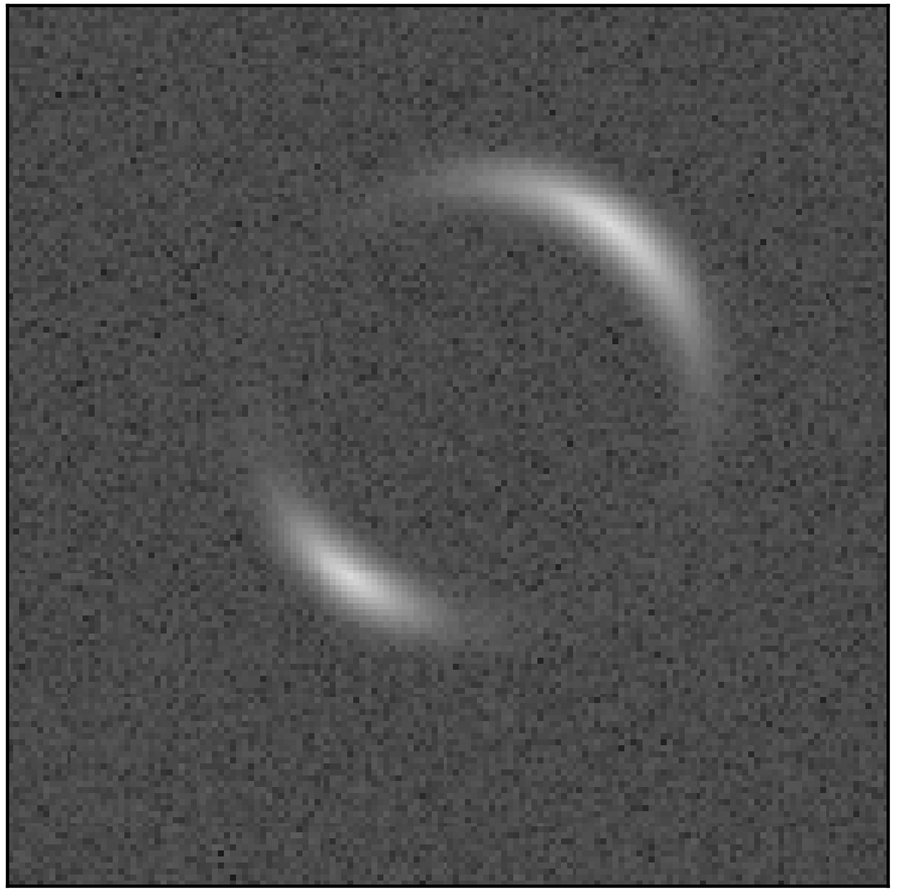


Many light subhalos collectively affect strong lensing

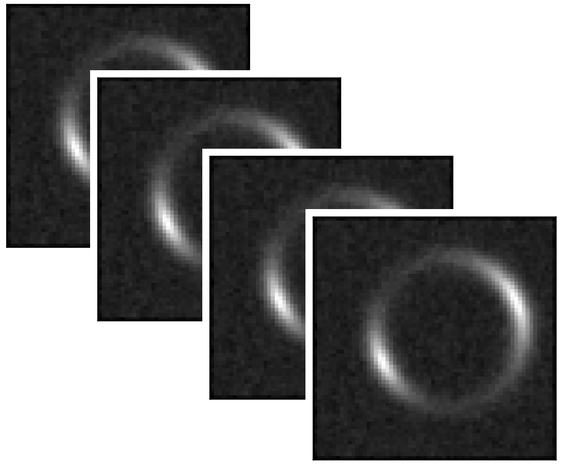
Smooth halo only

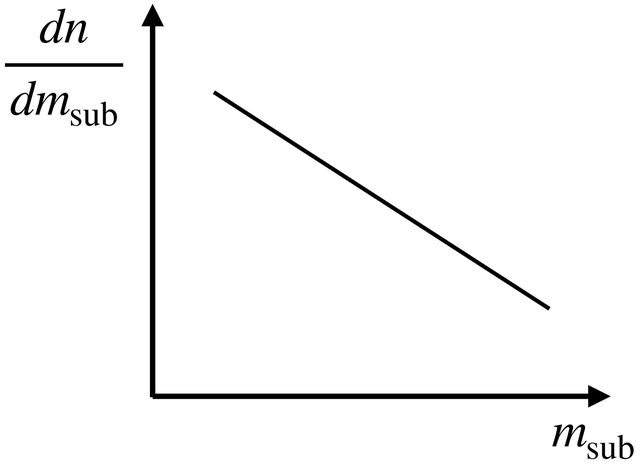


Smooth halo + subhalos



Goal: Scalable inference for small subhalos

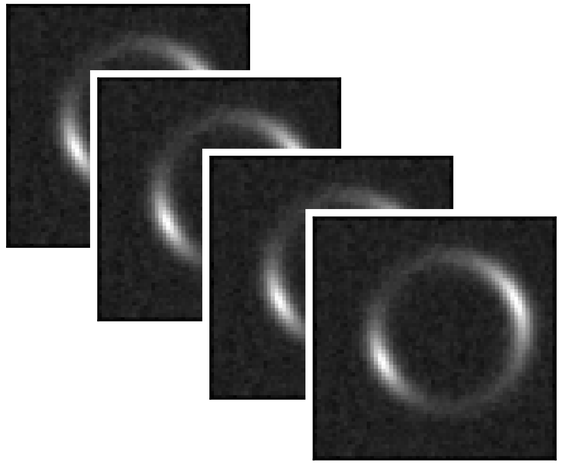


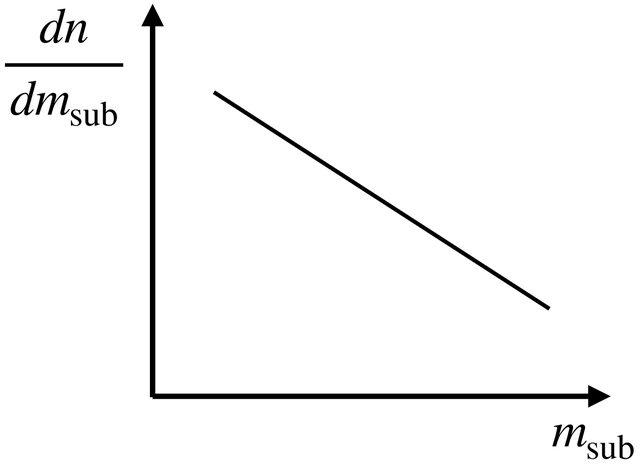


Future surveys (LSST, Euclid) are expected to deliver large samples of galaxy-galaxy strong lenses [Collett et al 1507.02657]

Goal: infer subhalo mass distribution through collective effects of many light subhalos

Goal: Scalable inference for small subhalos





Future surveys (LSST, Euclid) are expected to deliver large samples of galaxy-galaxy strong lenses [Collett et al 1507.02657]

Goal: infer subhalo mass distribution through collective effects of many light subhalos

⇒ Need inference technique that

- scales to many lenses
- can deal with a large number of subhalos
- captures subtle effects in high-dimensional image data

A key challenge is that
we cannot calculate the likelihood function.

Modeling strong lensing

Model parameters

Subhalo mass function parameters

θ

Modeling strong lensing

Model parameters

Subhalo mass function parameters

θ

Latent variables

Source / host halo properties

z_{source} z_{lens}

Subhalo number, masses, positions

n_{sub} $\{z_{\mathrm{sub}\,i}\}$

$$p(z_{\mathrm{source}}) \quad p(z_{\mathrm{lens}}) \quad p(n_{\mathrm{sub}}|\theta) \prod_{i}^{n_{\mathrm{sub}}} p(z_{\mathrm{sub}\,i}|\theta)$$

Modeling strong lensing

Model parameters

Subhalo mass function parameters

θ

Latent variables

Source / host halo properties

$z_{\rm source}$ $z_{\rm lens}$

Subhalo number, masses, positions

$n_{\rm sub}$ $\{z_{{\rm sub}\,i}\}$

Observables

Observed lens image

x

$p(z_{\rm source})$ $p(z_{\rm lens})$ $p(n_{\rm sub}|\theta) \prod_i^{n_{\rm sub}} p(z_{{\rm sub}\,i}|\theta)$ $p(x|z_{\rm source}, z_{\rm lens}, \{z_{{\rm sub}\,i}\})$

Modeling strong lensing

Model parameters

Subhalo mass function parameters

θ

Latent variables

Source / host halo properties

z_{source} z_{lens}

Subhalo number, masses, positions

n_{sub} $\{z_{\mathrm{sub}\,i}\}$

Observables

Observed lens image

x

We can easily write a simulator that samples from

$$p(x, z|\theta) = p(z_{\mathrm{source}}) \quad p(z_{\mathrm{lens}}) \quad p(n_{\mathrm{sub}}|\theta) \prod_i^{n_{\mathrm{sub}}} p(z_{\mathrm{sub}\,i}|\theta) \qquad p(x|z_{\mathrm{source}}, z_{\mathrm{lens}}, \{z_{\mathrm{sub}\,i}\})$$

Prediction (simulation)

Modeling strong lensing

Model parameters

Subhalo mass function parameters

θ

Latent variables

Source / host halo properties

$z_{\rm source}$ $z_{\rm lens}$

Subhalo number, masses, positions

$n_{\rm sub}$ $\{z_{{\rm sub}\,i}\}$

Observables

Observed lens image

x

The key quantity for inference is the marginal likelihood

$$p(x|\theta) = \int \mathrm{d}z_{\rm source} \int \mathrm{d}z_{\rm lens} \sum_{n_{\rm sub}} \int \mathrm{d}^{n_{\rm sub}} z_{\rm sub} \quad p(z_{\rm source}) \quad p(z_{\rm lens}) \quad p(n_{\rm sub}|\theta) \prod_i^{n_{\rm sub}} p(z_{{\rm sub}\,i}|\theta) \quad p(x|z_{\rm source}, z_{\rm lens}, \{z_{{\rm sub}\,i}\})$$

Inference

Modeling strong lensing

Model parameters

Subhalo mass function parameters

θ

Latent variables

Source / host halo properties

$z_{\rm source}$ $z_{\rm lens}$

Subhalo number, masses, positions

$n_{\rm sub}$ $\{z_{{\rm sub}\,i}\}$

Observables

Observed lens image

x

The key quantity for inference is the marginal likelihood

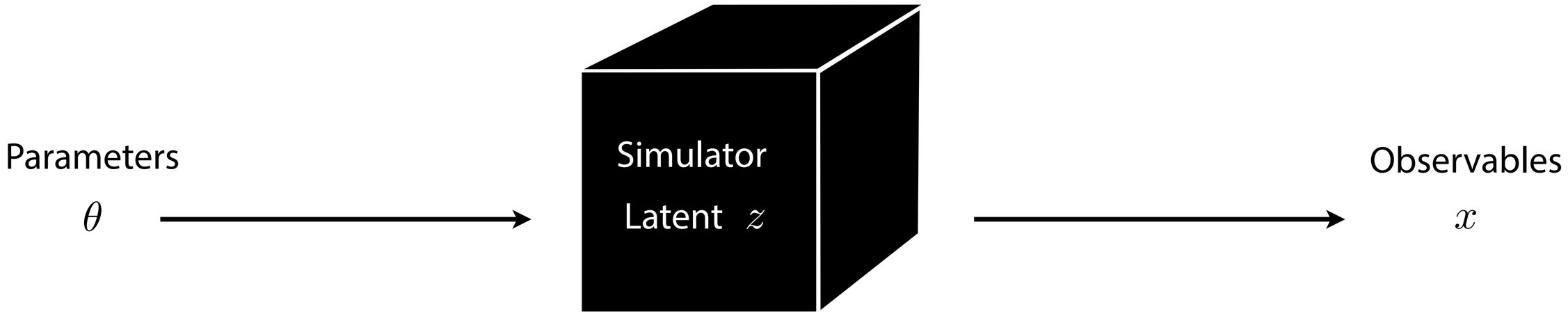
$$p(x|\theta) = \boxed{\int \mathrm{d}z_{\rm source} \int \mathrm{d}z_{\rm lens} \sum_{n_{\rm sub}} \int \mathrm{d}^{n_{\rm sub}} z_{\rm sub}} \quad p(z_{\rm source}) \quad p(z_{\rm lens}) \quad p(n_{\rm sub}|\theta) \prod_i^{n_{\rm sub}} p(z_{{\rm sub}\,i}|\theta) \qquad p(x|z_{\rm source}, z_{\rm lens}, \{z_{{\rm sub}\,i}\})$$

Unfortunately, it is intractable:
we cannot compute the integral over this huge latent space. This is a major challenge for inference.

Similarly, MCMC based on $p(x, z|\theta)$ is extremely inefficient.
(More subtle: We cannot sample from $p(z|x, \theta)$ efficiently.)

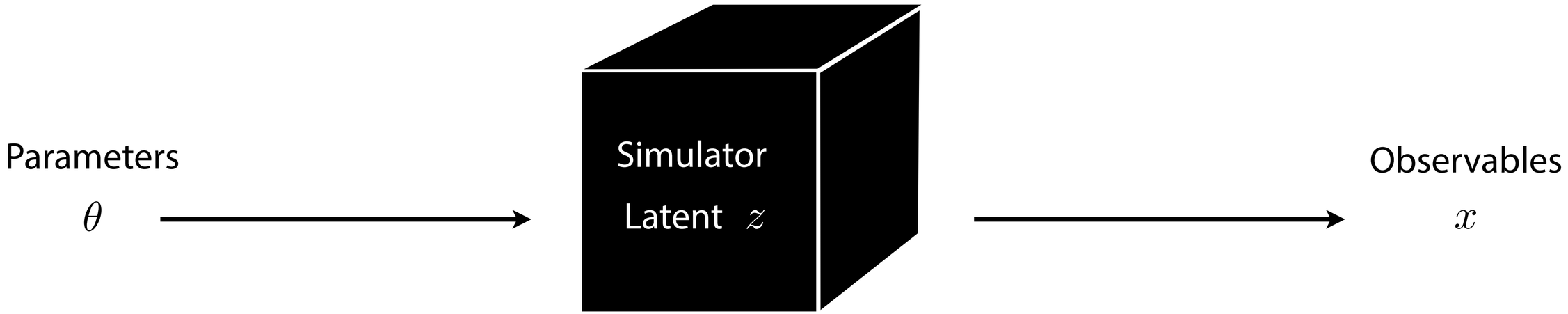
Inference

Simulation-based (“likelihood-free”) inference



Prediction:
- Well-understood mechanistic model
- Simulator can generate samples $x \sim p(x|\theta)$

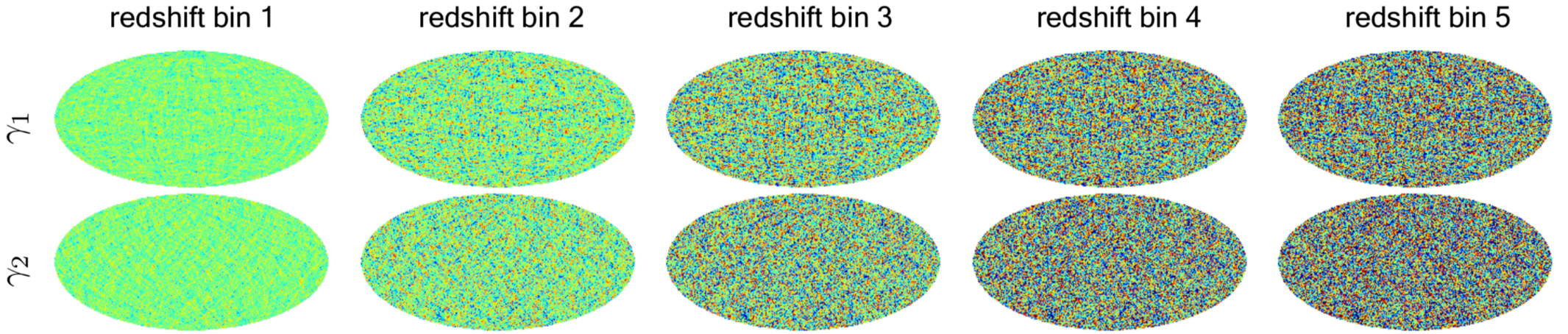
Simulation-based ("likelihood-free") inference



Prediction:
- Well-understood mechanistic model
- Simulator can generate samples $x \sim p(x|\theta)$

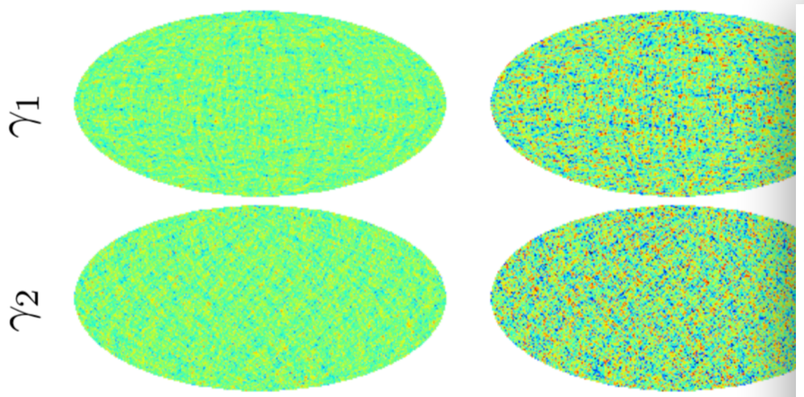
Inference:
- Likelihood $p(x|\theta) = \int \mathrm{d}z\, p(x, z|\theta)$ is intractable
- Inference is challenging

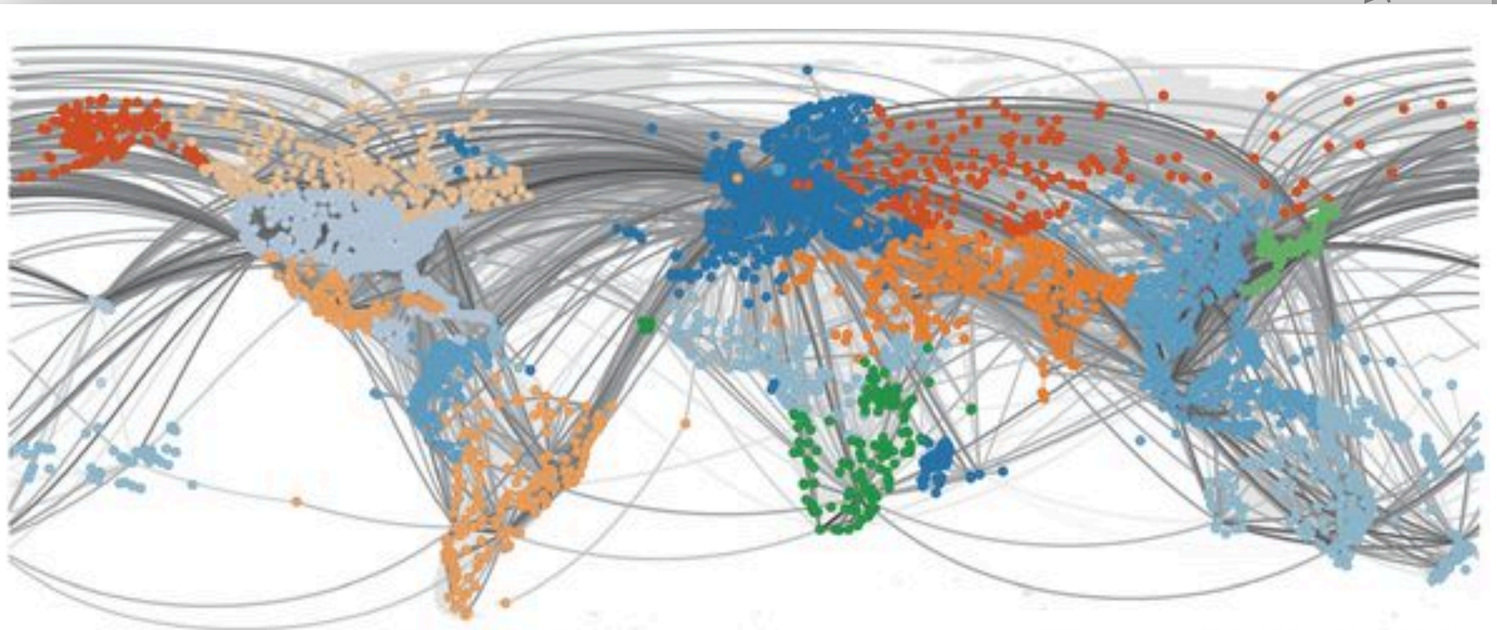
Simulation-based inference appears all over science



[J. Alsing, T. Charnock, S. Feeney, B. Wandeelt 1903.00007]

Simulation-based inference appears all over science

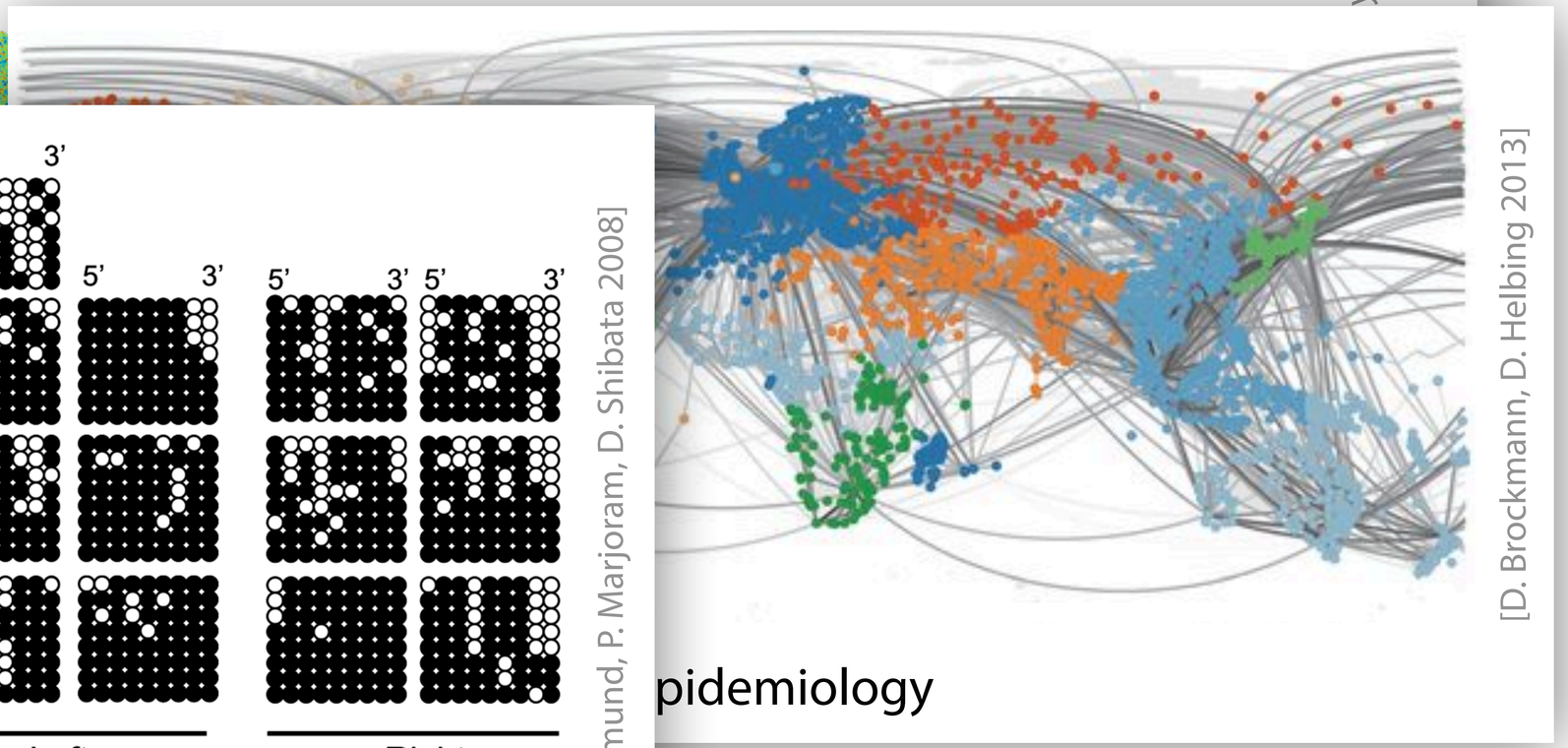




Epidemiology

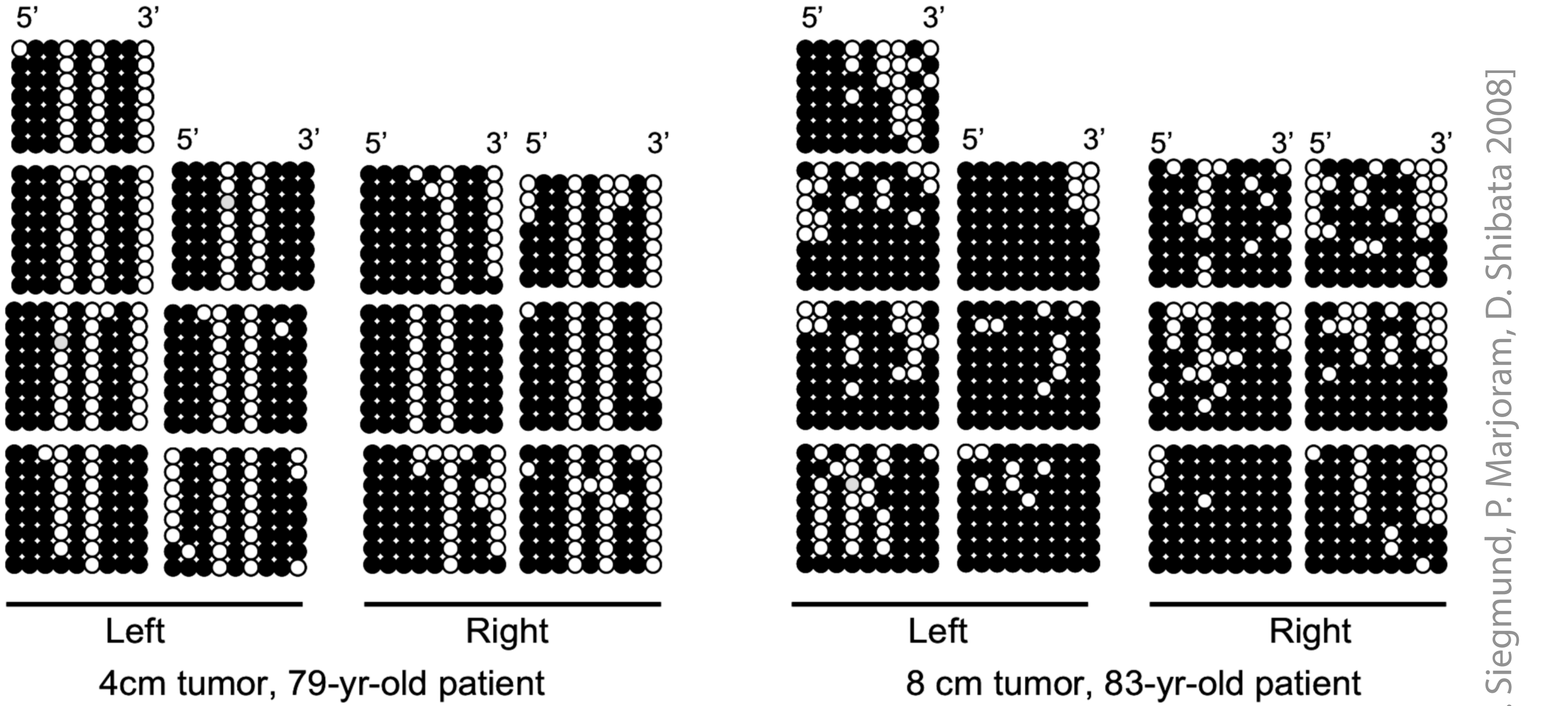
[D. Brockmann, D. Helbing 2013]

Simulation-based inference appears all over science



[D. Brockmann, D. Helbing 2013]

pidemiology

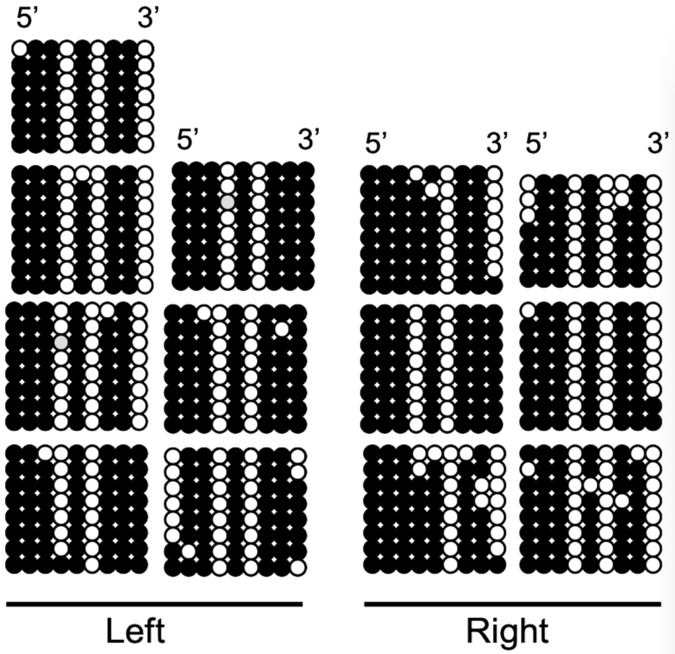


[K. Siegmund, P. Marjoram, D. Shibata 2008]

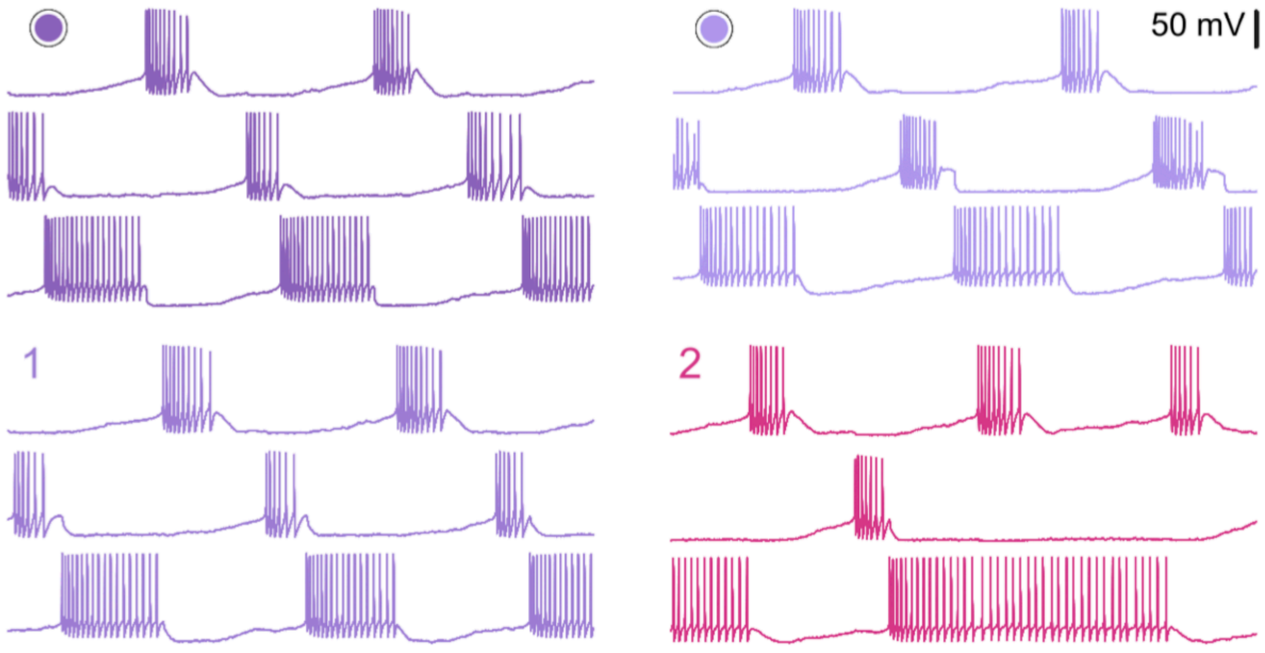
Genetics

Simulation-based inference appears all over science

[D. Brockmann, D. Helbing 2013]



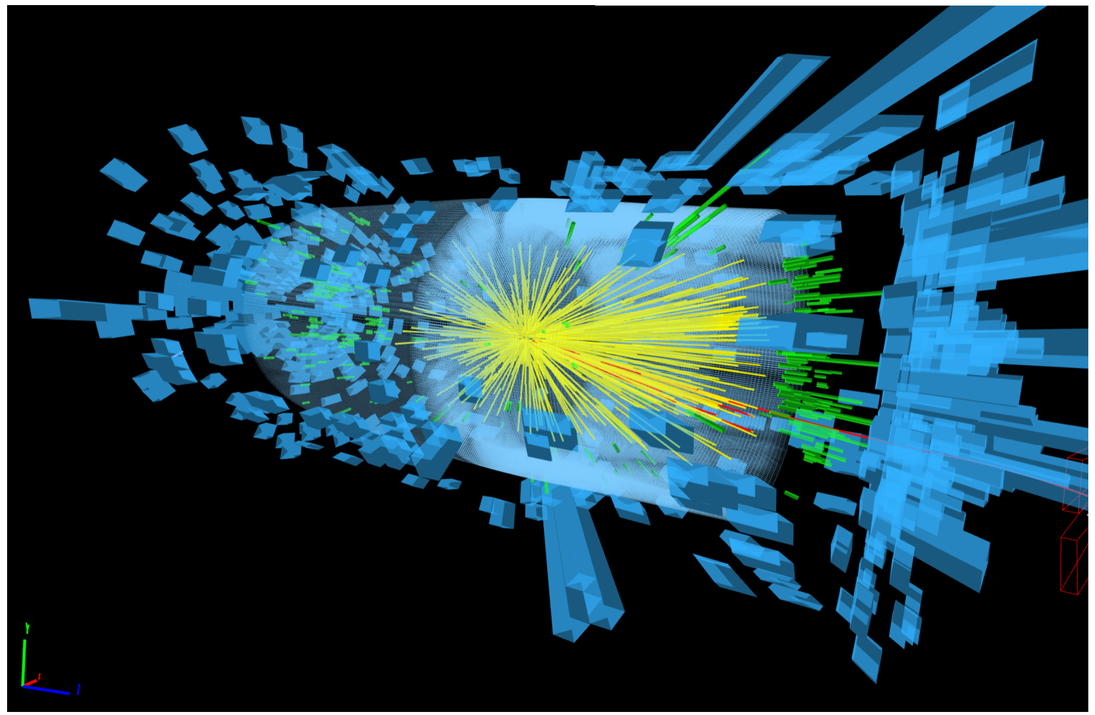
Ge



Neuroscience

[Pedro J. Goncalves et al. biorxiv:10.1101/838383]

Simulation-based inference appears all over science



[CMS]

Particle physics

redshift bin 4

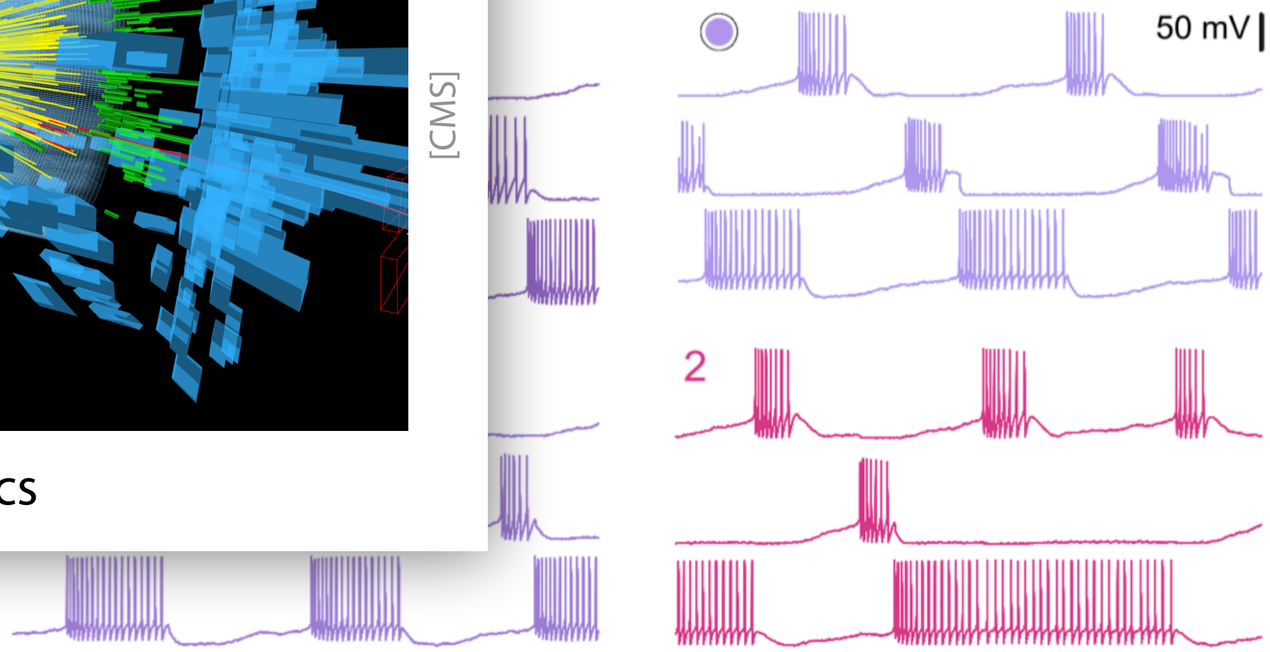
redshift bin 5

[D. Brockmann, D. Helbing 2013]

5’

4cm tumor, 79-yr-old patient

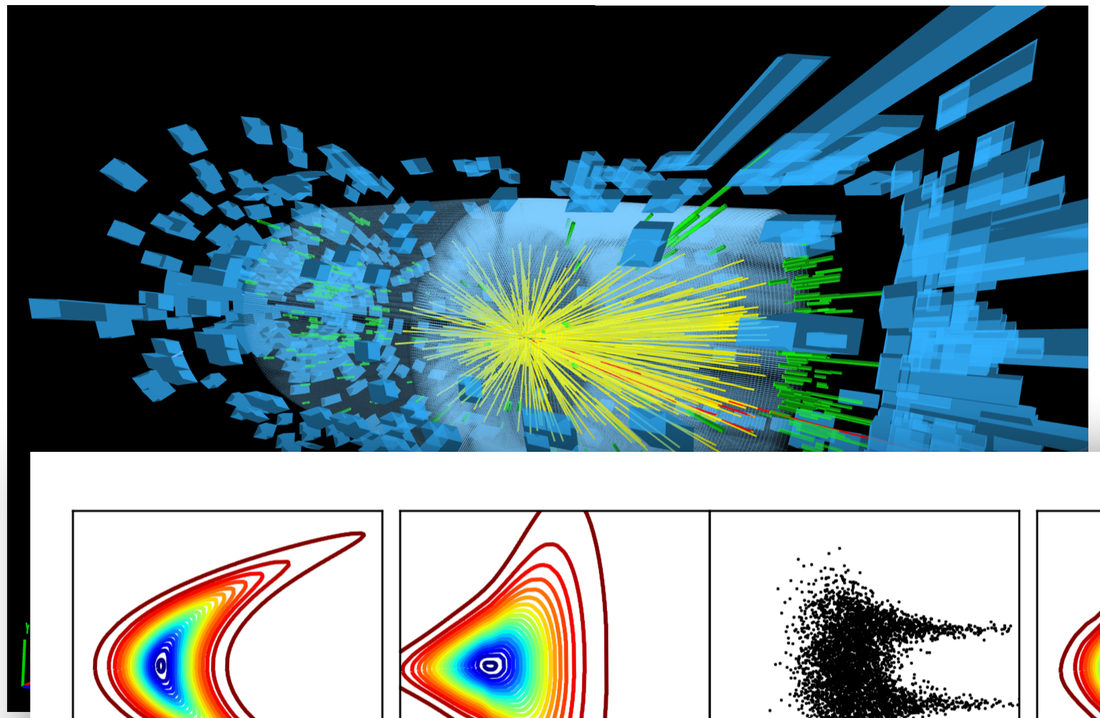
Ger



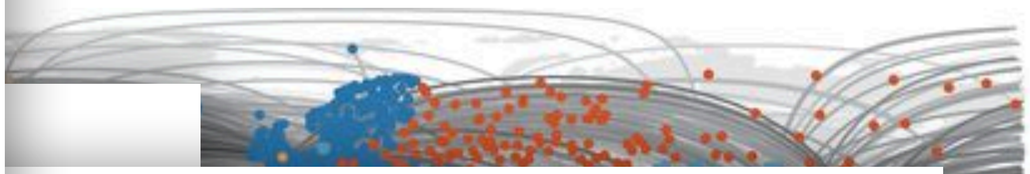
[Pedro J. Goncalves et al. biorxiv:10.1101/838383]

Neuroscience

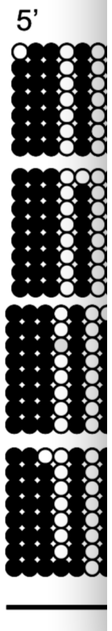
Simulation-based inference appears all over science

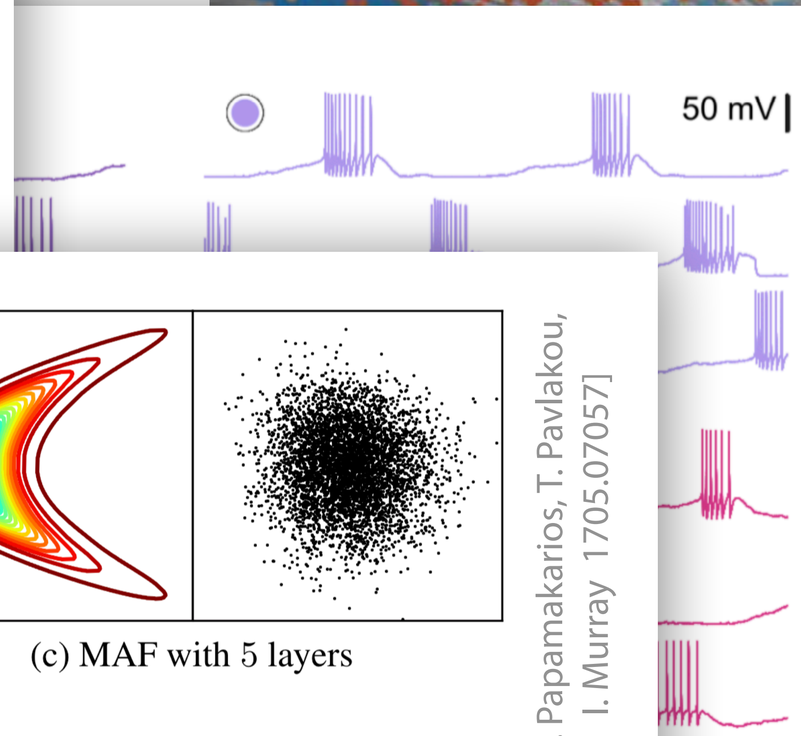


[CMS]

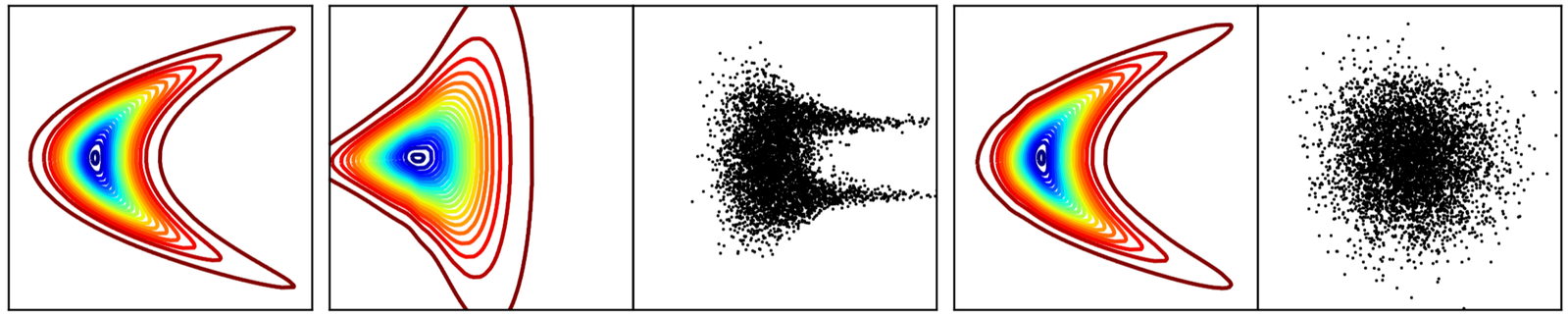


[D. Brockmann, D. Helbing 2013]





[Pedro J. Goncalves et al. biorxiv:10.1101/838383]



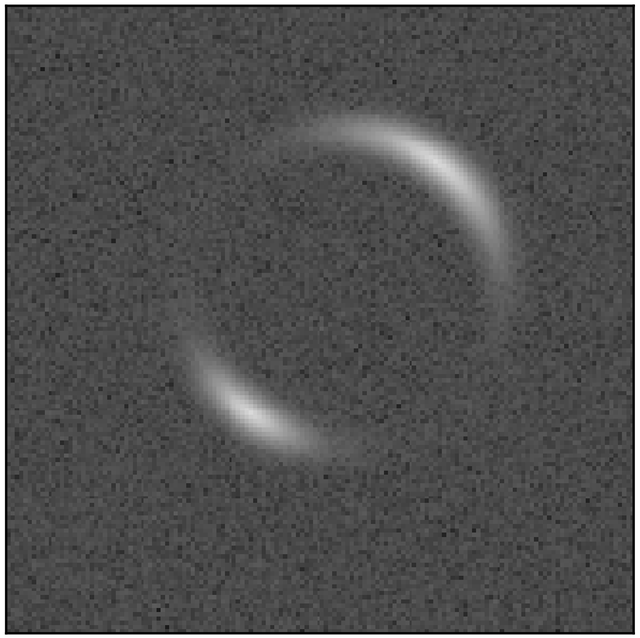
(a) Target density (b) MADE with Gaussian conditionals (c) MAF with 5 layers

[G. Papamakarios, T. Pavlakou, I. Murray 1705.07057]

Computer Science / Machine Learning / Statistics

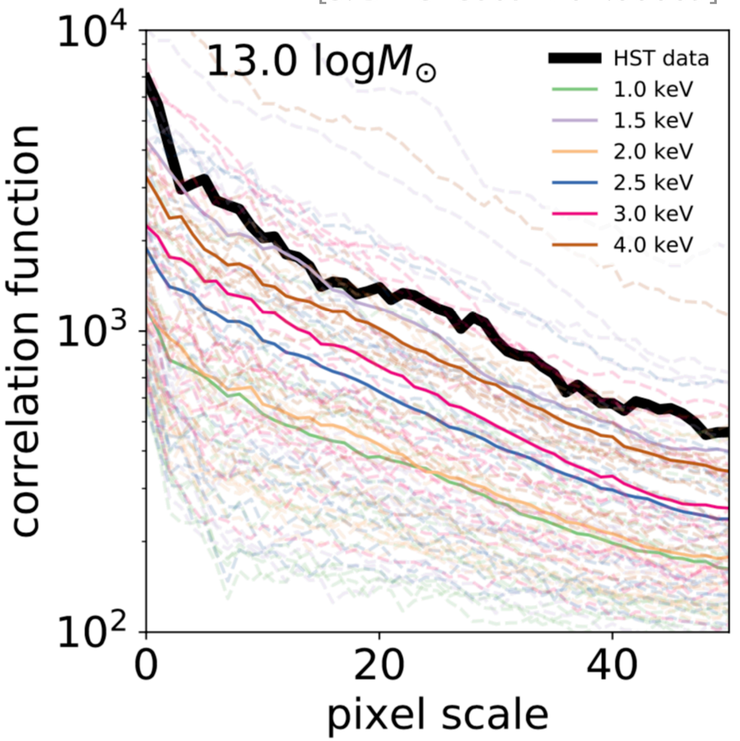
Traditionally, inference is made possible with summary statistics.

Solve it with summary statistics



High-dimensional observable data x
(pixel values)

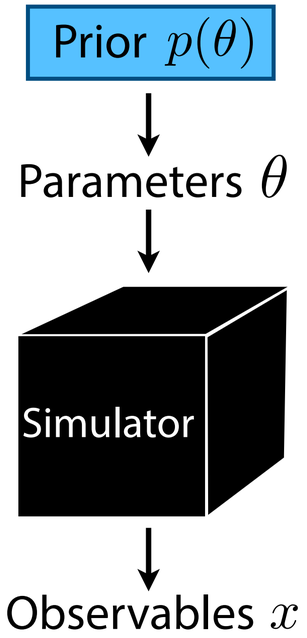
[S. Birrer et al 1702.00009]



Low-dimensional summary statistics x'
(e.g. correlation functions)

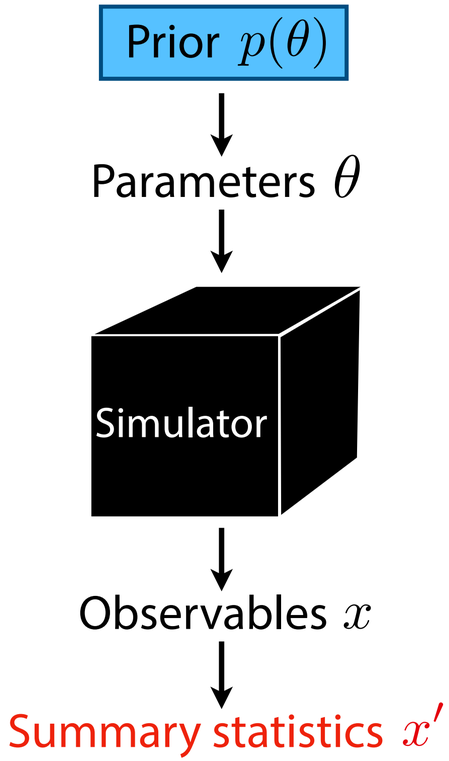
Approximate Bayesian Computation (ABC)

[D. Rubin 1984]



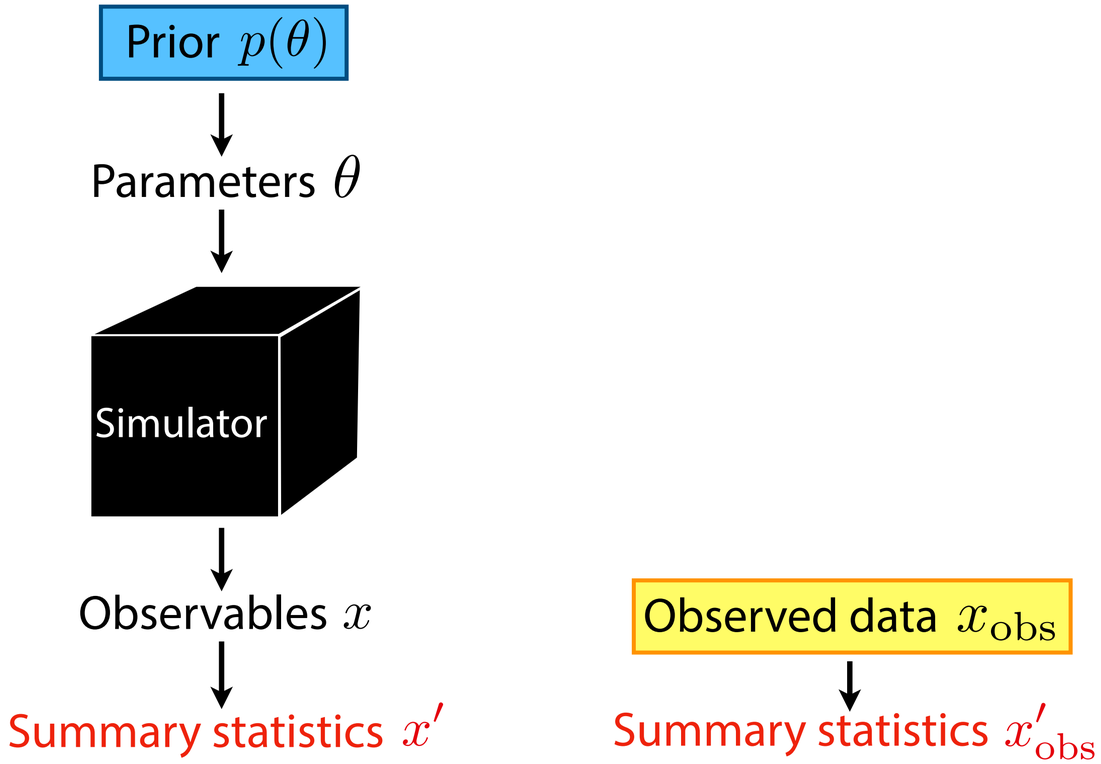
Approximate Bayesian Computation (ABC)

[D. Rubin 1984]



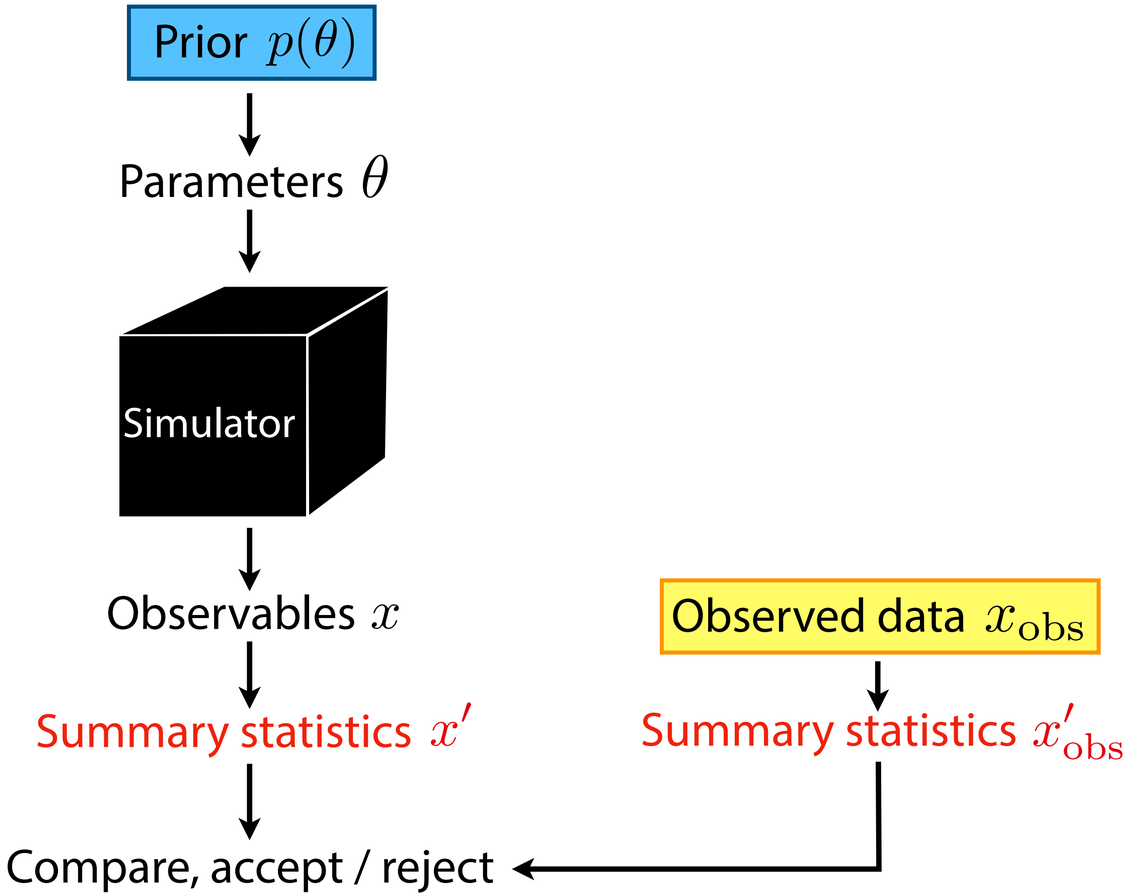
Approximate Bayesian Computation (ABC)

[D. Rubin 1984]



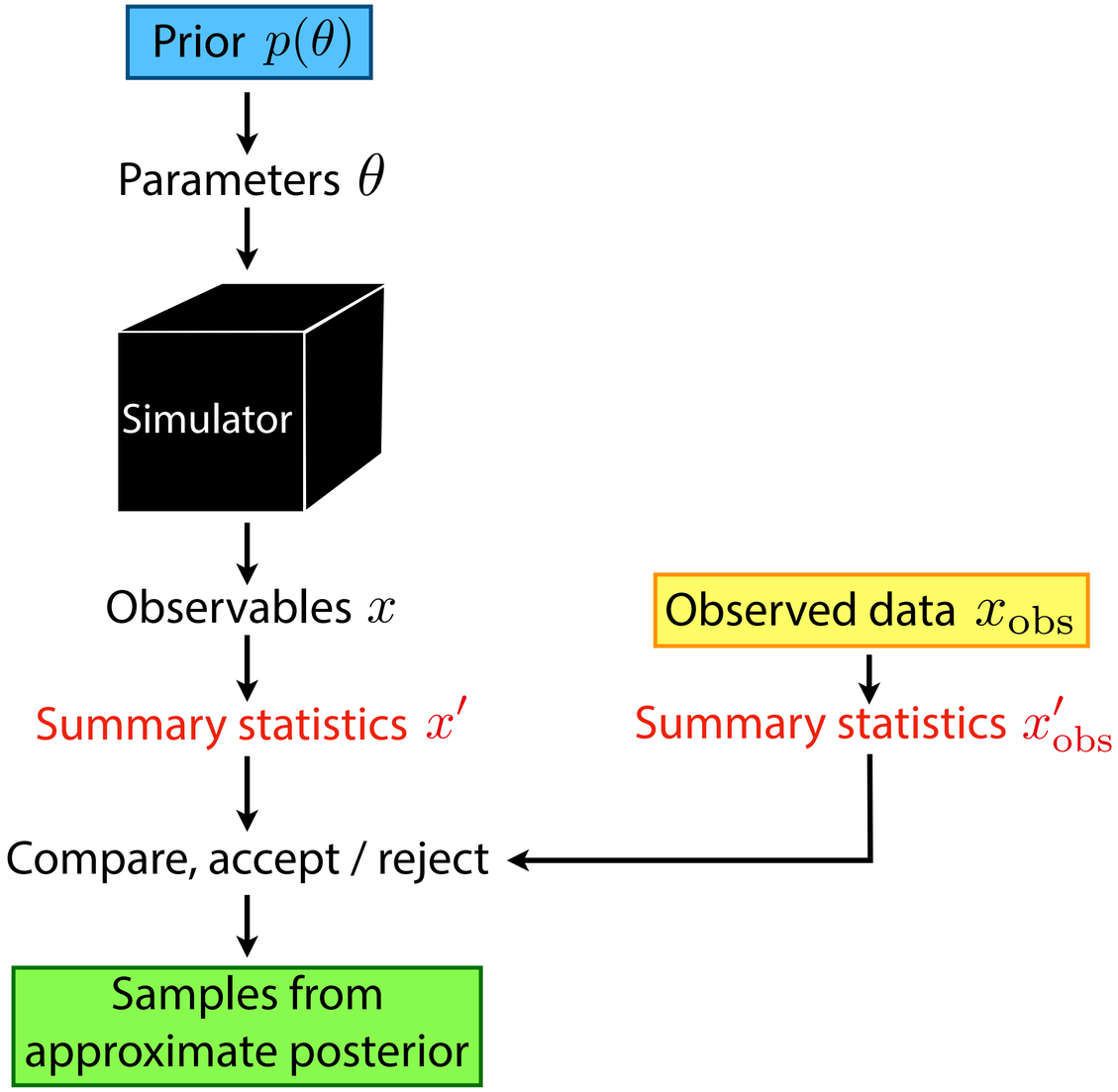
Approximate Bayesian Computation (ABC)

[D. Rubin 1984]



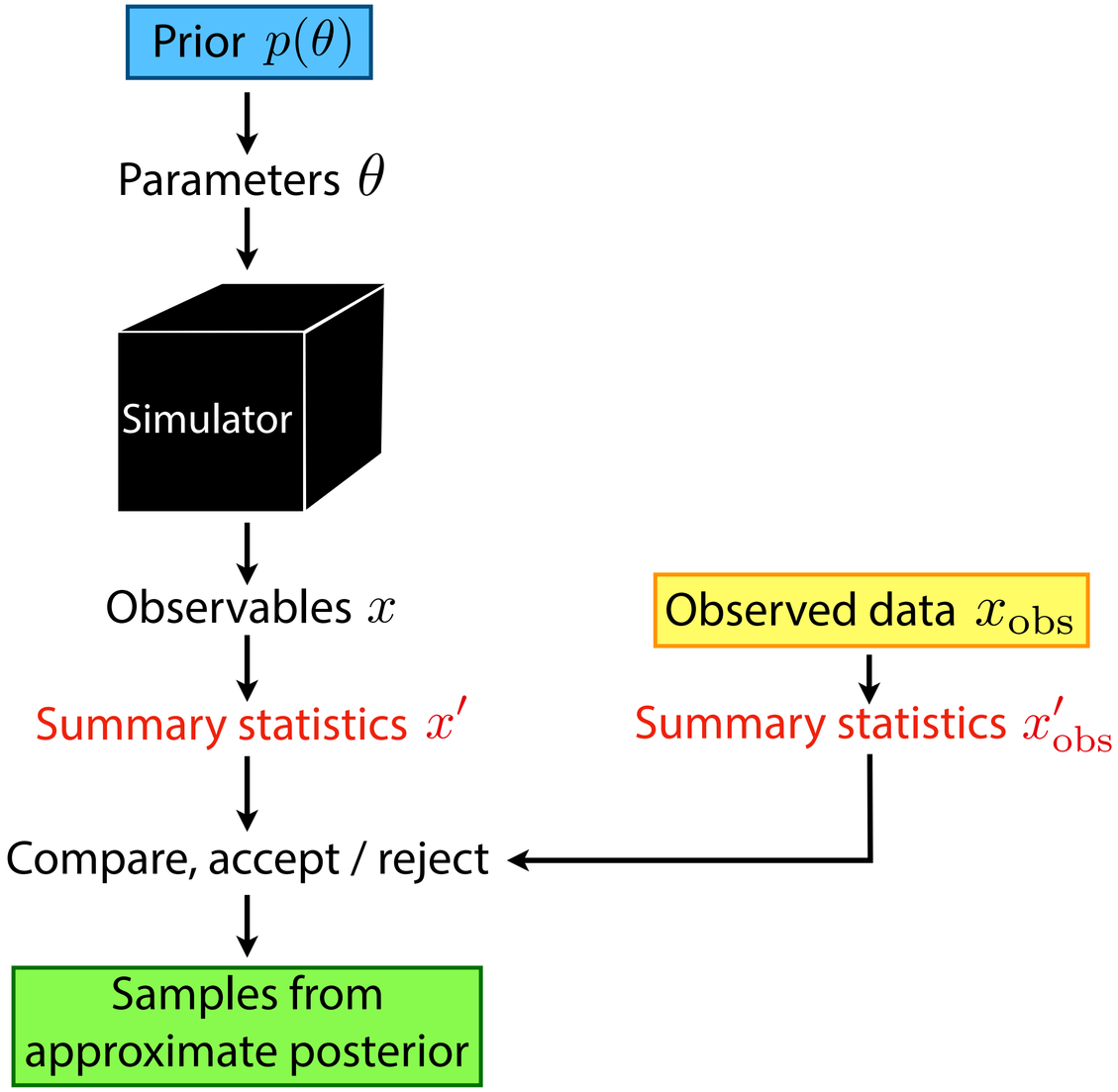
Approximate Bayesian Computation (ABC)

[D. Rubin 1984]



Approximate Bayesian Computation (ABC)

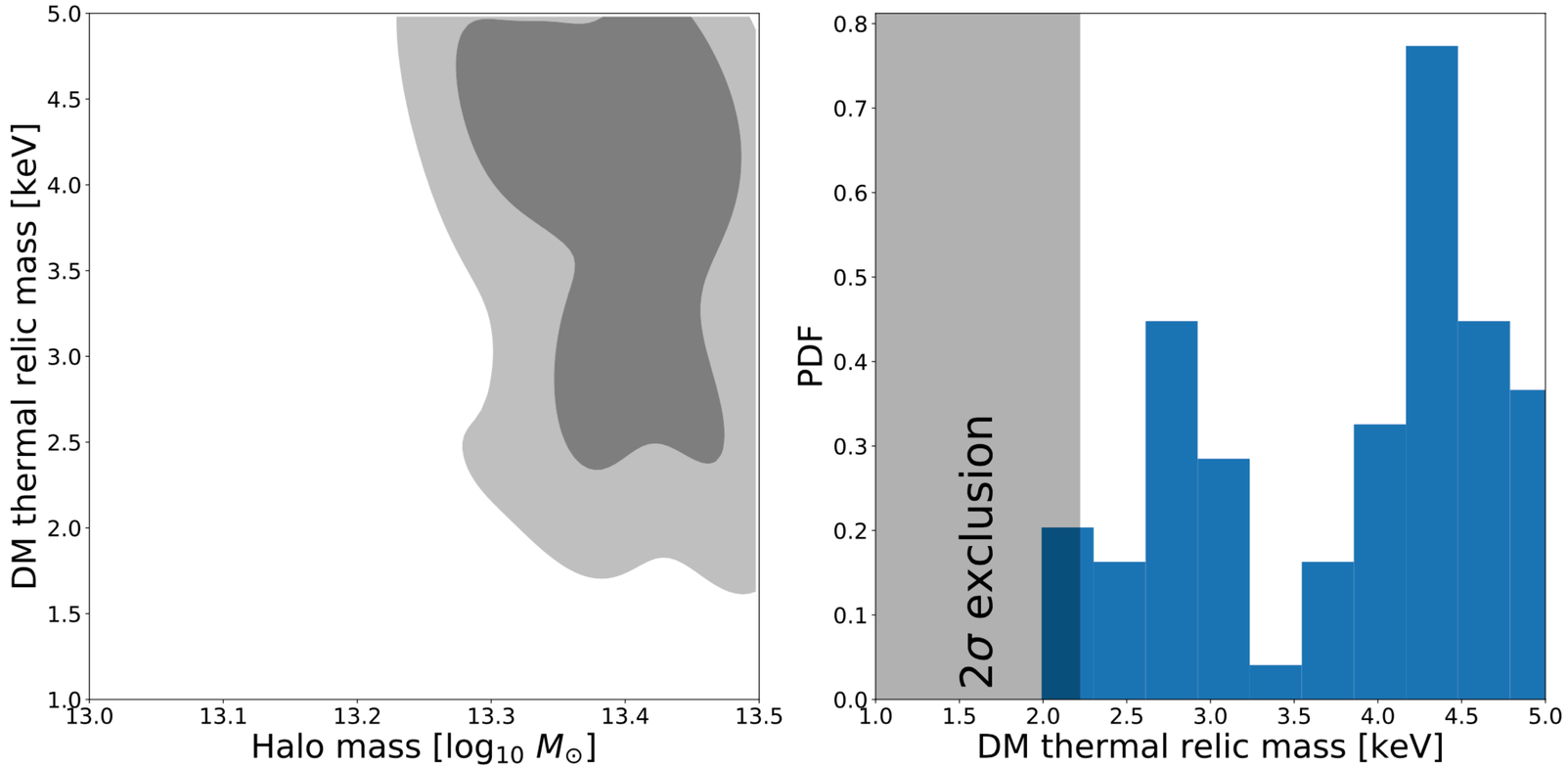
[D. Rubin 1984]



- Reduction to summary statistics makes inference possible, but throws away information
- Rejection algorithm can be very sample inefficient

Substructure ABC

[S. Birrer et al 1702.00009]



- Successful proof of principle application on HST data (RXJ1131-1231)
- Summary statistics based on correlation functions: results may be overly conservative
- Poor sample efficiency of ABC rejection: “posterior not converged”

Now, we can also use machine learning
to enable inference.

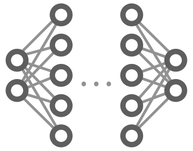
[K. Cranmer, JB, G. Louppe 1911.01429]

The likelihood ratio trick

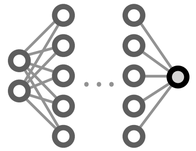
- Generative Adversarial Networks (GANs):

[I. Goodfellow et al. 1406.2661]

Generator
(intractable likelihood $p_{\text{gen}}(x)$)



Examples
(intractable likelihood $p_{\text{true}}(x)$)



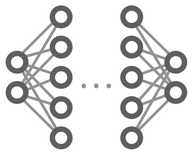
Discriminator

The likelihood ratio trick

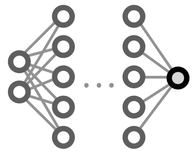
- Generative Adversarial Networks (GANs):

[I. Goodfellow et al. 1406.2661]

Generator
(intractable likelihood $p_{\text{gen}}(x)$)



Examples
(intractable likelihood $p_{\text{true}}(x)$)



Discriminator

Discriminator learns decision function

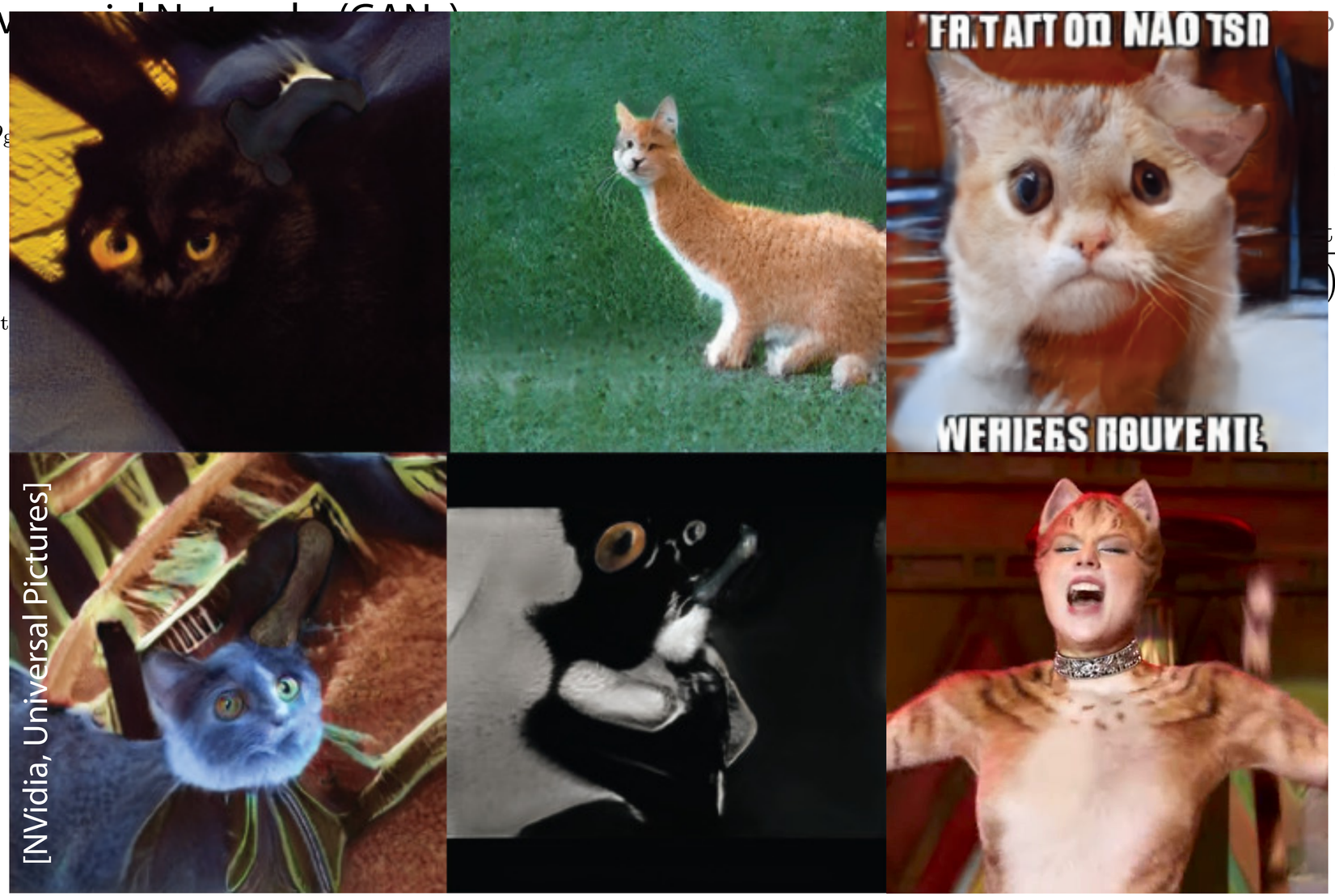
$$s(x) \rightarrow \frac{p_{\text{true}}(x)}{p_{\text{gen}}(x) + p_{\text{true}}(x)}$$

The likelihood ratio trick

- Generative Adv[illegible]Networks (GANs) [Goodfellow et al. 1406.2661]

Generator
(intractable likelihood p_g

Examples
(intractable likelihood p_t

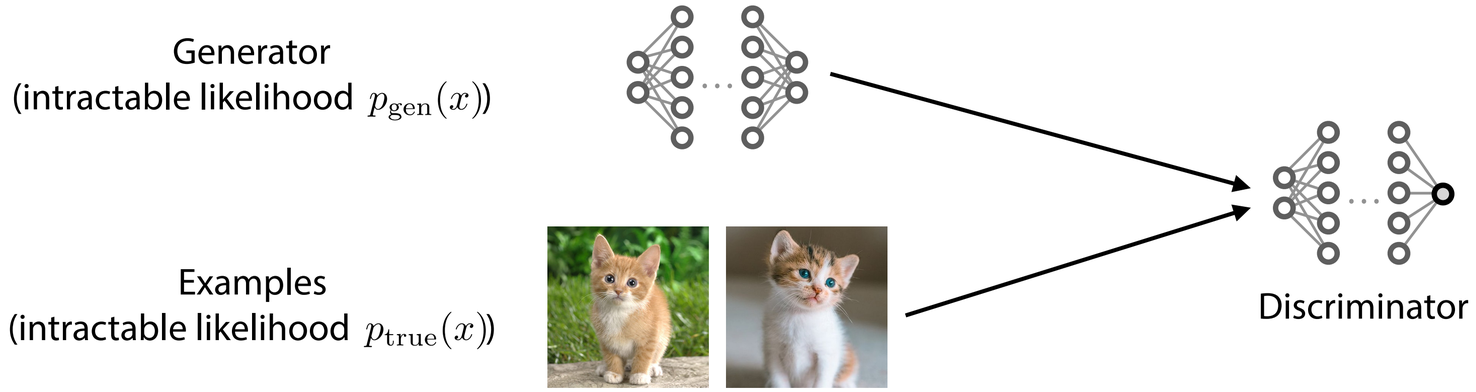


decision function

$$\frac{\text{true}(x)}{) + p_{\text{true}}(x)}$$

The likelihood ratio trick

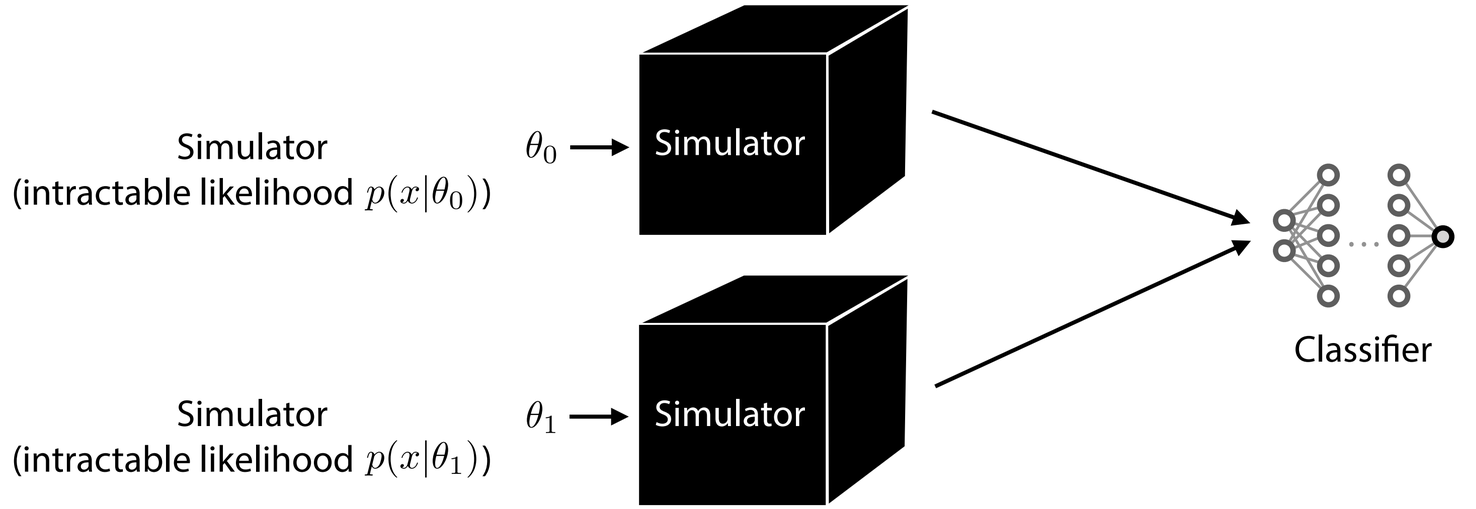
- Generative Adversarial Networks (GANs): [I. Goodfellow et al. 1406.2661]



Discriminator learns decision function

$$s(x) \to \frac{p_{\rm true}(x)}{p_{\rm gen}(x) + p_{\rm true}(x)}$$

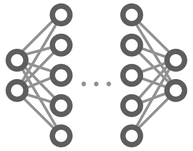
- Similarly, we can train a classifier between two sets of simulated samples [K. Cranmer, J. Pavez, G. Louppe 1506.02169]



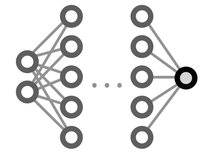
The likelihood ratio trick

- Generative Adversarial Networks (GANs): [I. Goodfellow et al. 1406.2661]

Generator
(intractable likelihood $p_{\rm gen}(x)$)



Examples
(intractable likelihood $p_{\rm true}(x)$)



Discriminator

Discriminator learns decision function

$$s(x) \to \frac{p_{\rm true}(x)}{p_{\rm gen}(x) + p_{\rm true}(x)}$$

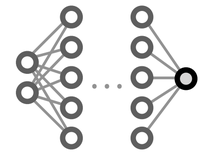
- Similarly, we can train a classifier between two sets of simulated samples [K. Cranmer, J. Pavez, G. Louppe 1506.02169]

Simulator
(intractable likelihood $p(x|\theta_0)$)

θ_0 → Simulator

Simulator
(intractable likelihood $p(x|\theta_1)$)

θ_1 → Simulator



Classifier

Classifier learns decision function

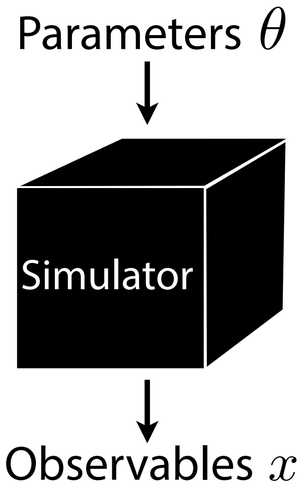
$$s(x) \to \frac{p(x|\theta_1)}{p(x|\theta_0) + p(x|\theta_1)}$$

⇒ Estimator for likelihood ratio

$$\hat{r}(x) = \frac{1 - s(x)}{s(x)} \to \frac{p(x|\theta_0)}{p(x|\theta_1)}$$

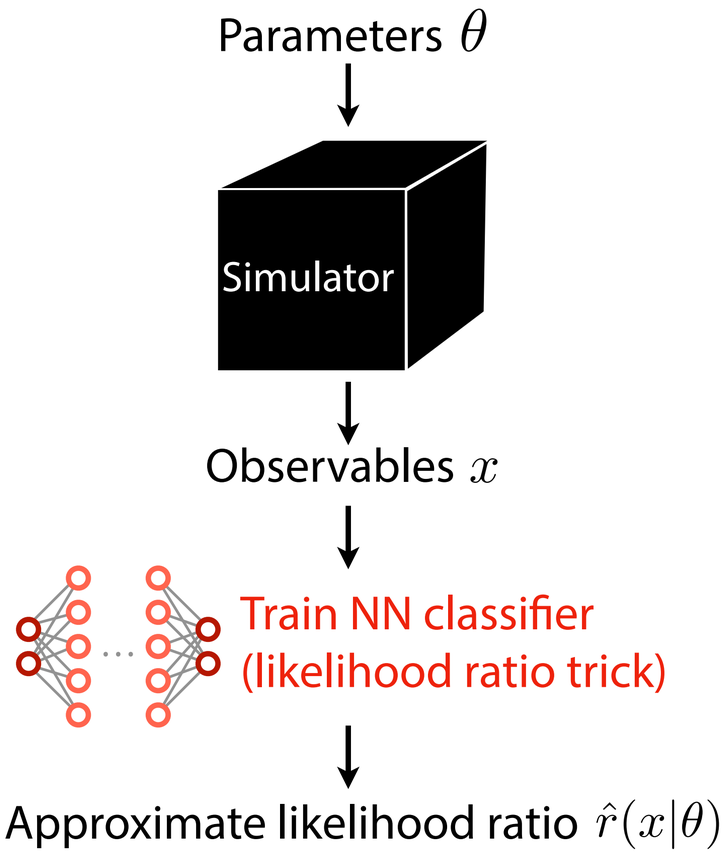
Inference by likelihood ratio trick

[K. Cranmer J. Pavez, G. Louppe 1506.02169]



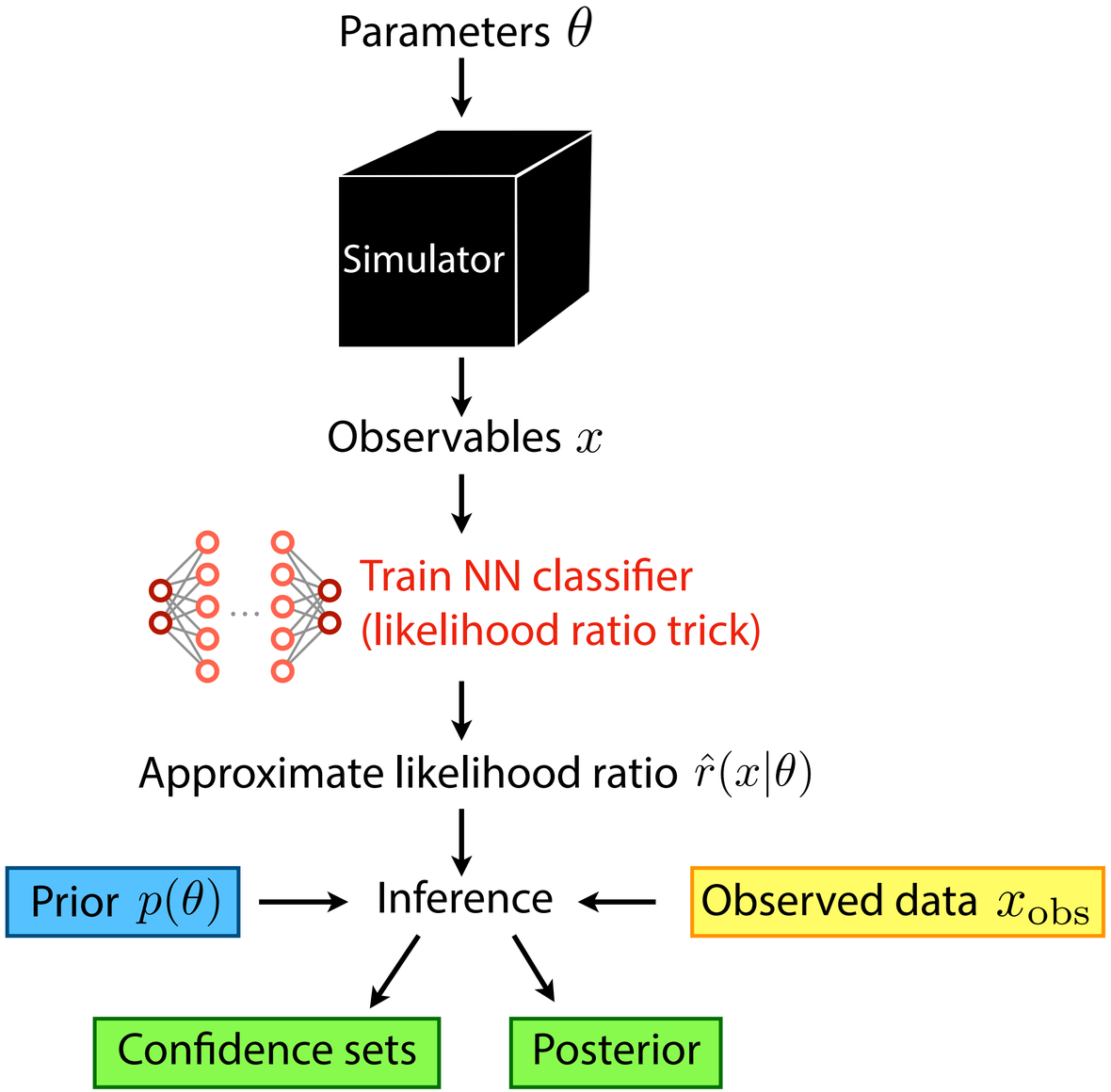
Inference by likelihood ratio trick

[K. Cranmer J. Pavez, G. Louppe 1506.02169]



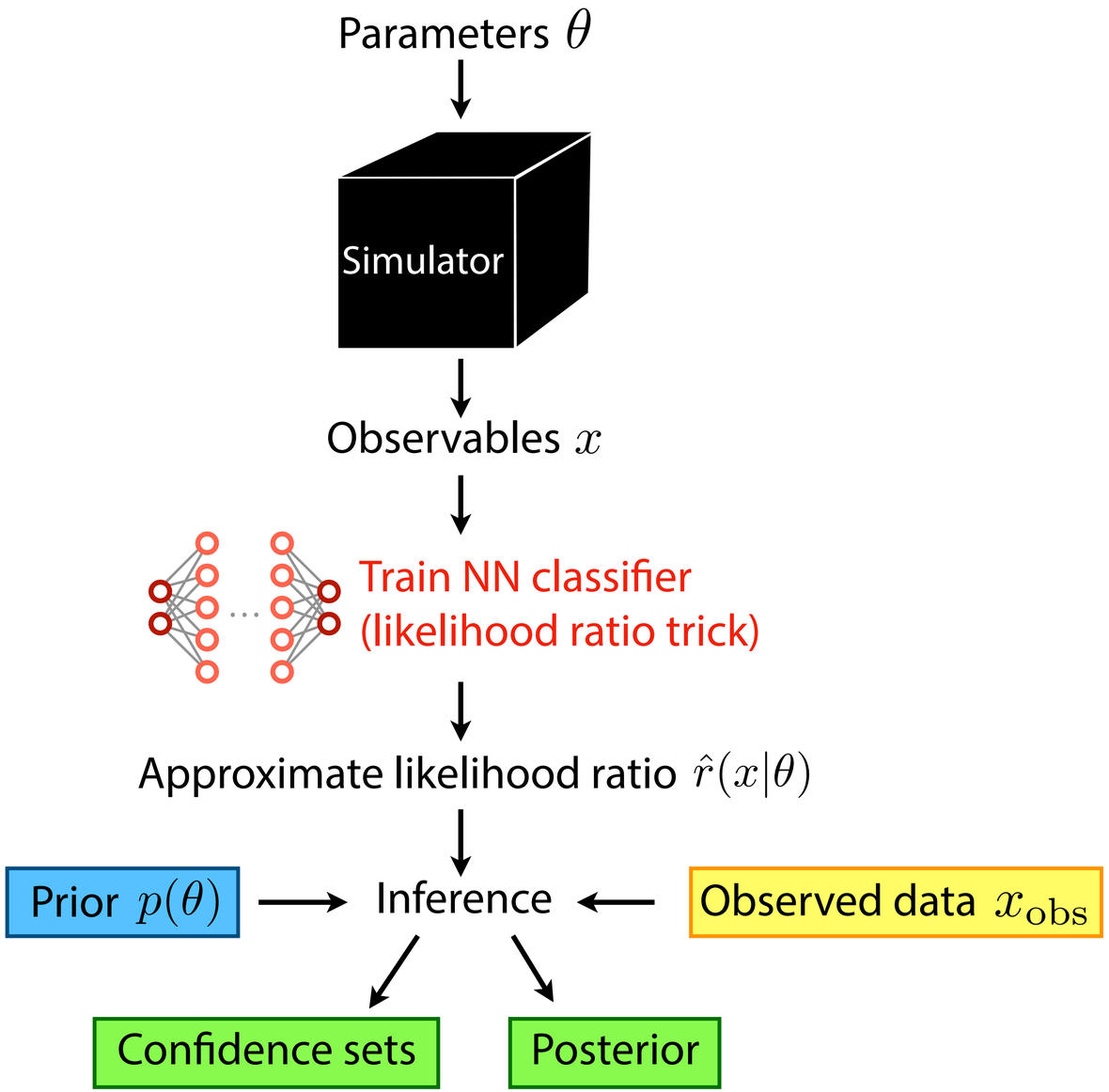
Inference by likelihood ratio trick

[K. Cranmer J. Pavez, G. Louppe 1506.02169]



Inference by likelihood ratio trick

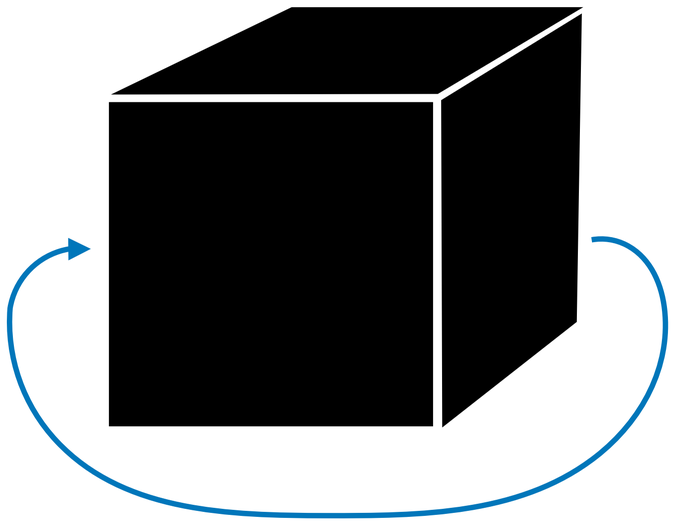
[K. Cranmer J. Pavez, G. Louppe 1506.02169]



- Instead of the intractable likelihood of the simulator, we use the likelihood ratio learned by a NN from simulated data
- Scales well to high-dimensional data
  (no compression to summary stats necessary)
- Amortized: After upfront simulation + training phase, inference is efficient for new data or prior
- Alternative approaches: train NN to learn likelihood or posterior
  [review: K. Cranmer, JB, G. Louppe 1911.01429]

Improving sample efficiency

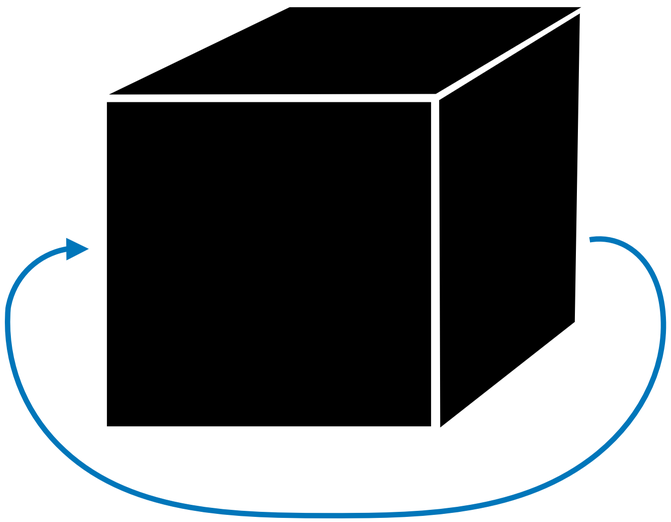
[K. Cranmer, JB, G. Louppe 1911.01429]



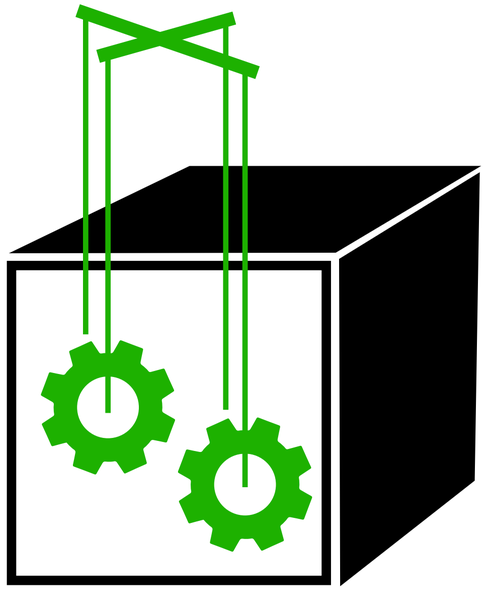
Active learning:
Iteratively guide simulator to important parameter regions based on past results

Improving sample efficiency

[K. Cranmer, JB, G. Louppe 1911.01429]



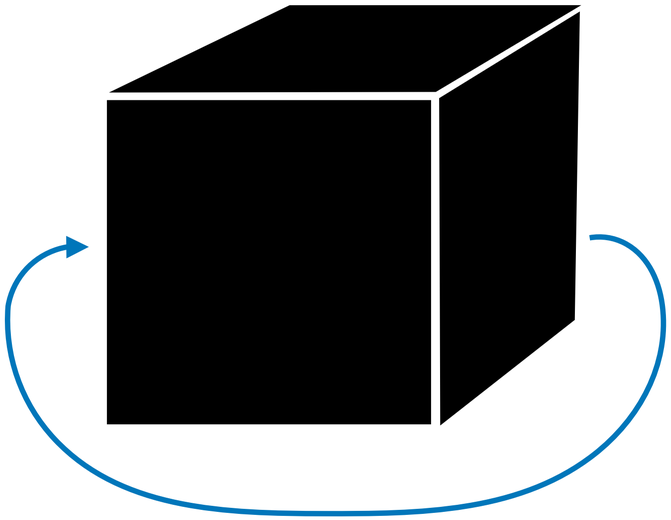
Active learning:
Iteratively guide simulator to important parameter regions based on past results



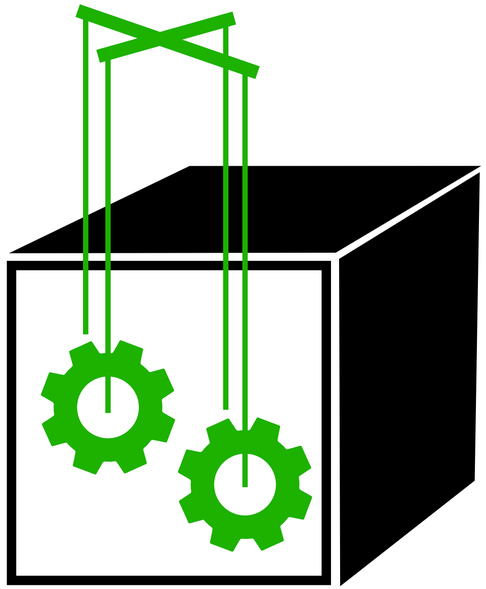
Probabilistic programming:
Explicitly write simulator as probabilistic program, with ability to condition execution trace on observations

Improving sample efficiency

[K. Cranmer, JB, G. Louppe 1911.01429]



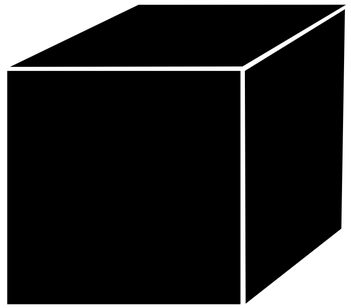
Active learning:
Iteratively guide simulator to important parameter regions based on past results



Probabilistic programming:
Explicitly write simulator as probabilistic program, with ability to condition execution trace on observations

Mining gold:
Extract and leverage information from the simulator that characterizes the latent process

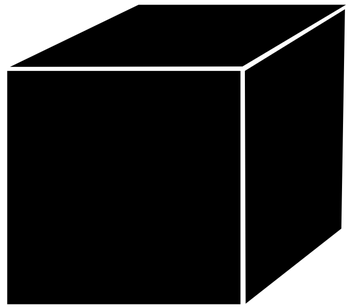
Improving robustness to mismodelling

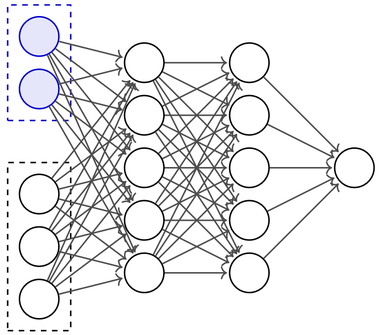


Don’t fully trust the simulator?

- Nuisance parameters to model systematic uncertainties
- Methods learn dependence both on parameters of interest and nuisance parameters
- Profile / marginalize over nuisance parameters
- Alternatively: Robustness to nuisance with adversarial training
  [G. Louppe, M. Kagan, K. Cranmer 1611.01046]

Improving robustness to mismodelling





Don’t fully trust the simulator?

- Nuisance parameters to model systematic uncertainties
- Methods learn dependence both on parameters of interest and nuisance parameters
- Profile / marginalize over nuisance parameters
- Alternatively: Robustness to nuisance with adversarial training
  [G. Louppe, M. Kagan, K. Cranmer 1611.01046]

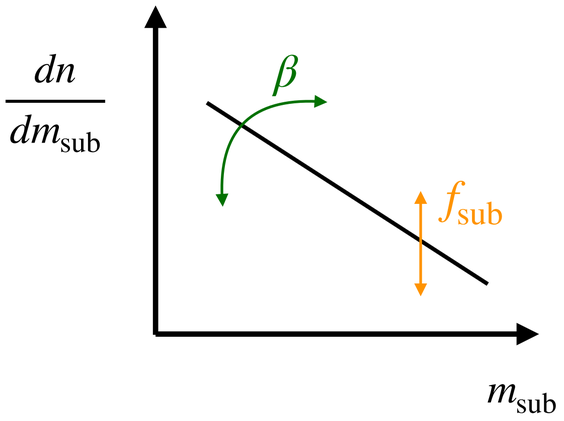
Don’t blindly trust the neural network?

- Sanity checks: expectation values, “critic” tests
- Calibration / Neyman construction with toys
  (badly trained network can lead to suboptimal limits, but not to wrong limits)

This method lets us infer Dark Matter substructure from strong lensing observations.

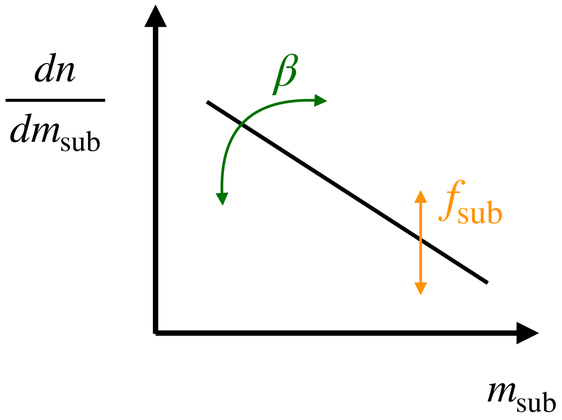
Overview

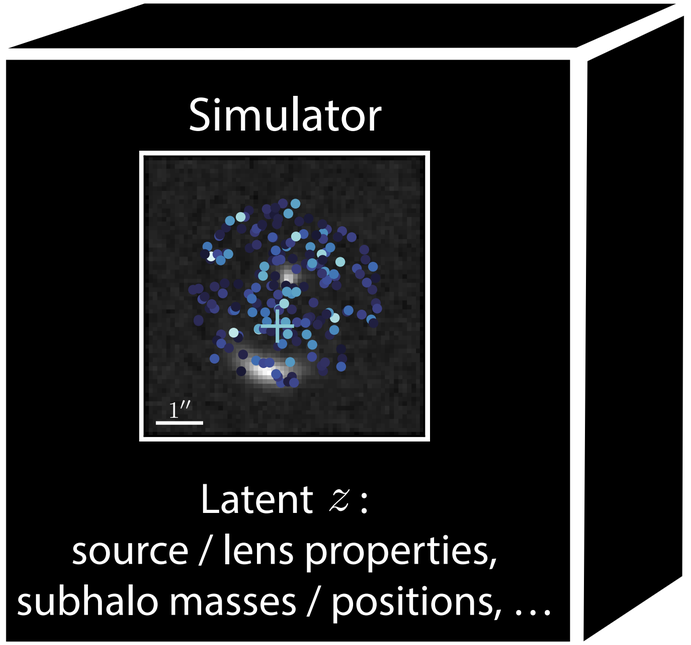
2 parameters $\theta = (\beta, f_{\rm sub})$



Overview

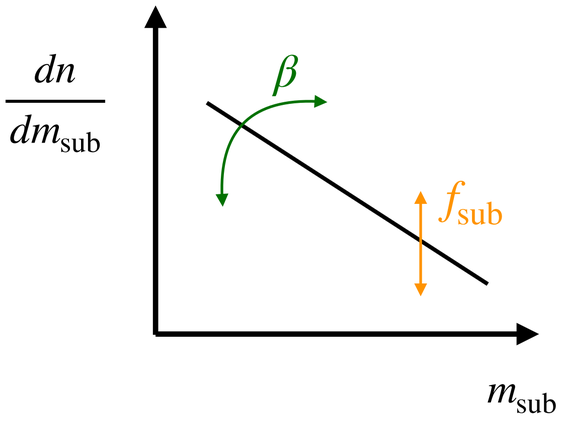
2 parameters $\theta = (\beta, f_{\rm sub})$

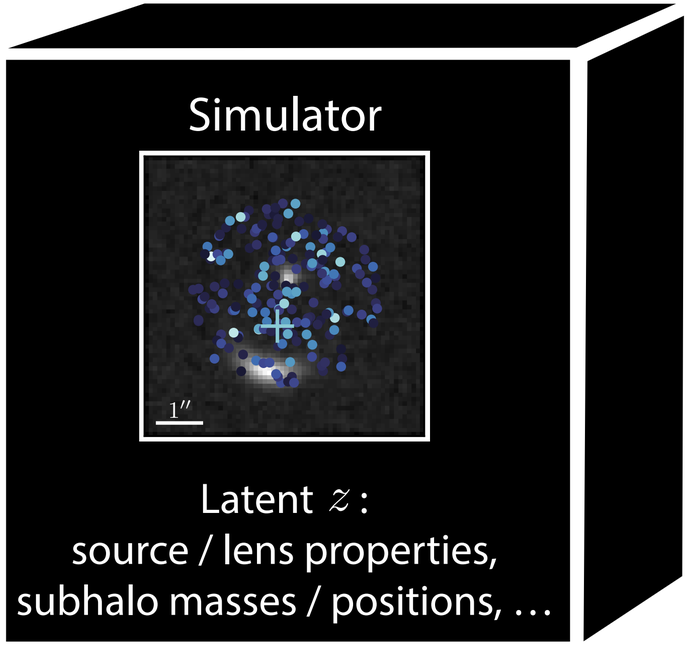




Overview

2 parameters $\theta = (\beta, f_{\rm sub})$

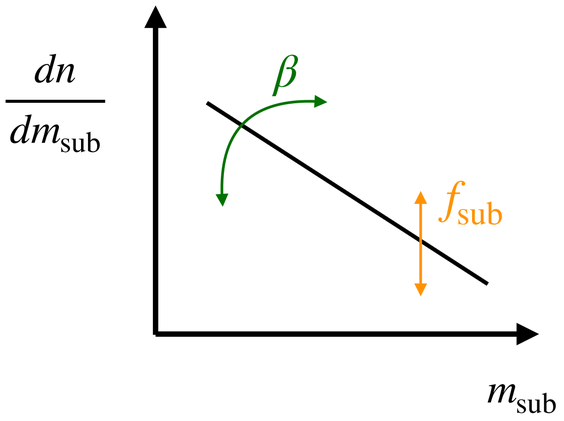


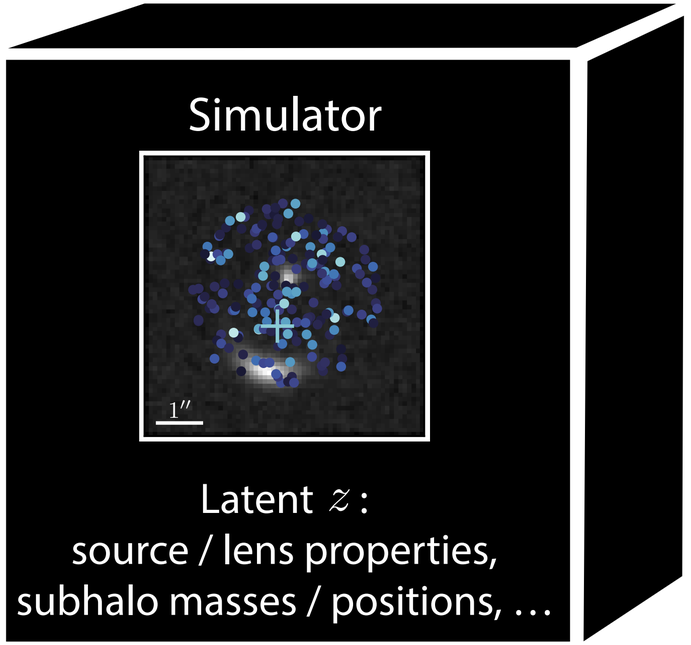


64^2 observables x

Overview

2 parameters $\theta = (\beta, f_{\rm sub})$



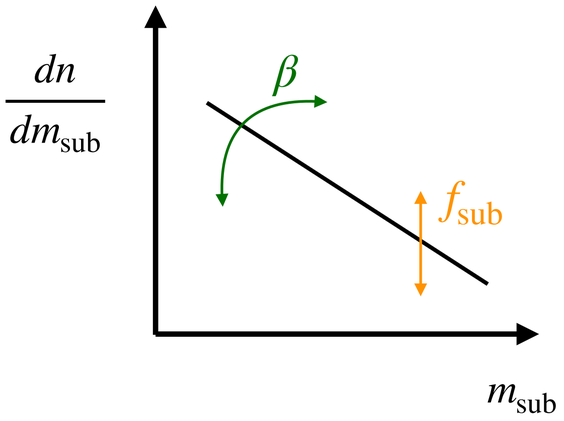


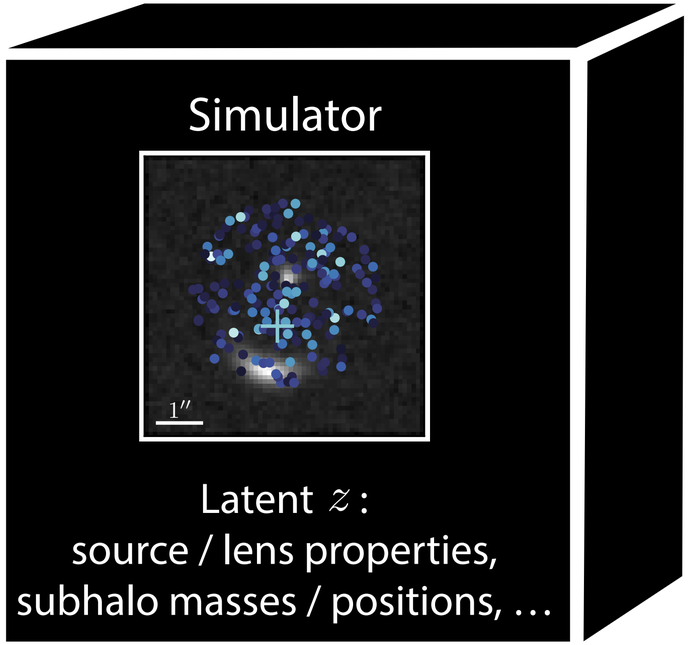
64^2 observables x

Prediction: We construct a simulator that can sample $x \sim p(x|\theta)$

Overview

2 parameters $\theta = (\beta, f_{\rm sub})$





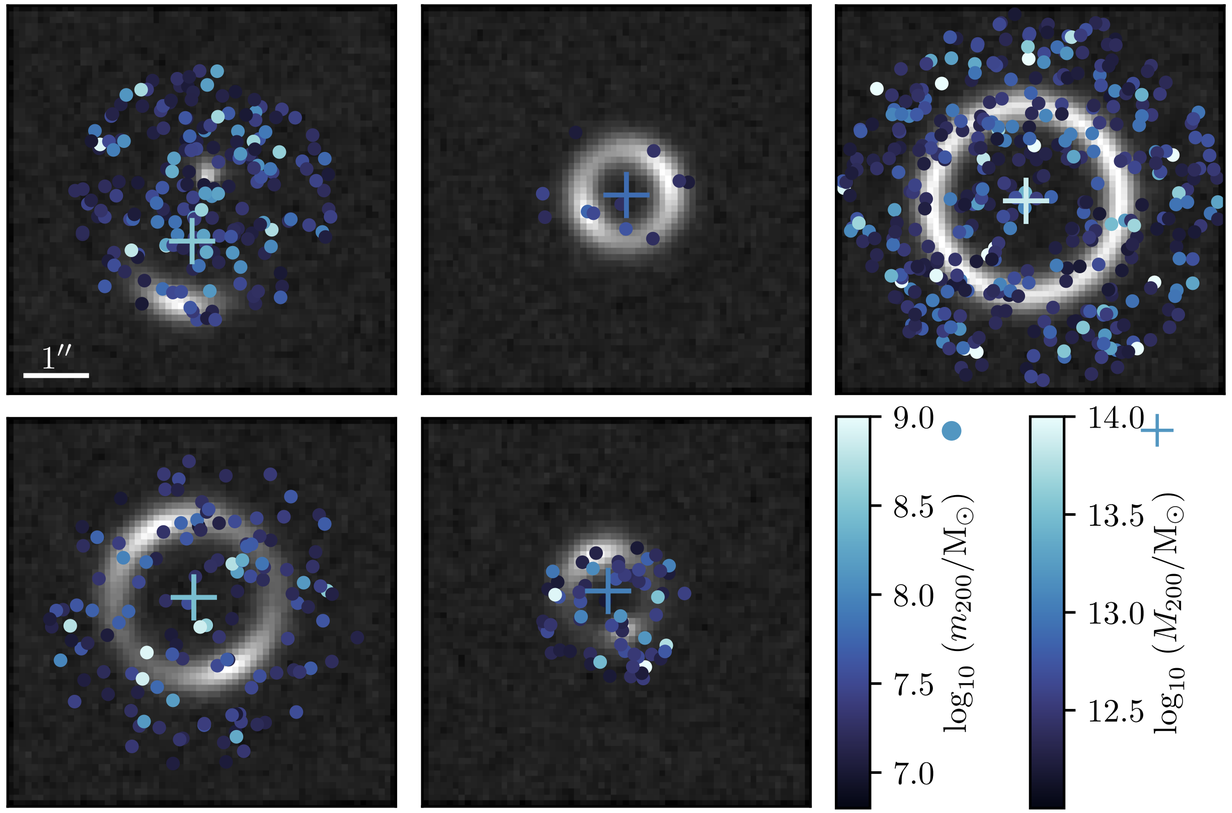
64^2 observables x

Prediction: We construct a simulator that can sample $x \sim p(x|\theta)$

Inference: We train neural likelihood ratio estimators $\hat{r}(x|\theta)$

Proof-of-principle simulator

[following T. Collett 1507.02657]

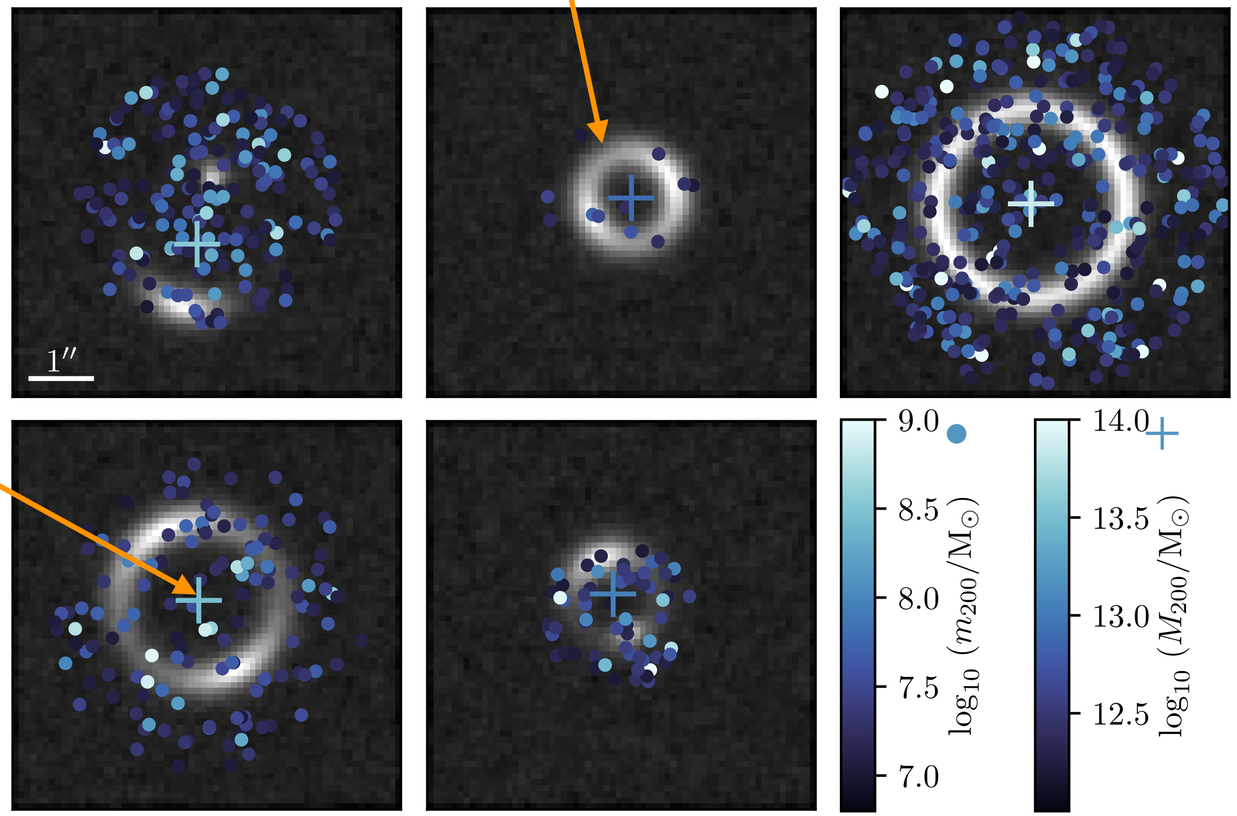


Proof-of-principle simulator

[following T. Collett 1507.02657]

Spherical lensing host galaxies
at redshift ~0.5…1

Extended galaxy sources
at redshift 1.5

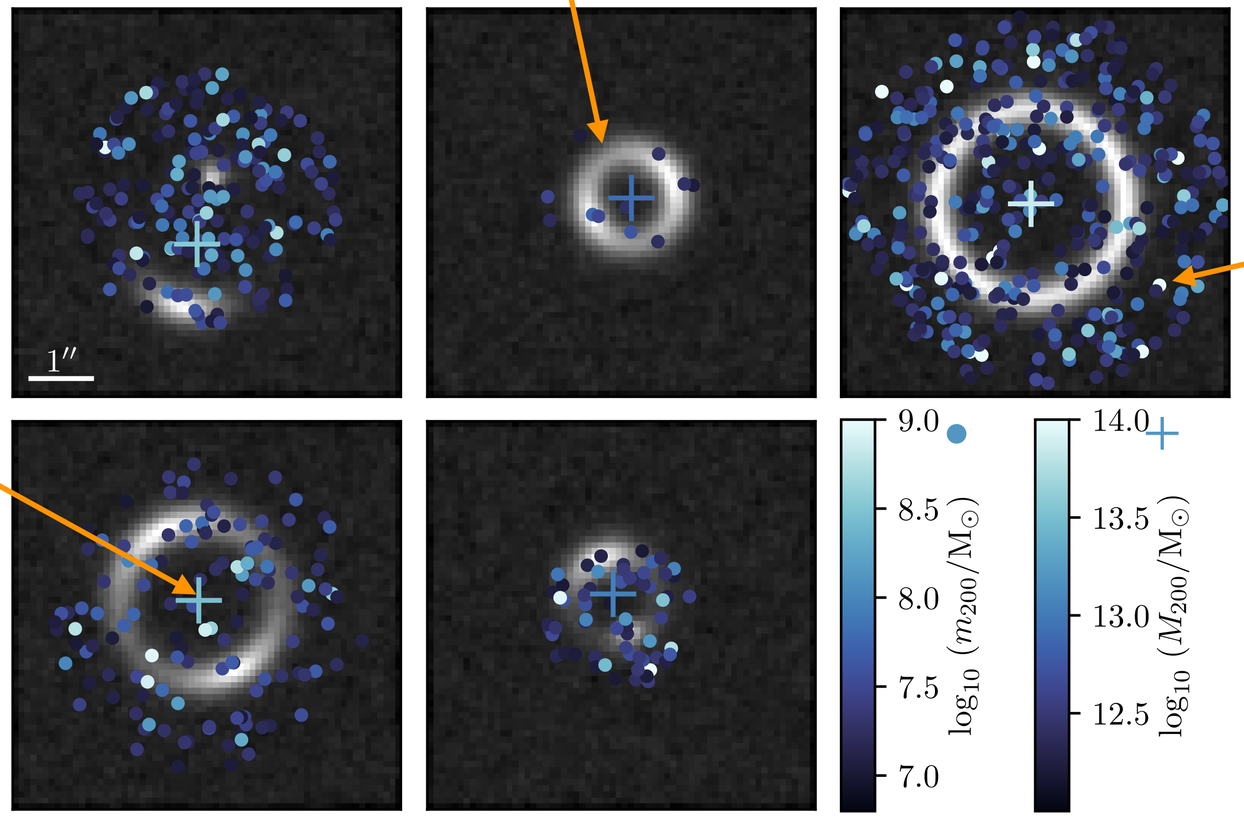


Proof-of-principle simulator

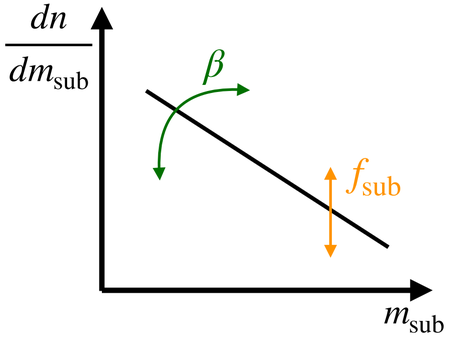
[following T. Collett 1507.02657]

Spherical lensing host galaxies
at redshift ~0.5…1

Extended galaxy sources
at redshift 1.5



Subhalos follow mass distribution with two parameters



Proof-of-principle simulator

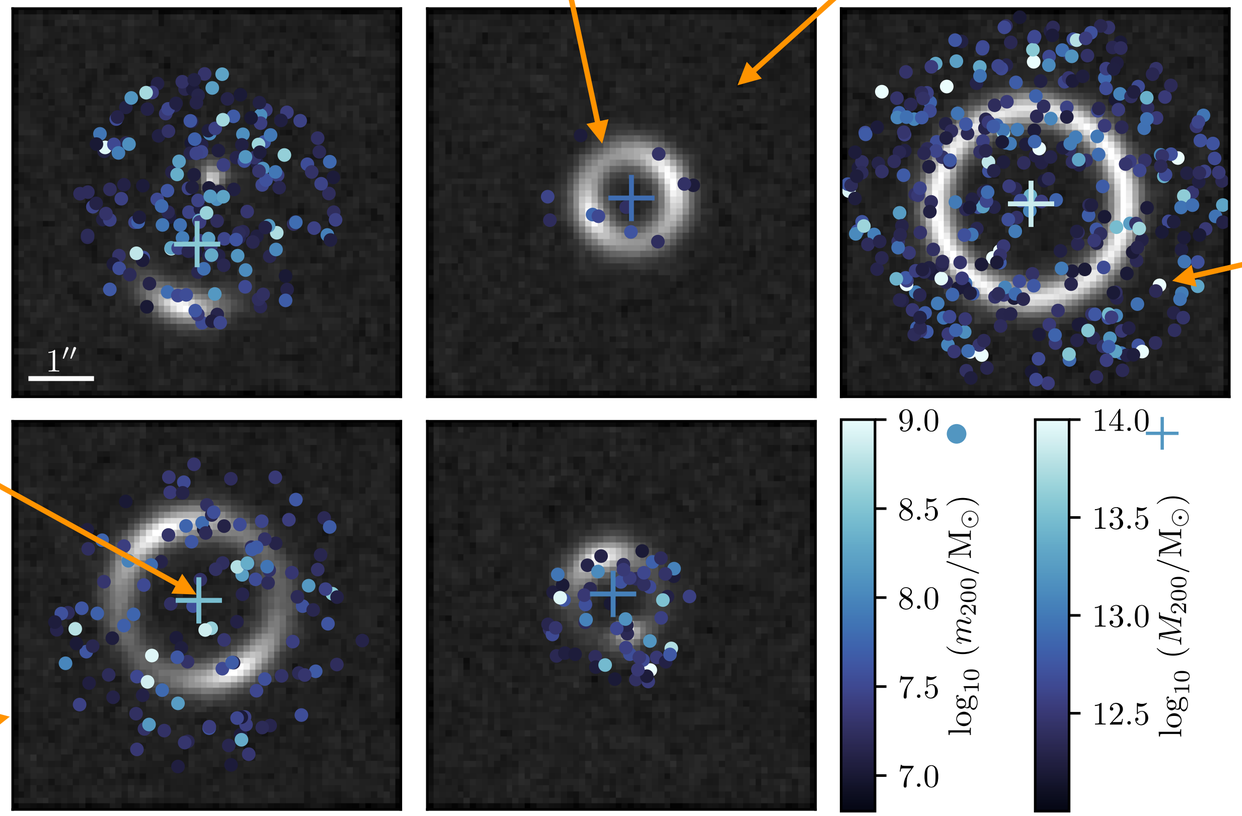
[following T. Collett 1507.02657]

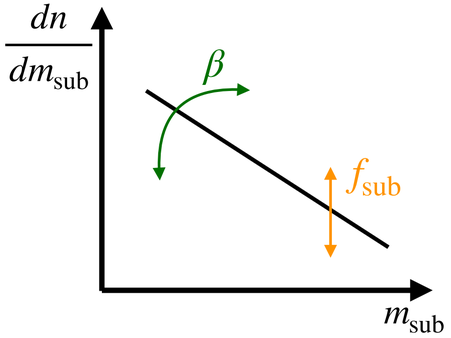
Spherical lensing host galaxies
at redshift ~0.5…1

Poisson fluctuations

Extended galaxy sources
at redshift 1.5

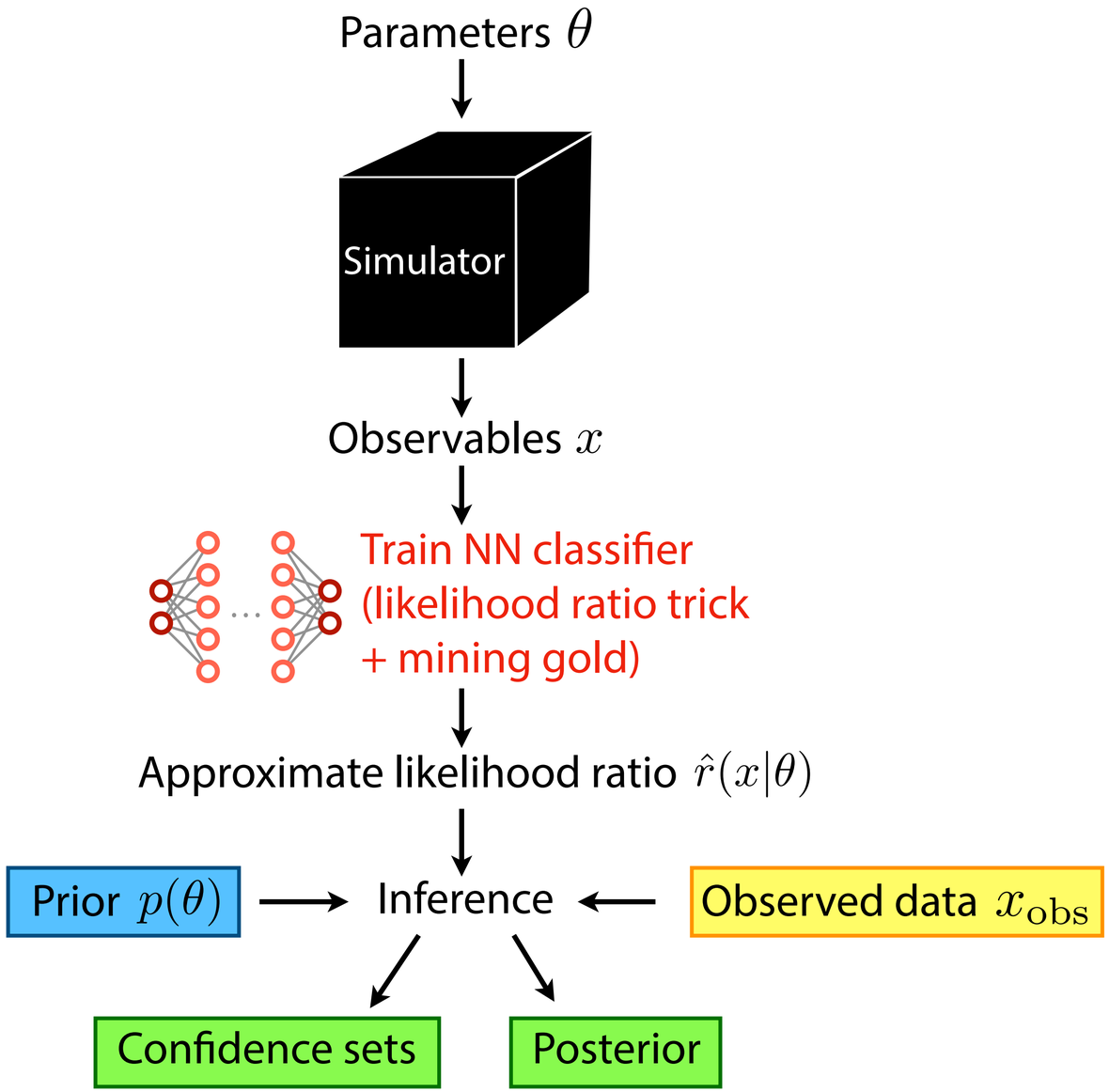
Subhalos follow mass distribution with two parameters



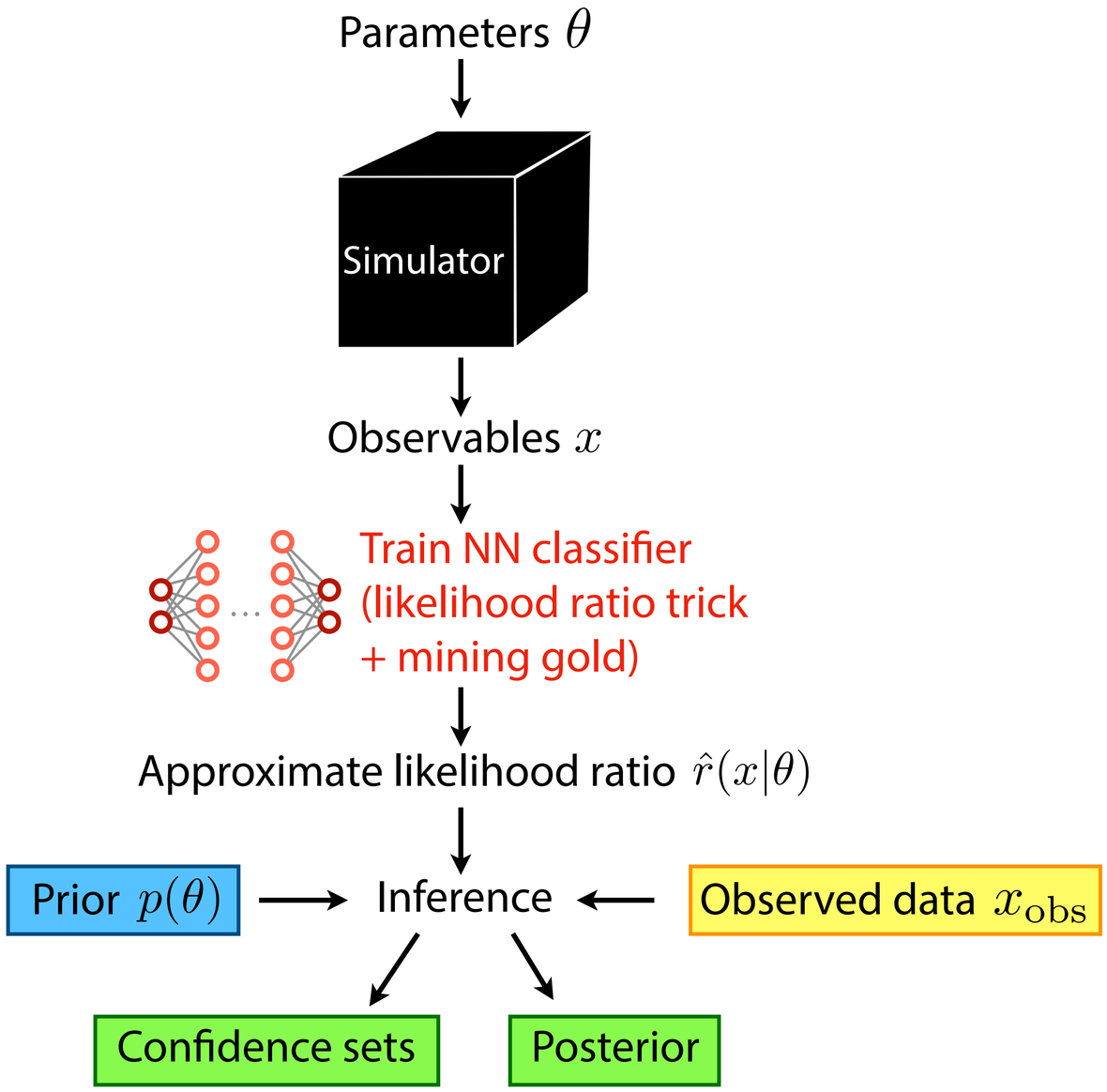


Euclid detector model
(single optical band,
64^2 pixel, Gaussian PSF)

Inference setup

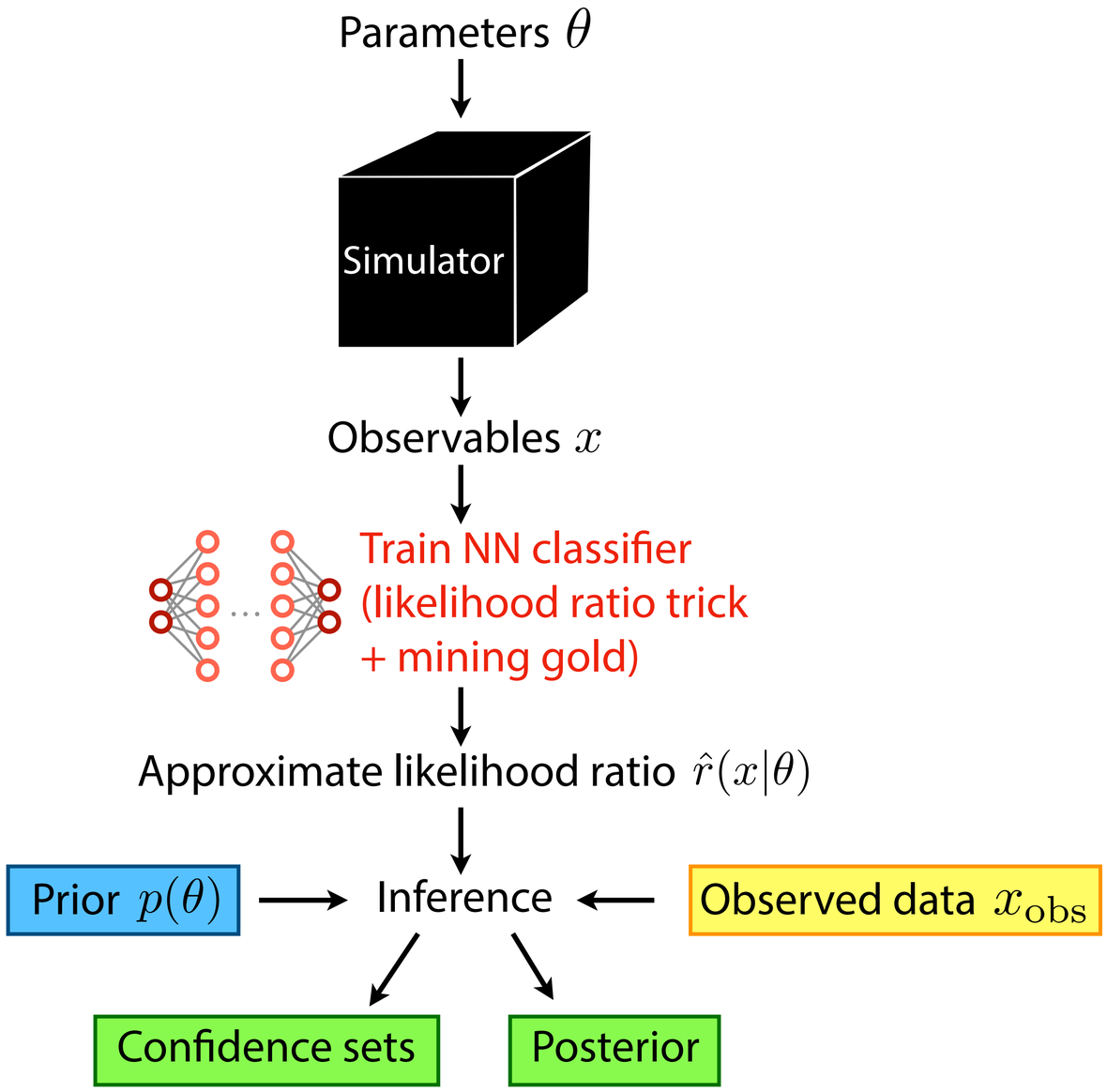


Inference setup



Training data: 10^6 lensed images with
$0 \leq f_{\rm sub} \leq 0.2, -1.5 \leq \beta \leq -0.5$

Inference setup



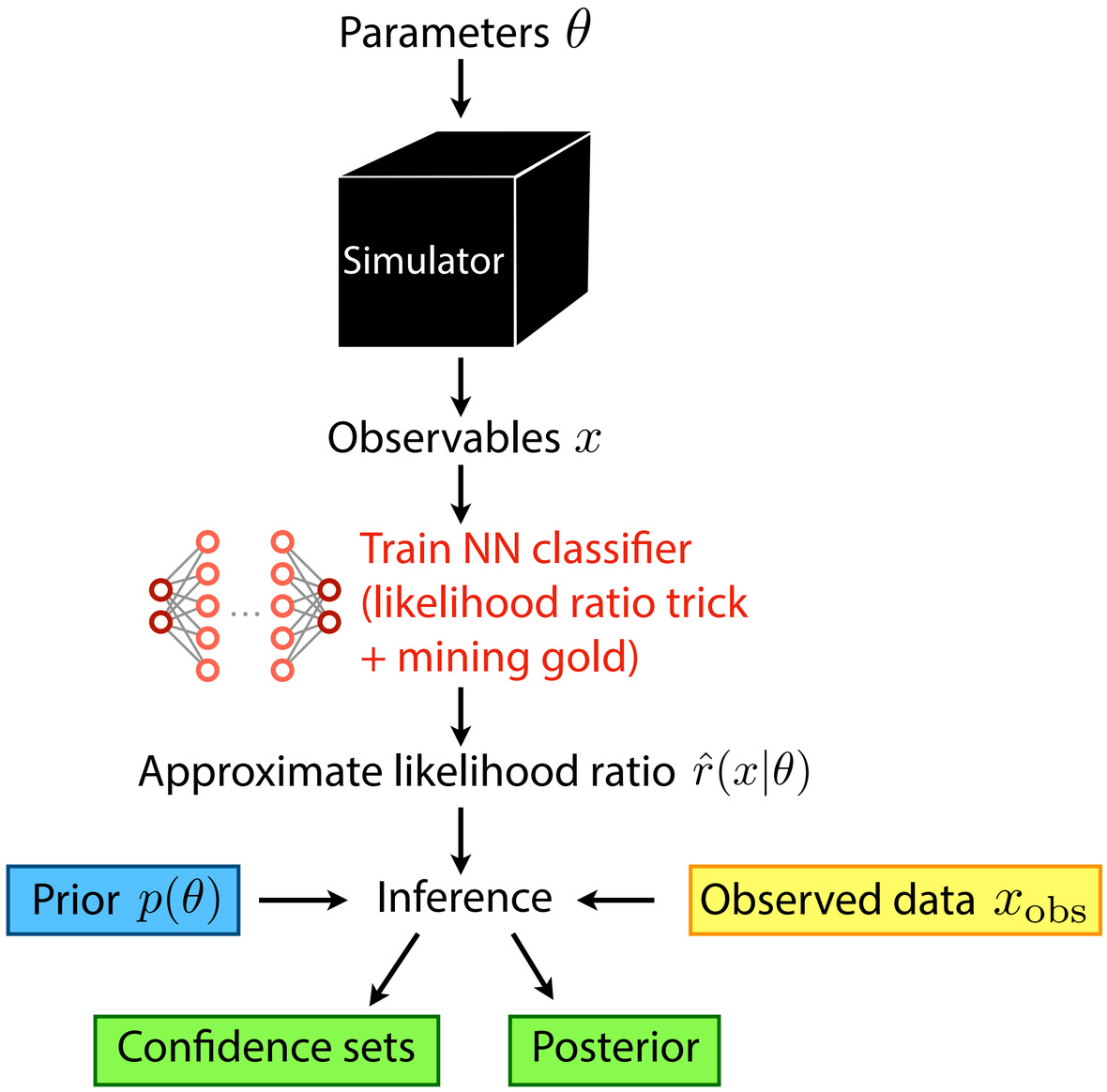
Training data: 10^6 lensed images with
$0 \leq f_{\rm sub} \leq 0.2, -1.5 \leq \beta \leq -0.5$

Convolutional neural network (modified ResNet-18) trained on ALICES loss

[M. Stoye, JB, J. Pavez, G, Louppe, K. Cranmer 1808.00973]

Calibration of network output

Inference setup



Training data: 10^6 lensed images with
$0 \leq f_{\rm sub} \leq 0.2, -1.5 \leq \beta \leq -0.5$

Convolutional neural network (modified ResNet-18) trained on ALICES loss

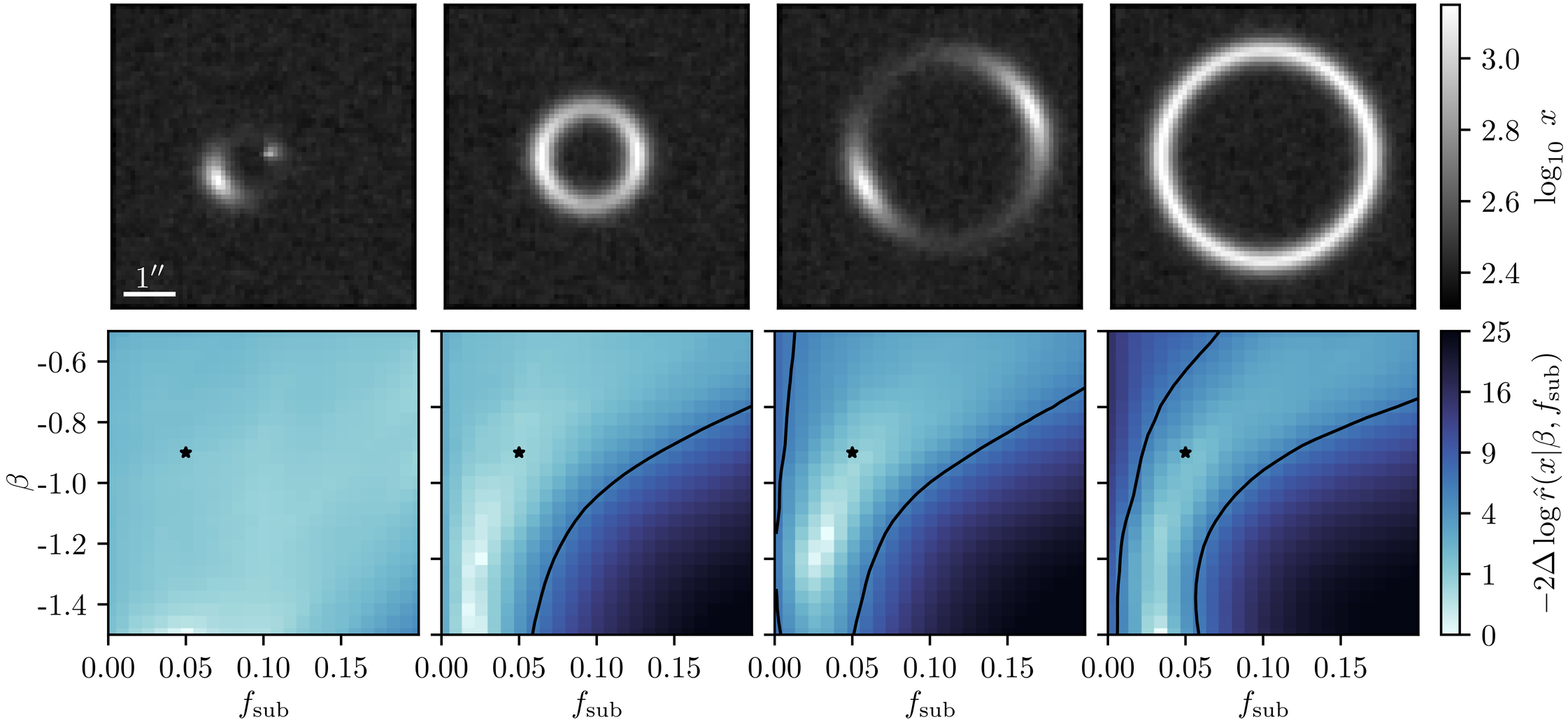
[M. Stoye, JB, J. Pavez, G, Louppe, K. Cranmer 1808.00973]

Calibration of network output

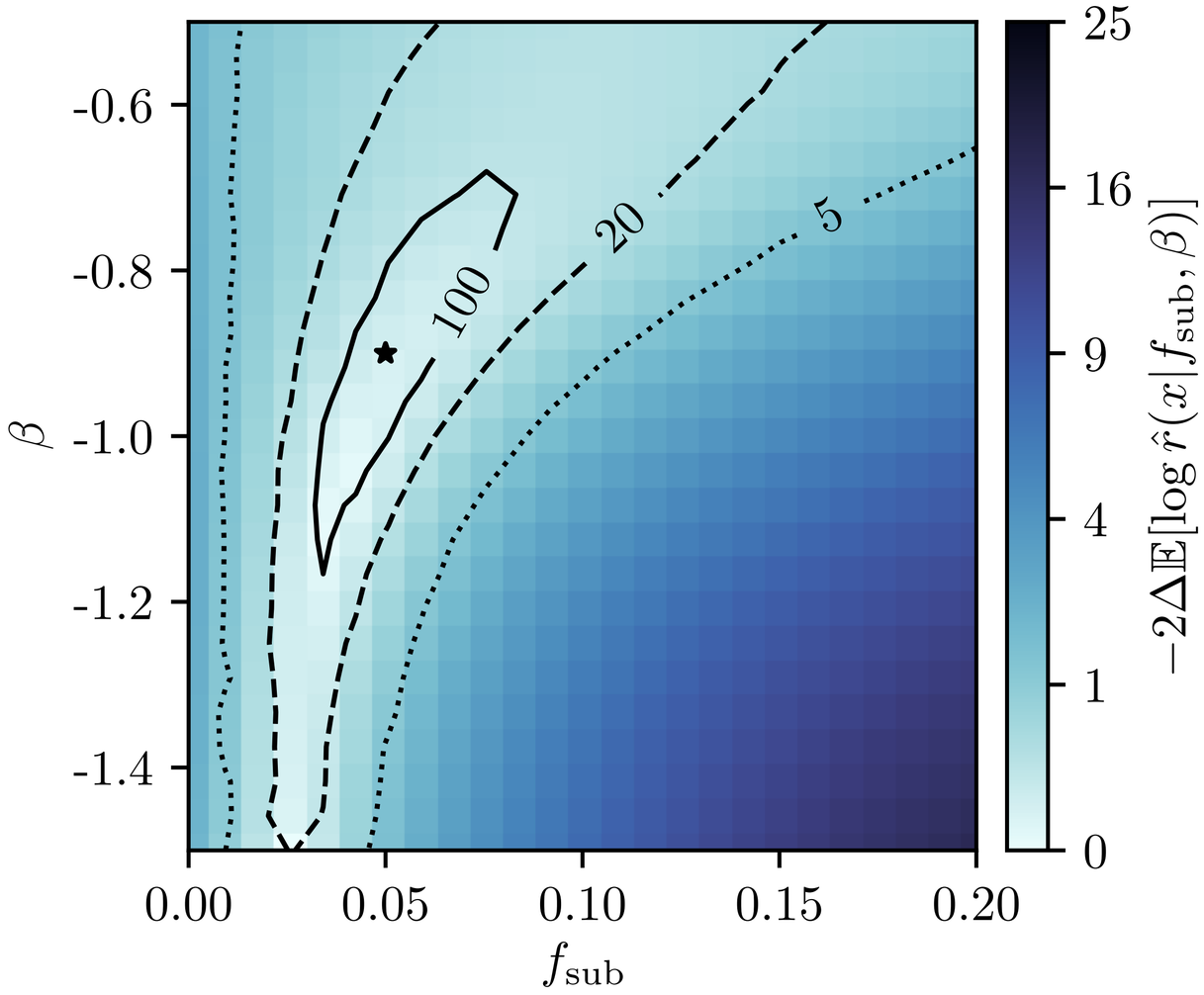
Synthetic "observed" data set: $f_{\rm sub} = 0.05, \beta = -0.9$

Bayesian & frequentist inference

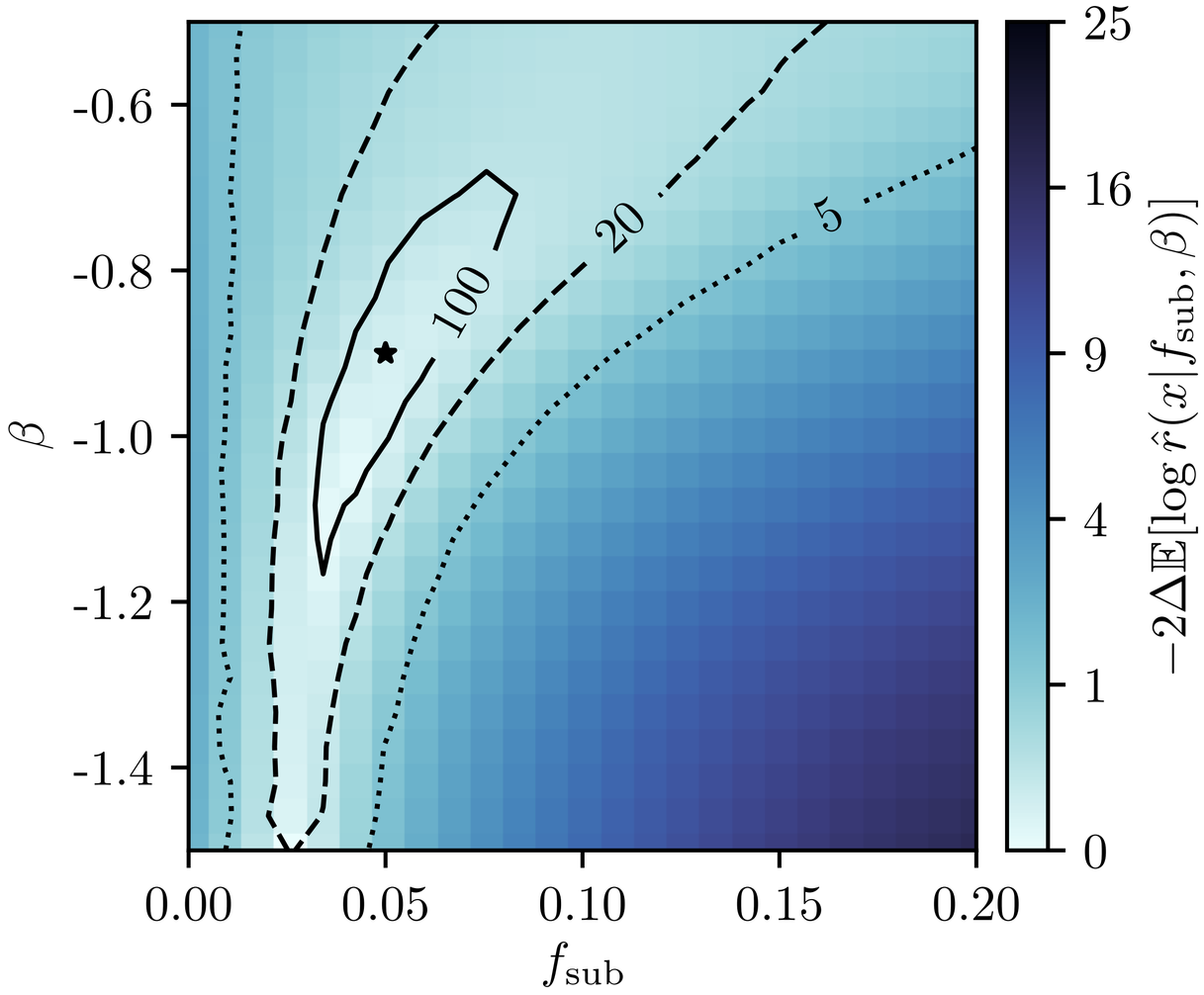
Inferring parameters from individual images

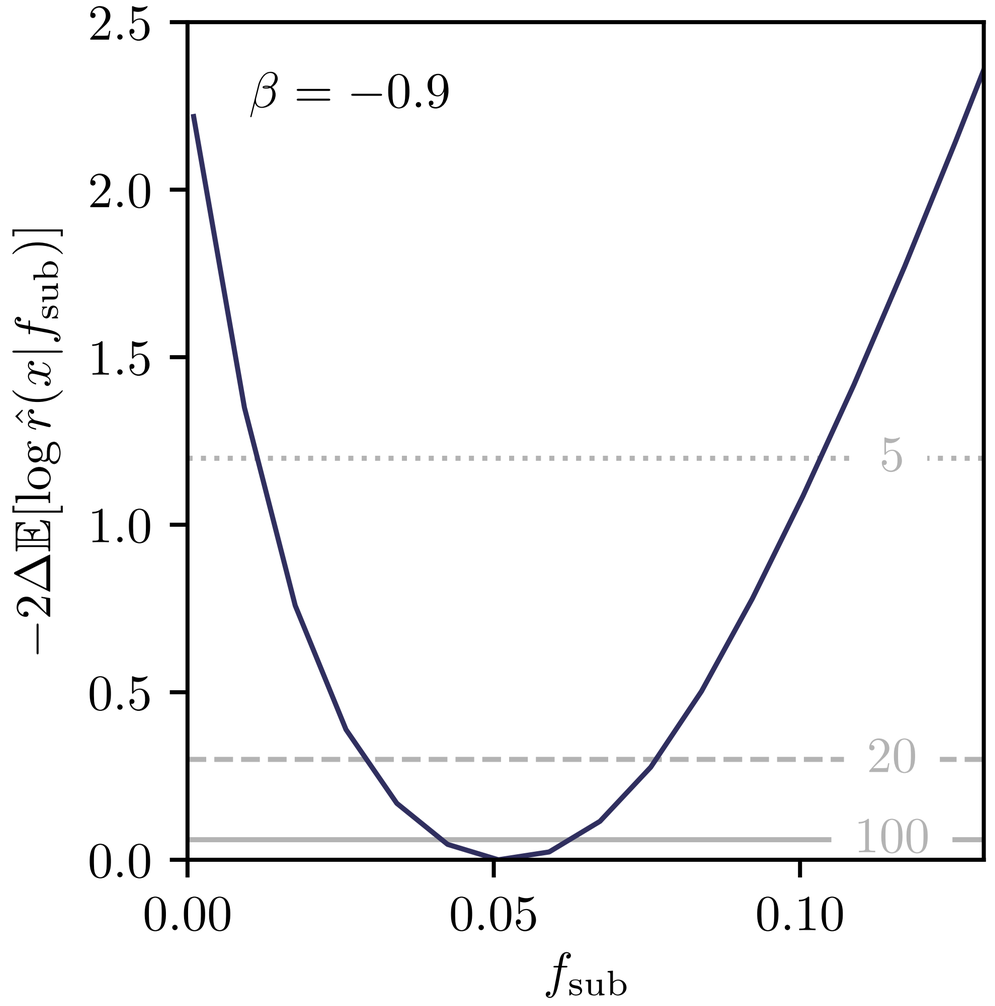


Expected likelihood ratio map

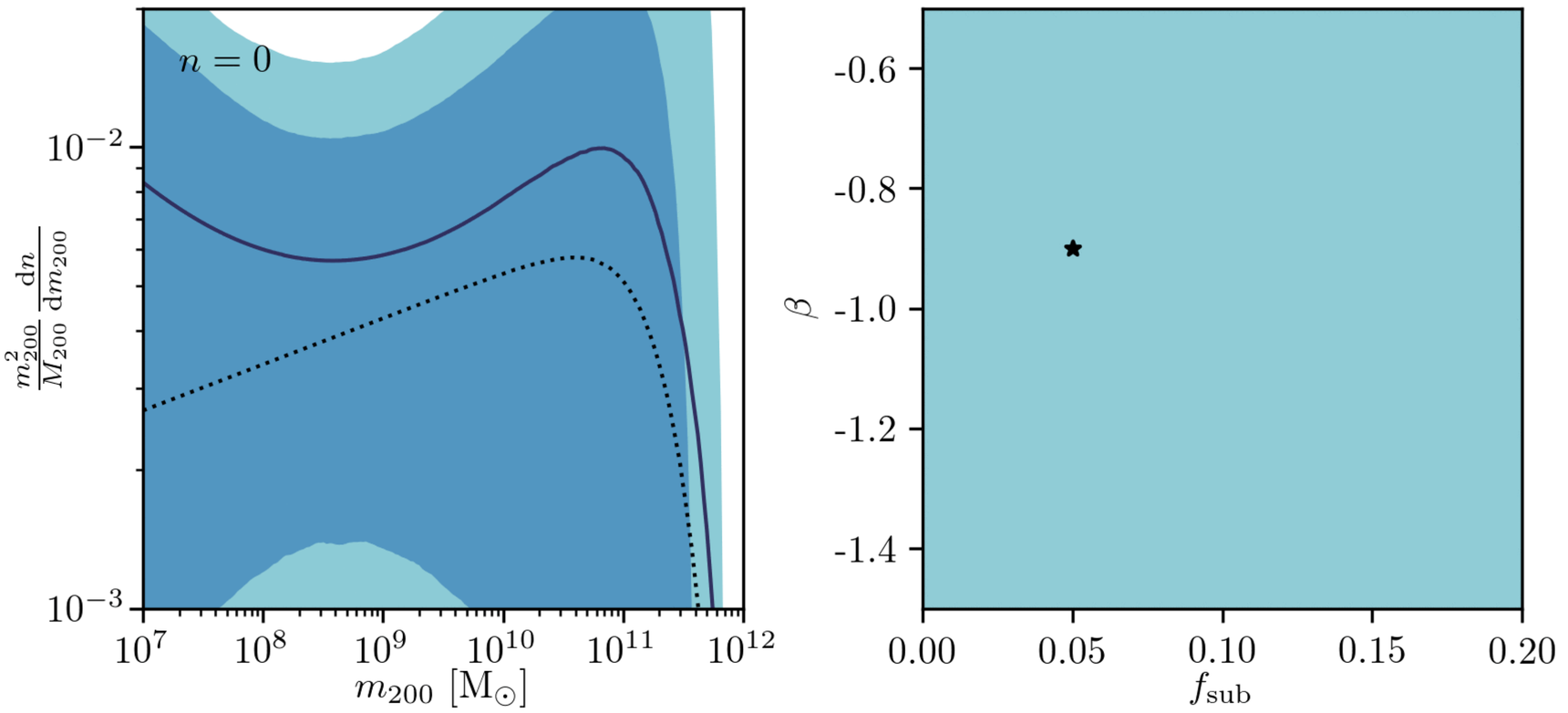


Expected likelihood ratio map

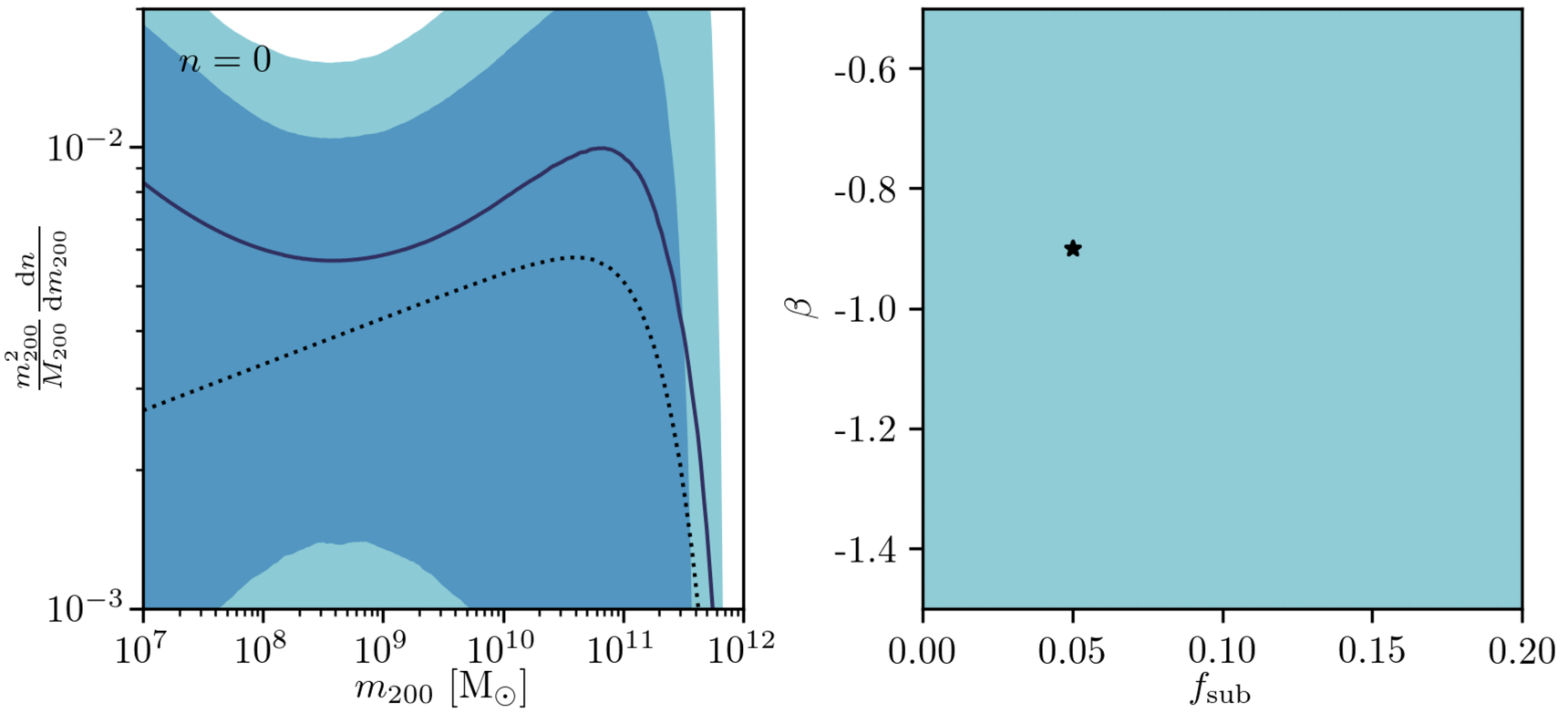




Bayesian inference

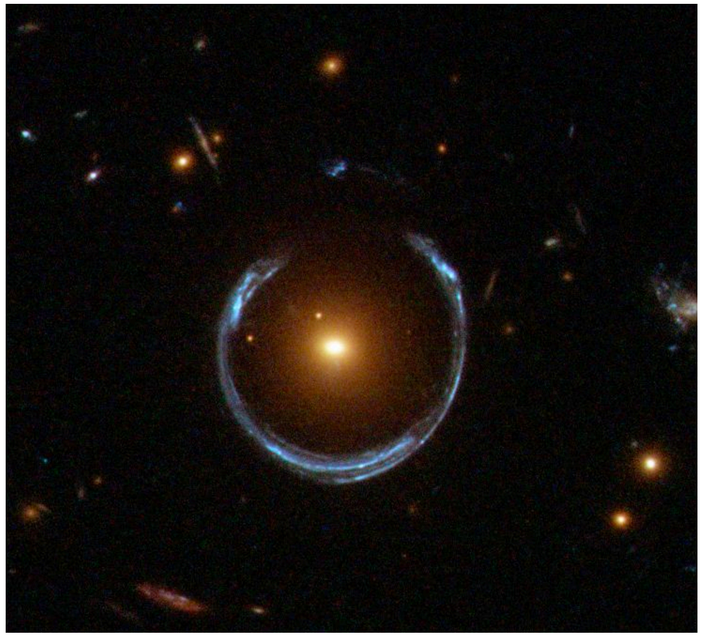


Bayesian inference



All the things we didn’t do

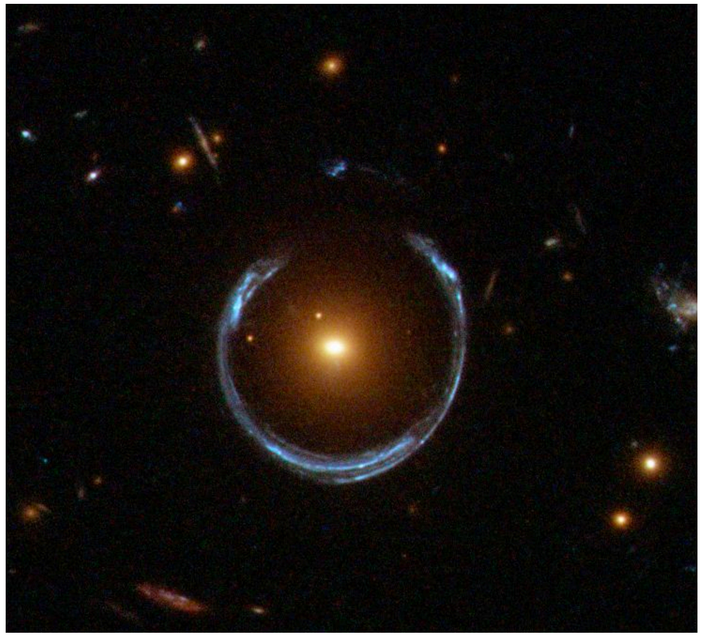
- More involved subhalo mass function
  - Warm DM with DM mass as parameter



[ESA/Hubble/NASA]

All the things we didn’t do

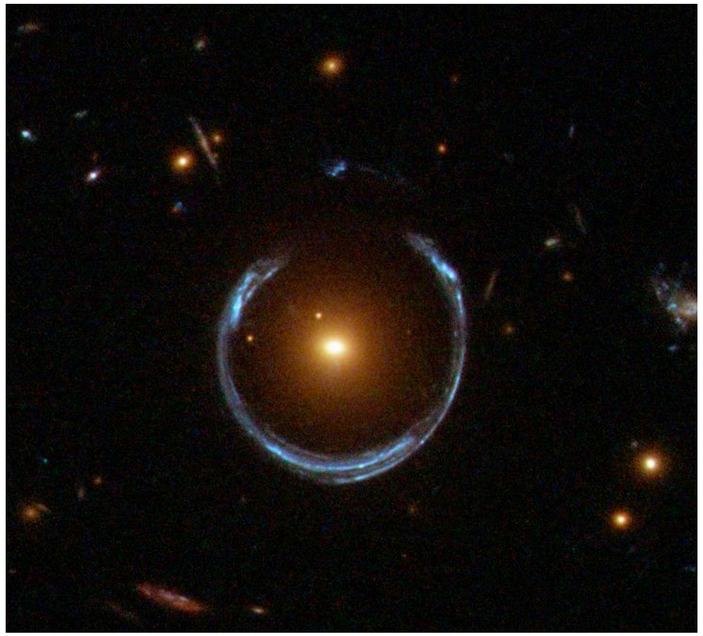
- More involved subhalo mass function
  - Warm DM with DM mass as parameter
- Realistic simulators
  - More diverse source and host galaxies (e.g. data-driven)
  - Realistic subhalo modelling (tidal disruption, redshift dependence…)
  - Line-of-sight substructure
  - Realistic observation model (variable exposure / PSF, multiple bands…)



[ESA/Hubble/NASA]

All the things we didn’t do

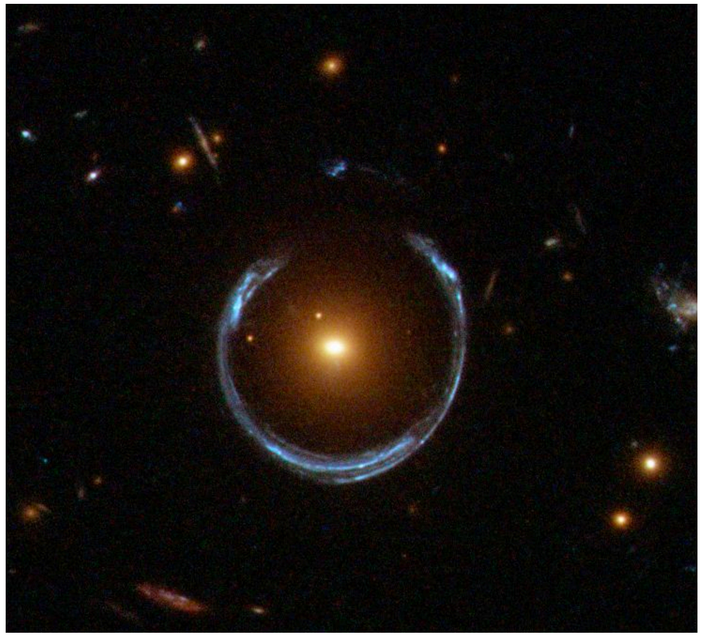
- More involved subhalo mass function
  - Warm DM with DM mass as parameter
- Realistic simulators
  - More diverse source and host galaxies (e.g. data-driven)
  - Realistic subhalo modelling (tidal disruption, redshift dependence…)
  - Line-of-sight substructure
  - Realistic observation model (variable exposure / PSF, multiple bands…)
- Use auxiliary information during inference



[ESA/Hubble/NASA]

All the things we didn’t do

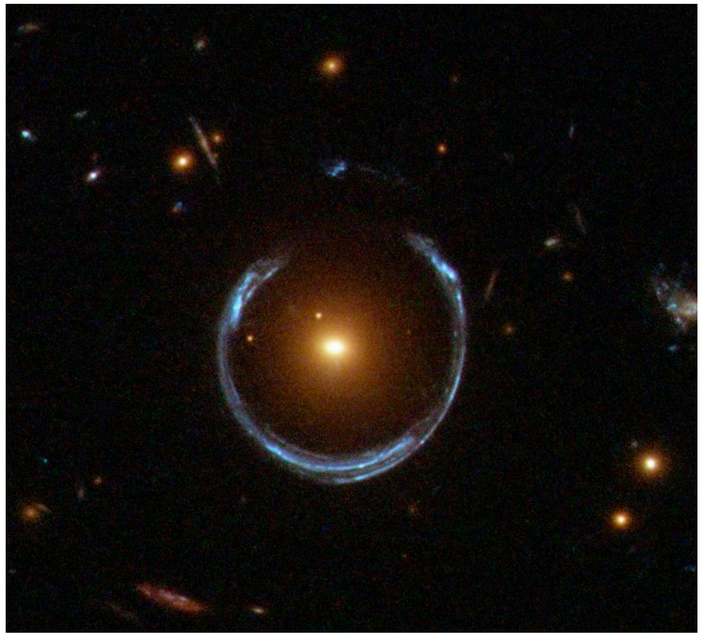
- More involved subhalo mass function
  - Warm DM with DM mass as parameter
- Realistic simulators
  - More diverse source and host galaxies (e.g. data-driven)
  - Realistic subhalo modelling (tidal disruption, redshift dependence…)
  - Line-of-sight substructure
  - Realistic observation model (variable exposure / PSF, multiple bands…)
- Use auxiliary information during inference
- Evaluation on real data



[ESA/Hubble/NASA]

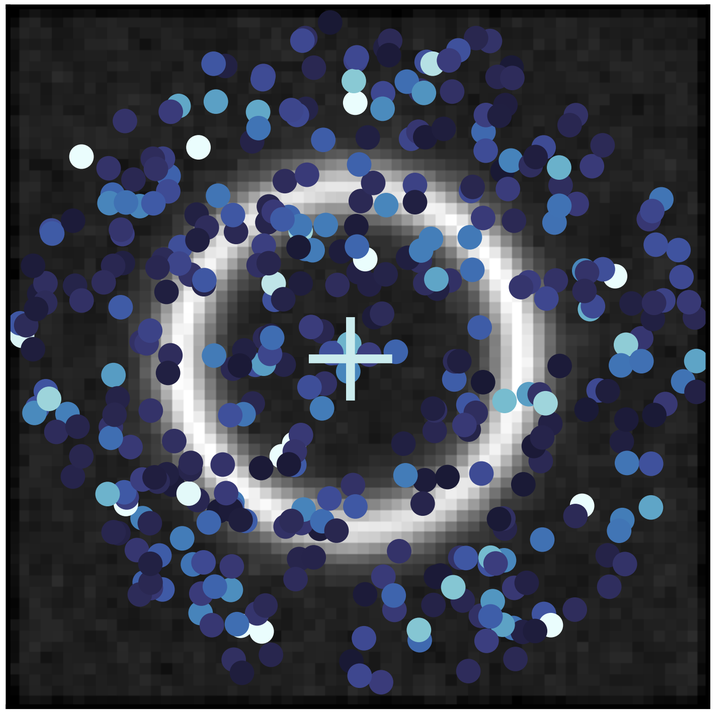
All the things we didn't do

- More involved subhalo mass function
  - Warm DM with DM mass as parameter
- Realistic simulators
  - More diverse source and host galaxies (e.g. data-driven)
  - Realistic subhalo modelling (tidal disruption, redshift dependence…)
  - Line-of-sight substructure
  - Realistic observation model (variable exposure / PSF, multiple bands…)
- Use auxiliary information during inference
- Evaluation on real data

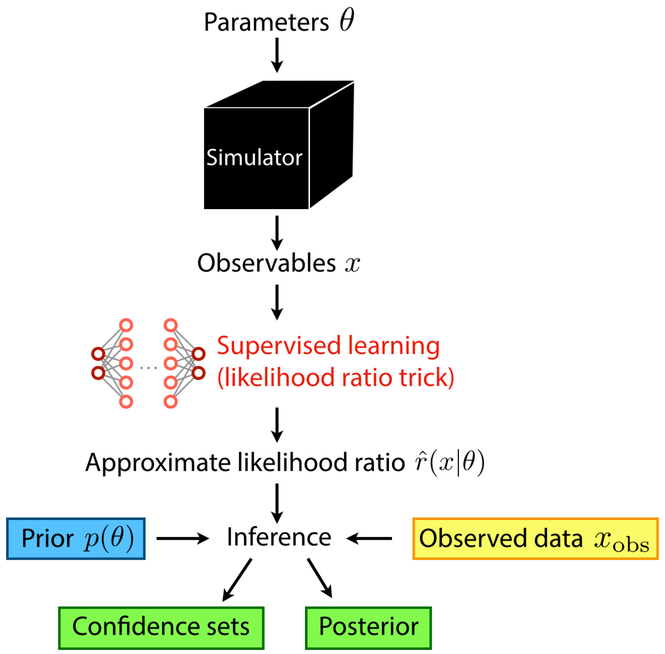


[ESA/Hubble/NASA]

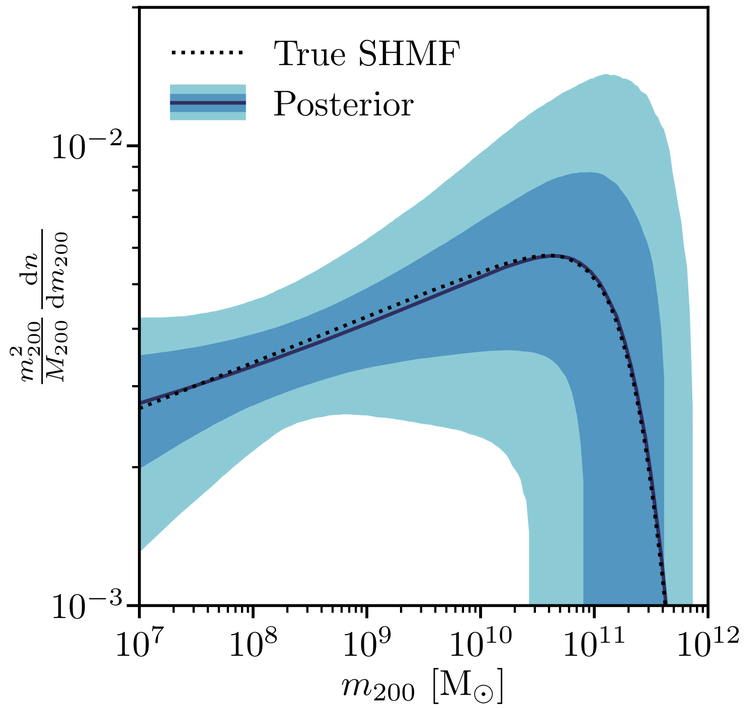
⇒ Our method should scale to a realistic setting, but will require more simulations and careful sanity checks



The small-scale structure of Dark Matter affects strong lensing, but teasing out this information is challenging.



New simulation-based inference algorithms based on machine learning can solve this problem in a scalable way.



In a proof-of-concept study, they allow us to constrain Dark Matter substructure from Euclid-like lensing observations.

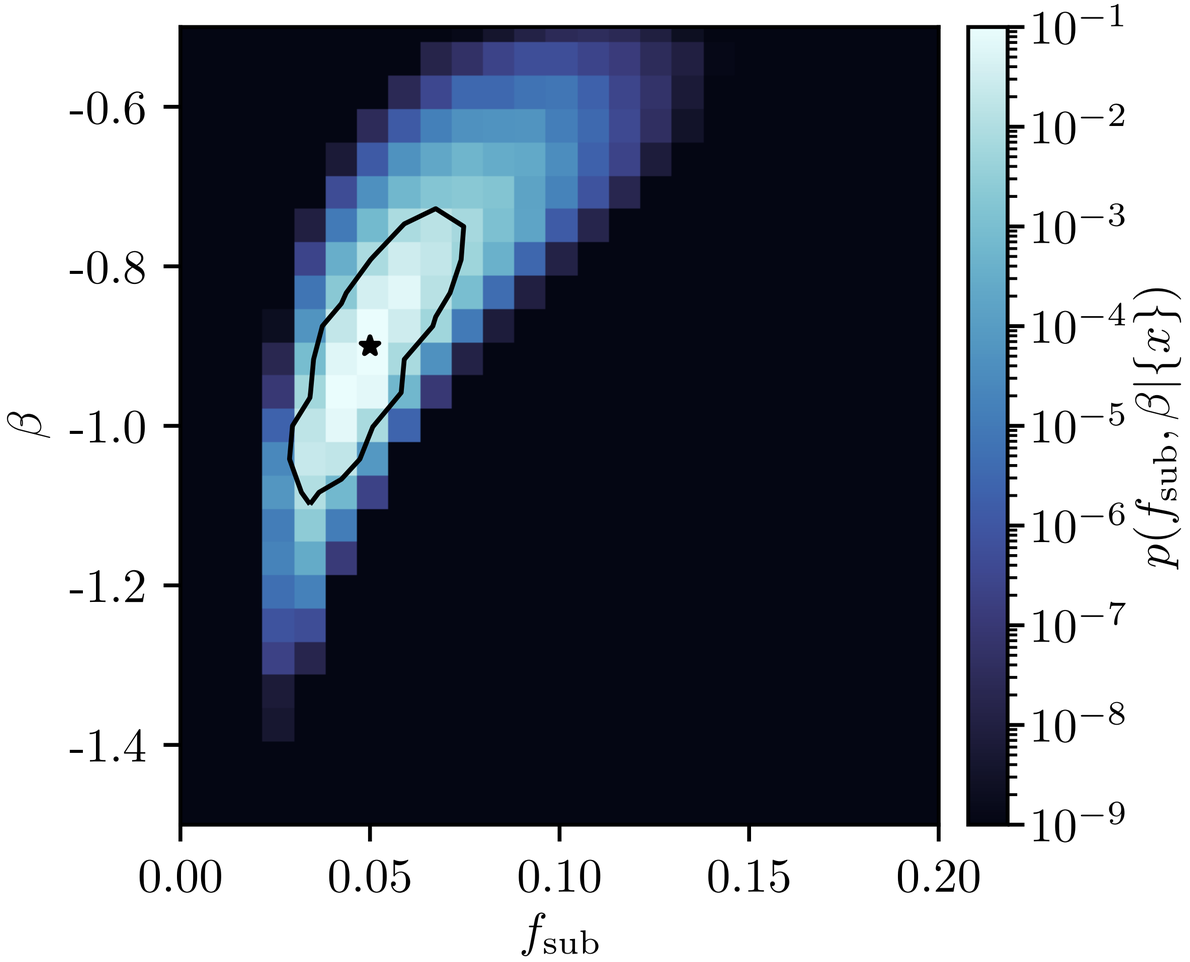
Strong lensing: **J**B, S. Mishra-Sharma, J. Hermans, G. Louppe, K. Cranmer 1909.02005

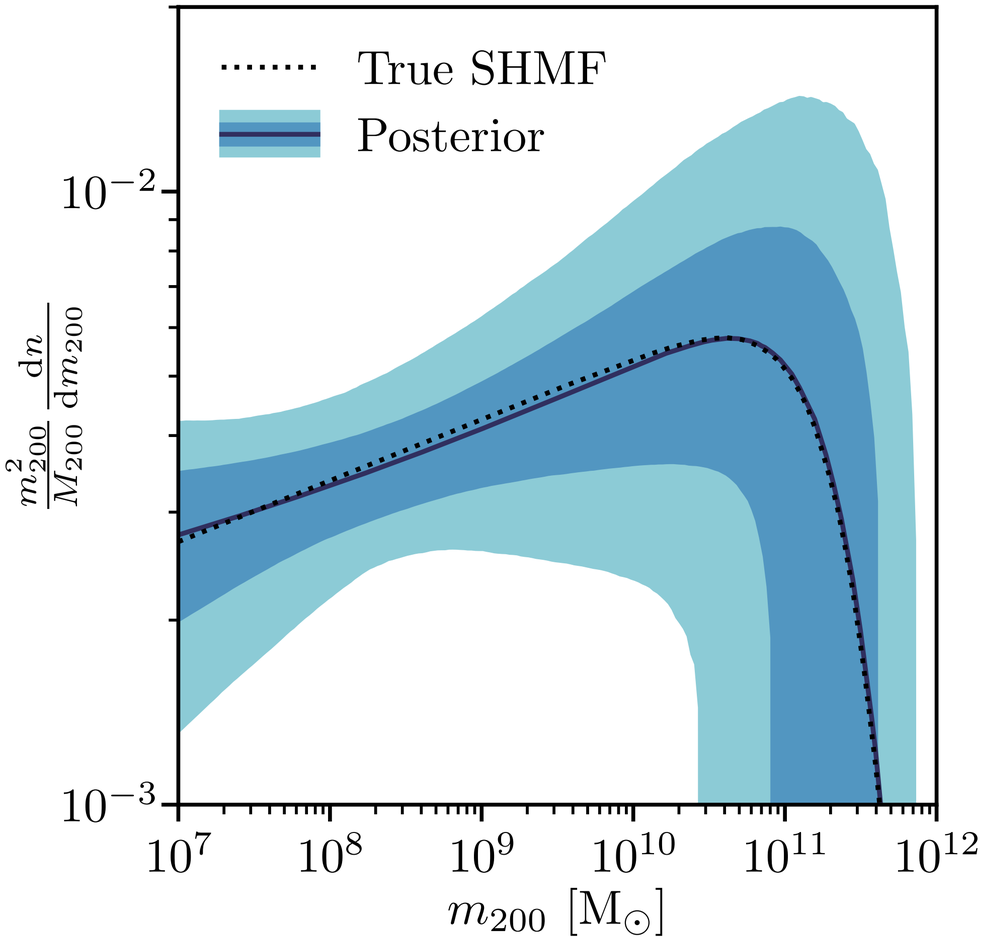
Broader review of simulation-based inference: K. Cranmer, JB, G. Louppe 1911.01429

“Hooray! Question mark?”

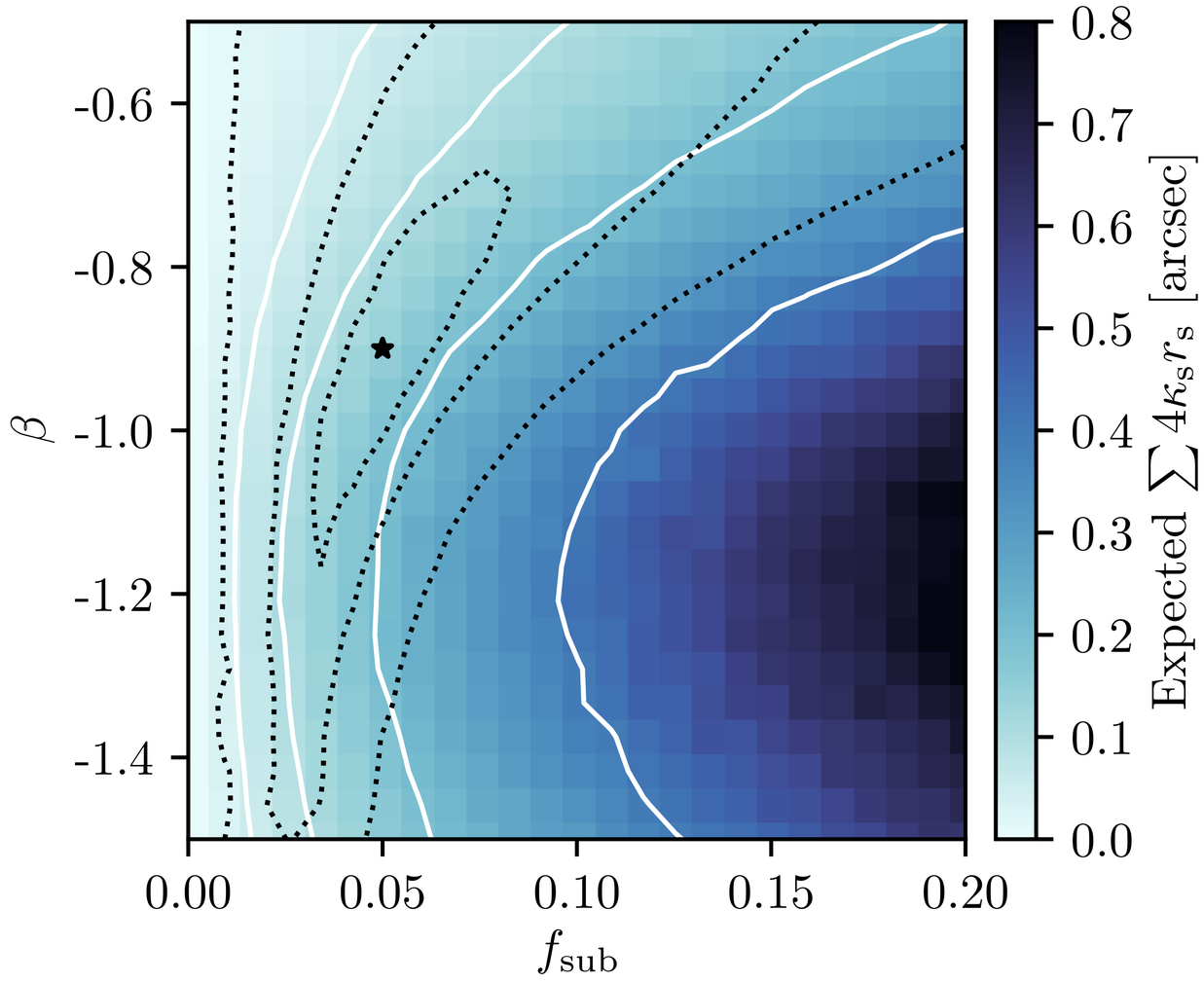
— Todd Chavez

Expected posterior

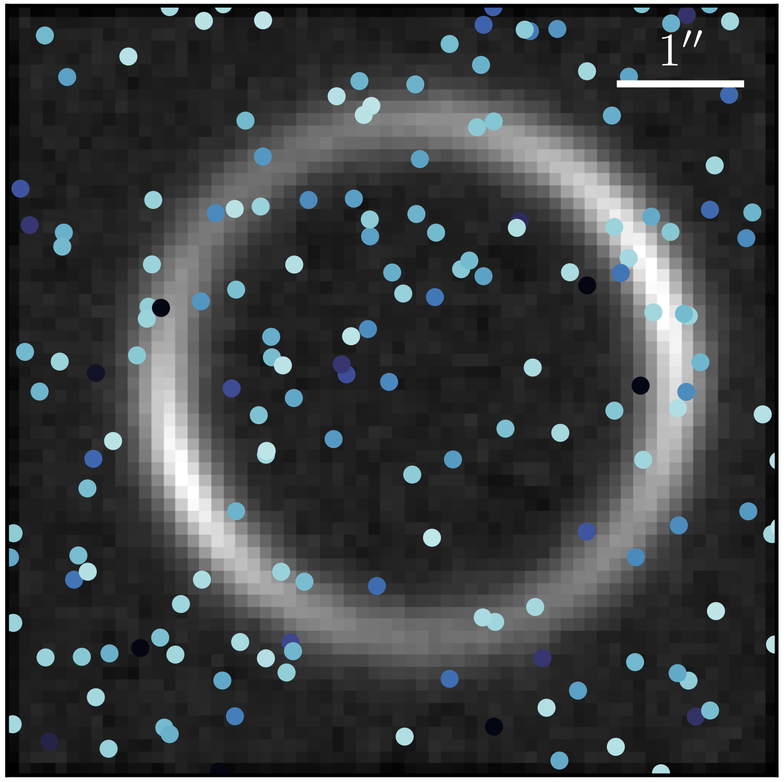


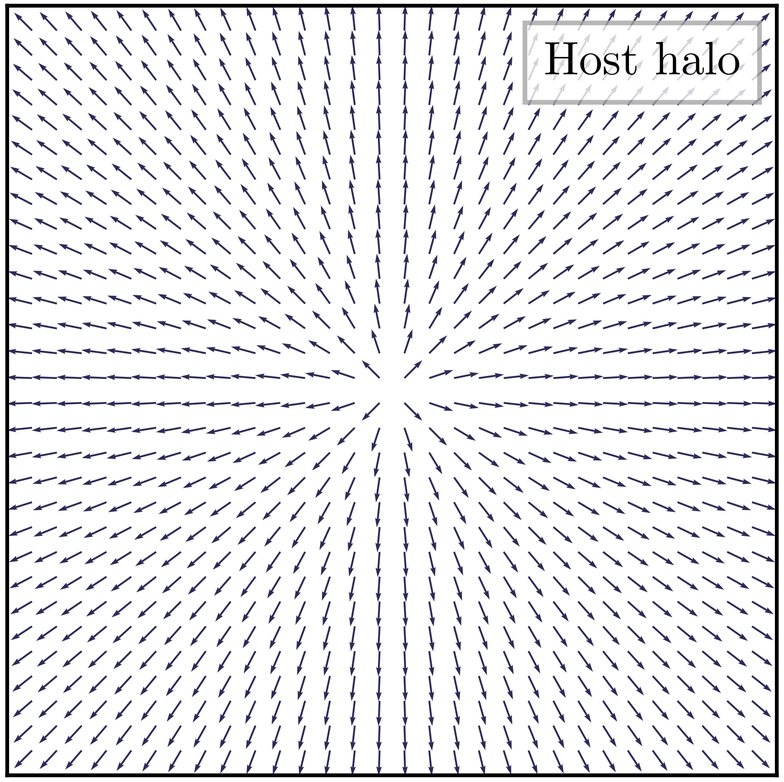


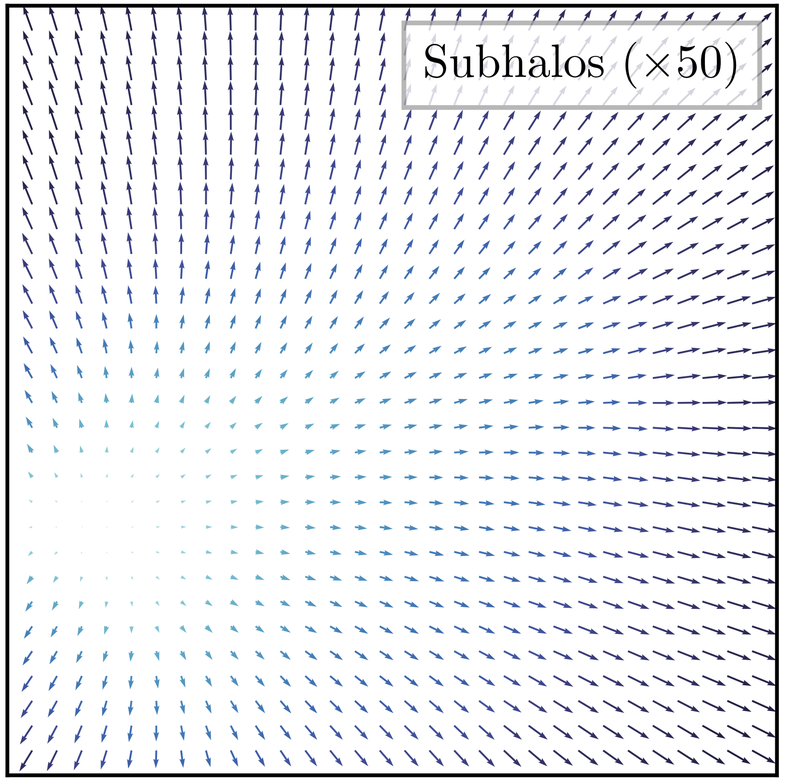
Likelihood vs total subhalo deflection



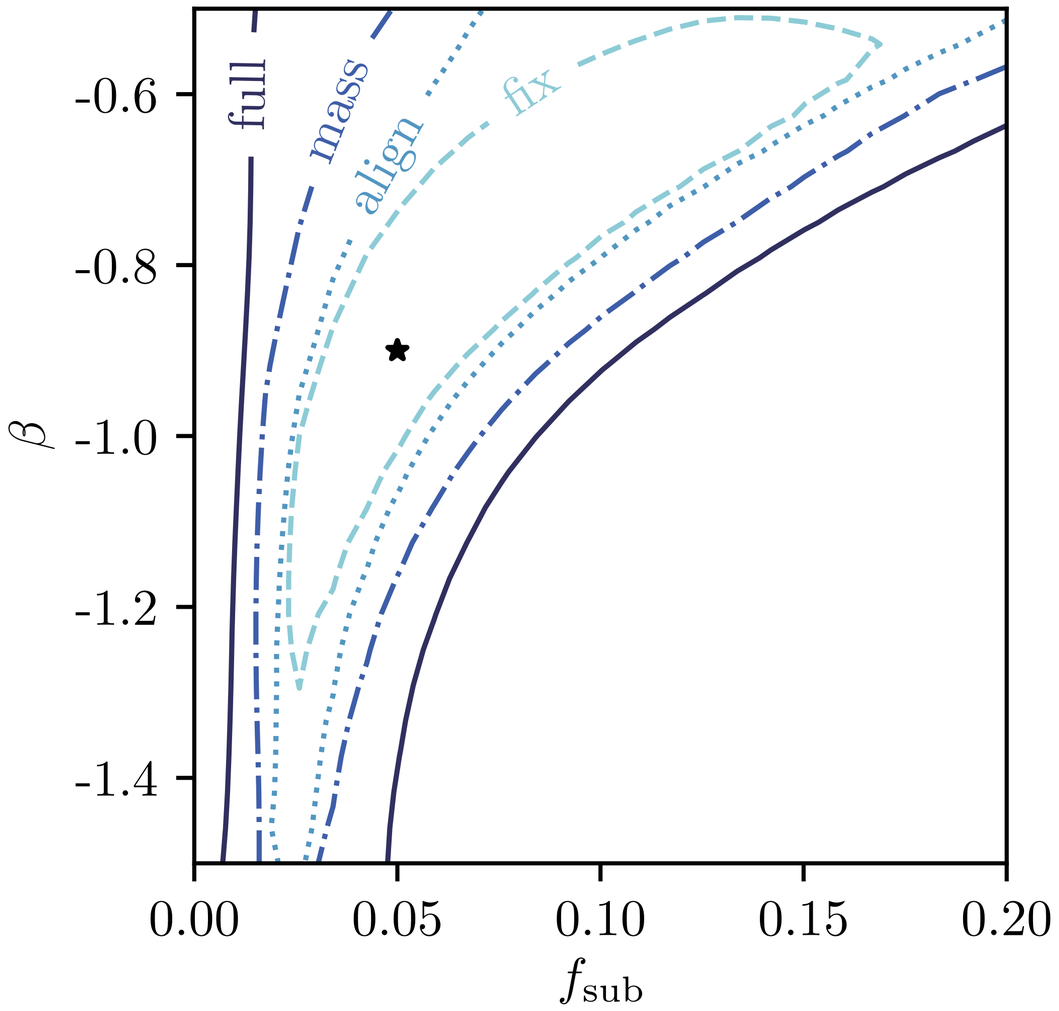
Deflection maps

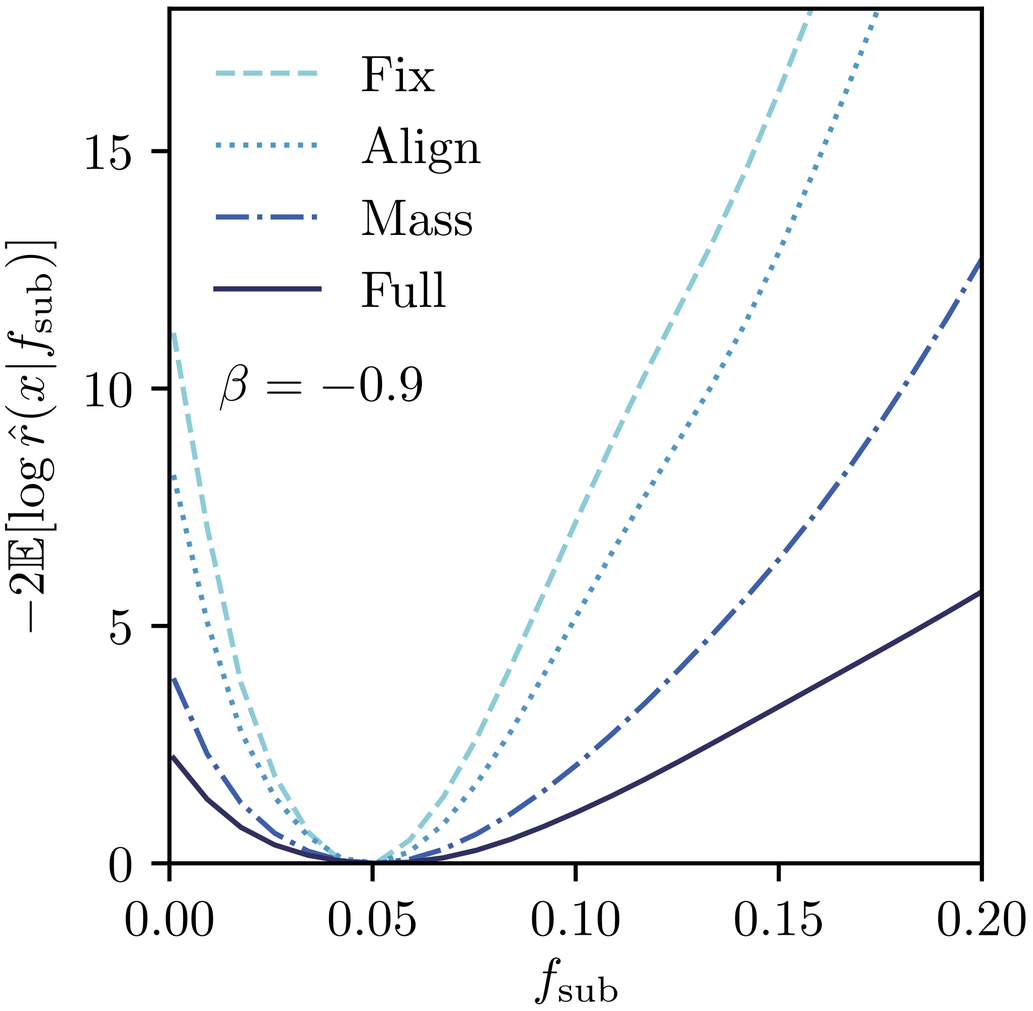






Simplified scenarios

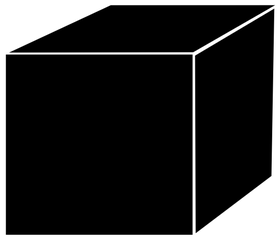




Formalizing the problem

You are given

- a simulator that lets you generate N samples $x_i \sim p(x_i|\theta_i)$ for parameter points θ_i of your choice



- observed data $x_{\mathrm{obs}} \sim p(x_{\mathrm{obs}}|\theta_{\mathrm{true}})$
- prior belief $p(\theta)$

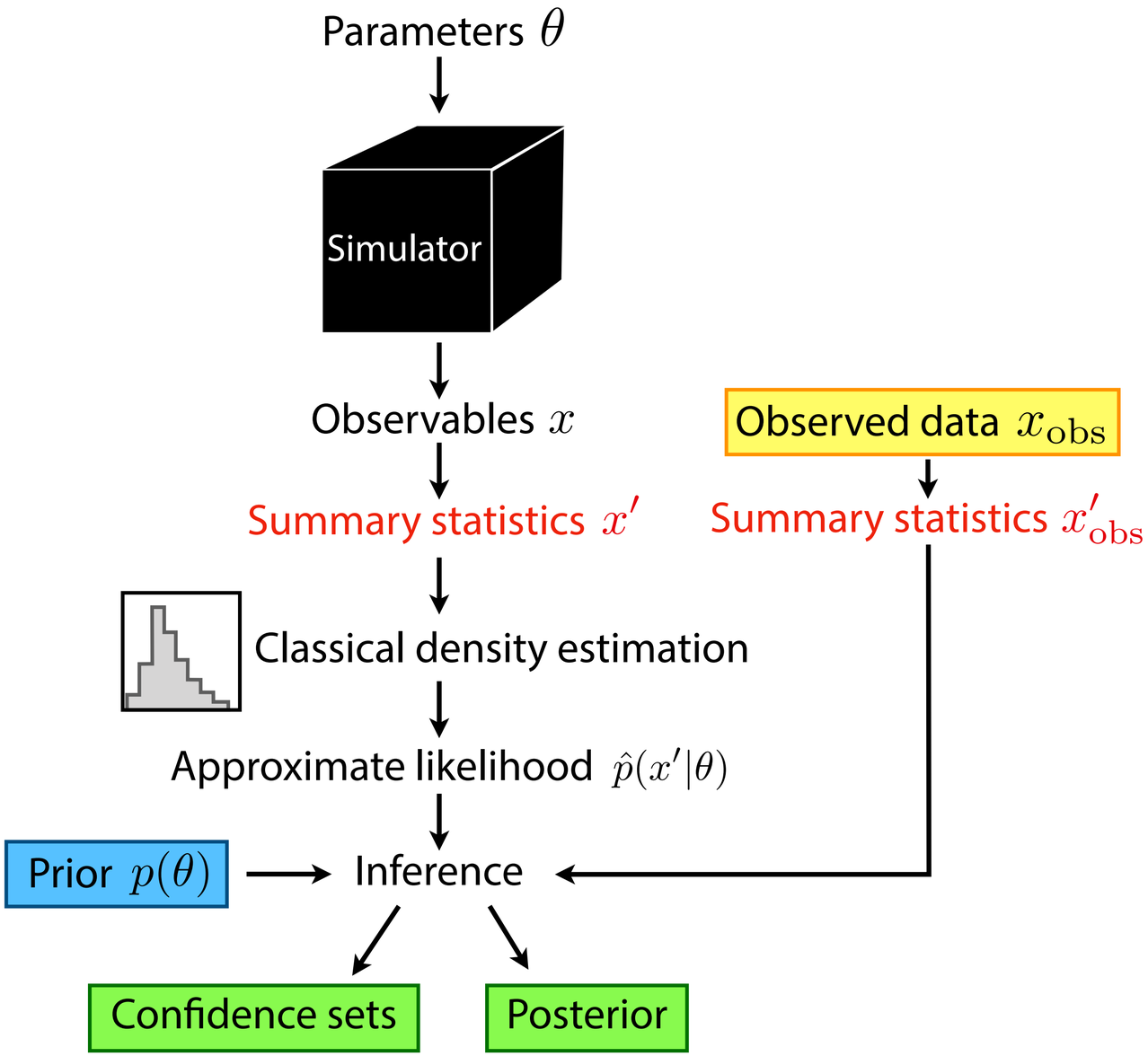
Goals: estimate either…

- true parameters $\hat{\theta}_{\mathrm{true}}$
- confidence sets based on likelihood $\hat{p}(x_{\mathrm{obs}}|\theta)$
- posterior $\hat{p}(\theta|x_{\mathrm{obs}}) = \dfrac{\hat{p}(x_{\mathrm{obs}}|\theta)\, p(\theta)}{\int \mathrm{d}\theta'\, \hat{p}(x_{\mathrm{obs}}|\theta')\, p(\theta')}$
  or samples from posterior $\theta \sim \hat{p}(\theta|x_{\mathrm{obs}})$

… depending on domain conventions

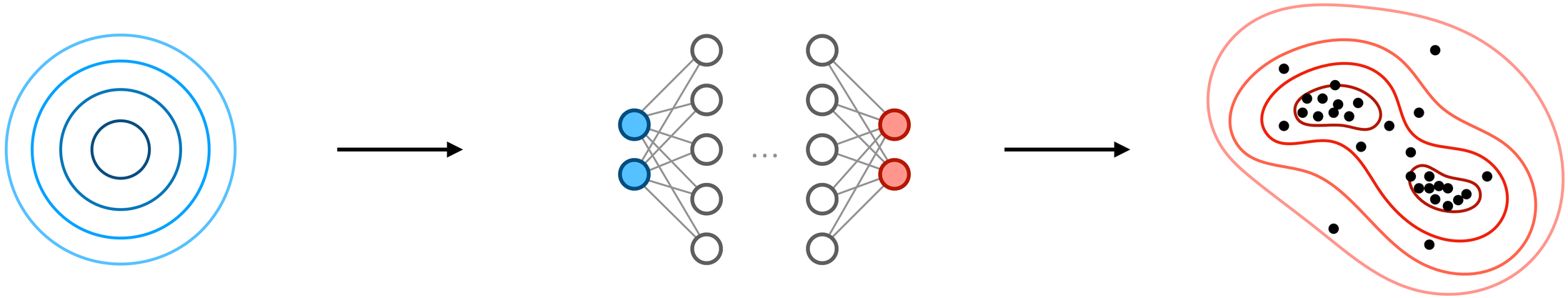
Inference by estimating the likelihood

[e.g. P. Diggle, R. Gratton 1984]



- Compression to summary statistics reduces quality of inference
- Curse of dimensionality: does not scale to more than a few summary statistics

High-dimensional density estimation with normalizing flows



Simple base density

$u \sim \pi(u)$

NN: transformation $x = f(u)$

- one-to-one and invertible
- differentiable
- f^{-1} and $\det \nabla f$ are tractable

Target density is given by

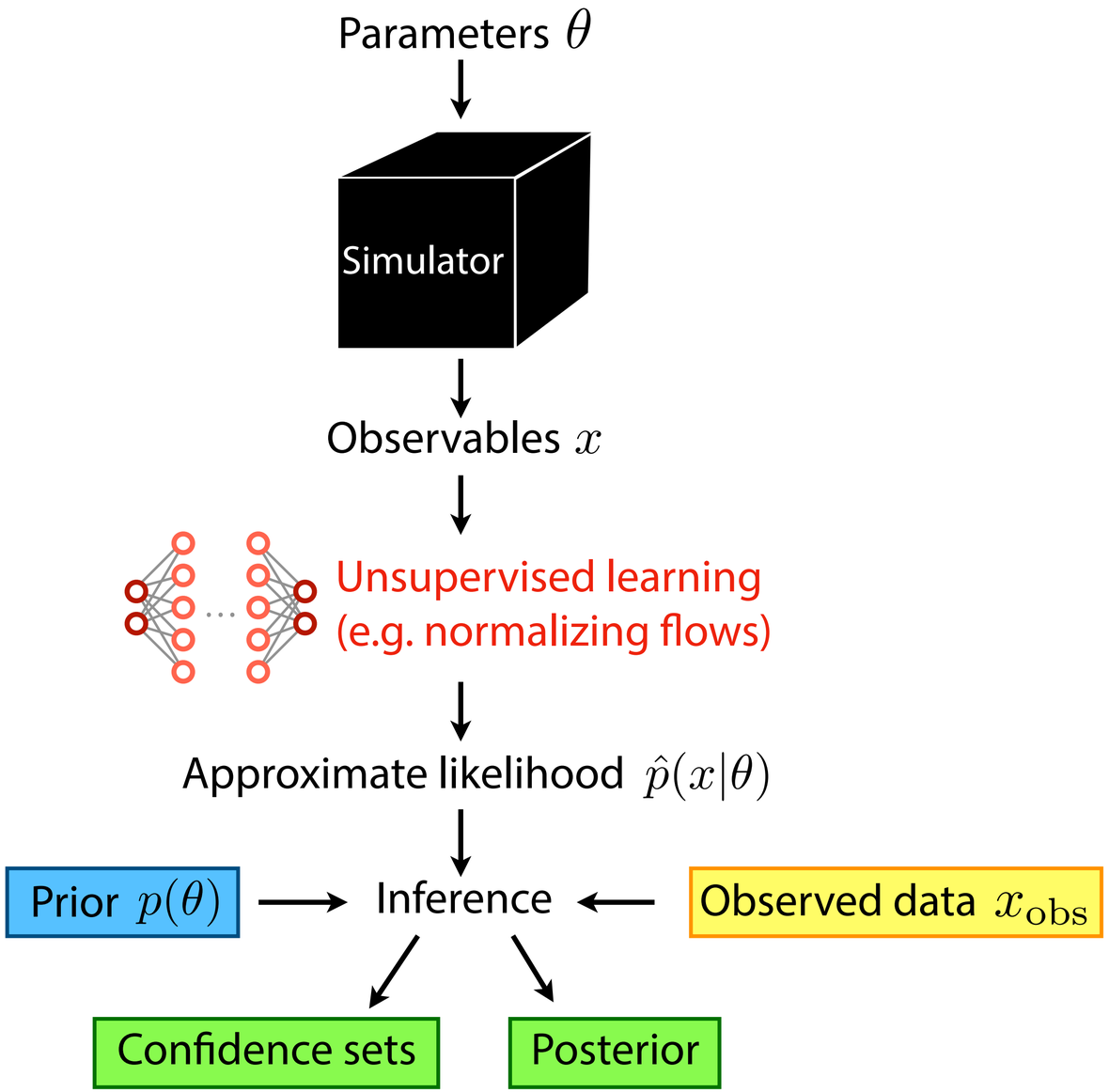
$\hat{p}(x) = \pi(f^{-1}(x)) \, |\det \nabla f|^{-1}$

Train transformation by maximizing $\log \hat{p}(x)$

Transformation can depend on θ to model conditional density $\log \hat{p}(x|\theta)$

[lots of great work recently; see G. Papamakarios et al 1912.02762 for a review]

Inference with neural likelihood estimation

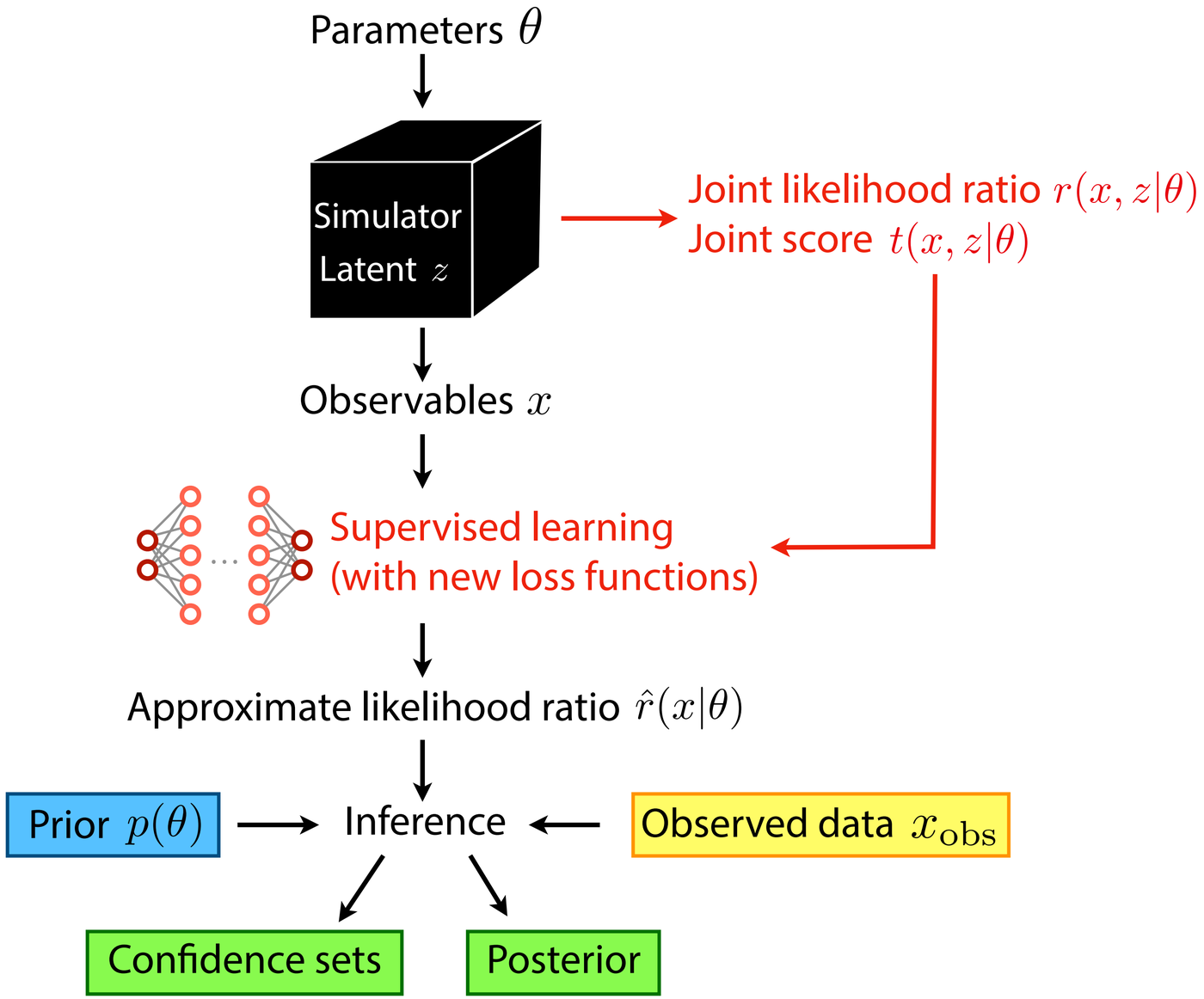


[G. Papamakarios, D. Sterrart, I. Murray 1805.07226;
J.-M. Lueckmann, G. Bassetto, T. Karaletsos, J. Macke 1805.09294]

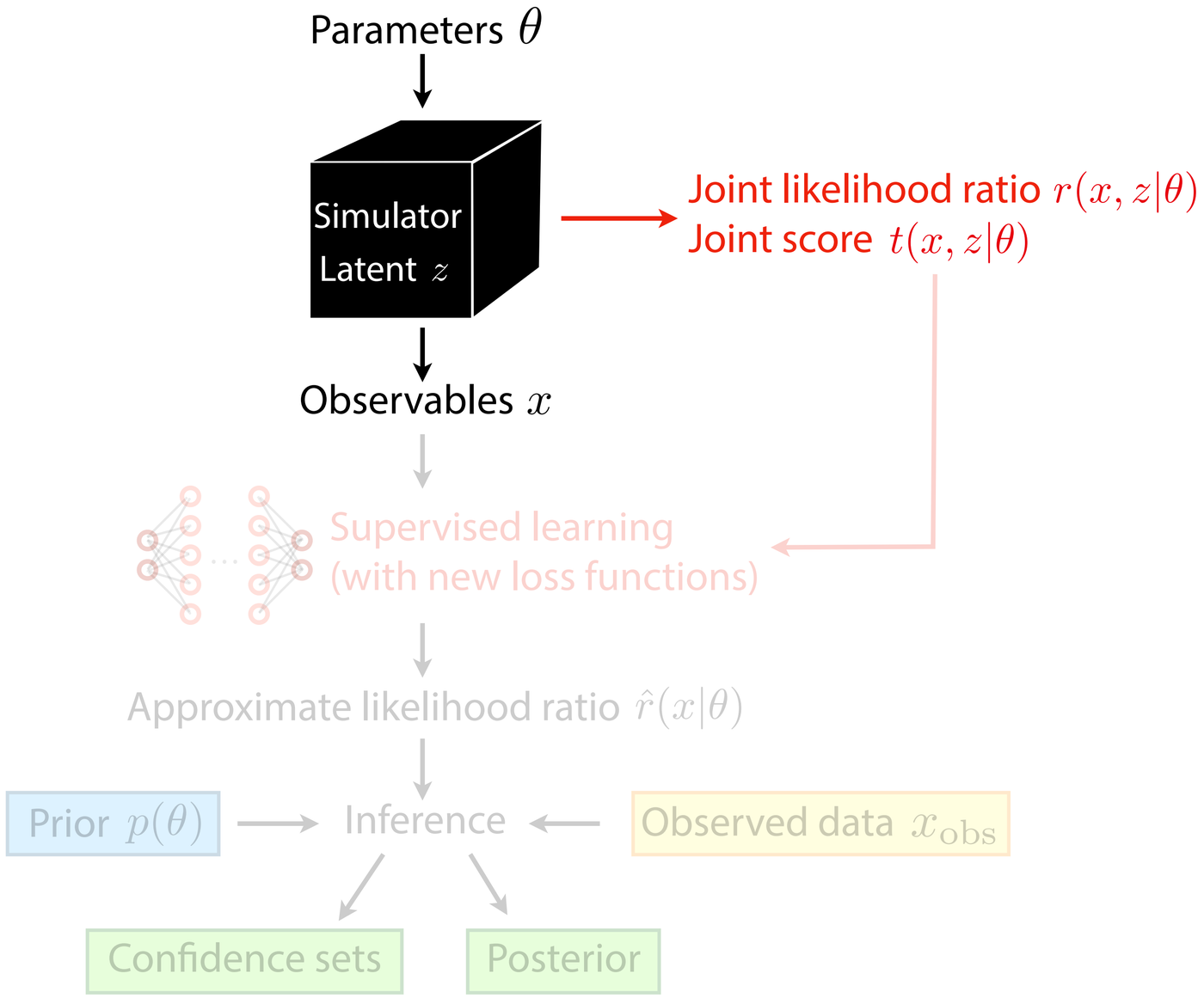
- Conditional neural density estimator (e.g. normalizing flow) as tractable surrogate for simulator likelihood
- Scales well to high-dimensional data (no compression to summary stats necessary)
- Amortized: After upfront simulation + training phase, inference is efficient for new data or prior
- Related alternative: learn posterior $\hat{p}(\theta|x_{\rm obs})$

[G. Papamakarios et al 1605.06376;
J.-M. Lueckmann et al 1711.01861]

Mining gold


Parameters θ
Simulator
Latent z
Joint likelihood ratio $r(x, z|\theta)$
Joint score $t(x, z|\theta)$
Observables x
Supervised learning
(with new loss functions)
Approximate likelihood ratio $\hat{r}(x|\theta)$
Prior $p(\theta)$
Inference
Observed data $x_{\rm obs}$
Confidence sets
Posterior

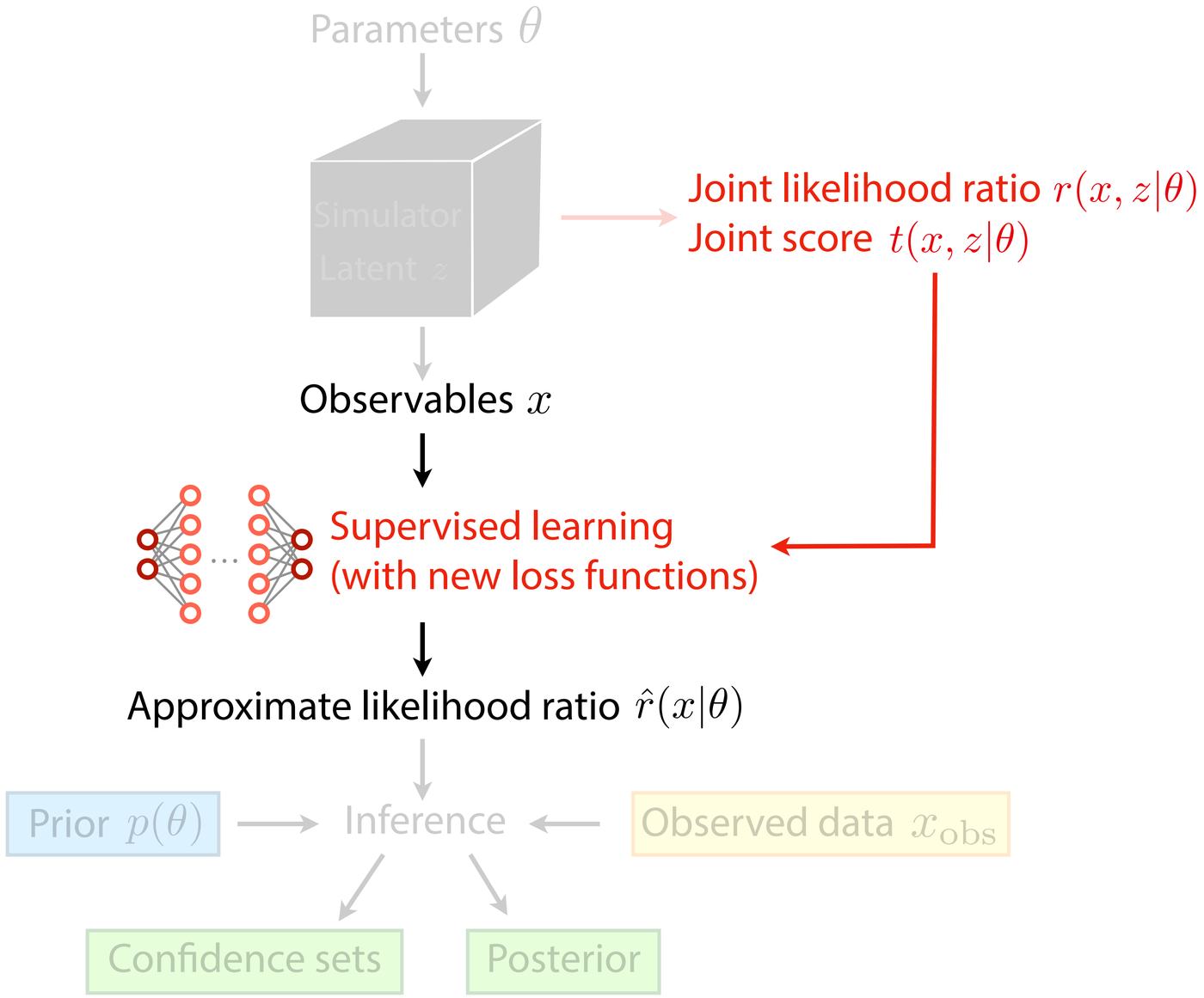
Step 1: Extracting more information from simulations



- For each simulated event, calculate
  - joint likelihood ratio $r(x, z|\theta) = \dfrac{p(x, z|\theta)}{p_{\rm ref}(x, z)}$
  - joint score $t(x, z|\theta) = \nabla_\theta \log p(x, z|\theta)$

(How much more or less likely would this simulated event (fixing all latent variables) be when changing the theory parameters?)

Step 2: Machine learning



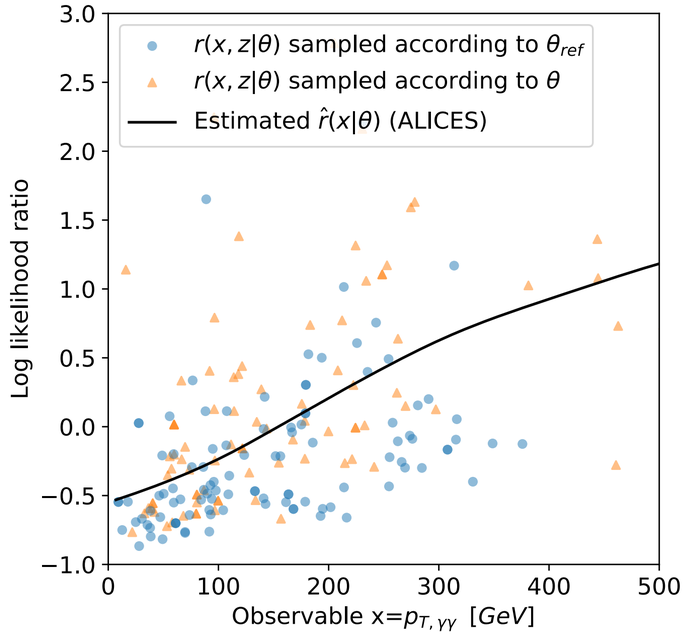
- Train a neural network $g(x, \theta)$ on loss functionals like

$$L[g] = \frac{1}{N} \sum_i |g(x_i, \theta_i) - r(x_i, z_i|\theta_i)|^2$$

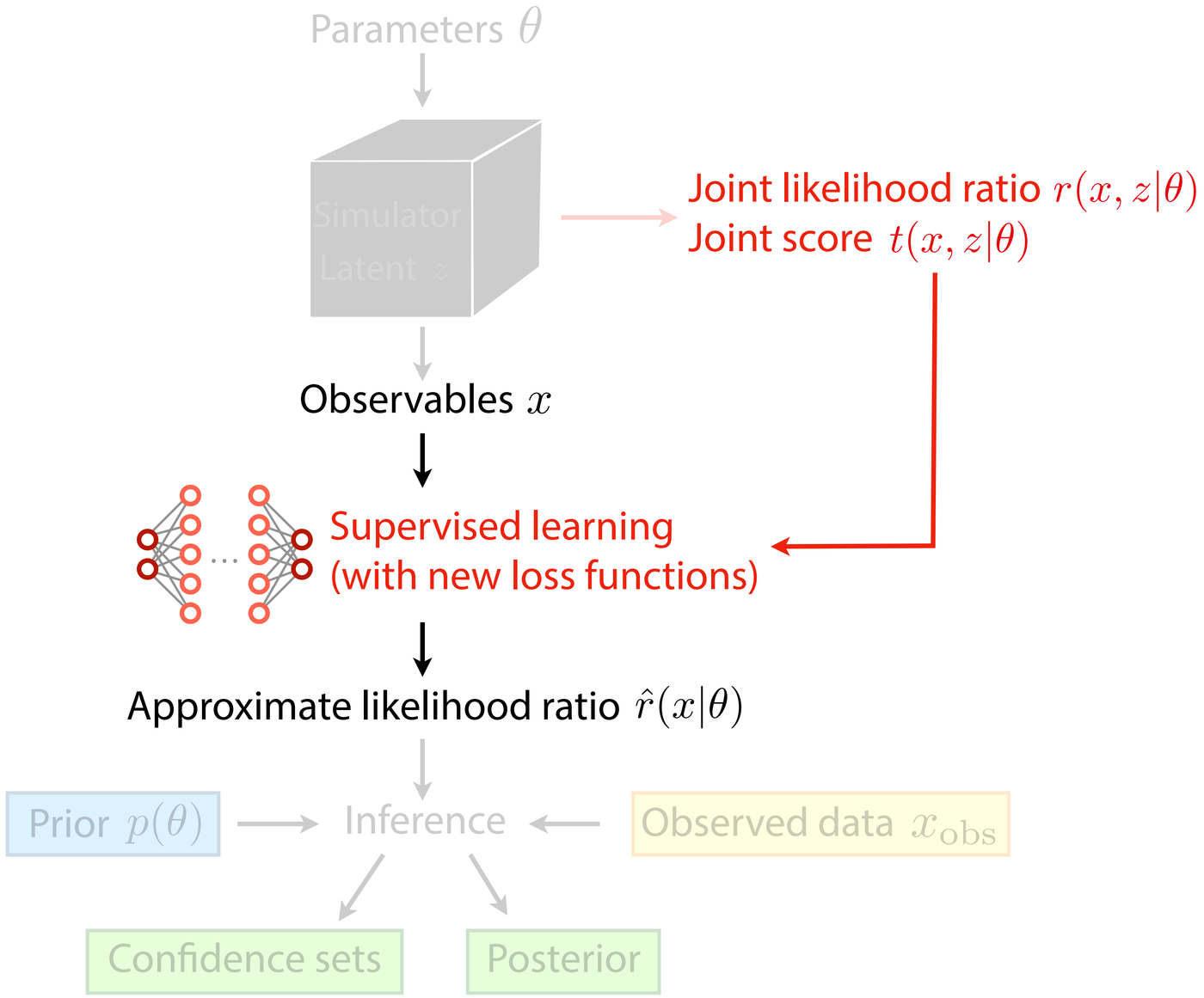
- The network will converge to

$$g(x, \theta) \to \arg\min L[g] = r(x|\theta) = \frac{p(x|\theta)}{p_{\rm ref}(x)} \quad !$$

(for sufficient training data, NN capacity, efficient optimization)



Step 2: Machine learning



- Train a neural network $g(x, \theta)$ on loss functionals like

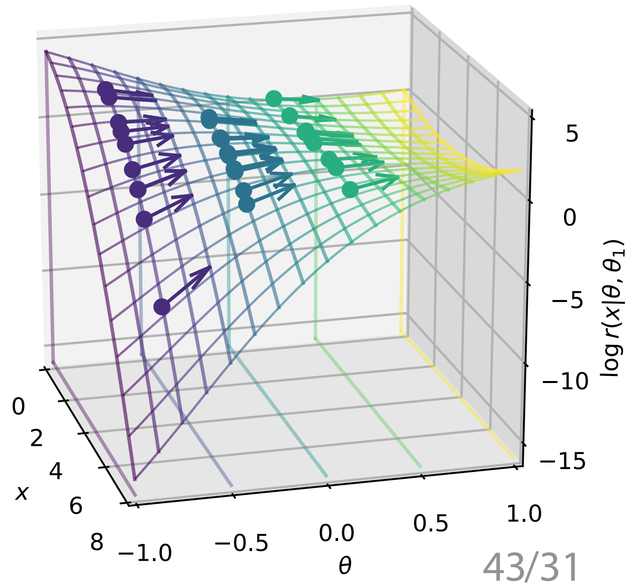
$$L[g] = \frac{1}{N} \sum_i |g(x_i, \theta_i) - r(x_i, z_i|\theta_i)|^2$$

- The network will converge to

$$g(x, \theta) \to \arg\min L[g] = r(x|\theta) = \frac{p(x|\theta)}{p_{\rm ref}(x)} \quad !$$

(for sufficient training data, NN capacity, efficient optimization)

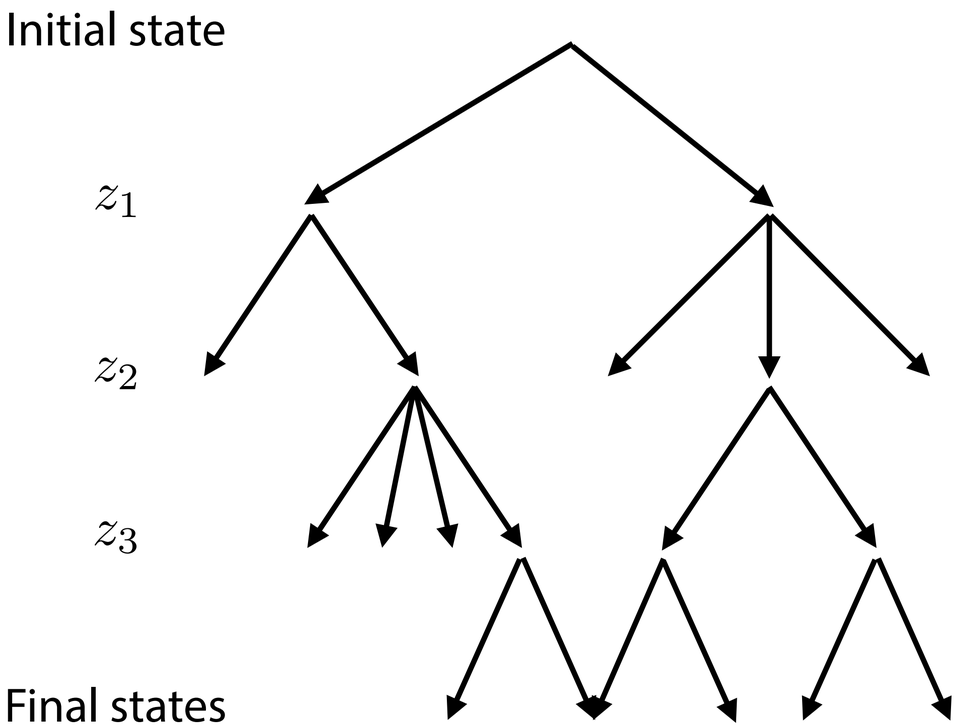
- RASCAL: Joint score adds gradient information ⇒ three orthogonal pieces of information



Mining gold from any simulation

- Computer simulation typically evolve along a tree-like structure of successive random branchings
- The probabilities of each branching $p_i(z_i|z_{i-1},\theta)$ are often clearly defined in the code:

```
if random() > 0.1 + 2.5 * model_parameter:
    do_one_thing()
else:
    do_another_thing()
```



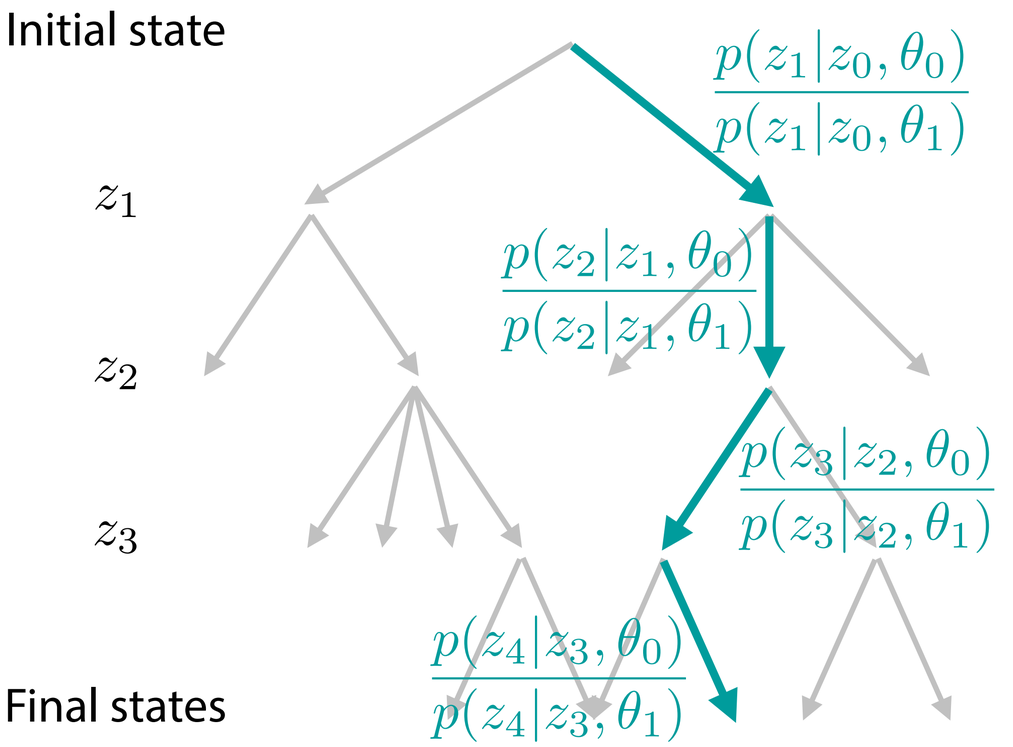
Mining gold from any simulation

- Computer simulation typically evolve along a tree-like structure of successive random branchings
- The probabilities of each branching $p_i(z_i|z_{i-1},\theta)$ are often clearly defined in the code:

```
if random() > 0.1 + 2.5 * model_parameter:
    do_one_thing()
else:
    do_another_thing()
```

- For each run of the simulator, we can calculate the probability of the chosen path for different values of the parameters, and the "joint likelihood ratio":

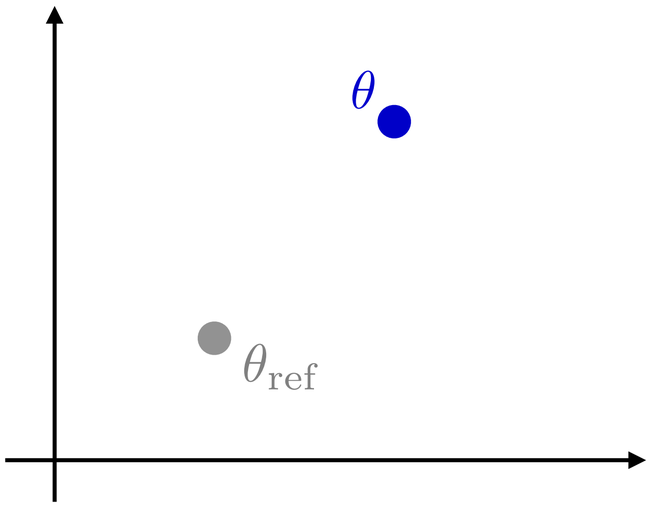
$$r(x,z|\theta_0,\theta_1) = \frac{p(x,z|\theta_0)}{p(x,z|\theta_1)} = \prod_i \frac{p(z_i|z_{i-1},\theta_0)}{p(z_i|z_{i-1},\theta_1)}$$



Frequentist inference

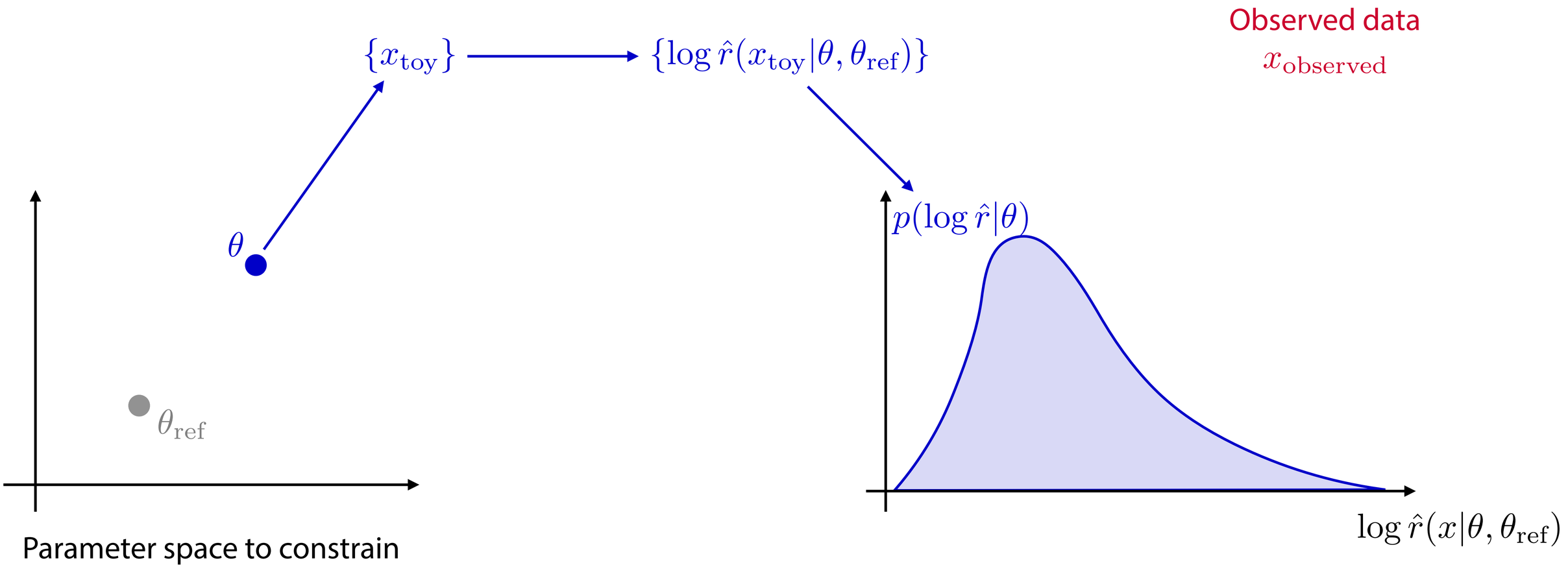
Observed data

x_{observed}

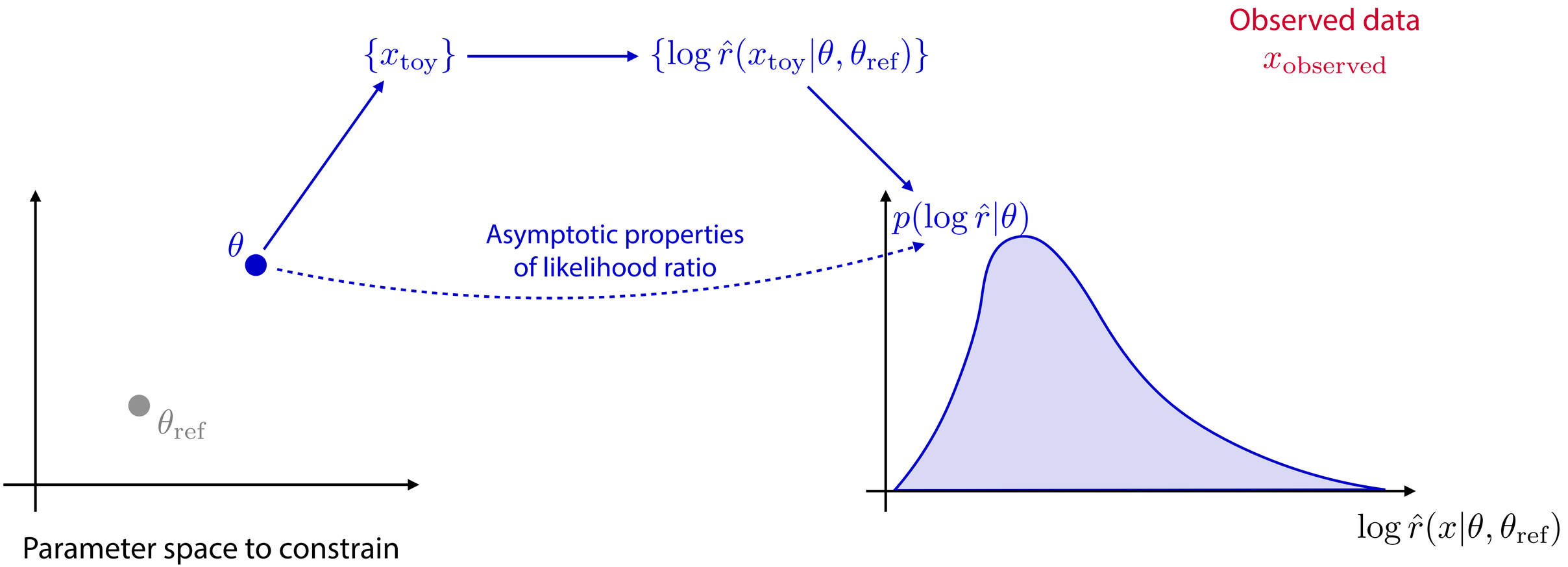


Parameter space to constrain

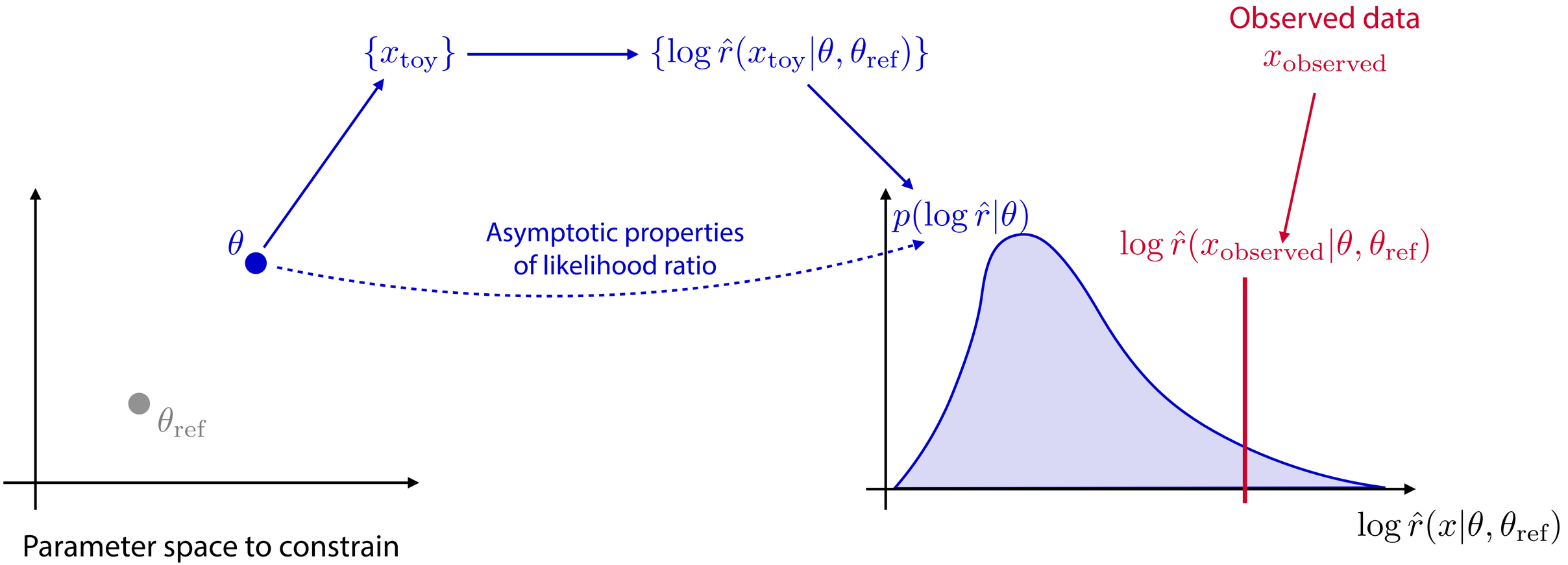
Frequentist inference



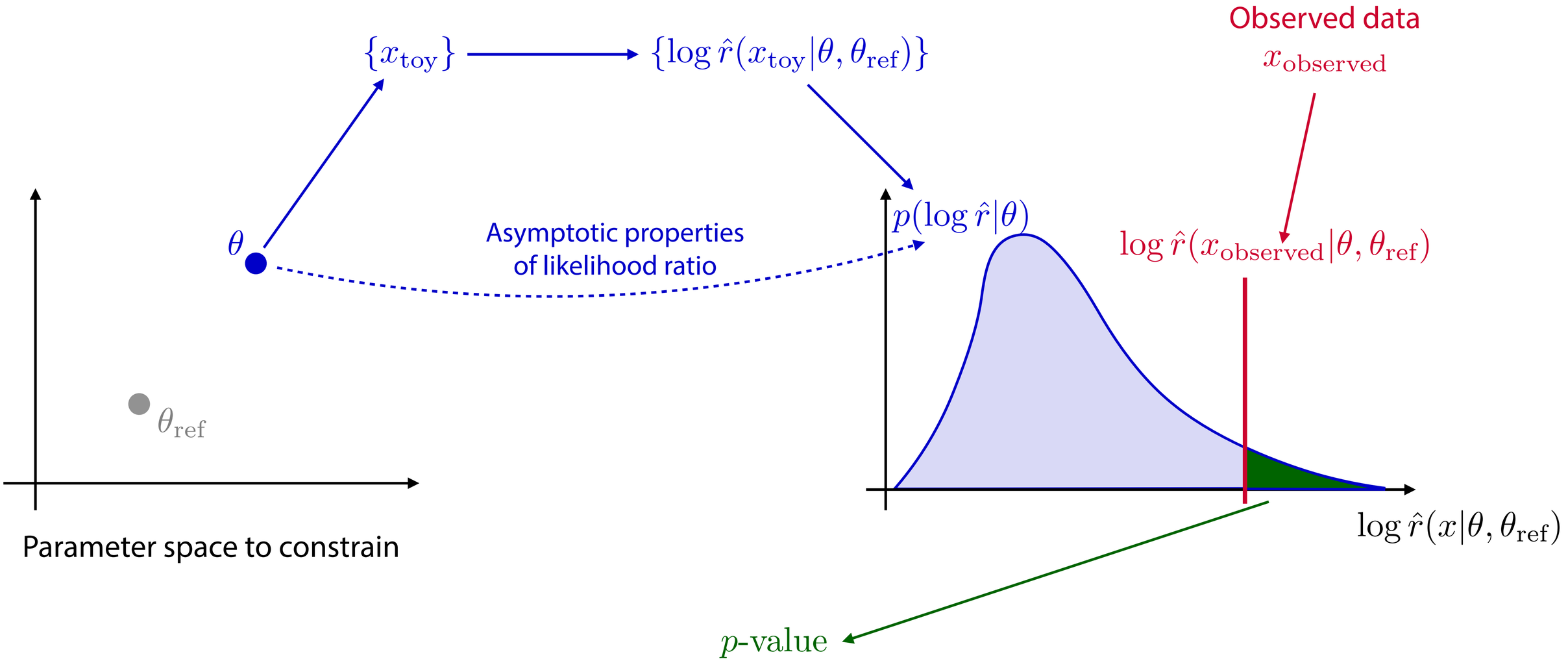
Frequentist inference



Frequentist inference



Frequentist inference



Frequentist inference