Simulation-Based Inference

Aishik Ghosh

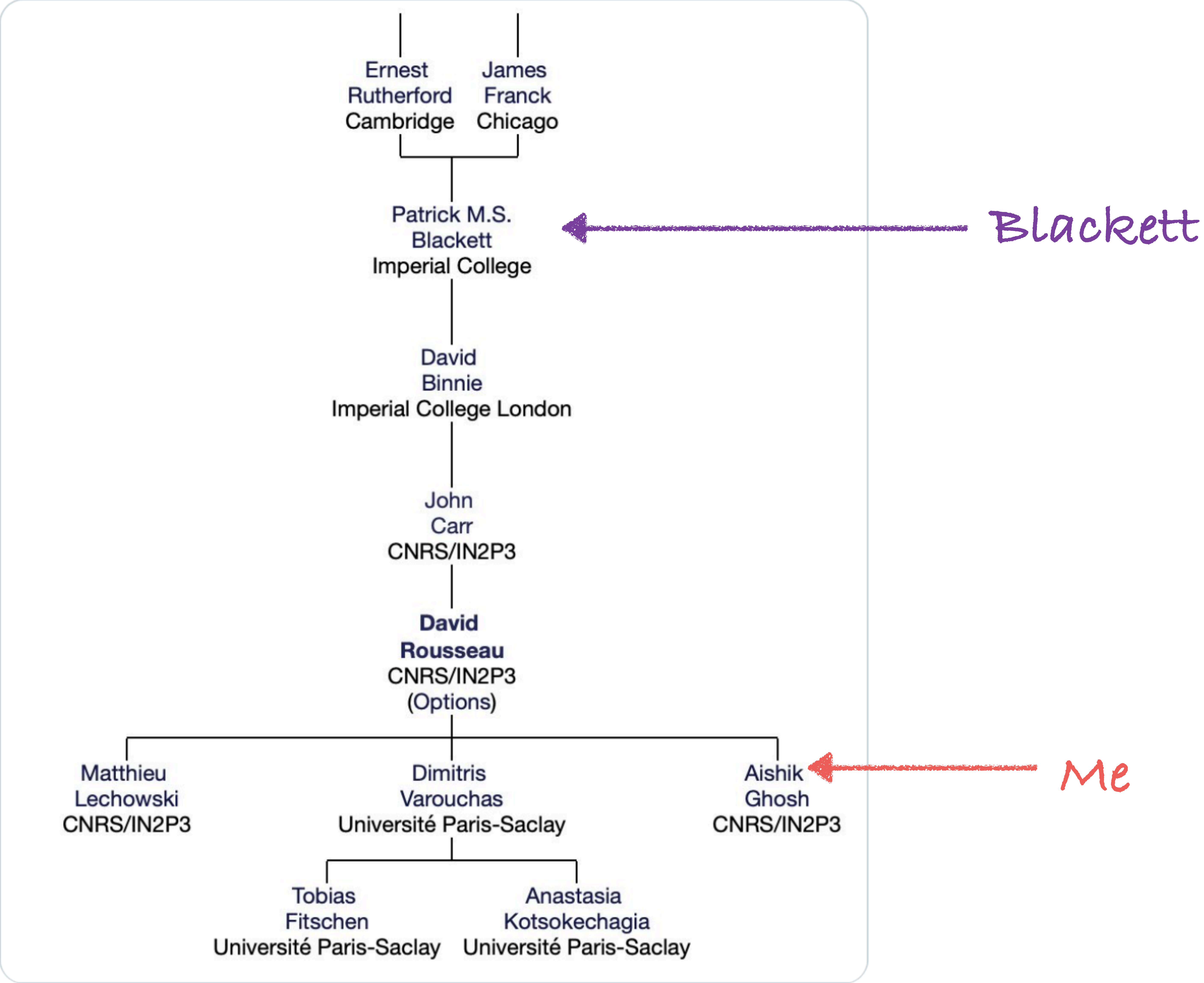
PHY-STAT, London

11 September 2024

@Aishik_Ghosh_

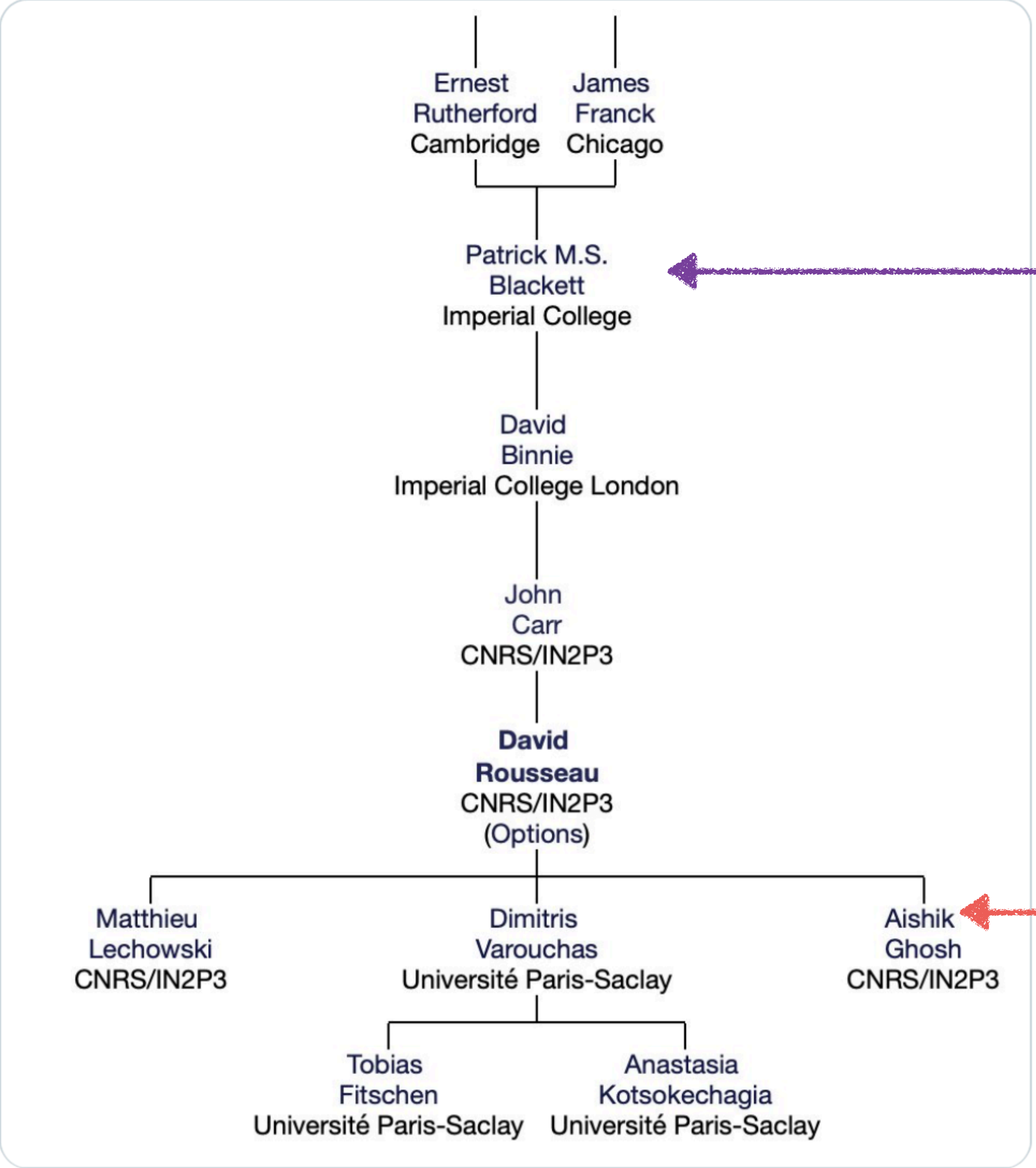
David Rousseau
@dhpmrou

I just learned that I am a great-great-grandson of Ernest Rutherford !!!
Not quite sure what to do with this.
#academiclife academictree.org/physics/tree.p...

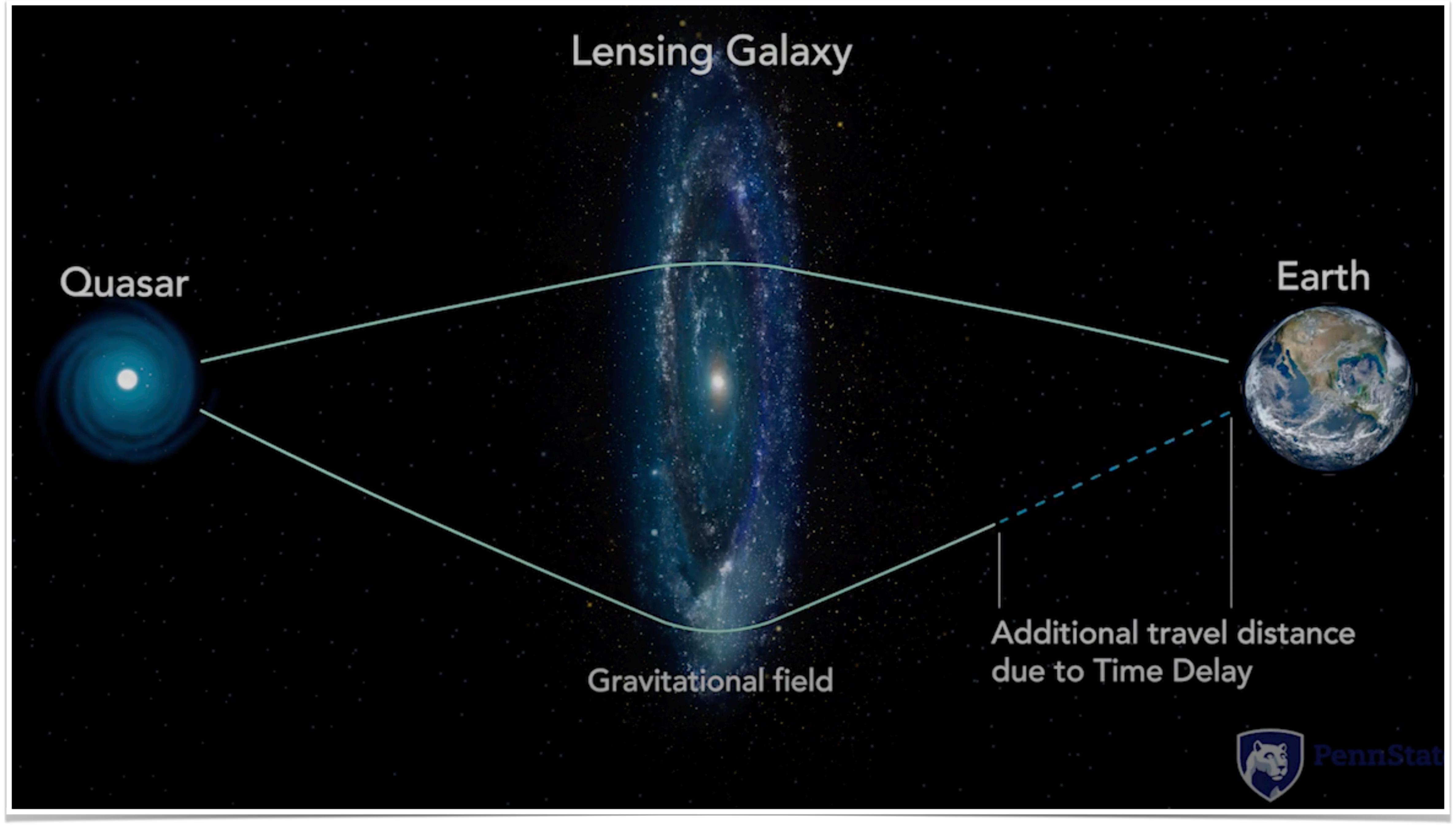


David Rousseau
@dhpmrou

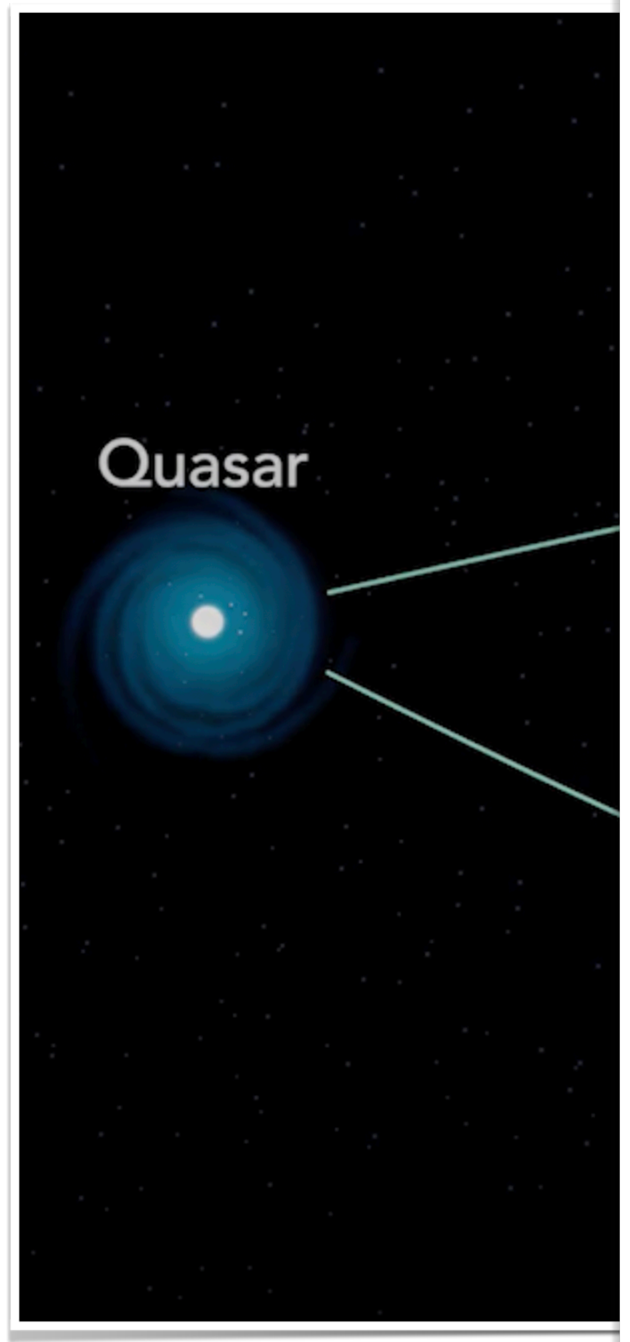
I just learned that I am a great-great-grandson of Ernest Rutherford !!!
Not quite sure what to do with this.
#academiclifeacademictree.org/physics/tree.p...




Lensing Galaxy
Quasar
Earth
Gravitational field
Additional travel distance
due to Time Delay

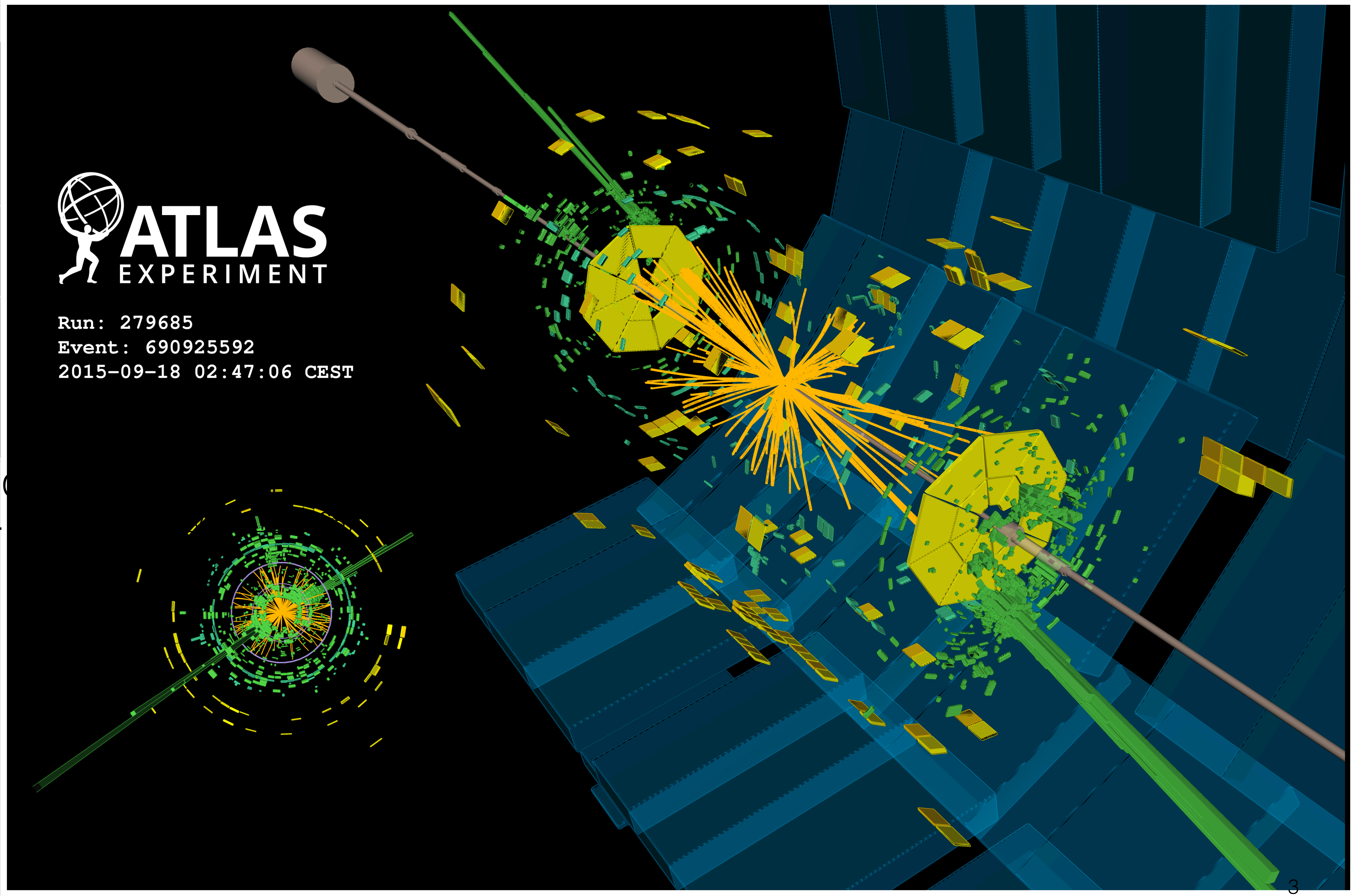


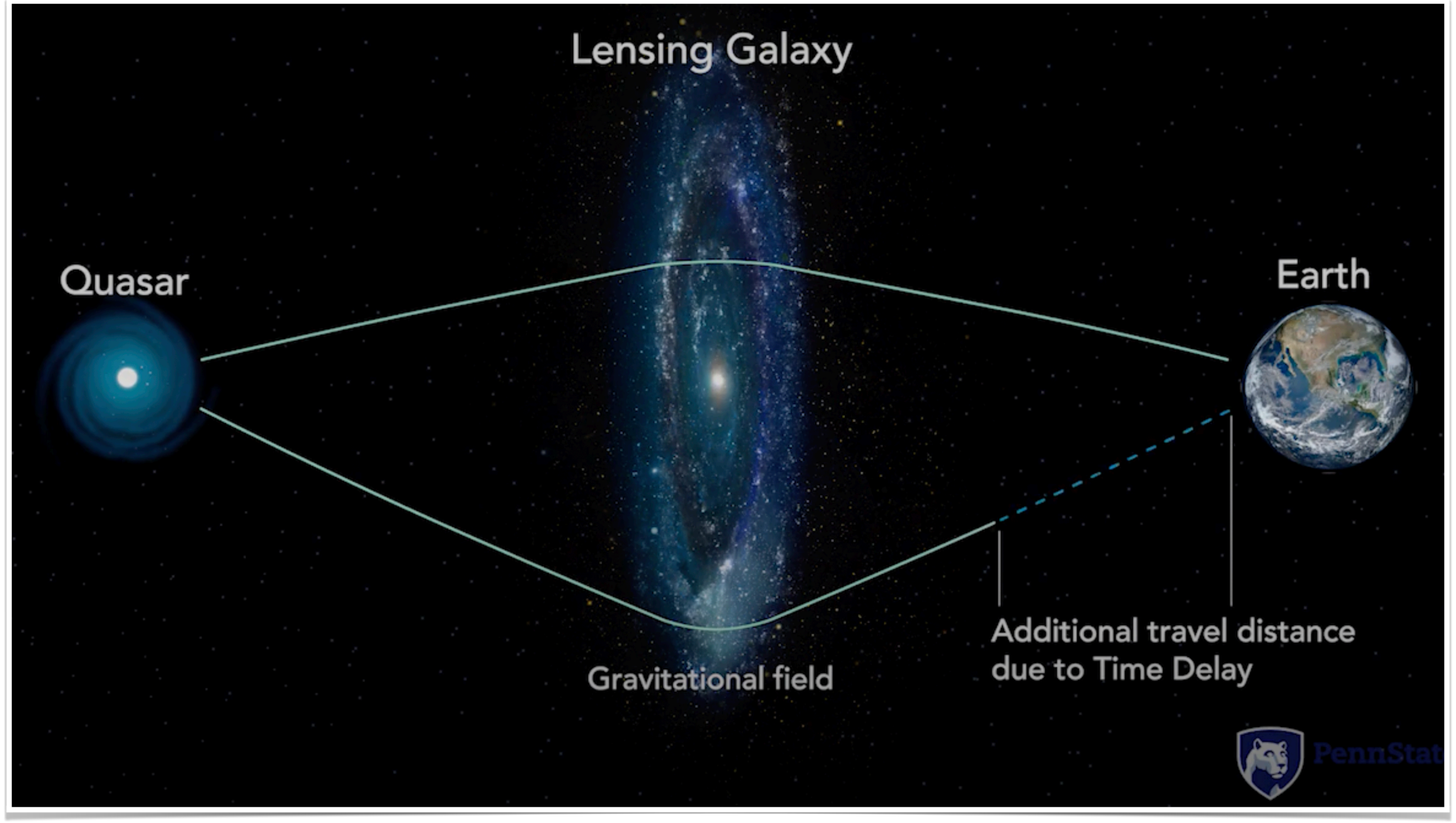
Gravitational lensing
(Eg. Adam et al. arXiv:2301.04168)


Quasar

(

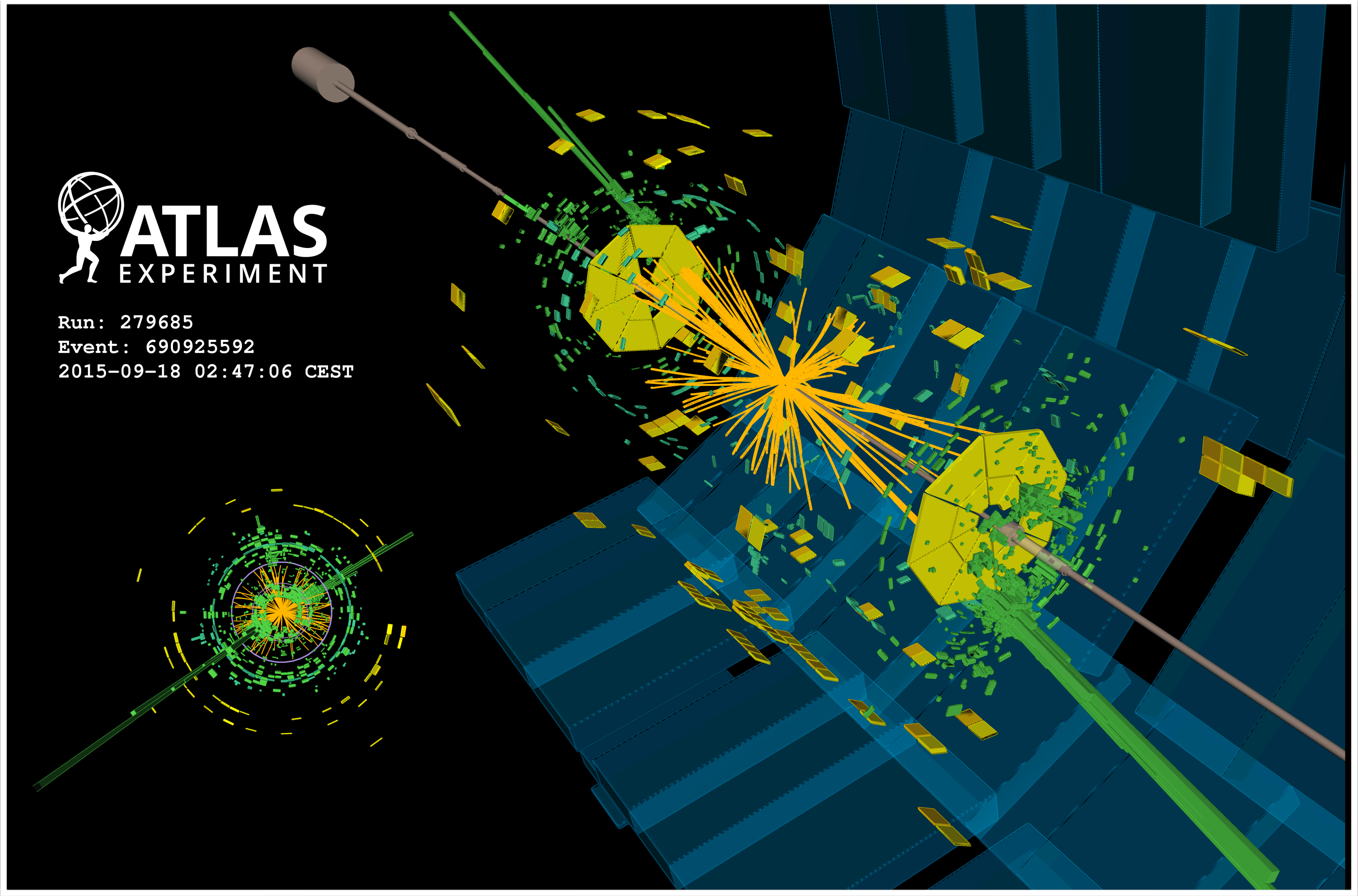
(Eg.


ATLAS
EXPERIMENT
Run: 279685
Event: 690925592
2015-09-18 02:47:06 CEST



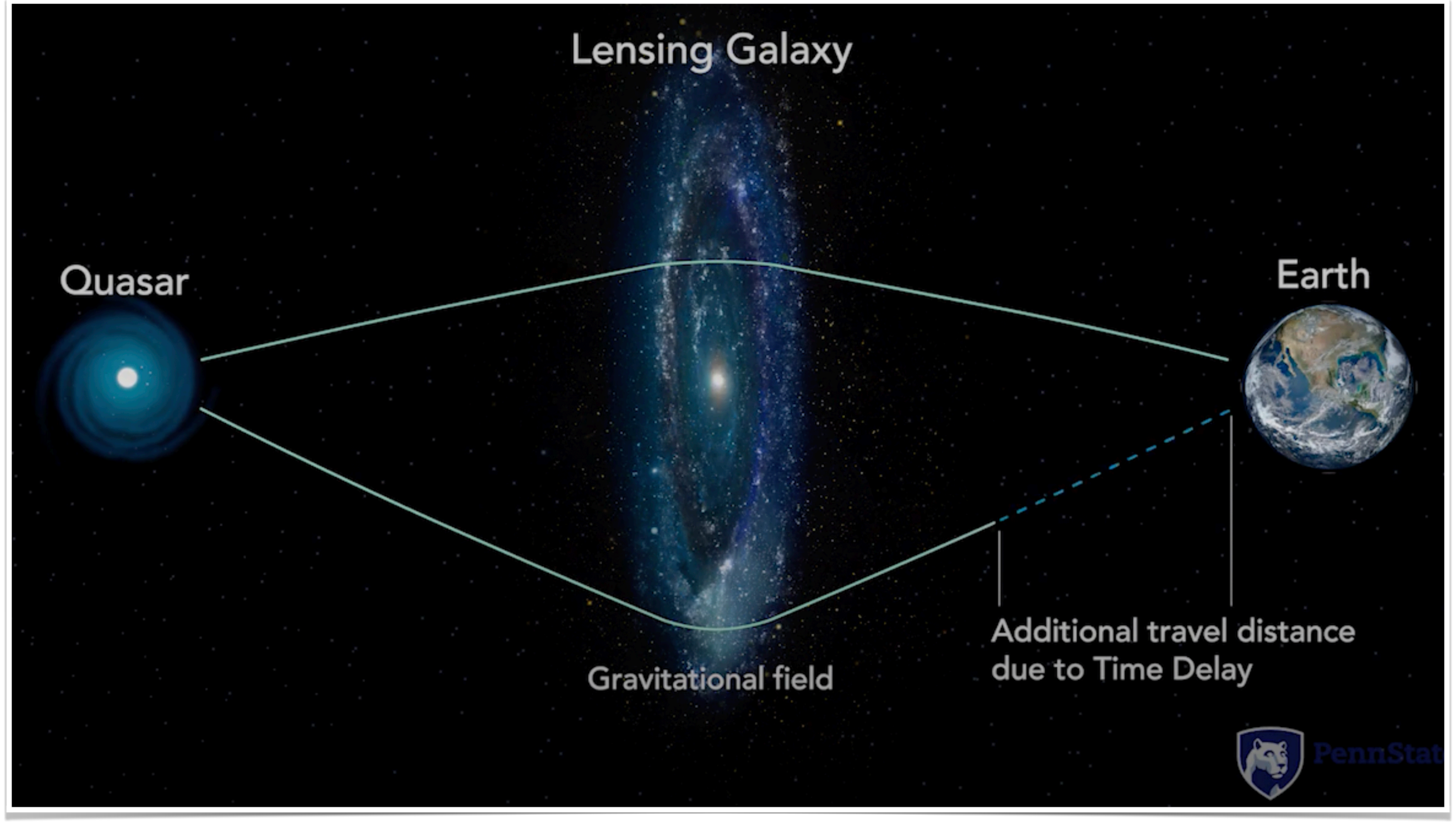
Gravitational lensing

(Eg. Adam et al. arXiv:2301.04168)

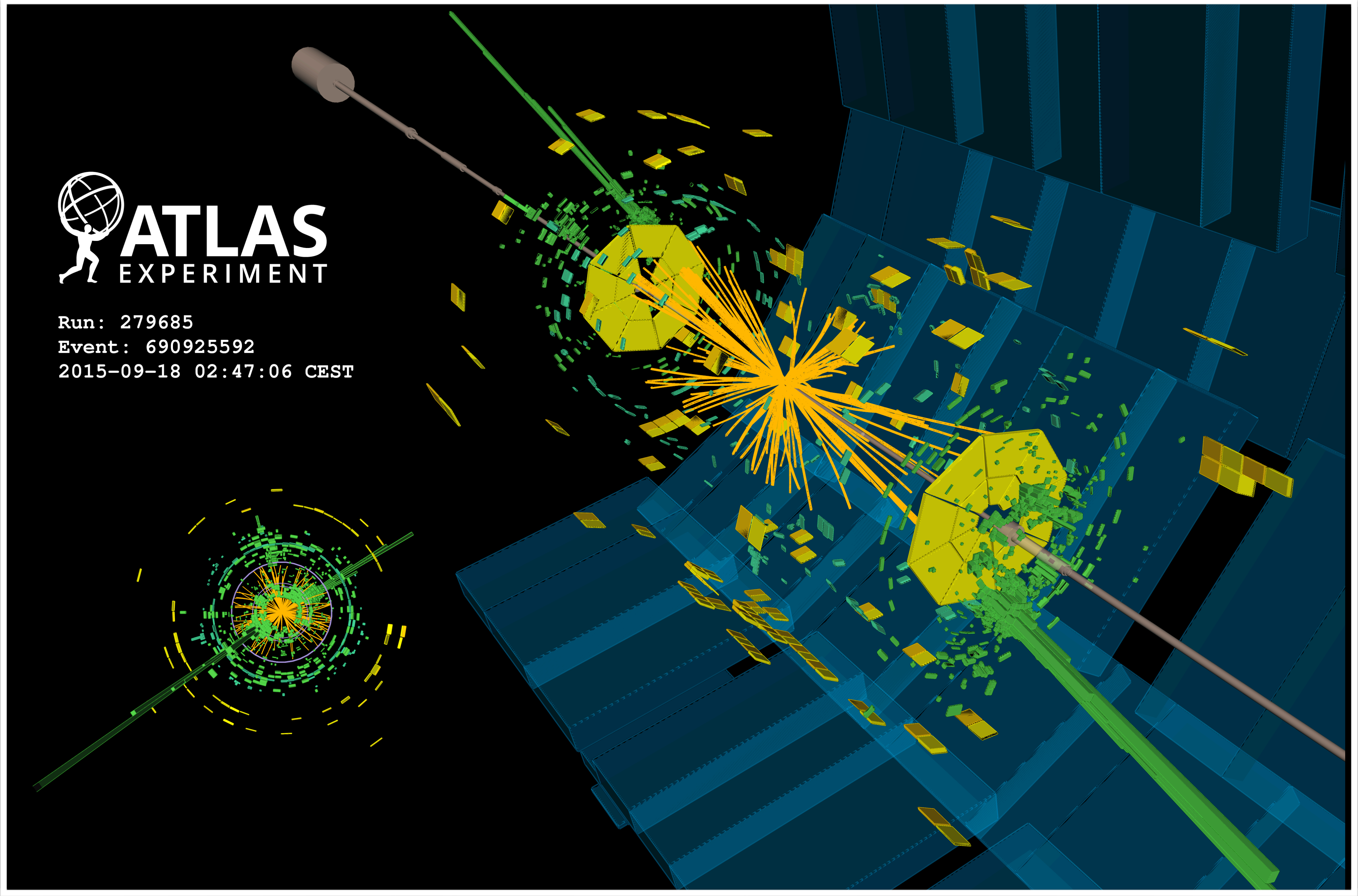


Higgs physics

(Eg. Cranmer et al arXiv:1506.02169, Ghosh & Rousseau al hal-02971995v3)

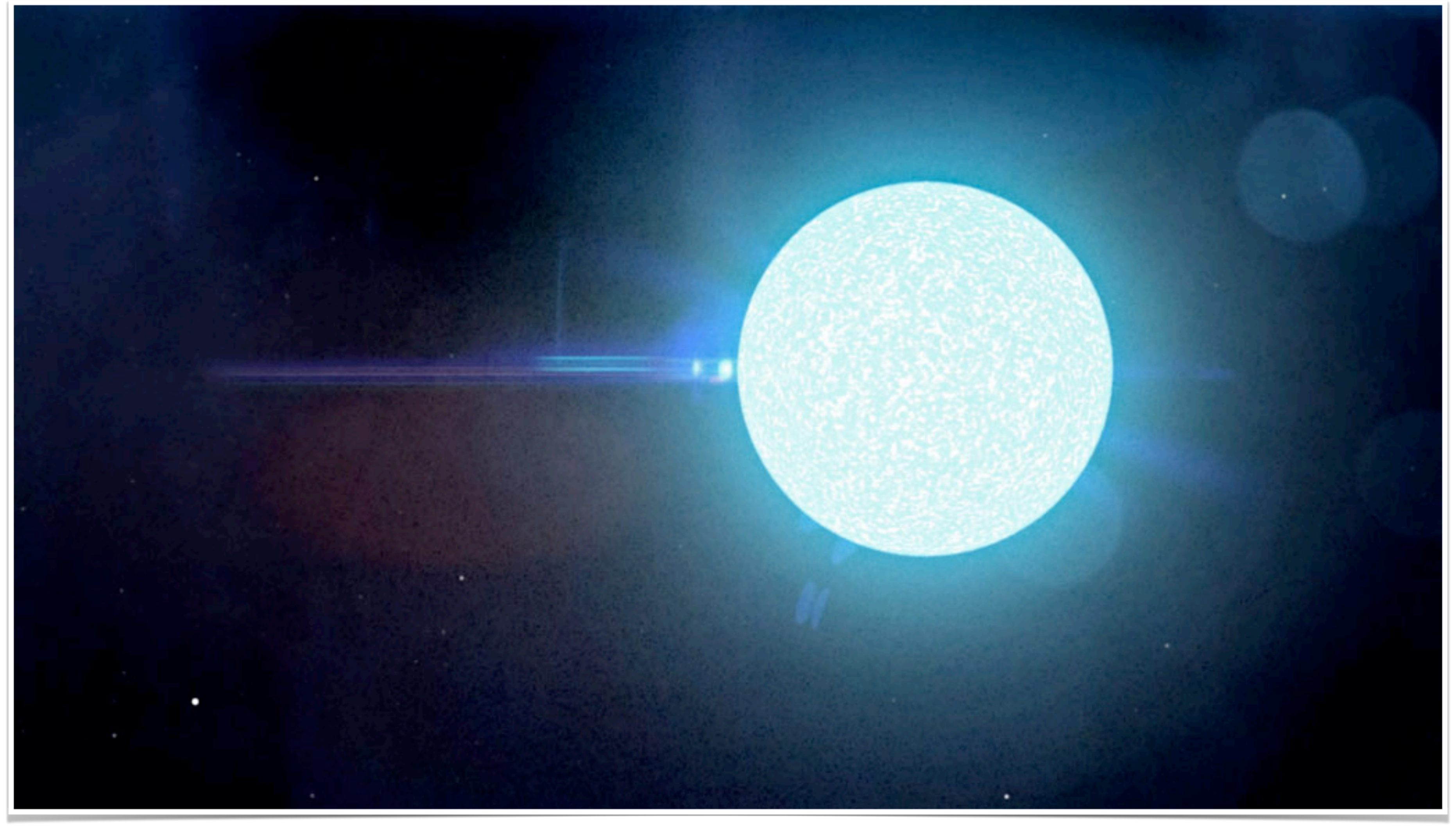


Gravitational lensing
(Eg. Adam et al. arXiv:2301.04168)



Higgs physics
(Eg. Cranmer et al arXiv:1506.02169, Ghosh & Rousseau al hal-02971995v3)

Neutron star astro/nuclear physics
(Eg. Brandes et al, incl Ghosh, arXiv:2403.00287,
Mishra-Sharma & Cranmer, arXiv: 2110.06931)



Simulation-Based Inference in PHY-STAT

Workshop in Munich this summer

Simulation-Based Inference in PHY-STAT

Workshop in Munich this summer

11:45	Simulation-based machine learning for gravitational-wave analysis	45m
17:10	Anomaly aware machine learning for dark matter direct detection at the DARWIN experiment	25m
14:25	lsbi: linear simulation based inference	25m
11:15	pop-cosmos: an interpretable generative model for the galaxy population over cosmic time	30m
17:35	Feldman-Cousins' ML Cousin	25m
11:00	Cosmology and machine learning	45m
18:09	Advanced techniques for Simulation Based Inference in collider physics	1m
18:10	SBI for wide field weak lensing	1m
18:13	Accounting for Selection Effects in Supernova Cosmology with Simulation-Based Inference and Hierarchical Bayesian Modelling	1m
11:00	Simulation-based Inference (SBI)	45m

This workshop alone !

Simulation-Based Inference in PHY-STAT

11:45	Simulation-based machine learning for gravitational-wave analysis	45m
17:10	Anomaly aware machine learning for dark matter direct detection at the DARWIN experiment	25m
14:25	lsbi: linear simulation based inference	25m
11:15	pop-cosmos: an interpretable generative model for the galaxy population over cosmic time	30m
17:35	Feldman-Cousins' ML Cousin	25m
11:00	Cosmology and machine learning	45m
18:09	Advanced techniques for Simulation Based Inference in collider physics	1m
18:10	SBI for wide field weak lensing	1m
18:13	Accounting for Selection Effects in Supernova Cosmology with Simulation-Based Inference and Hierarchical Bayesian Modelling	1m
11:00	Simulation-based Inference (SBI)	45m

This talk: Statistical questions, with bias towards frequentist LHC applications

Workshop in Munich this summer

This workshop alone !

Testing theories / hypothesis

$$p(\text{theory} \,|\, \text{data}) = \frac{p(\text{data} \,|\, \text{theory}) p(\text{theory})}{p(\text{data})}$$

Testing theories / hypothesis

What we all
want
(Posterior)

$$p(\text{theory}\,|\,\text{data}) = \frac{p(\text{data}\,|\,\text{theory})p(\text{theory})}{p(\text{data})}$$

Testing theories / hypothesis

What we all want (Posterior)

Likelihood

$$p(\mathrm{theory}\,|\,\mathrm{data}) = \frac{p(\mathrm{data}\,|\,\mathrm{theory})p(\mathrm{theory})}{p(\mathrm{data})}$$

Testing theories / hypothesis

What we all want (Posterior)

Likelihood

Prior

$$p(\text{theory} \mid \text{data}) = \frac{p(\text{data} \mid \text{theory})\, p(\text{theory})}{p(\text{data})}$$

Evidence

Testing theories / hypothesis

What we all want (Posterior)

Likelihood

Prior

$$p(\text{theory}\,|\,\text{data}) = \frac{p(\text{data}\,|\,\text{theory})p(\text{theory})}{p(\text{data})}$$

Evidence

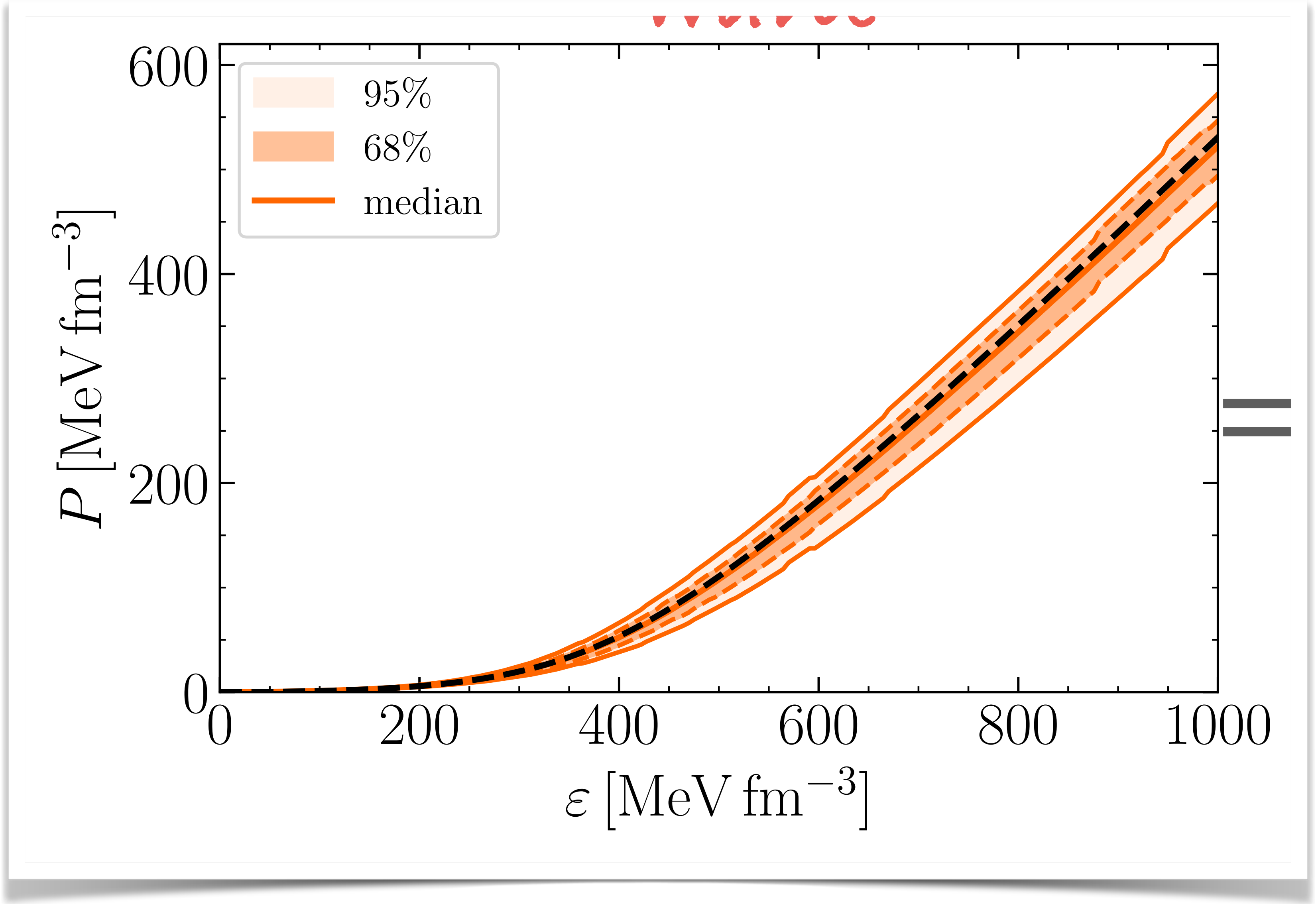
Bayesian statistics: Assign a prior, then calculate the posterior → Credible intervals

Frequentist statistics: No prior, so no posterior. Statements about confidence in our analysis method → Confidence intervals

What we both like: **Likelihoods**

Testing theories / hypothesis

What we all want



Likelihood

Prior

$$= \frac{p(\text{data}\,|\,\text{theory})\,p(\text{theory})}{p(\text{data})}$$

Evidence

Bayesian statistics: Assign a prior, then calculate the posterior → Credible intervals

Frequentist statistics: No prior, so no posterior. Statements about confidence in our analysis method → Confidence intervals

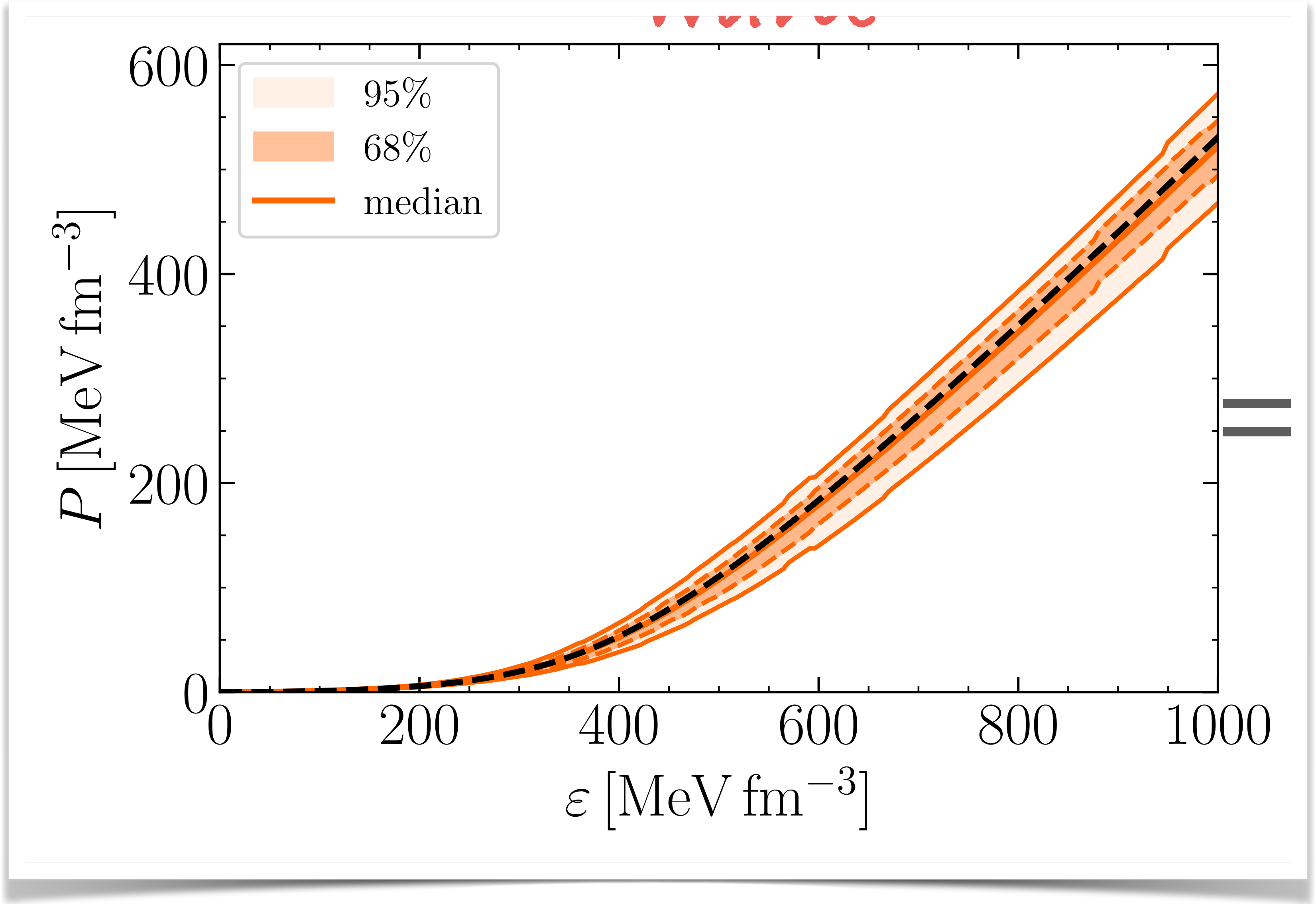
What we both like: **Likelihoods**

Testing theories / hypothesis

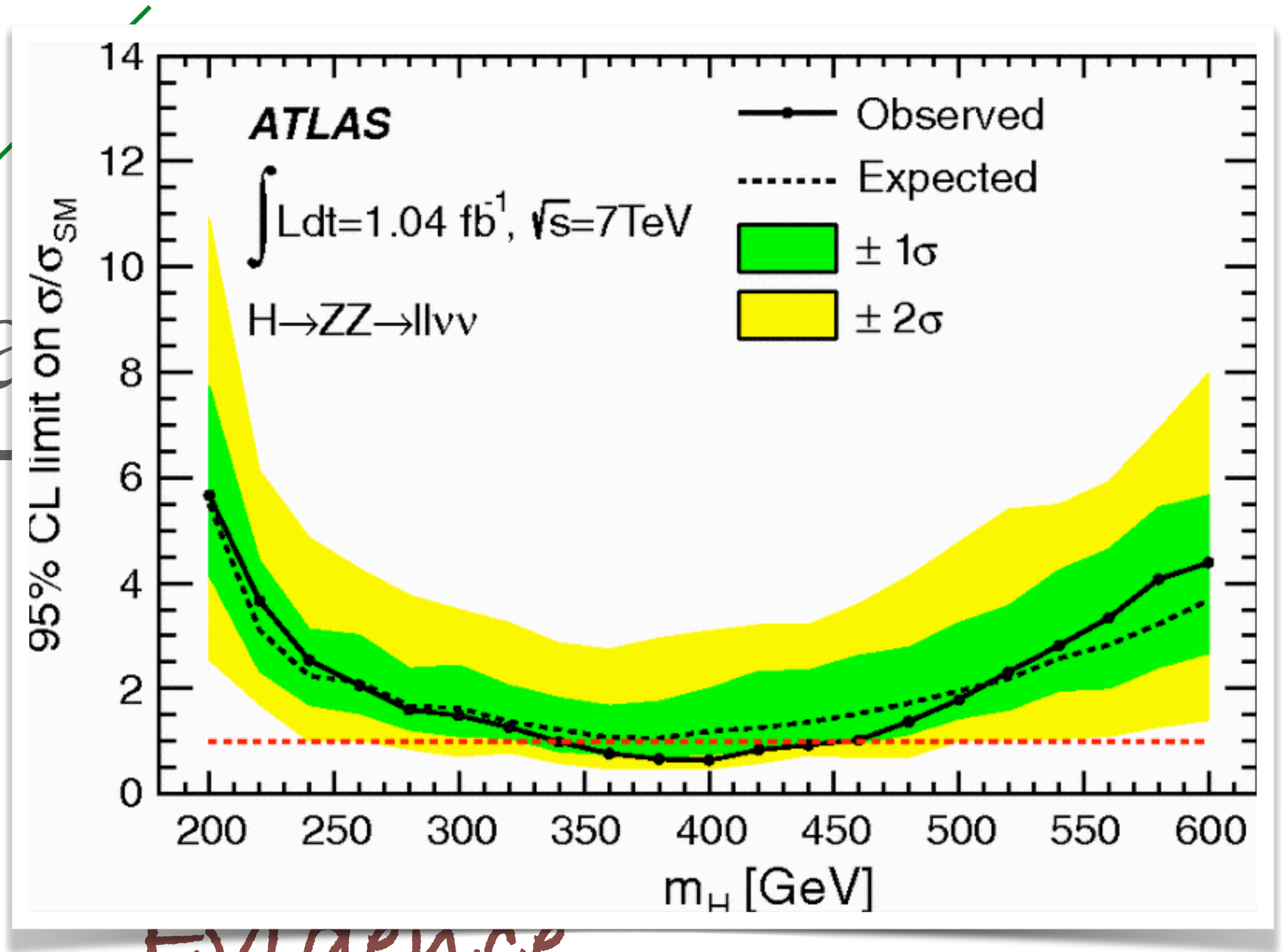
What we all want

Likelihood

Prior



$= \; p(da$



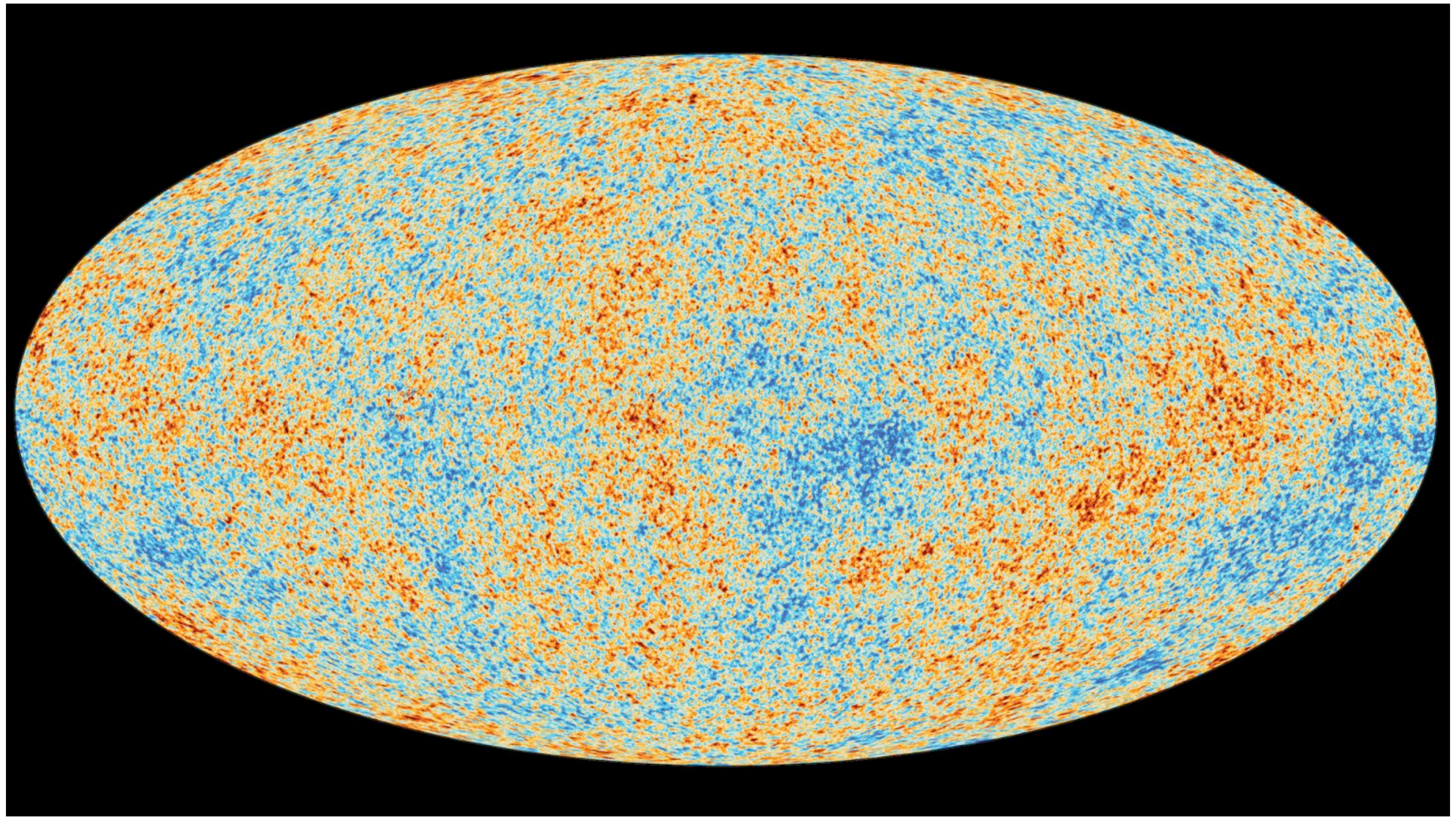
Evidence

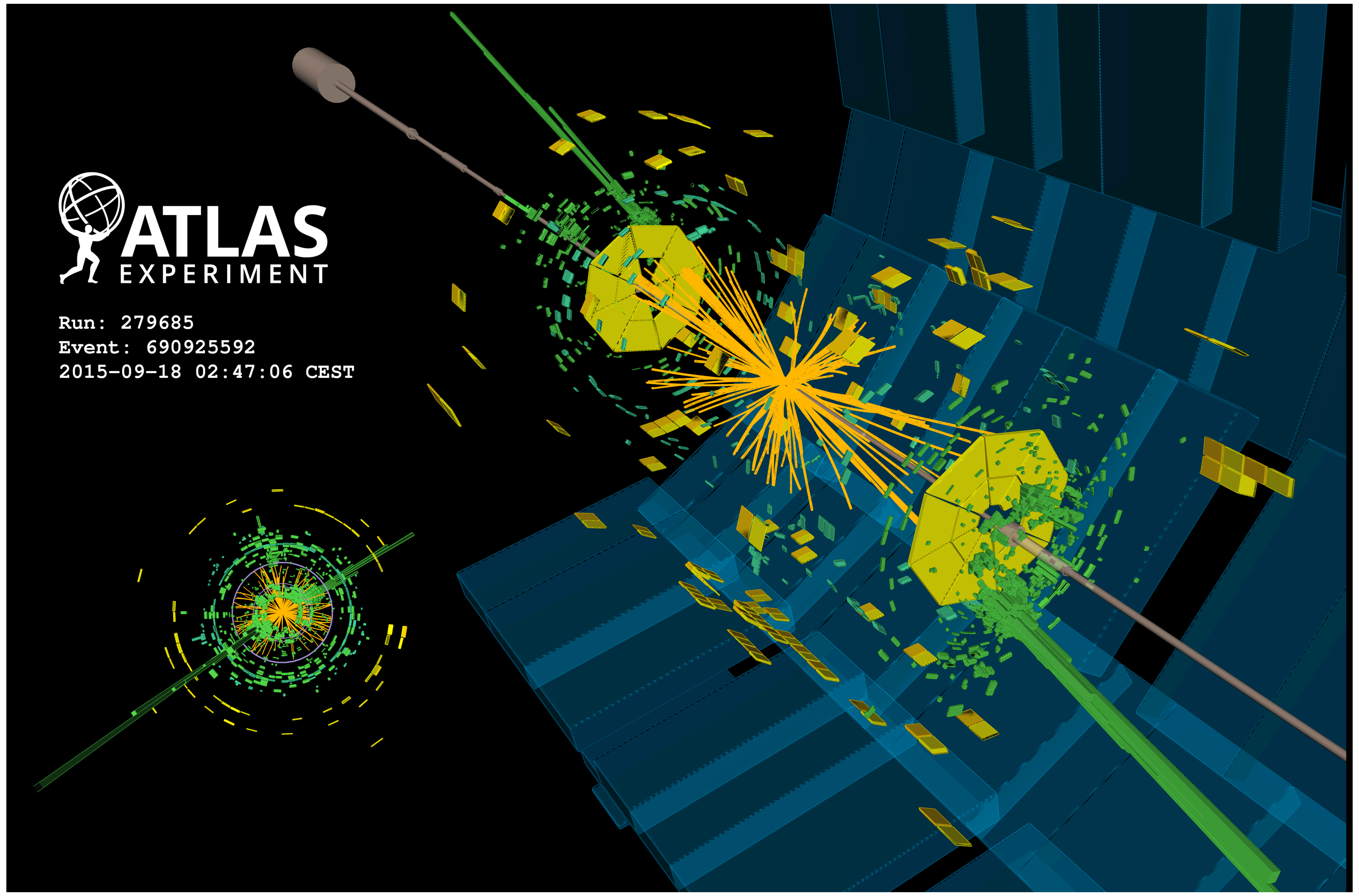
Bayesian statistics: Assign a prior, then calculate the posterior → Credible intervals

Frequentist statistics: No prior, so no posterior. Statements about confidence in our analysis method → Confidence intervals

What we both like: **Likelihoods**

How to obtain the likelihood?

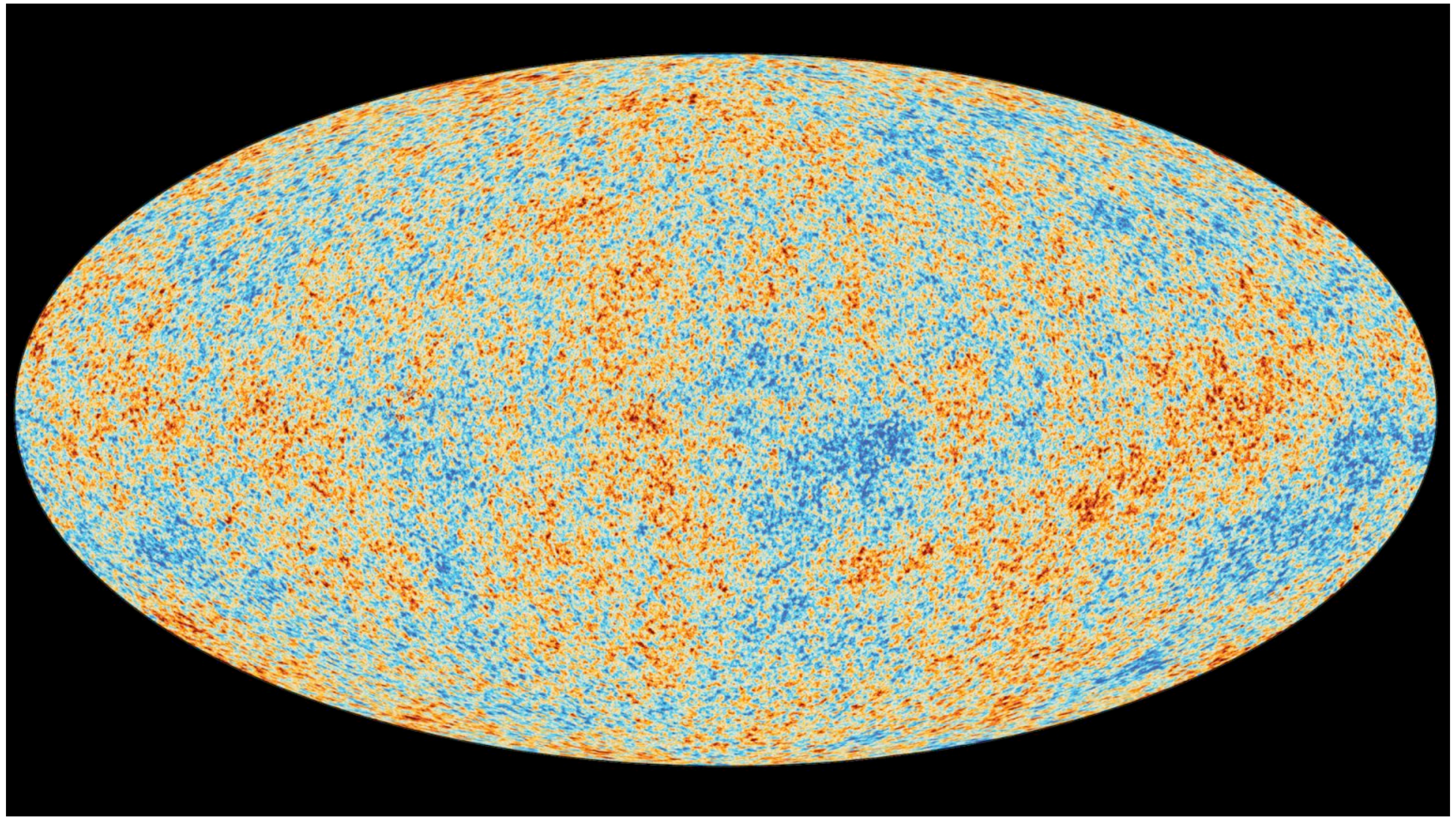


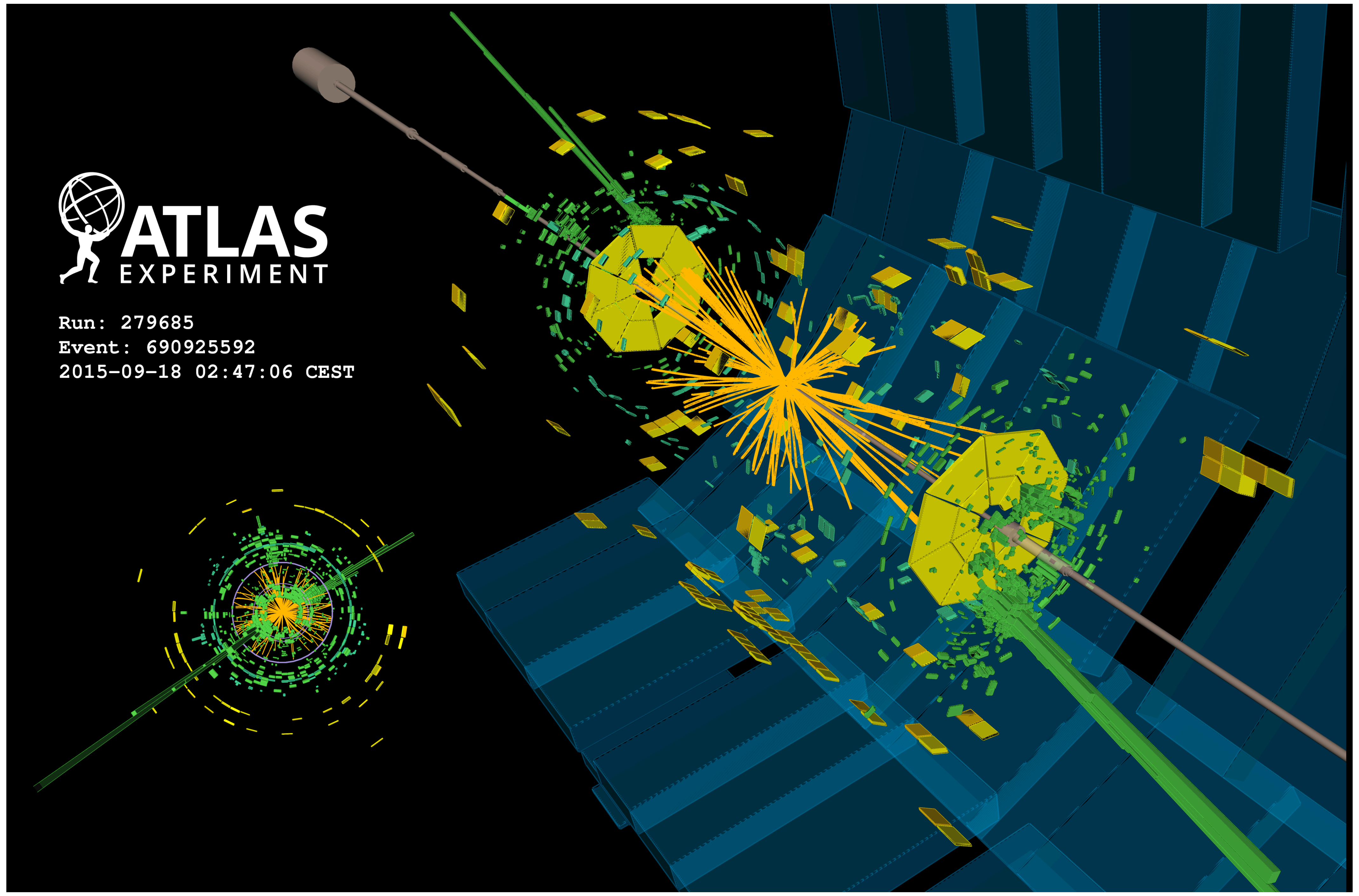


We got our data from observation / experiment

Now how do we calculate $p(\mathrm{data}\,|\,\mathrm{theory})$?

How to obtain the likelihood?





We got our data from observation / experiment

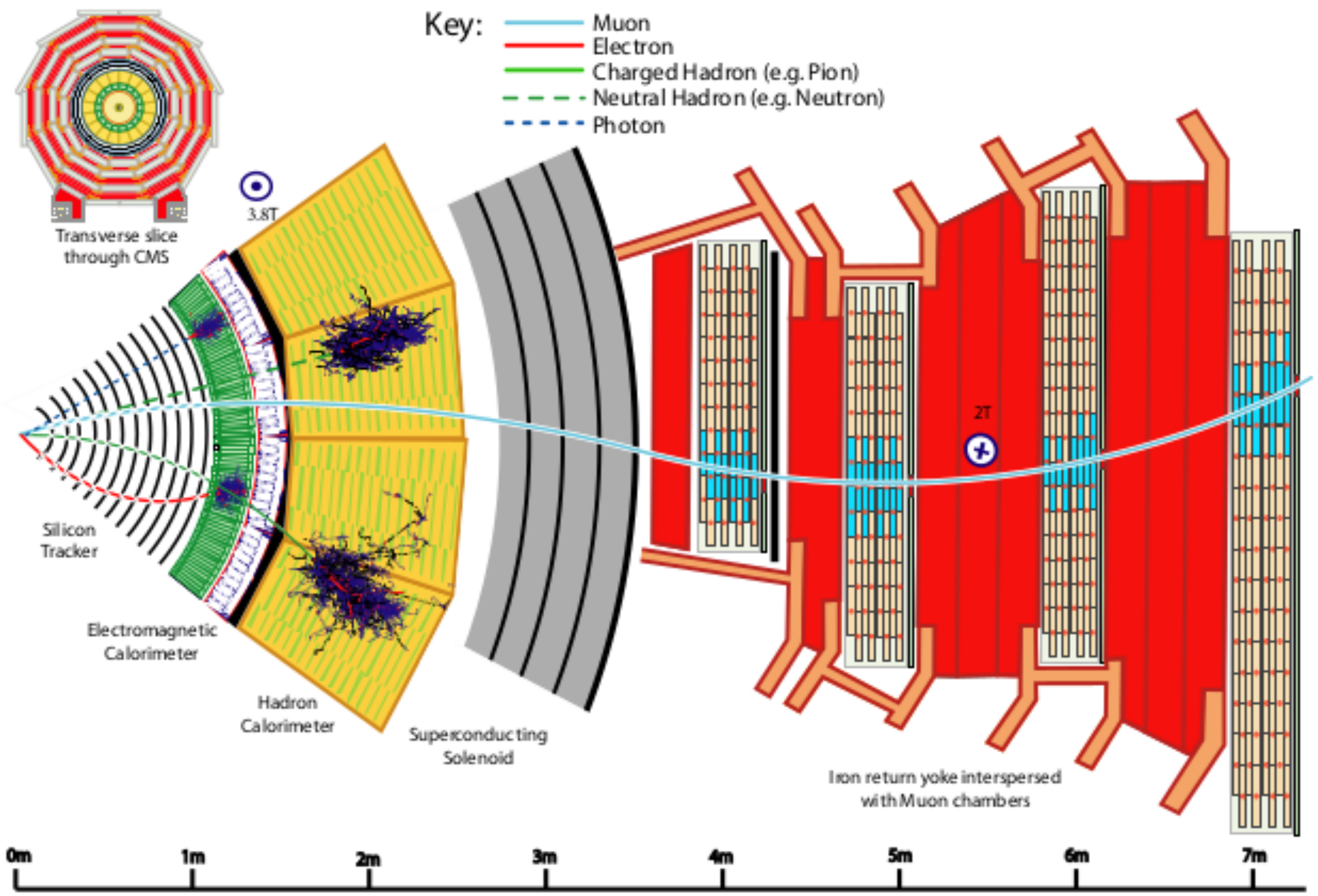
Now how do we calculate $p(\text{data} \mid \text{theory})$?

A known analytical formula

- Often uses approximations
- Restrict data space to where it works

Most domains in science have a **detailed forward simulator**

Forward simulator but inverse problem



Intractable:

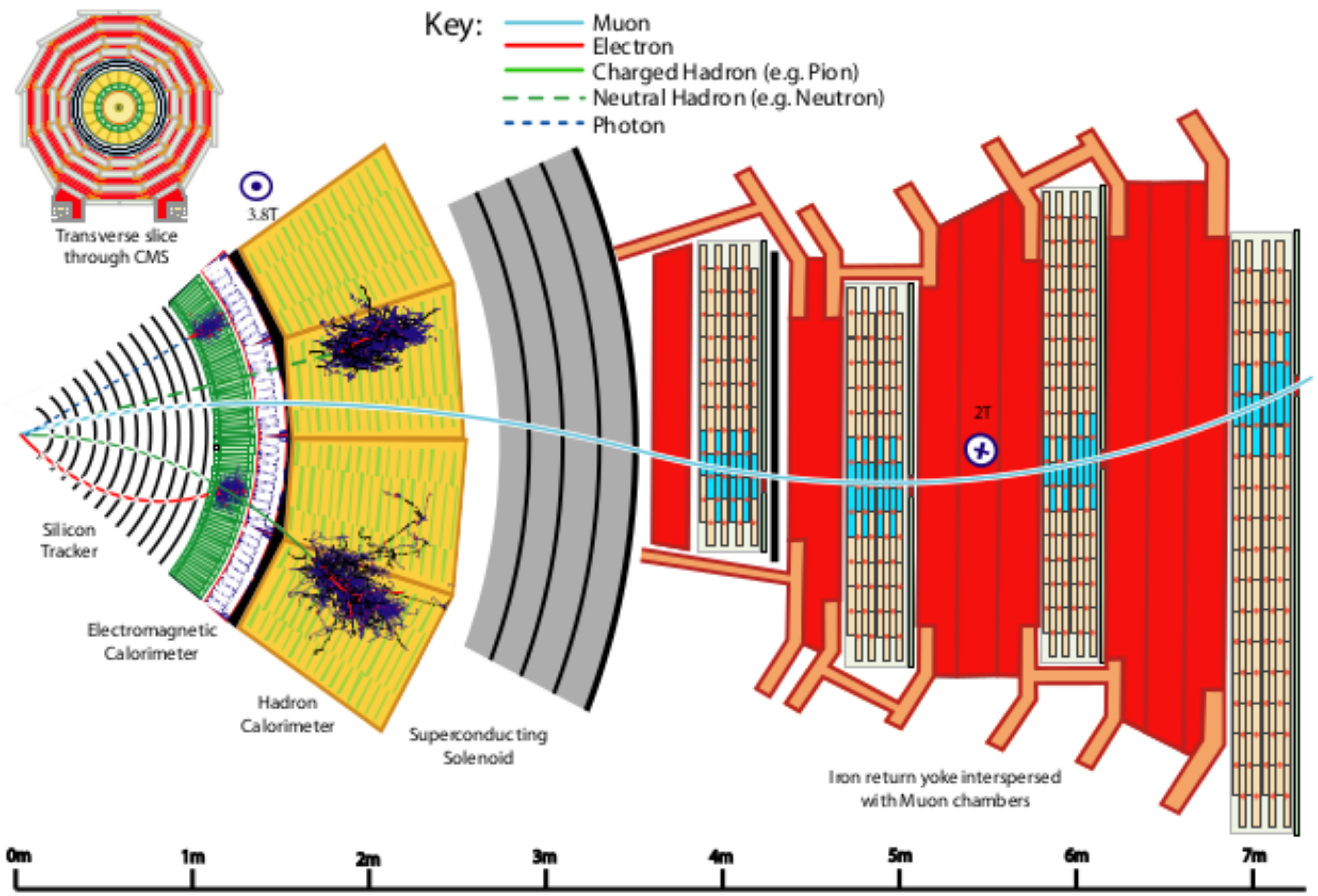
$$p(x|\theta) = \int dz \;\; p(x|z_h)\, p(z_h|z_p)\, p(z_p|\theta)$$

Forward simulator but inverse problem

Simulator let's you sample from the likelihood

But no analytical formula, no tractable likelihood

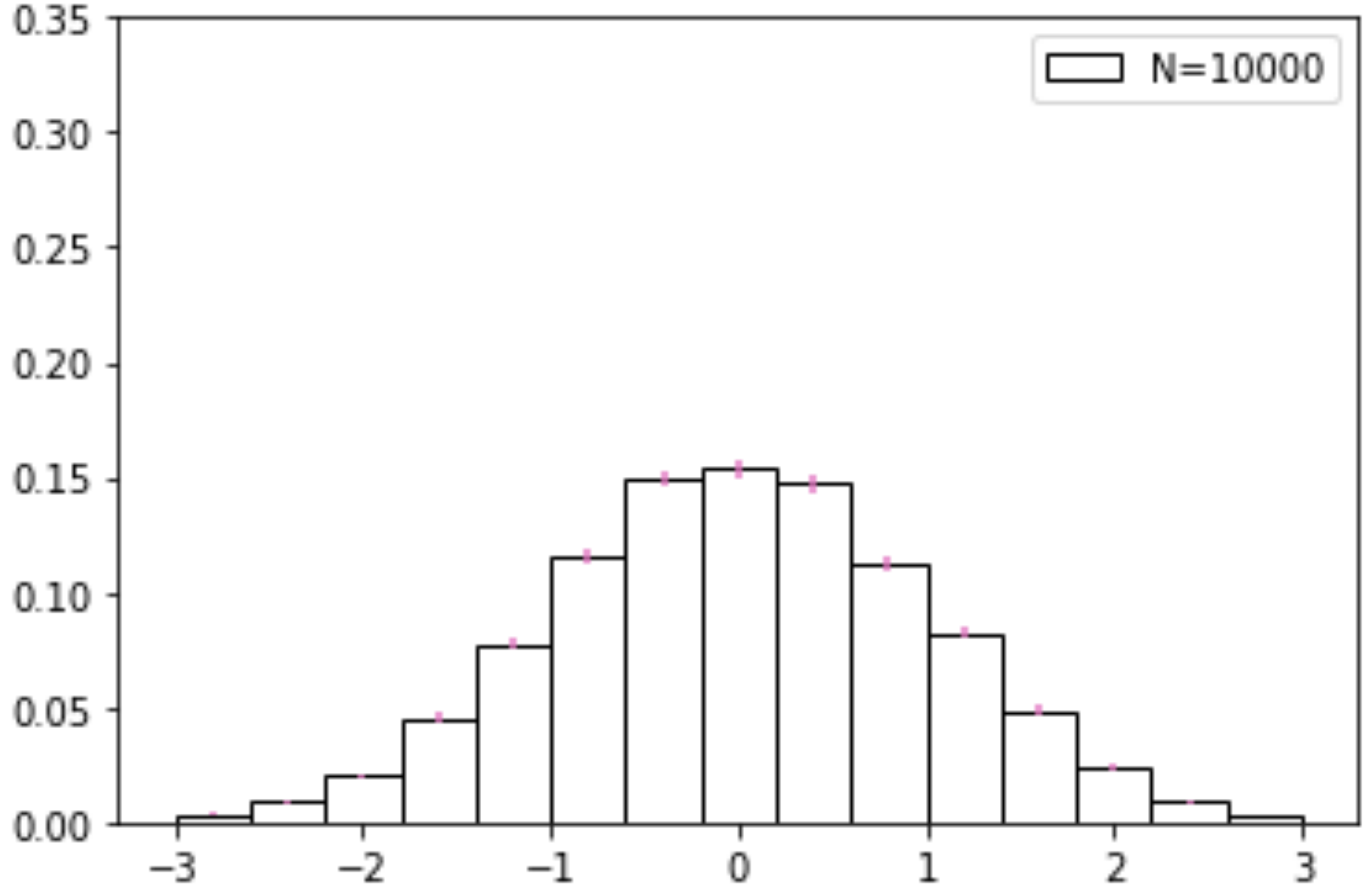
Typically, simulator cannot be run in reverse



Intractable:

$$p(x|\theta) = \int dz \;\; p(x|z_h)\; p(z_h|z_p)\; p(z_p|\theta)$$

Traditional solution at LHC: Histograms
Likelihood can be calculated in low-dimensional summary statistic



High-dimensional data compressed into 1 variable and binned into histogram

The count of events in *each bin* follows a Poisson distribution

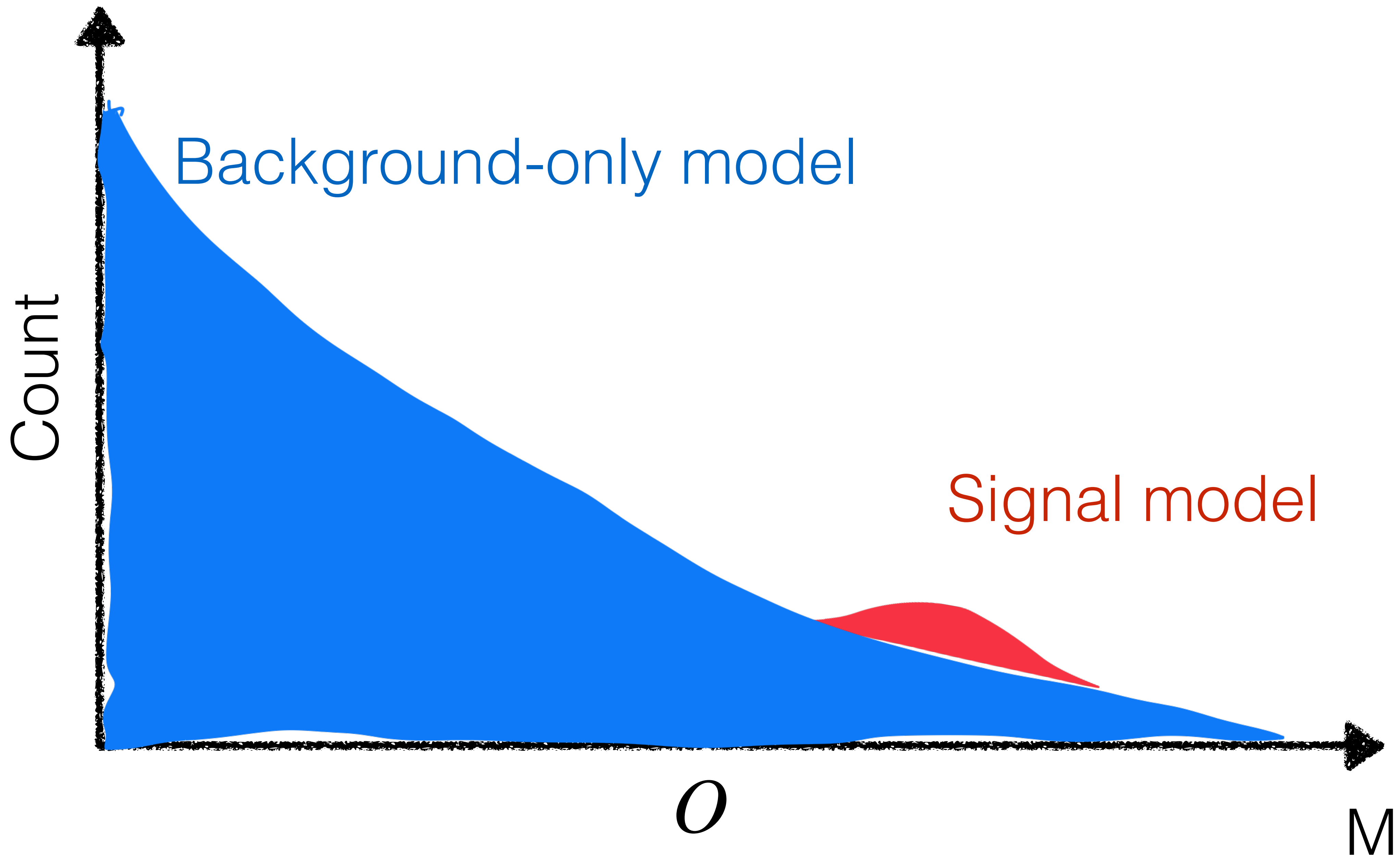
$$P(N_{obs} = k \mid N_{exp} = \lambda) = \frac{\lambda^k e^{-k}}{k!}$$

From experiment data

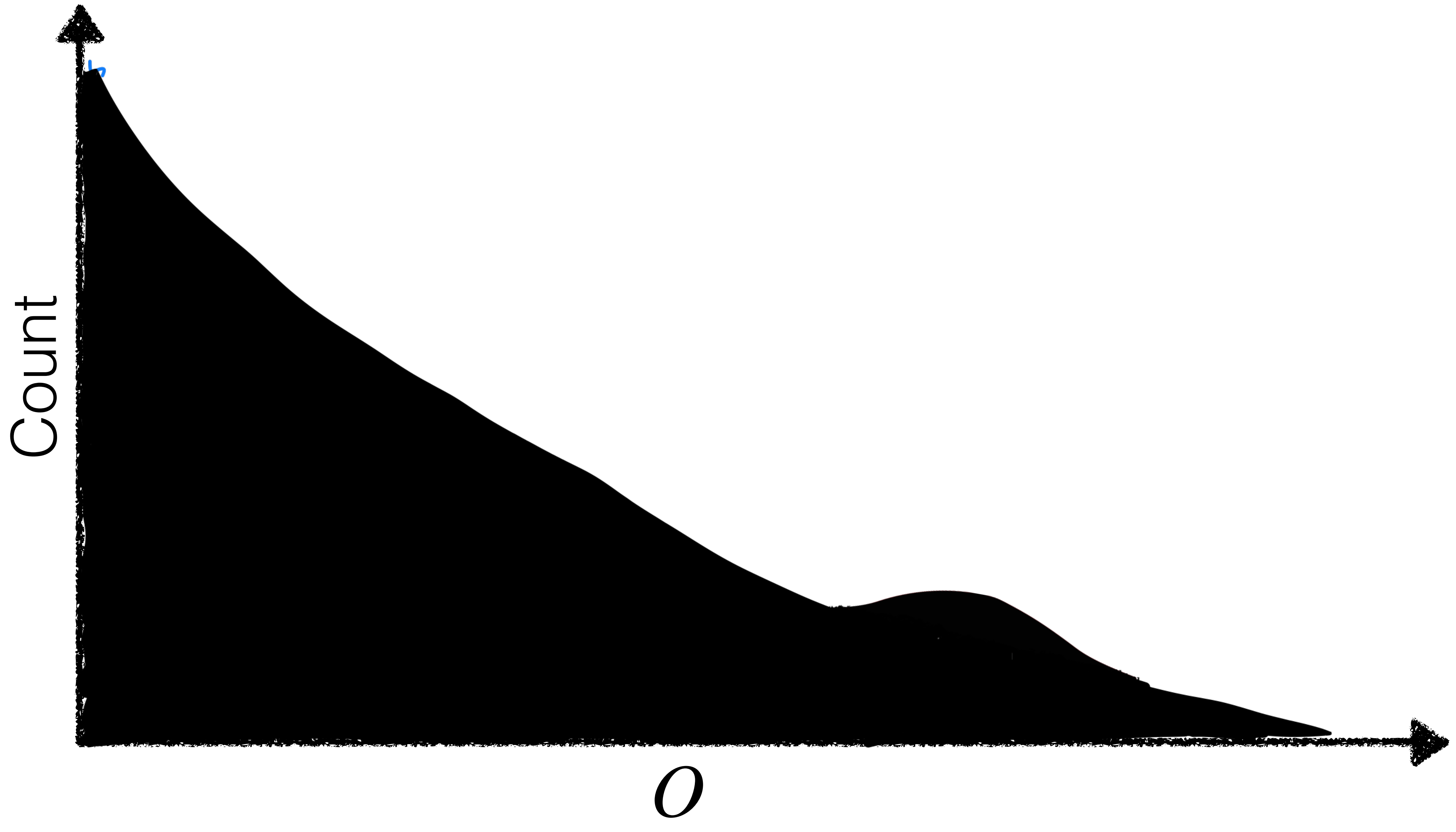
From theory simulation

Density estimation with summary statistic

Theory predictions from simulator



Experiment Data

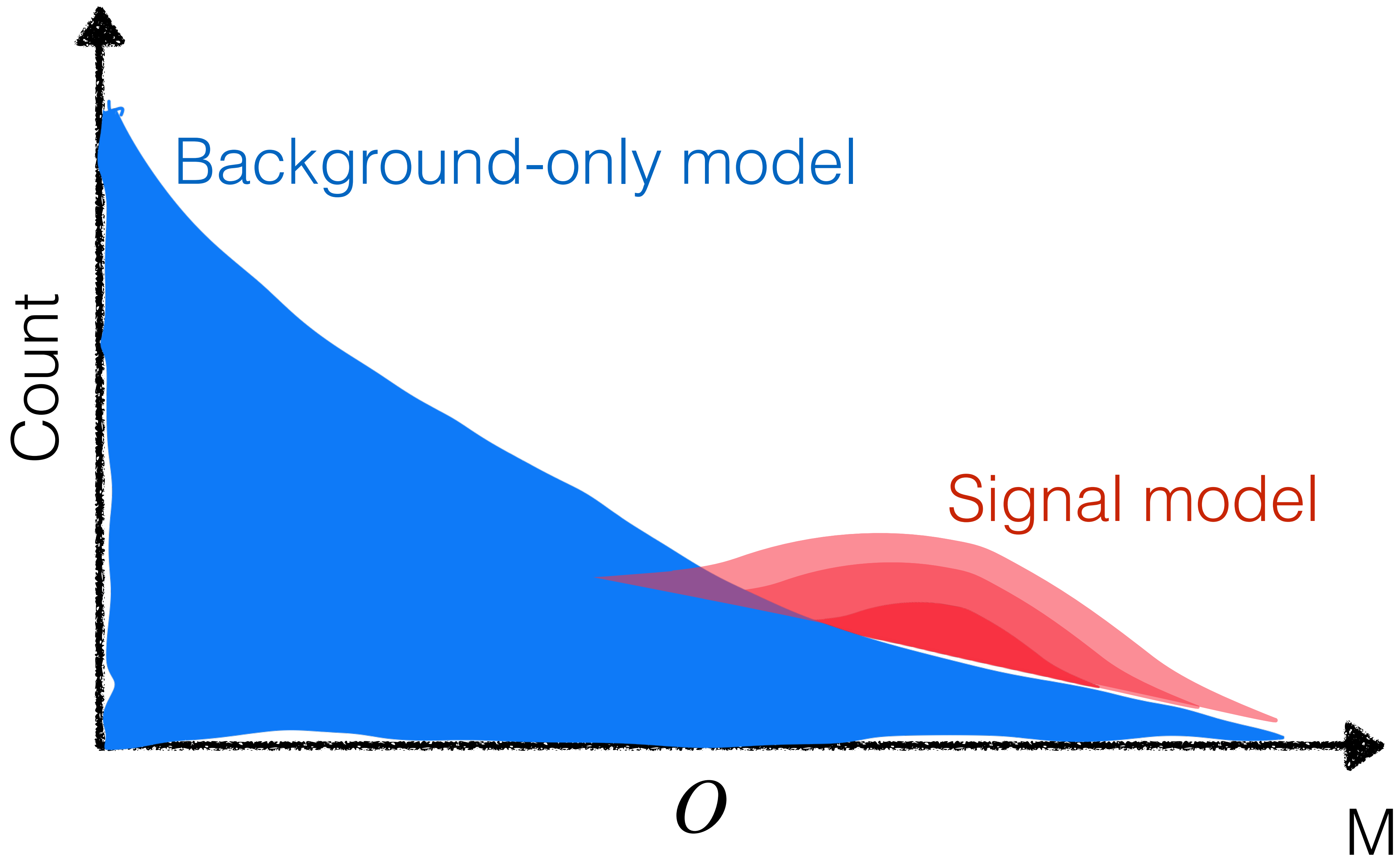


Measure single strength μ

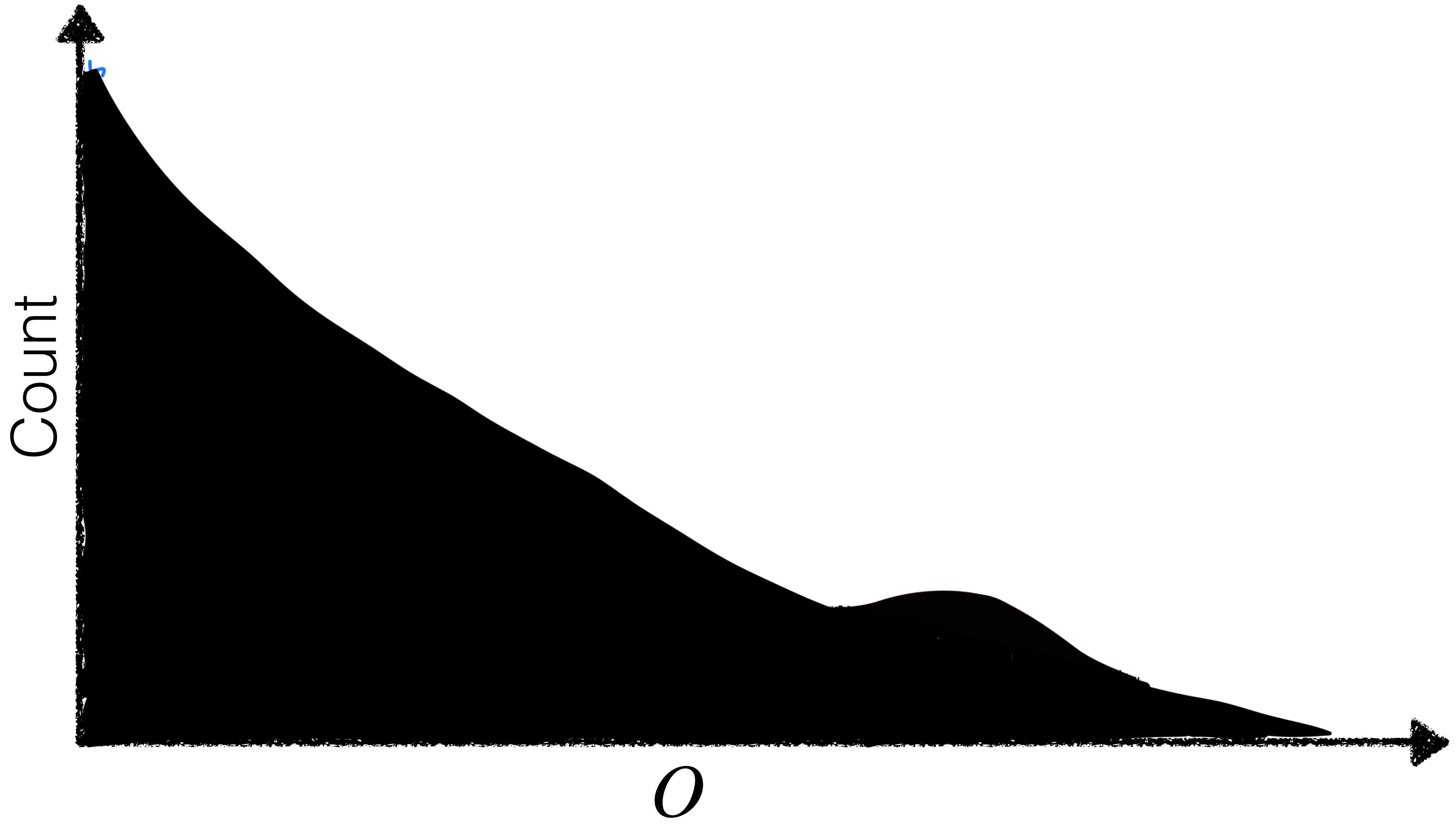
With histograms we can ask "Given the data, what is the likelihood a $\mu = 1$ hypothesis vs $\mu = 2$ hypothesis?"

Density estimation with summary statistic

Theory predictions from simulator



Experiment Data



Measure single strength μ

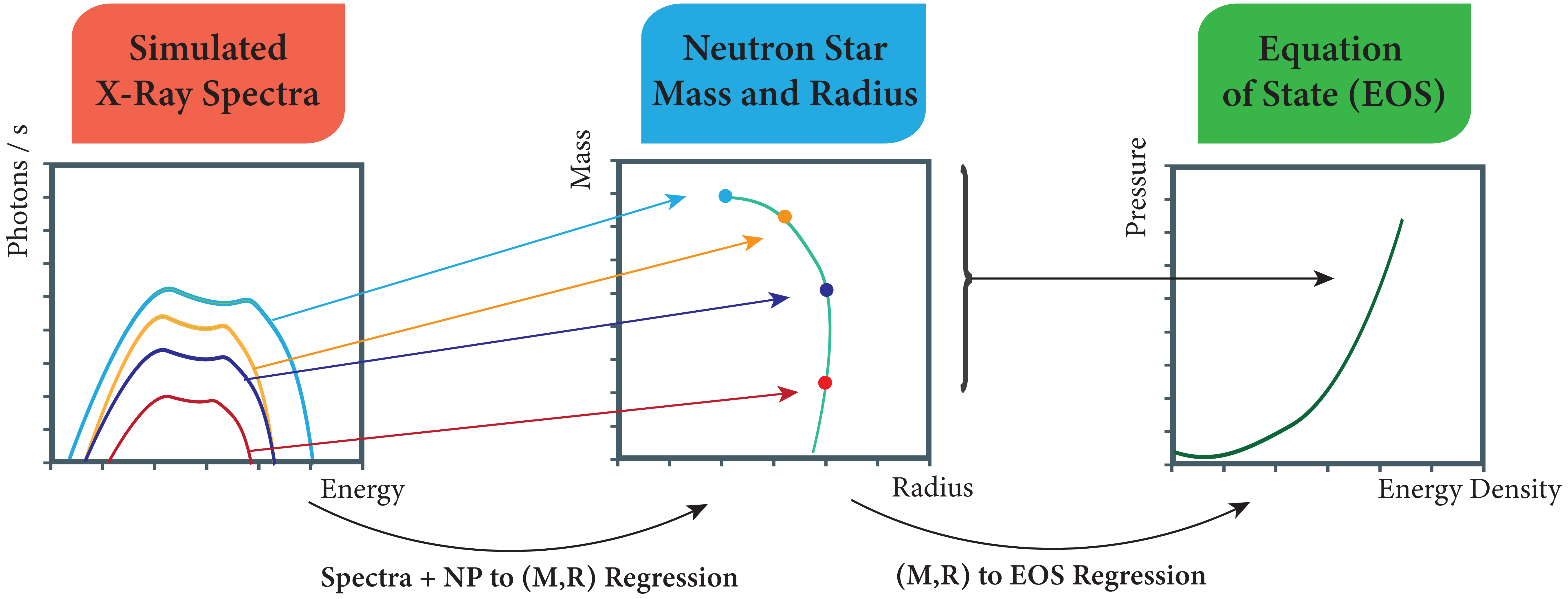
With histograms we can ask "Given the data, what is the likelihood a $\mu = 1$ hypothesis vs $\mu = 2$ hypothesis?"

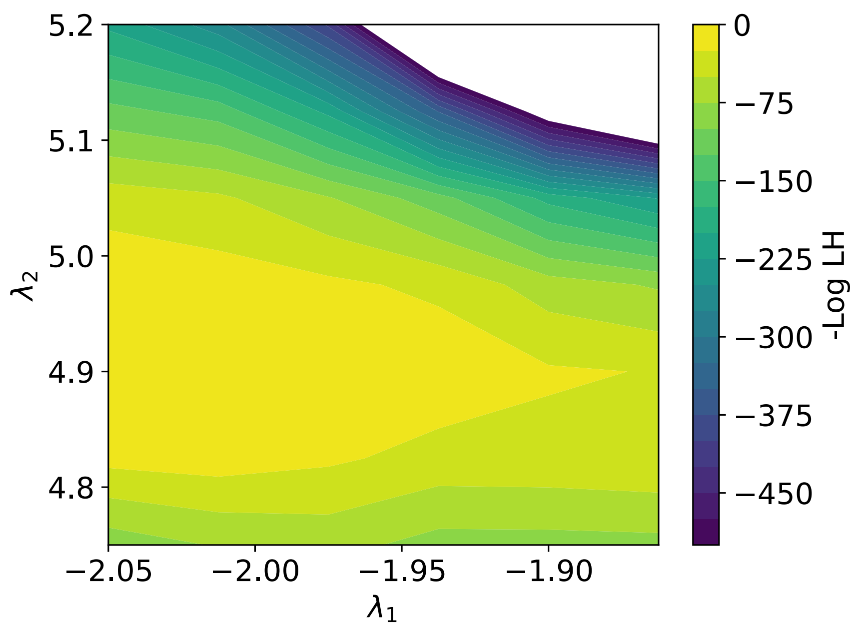
Summary statistics, a neutron stars example

Traditional method collapsed information about star into 2 numbers: mass and radius

Perform statistical inference in this low dimensional space

Lossy compression !





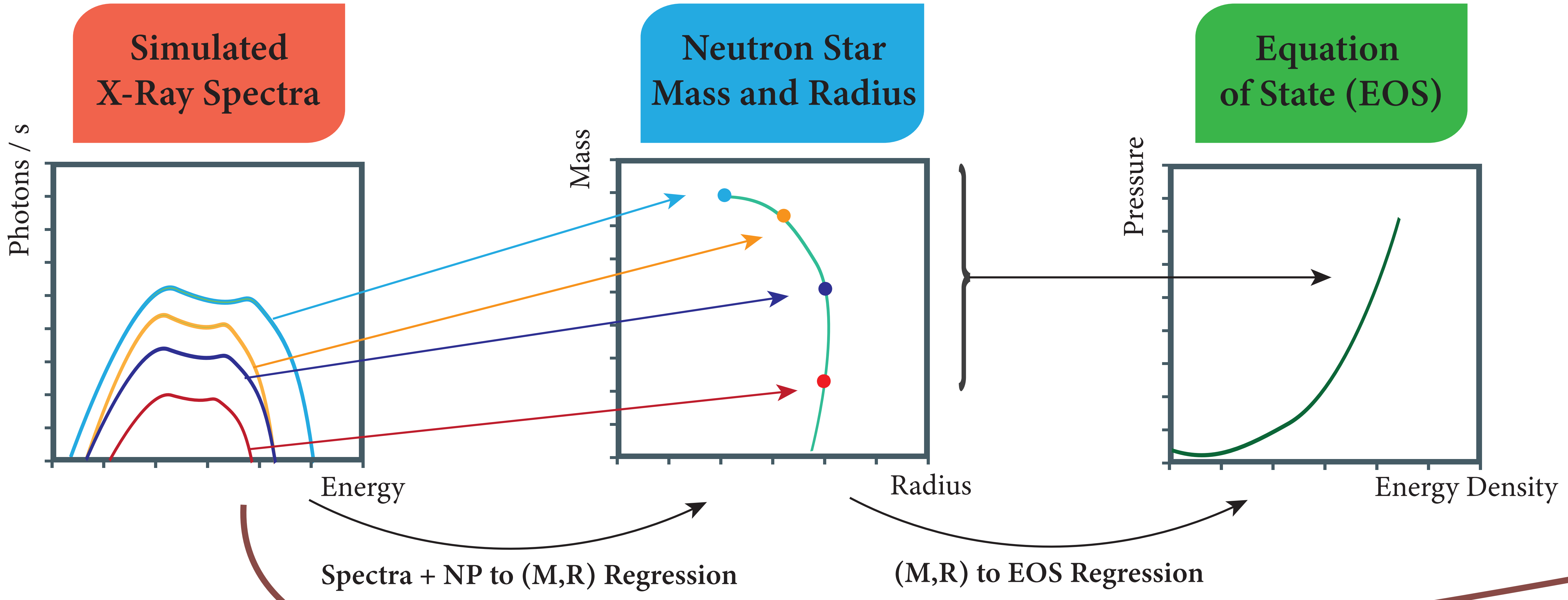
arxiv:2403.00287: Len Brandes, Chirag Modi, Aishik Ghosh, et al.

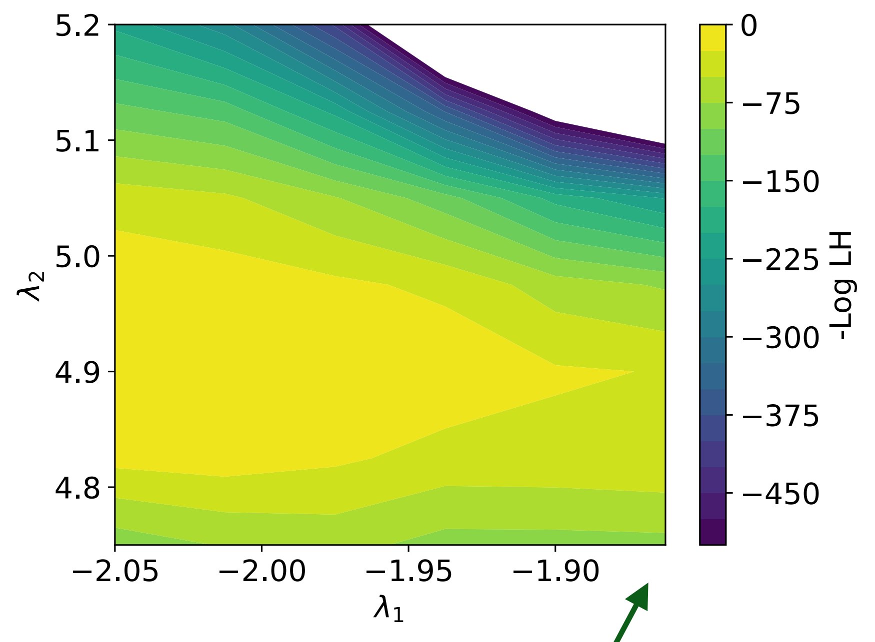
Summary statistics, a neutron stars example

Traditional method collapsed information about star into 2 numbers: mass and radius

Perform statistical inference in this low dimensional space

Lossy compression !





NPs

Direct estimation of likelihood from high-dimensional raw data gives better results!

arxiv:2403.00287: Len Brandes, Chirag Modi, Aishik Ghosh, et al.

This is a density estimation problem, not a supervised regression

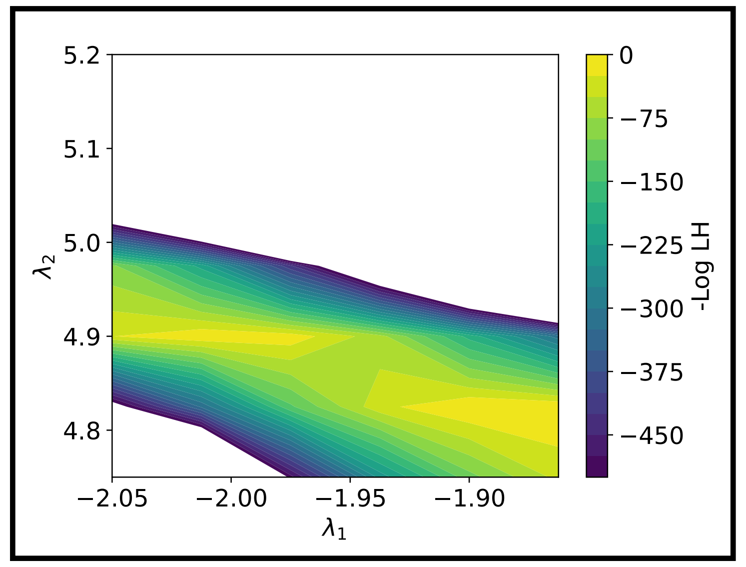
We do not know the true $p(x_i | theory)$ of an event even in our simulations

This is a density estimation problem, not a supervised regression

We do not know the true $p(x_i \,|\, theory)$ of an event even in our simulations

High-dimensional density estimation with neural networks, unsupervised methods like:

- Normalising flows for neural likelihood estimation
- Diffusion models for neural posterior estimation



Or ‘supervised’ method!

- Classifiers for neural ratio estimation

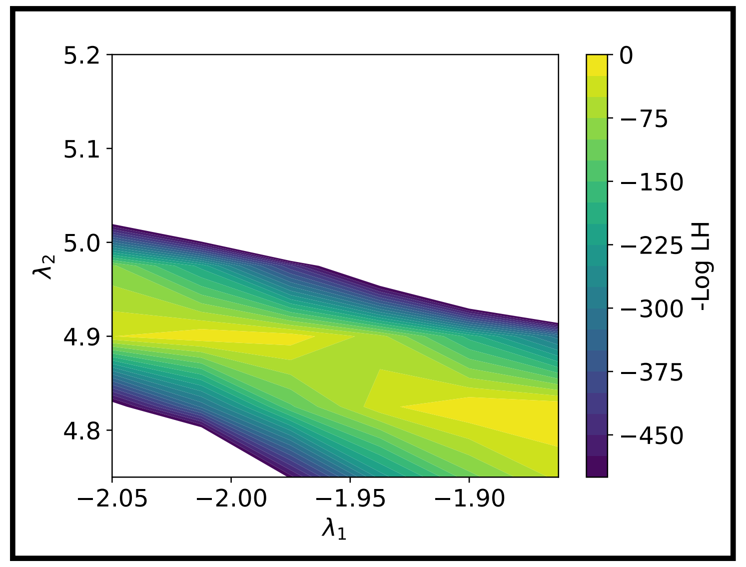
See PHY-STAT Munich workshop for different examples

This is a density estimation problem, not a supervised regression

We do not know the true $p(x_i \,|\, theory)$ of an event even in our simulations

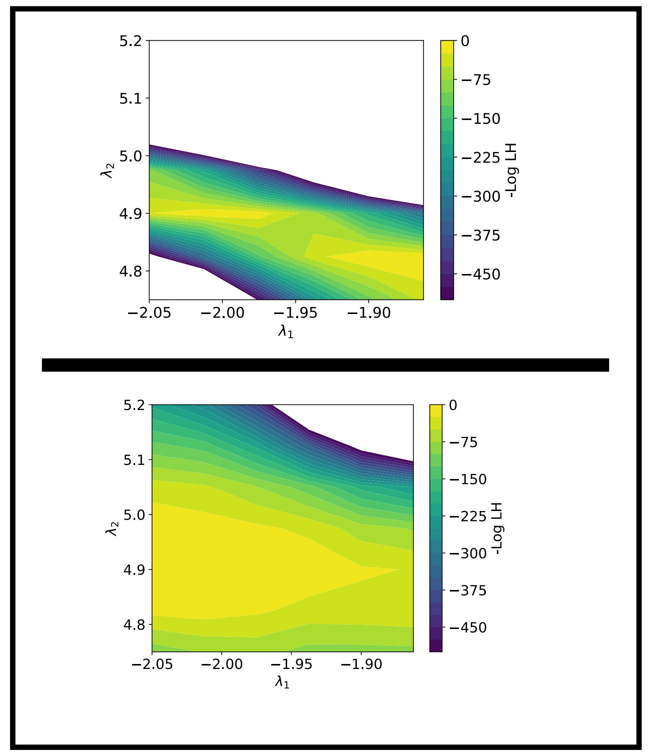
High-dimensional density estimation with neural networks, unsupervised methods like:

- Normalising flows for neural likelihood estimation
- Diffusion models for neural posterior estimation



Or 'supervised' method!

- Classifiers for neural ratio estimation



See PHY-STAT Munich workshop for different examples

The motivation for Neural SBI in particle physics

Traditional {S vs B} classifier often good enough

Traditional {S vs B} classifier often good enough

$$\mathscr{L}(\mu \mid \mathscr{D}) = p(\mathscr{D} \mid \mu)$$

Neyman–Pearson lemma: Likelihood ratio is the most powerful test statistic

We want to compare likelihoods:
$$\frac{p(\mathscr{D} \mid \mu)}{p(\mathscr{D} \mid \mu_0)}$$

Traditional {S vs B} classifier often good enough

$$\mathcal{L}(\mu \,|\, \mathcal{D}) = p(\mathcal{D} \,|\, \mu)$$

Neyman–Pearson lemma: Likelihood ratio is the most powerful test statistic

We want to compare likelihoods:

$$\frac{p(\mathcal{D} \,|\, \mu)}{p(\mathcal{D} \,|\, \mu_0)}$$

Traditional {S vs B} classifier often good enough

$$\mathcal{L}(\mu \mid \mathcal{D}) = p(\mathcal{D} \mid \mu)$$

Neyman–Pearson lemma: Likelihood ratio is the most powerful test statistic

We want to compare likelihoods: $$\frac{p(\mathcal{D} \mid \mu)}{p(\mathcal{D} \mid \mu_0)}$$

A neural network classifier trained on S vs B, estimates the decision function*: $$s(x_i) = \frac{p(x_i \mid S)}{p(x_i \mid S) + p(x_i \mid B)}$$

* Equal class weights

Traditional {S vs B} classifier often good enough

$$\mathscr{L}(\mu \,|\, \mathscr{D}) = p(\mathscr{D} \,|\, \mu)$$

Neyman–Pearson lemma: **Likelihood ratio is the most powerful test statistic**

We want to compare likelihoods: $$\frac{p(\mathscr{D} \,|\, \mu)}{p(\mathscr{D} \,|\, \mu_0)}$$

A neural network classifier trained on S vs B, estimates the decision function*: $$s(x_i) = \frac{p(x_i \,|\, S)}{p(x_i \,|\, S) + p(x_i \,|\, B)}$$

Which contains all the information required for the likelihood ratio:

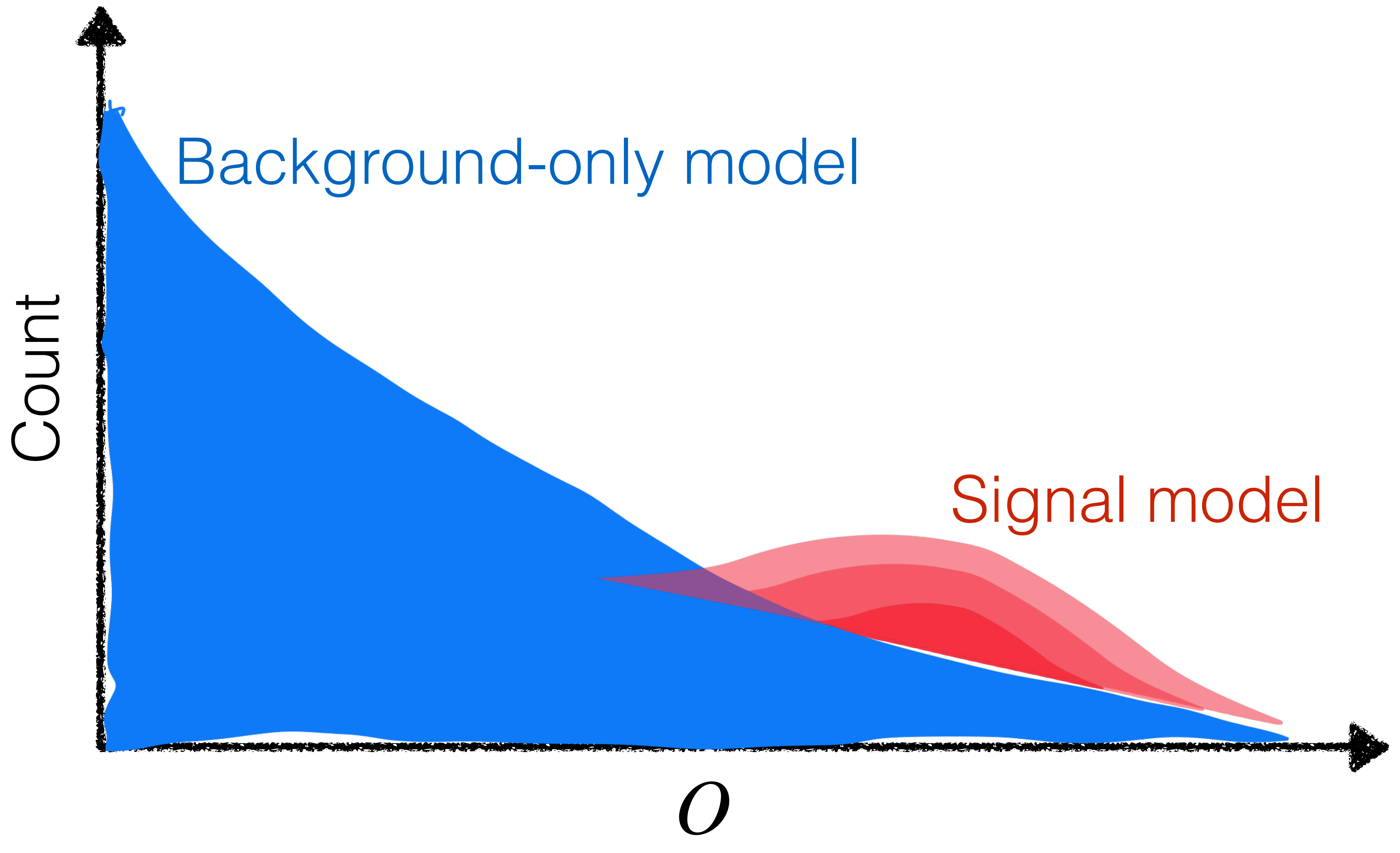
$$\frac{p(x_i \,|\, \mu)}{p(x_i \,|\, \mu = 0)} = \frac{\mu \cdot \sigma_S \cdot p(x_i \,|\, S) + \sigma_B \cdot p(x_i \,|\, B)}{\sigma_B \cdot p(x_i \,|\, B)} = \mu \cdot \frac{\sigma_S}{\sigma_B} \cdot \frac{s(x_i)}{1 - s(x_i)} + 1.$$

Same observable s is optimal to test all μ hypotheses!

No need to develop separate analysis per hypothesis μ

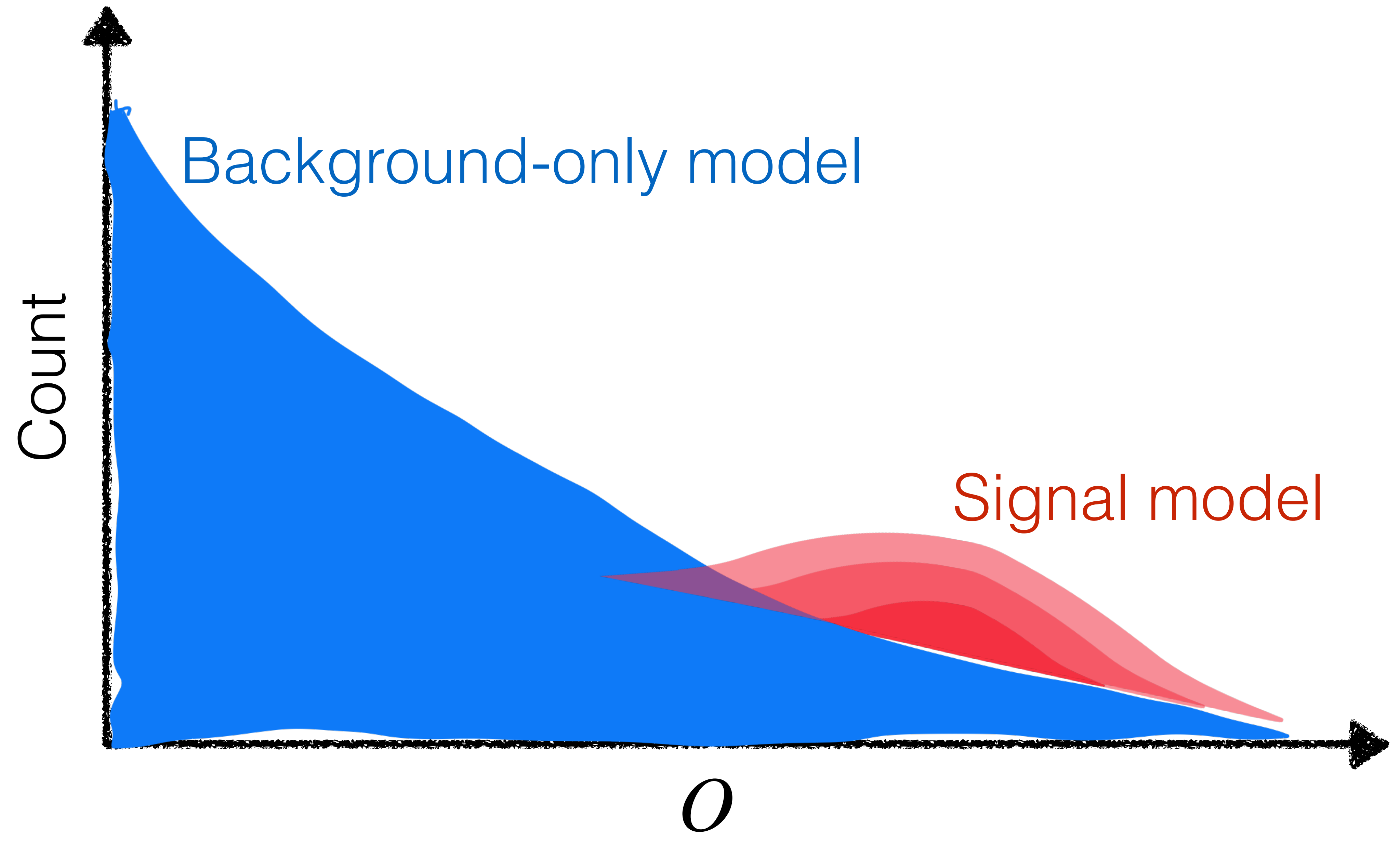
* Equal class weights

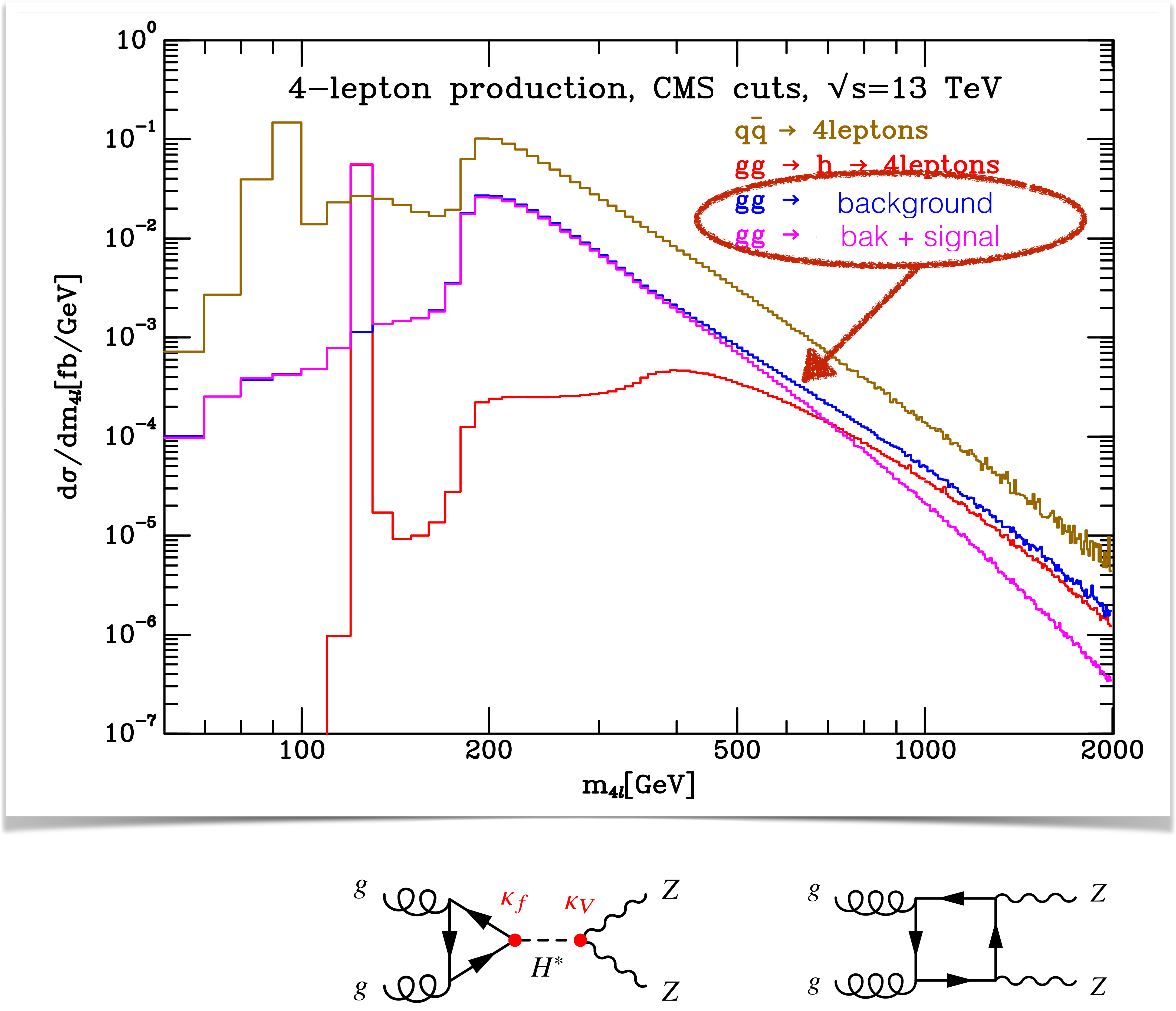
New challenges, eg. quantum interference



A histogram of any single observable is no longer optimal (see Ghosh et al.)

New challenges, eg. quantum interference





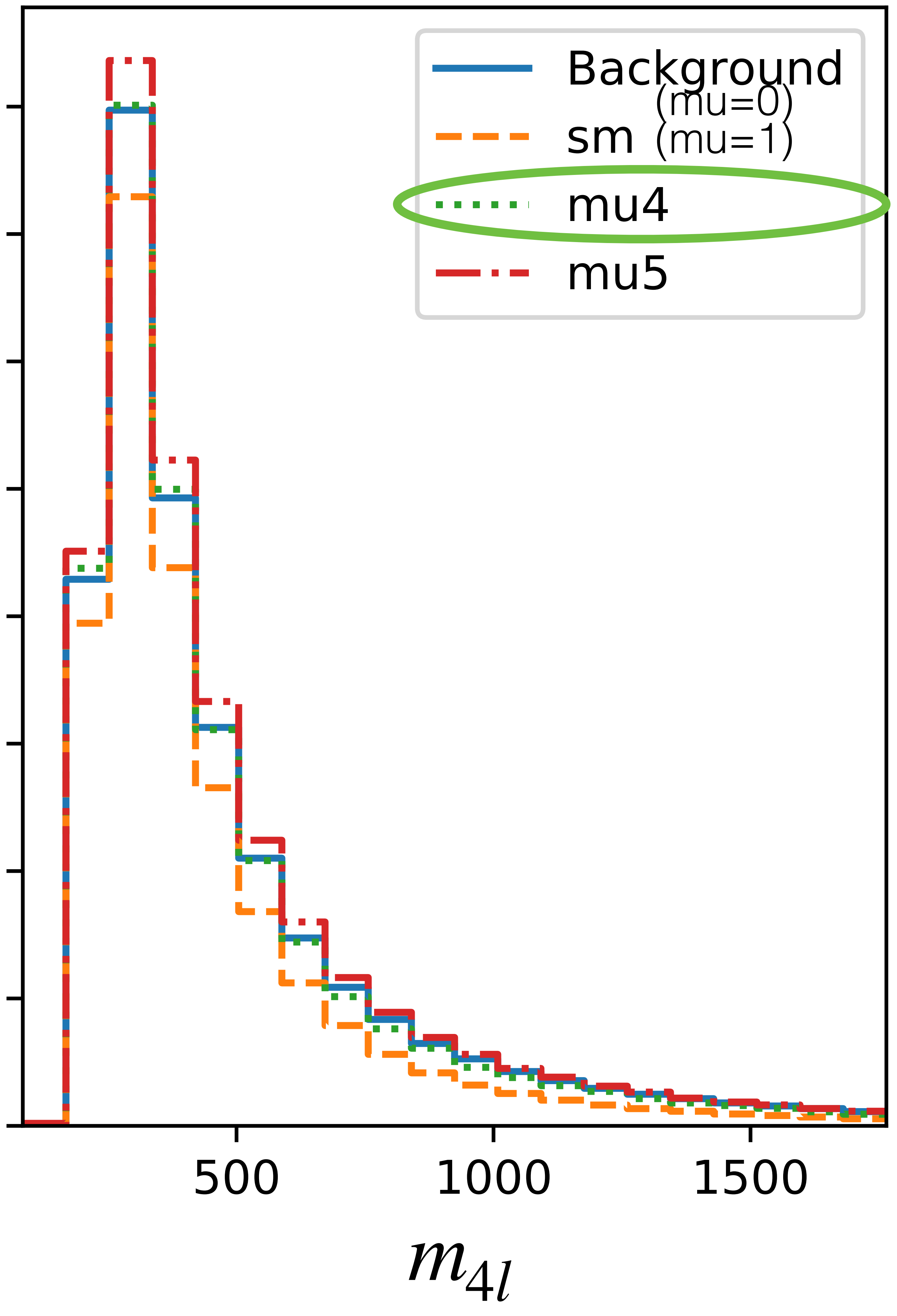
A histogram of any single observable is no longer optimal (see [Ghosh et al.](#))

Example of where summary statistics break down in presence of quantum interference

$$H^* \rightarrow ZZ \rightarrow 4l$$

Example of where summary statistics break down in presence of quantum interference

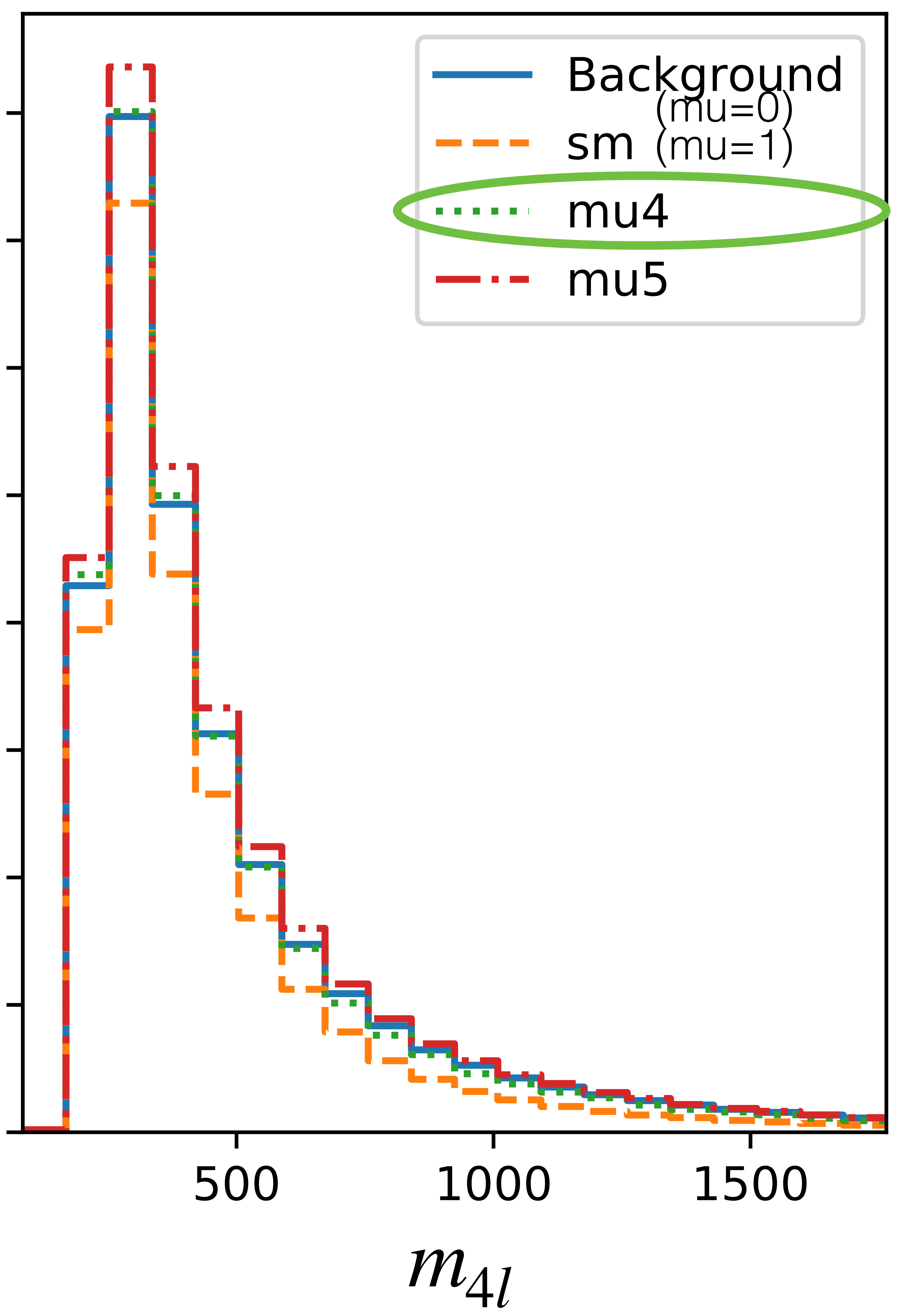
$$H^* \rightarrow ZZ \rightarrow 4l$$



Can you spot the green plot?

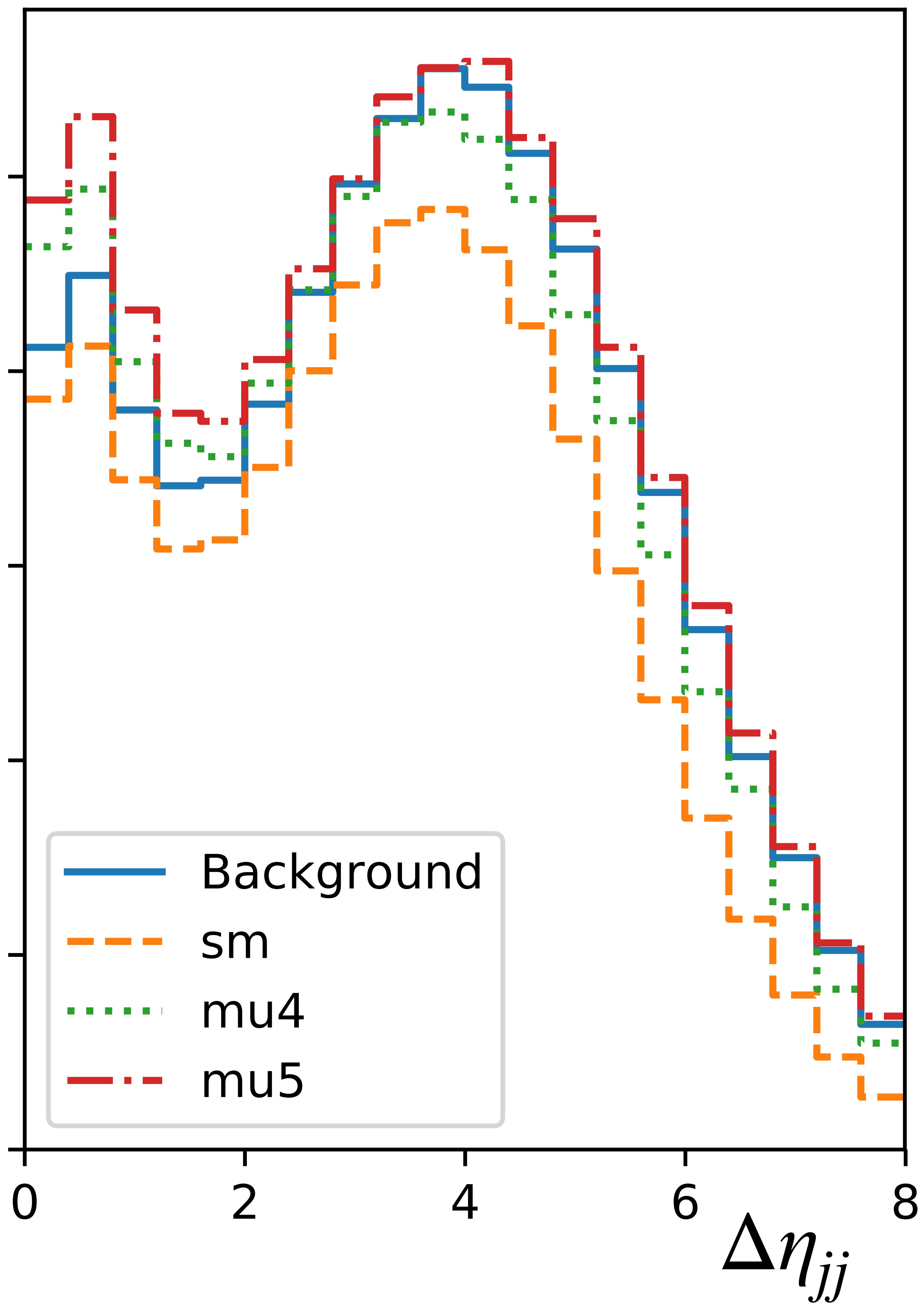
Example of where summary statistics break down in presence of quantum interference

$$H^* \to ZZ \to 4l$$



Can you spot the green plot?

μ=4 indistinguishable from μ=0 but other observables can break the degeneracy



Optimal observable now changes as a function of **μ**: Cannot collapse problem to 1 dimension

What breaks down?

$$P(X) = |M_s(X) + M_b(X)|^2 = \underbrace{|M_s(X)|^2}_{P_s(X)} + \underbrace{|M_b(X)|^2}_{P_b(X)} + \underbrace{2\,\mathrm{Re}(\overline{M_s(X)}M_b(X))}_{P_i(X)}$$

$$N_{exp} = \mu \cdot S + B + \sqrt{\mu} \cdot I$$

A neural network classifier trained on S vs B, estimates the decision function: $s(x_i) = \dfrac{p(x_i\,|\,S)}{p(x_i\,|\,S) + p(x_i\,|\,B)}$

Which contains all the information required for the likelihood ratio:

$$\frac{p(x_i\,|\,\mu)}{p(x_i\,|\,\mu=0)} = \frac{\mu \cdot \sigma_S \cdot p(x_i\,|\,S) + \sigma_B \cdot p(x_i\,|\,B)}{\sigma_B \cdot p(x_i\,|\,B)} = \mu \cdot \frac{\sigma_S}{\sigma_B} \cdot \frac{s(x_i)}{1 - s(x_i)} + 1.$$

Same observable s is optimal to test all μ hypotheses!
No need to develop separate analysis per hypothesis μ

8

No longer in this convenient spacial case: The same observable **no longer optimal due to non-linear effects** coming from quantum interference

- **Does not generalise to an arbitrary theory parameter θ**, (eg. Effective Field Theory parameters)
- Summary statistics also not optimised for systematics

So many LHC analyses would benefit from high-dimensional inference!

Neural ratio estimation with a classifier: Hypothesis vs Ref Hypothesis

$$\mathscr{L}(\theta | \mathscr{D}) = p(\mathscr{D} | \theta)$$

Neyman–Pearson lemma: **Likelihood ratio is the most powerful test statistic**

We want to compare likelihoods: $\dfrac{p(\mathscr{D} | \theta)}{p(\mathscr{D} | ref)}$

A neural network classifier trained on simulated samples from θ_1 vs simulated samples from ref, estimates the decision function:

$$s(x_i) = \frac{p(x_i | \theta_1)}{p(x_i | \theta_1) + p(x_i | ref)}$$

Which contains all the information required for the likelihood ratio:

$$\frac{p(x_i | \theta_1)}{p(x_i | ref)} = \frac{s(x_i)}{1 - s(x_i)}$$

Cranmer et al

Neural ratio estimation with a classifier: Hypothesis vs Ref Hypothesis

$$\mathcal{L}(\theta \mid \mathcal{D}) = p(\mathcal{D} \mid \theta)$$

Neyman–Pearson lemma: **Likelihood ratio is the most powerful test statistic**

We want to compare likelihoods: $$\frac{p(\mathcal{D} \mid \theta)}{p(\mathcal{D} \mid ref)}$$

A neural network classifier trained on simulated samples from θ_1 vs simulated samples from ref, estimates the decision function:

$$s(x_i, \theta = \theta_1) = \frac{p(x_i \mid \theta_1)}{p(x_i \mid \theta_1) + p(x_i \mid ref)}$$

Which contains all the information required for the likelihood ratio:

We can even obtain this as a function of θ !

$$\frac{p(x_i \mid \theta_1)}{p(x_i \mid ref)} = \frac{s(x_i, \theta = \theta_1)}{1 - s(x_i, \theta = \theta_1)}$$

Neural ratio estimation with a classifier: Hypothesis vs Ref Hypothesis

$$\mathcal{L}(\theta \,|\, \mathcal{D}) = p(\mathcal{D} \,|\, \theta)$$

Neyman–Pearson lemma: **Likelihood ratio is the most powerful test statistic**

We want to compare likelihoods:

$$\frac{p(\mathcal{D} \,|\, \theta)}{p(\mathcal{D} \,|\, ref)}$$

A neural network classifier trained on simulated samples from θ_1 vs simulated samples from ref, estimates the decision function:

$$s(x_i, \theta = \theta_1) = \frac{p(x_i \,|\, \theta_1)}{p(x_i \,|\, \theta_1) + p(x_i \,|\, ref)}$$

Which contains all the information required for the likelihood ratio:

$$\frac{p(x_i \,|\, \theta_1)}{p(x_i \,|\, ref)} = \frac{s(x_i, \theta = \theta_1)}{1 - s(x_i, \theta = \theta_1)}$$

We can even obtain this as a function of θ !

$$\frac{p(x_i \,|\, \theta_2)}{p(x_i \,|\, ref)} = \frac{s(x_i, \theta = \theta_2)}{1 - s(x_i, \theta = \theta_2)}$$

$$\frac{p(x_i \,|\, \theta_3)}{p(x_i \,|\, ref)} = \frac{s(x_i, \theta = \theta_3)}{1 - s(x_i, \theta = \theta_3)}$$

$$\cdots$$

Neural ratio estimation with a classifier: Hypothesis vs Ref Hypothesis

$$\mathscr{L}(\theta \mid \mathscr{D}) = p(\mathscr{D} \mid \theta)$$

Neyman–Pearson lemma: **Likelihood ratio is the most powerful test statistic**

We want to compare likelihoods:
$$\frac{p(\mathscr{D} \mid \theta)}{p(\mathscr{D} \mid ref)}$$

A neural network classifier trained on simulated samples from θ_1 vs simulated samples from ref, estimates the decision function:

$$s(x_i, \theta = \theta_1) = \frac{p(x_i \mid \theta_1)}{p(x_i \mid \theta_1) + p(x_i \mid ref)}$$

Which contains all the information required for the likelihood ratio:

We can even obtain this as a function of θ !

$$\frac{p(x_i \mid \theta_1)}{p(x_i \mid ref)} = \frac{s(x_i, \theta = \theta_1)}{1 - s(x_i, \theta = \theta_1)}$$

$$\frac{p(x_i \mid \theta_2)}{p(x_i \mid ref)} = \frac{s(x_i, \theta = \theta_2)}{1 - s(x_i, \theta = \theta_2)}$$

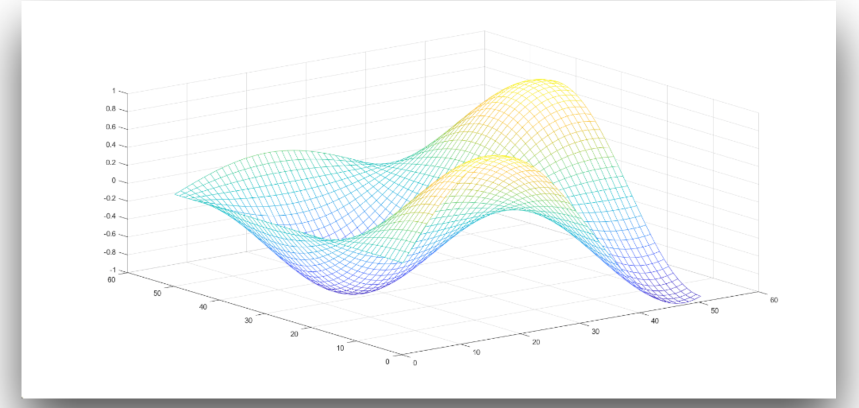
$$\frac{p(x_i \mid \theta_3)}{p(x_i \mid ref)} = \frac{s(x_i, \theta = \theta_3)}{1 - s(x_i, \theta = \theta_3)}$$

$\ldots$

- Optimal statistic to test each value of θ
- We get the LR *per event* (unbinned)

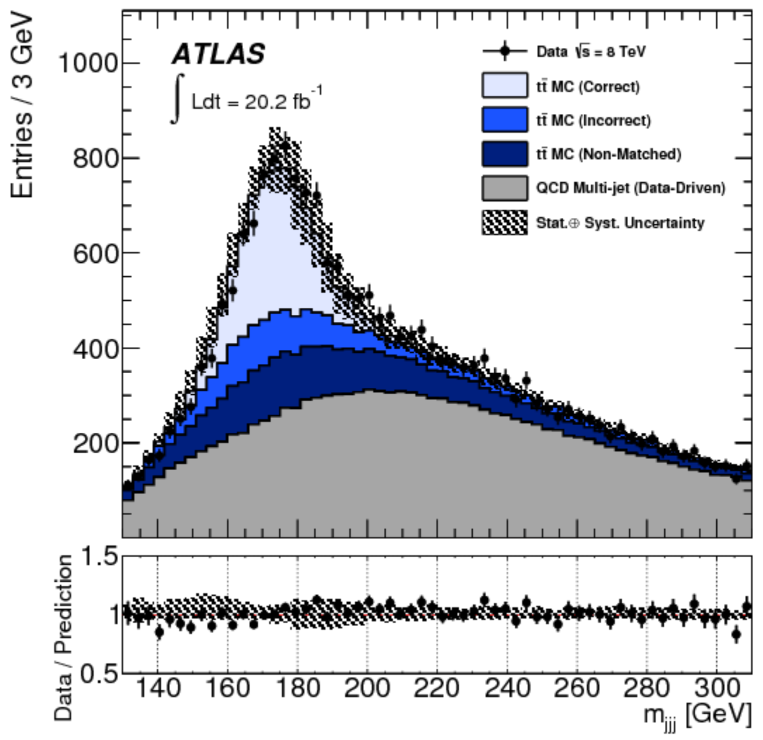
Core idea

Traditional framework:



High-dim data

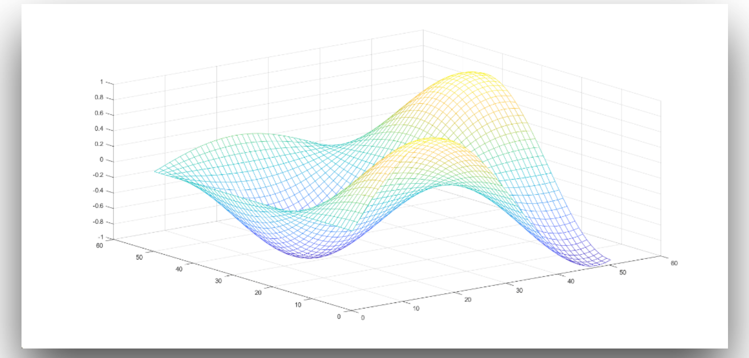
Summarisation to histogram



Summary Histogram → Statistical Fit → Likelihood $\mathscr{L}(\theta_1 | \mathscr{D})$

θ_1 → Statistical Fit

The neural inference framework:



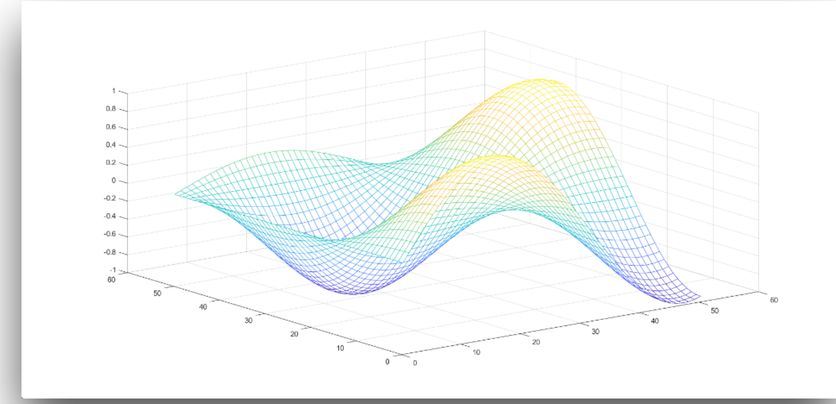
High-dim data

Obs Data → Neural Networks → Likelihood Ratio $\left(\frac{\mathscr{L}(\theta_1 | \mathscr{D})}{\mathscr{L}(ref | \mathscr{D})} \right)$

θ_1 → Neural Networks

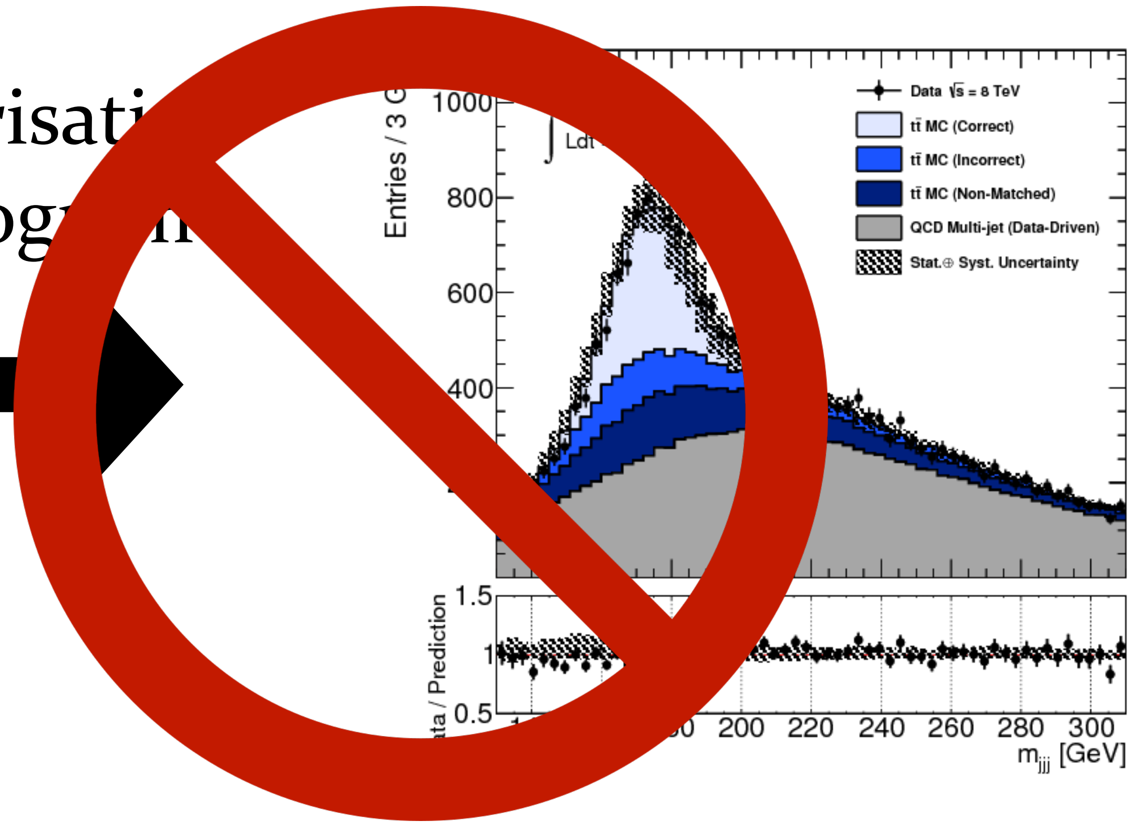
Core idea

Traditional framework:



High-dim data

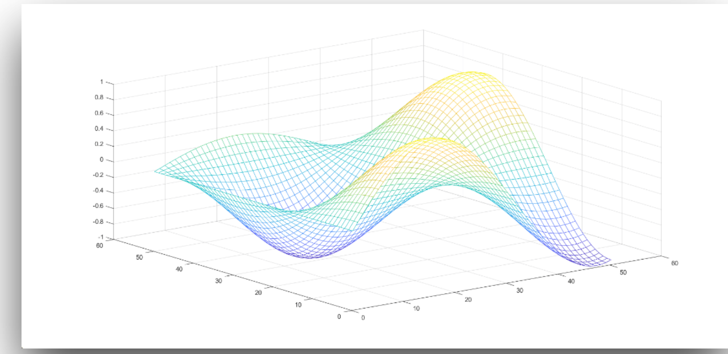
Summarisation to histogram



Summary Histogram → Statistical Fit → Likelihood $\mathscr{L}(\theta_1 | \mathscr{D})$

θ_1 → Statistical Fit

The neural inference framework:



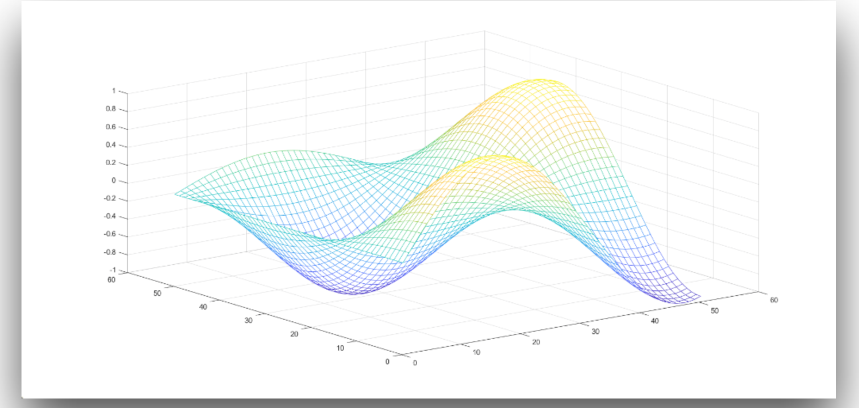
High-dim data

Obs Data → Neural Networks → Likelihood Ratio $\left(\frac{\mathscr{L}(\theta_1 | \mathscr{D})}{\mathscr{L}(ref | \mathscr{D})}\right)$

θ_1 → Neural Networks

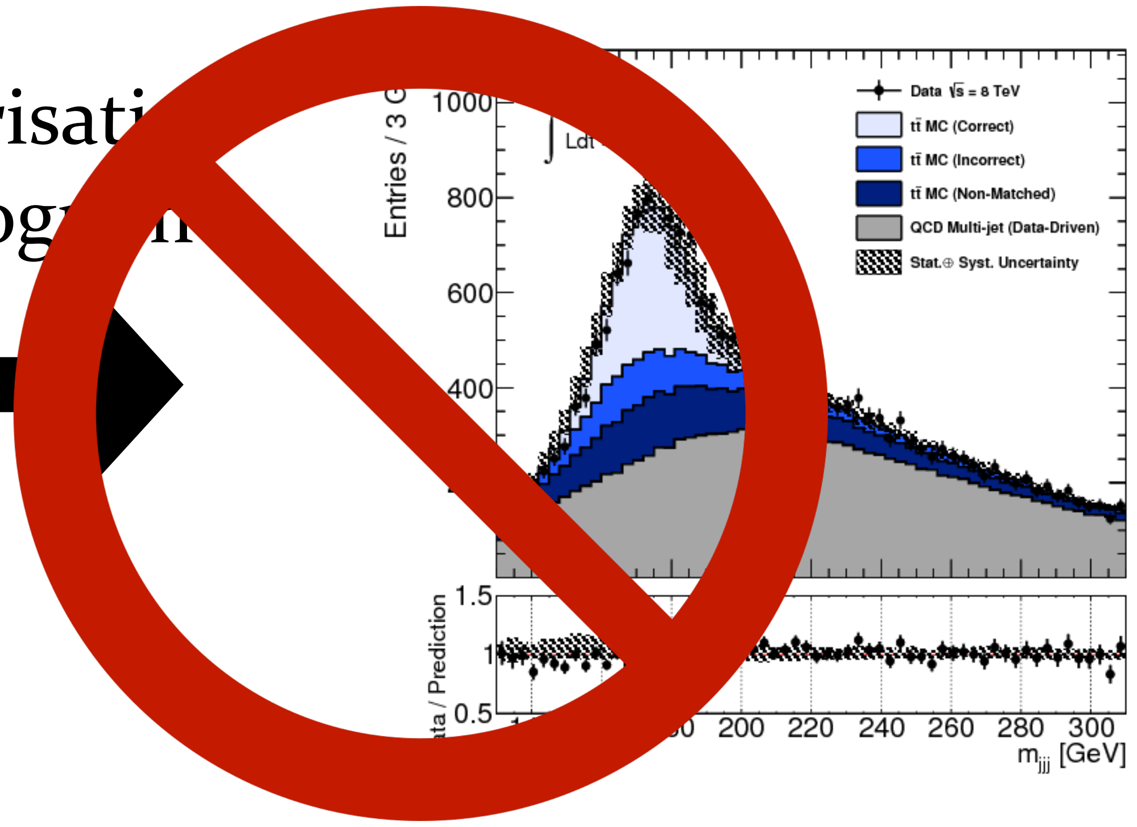
Core idea

Traditional framework:



High-dim data

Summarisati
to histog

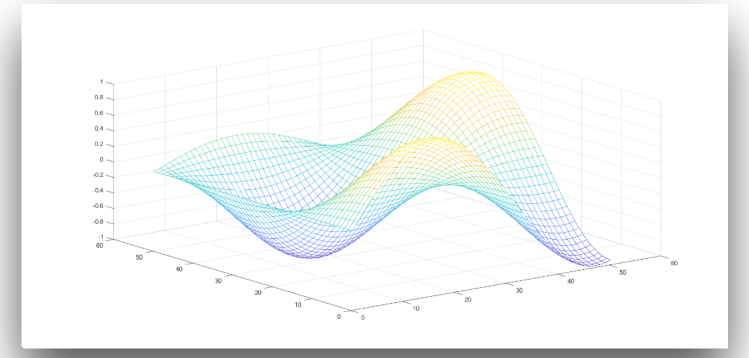


Summary Histogram → Statistical Fit → Likelihood

θ_1 → Statistical Fit

$\mathscr{L}(\theta_1 | \mathscr{D})$

The neural inference framework:



High-dim data

Obs Data → Neural Networks → Likelihood Ratio

θ_1 → Neural Networks

$$\left(\frac{\mathscr{L}(\theta_1 | \mathscr{D})}{\mathscr{L}(ref | \mathscr{D})}\right)$$

- Fully leverage detailed physics knowledge stored in simulators
- Perform high-dimensional inference

Phenomenology studies promise a dramatic improvement with high-dimensional inference

hal-02971995v3: Aishik Ghosh, David Rousseau

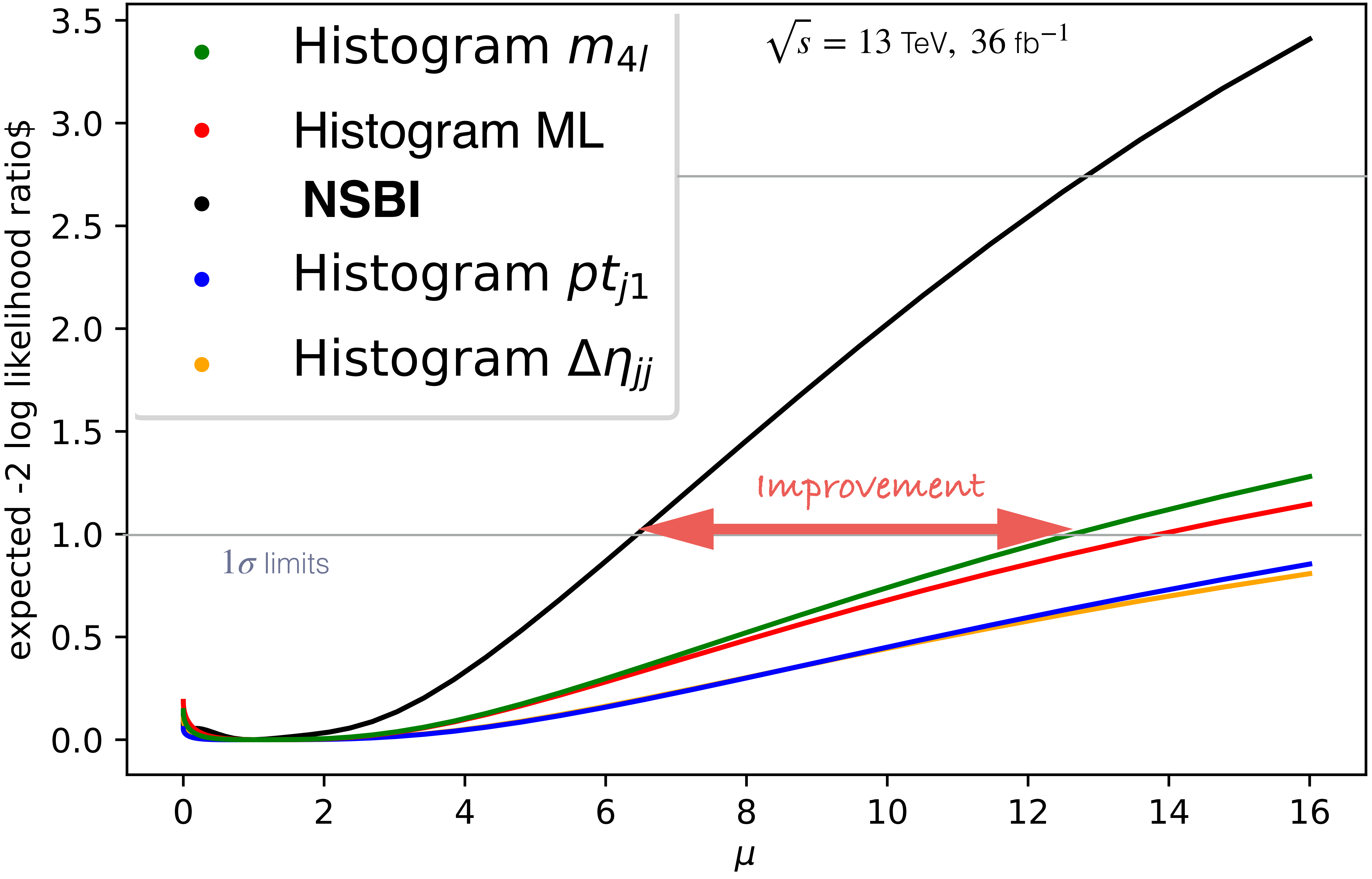
An example for Higgs width measurement

Phenomenology studies promise a dramatic improvement with high-dimensional inference

hal-02971995v3: Aishik Ghosh, David Rousseau

Expected sensitivity

An example for Higgs width measurement



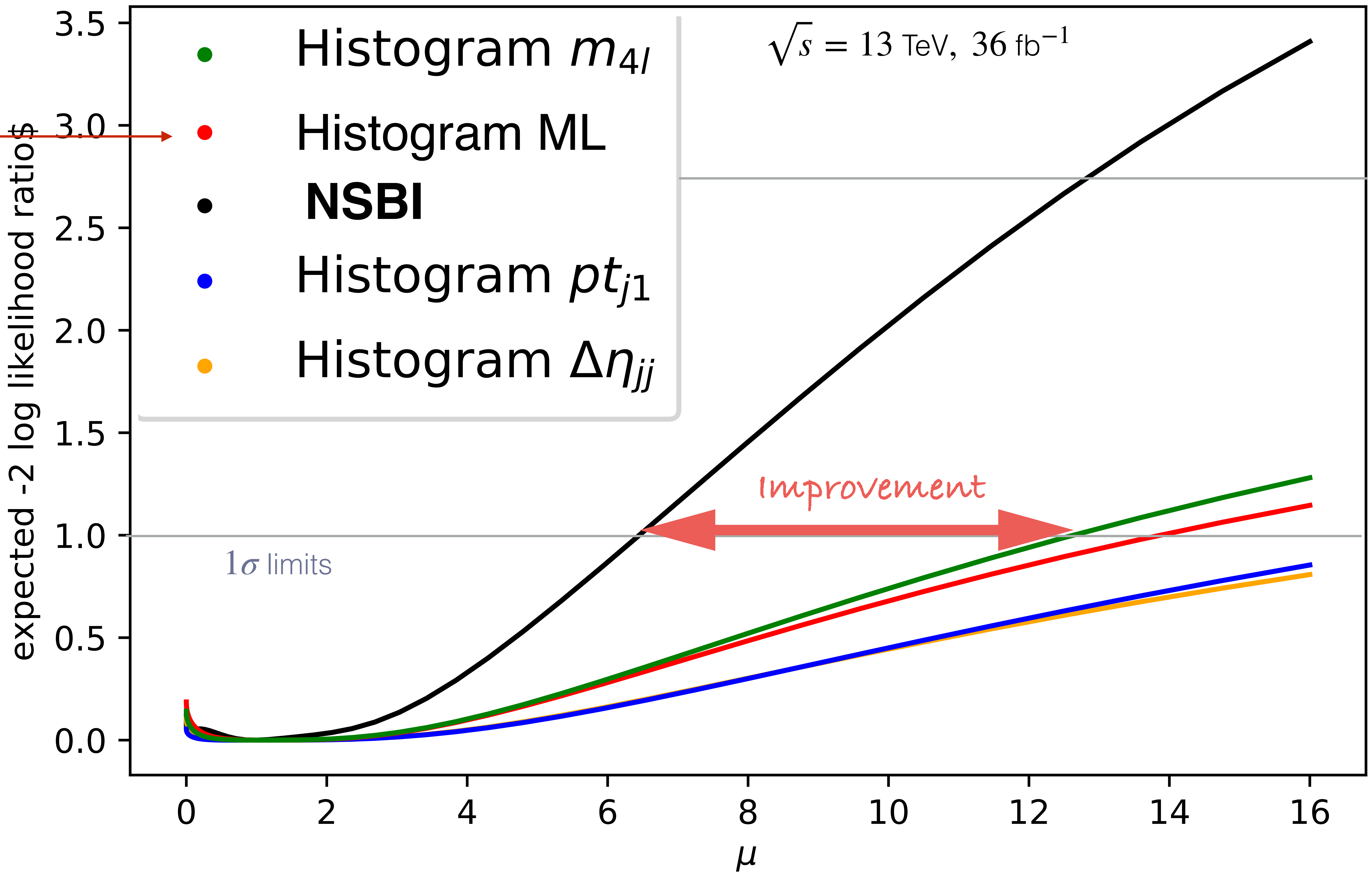
Phenomenology studies promise a dramatic improvement with high-dimensional inference

hal-02971995v3: Aishik Ghosh, David Rousseau

Expected sensitivity

“Traditional ML” baseline

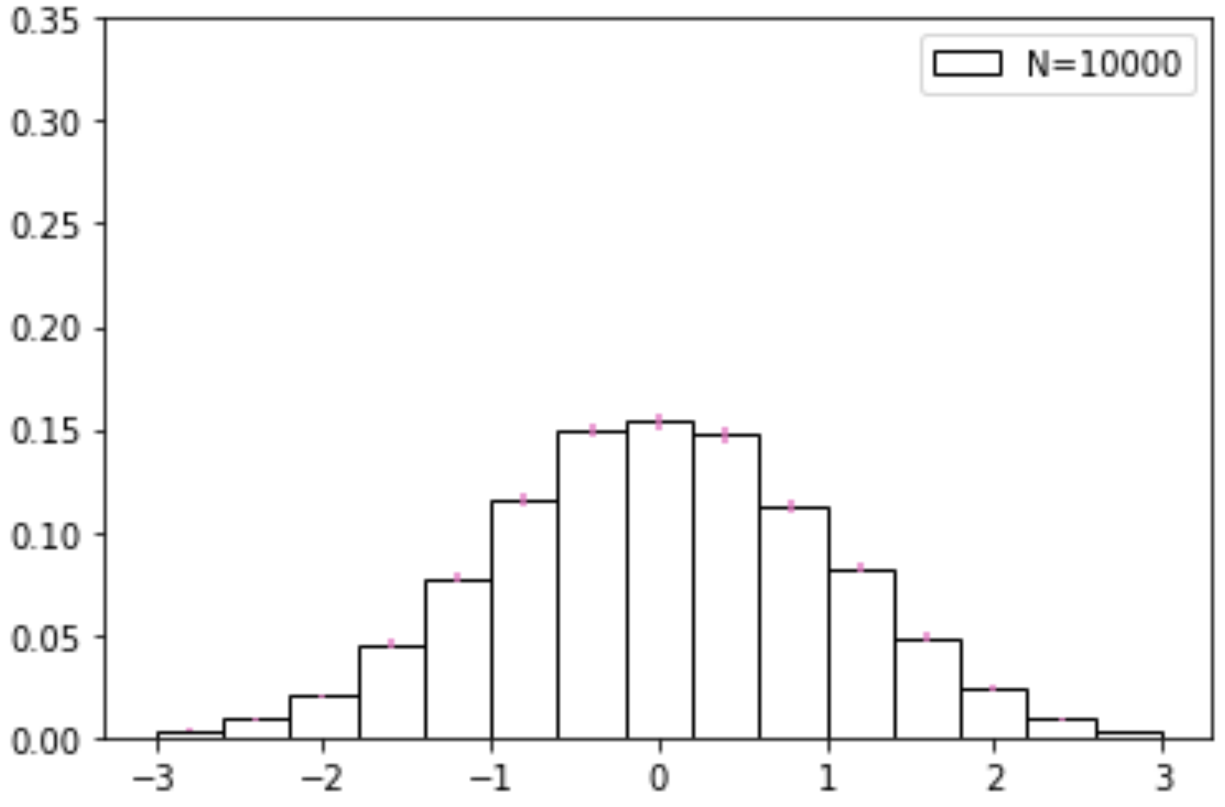
An example for Higgs width measurement



Challenges for NSBI:

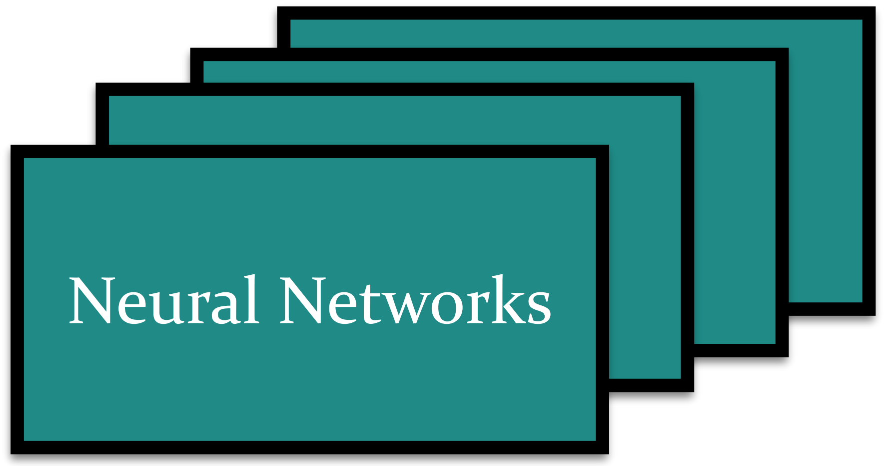
- Robustness: Design and validation
- Uncertainties: Quantifying and propagating systematics
- Neyman Construction: Throwing toys in high-dimensions

Giving up analytically known form



Low-dim histogram: Poisson likelihood computed exactly

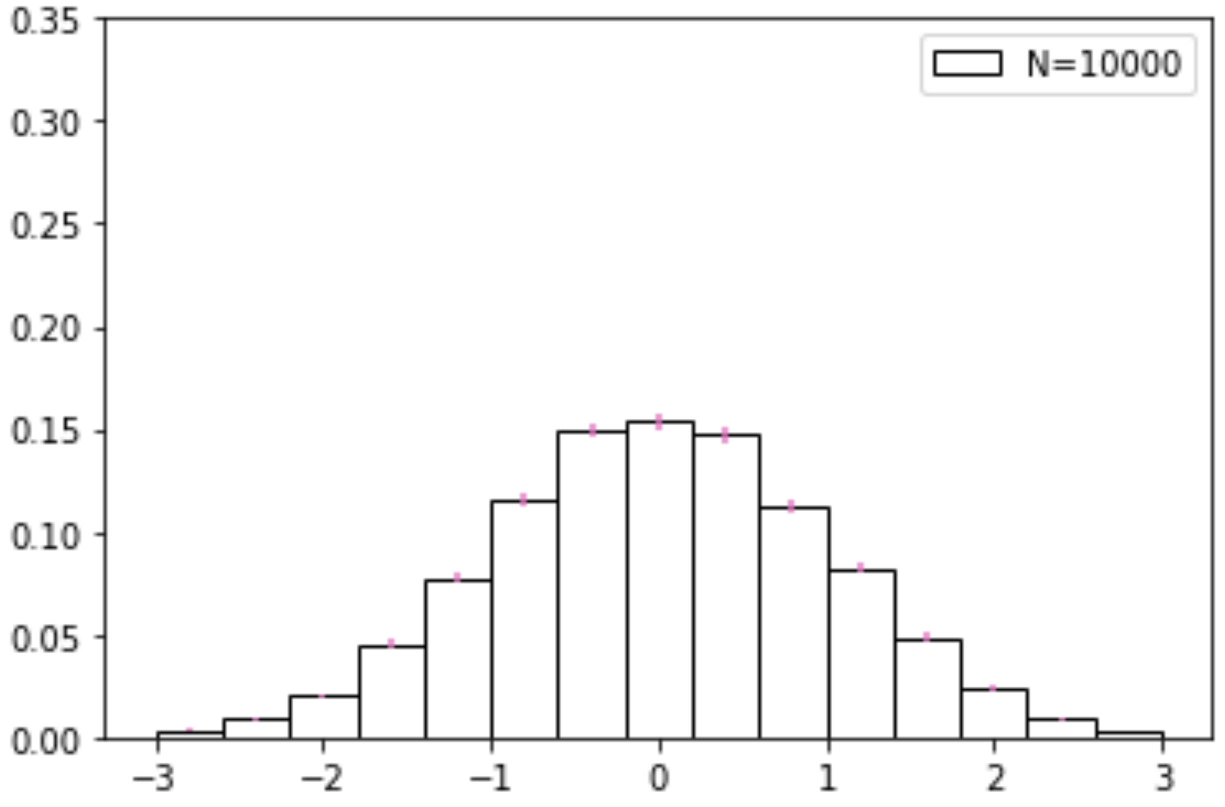
$$P(N_{obs} = k \,|\, N_{exp} = \lambda) = \frac{\lambda^k e^{-k}}{k!}$$



High-dim, unbinned NSBI:
Merely an estimation of the likelihood

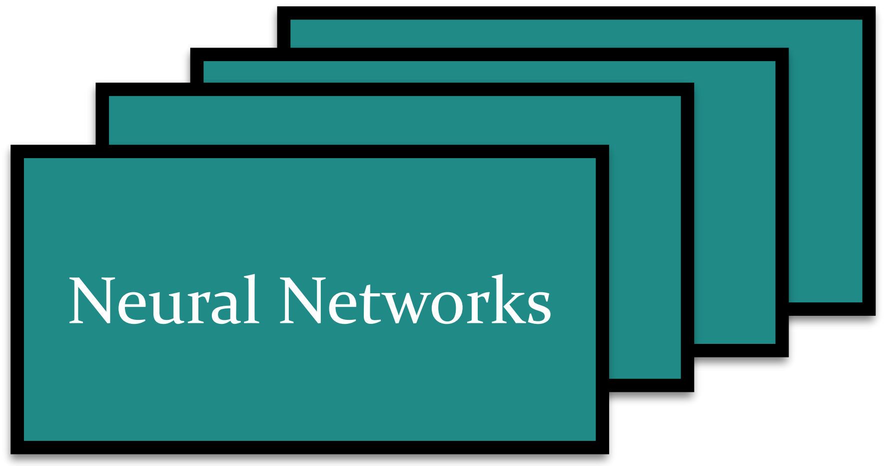
What if the network is
overconfident ?

Giving up analytically known form



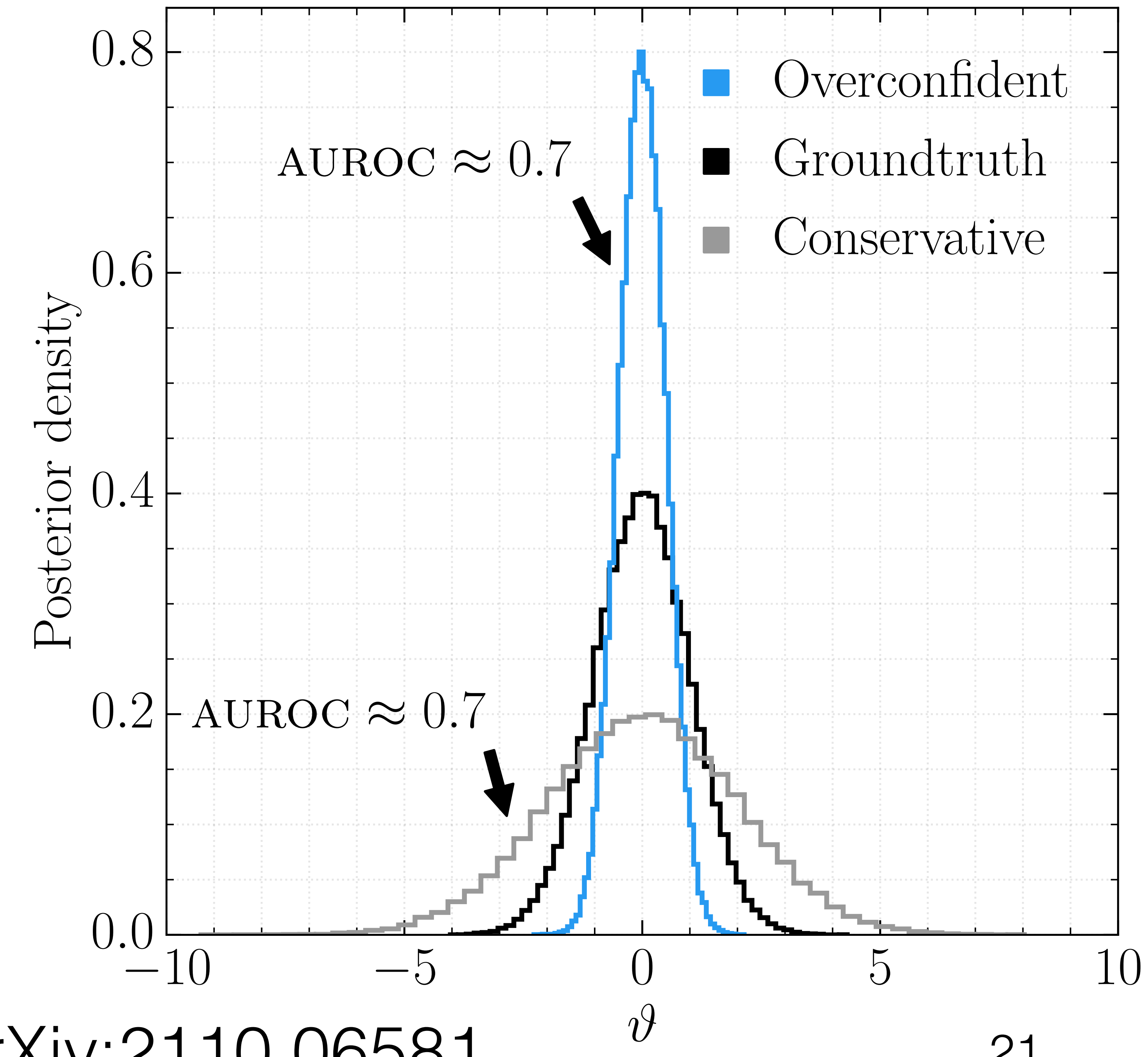
Low-dim histogram: Poisson likelihood computed exactly

$$P(N_{obs} = k \,|\, N_{exp} = \lambda) = \frac{\lambda^k e^{-k}}{k!}$$



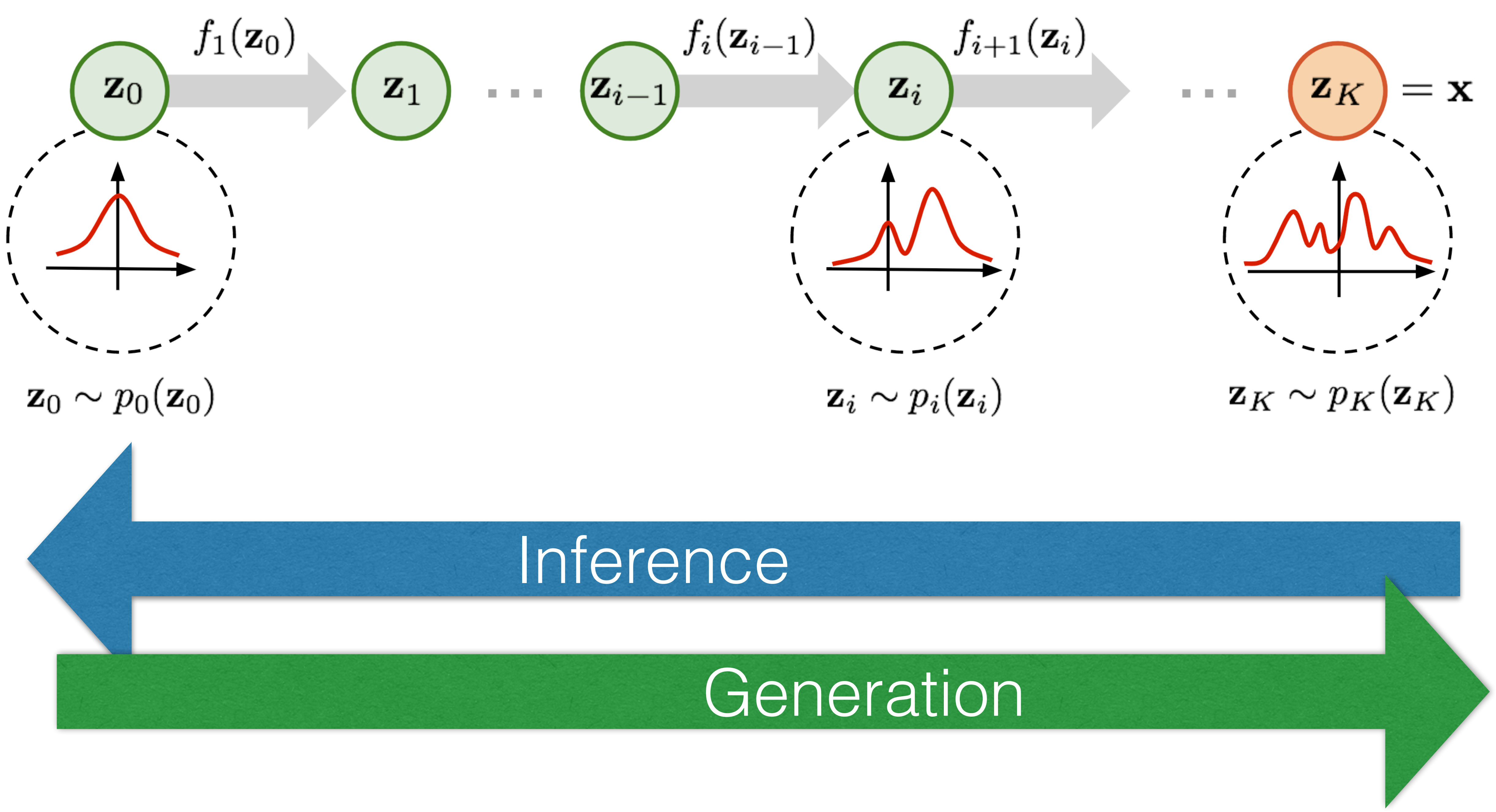
High-dim, unbinned NSBI:
Merely an estimation of the likelihood

What if the network is
overconfident ?



Hermans et al: arXiv:2110.06581

Diagnostic Checks

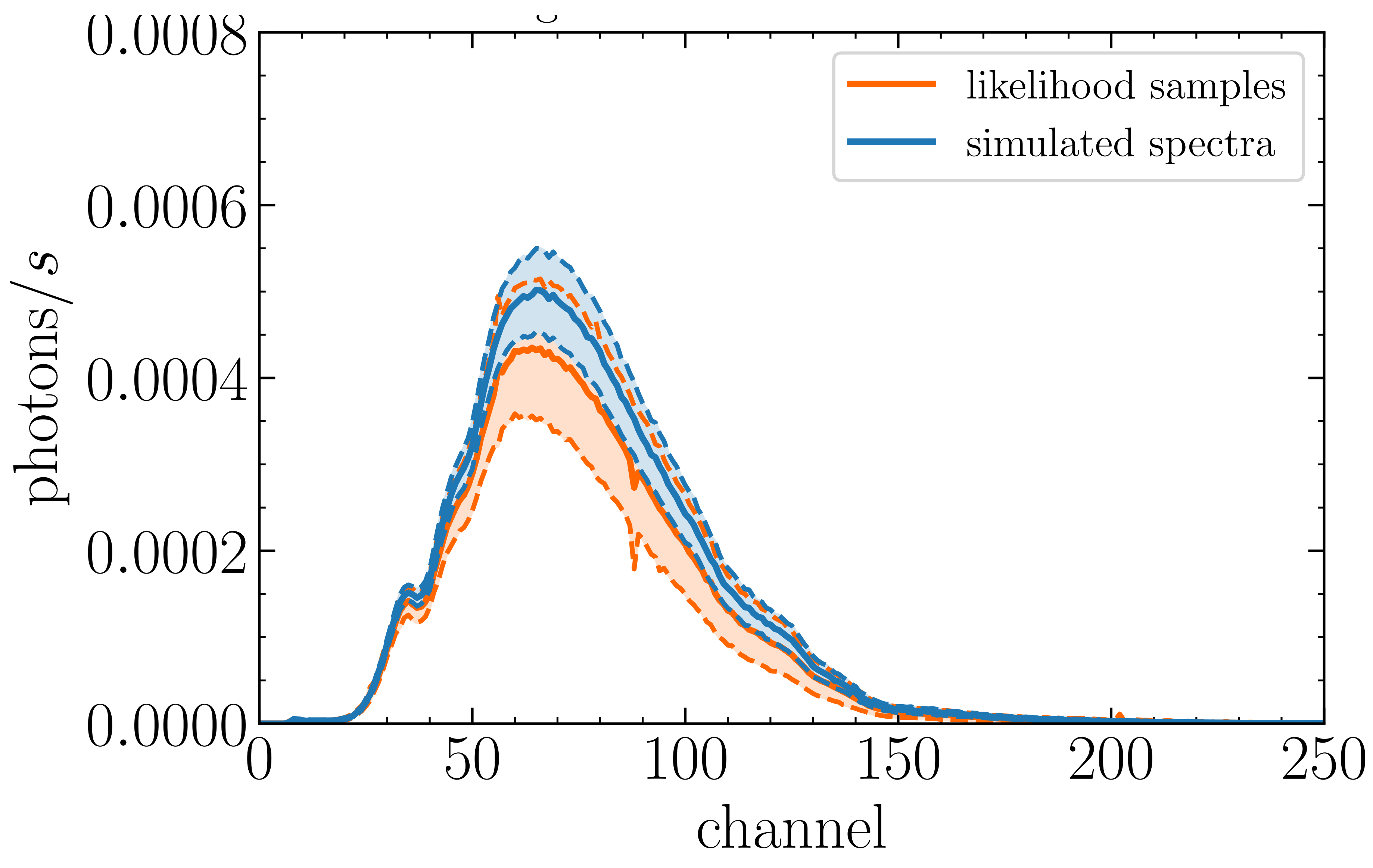


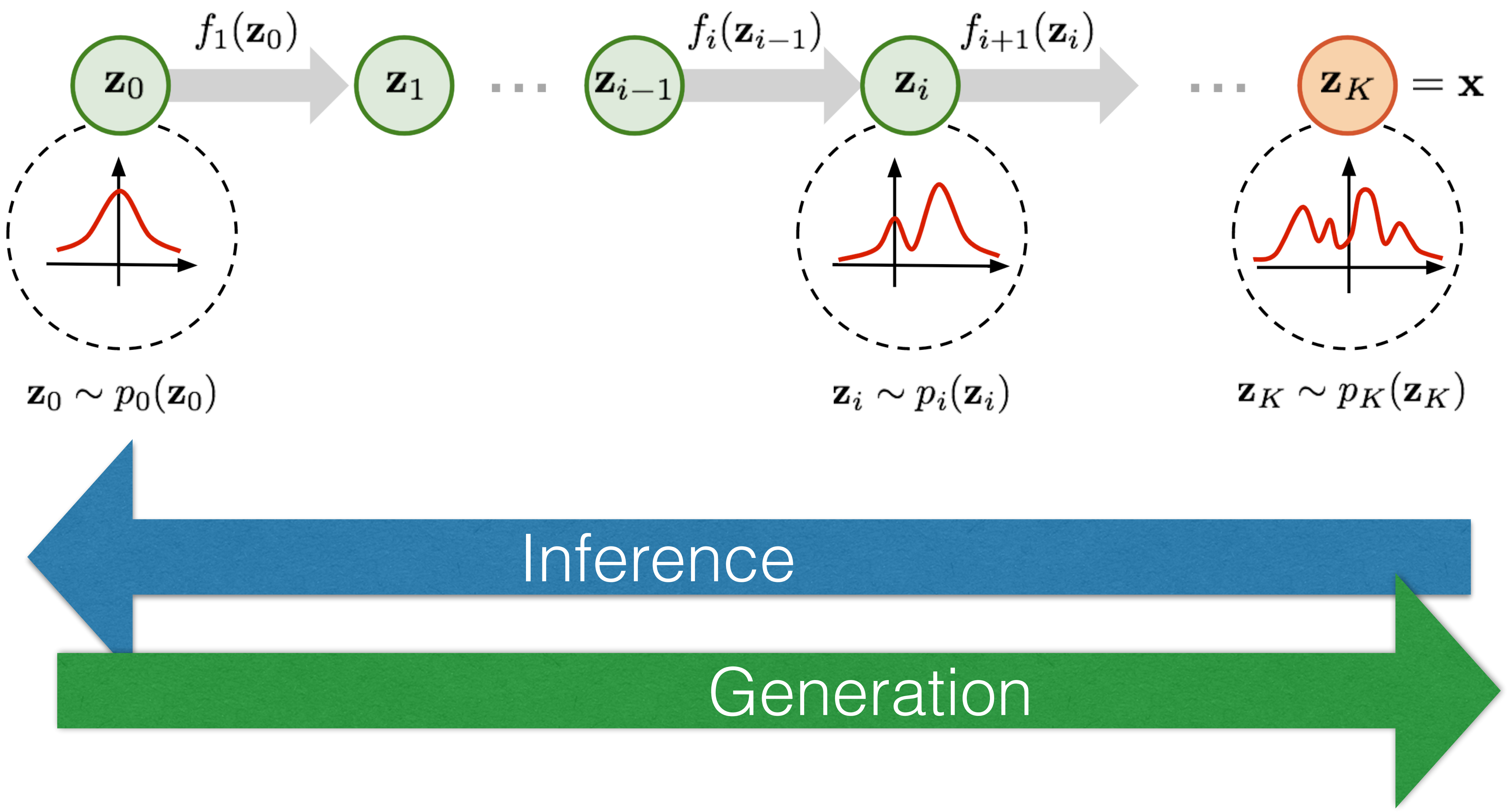
Diagnostic Checks

For neural likelihood estimation, run the normalising flow backwards, as a generative model and visualise!

Find areas of mismodelling the likelihood

What about neural ratio estimation ?





Diagnostic Checks

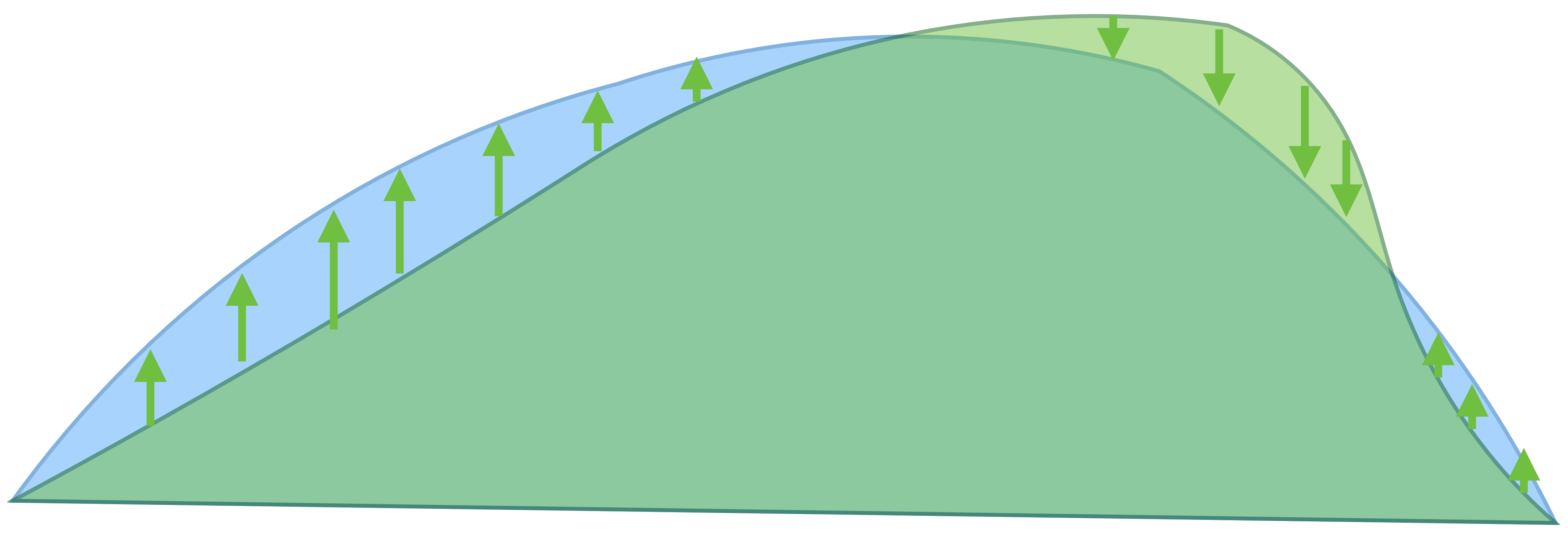
For neural likelihood estimation, run the normalising flow backwards, as a generative model and visualise!

Find areas of mismodelling the likelihood

What about neural ratio estimation ?

Validate quality of LR estimation with re-weighting task

Reweighting: Calculate weights w_i for events x_i in green sample to match blue sample

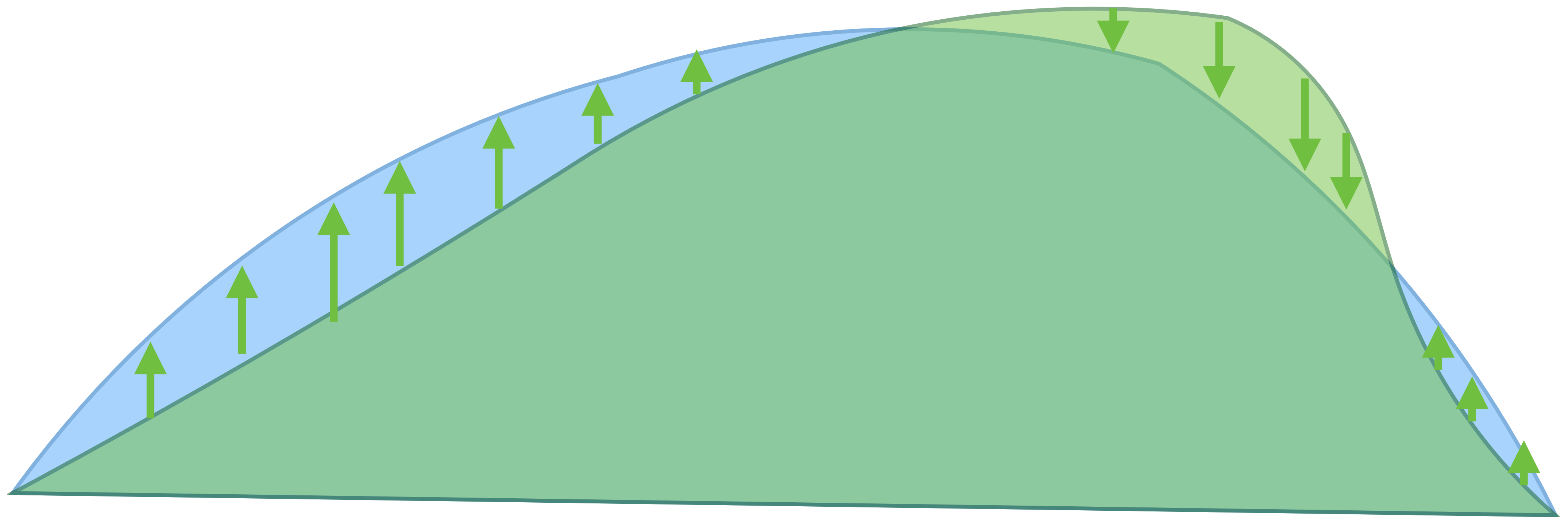


Validate quality of LR estimation with re-weighting task

Reweighting: Calculate weights w_i for events x_i in green sample to match blue sample

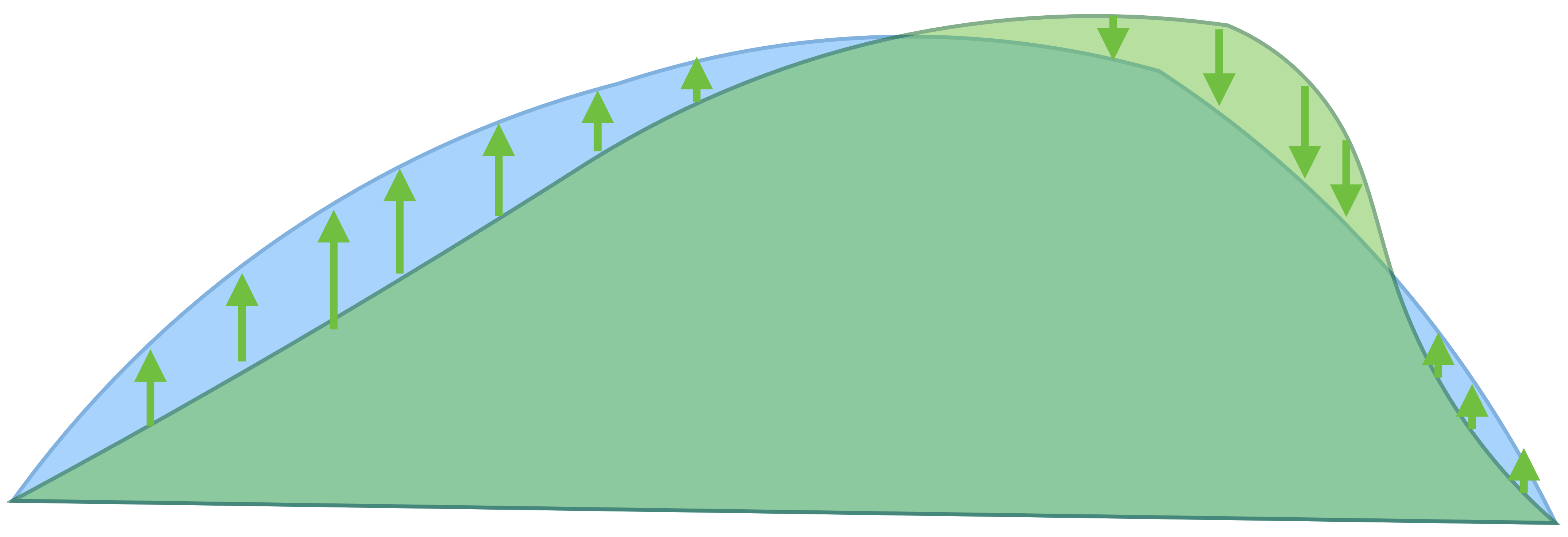
$$w_i = \frac{P(x_i | \theta_0)}{P(x_i | \theta_1)}$$

Already estimated using classifiers

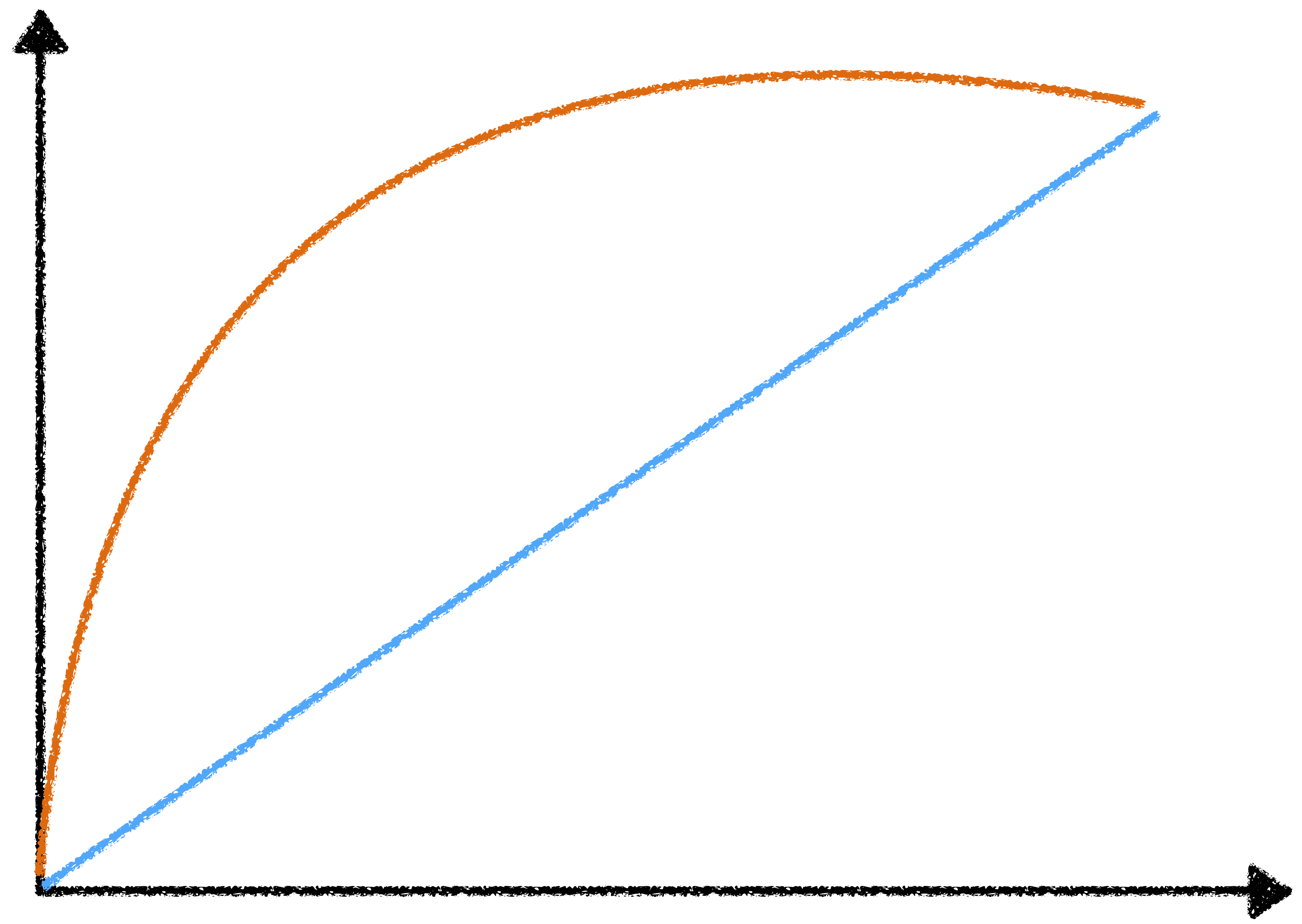


Re-weight diagnostics

One-dimensional visualisations



High-dimensional classifier test

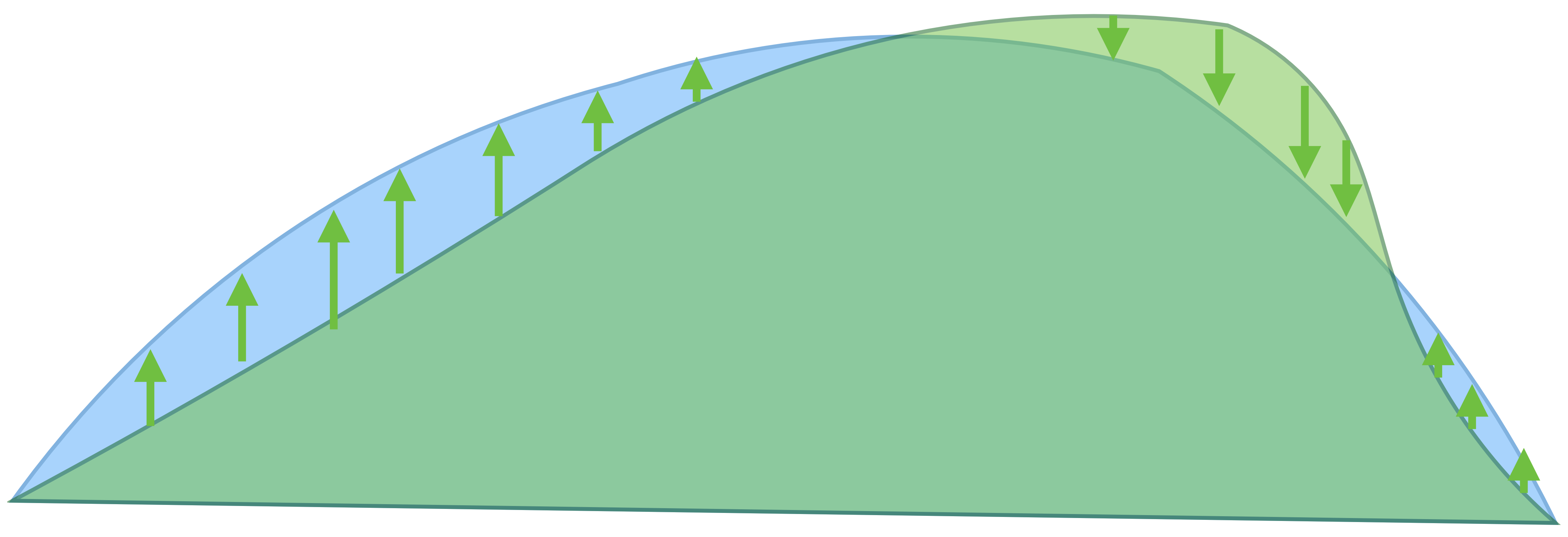


ROC Curve

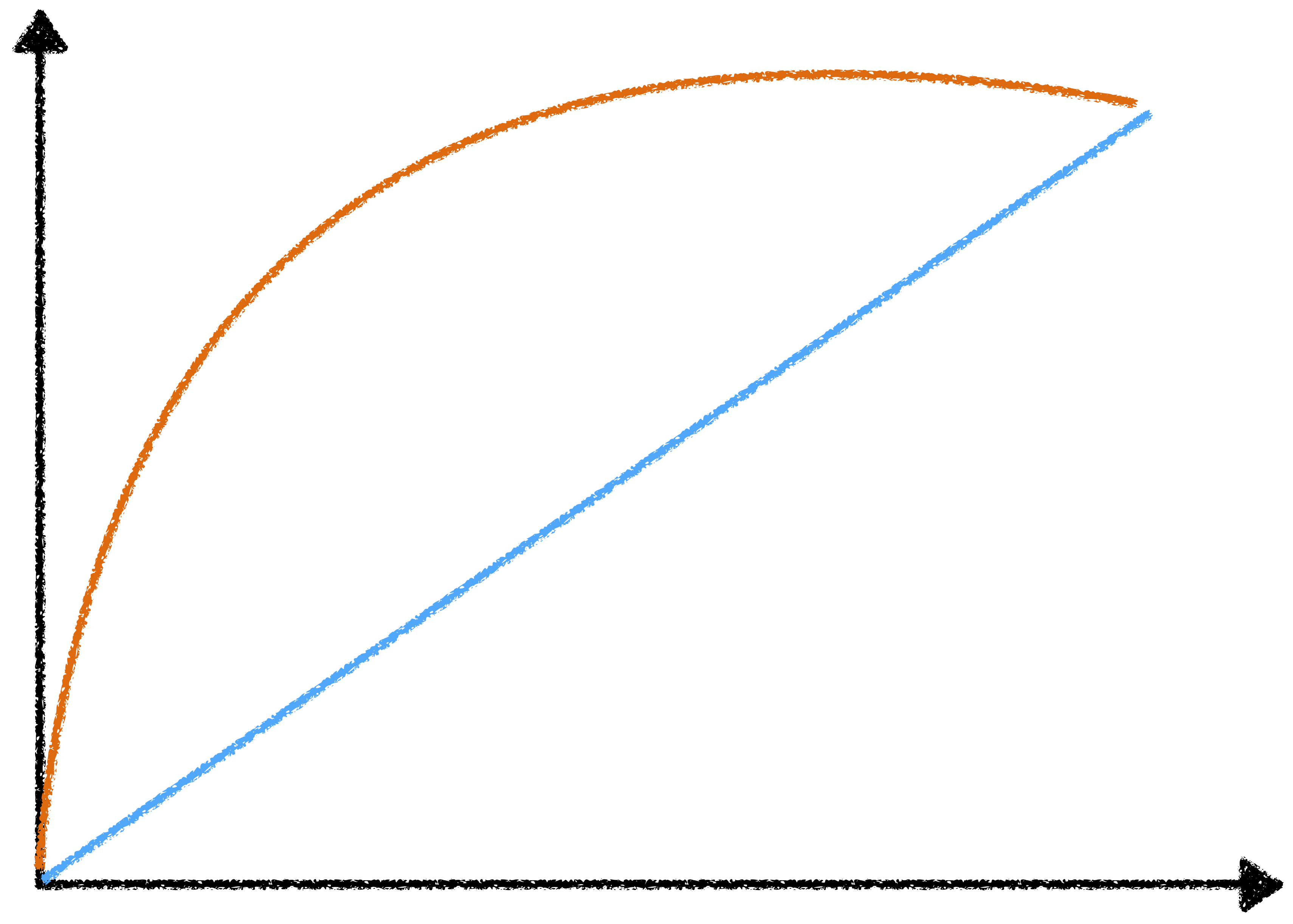
Independent classifier trained to separate re-weighted vs target

Re-weight diagnostics

One-dimensional visualisations



High-dimensional classifier test

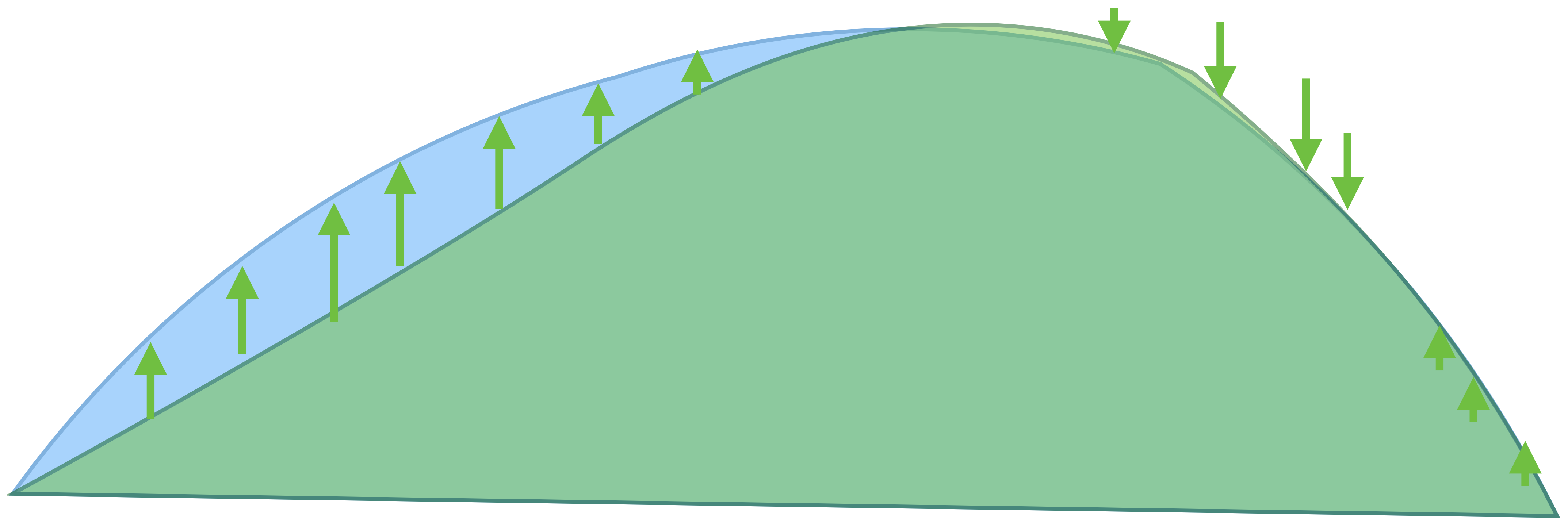


ROC Curve

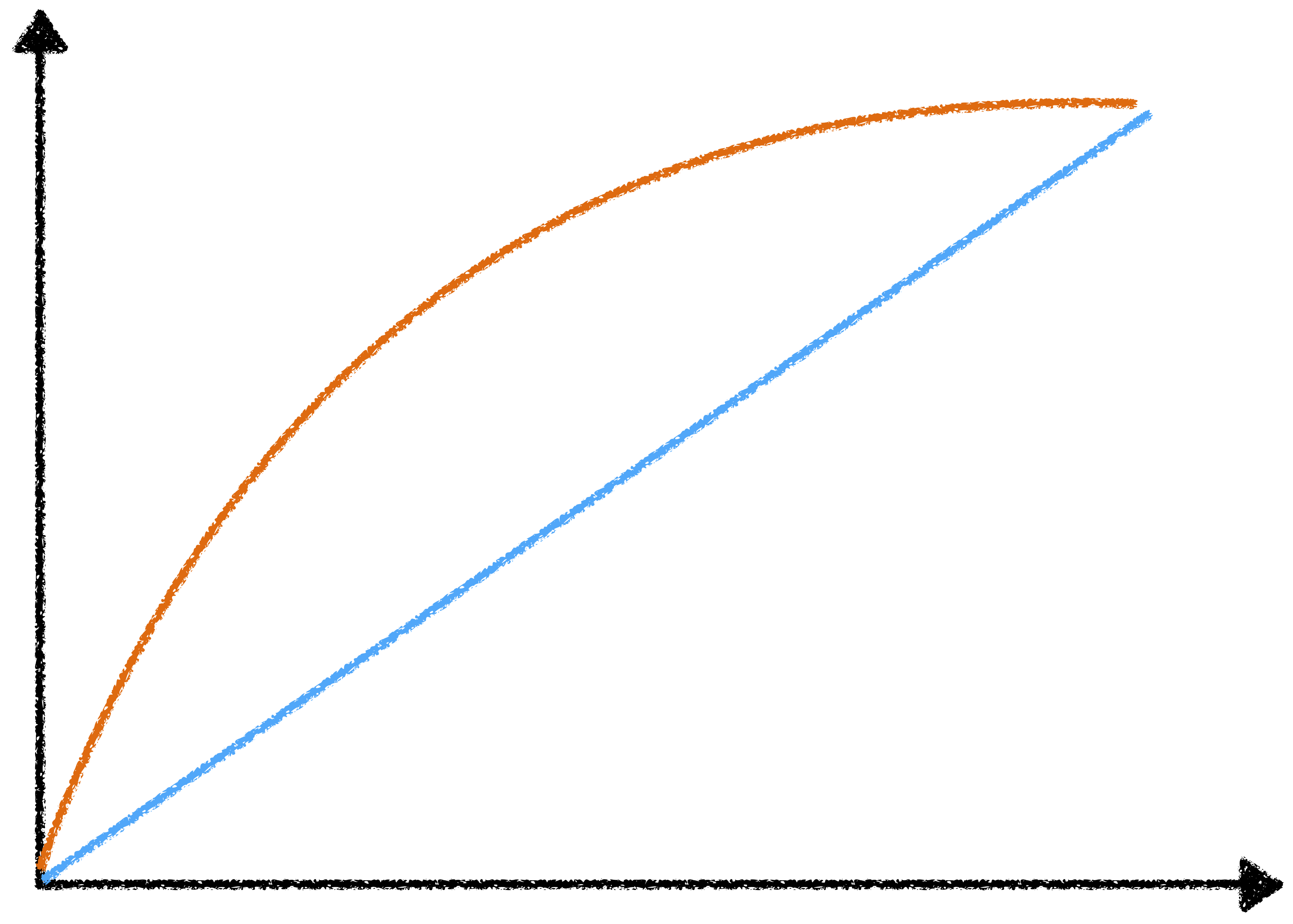
Independent classifier trained to separate re-weighted vs target

Re-weight diagnostics

One-dimensional visualisations



High-dimensional classifier test

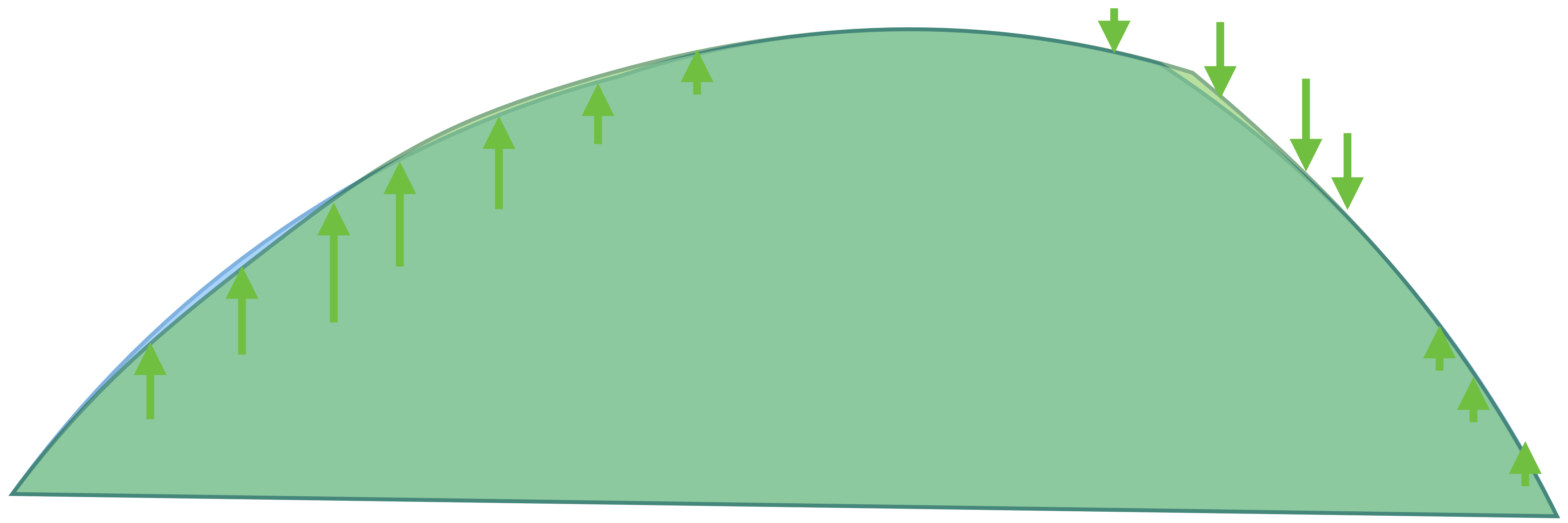


ROC Curve

Independent classifier trained to separate re-weighted vs target

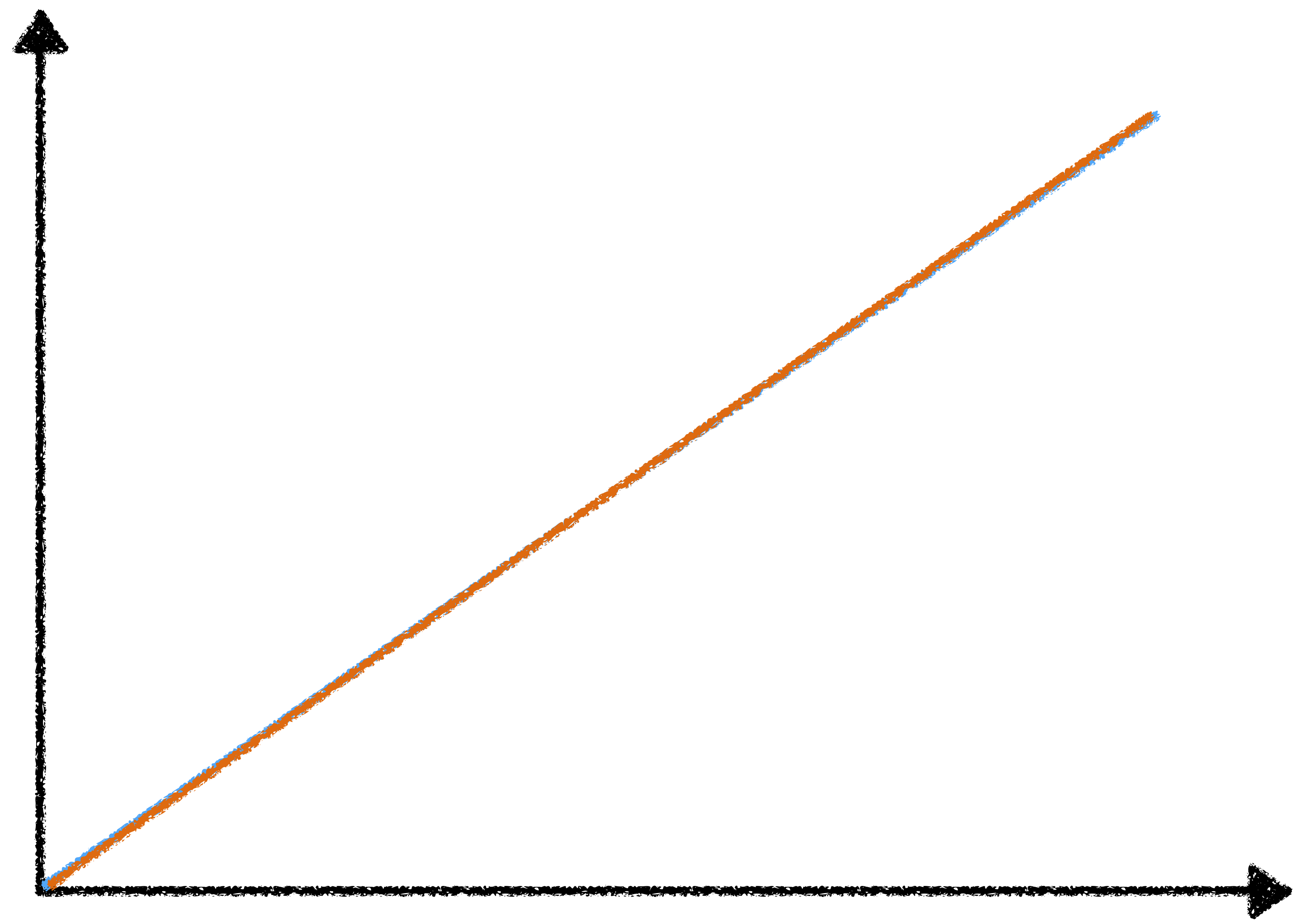
Re-weight diagnostics

One-dimensional visualisations



$$w_i = \frac{P(x_i | \theta_0)}{P(x_i | \theta_1)}$$

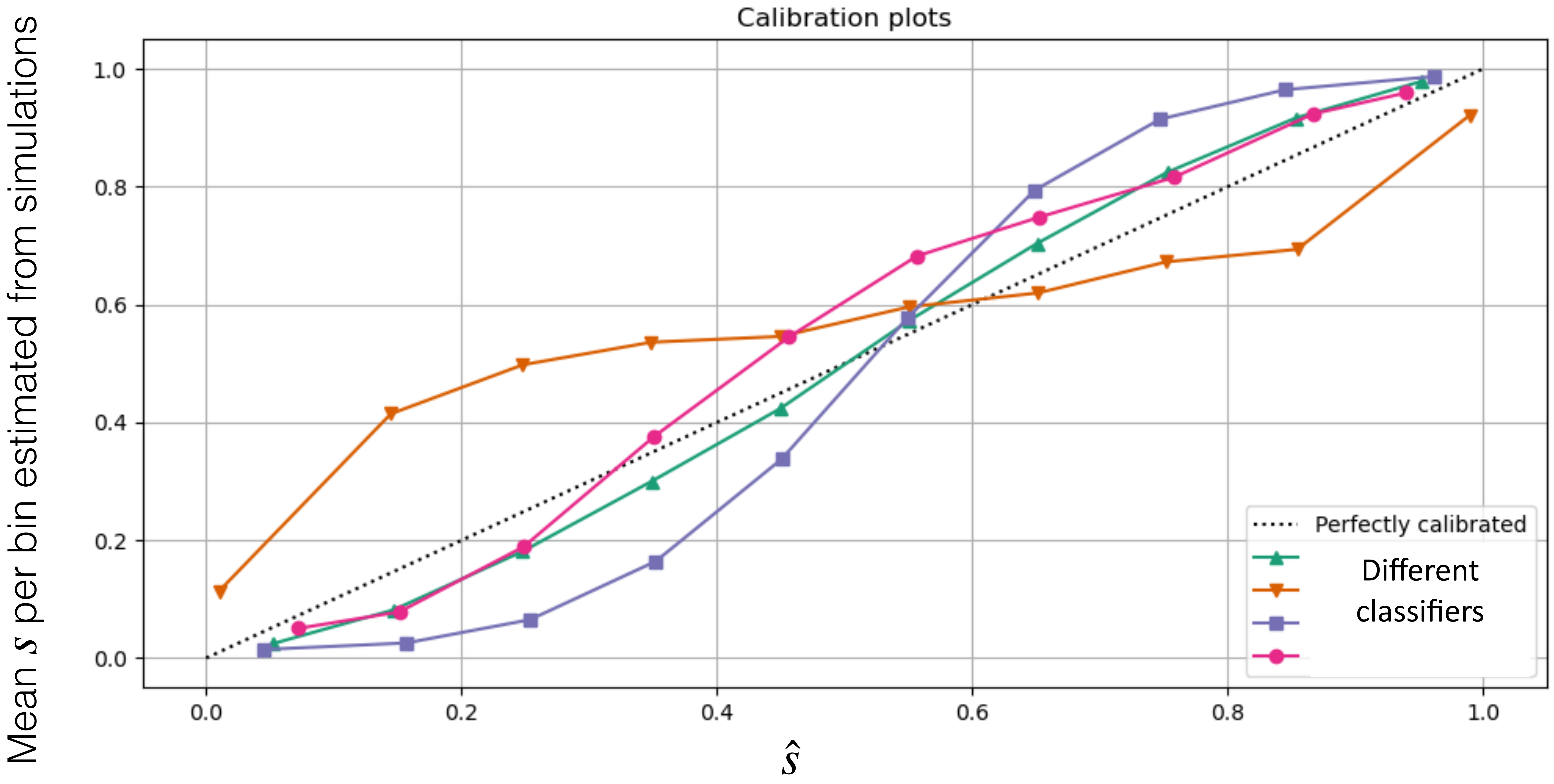
High-dimensional classifier test



ROC Curve

Independent classifier trained to separate re-weighted vs target

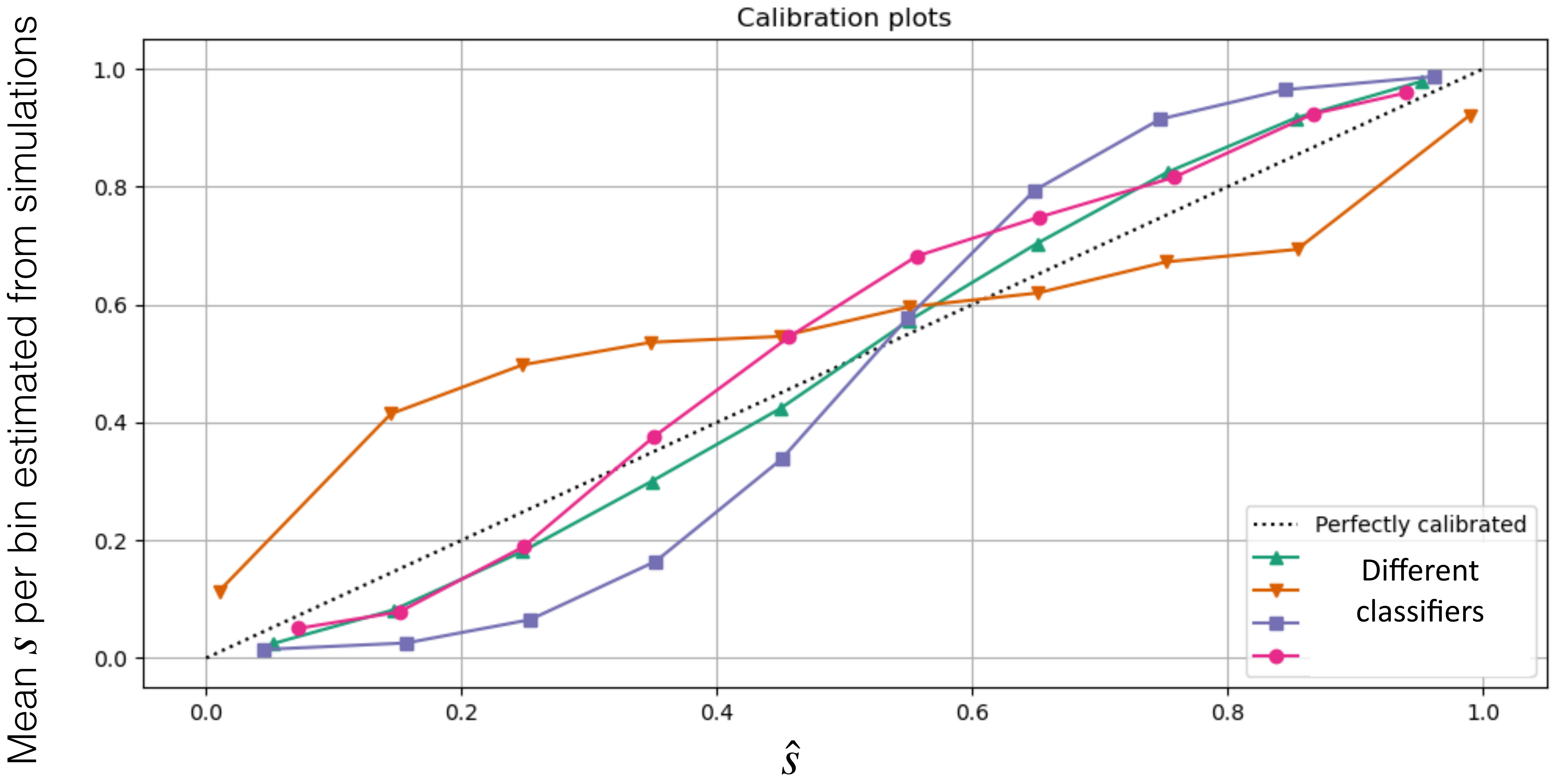
Calibration Curves



Recall, classifier was trained to estimate:

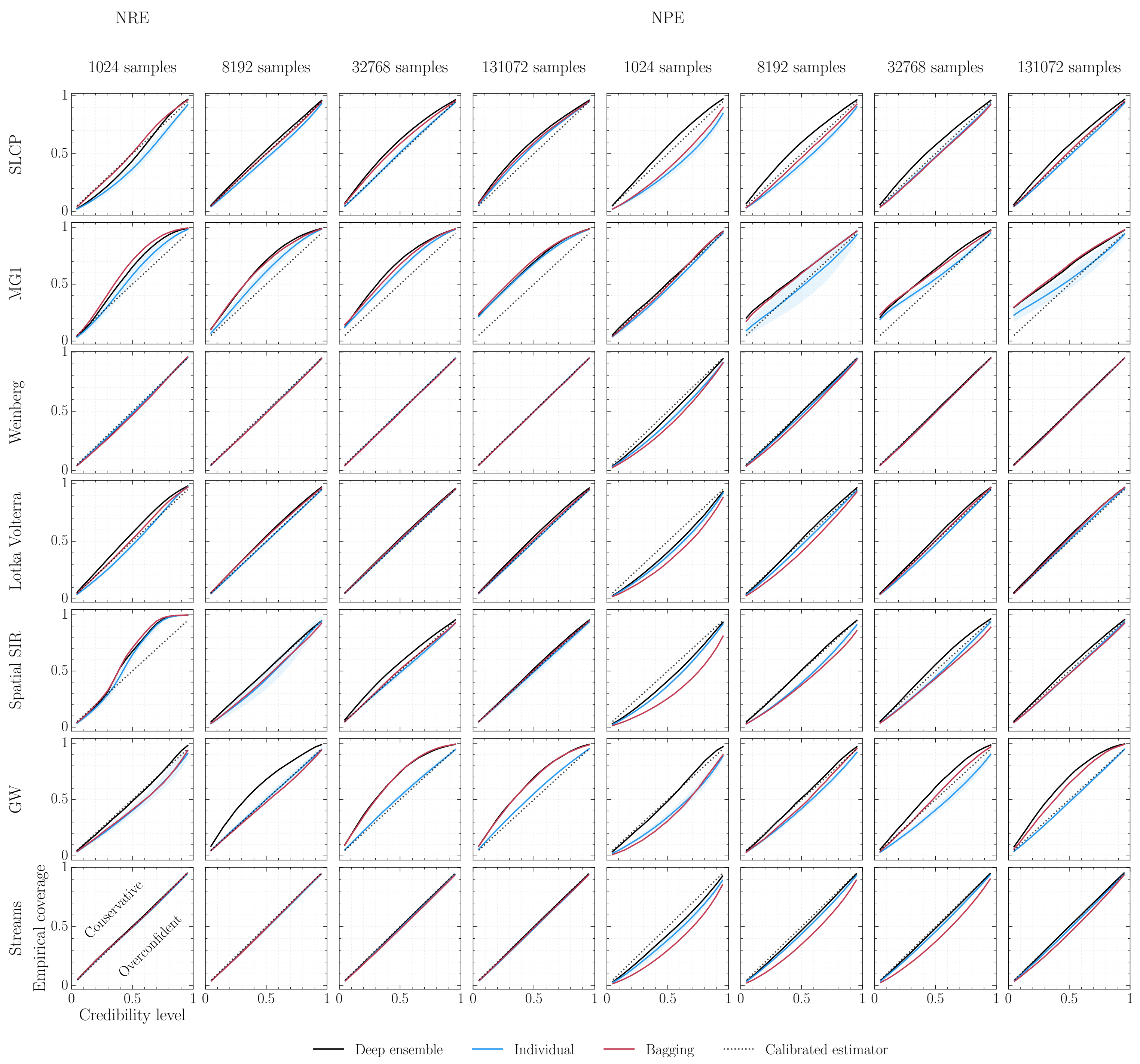
$$s(x_i) = \frac{p(x_i \mid \theta_0)}{p(x_i \mid \theta_0) + p(x_i \mid \theta_1)}$$

Calibration Curves



Recall, classifier was trained to estimate:

$$s(x_i) = \frac{p(x_i\,|\,\theta_0)}{p(x_i\,|\,\theta_0) + p(x_i\,|\,\theta_1)}$$

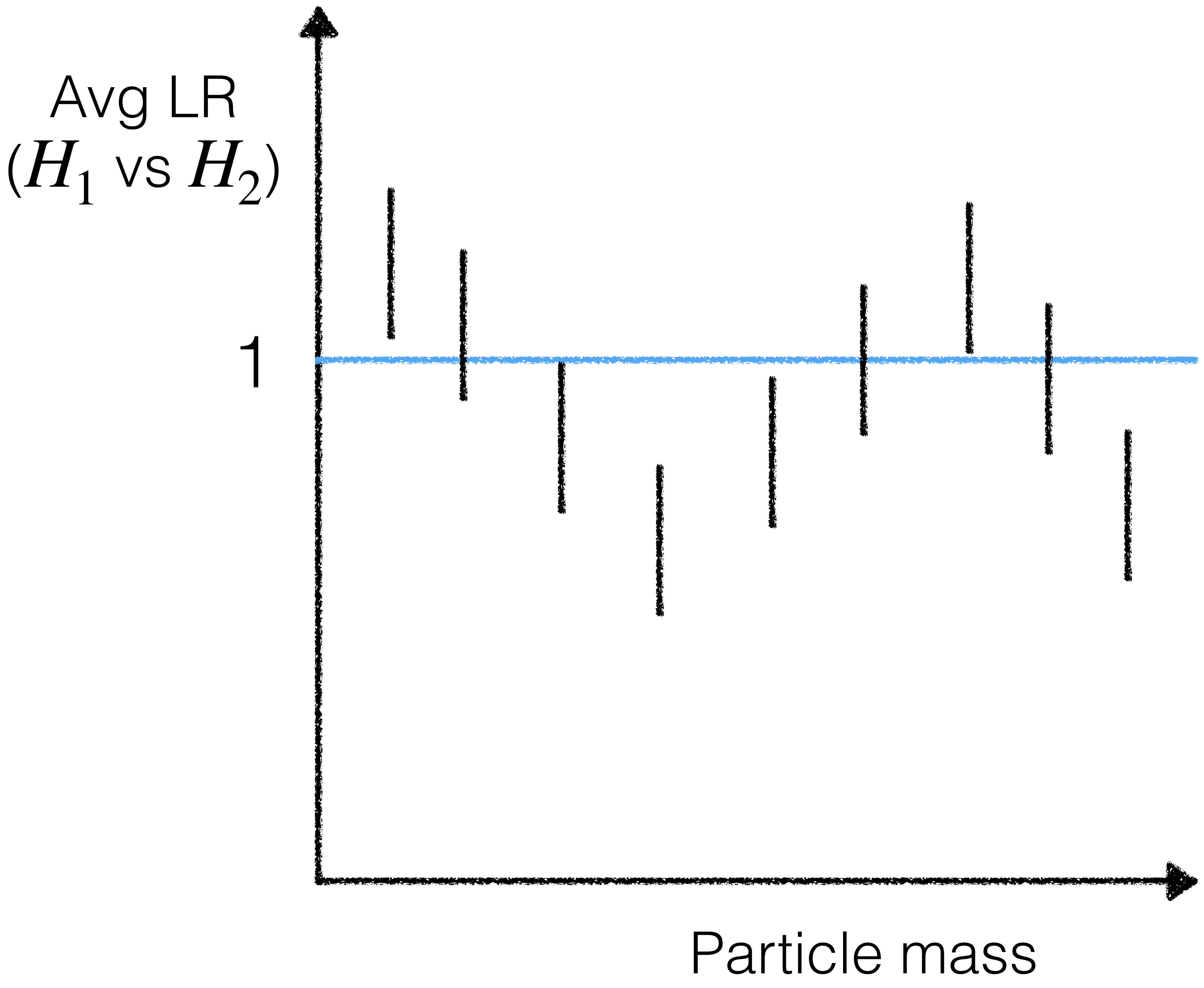


Similar tests possibly for many NSBI methods, see Hermans et al

NSBI also provides new tools to inspect my data & analysis

Which events favour my hypothesis, which don't ?

Can go down to inspecting the contribution of each individual event!



Parameterisation challenge

Parameterising networks and validating them

Network can learn a function parameterised in:

- parameter of interest θ (eg. W Mass)
- nuisance parameters z (eg. Jet energy scale)

Questions:

- How do we validate it for the full space of $\{\theta, z\}$?

Cranmer et al

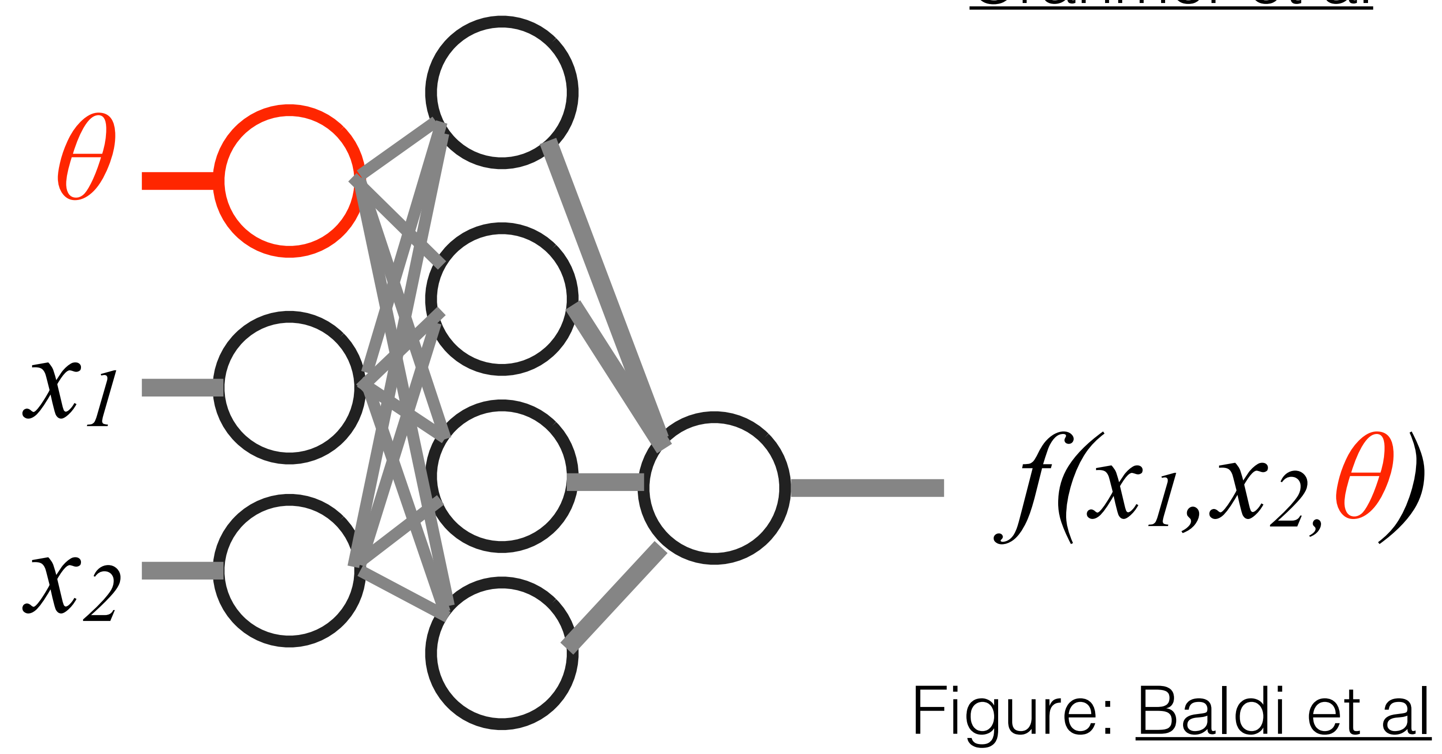


Figure: Baldi et al

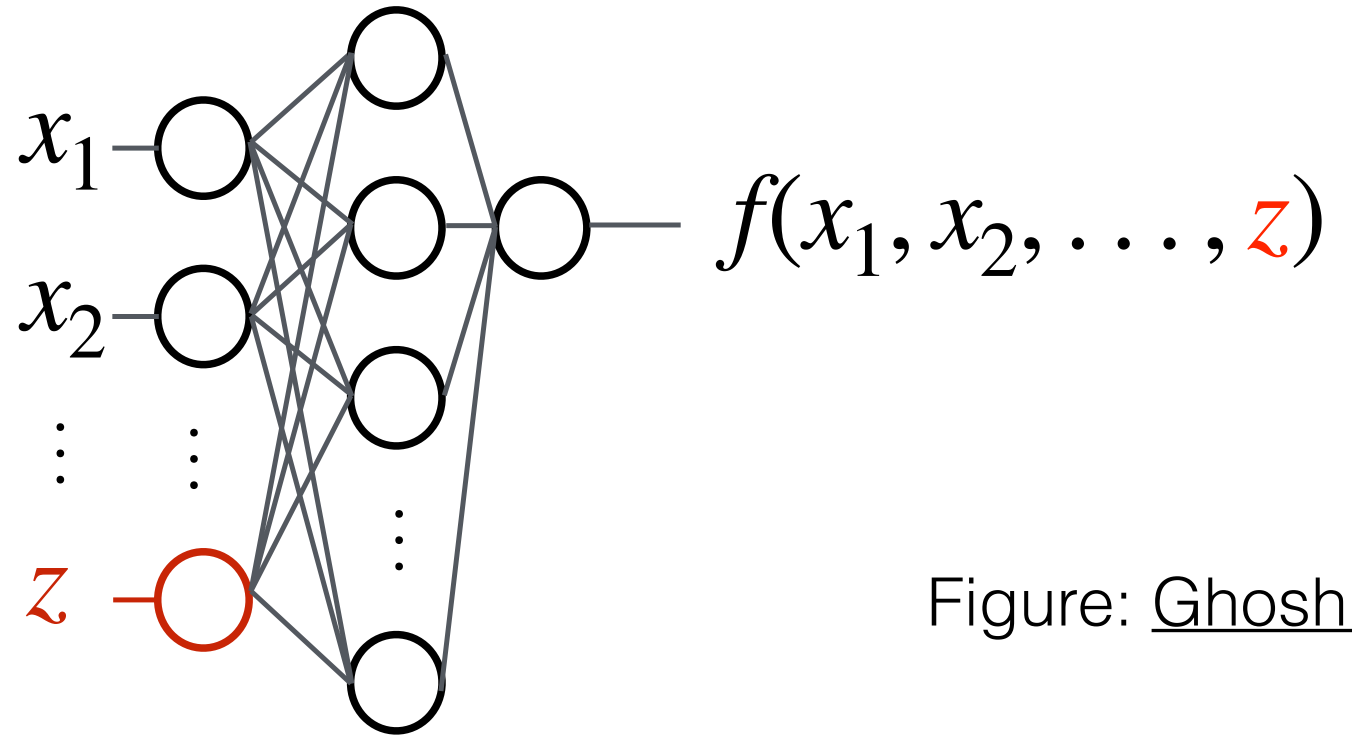


Figure: Ghosh et al

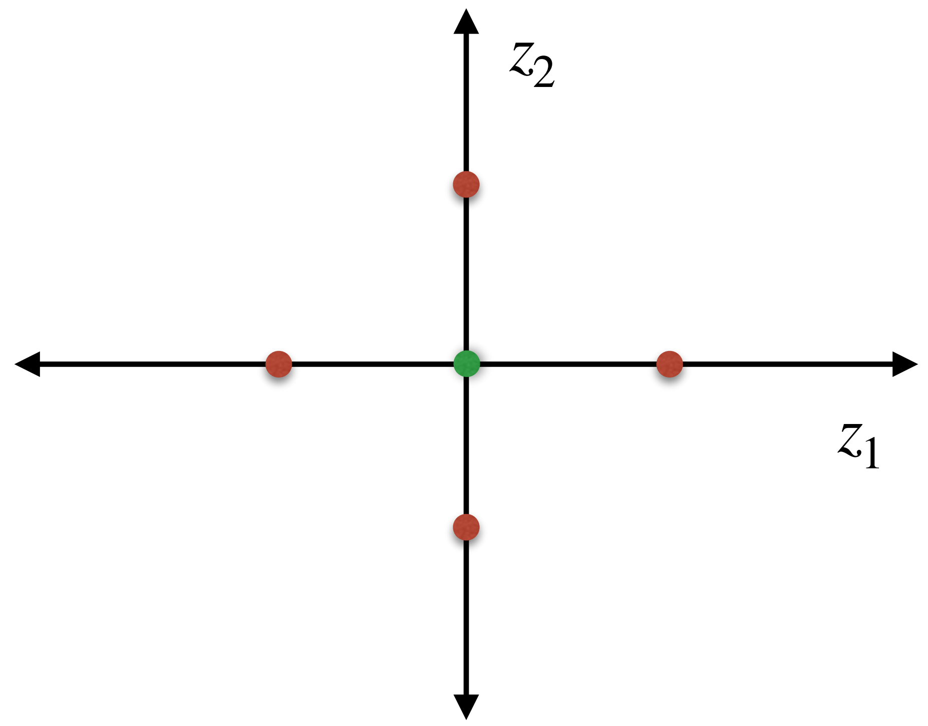
Parameterising networks and validating them

Network can learn a function parameterised in:

- parameter of interest θ (eg. W Mass)
- nuisance parameters z (eg. Jet energy scale)

Questions:

- How do we validate it for the full space of $\{\theta, z\}$?
- How to parameterise on nuisance parameters for which we only have 3 examples ?



Cranmer et al

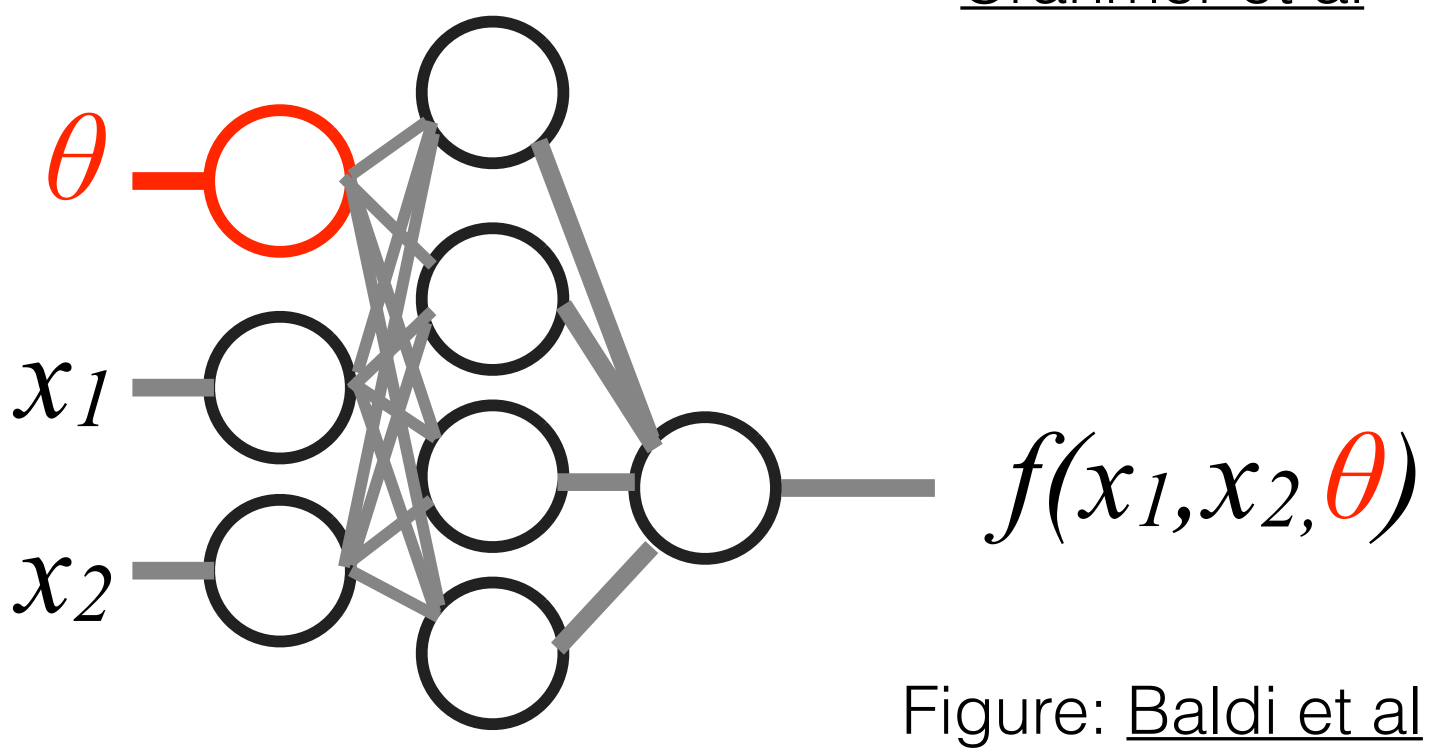


Figure: Baldi et al

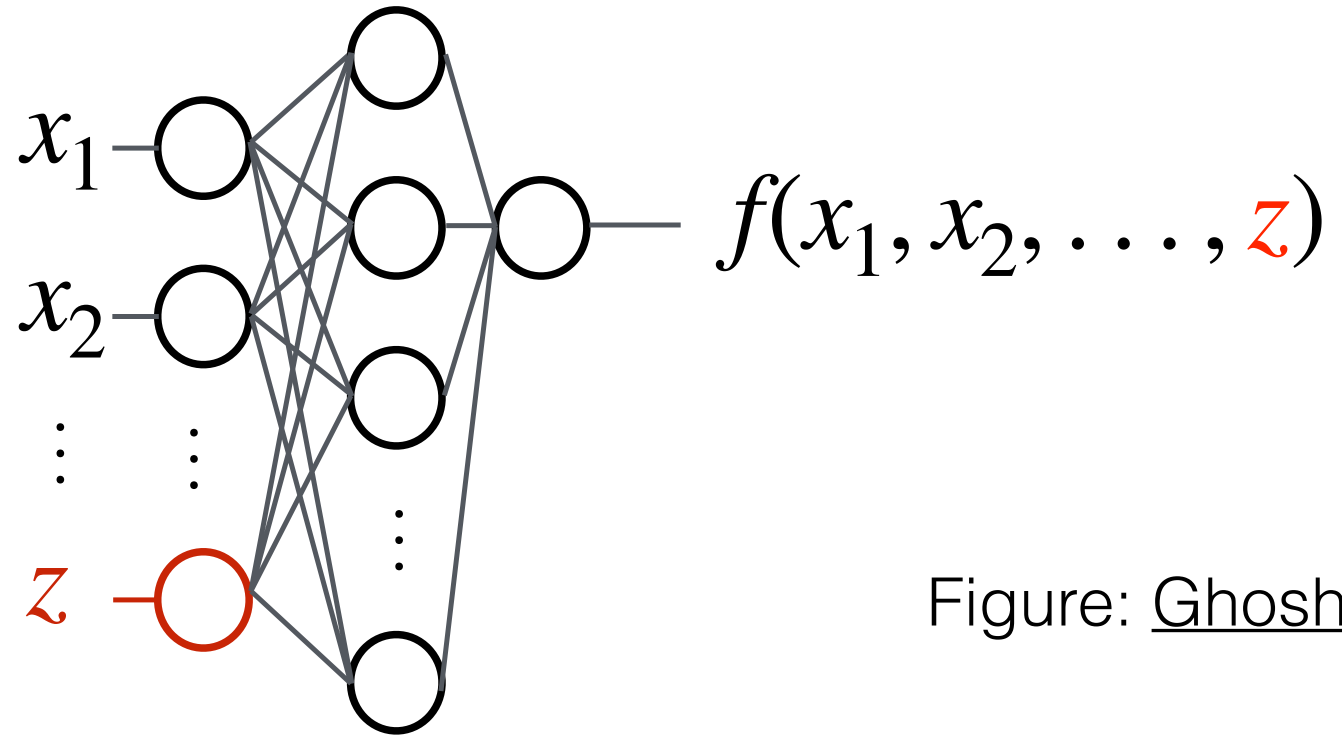


Figure: Ghosh et al

Parameterising networks and validating them

Network can learn a function parameterised in:

- parameter of interest θ (eg. W Mass)
- nuisance parameters z (eg. Jet energy scale)

Questions:

- How do we validate it for the full space of $\{\theta, z\}$?
- How to parameterise on nuisance parameters for which we only have 3 examples ?

Cranmer et al

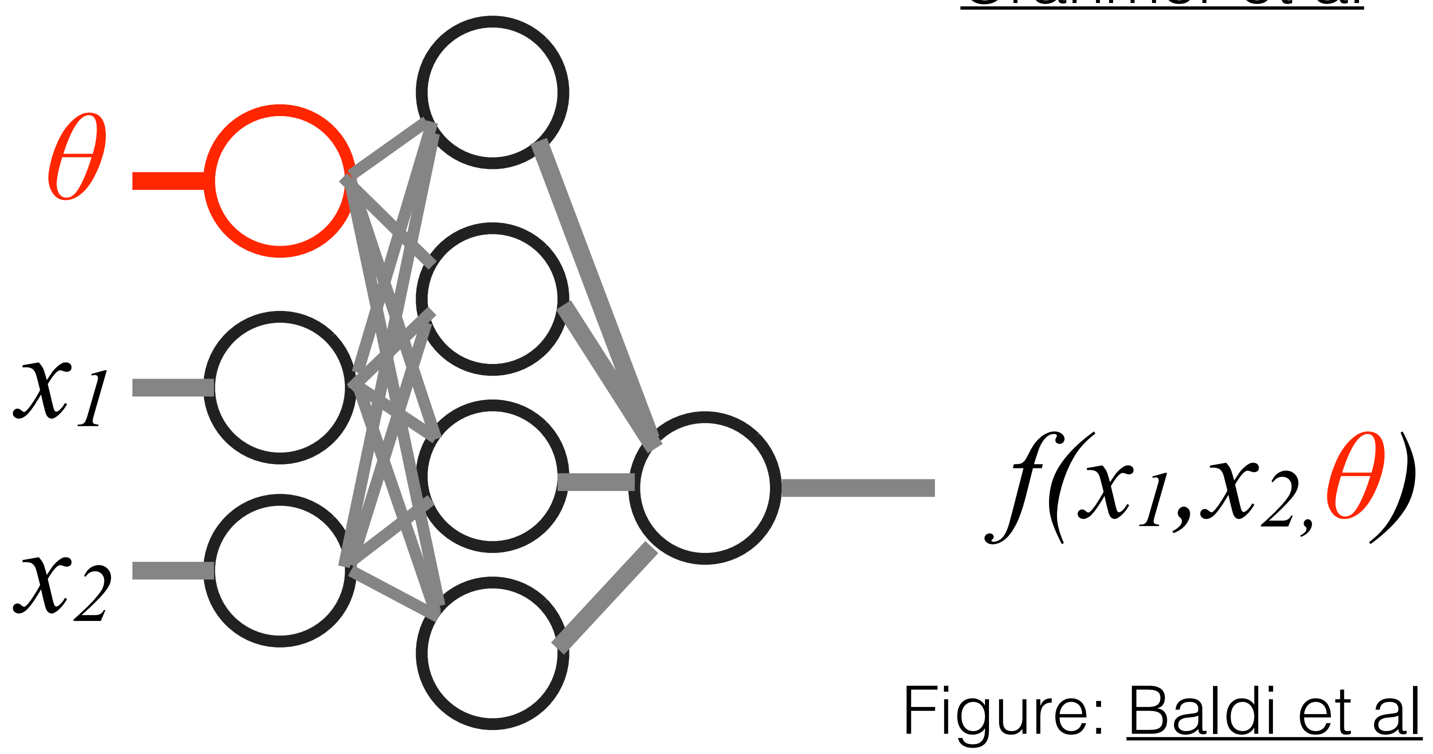


Figure: Baldi et al

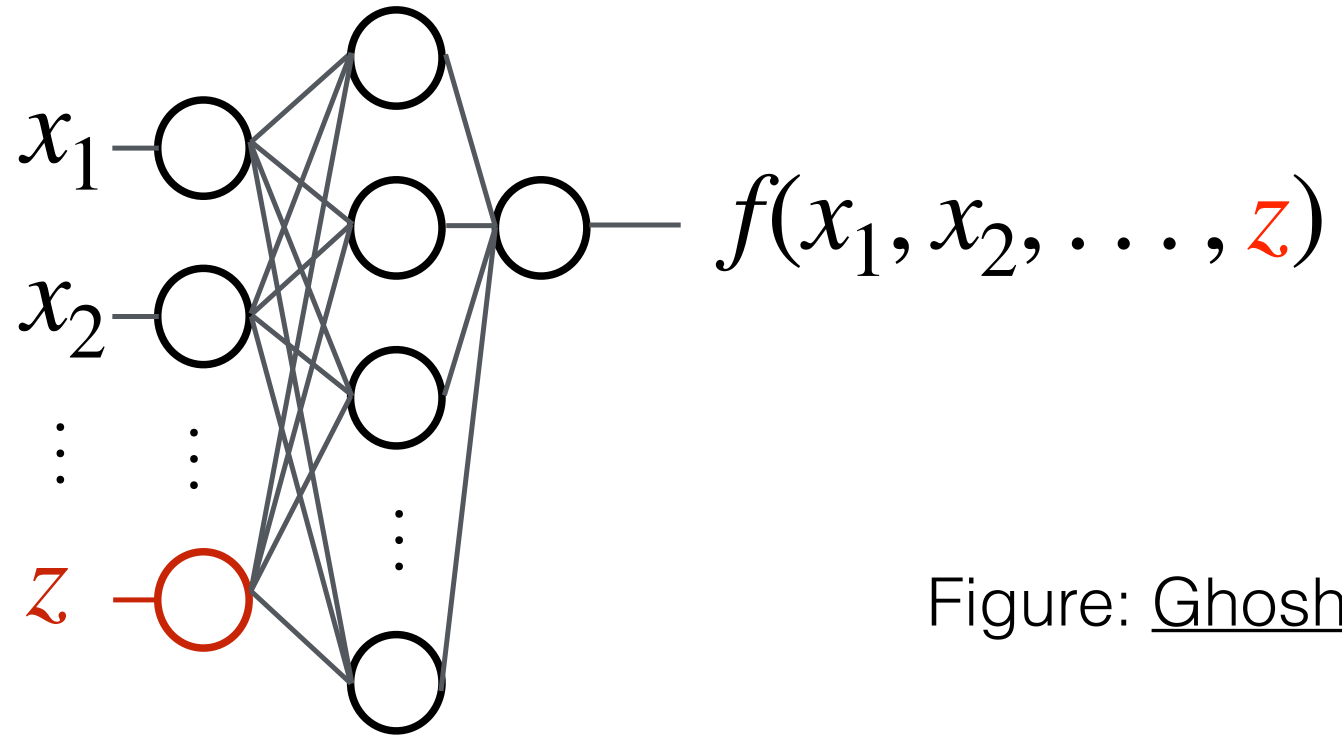
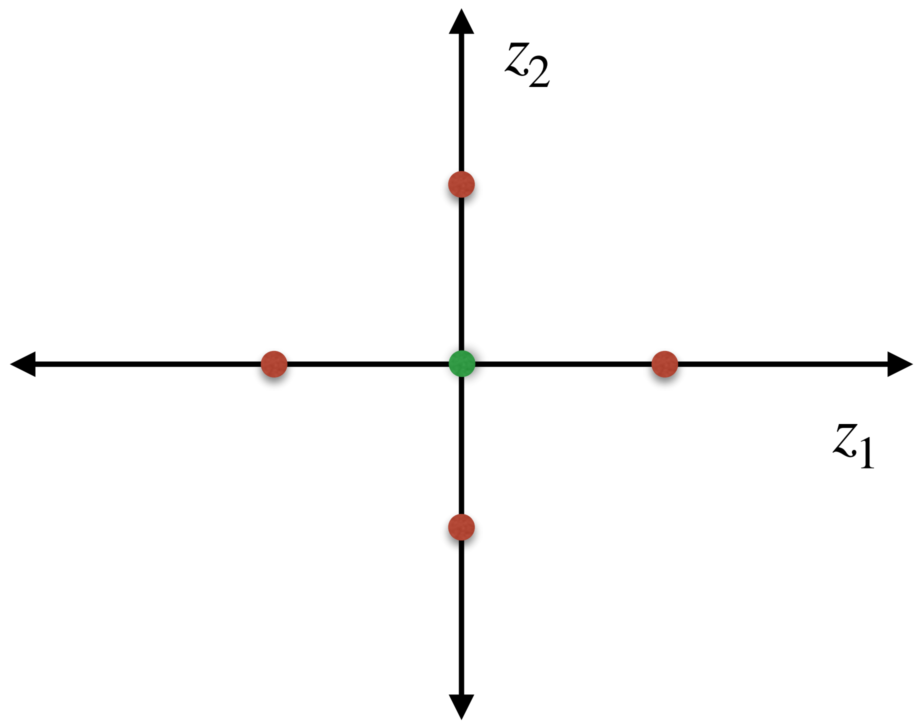


Figure: Ghosh et al

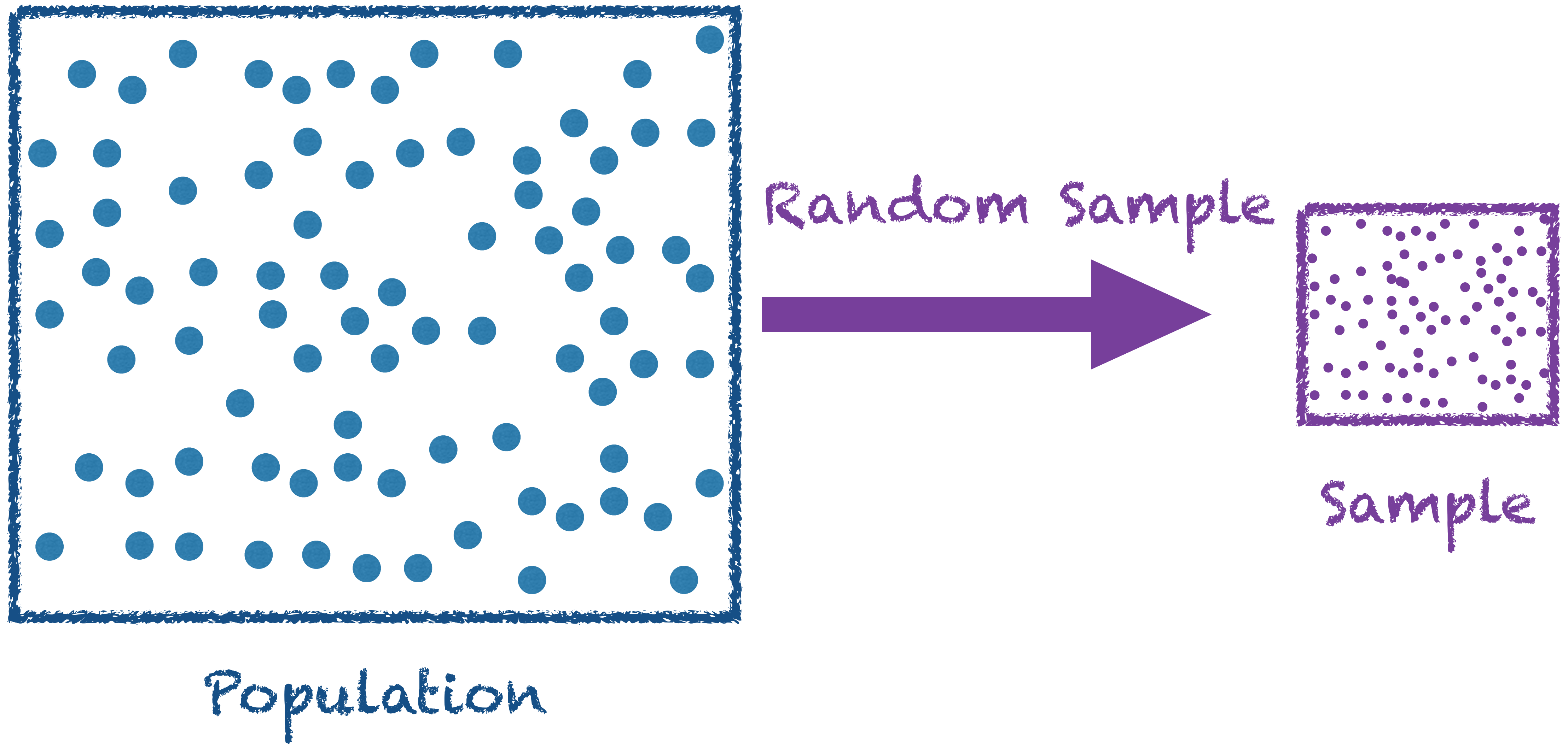


If these questions interest you, come chat with me!

Uncertainties in the likelihood estimation: Training statistics & random initialisation

Estimating the variance on mean: Bootstrapping

Want to estimate mean of population



Re-Sample
with
replacement

Image: [Source]

Estimating the variance on mean: Bootstrapping

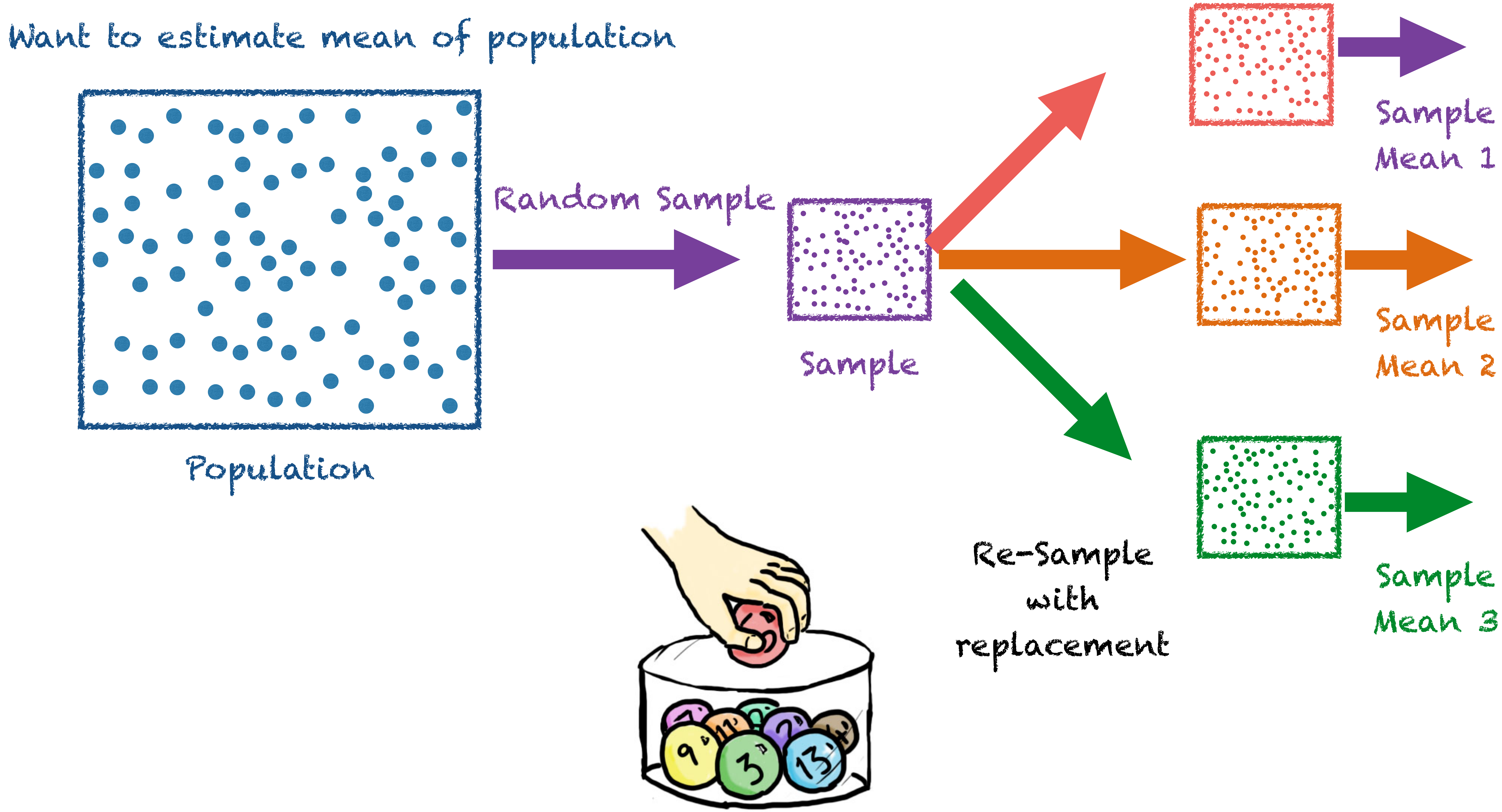


Image: [Source]

Estimating the variance on mean: Bootstrapping

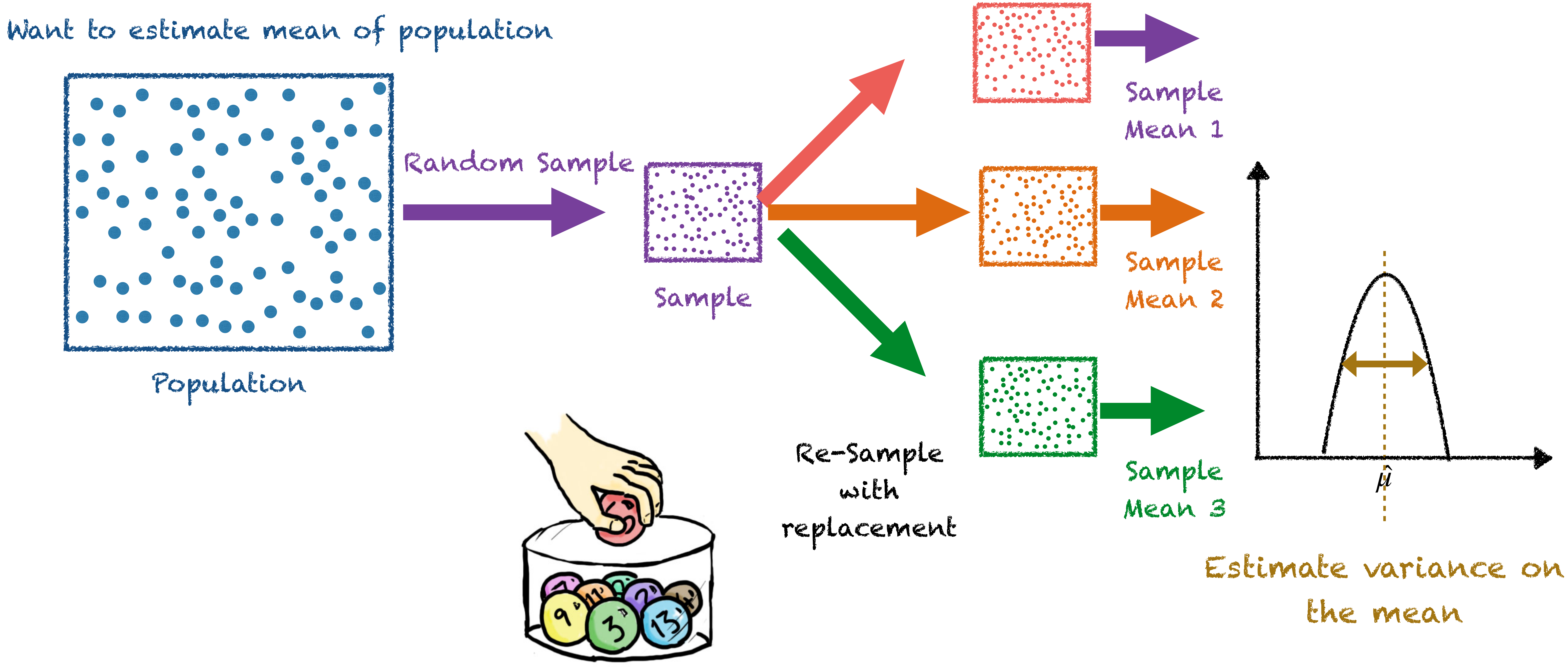


Image: [Source]

Quantifying uncertainty on estimated likelihood..

- Train an ensemble of networks, **each on a bootstrapped version of the training dataset**
    - Or Bayesian networks ? [Delaunoy et al, arXiv2408.15136]

- The spread in their prediction provides the uncertainty due to limited training statistics, and random network initialisation

- **Ensemble average used as final prediction**, so what's the uncertainty on that ?
    - Too expensive to train thousands of ensembles
    - Create bootstrapped ensembles ?
        - Each network trained on bootstrapped training dataset ?

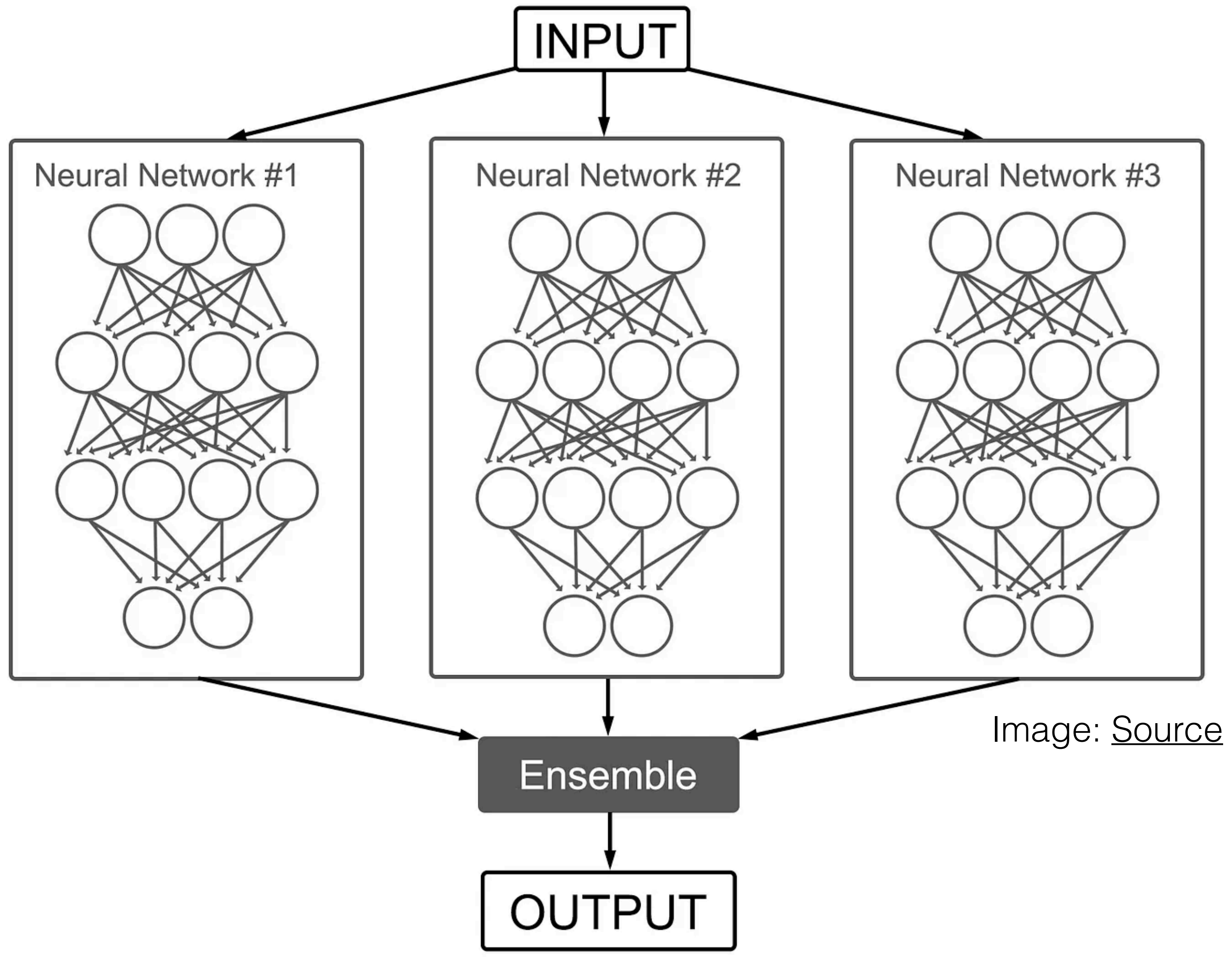


Image: Source

Quantifying uncertainty on estimated likelihood..

- Train an ensemble of networks, **each on a bootstrapped version of the training dataset**
    - Or Bayesian networks ? [Delaunoy et al, arXiv2408.15136]

- The spread in their prediction provides the uncertainty due to limited training statistics, and random network initialisation

- **Ensemble average used as final prediction**, so what's the uncertainty on that ?
    - Too expensive to train thousands of ensembles
    - Create bootstrapped ensembles ?
        - Each network trained on bootstrapped training dataset ?

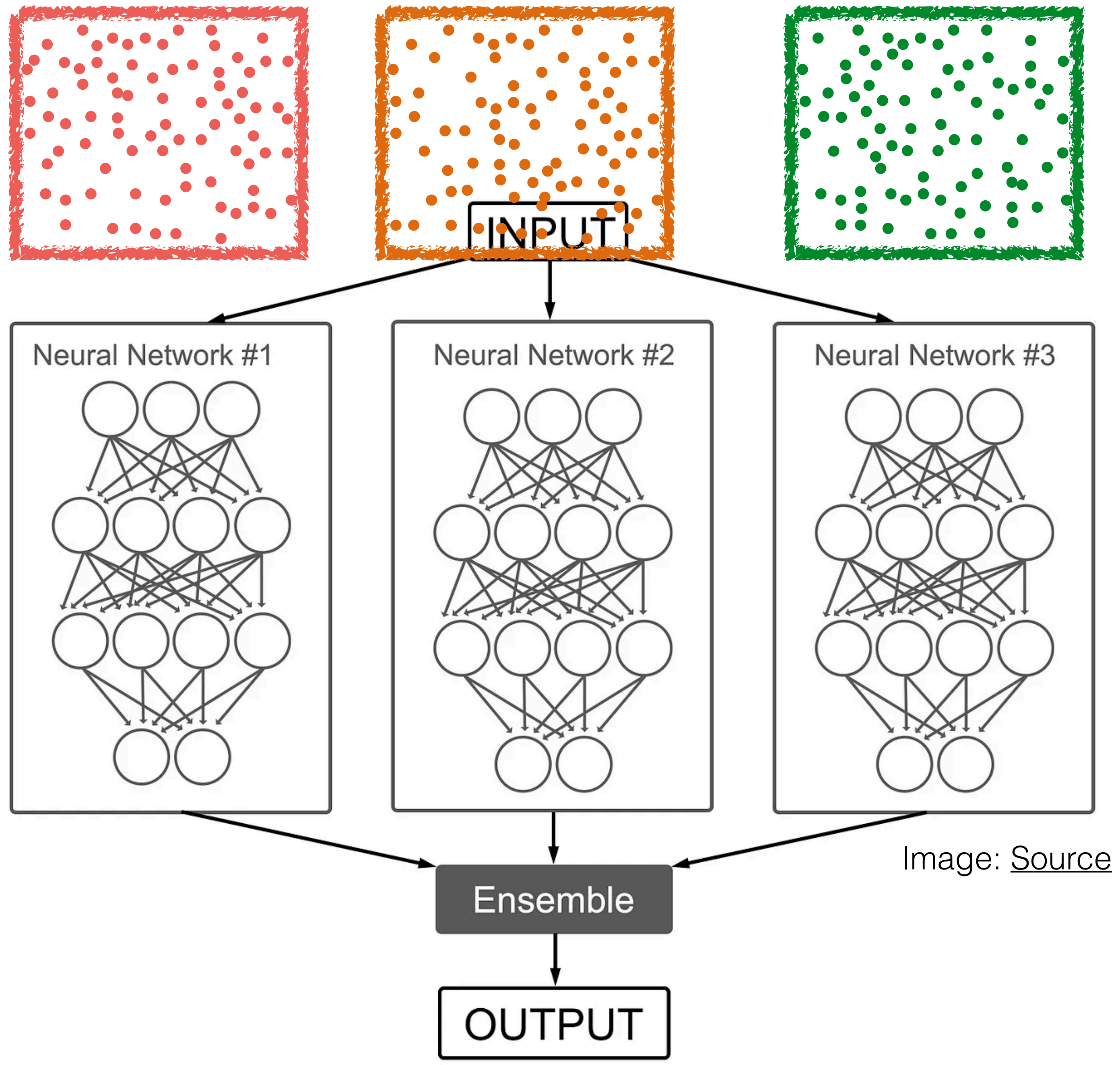


Image: Source

Quantifying uncertainty on estimated likelihood..

- Train an ensemble of networks, **each on a bootstrapped version of the training dataset**
    - Or Bayesian networks ? [Delaunoy et al, arXiv2408.15136]

- The spread in their prediction provides the uncertainty due to limited training statistics, and random network initialisation

- **Ensemble average used as final prediction**, so what's the uncertainty on that ?
    - Too expensive to train thousands of ensembles
    - Create bootstrapped ensembles ?
        - Each network trained on bootstrapped training dataset ?

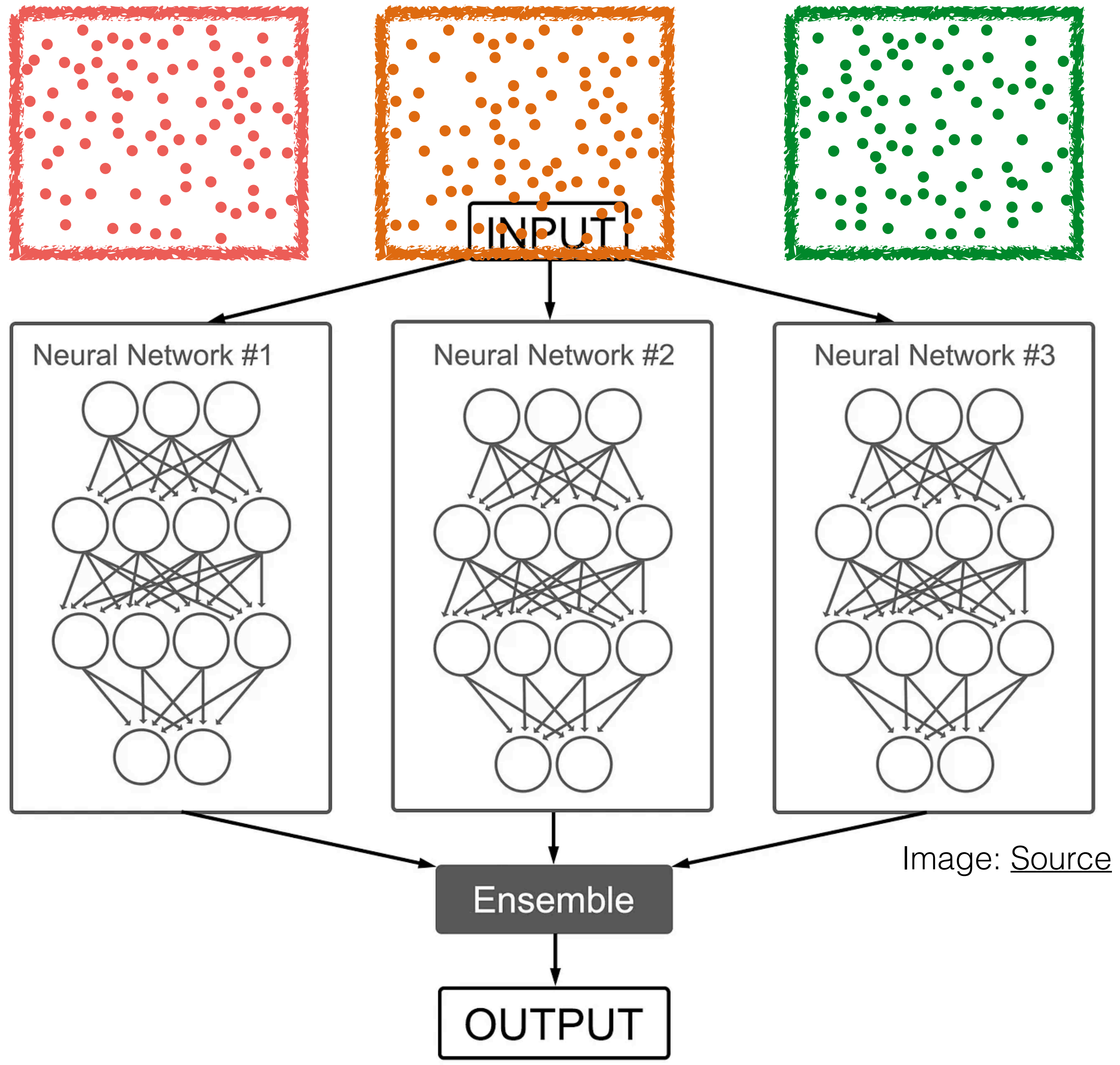
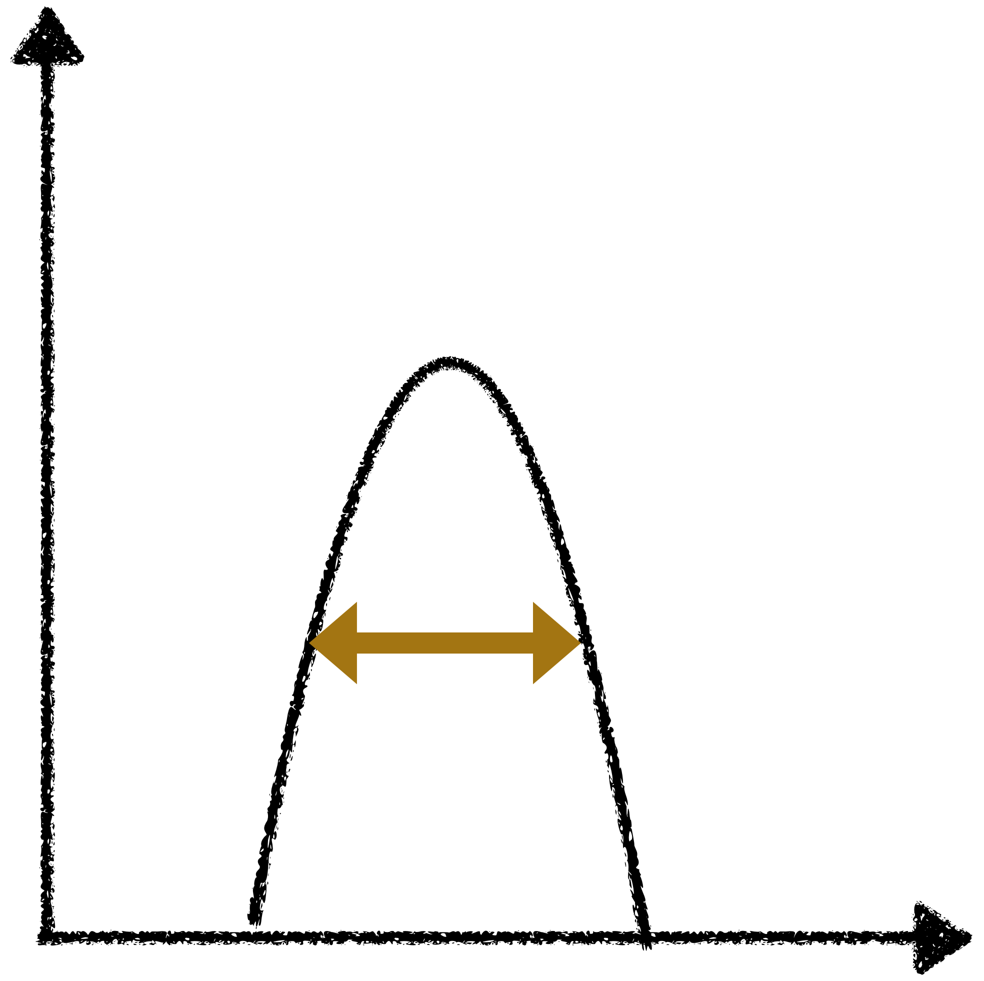


Image: Source

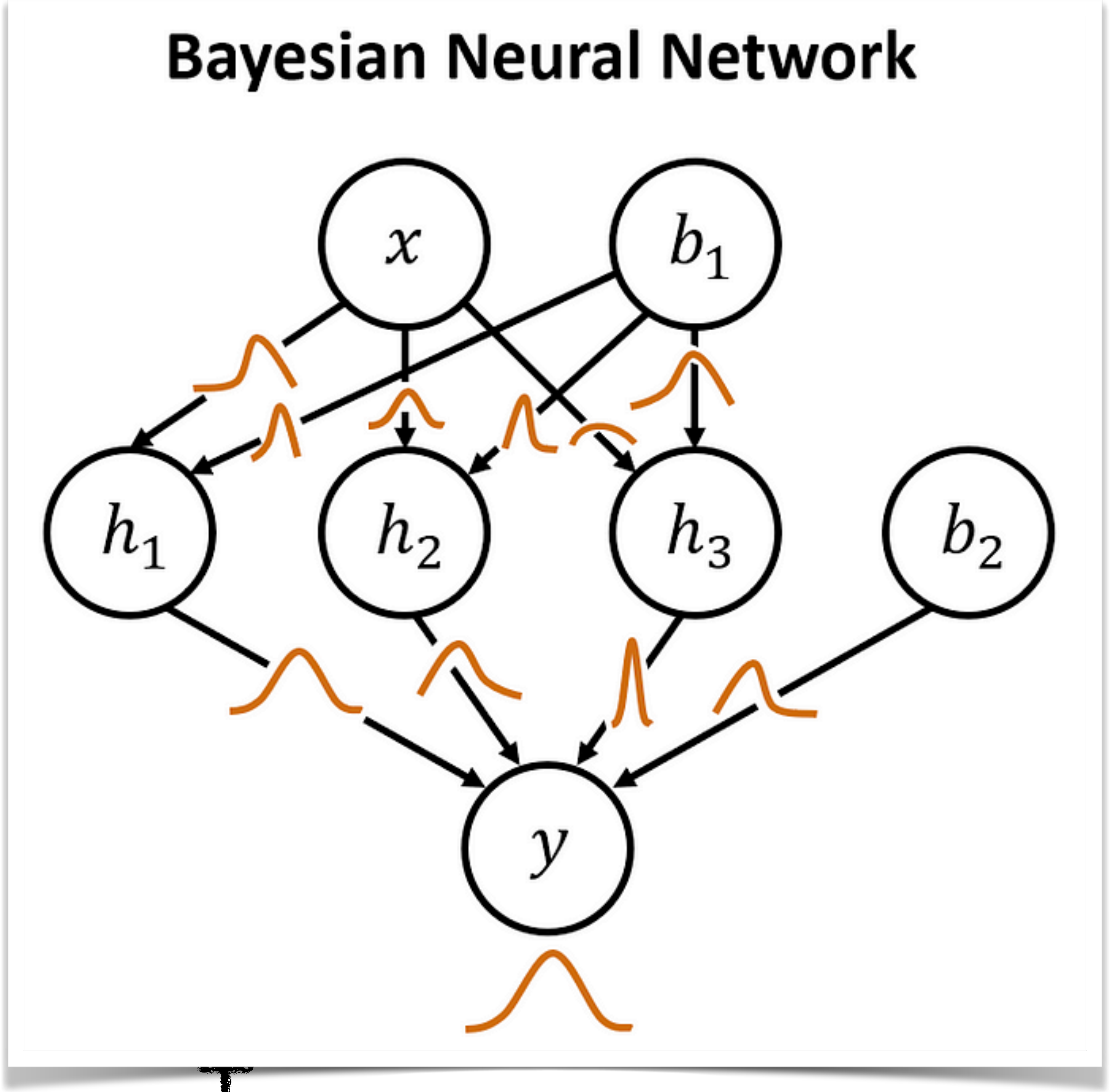


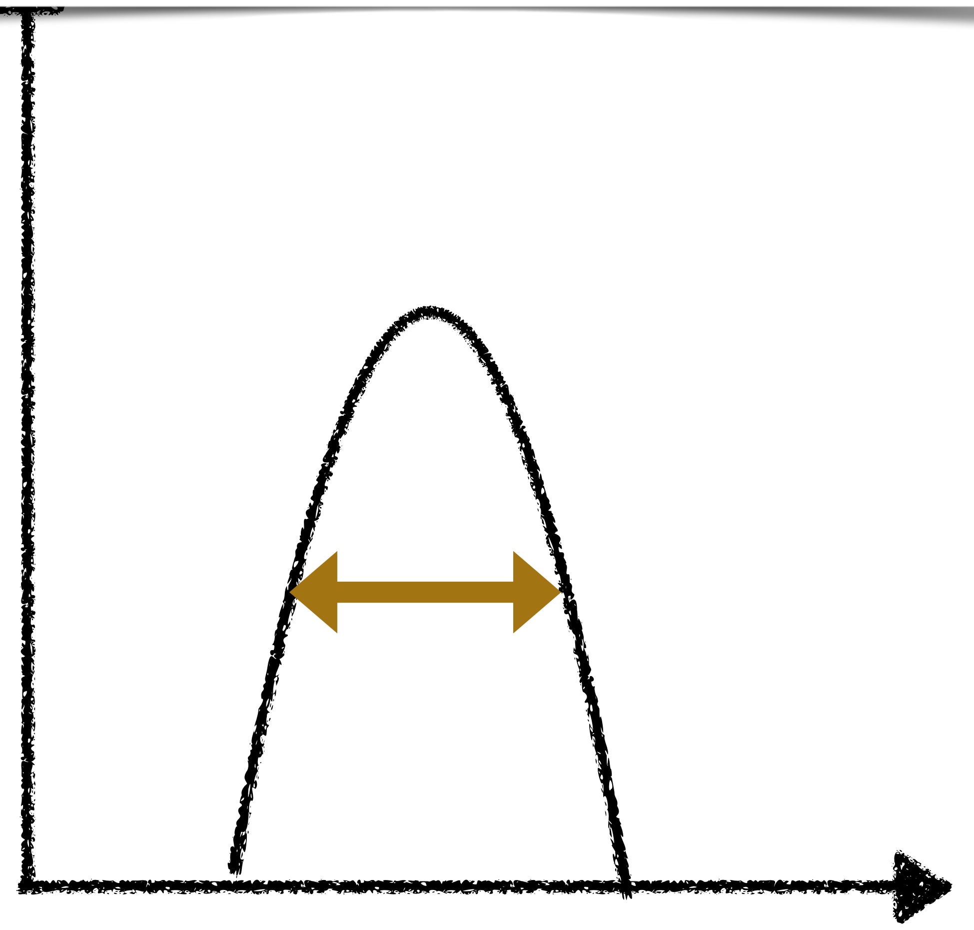
Quantifying uncertainty on estimated likelihood..

- Train an ensemble of networks, **each on a bootstrapped version of the training dataset**
    - Or Bayesian networks ? [Delaunoy et al, arXiv2408.15136]

- The spread in their prediction provides the uncertainty due to limited training statistics, and random network initialisation

- **Ensemble average used as final prediction**, so what's the uncertainty on that ?
    - Too expensive to train thousands of ensembles
    - Create bootstrapped ensembles ?
        - Each network trained on bootstrapped training dataset ?





Quantifying uncertainty on estimated likelihood..

- Train an ensemble of networks, **each on a bootstrapped version of the training dataset**
    - Or Bayesian networks ? [Delaunoy et al, arXiv2408.15136]

- The spread in their prediction provides the uncertainty due to limited training statistics, and random network initialisation

- **Ensemble average used as final prediction**, so what's the uncertainty on that ?
    - Too expensive to train thousands of ensembles
    - Create bootstrapped ensembles ?
        - Each network trained on bootstrapped training dataset ?

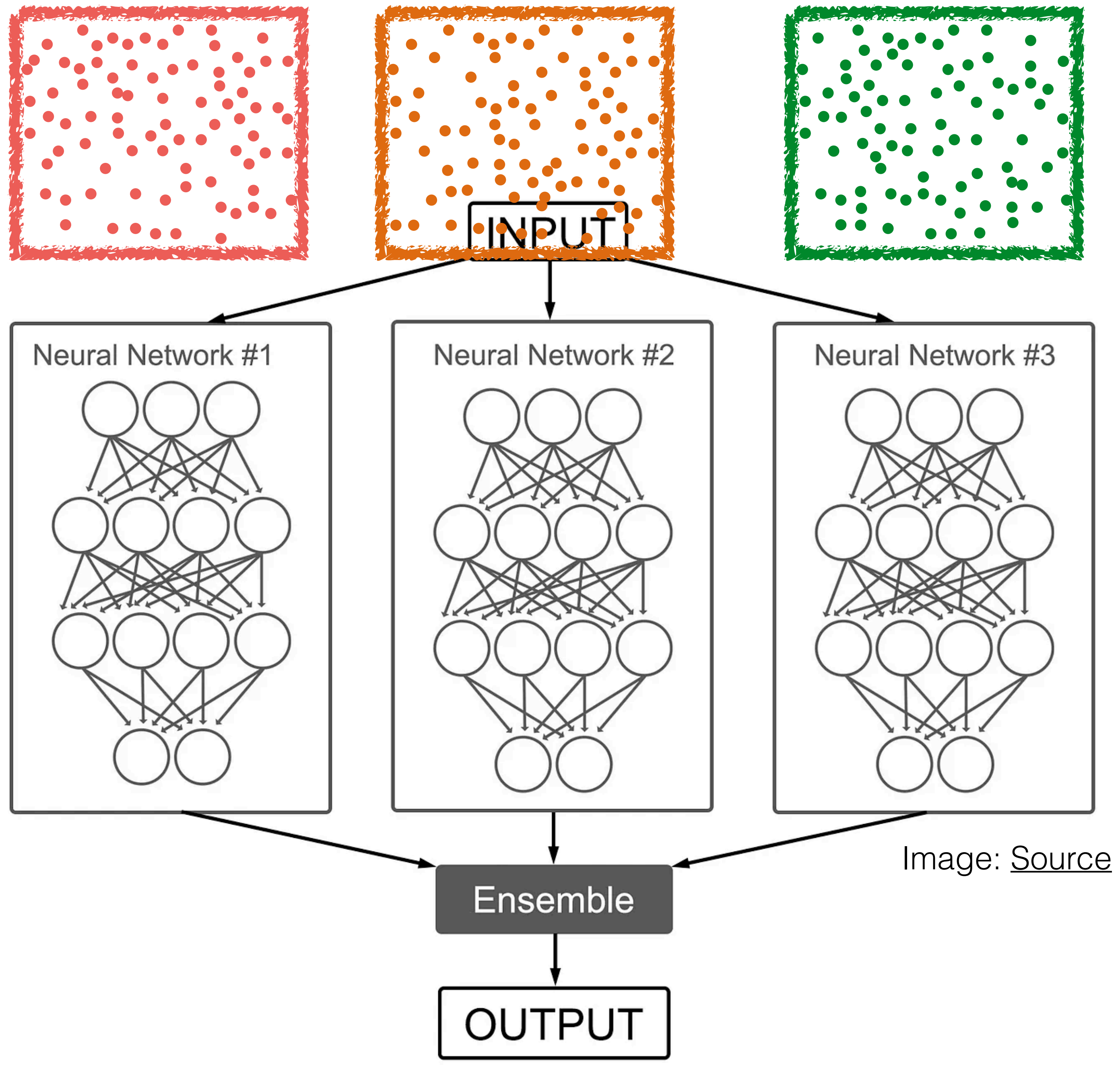
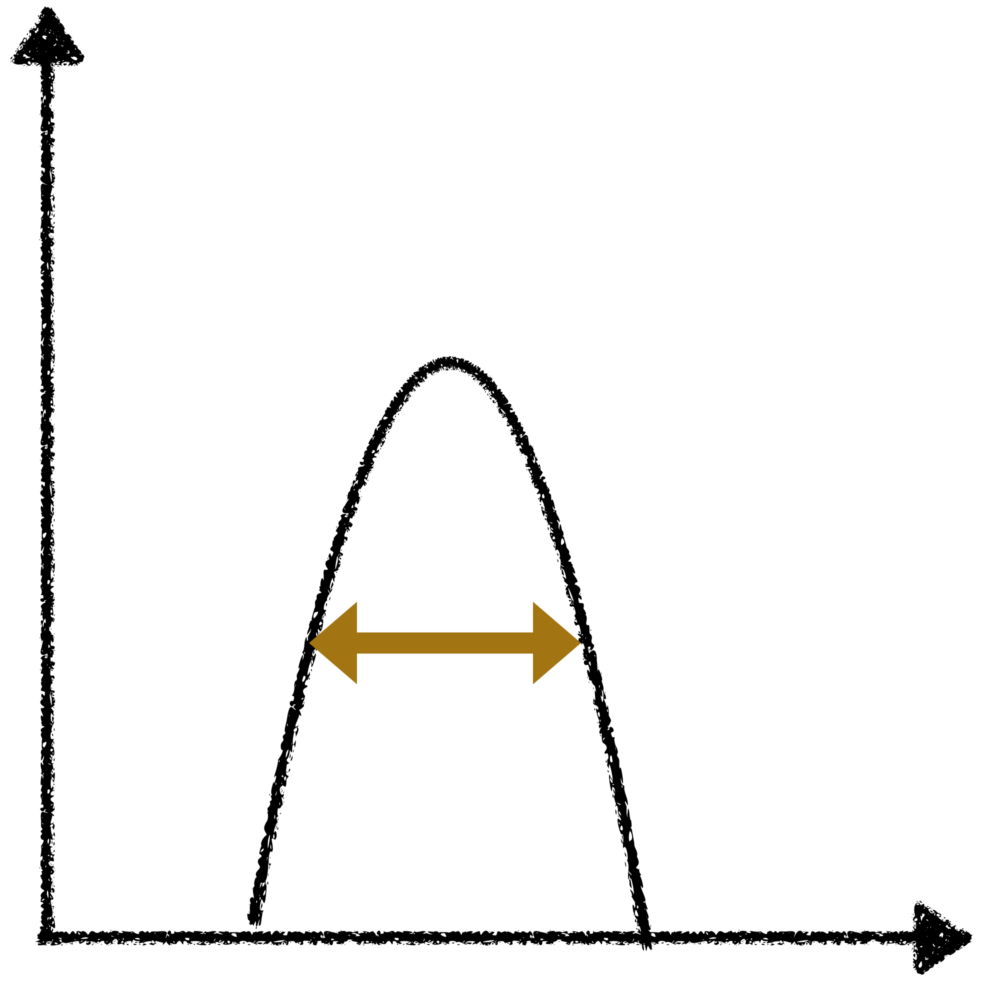


Image: Source



Quantifying uncertainty on estimated likelihood..

- Train an ensemble of networks, **each on a bootstrapped version of the training dataset**
    - Or Bayesian networks ? [Delaunoy et al, arXiv2408.15136]
- The spread in their prediction provides the uncertainty due to limited training statistics, and random network initialisation
- **Ensemble average used as final prediction**, so what's the uncertainty on that ?
    - Too expensive to train thousands of ensembles
    - Create bootstrapped ensembles ?
        - Each network trained on bootstrapped training dataset ?
- Does this trivially extend to pre-trained / foundation models?
- If your simulator is itself a generative model, how to efficiently propagate statistical uncertainties through?

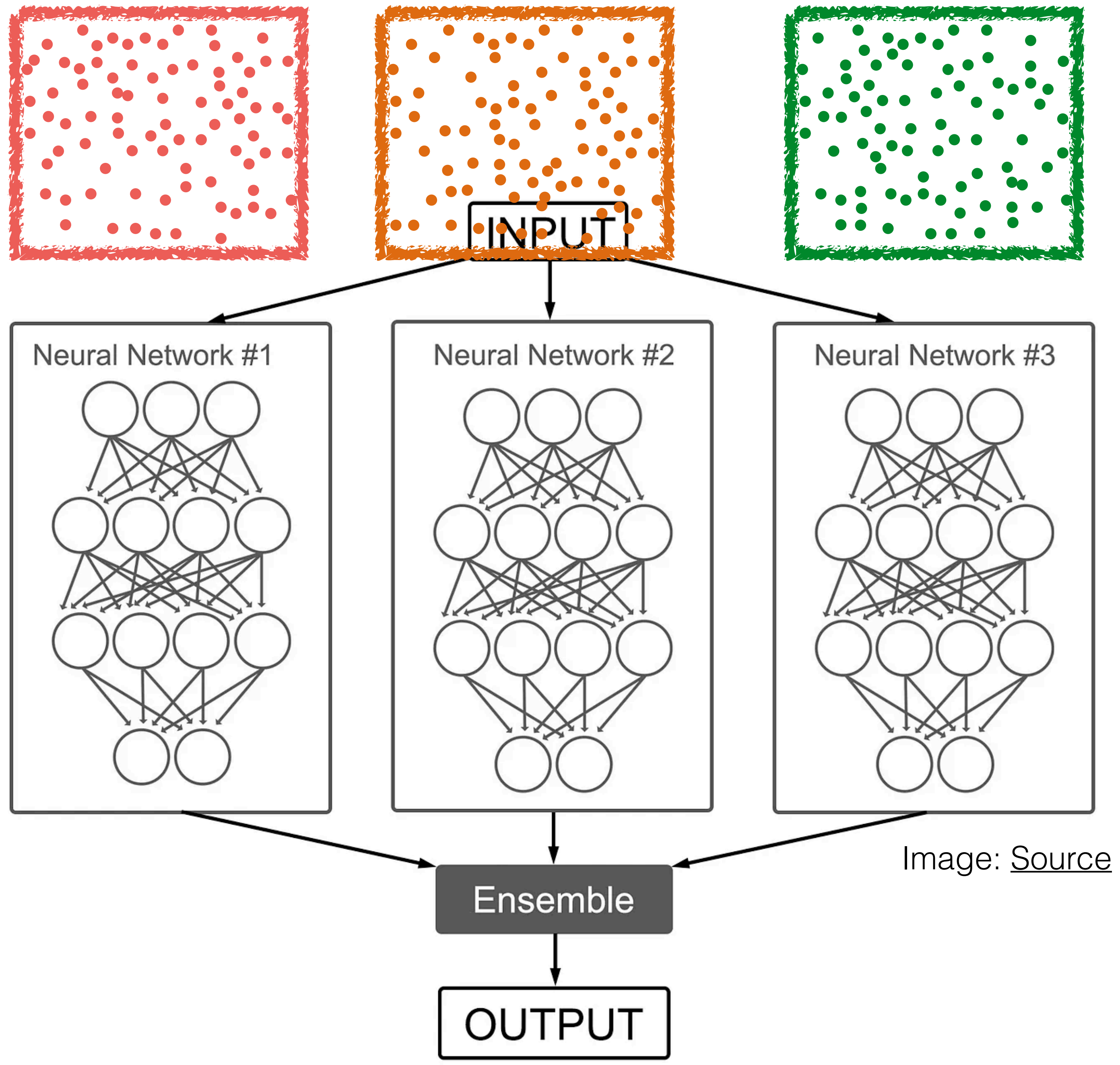
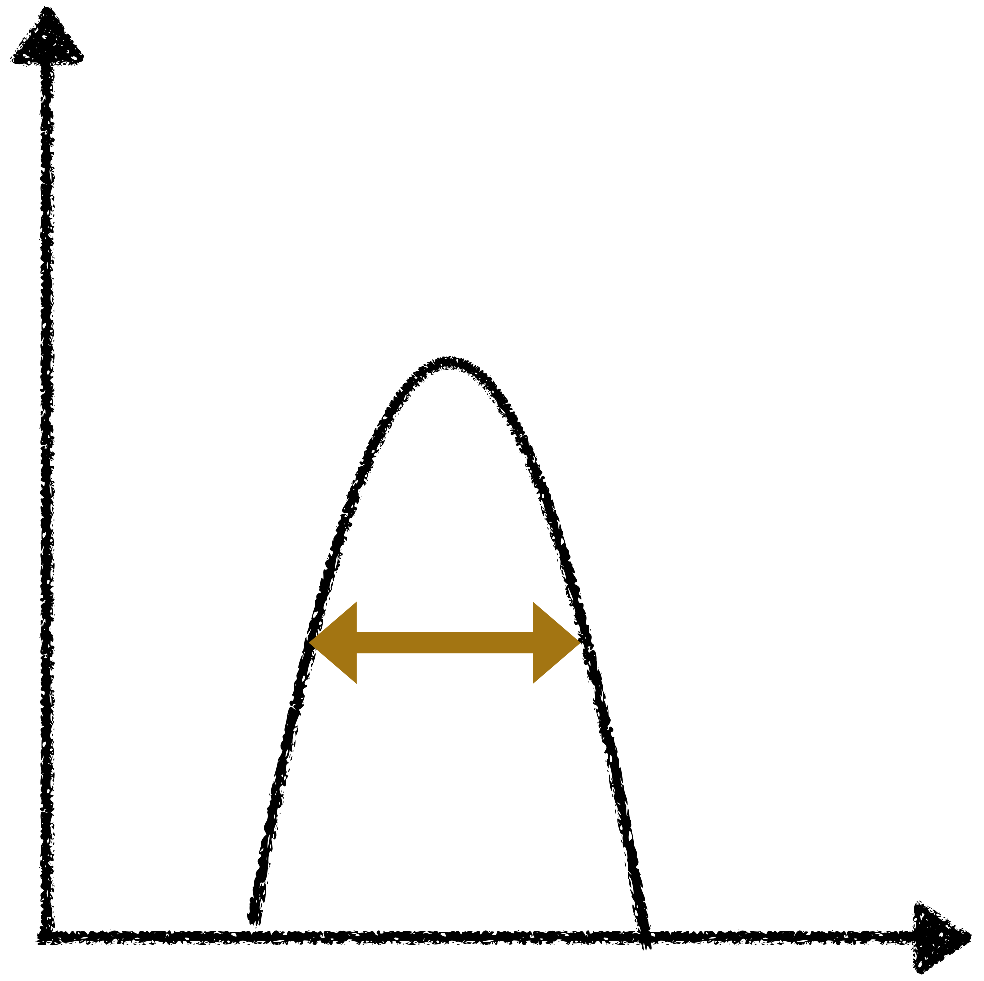


Image: Source



Quantifying uncertainty on estimated likelihood..

- Train an ensemble of networks, **each on a bootstrapped version of the training dataset**
    - Or Bayesian networks ? [Delaunoy et al, arXiv2408.15136]
- The spread in their prediction provides the uncertainty due to limited training statistics, and random network initialisation
- **Ensemble average used as final prediction**, so what's the uncertainty on that ?
    - Too expensive to train thousands of ensembles
    - Create bootstrapped ensembles ?
        - Each network trained on bootstrapped training dataset ?
- Does this trivially extend to pre-trained / foundation models?
- If your simulator is itself a generative model, how to efficiently propagate statistical uncertainties through?

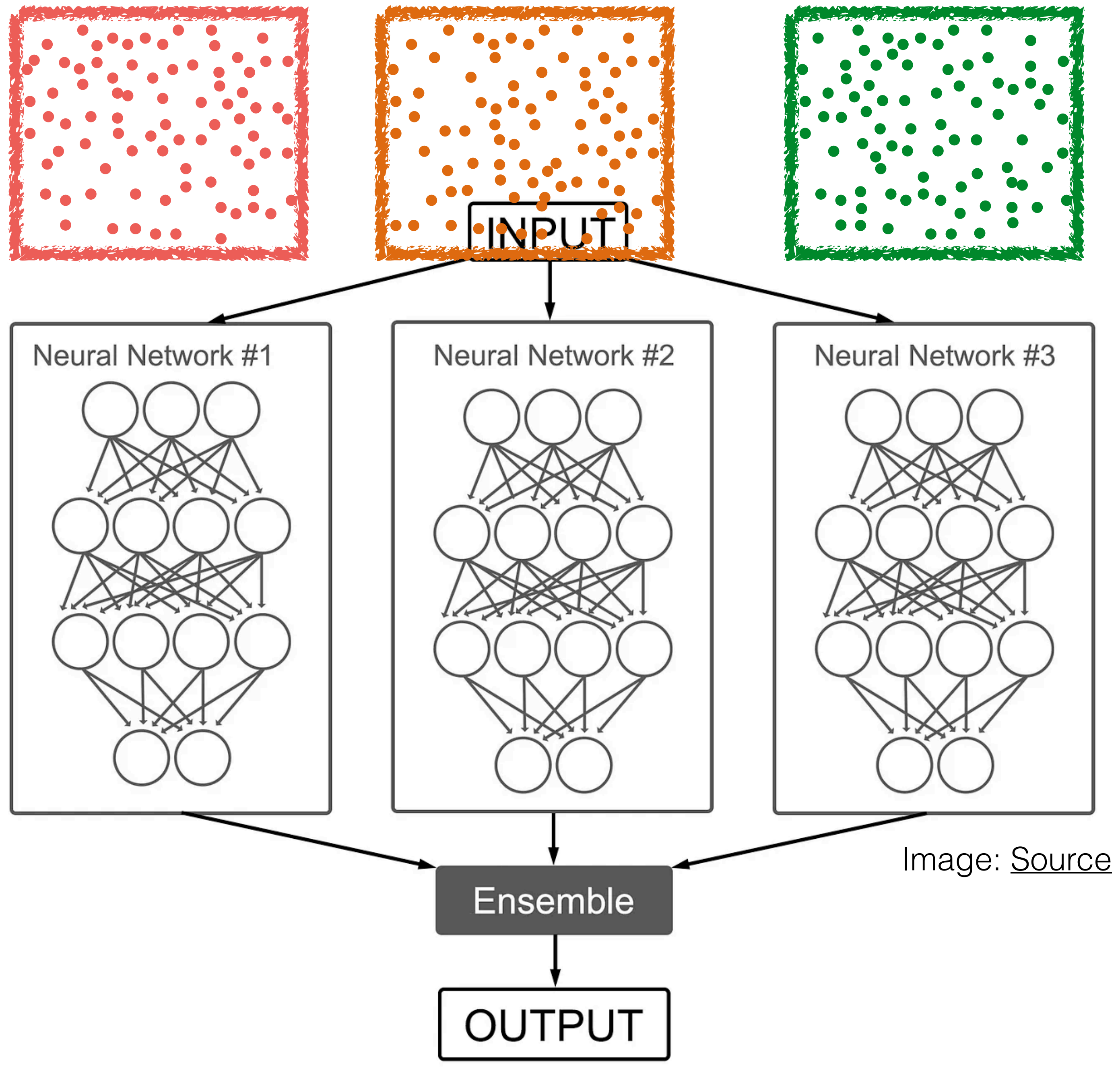
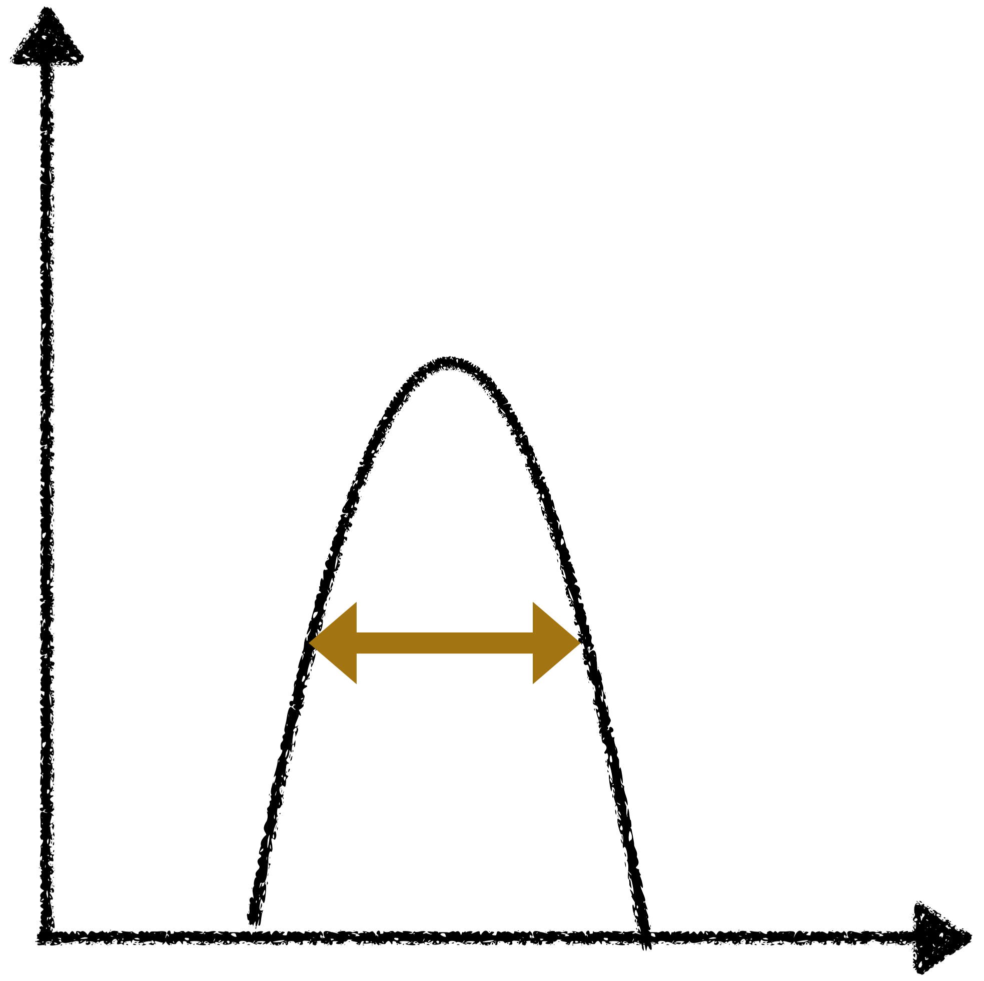


Image: Source

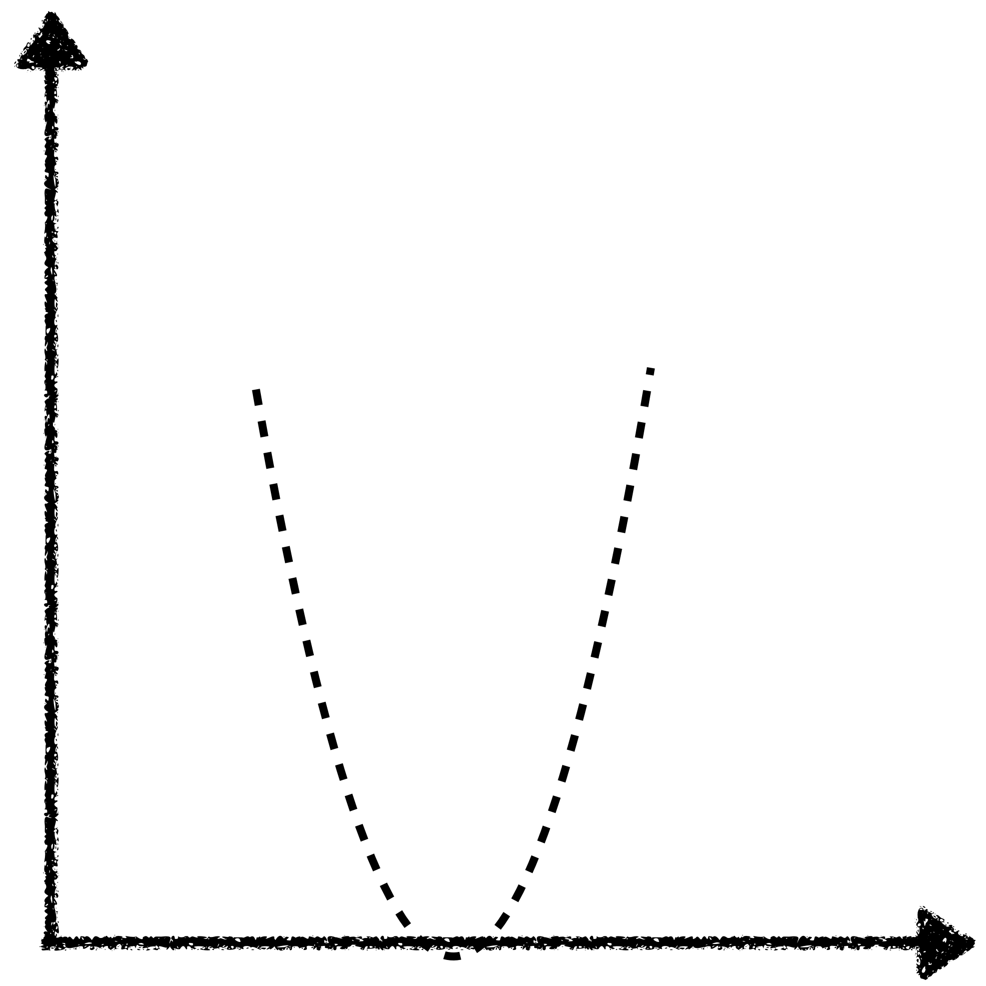


If these questions interest you, come chat with me!

Propagating statistical uncertainty to final result

- In histogram analysis we assign **1 nuisance parameter per bin** for statistical uncertainty in template histograms built from simulations
    - NSBI: **1 nuisance parameter per event?**

- Brute force: check impact on final result and ‘profile’ ?
- Use methods from traditional unbinned analyses ?
- Maybe all of this is overkill if we perform the Neyman construction?

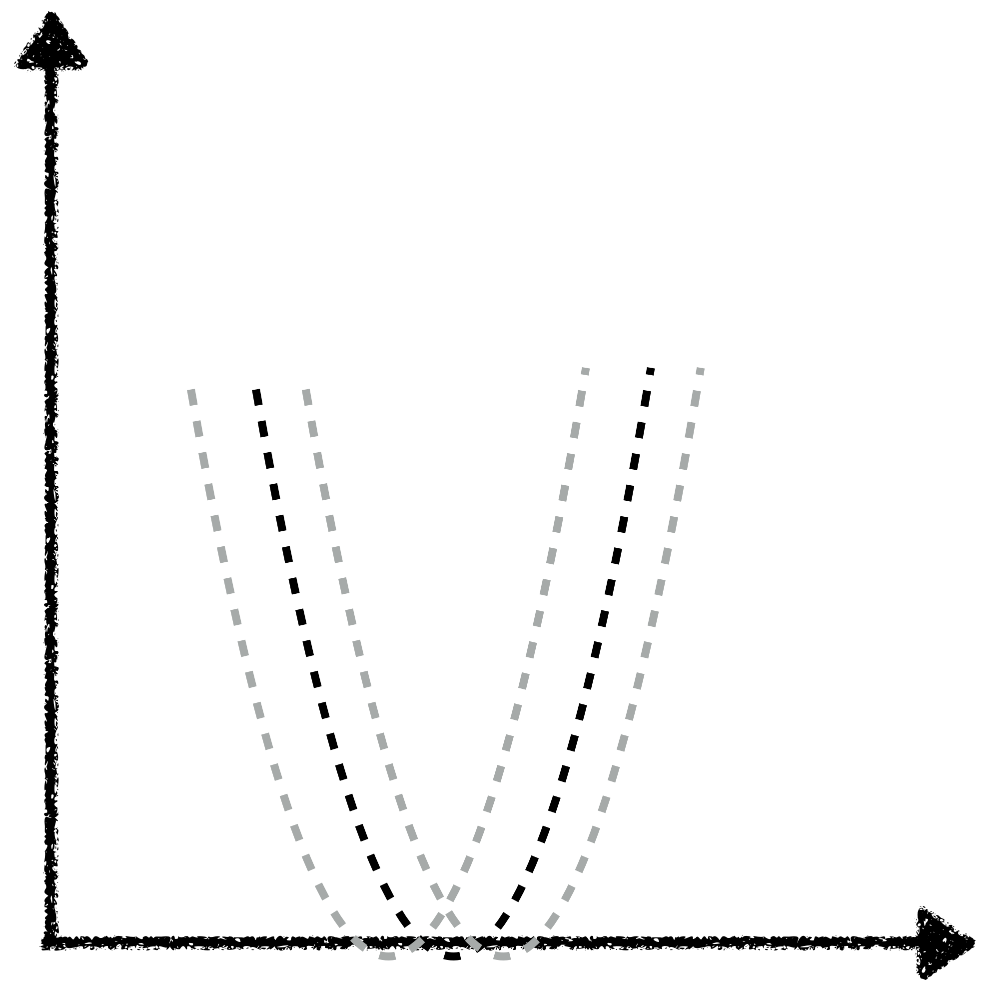


These are not completely unanswered questions, but more thought here would be valuable

Propagating statistical uncertainty to final result

- In histogram analysis we assign **1 nuisance parameter per bin** for statistical uncertainty in template histograms built from simulations
    - NSBI: **1 nuisance parameter per event?**

- Brute force: check impact on final result and ‘profile’ ?
- Use methods from traditional unbinned analyses ?
- Maybe all of this is overkill if we perform the Neyman construction?

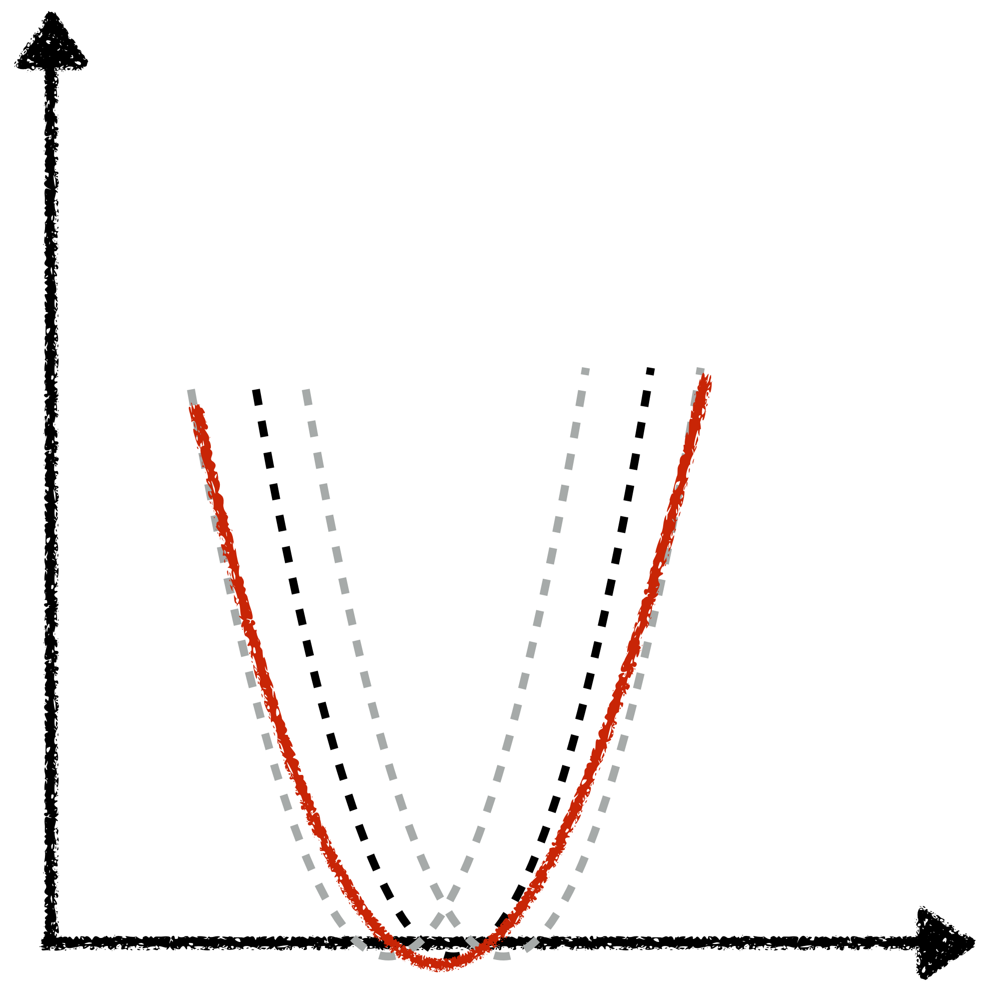


These are not completely unanswered questions, but more thought here would be valuable

Propagating statistical uncertainty to final result

- In histogram analysis we assign **1 nuisance parameter per bin** for statistical uncertainty in template histograms built from simulations
    - NSBI: **1 nuisance parameter per event?**

- Brute force: check impact on final result and 'profile' ?
- Use methods from traditional unbinned analyses ?
- Maybe all of this is overkill if we perform the Neyman construction?

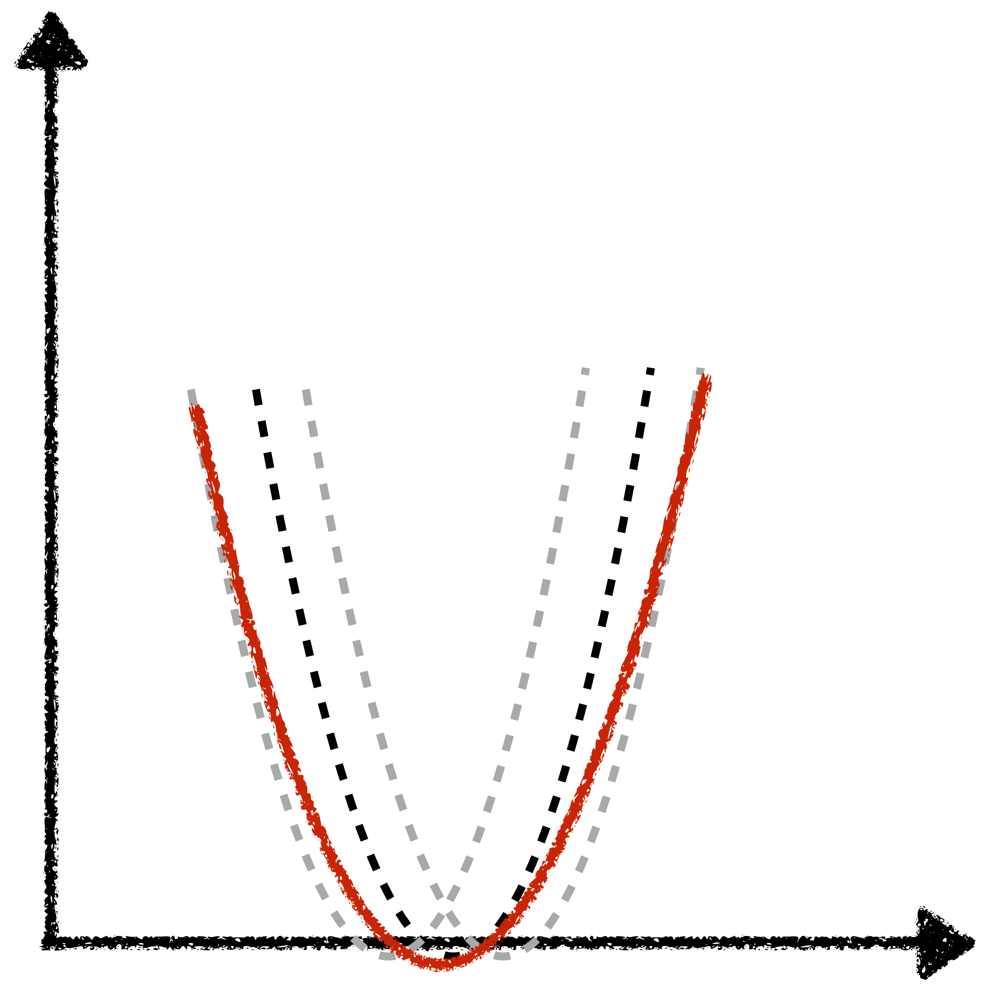


These are not completely unanswered questions, but more thought here would be valuable

Propagating statistical uncertainty to final result

- In histogram analysis we assign **1 nuisance parameter per bin** for statistical uncertainty in template histograms built from simulations
    - NSBI: **1 nuisance parameter per event?**

- Brute force: check impact on final result and 'profile' ?
- Use methods from traditional unbinned analyses ?
- Maybe all of this is overkill if we perform the Neyman construction?



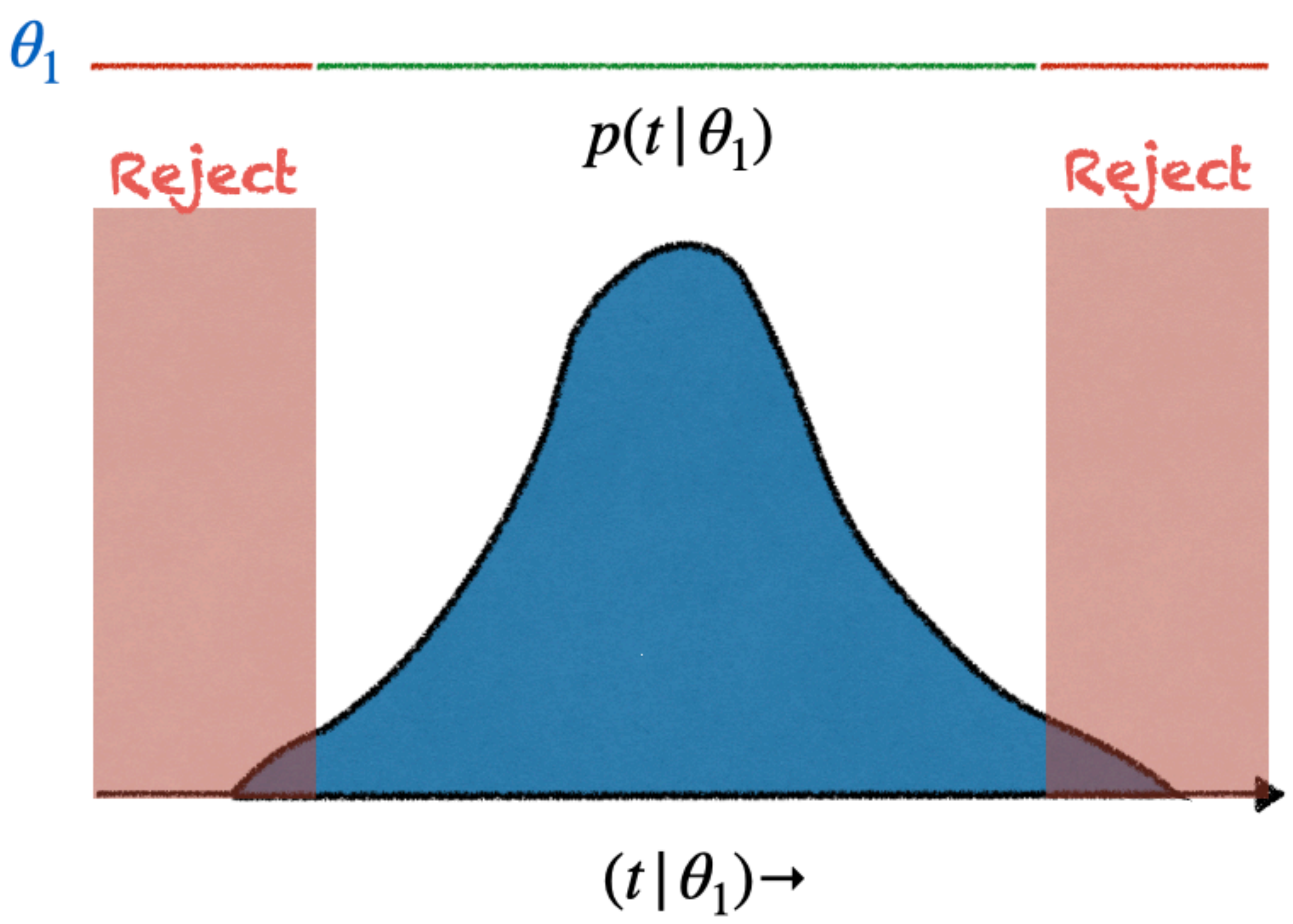
These are not completely unanswered questions, but more thought here would be valuable

If these questions interest you, come chat with me!

Neyman construction: Constructing confidence belts

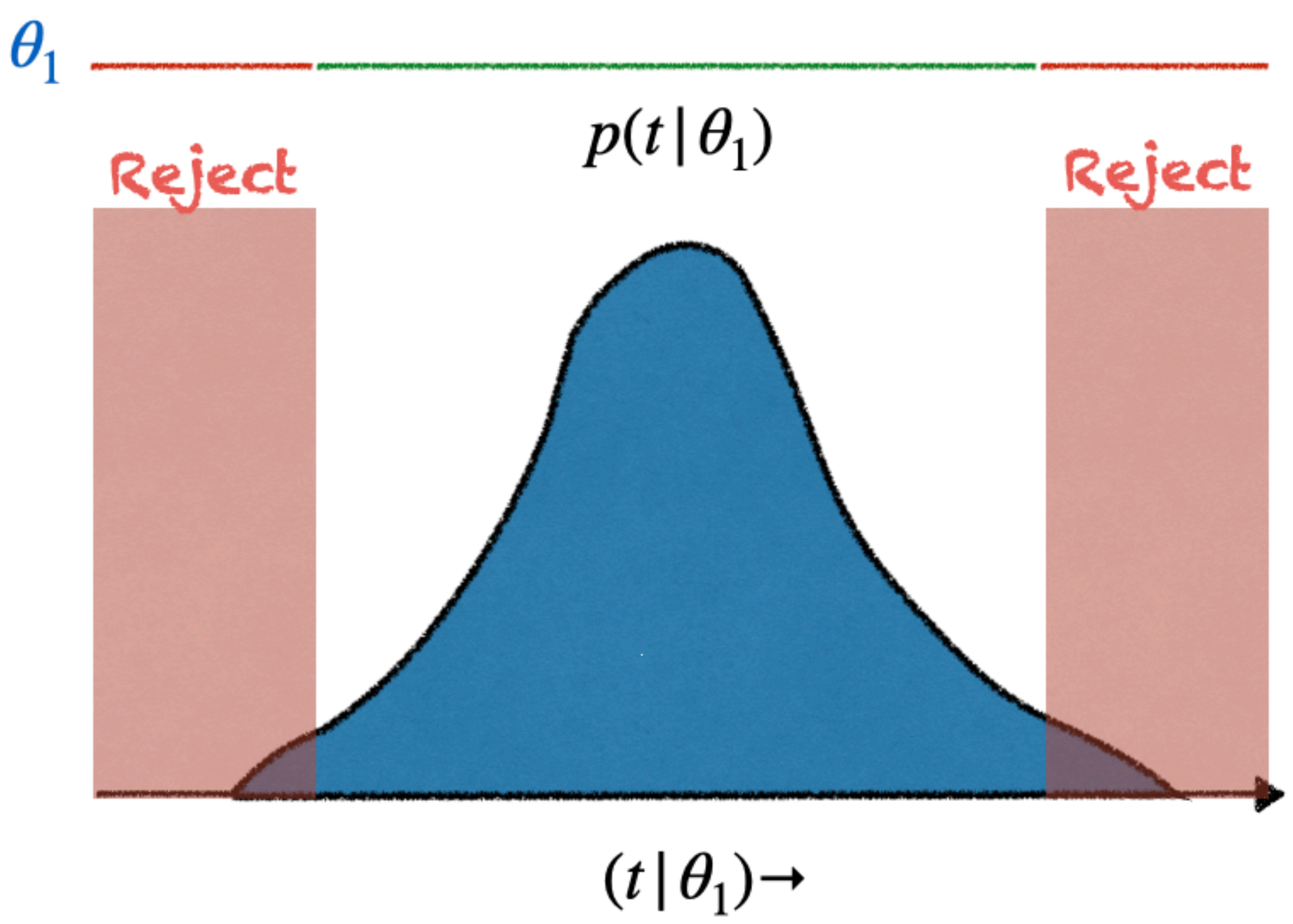
Neyman Construction

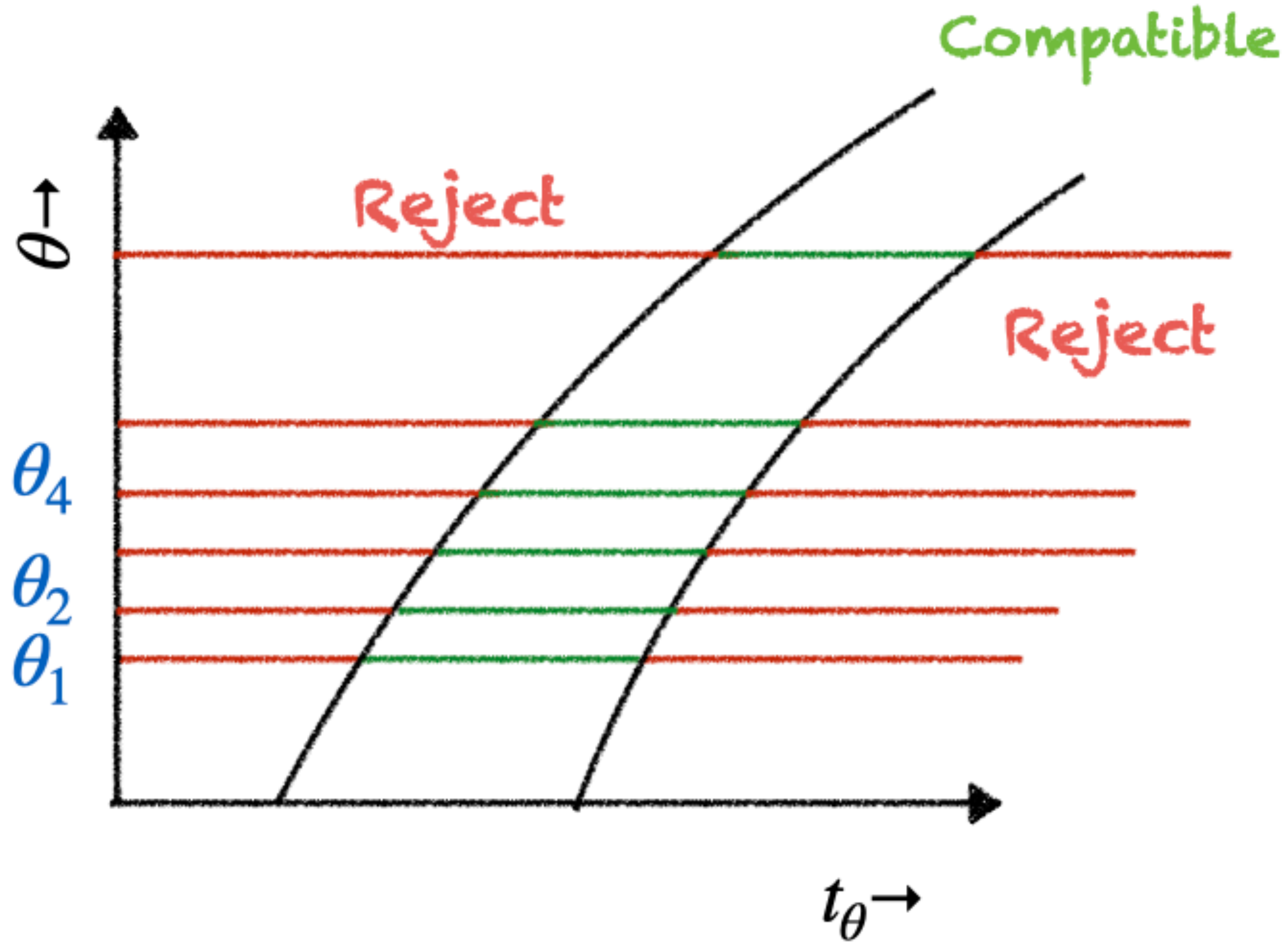
- To build confidence intervals for θ, we need to 'invert the hypothesis test'
- Generate pseudo-experiments ('toys') and determine 1σ & 2σ CI as a function of θ



Neyman Construction

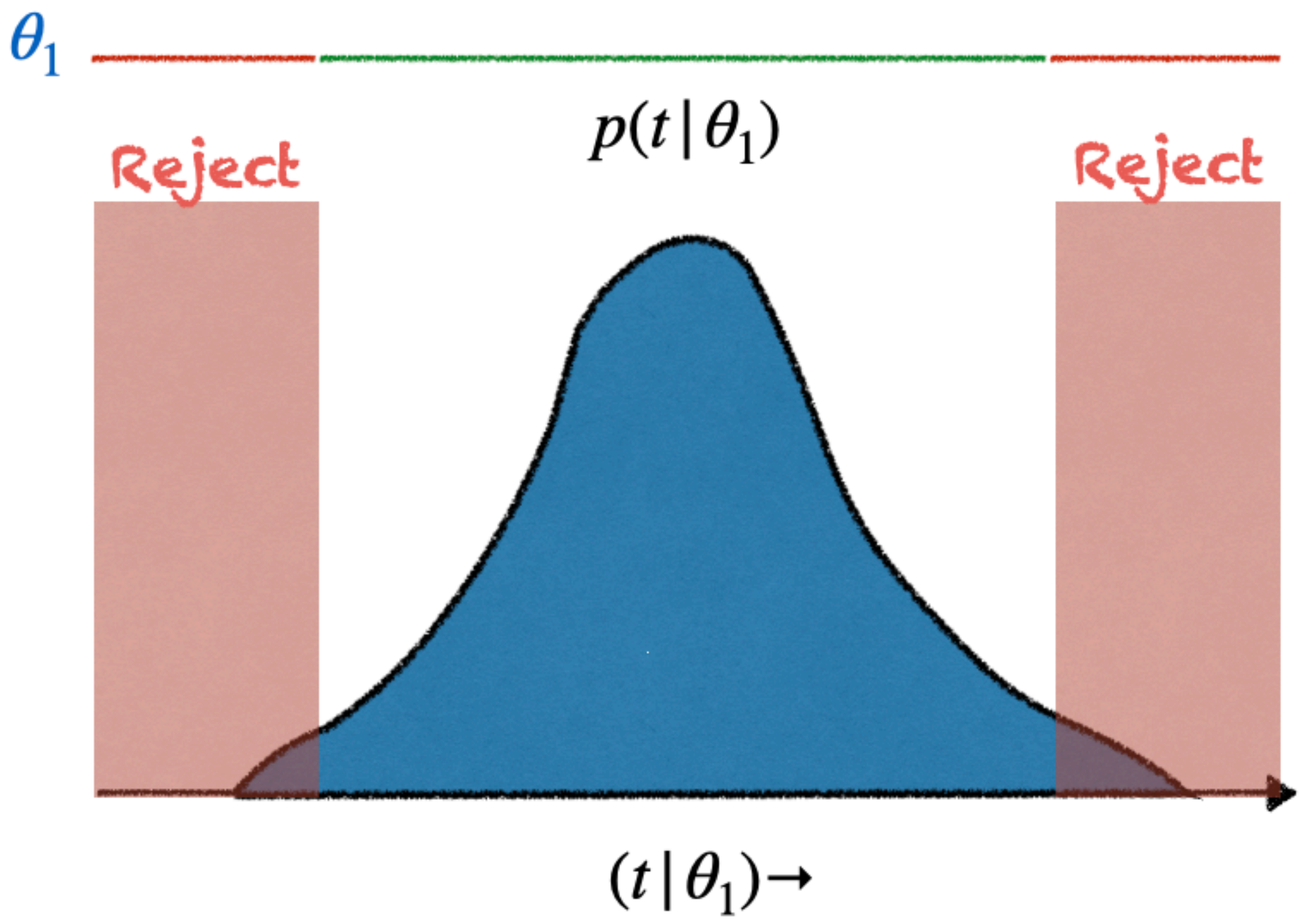
- To build confidence intervals for θ, we need to 'invert the hypothesis test'
- Generate pseudo-experiments ('toys') and determine 1σ & 2σ CI as a function of θ

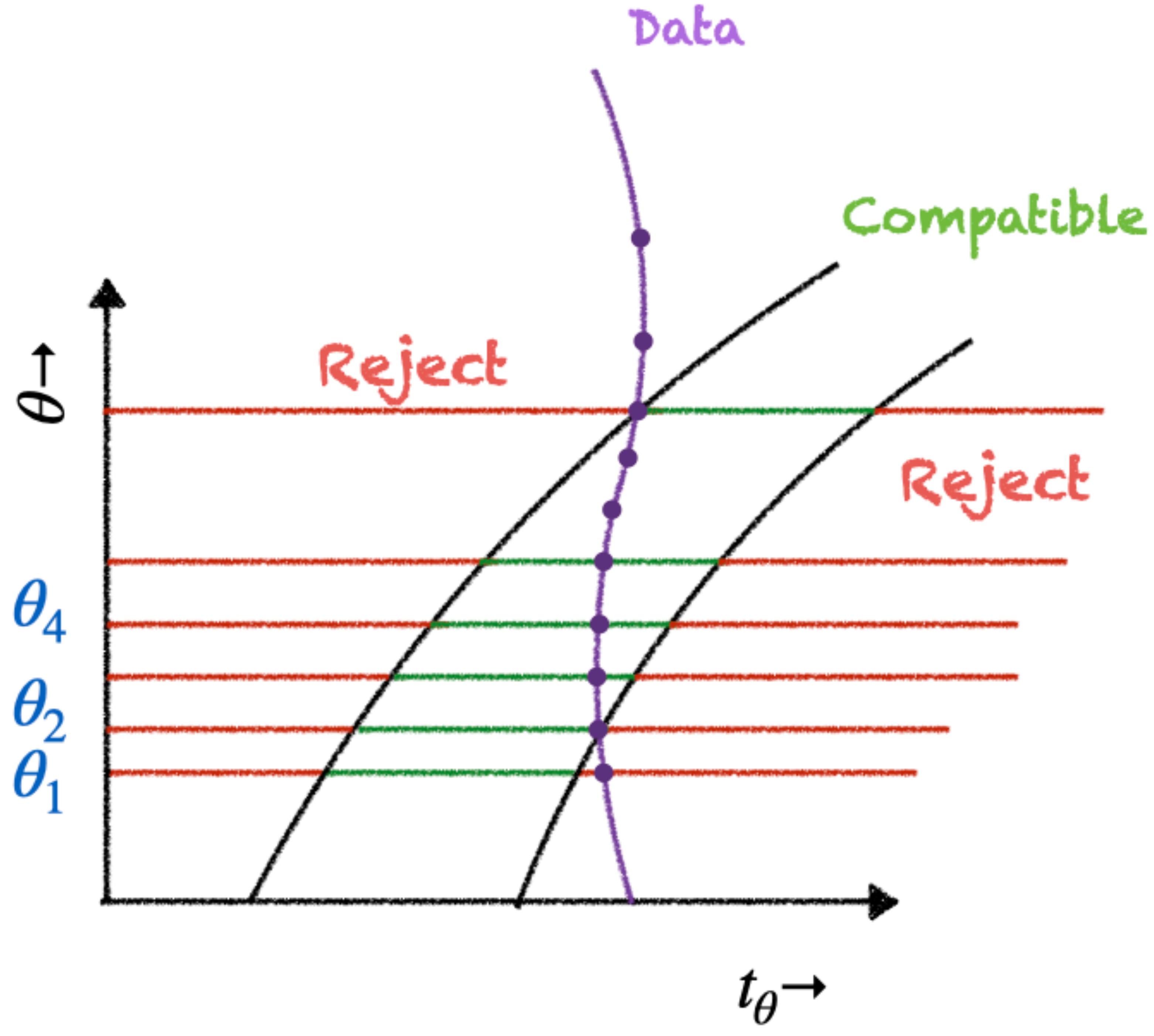




Neyman Construction

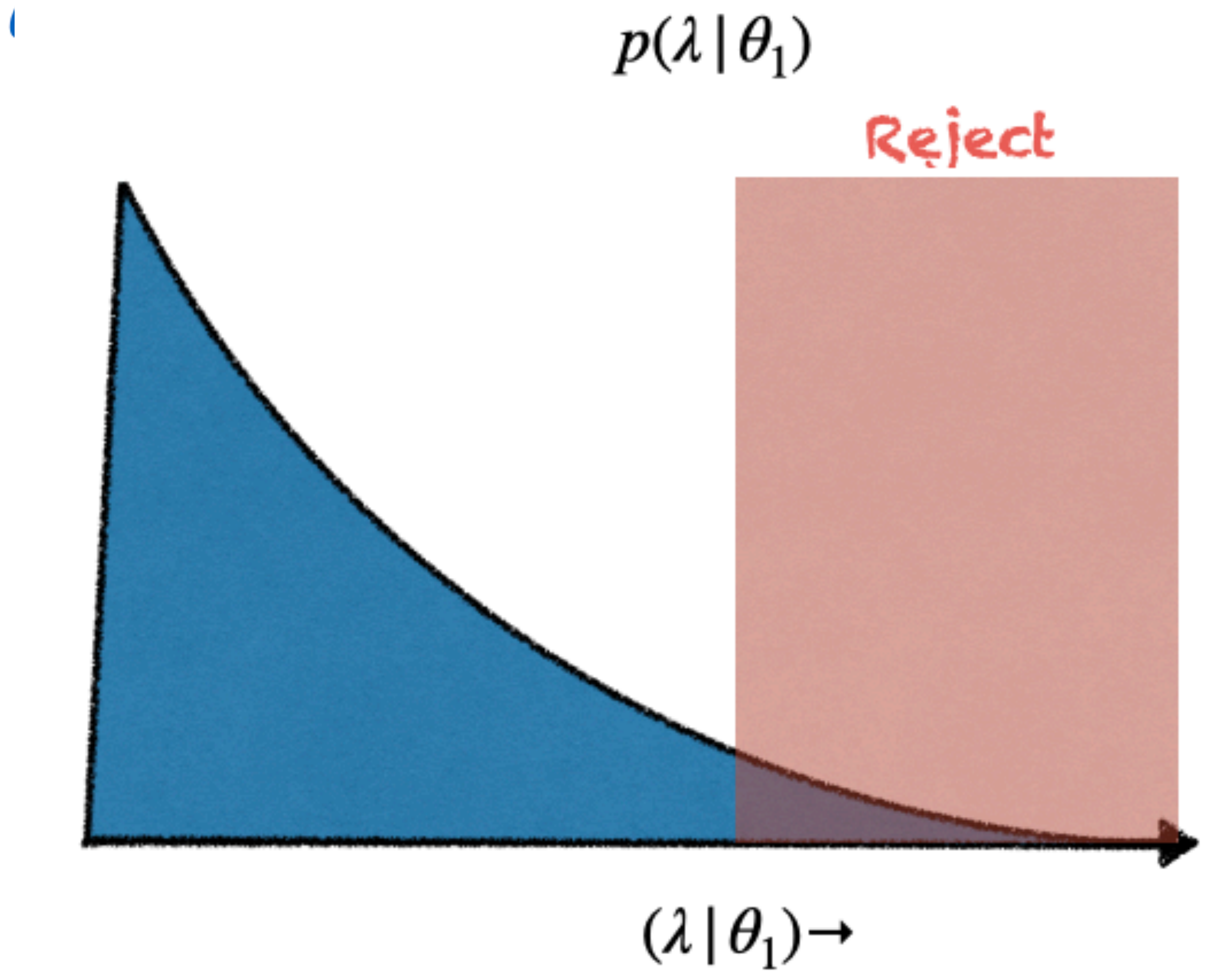
- To build confidence intervals for θ, we need to 'invert the hypothesis test'
- Generate pseudo-experiments ('toys') and determine 1σ & 2σ CI as a function of θ

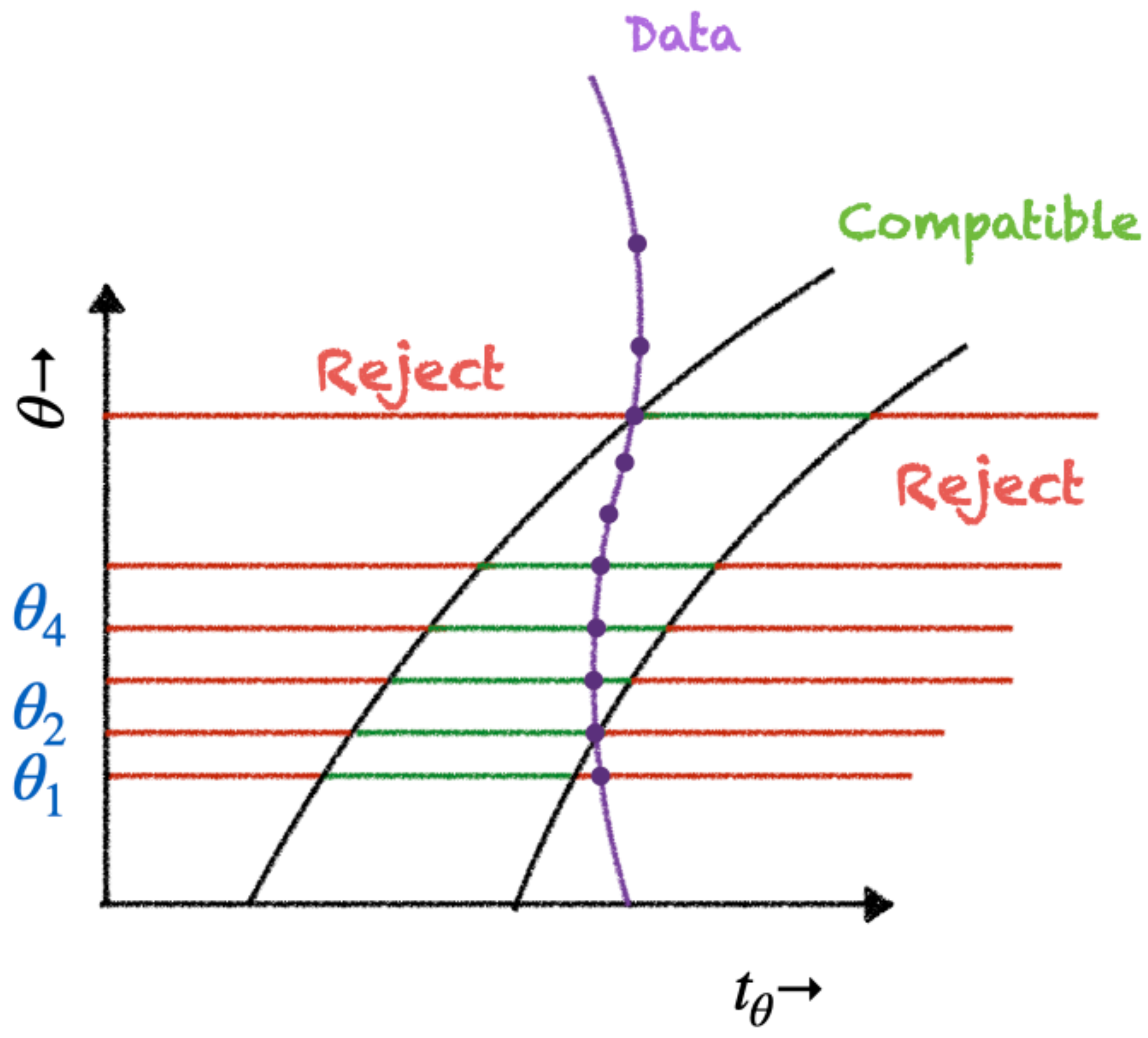




Neyman Construction

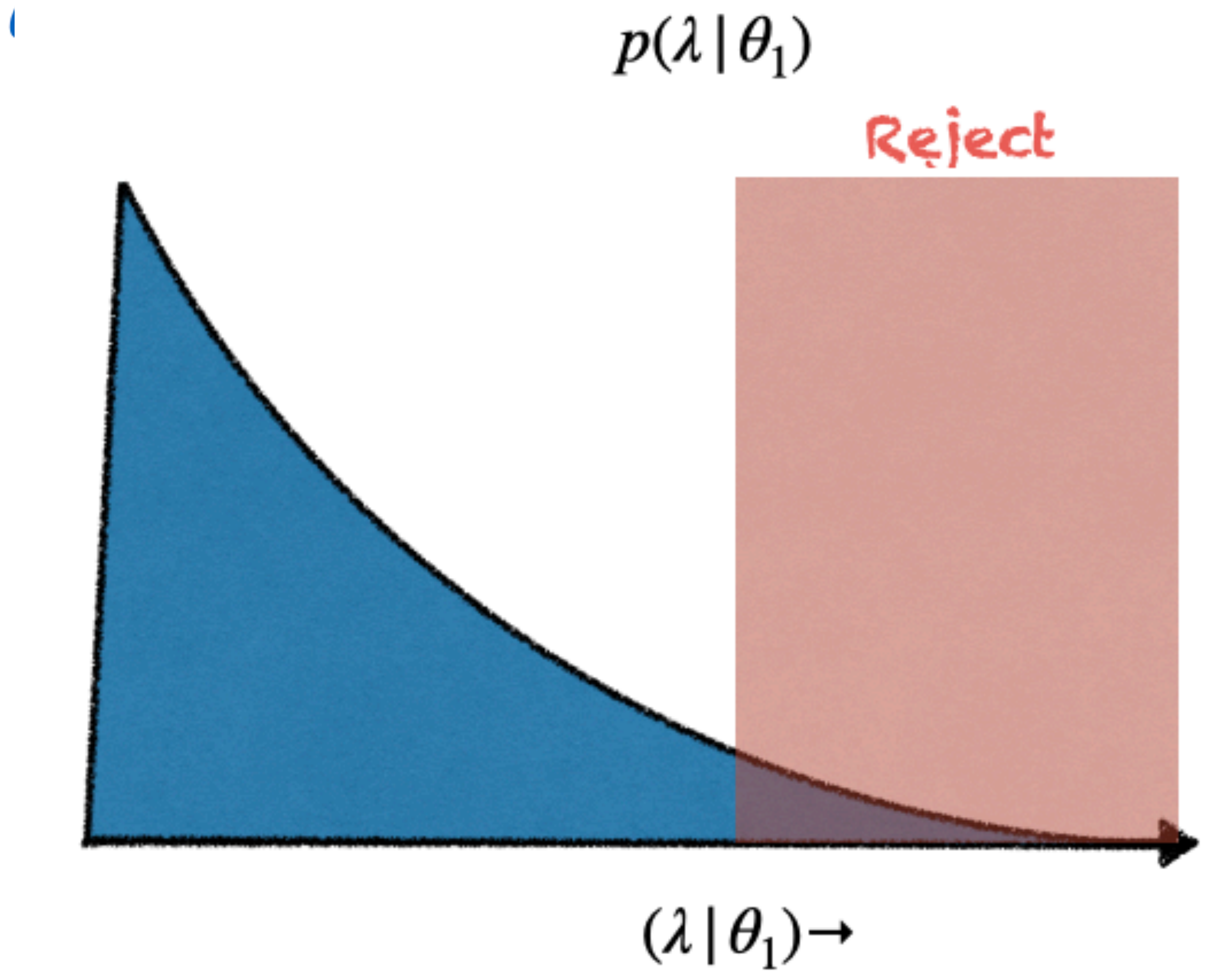
- To build confidence intervals for θ, we need to 'invert the hypothesis test'
- Generate pseudo-experiments ('toys') and determine 1σ & 2σ CI as a function of θ

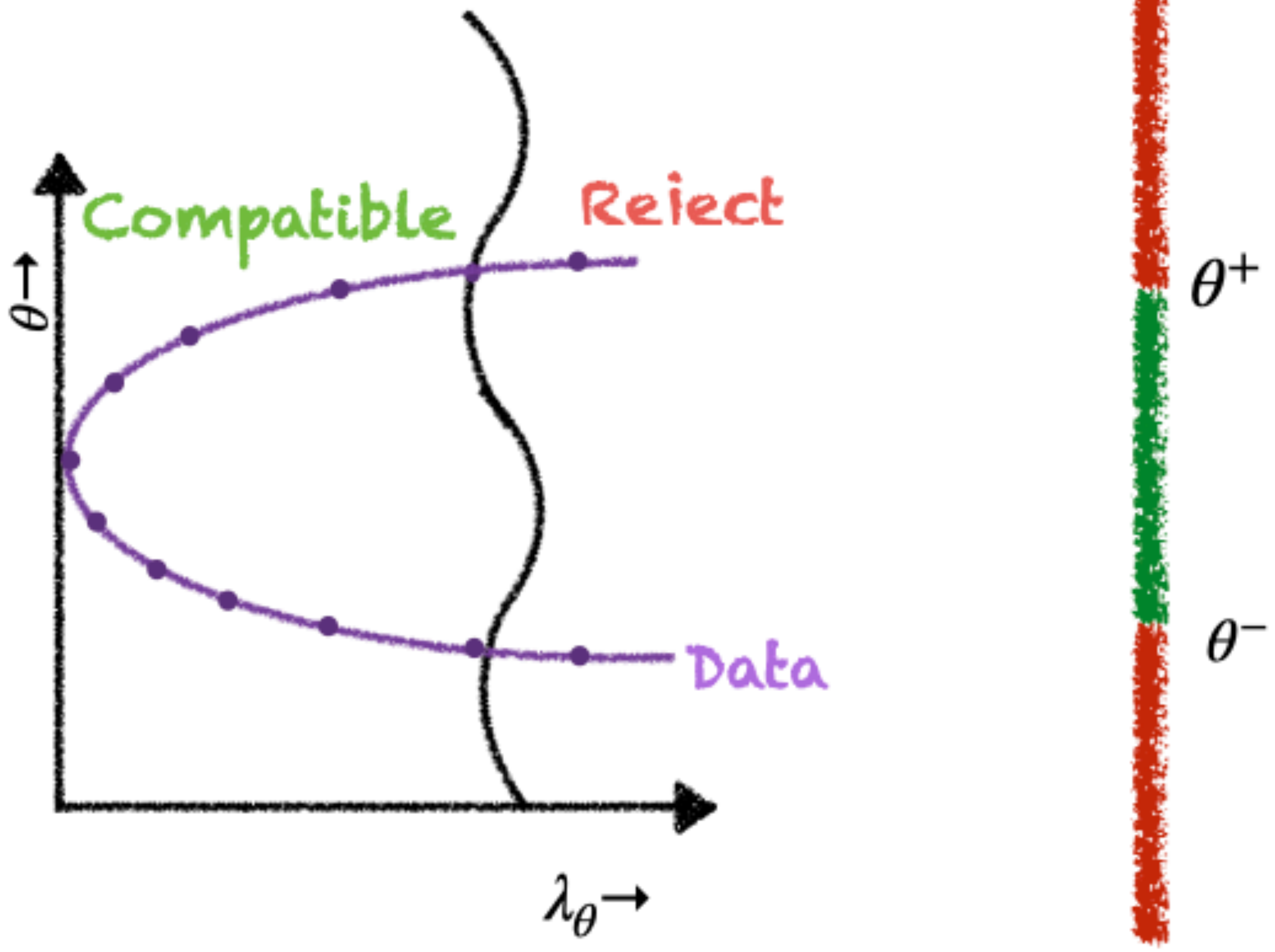




Neyman Construction

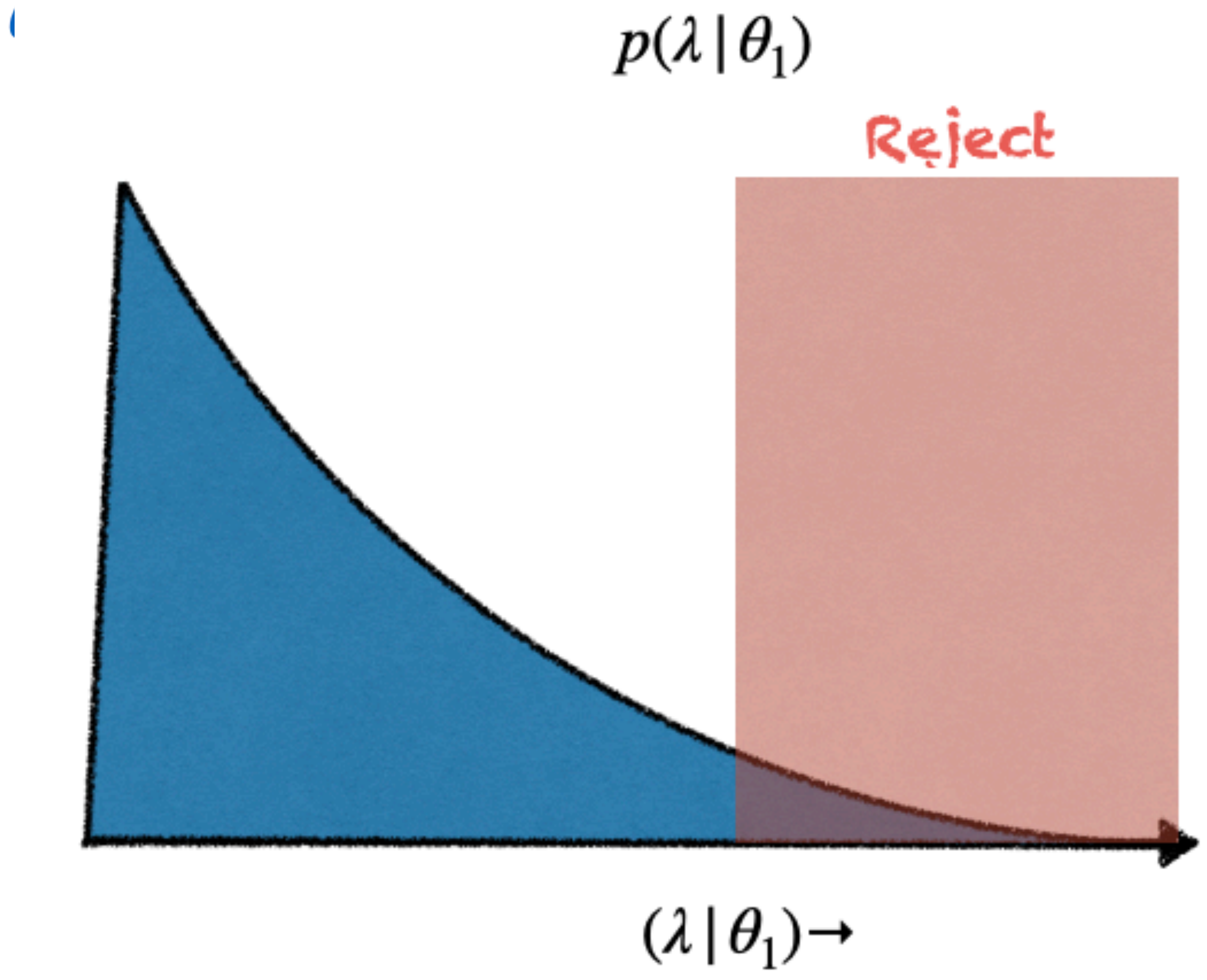
- To build confidence intervals for θ, we need to 'invert the hypothesis test'
- Generate pseudo-experiments ('toys') and determine 1σ & 2σ CI as a function of θ

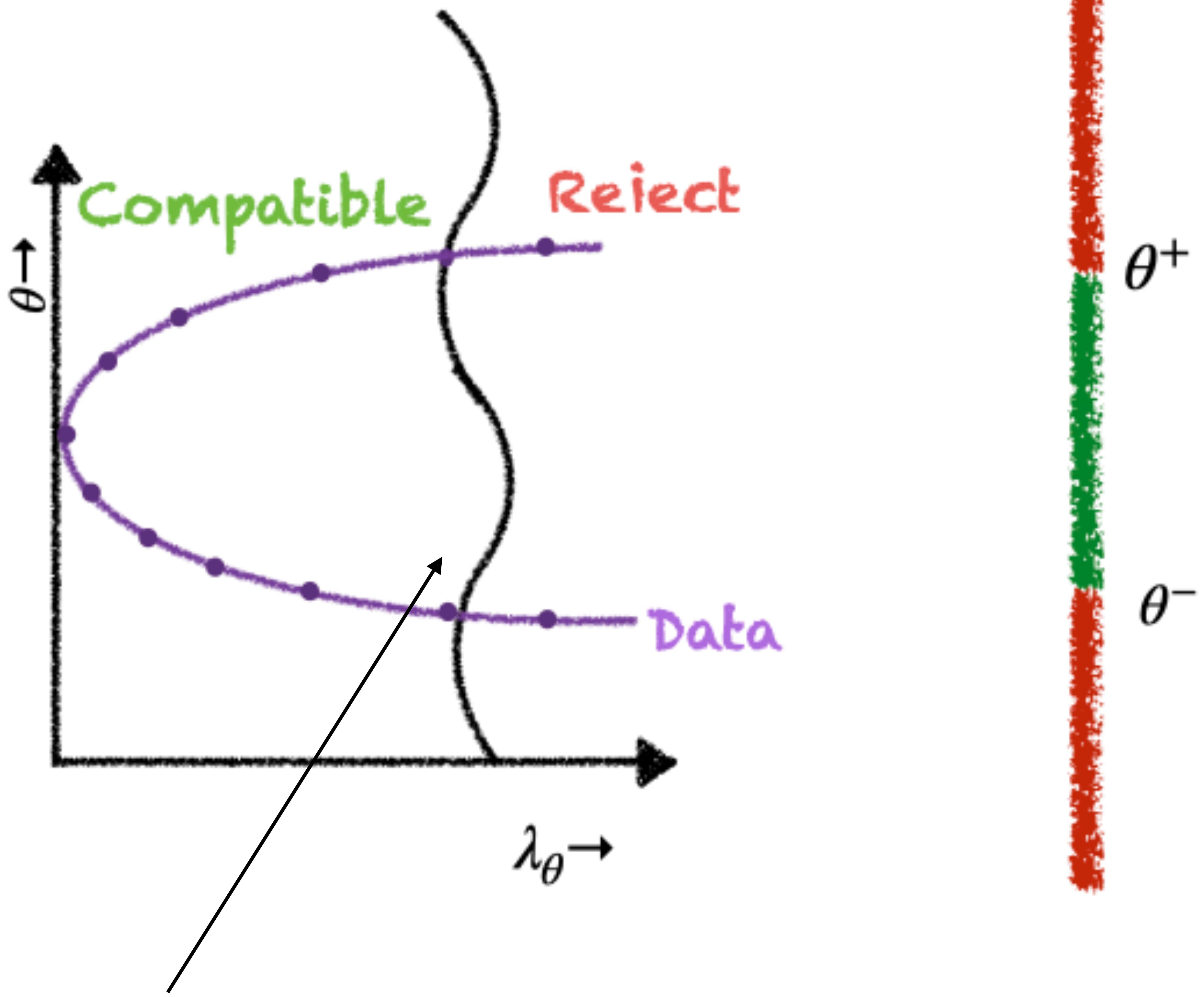




Neyman Construction

- To build confidence intervals for θ, we need to 'invert the hypothesis test'
- Generate pseudo-experiments ('toys') and determine 1σ & 2σ CI as a function of θ

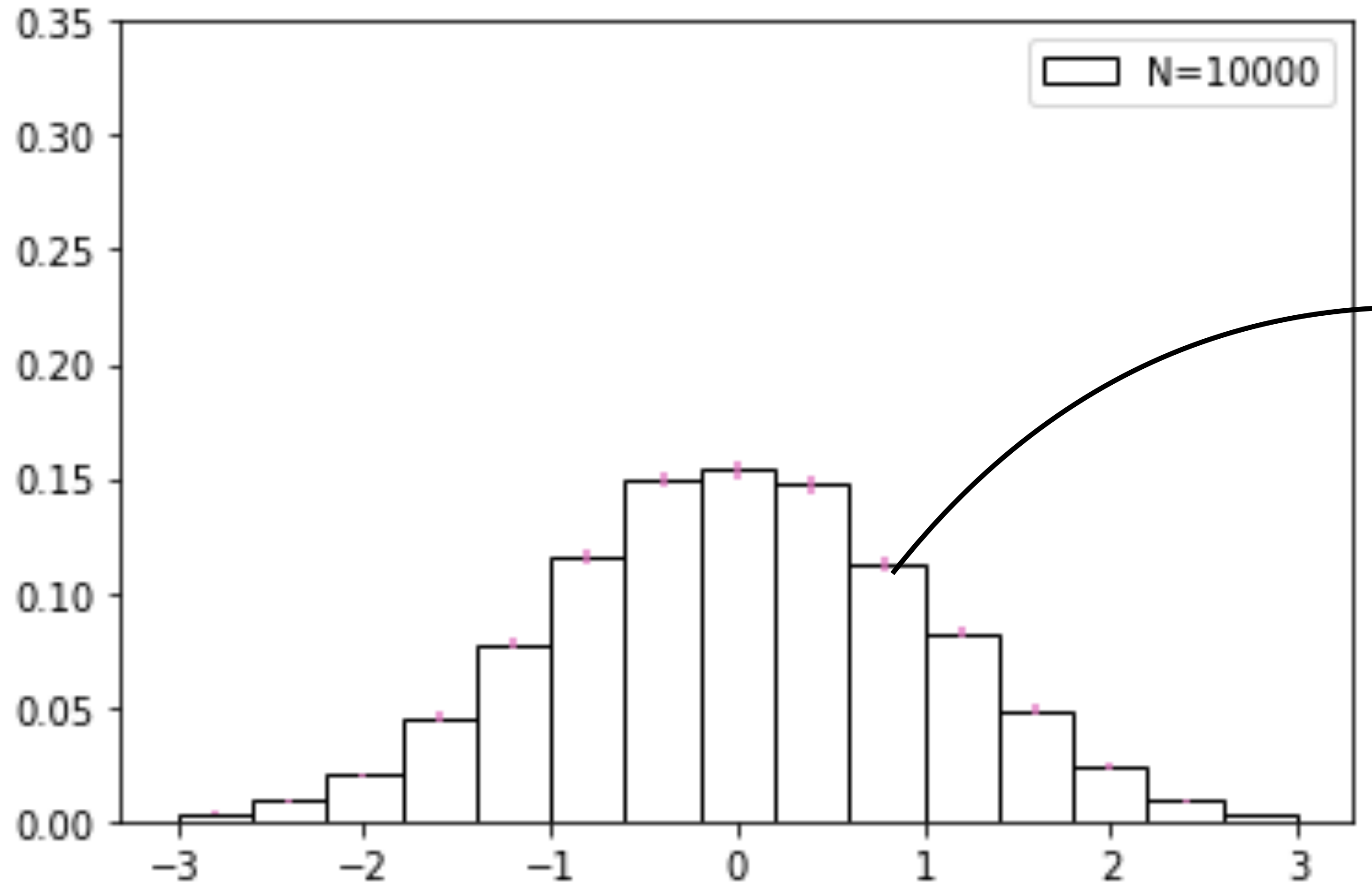




Estimated with pseudo-experiments
Can look wavy when away from asymptotic regime

Generating high-dimensional pseudo-experiments

Traditionally:



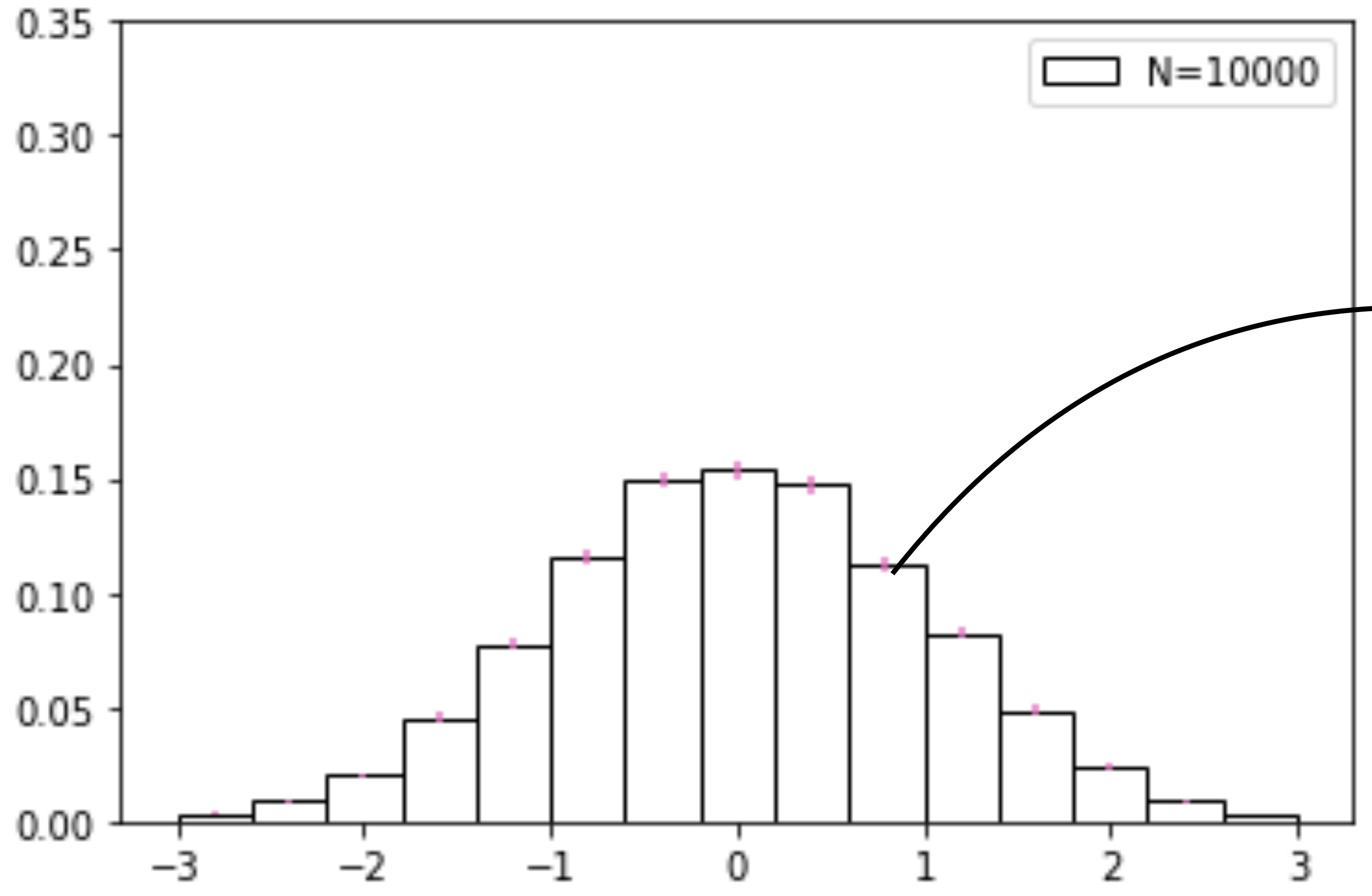
Asimov Histogram

Poisson per bin

$$N_i^{toy} = Poission(N_i^{Asimov})$$

Generating high-dimensional pseudo-experiments

Traditionally:



Poisson per bin

Asimov Histogram

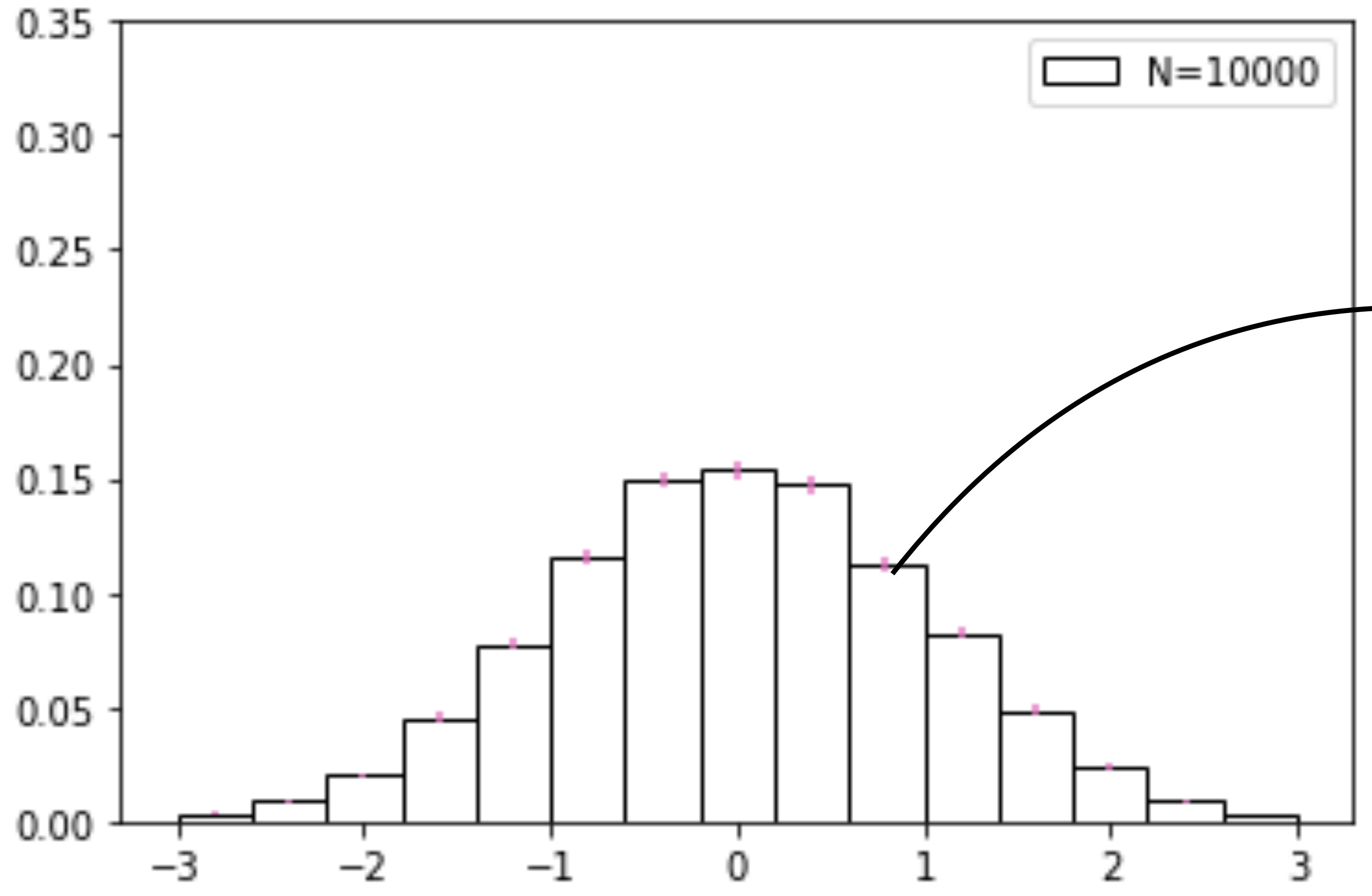
$$N_i^{toy} = Poission(N_i^{Asimov})$$

NSBI?

- More simulated samples?
- Amplify simulated statistics with generative models

Generating high-dimensional pseudo-experiments

Traditionally:



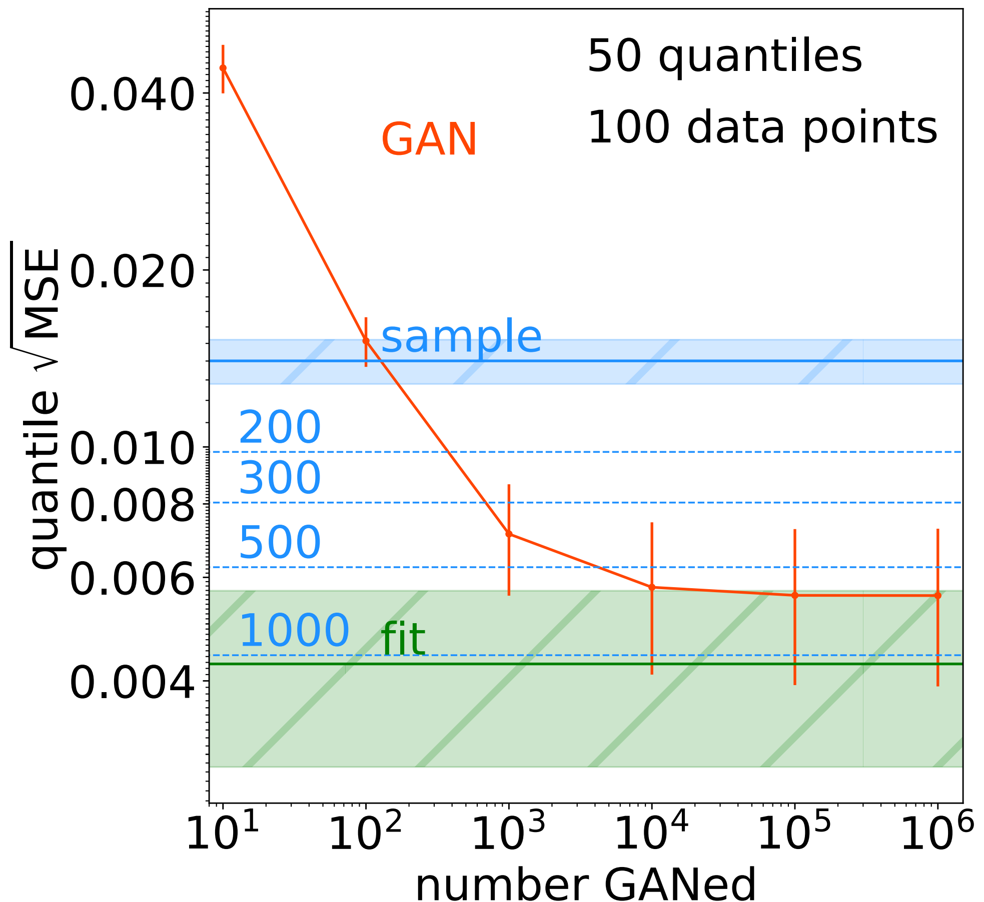
Poisson per bin

Asimov Histogram

$$N_i^{toy} = Poission(N_i^{Asimov})$$

NSBI?

- More simulated samples?
- Amplify simulated statistics with generative models

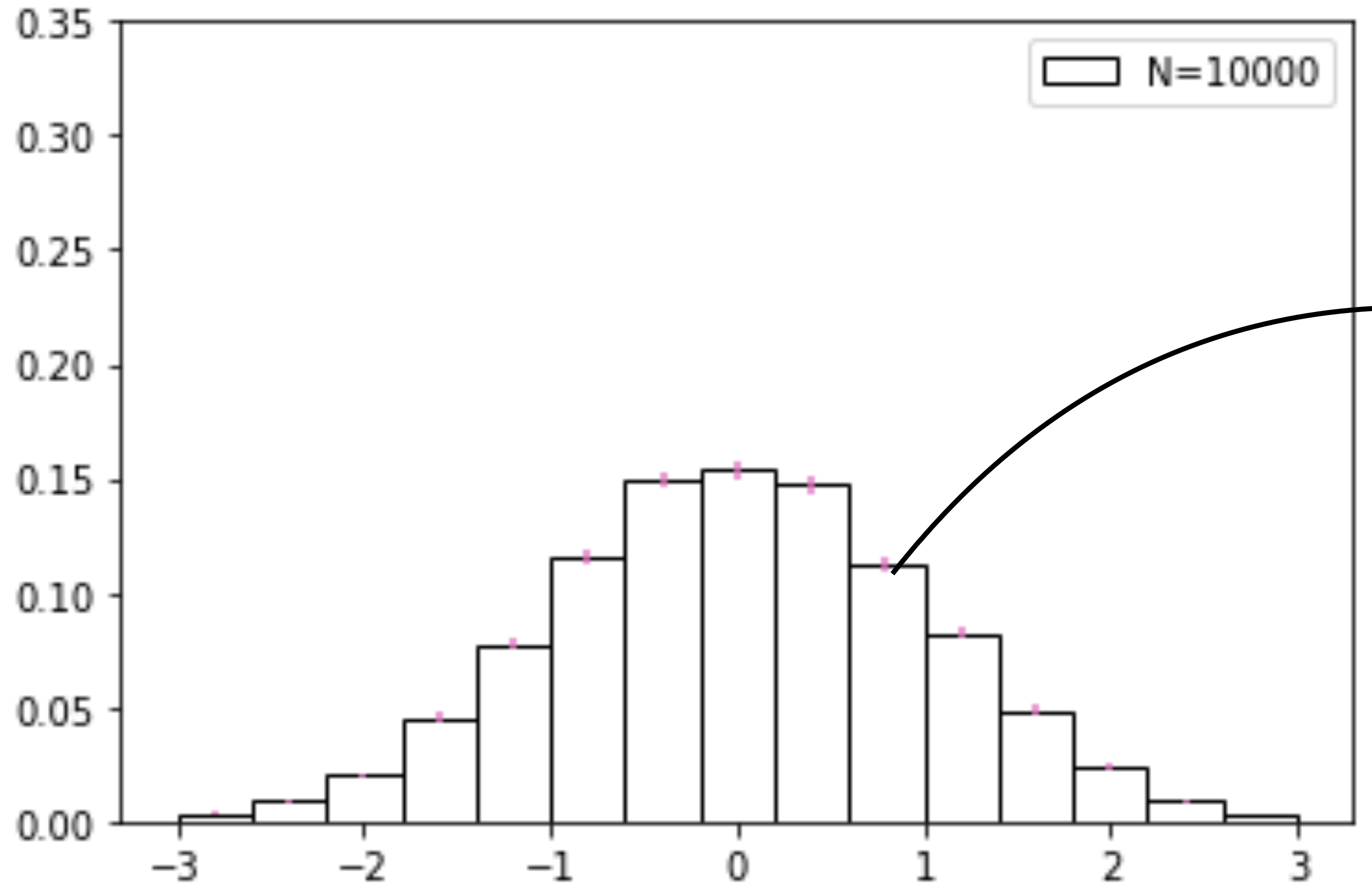


See backup slide

Butter, Diefenbacher et al, arXiv:2008.06545

Throwing high-dimensional toys

Traditionally:

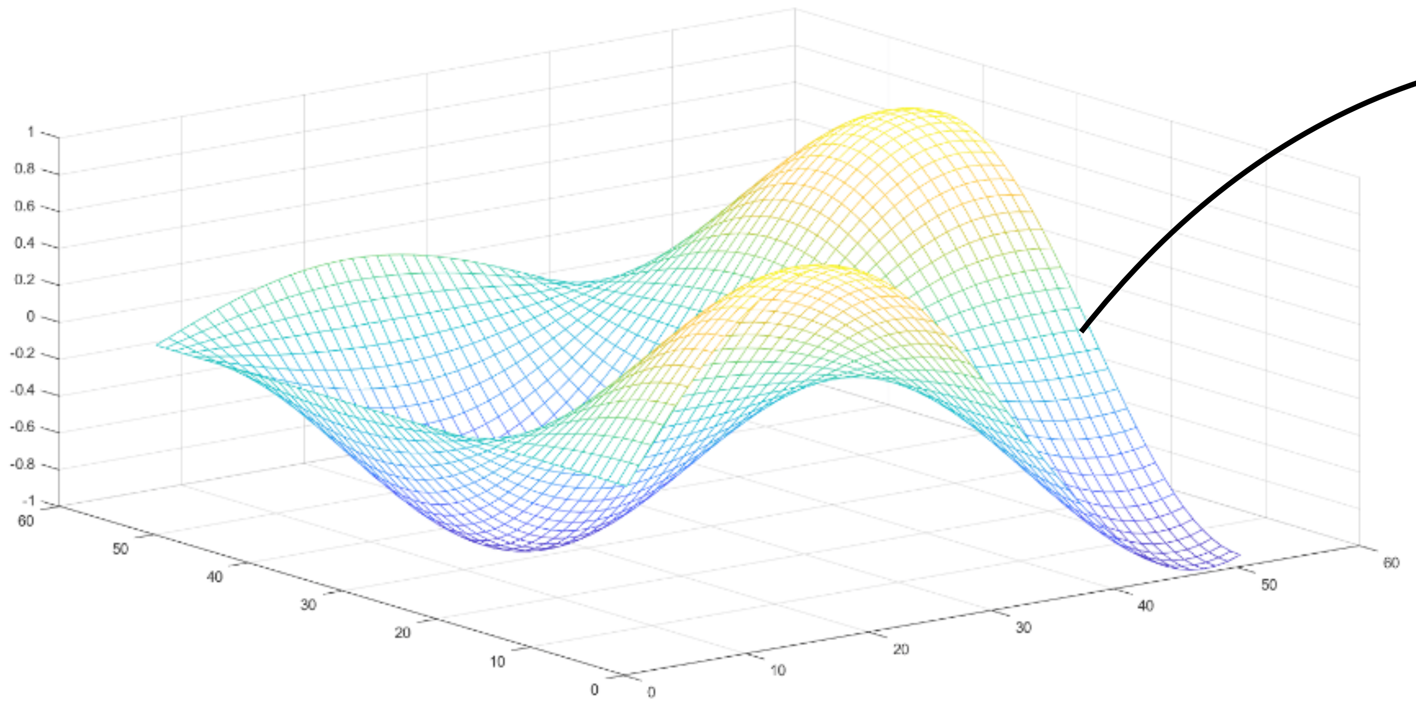


Poisson per bin

Asimov Histogram

$$N_i^{toy} = Poission(N_i^{Asimov})$$

NSBI?



Poisson per event

$$w_i^{toy} = Poisson(w_i^{Asimov})$$

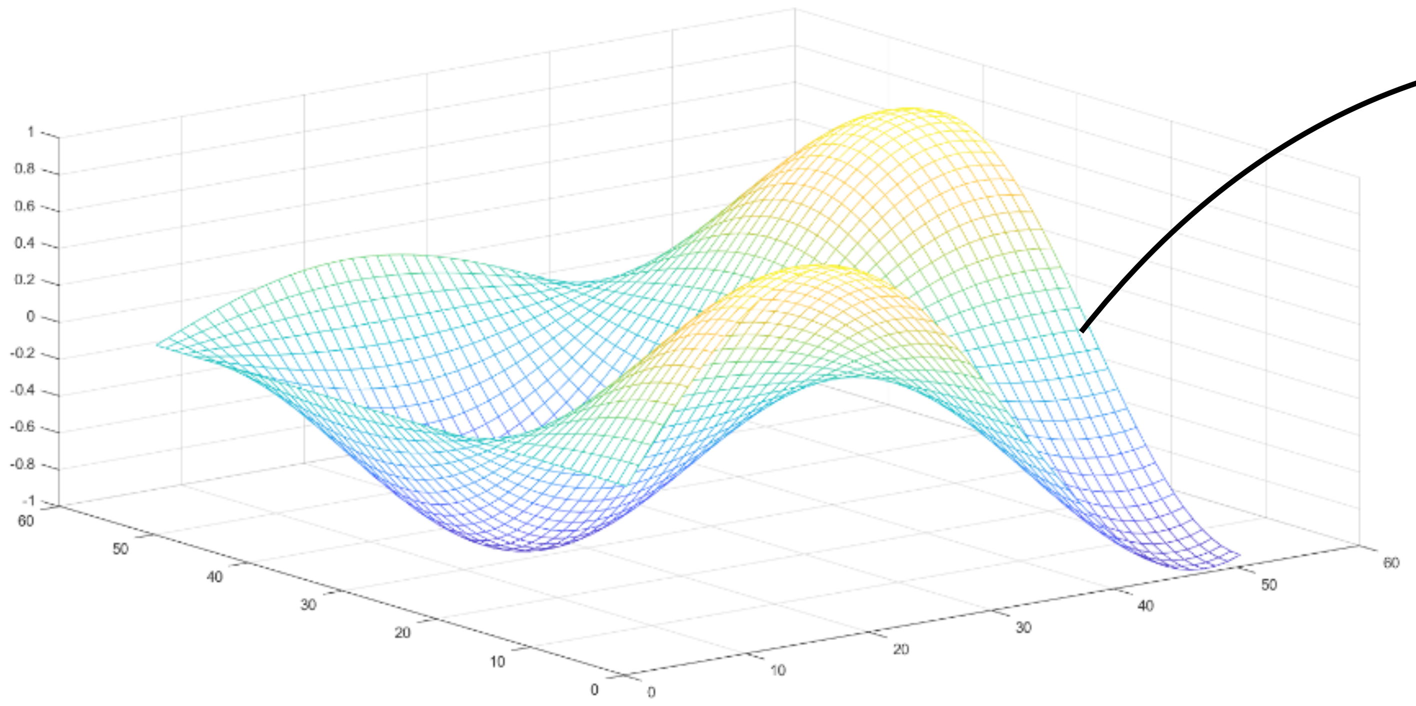
Or unweight + bootstrap

Throwing high-dimensional toys

- Negative weighted samples?
    - Don’t want negative weighted events in toys
    - NN positive resampler (Nachman & Thaler, arXiv:2007.11586) too expensive to perform+validate for each θ
- Using any ML method provokes the question: “Use NSBI to validate NSBI ?”

- Can we have a definitive statistical method to throw high-dimensional toys?

NSBI?



Poisson per event

$$w_i^{toy} = Poisson(w_i^{Asimov})$$

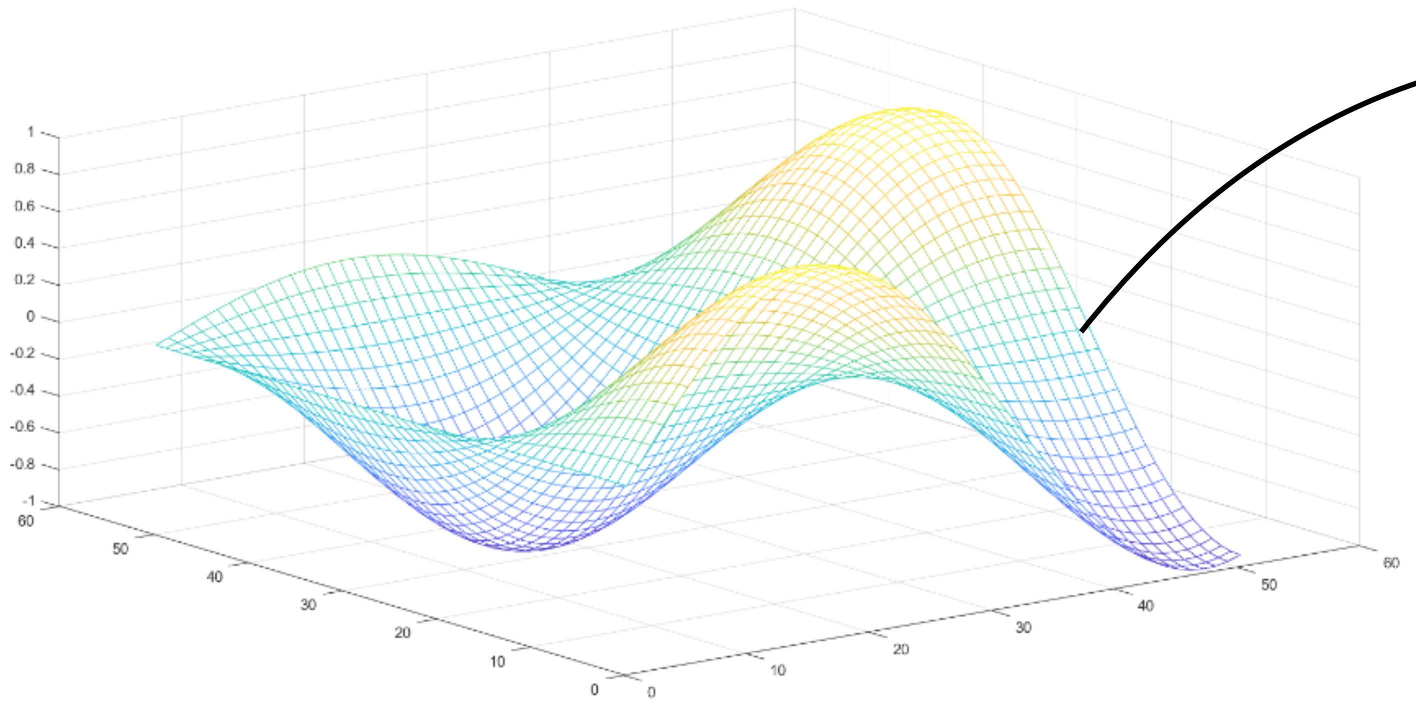
(‘Unweighted’ events, i.e. integer weights)

Throwing high-dimensional toys

- Negative weighted samples?
  - Don’t want negative weighted events in toys
  - NN positive resampler (Nachman & Thaler, arXiv:2007.11586) too expensive to perform+validate for each θ
- Using any ML method provokes the question: “Use NSBI to validate NSBI ?”

- Can we have a definitive statistical method to throw high-dimensional toys?

NSBI?



Poisson per event

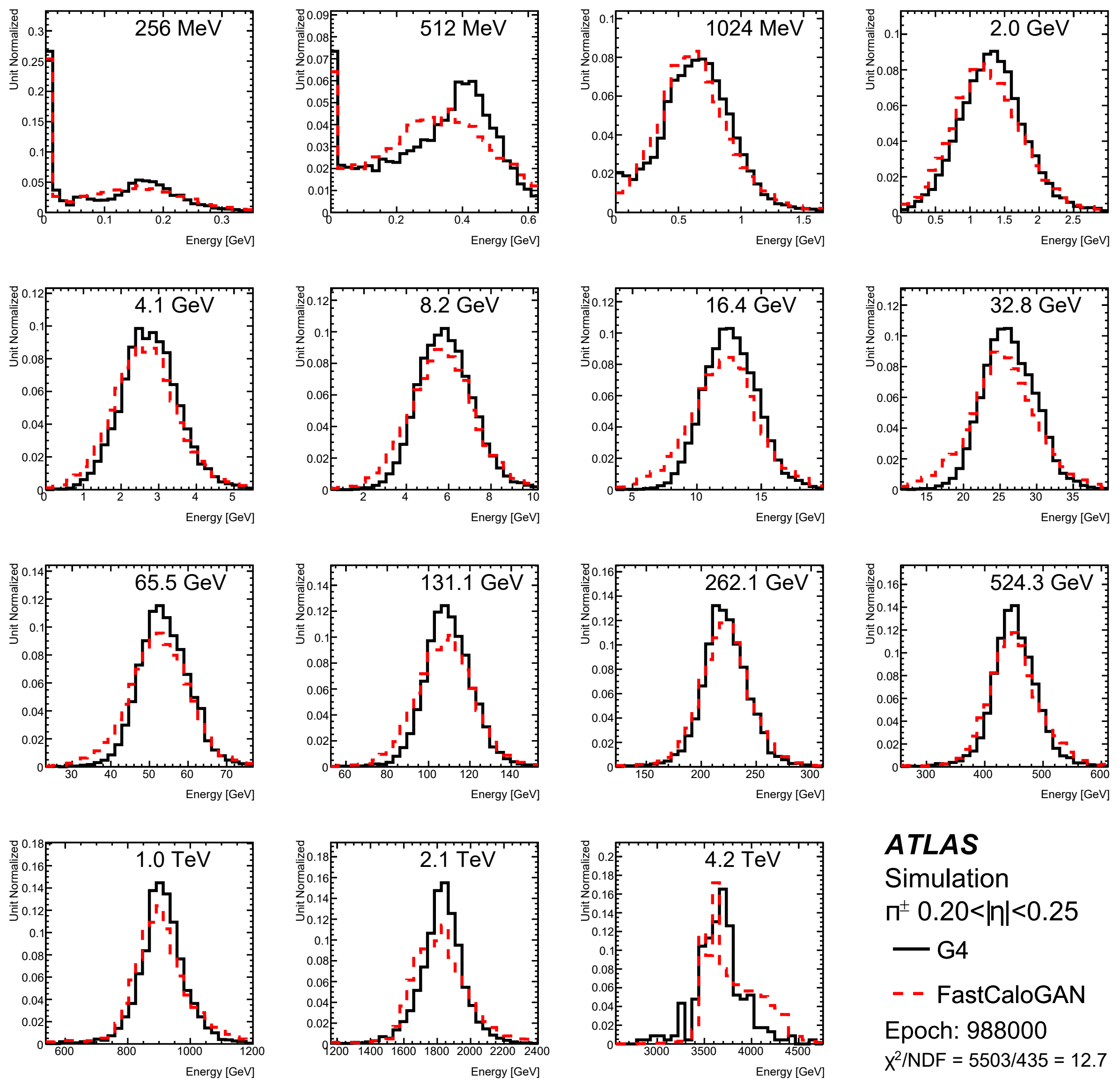
$$w_i^{toy} = Poisson(w_i^{Asimov})$$

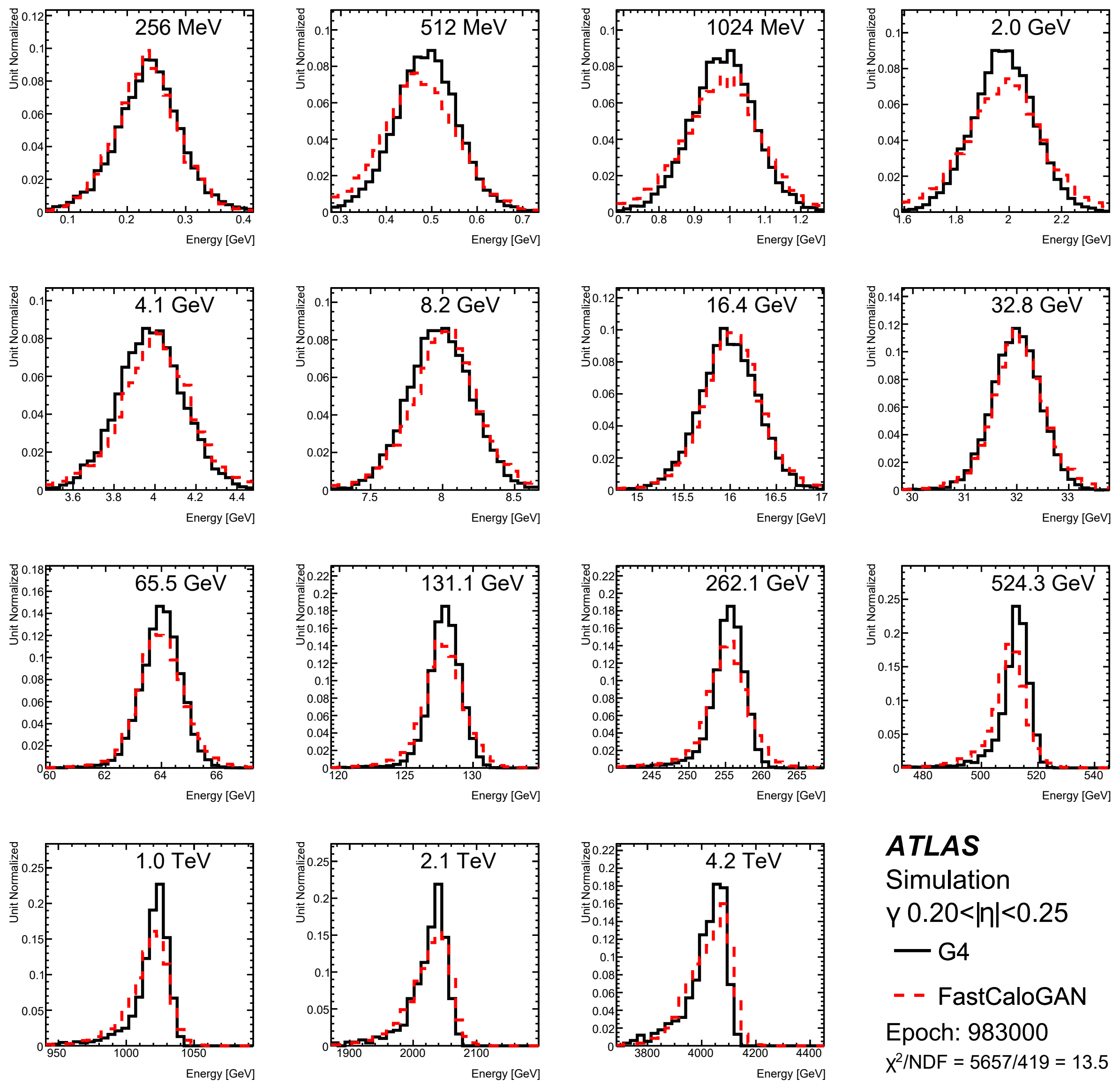
(‘Unweighted’ events, i.e. integer weights)

If you think you have a non-ML solution, come chat with me!

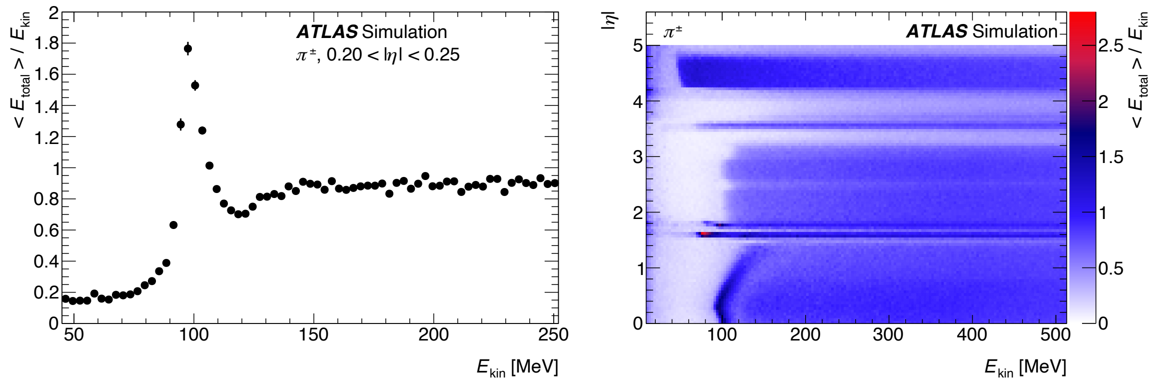
Automating network evaluation: Issue for NSBI and generative models

Evaluating Fast Calo Simulators

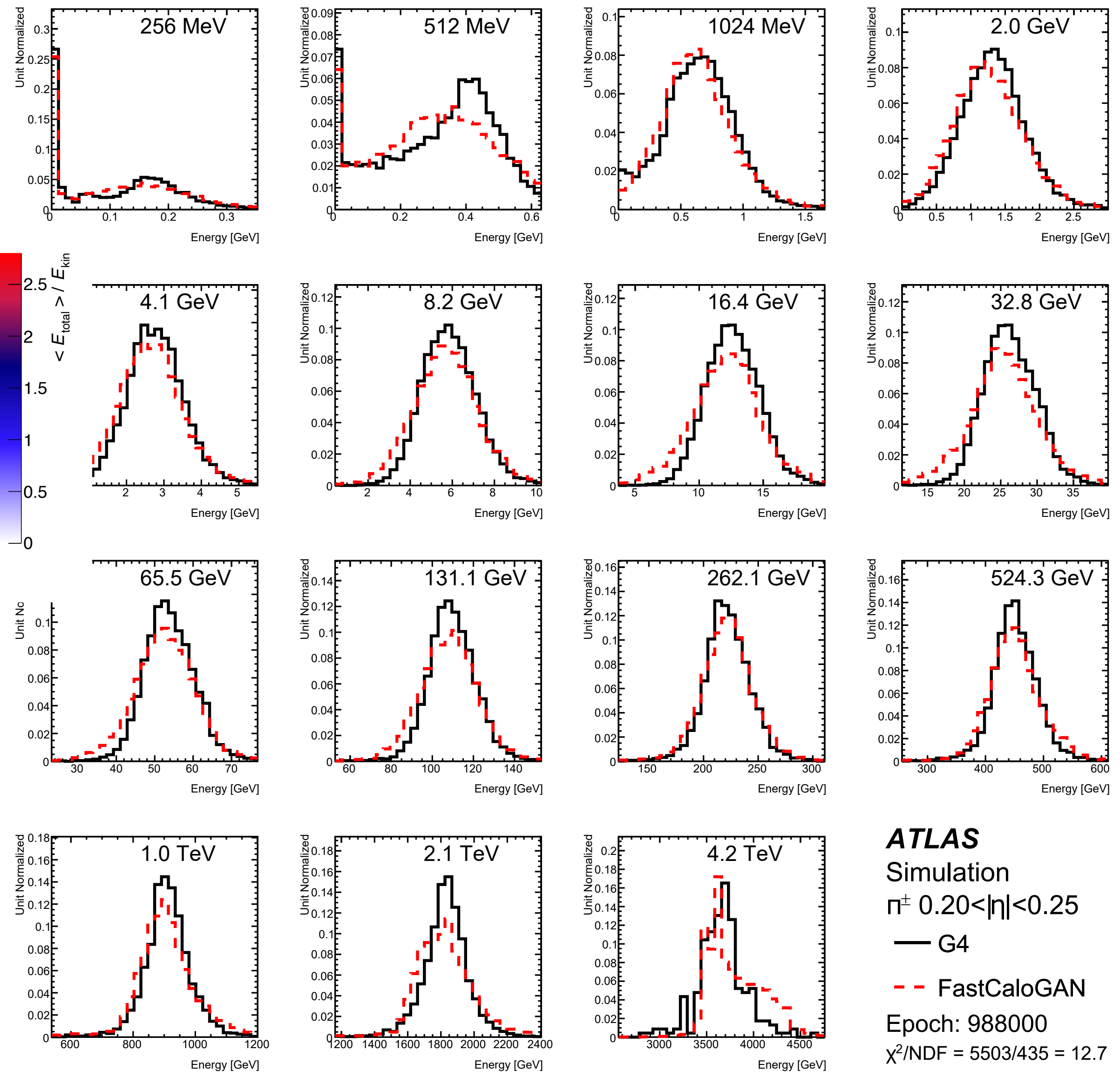


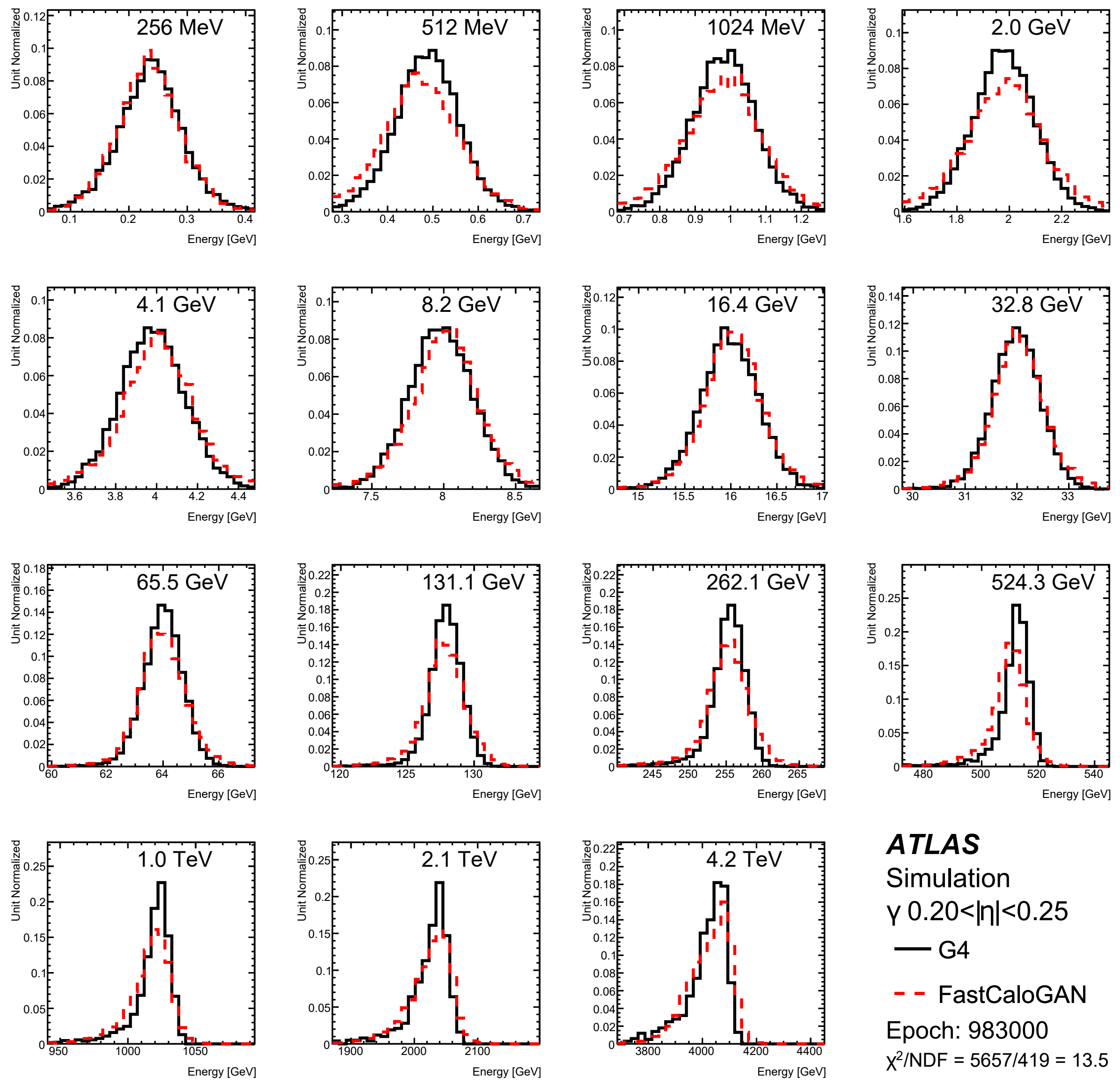


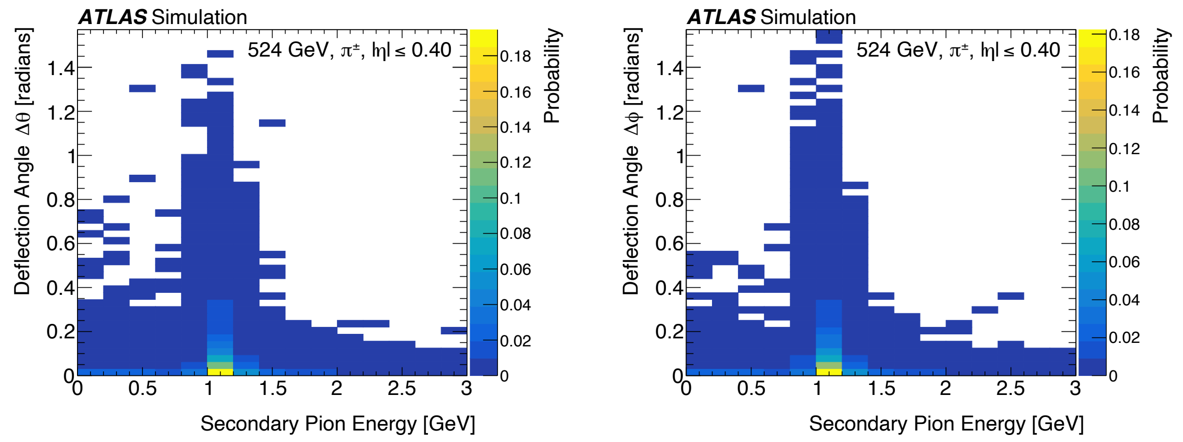
Evaluating Fast Calo Simulators



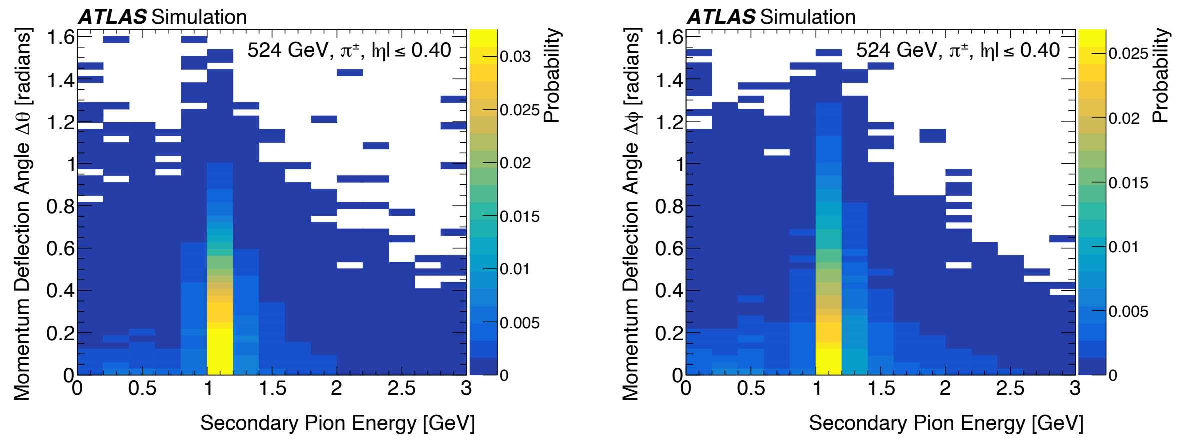
(a) (b)





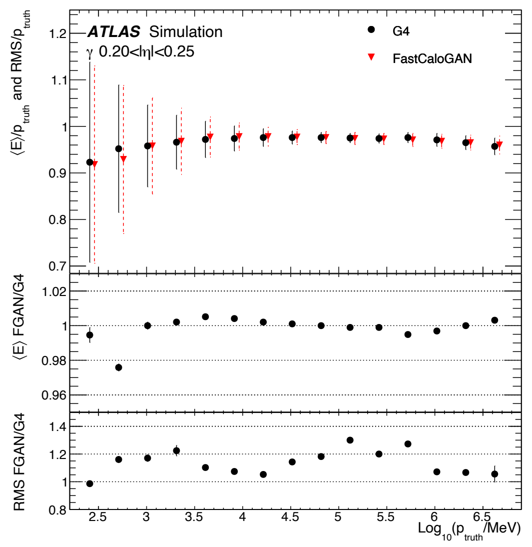


(a) (b)

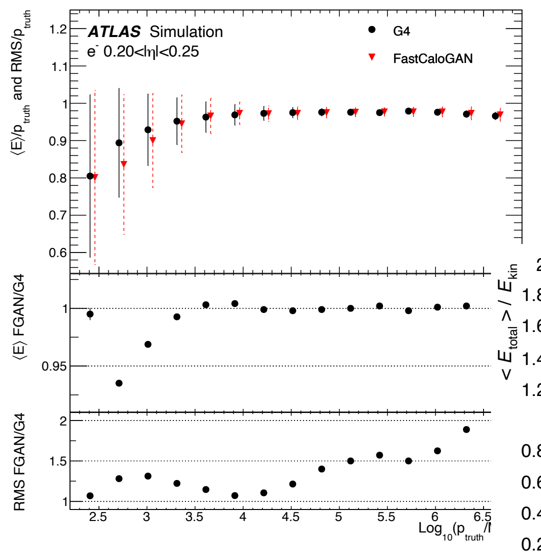


(c) (d)

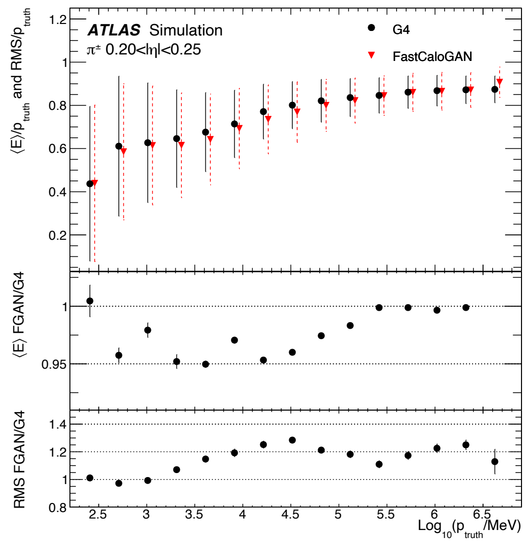
Evaluating Fast Calo Simulators



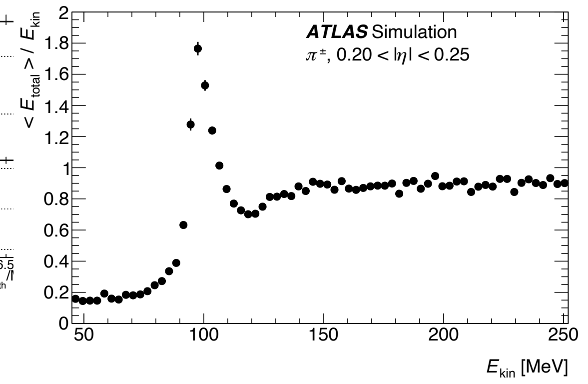
(a)



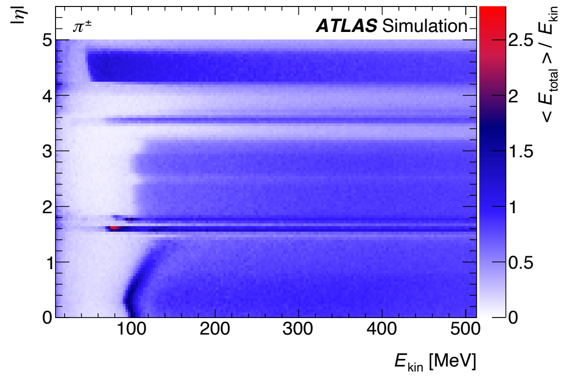
(b)



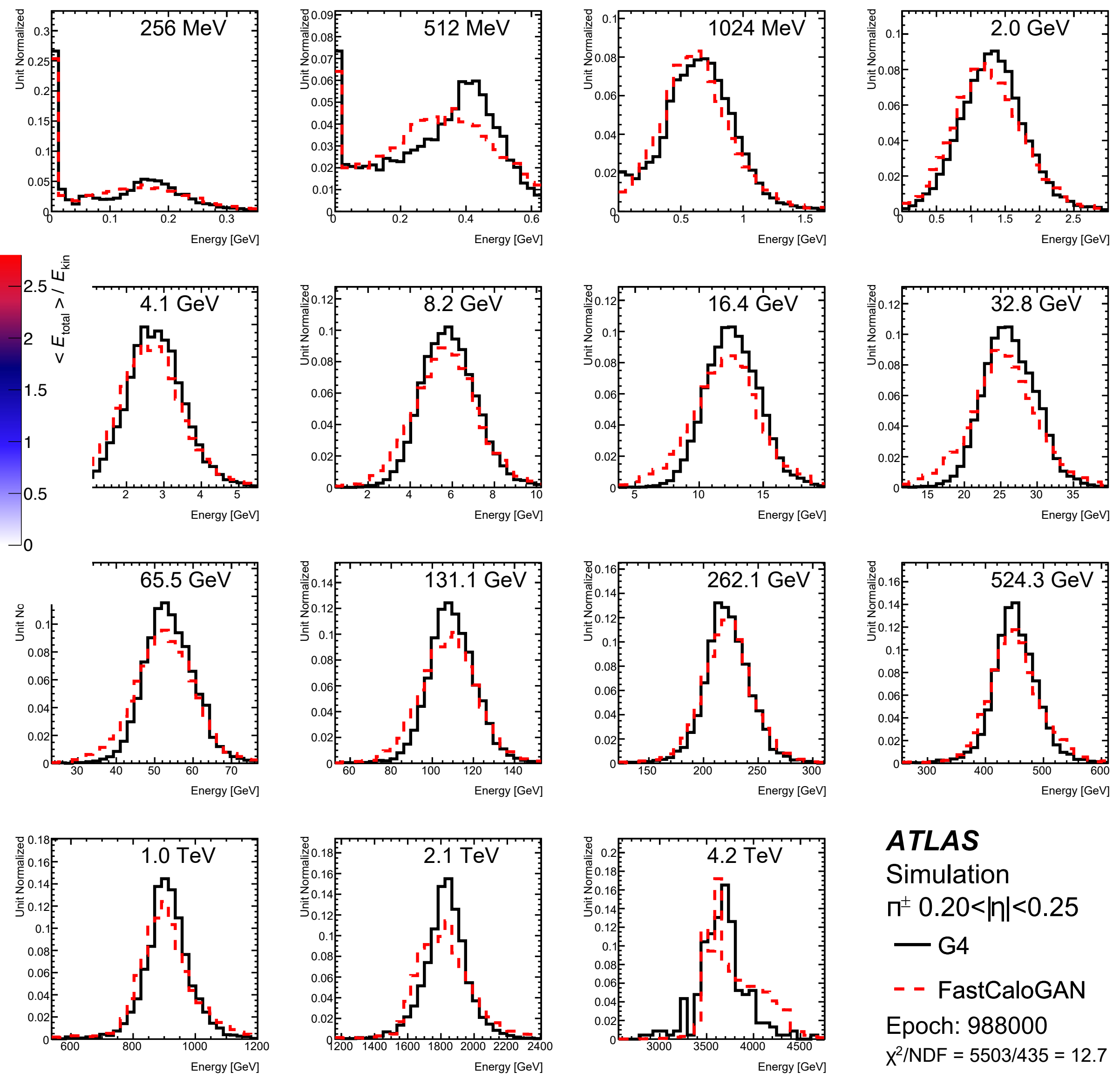
(c)

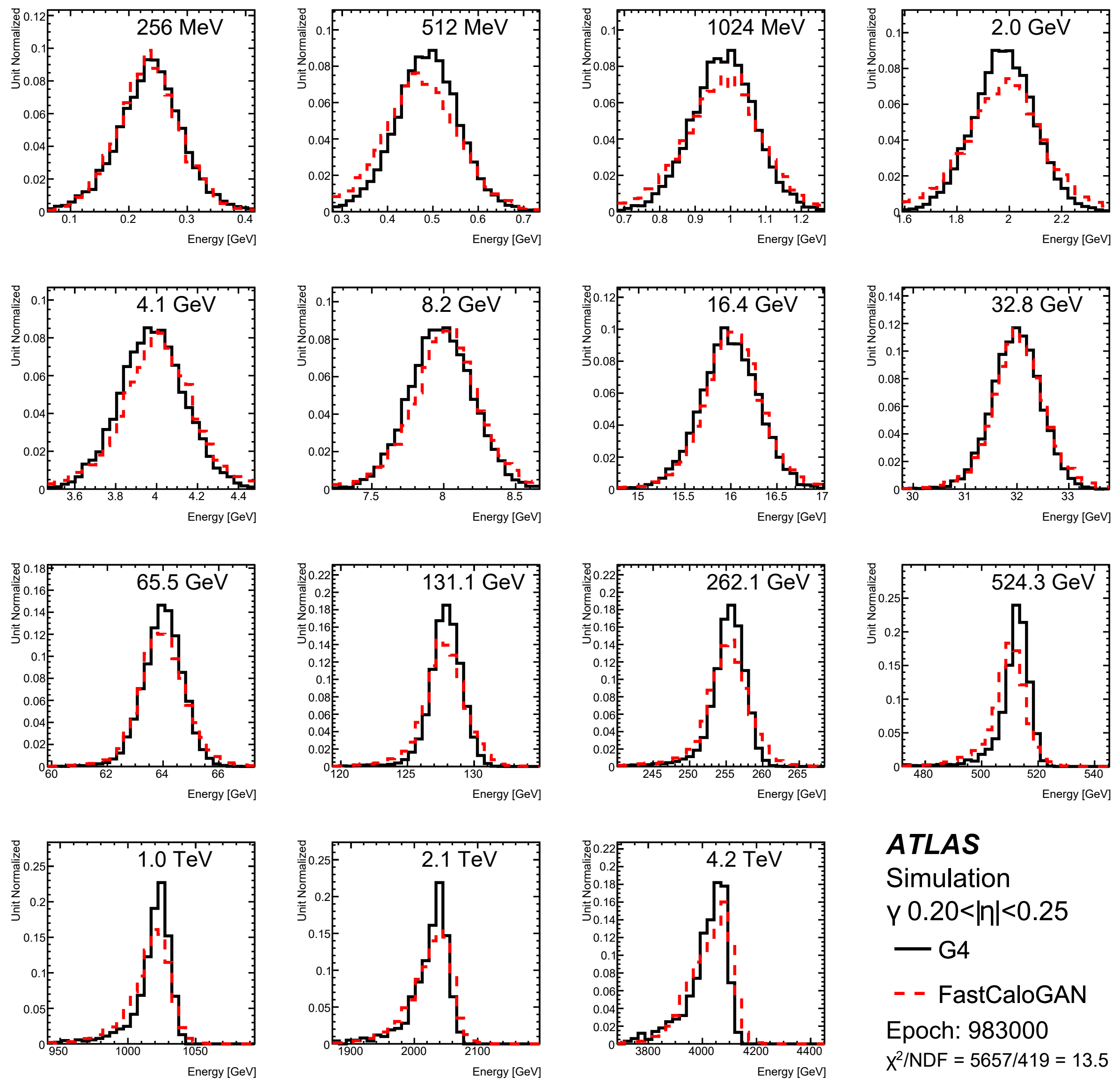


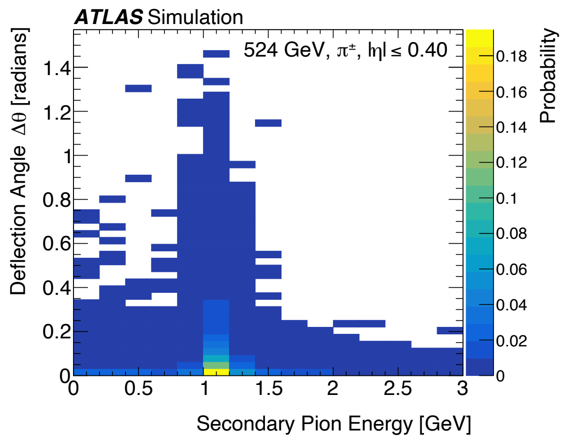
(a)



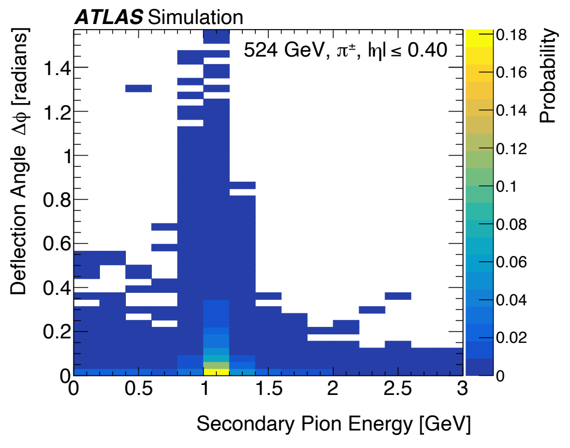
(b)



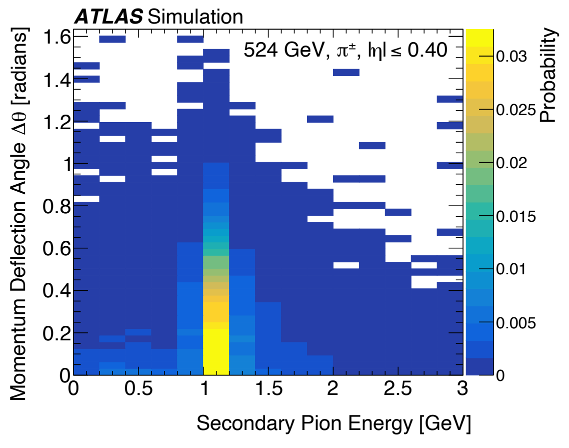




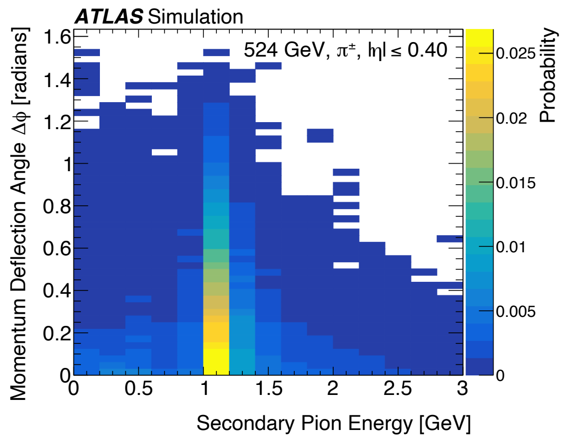
(a)



(b)

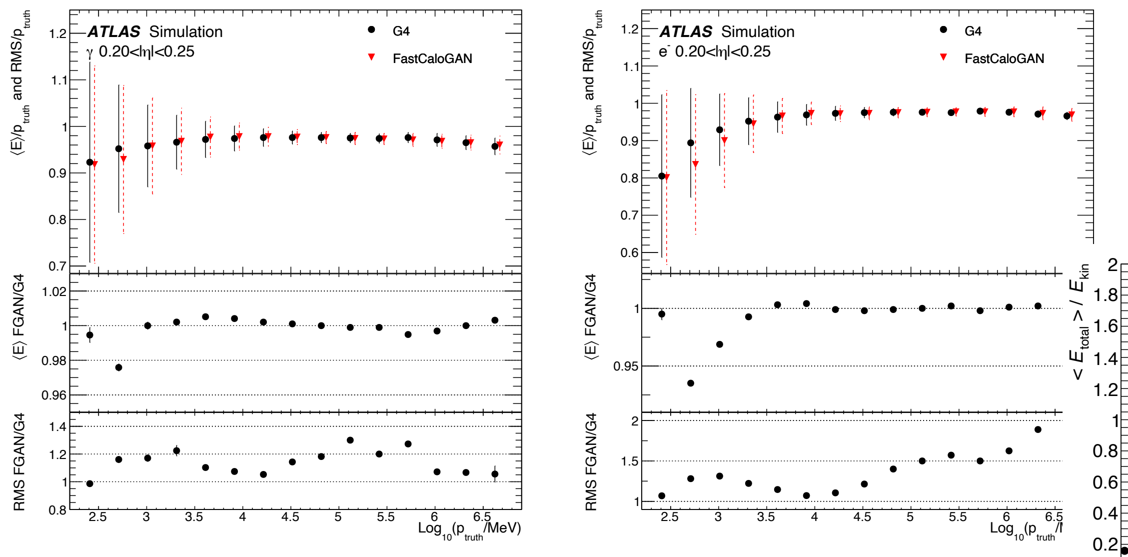


(c)

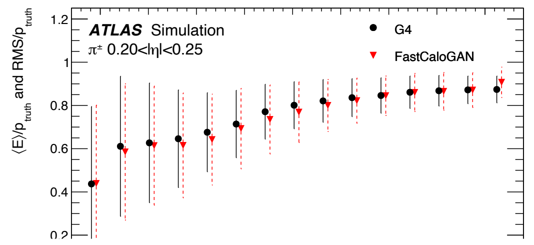


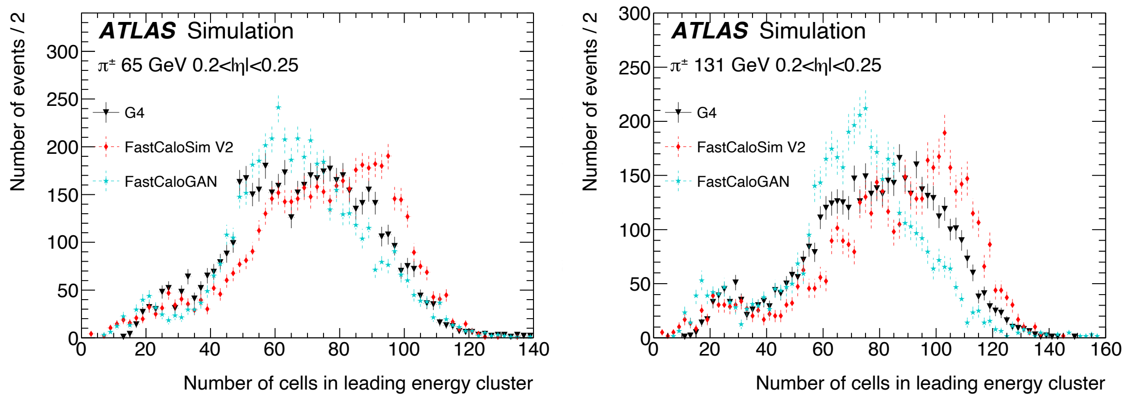
(d)

Evaluating Fast Calo Simulators

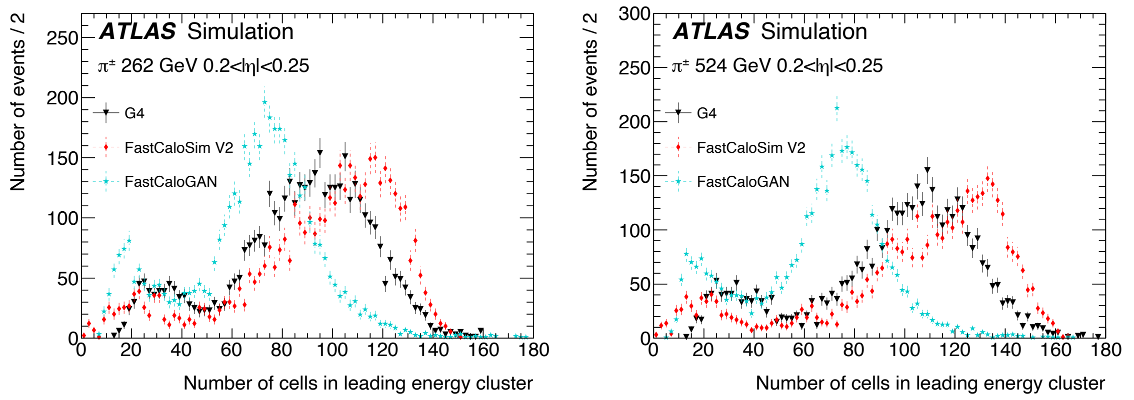


(a) (b)

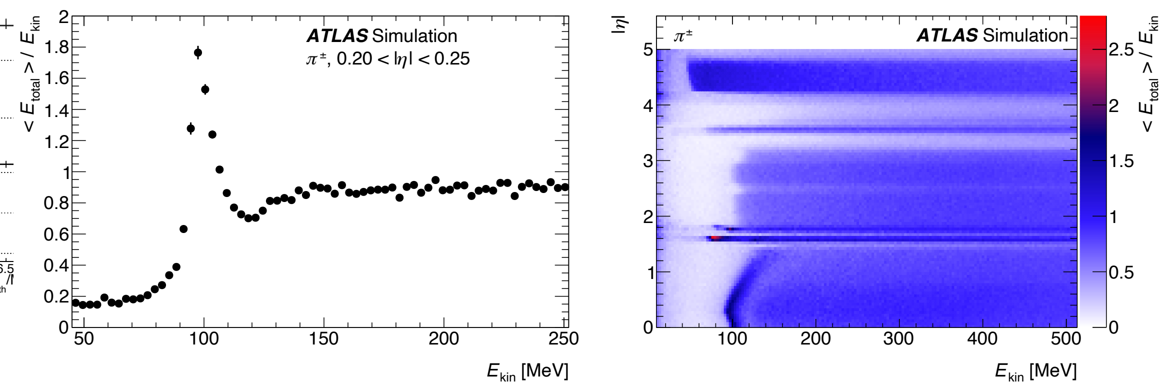




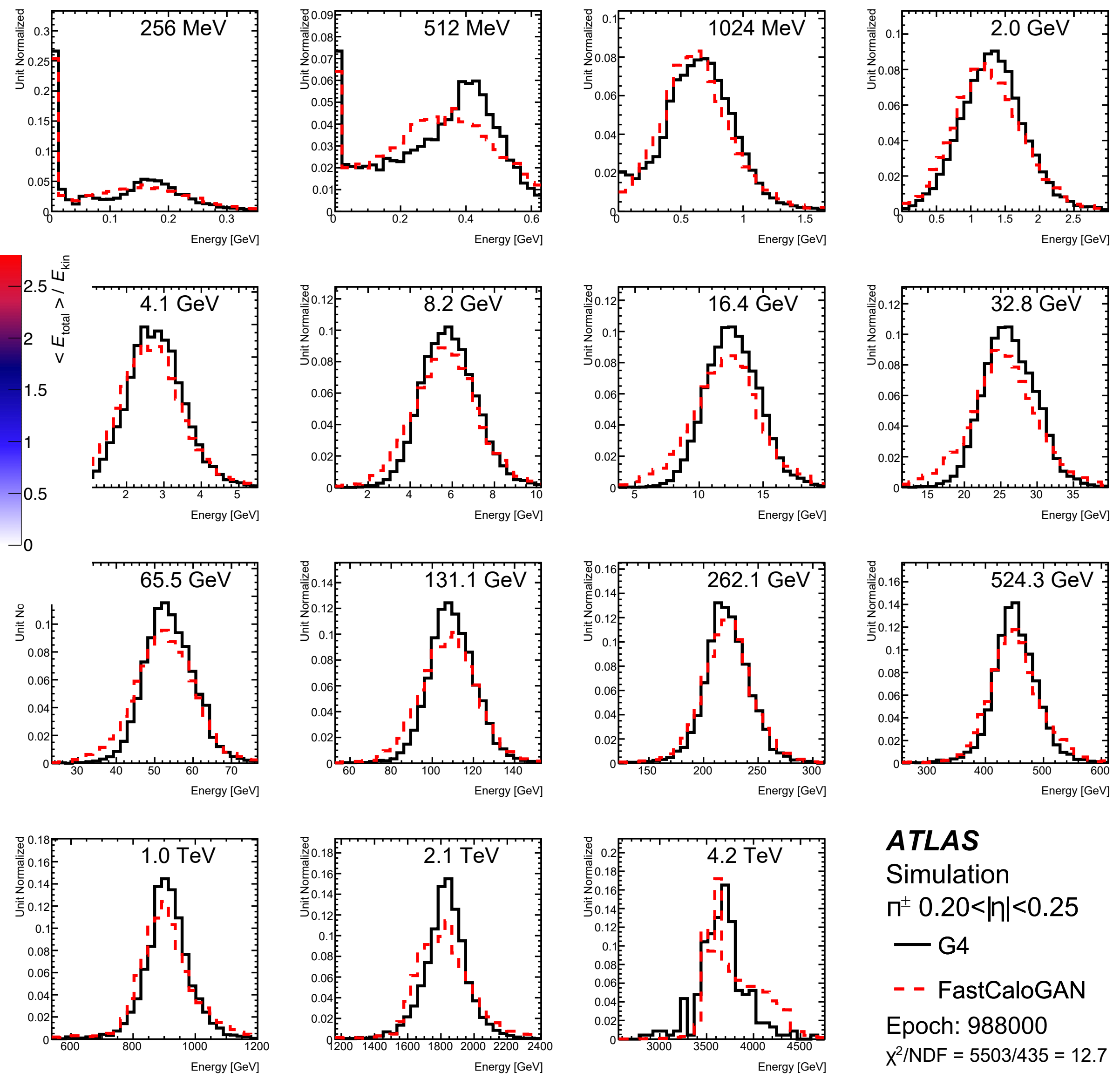
(a) (b)

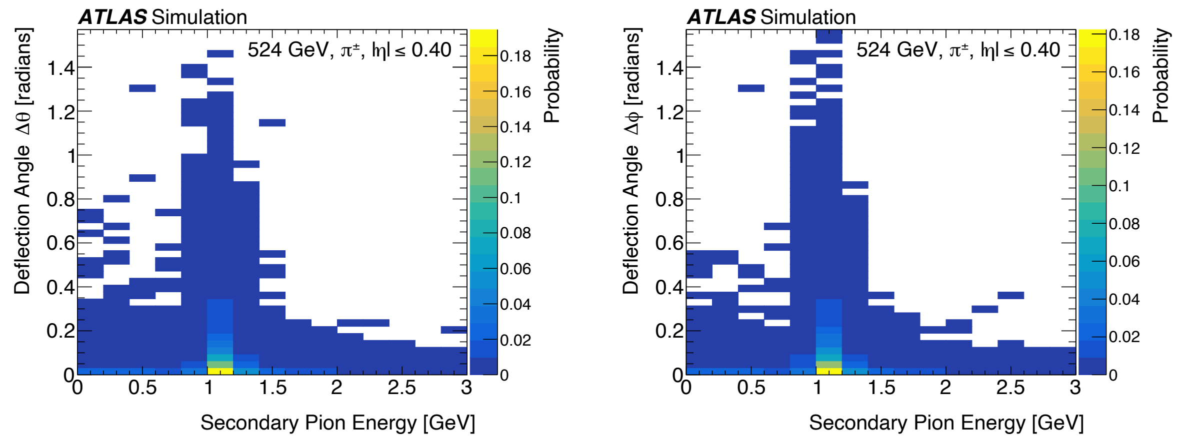


(c) (d)

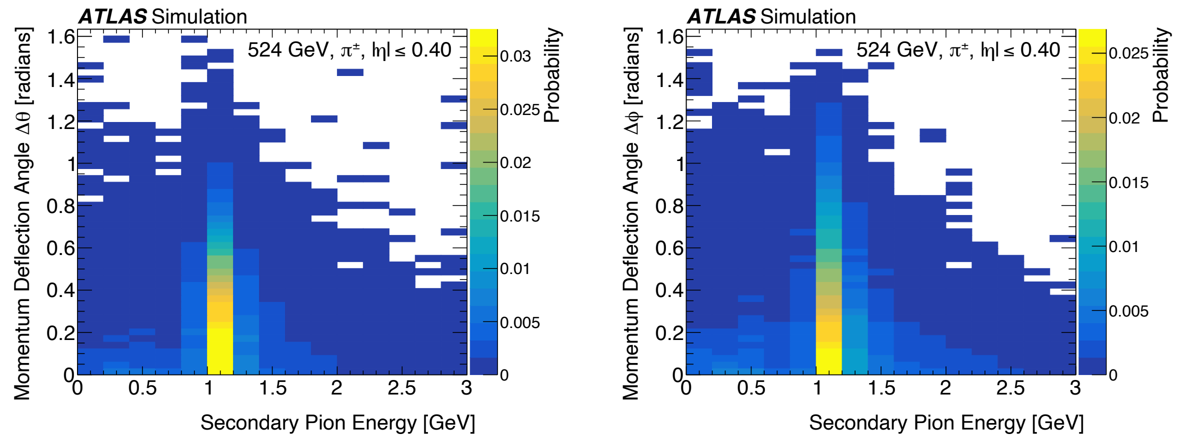


(a) (b)

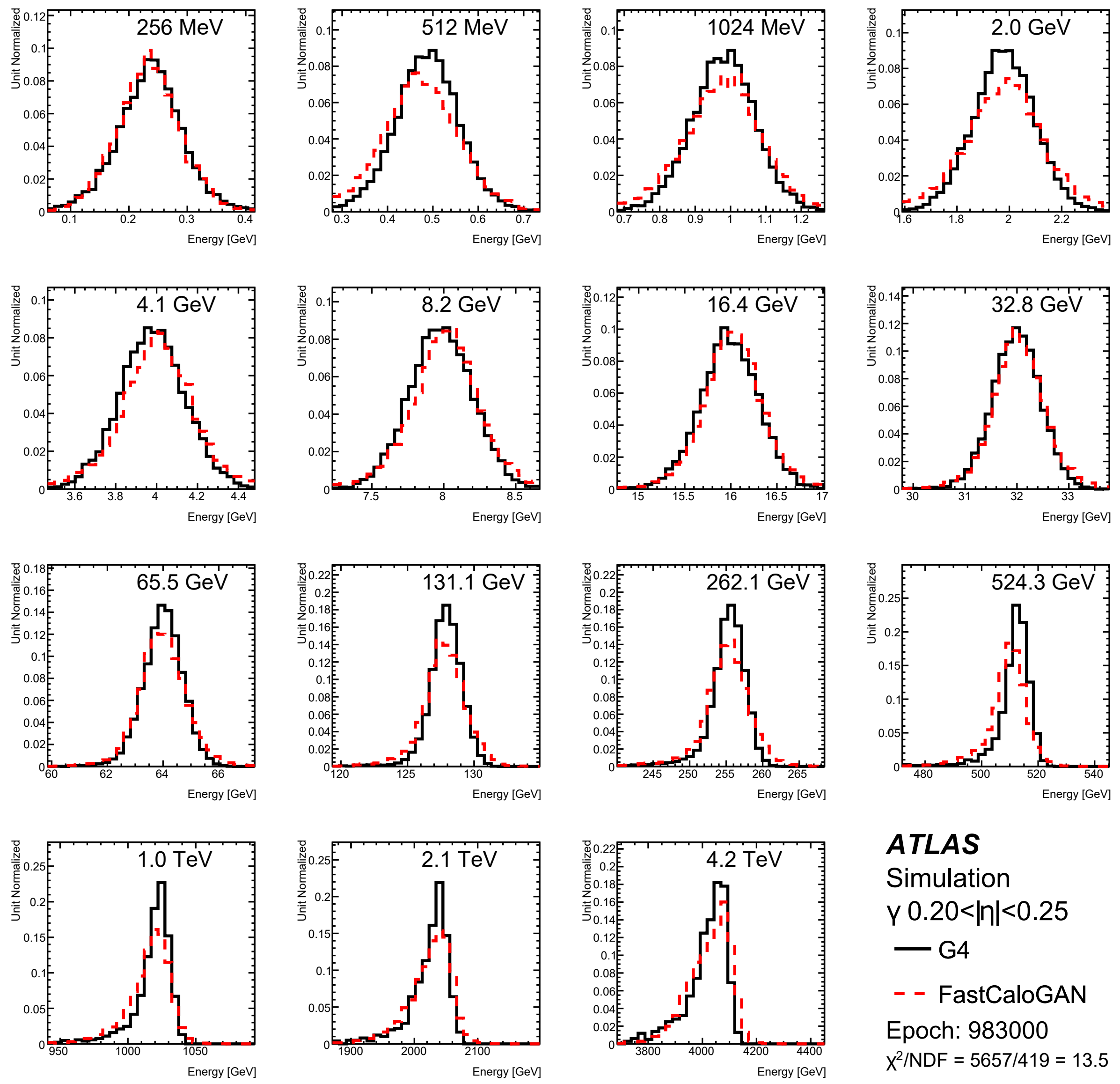




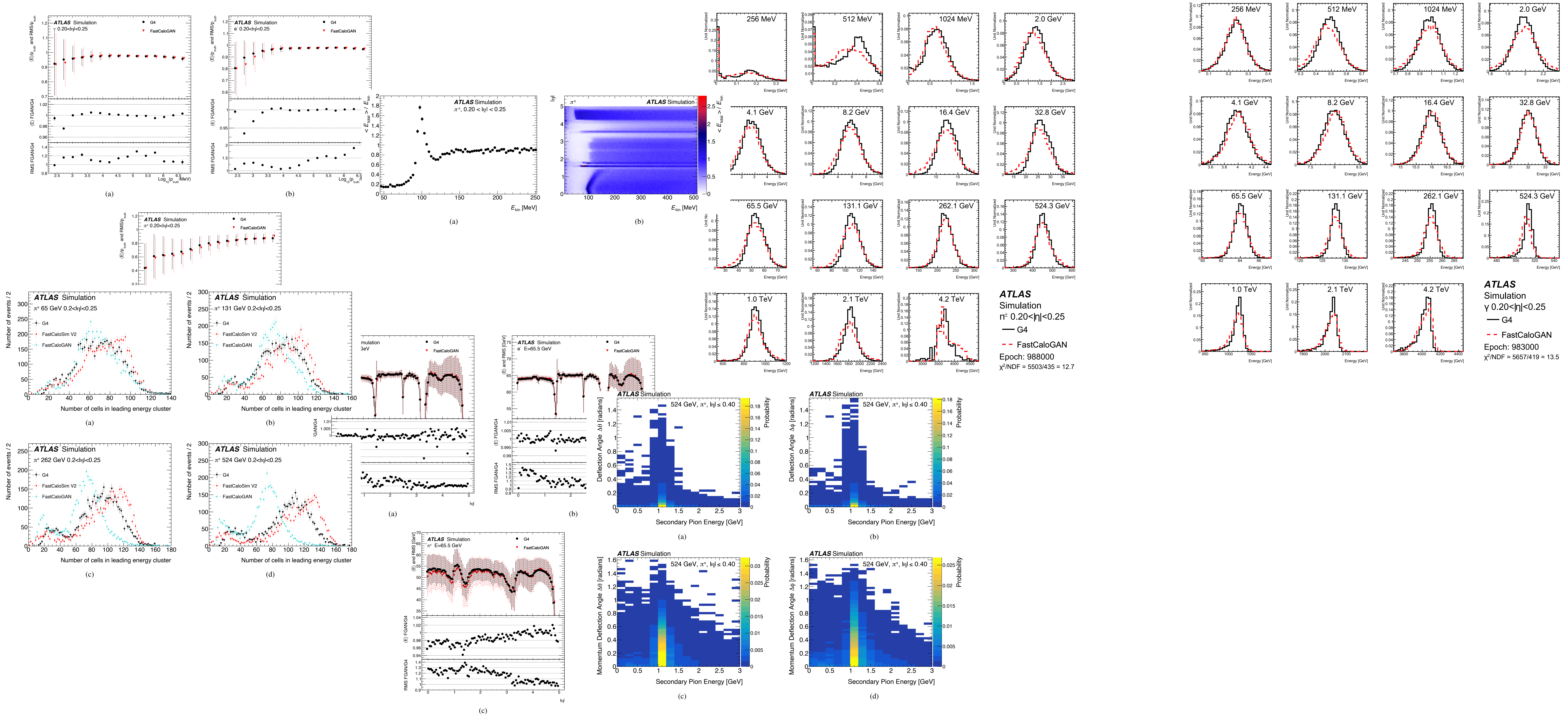
(a) (b)



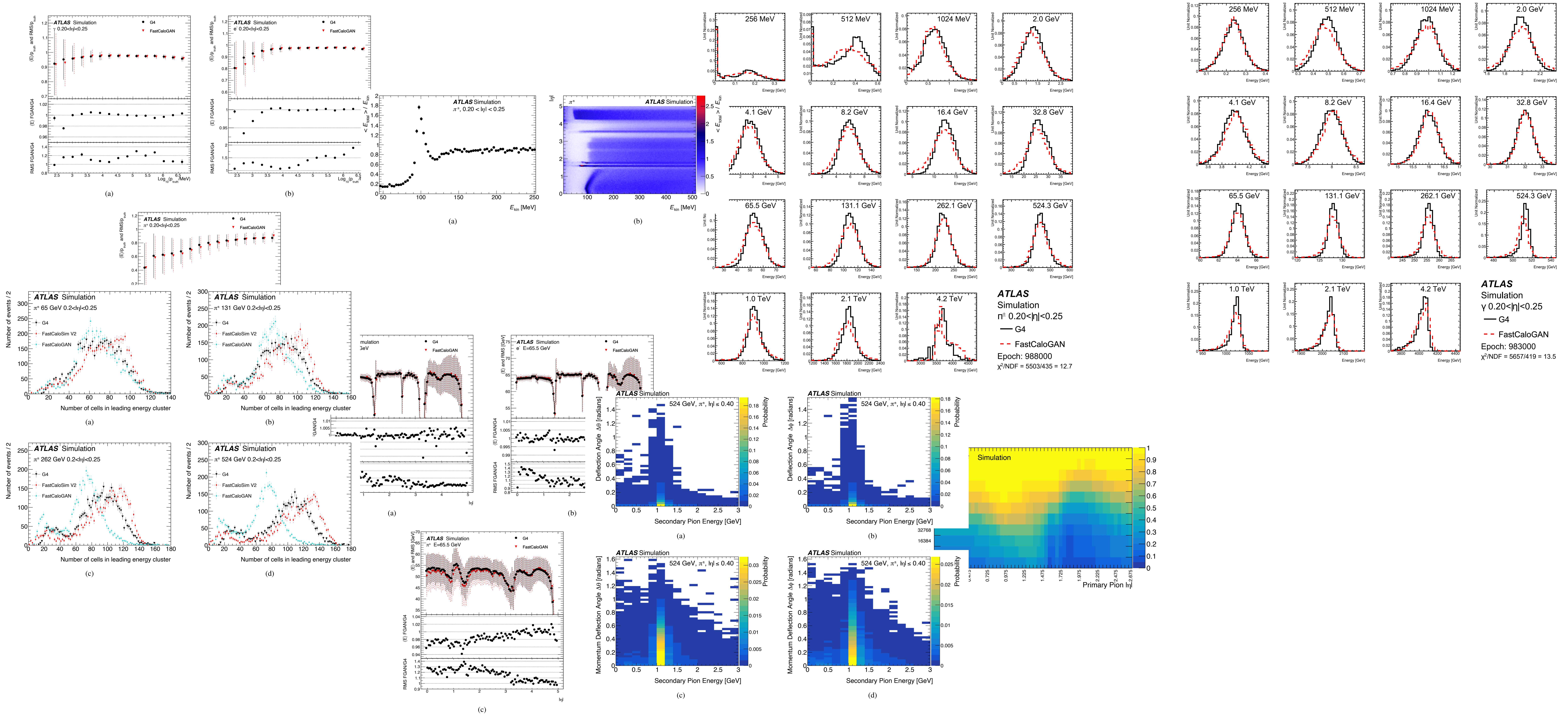
(c) (d)



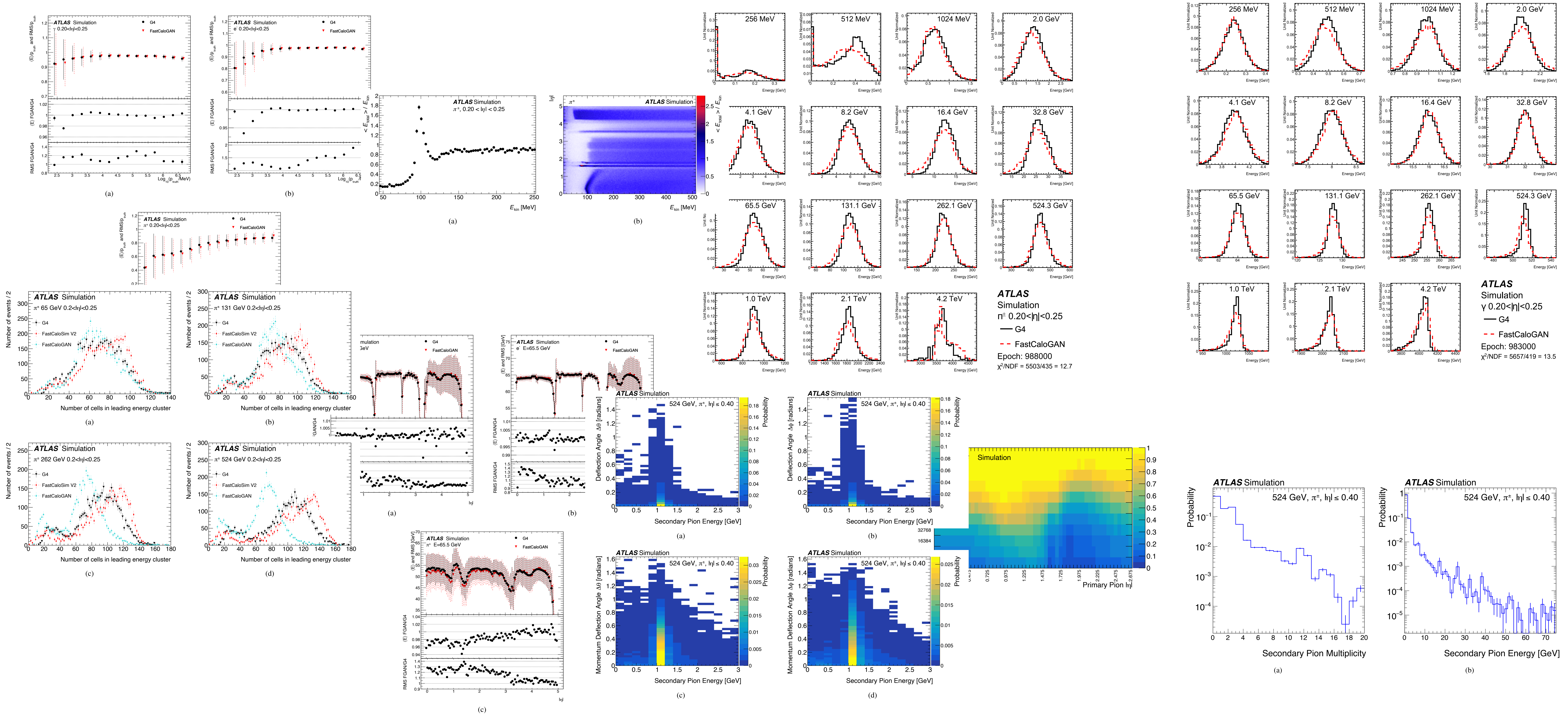
Evaluating Fast Calo Simulators



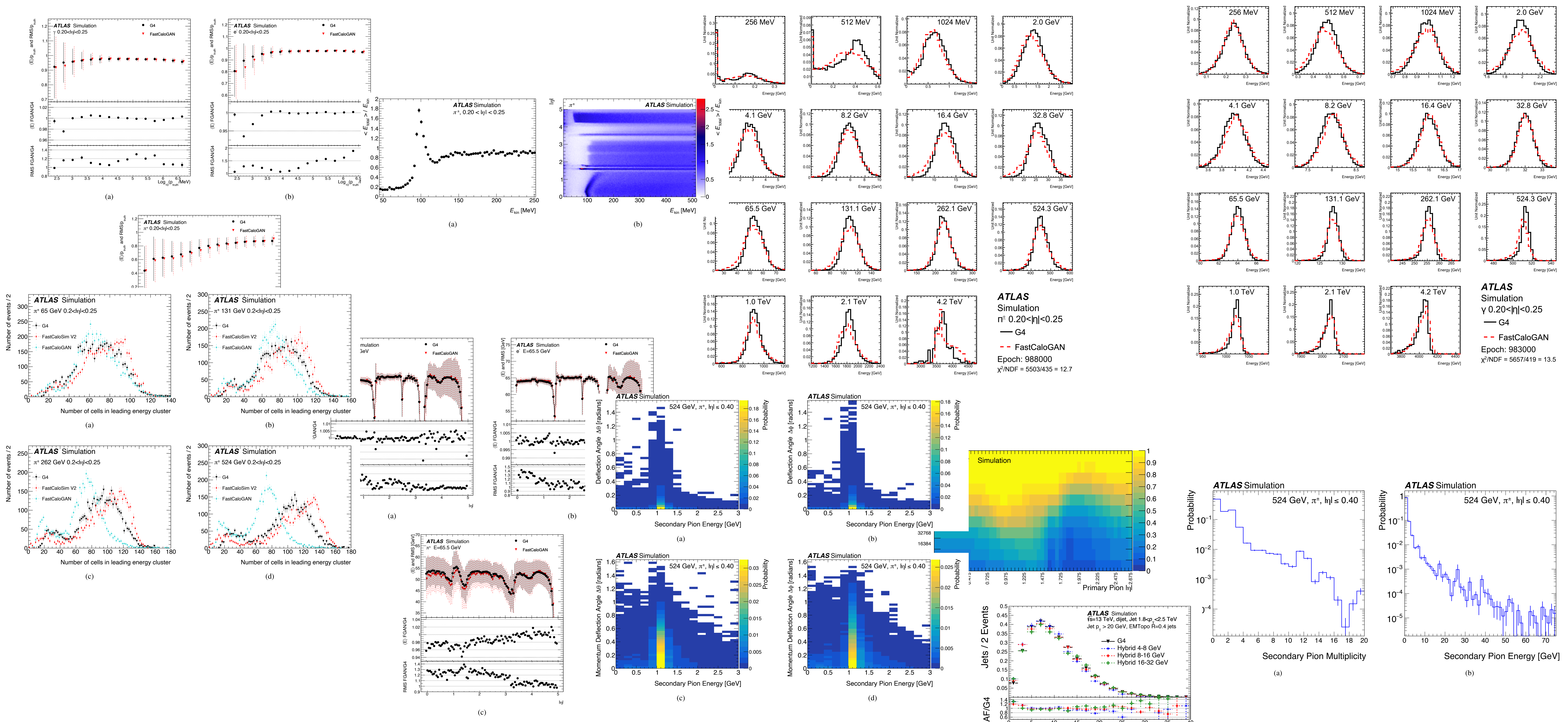
Evaluating Fast Calo Simulators



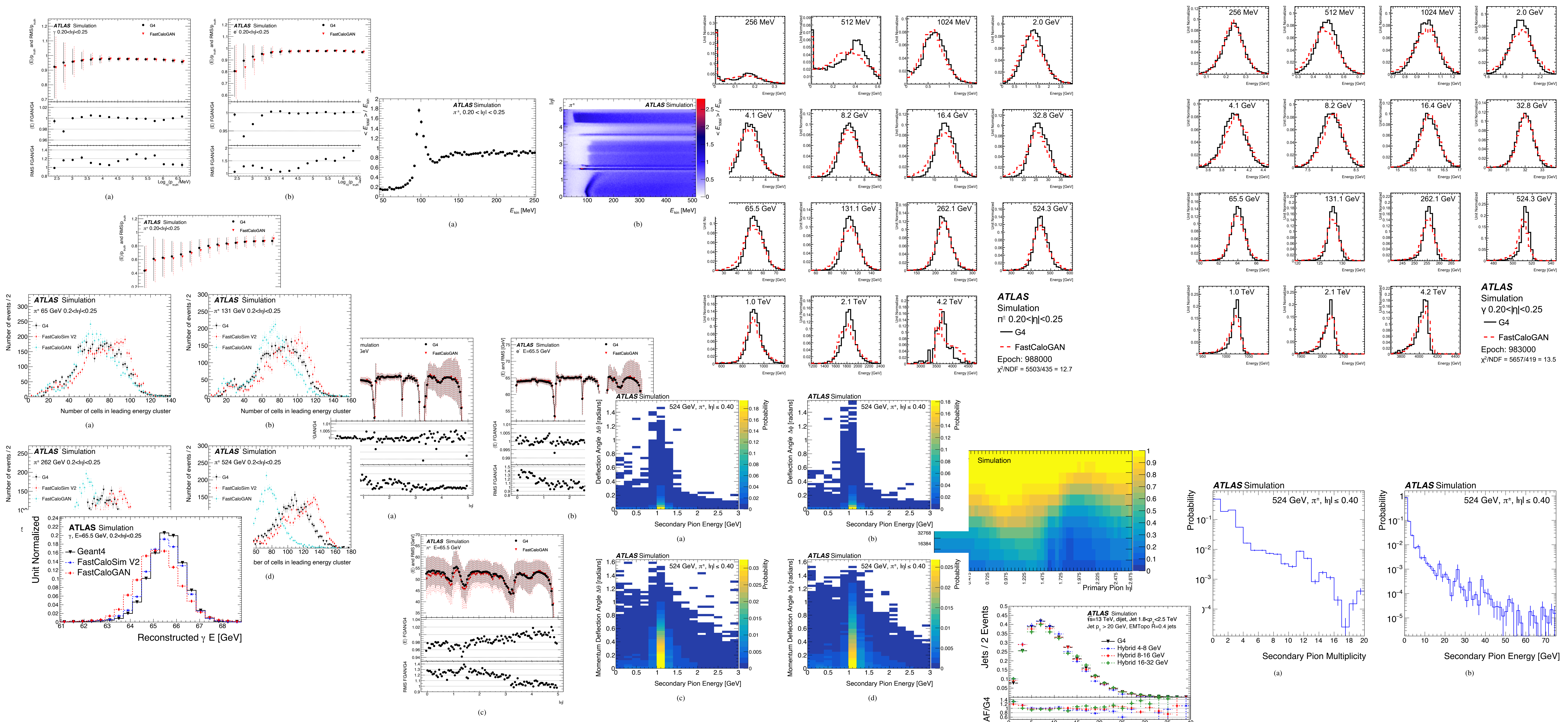
Evaluating Fast Calo Simulators



Evaluating Fast Calo Simulators



Evaluating Fast Calo Simulators



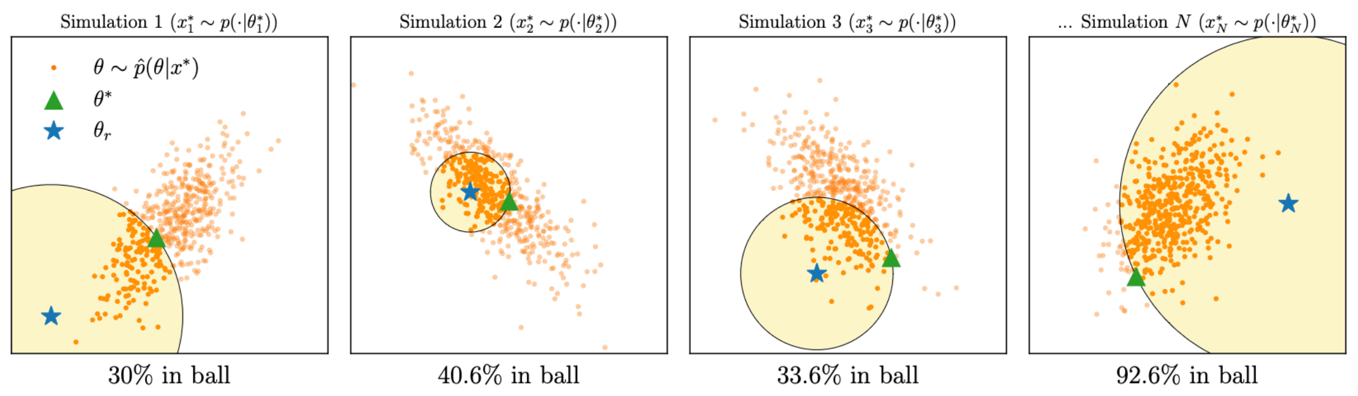
How can we automise the evaluation ?

Need robust measures of distance → Several options thrown around in recent years

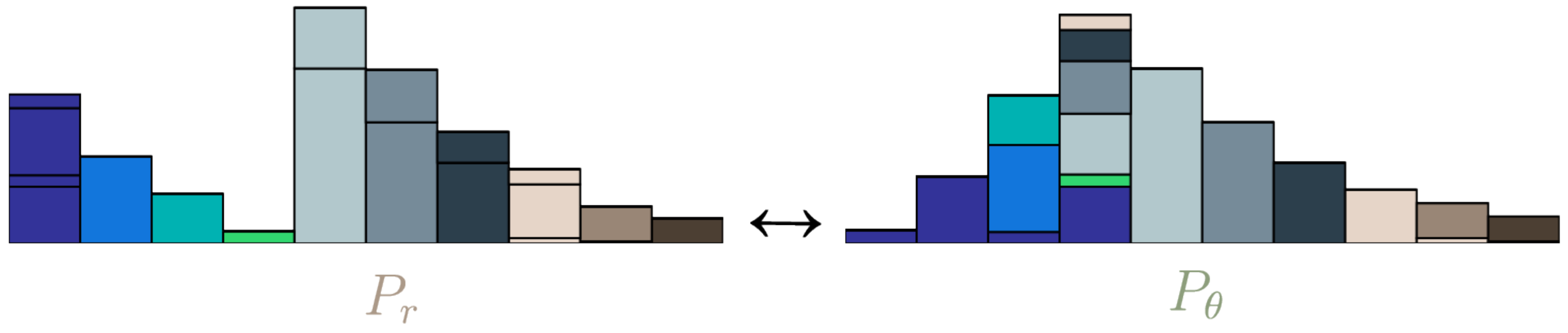
KS, Aderson-Darling, etc

$$\frac{P(x\,|\,Geant)}{P(x\,|\,Gen)}$$

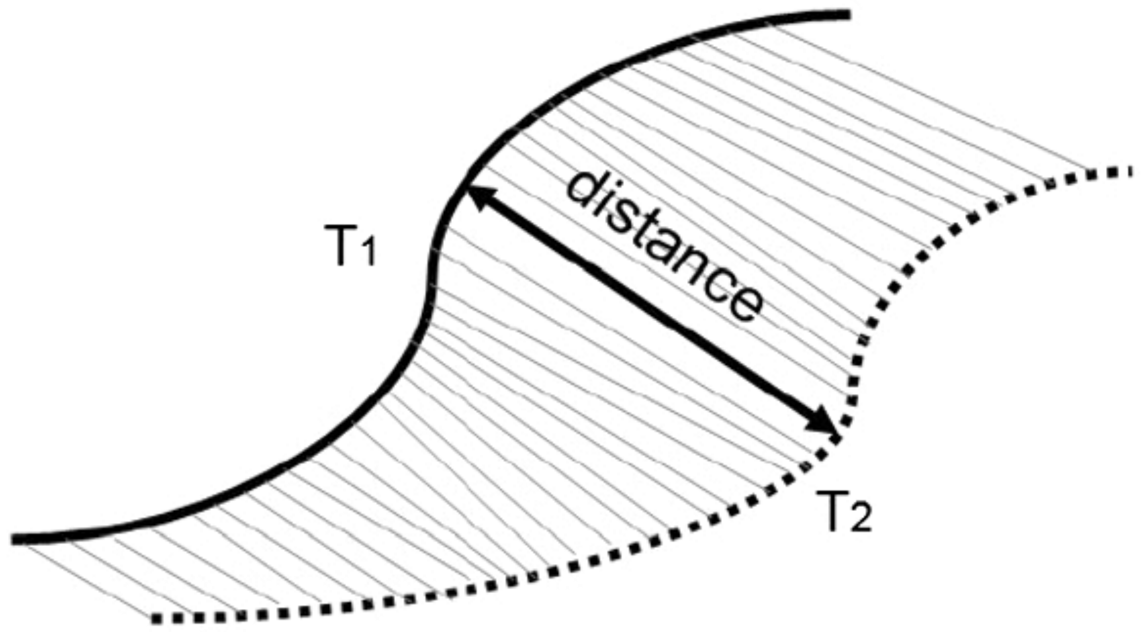
Independent classifier test



TARP



Wasserstein Distance



Fréchet Distance

A large comparison of metrics

[Kansal et al, 2022]

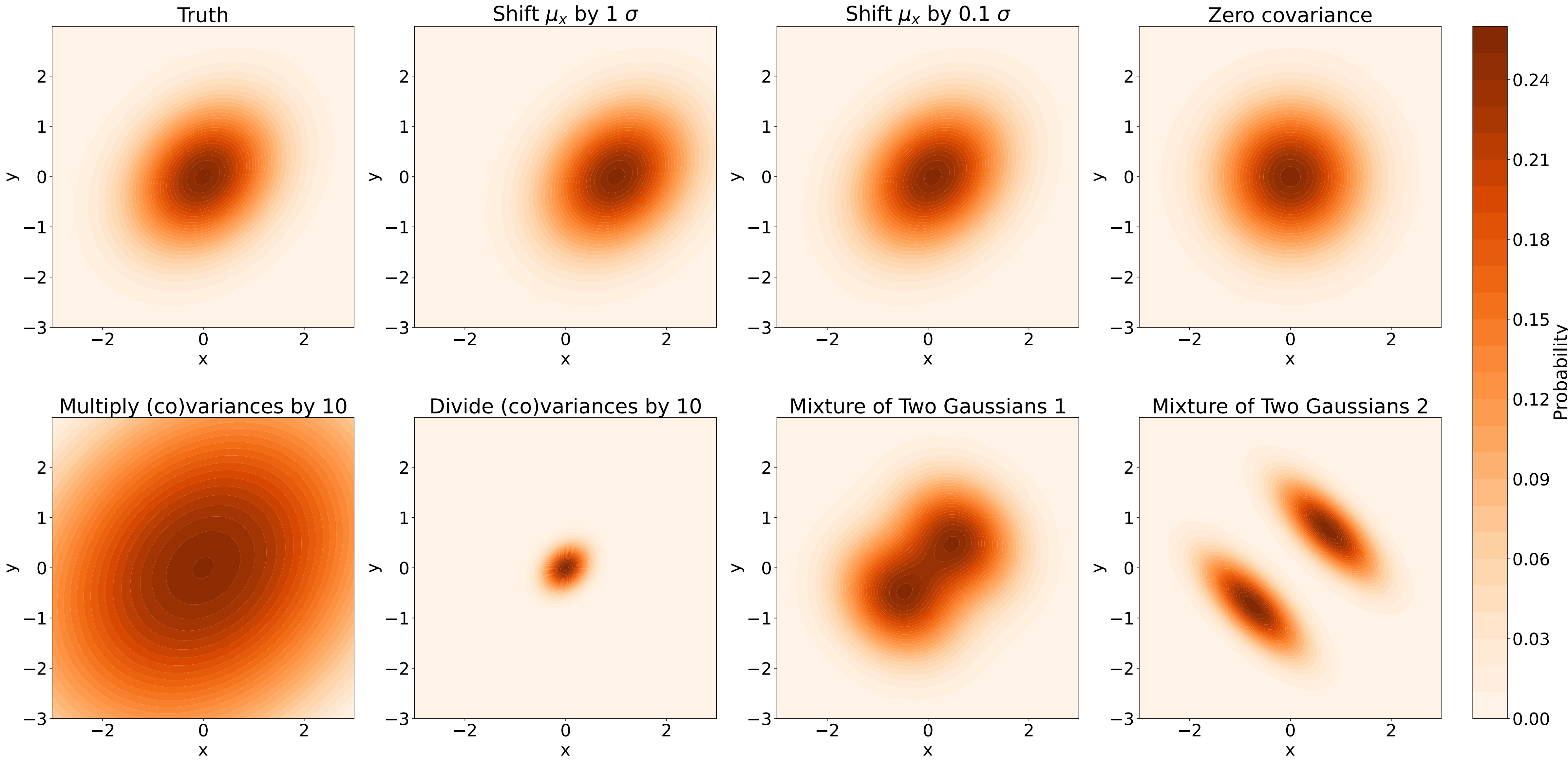
On the Evaluation of Generative Models in High Energy Physics

Raghav Kansal,* Anni Li, and Javier Duarte
University of California San Diego, La Jolla, CA 92093, USA

Nadezda Chernyavskaya, Maurizio Pierini
European Center for Nuclear Research (CERN), 1211 Geneva 23, Switzerland

Breno Orzari, Thiago Tomei
Universidade Estadual Paulista, São Paulo/SP, CEP 01049-010, Brazil

(Dated: November 21, 2022)



Detailed comparison on Gaussian toys where you have full control
Application on jet dataset with hand designed distortions
Suggests 'Fréchet Gaussian Distance'

My personal opinion: This is still an open question!

A large comparison of metrics

[Kansal et al, 2022](#)

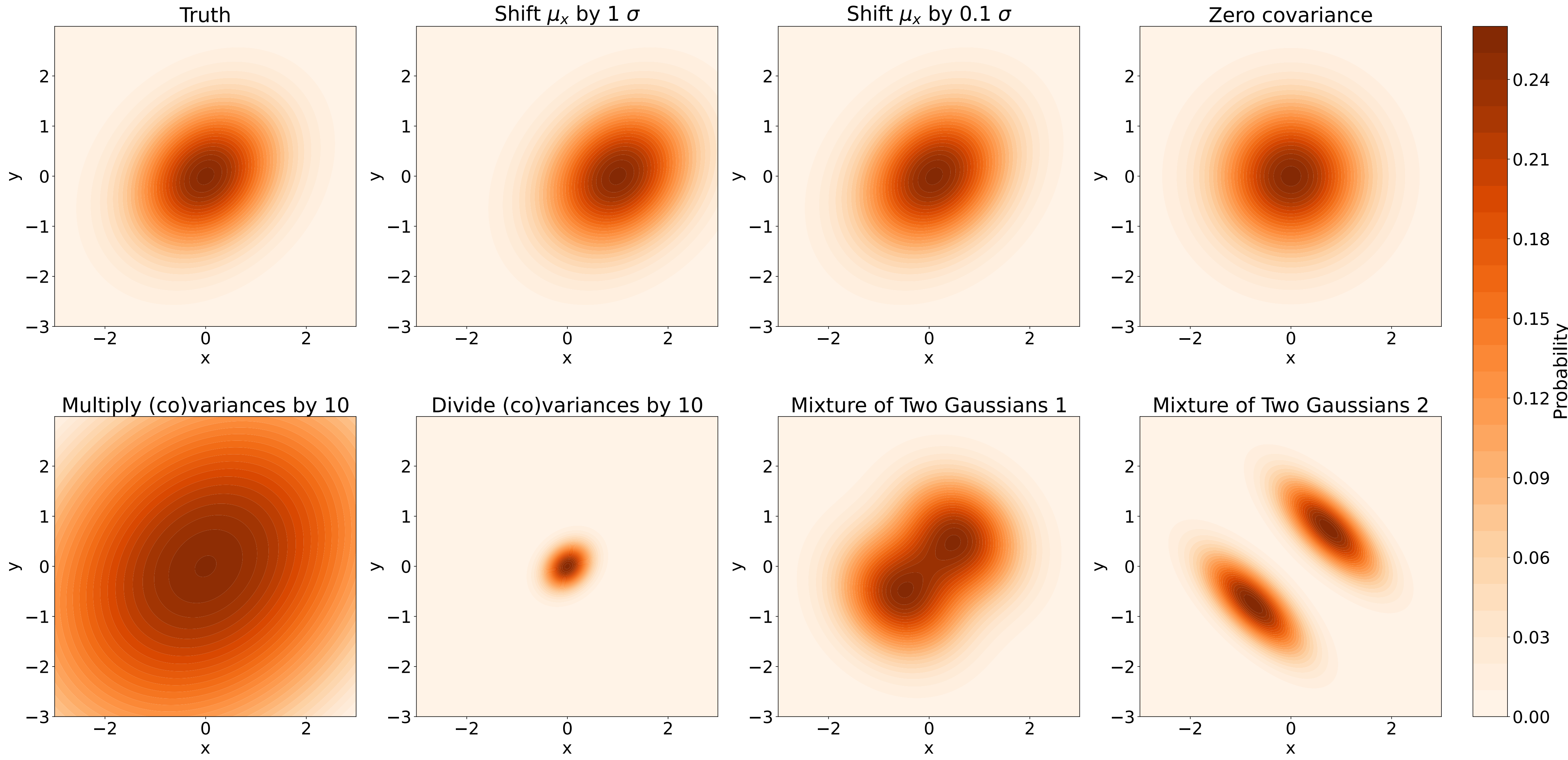
On the Evaluation of Generative Models in High Energy Physics

Raghav Kansal,* Anni Li, and Javier Duarte
University of California San Diego, La Jolla, CA 92093, USA

Nadezda Chernyavskaya, Maurizio Pierini
European Center for Nuclear Research (CERN), 1211 Geneva 23, Switzerland

Breno Orzari, Thiago Tomei
Universidade Estadual Paulista, São Paulo/SP, CEP 01049-010, Brazil

(Dated: November 21, 2022)



Detailed comparison on Gaussian toys where you have full control
Application on jet dataset with hand designed distortions
Suggests 'Fréchet Gaussian Distance'

My personal opinion: This is still an open question!

Think you have a solution? Come chat with me!

Conclusion

- Lots of exciting opportunities in HEP, Cosmo, Astro, elsewhere to use neural SBI
    - Leverages detailed knowledge in simulators
    - High-dimensional inference
    - Speed, amoritised inference
    - Differentiable likelihoods
- More powerful methods → more sensitivity → larger concern for model misspecification
- Interesting statistical questions on propagating uncertainties, coverage tests
- Tools in development to test robustness, interpretability, automatise performance evaluation

The devil is always in the details, come join us in answering these questions!

Conclusion

- Lots of exciting opportunities in HEP, Cosmo, Astro, elsewhere to use neural SBI
    - Leverages detailed knowledge in simulators
    - High-dimensional inference
    - Speed, amoritised inference
    - Differentiable likelihoods
- More powerful methods → more sensitivity → larger concern for model misspecification
- Interesting statistical questions on propagating uncertainties, coverage tests
- Tools in development to test robustness, interpretability, automatise performance evaluation

The devil is always in the details, come join us in answering these questions!

Thank you!

Make the learning task easier by leveraging the simulator

$$r(x_i|\theta, ref) = \frac{p(x_i|\theta_1)}{p(x_i|ref)} = \frac{s(x_i)}{1 - s(x_i)}$$

Intractable

$$p(x|\theta) = \int dz \;\; p(x|z_h)\, p(z_h|z_p)\, p(z_p|\theta)$$

This part is accessible

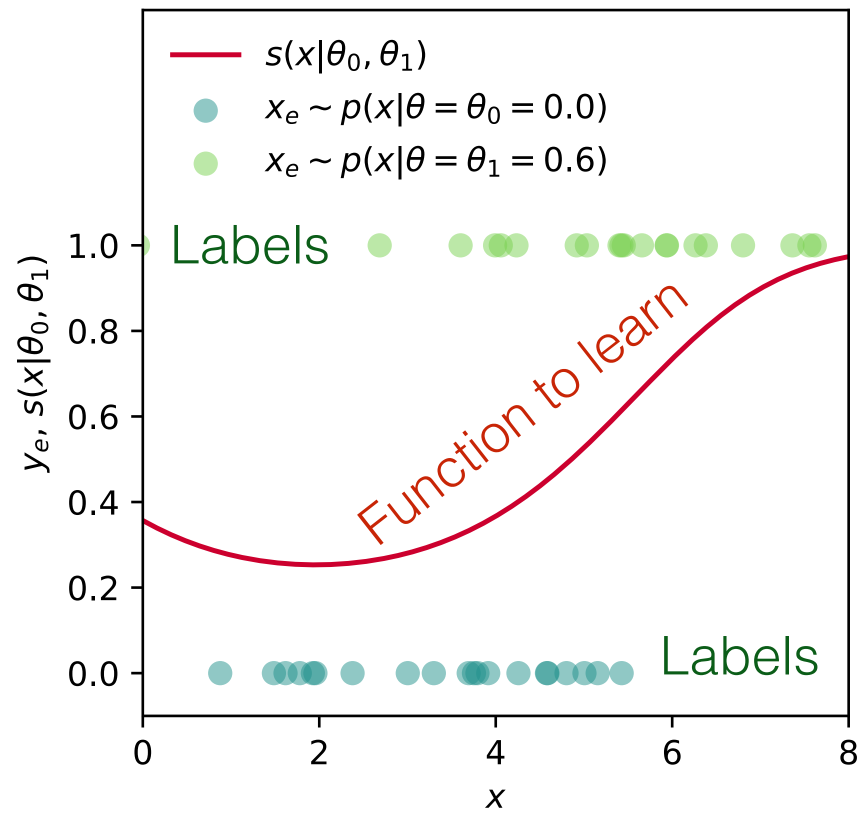
MADGRAPH

aMC@NLO

Madminer by Brehmer, et al, arXiv:1907.10621

Make the learning task easier by leveraging the simulator

$$r(x_i \,|\, \theta, ref) = \frac{p(x_i \,|\, \theta_1)}{p(x_i \,|\, ref)} = \frac{s(x_i)}{1 - s(x_i)}$$



Intractable

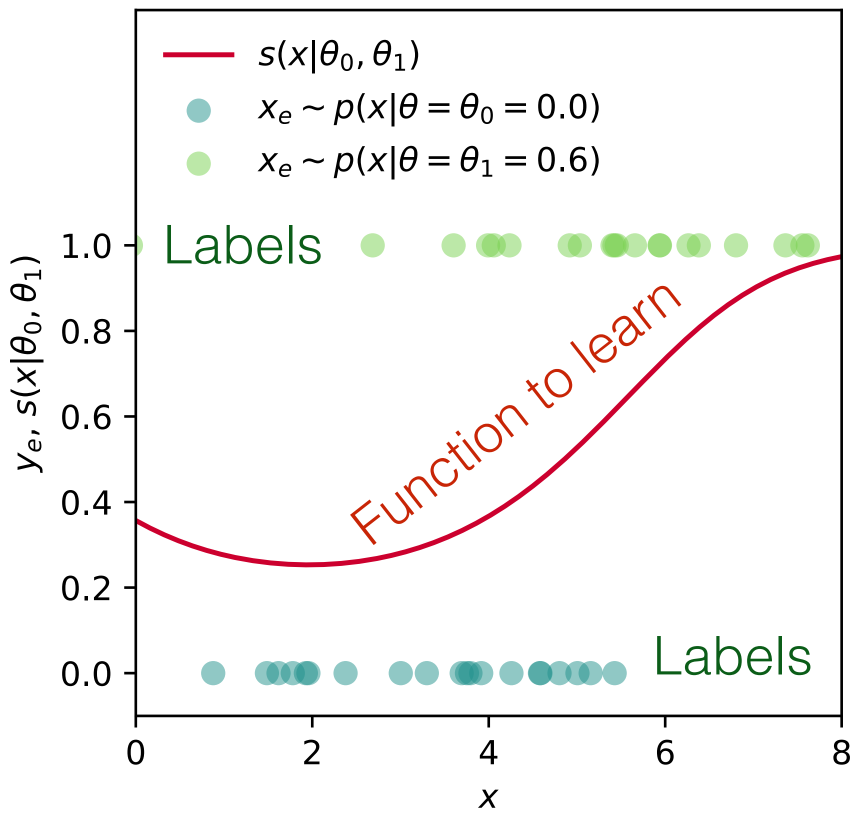
$$p(x \,|\, \theta) = \int dz \;\; p(x \,|\, z_h) \; p(z_h \,|\, z_p) \; p(z_p \,|\, \theta)$$

This part is accessible

Madminer by Brehmer, et al, arXiv:1907.10621

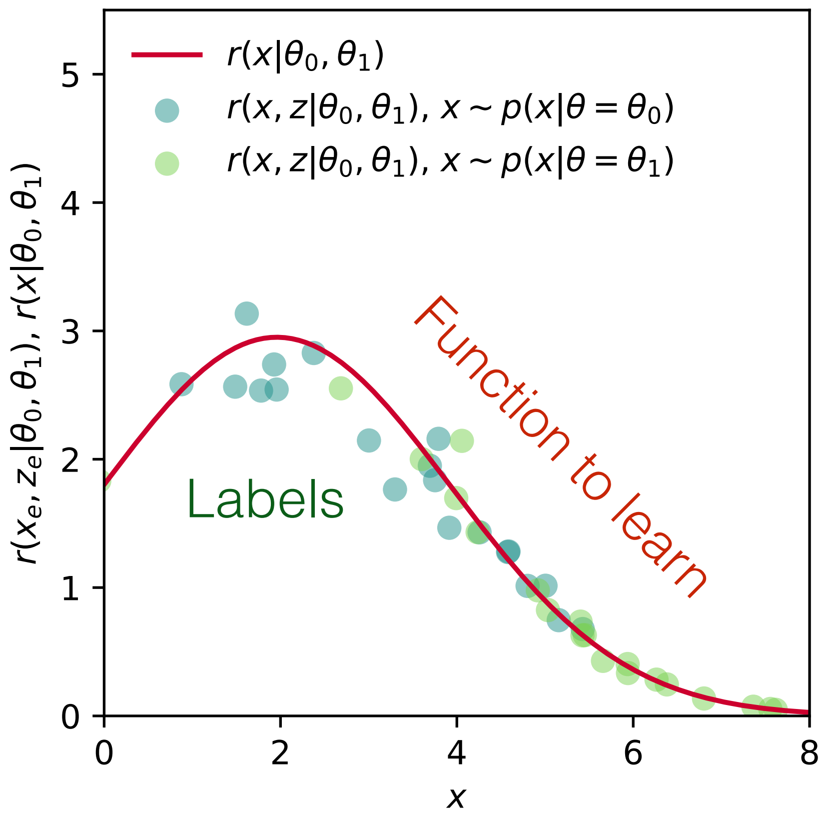
Make the learning task easier by leveraging the simulator

$$r(x_i | \theta, ref) = \frac{p(x_i | \theta_1)}{p(x_i | ref)} = \frac{s(x_i)}{1 - s(x_i)}$$



Change the learning task

Easier target labels:



Intractable

$$p(x | \theta) = \int dz \;\; p(x | z_h) \; p(z_h | z_p) \; p(z_p | \theta)$$

This part is accessible

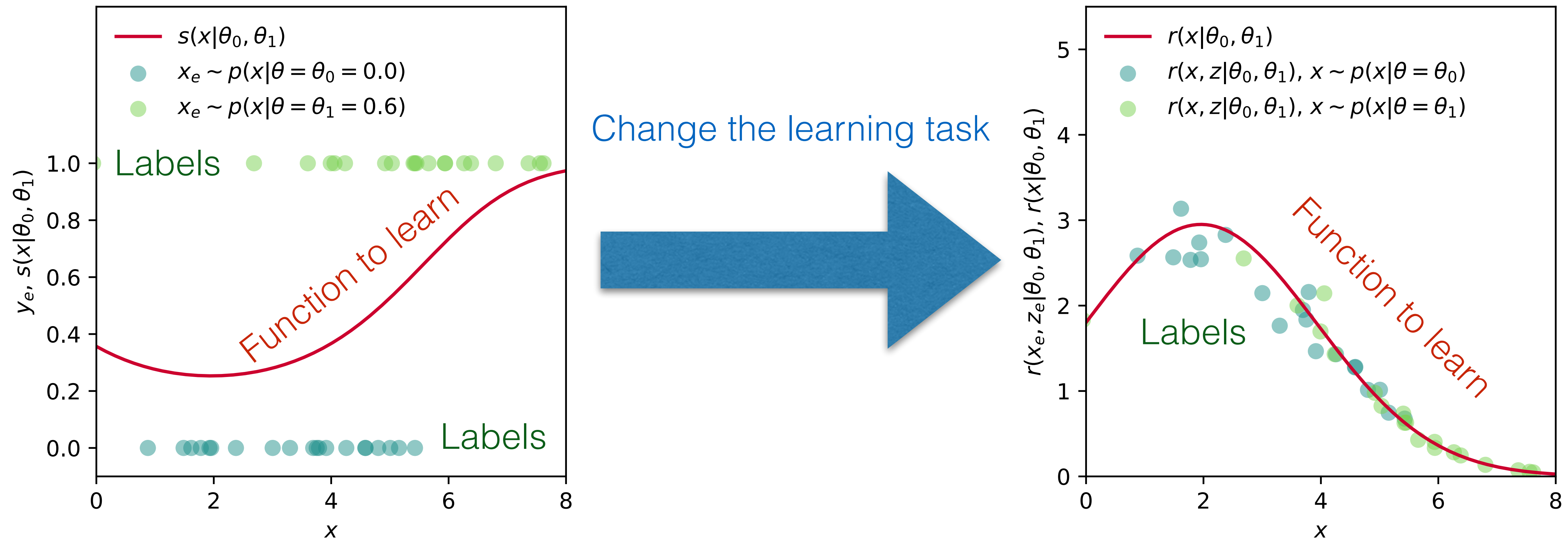
MADGRAPH 5 aMC@NLO

Madminer by Brehmer, et al, arXiv:1907.10621

Make the learning task easier by leveraging the simulator

$$r(x_i|\theta, ref) = \frac{p(x_i|\theta_1)}{p(x_i|ref)} = \frac{s(x_i)}{1-s(x_i)}$$

Easier target labels:

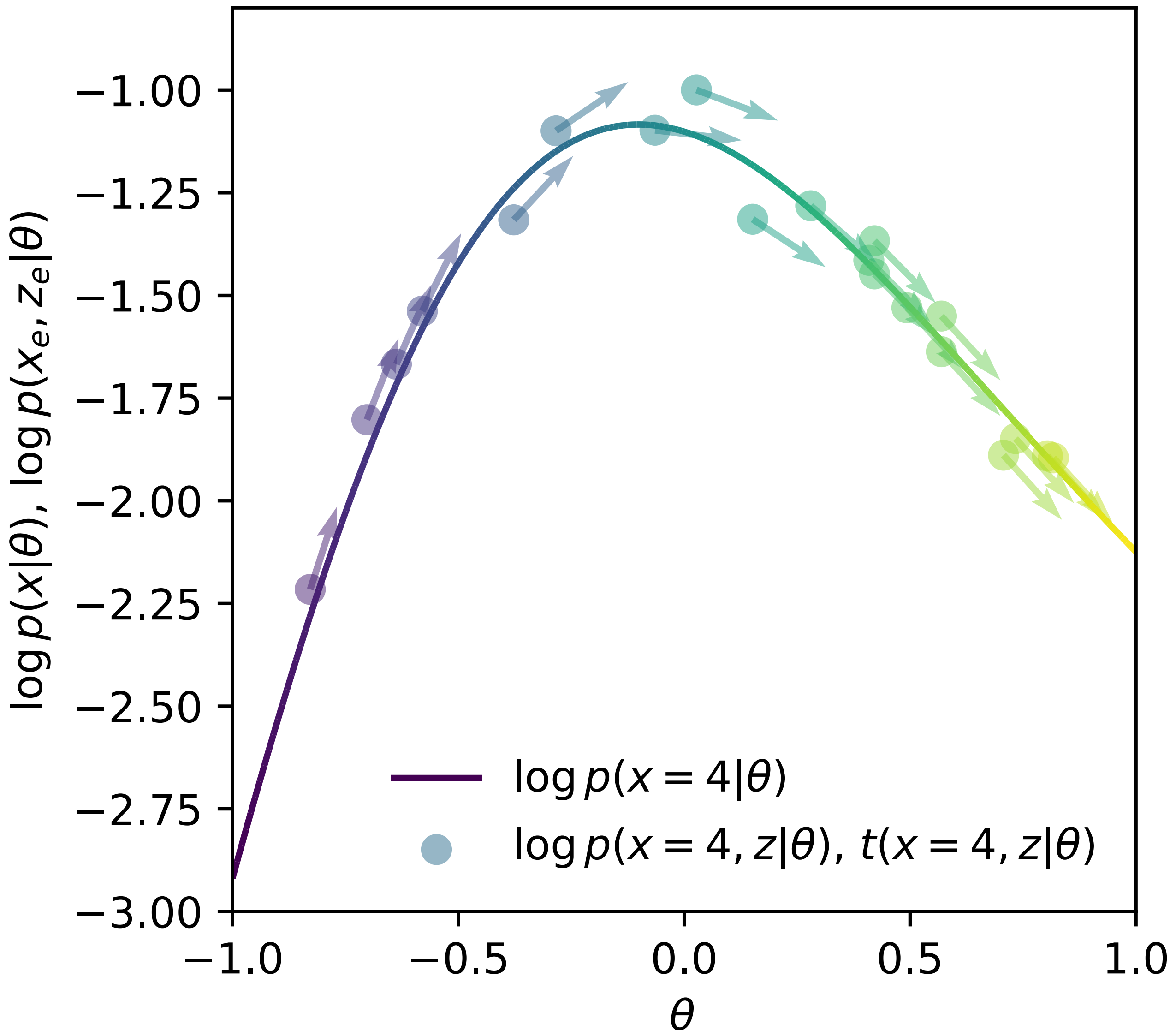


Intractable

$$p(x|\theta) = \int dz \;\; p(x|z_h)\; p(z_h|z_p)\; p(z_p|\theta)$$

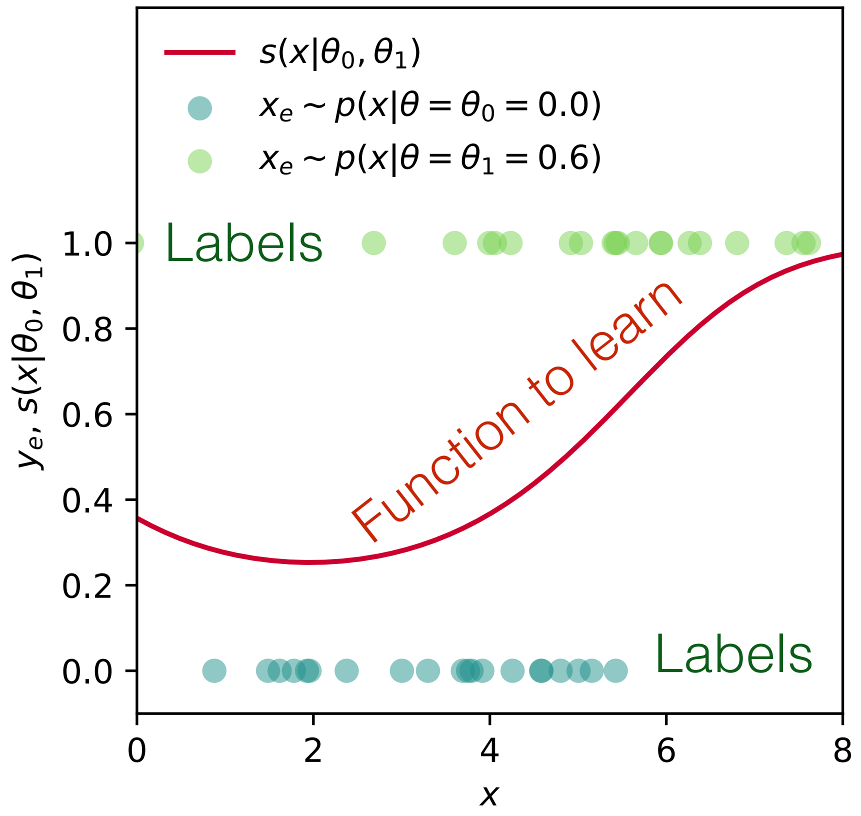
This part is accessible

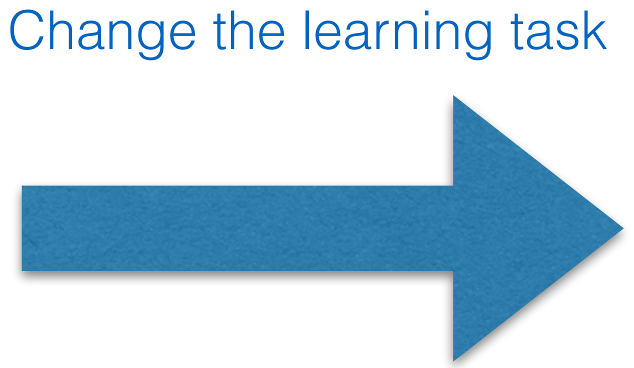
Gradients:

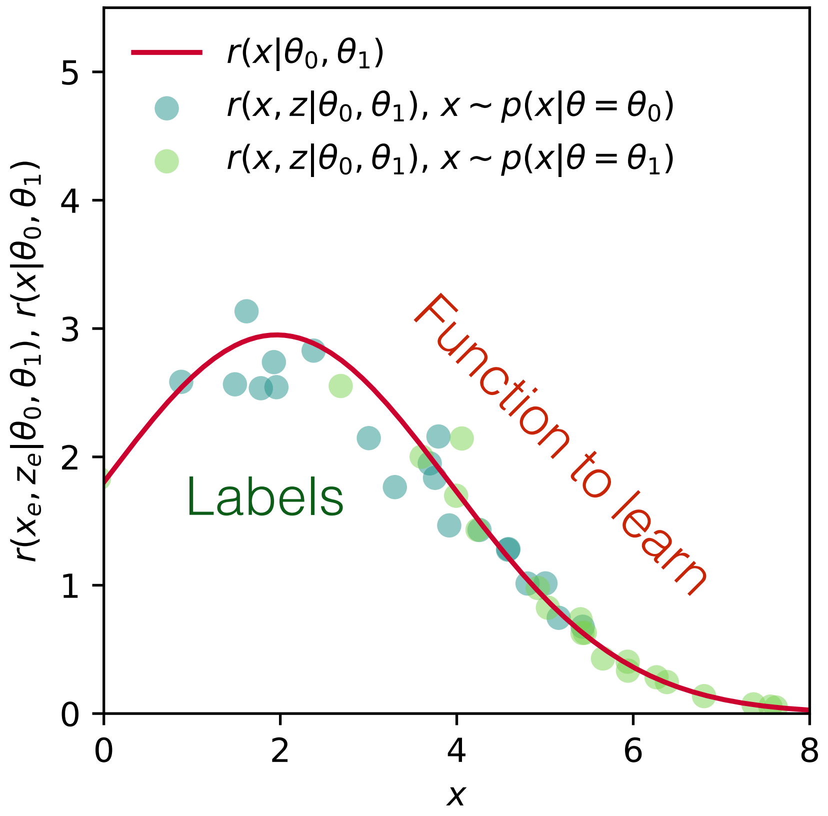


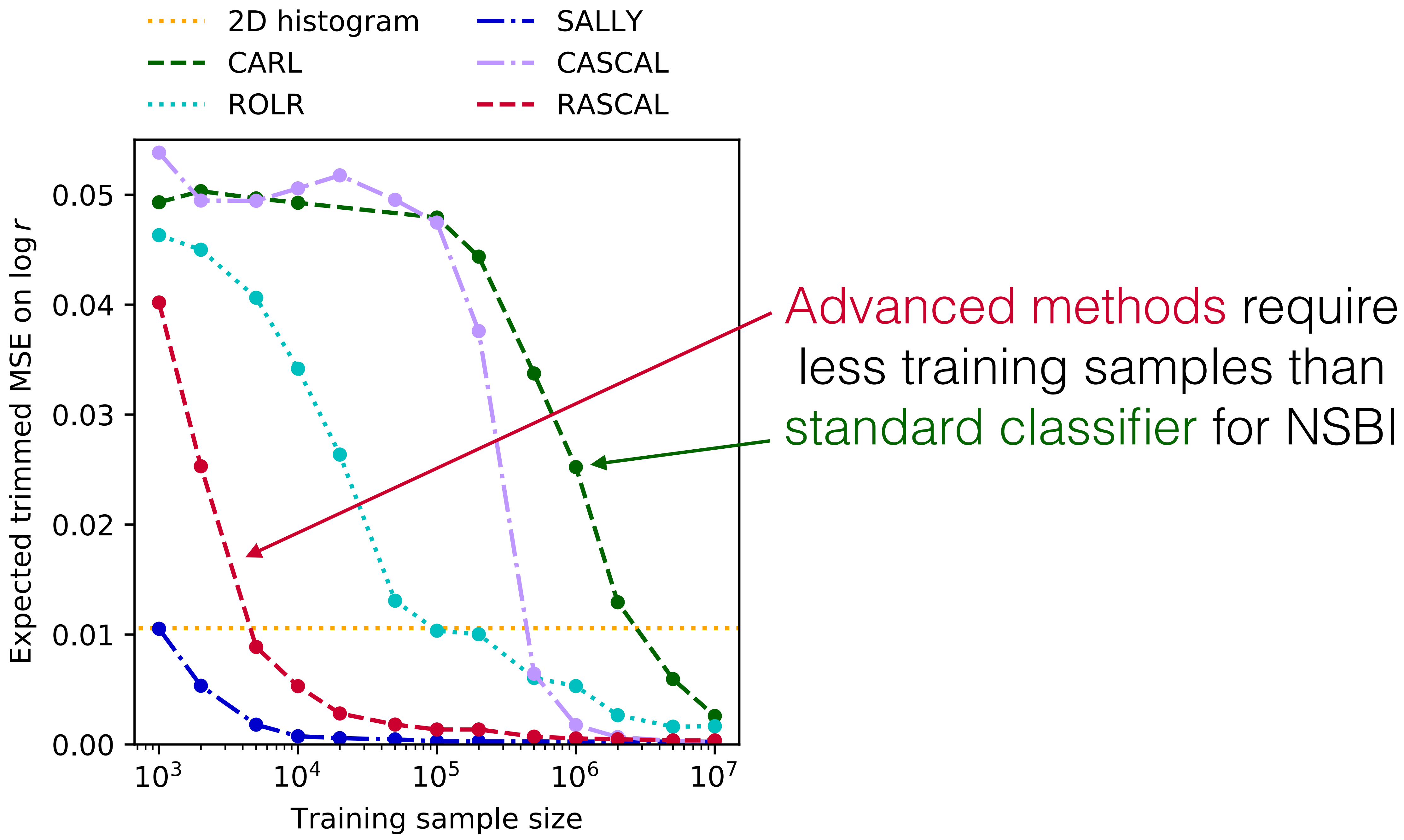
Madminer by Brehmer, et al, arXiv:1907.10621

What is it worth?



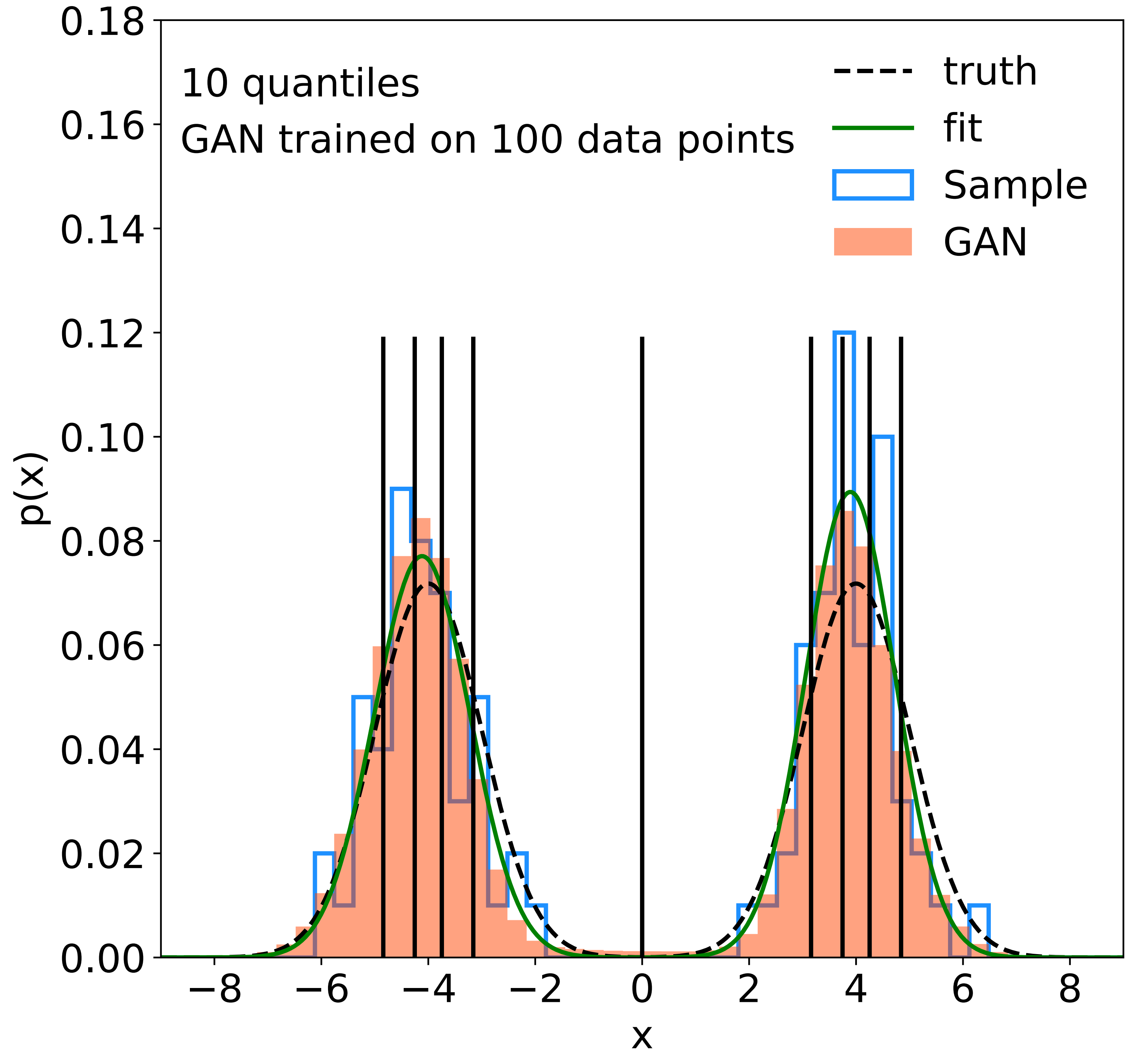


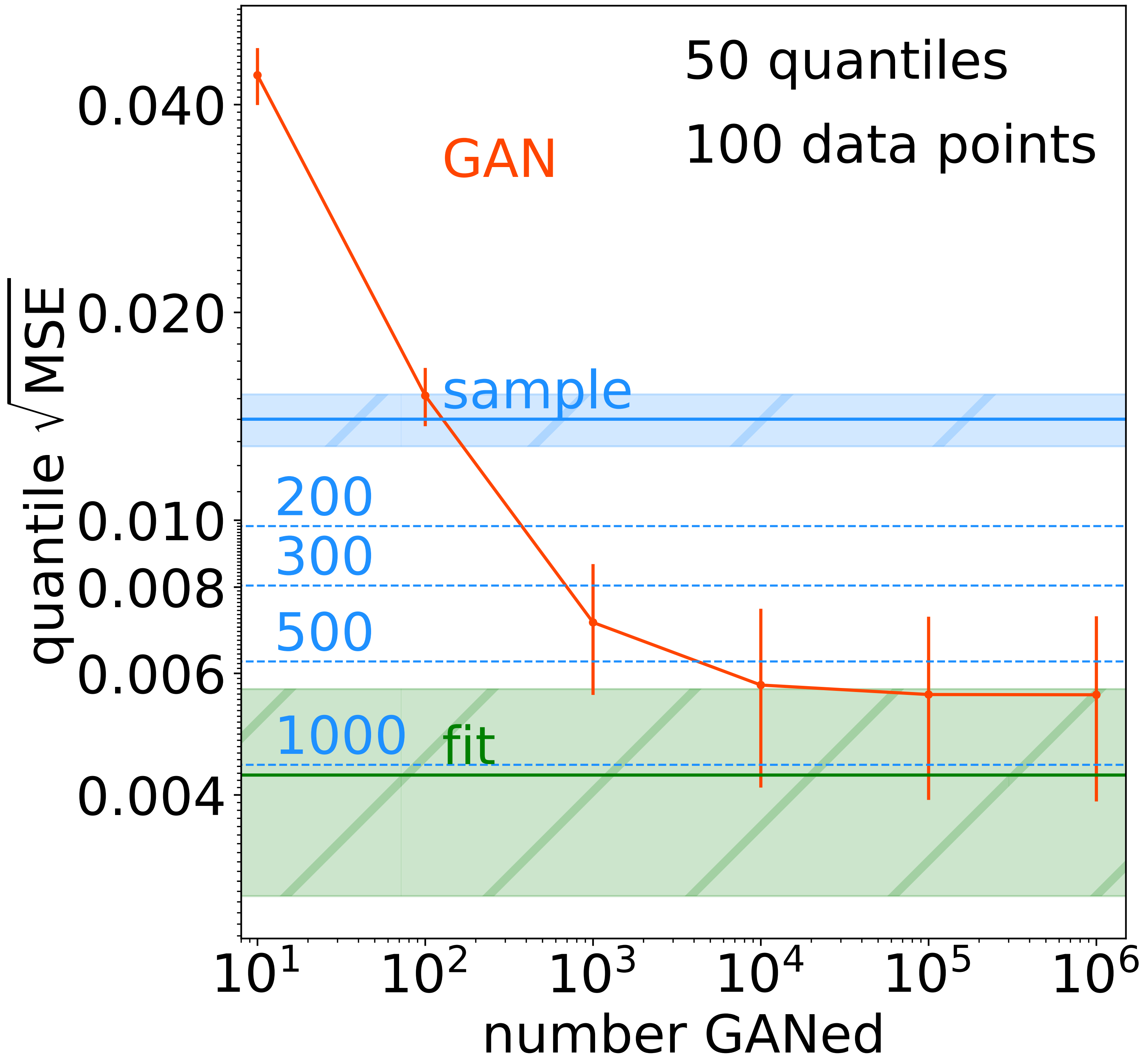




- Requires calling the simulator N times per event
- Instead, we could simulate N times the samples
- Intuition says former is more compute efficient, might be analysis dependent

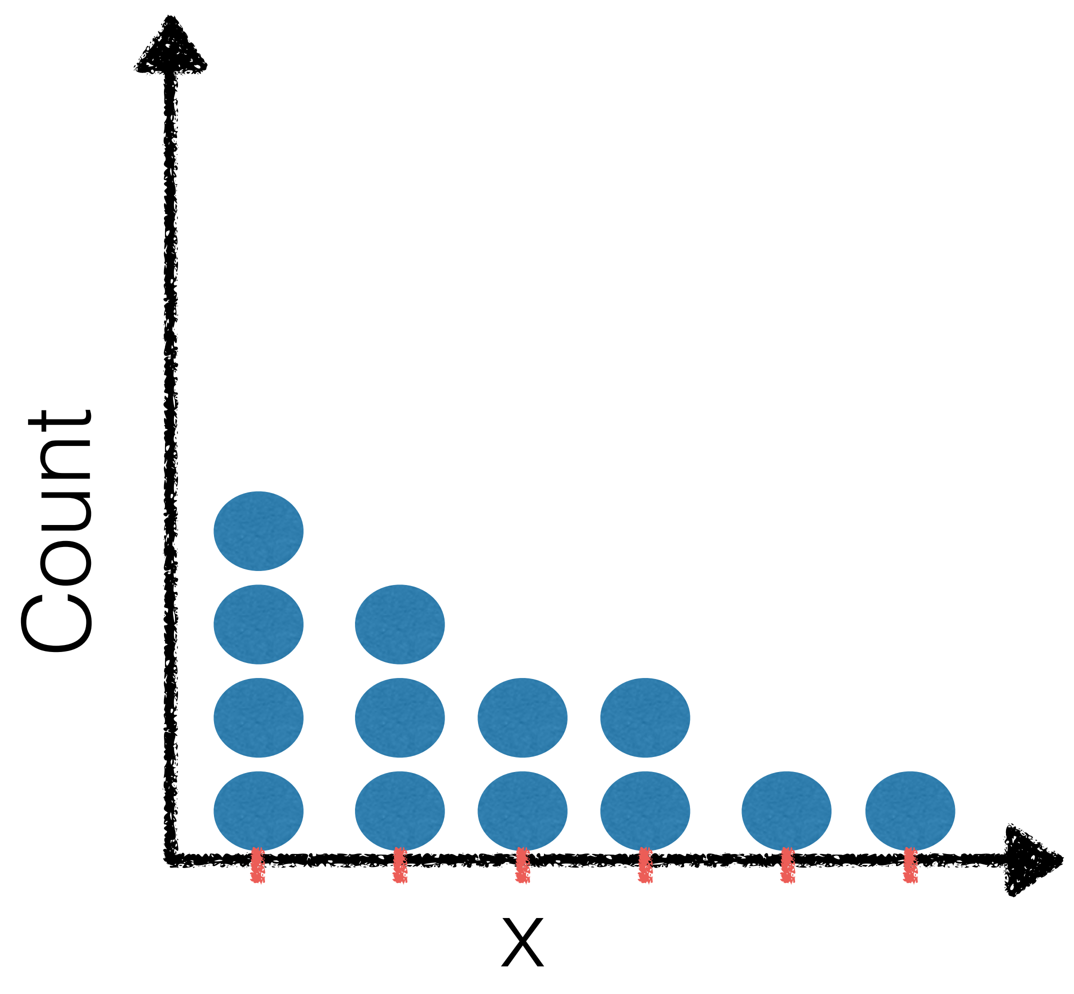
Amplify statistics with generative models



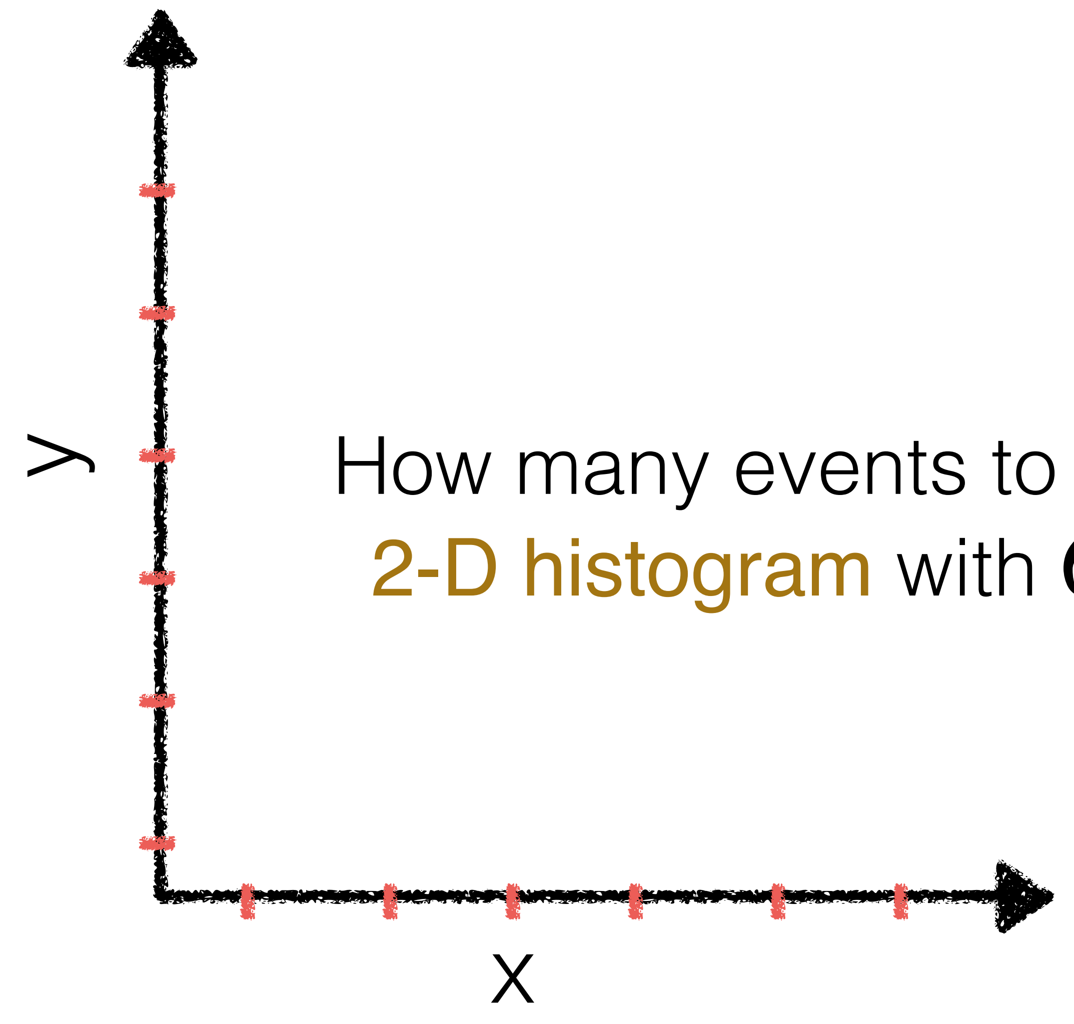


Generative models appear to produce more meaningful samples than training dataset
Smooths over the statistical fluctuations

Density estimation in higher dimensions, the curse (of dimensionality)



1-D histogram with 6 bins: few events enough to populate it



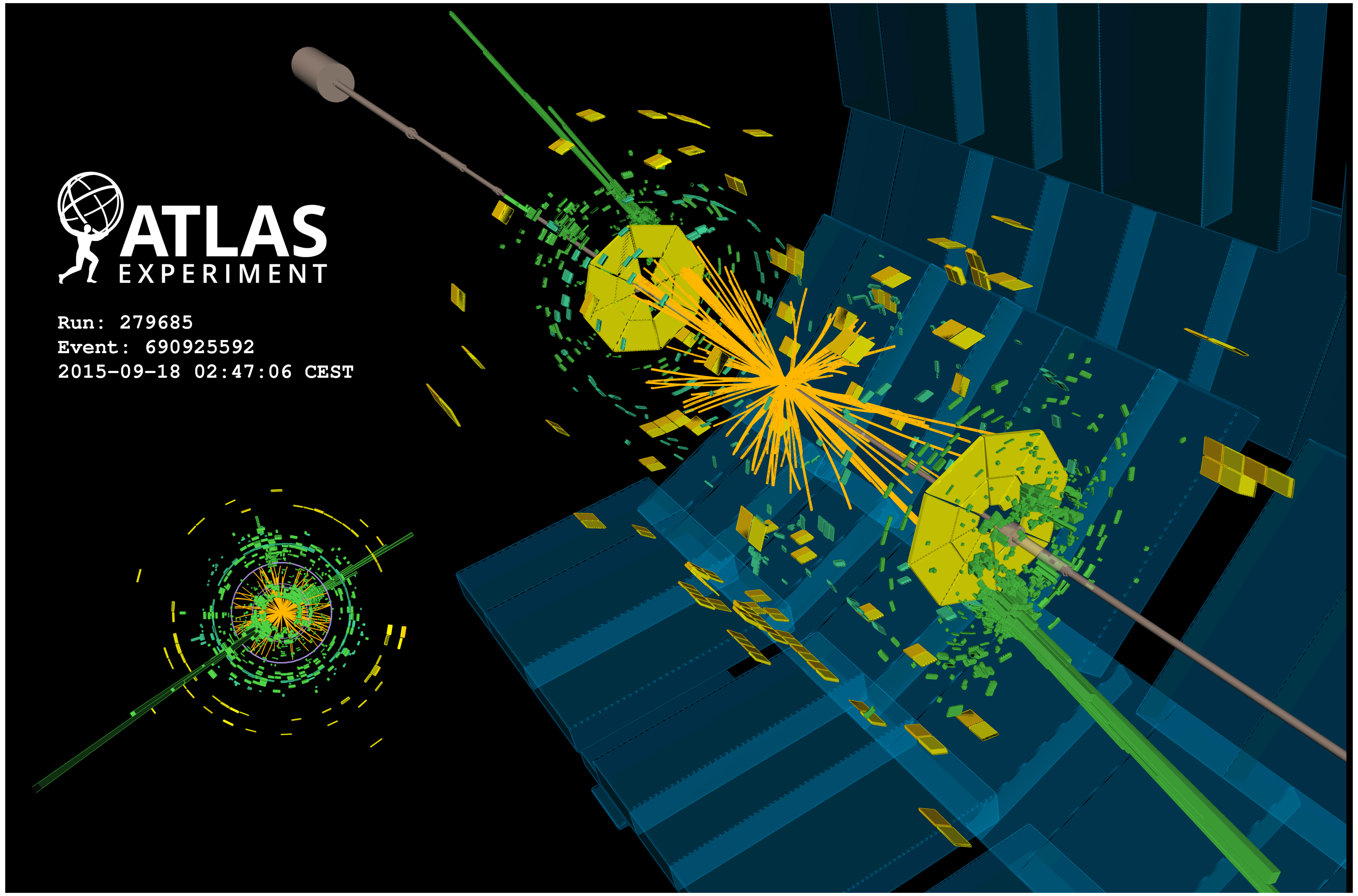
How many events to populate 2-D histogram with 6^2 bins ?

How many events for 50-D histogram with 6^{50} bins ?

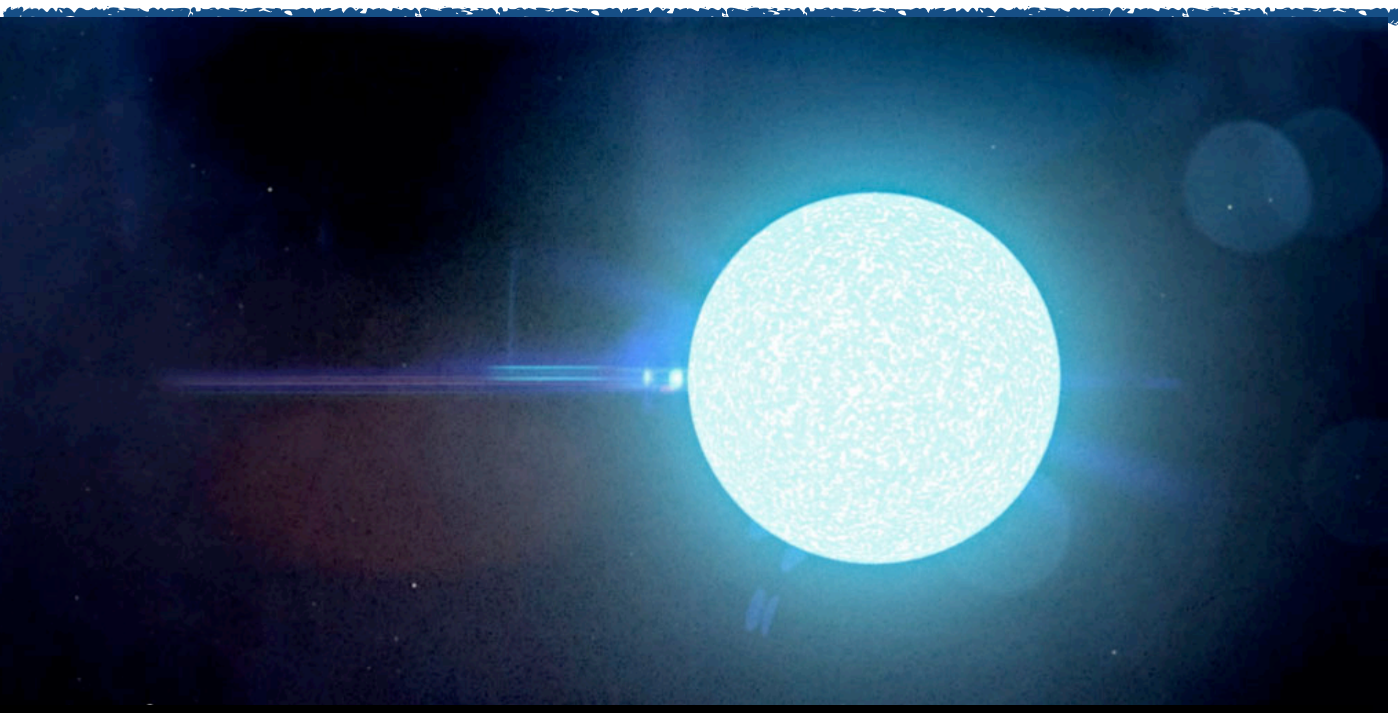
High-dimensional data

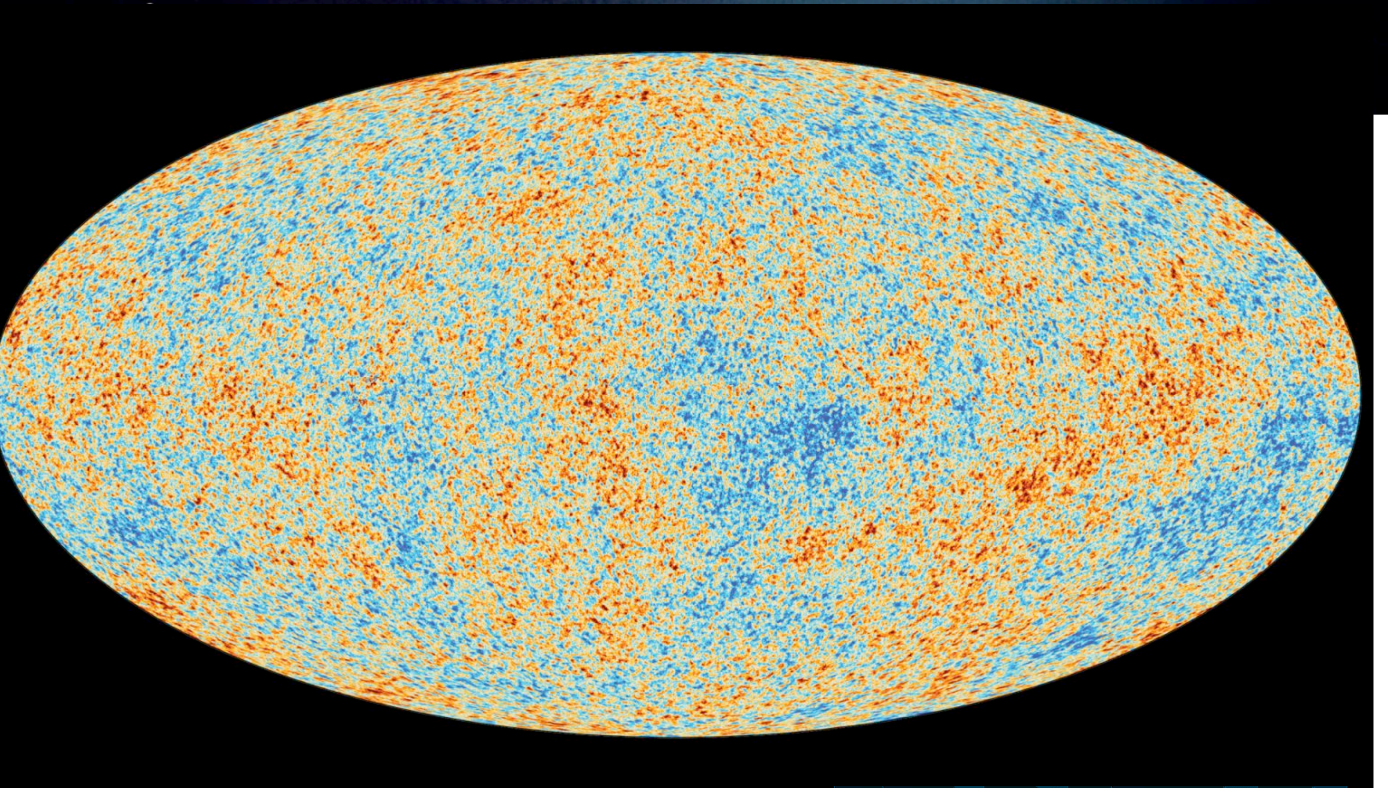
- Detector has O(100 million) sensors
- Can't build 100M dimensional histogram

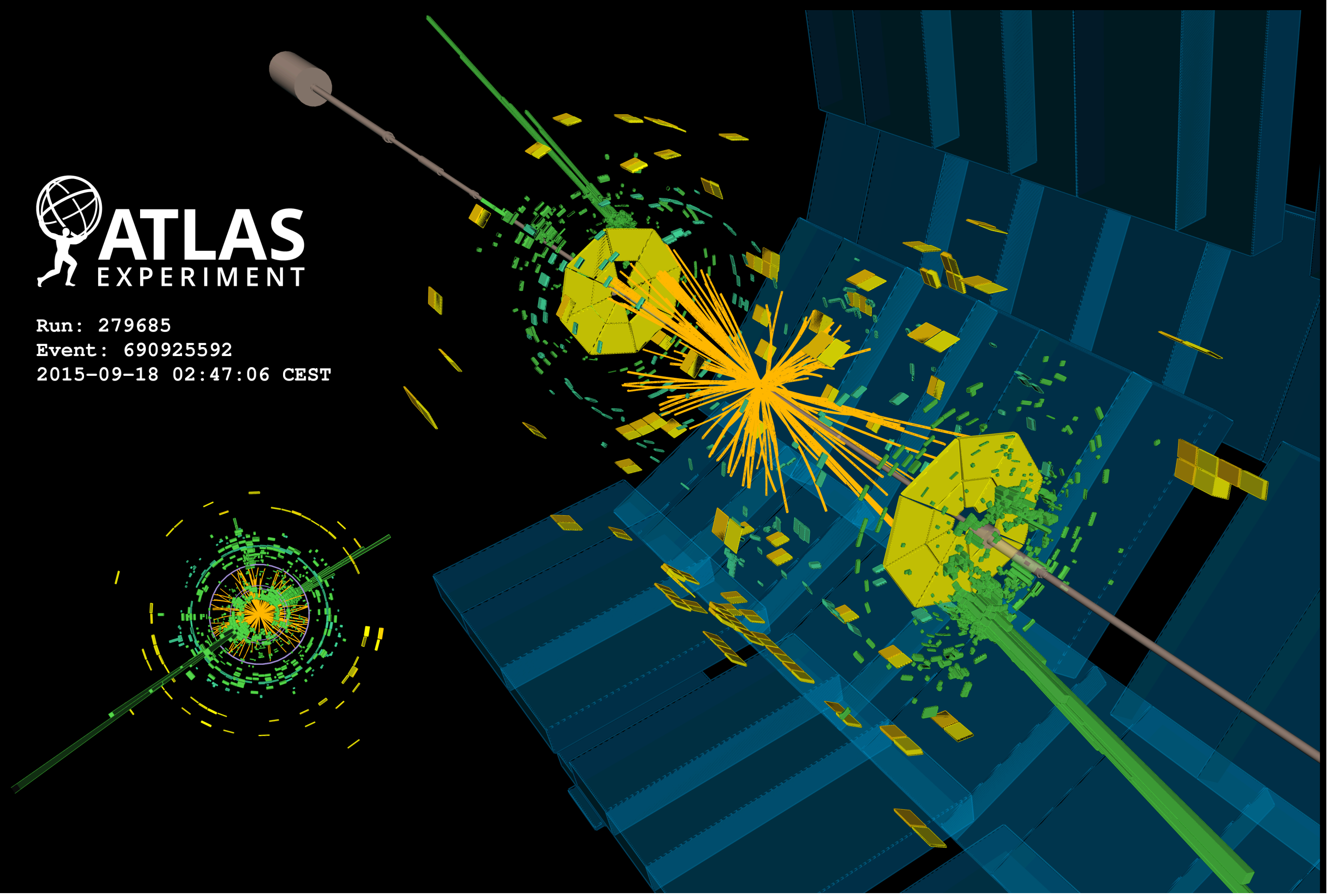
- Reconstruction pipeline, event selection
- Design sensitive one-dimensional observable



Core idea: Neural networks for inference







High-dim data $\longrightarrow$ Neural Networks $\longrightarrow$

Likelihood $p(x_i \mid theory)$

or

Likelihood Ratio $\dfrac{p(x_i \mid theory)}{p(x_i \mid ref)}$

or

Posterior $p(theory \mid x)$

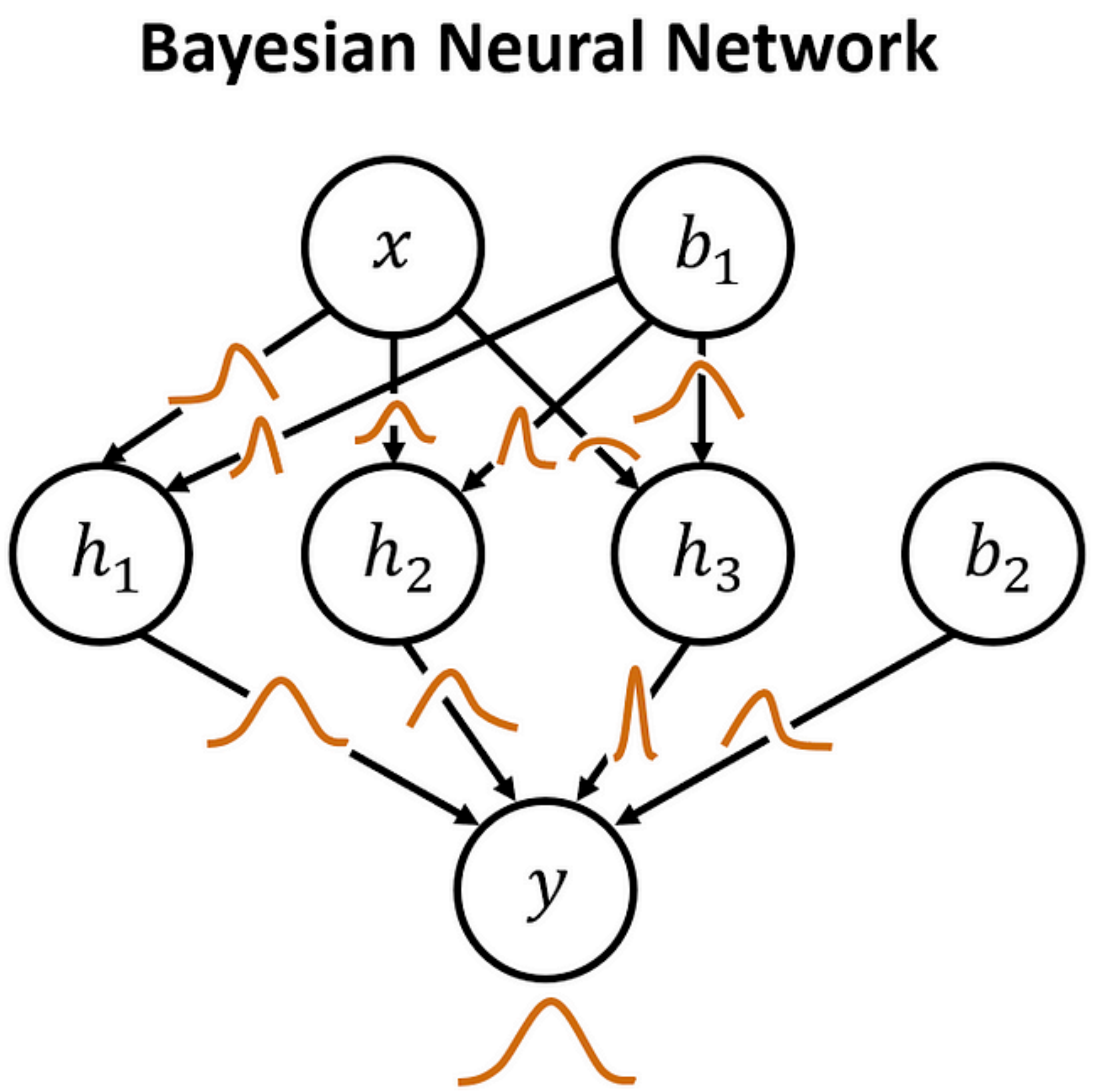
- Fully leverage detailed physics knowledge stored in simulators
- Perform high-dimensional inference

Bayesian Networks

- Each weight replaced by a distribution of weights
    - Eg. Sampled from learnt {mean, std}

- The distribution in NN prediction for each event gives you an uncertainty estimate

- Open question: How to interpret this uncertainty? What is the coverage?
    - Calibrate the uncertainties arXiv:2408.00838: Bringer et al (incl. Diefenbacher)
    - … more work needed here before if they are to become standard tools in frequentist frameworks

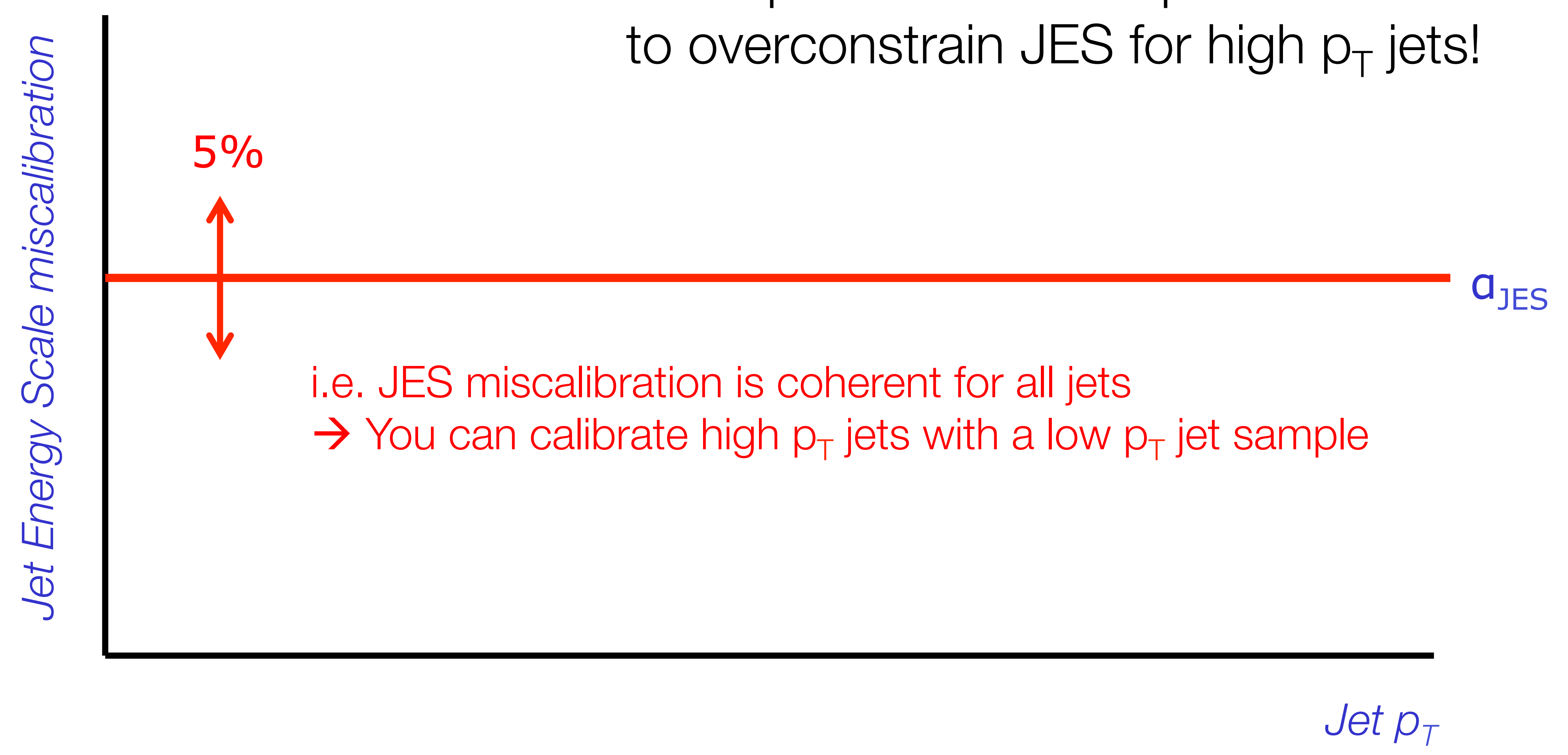


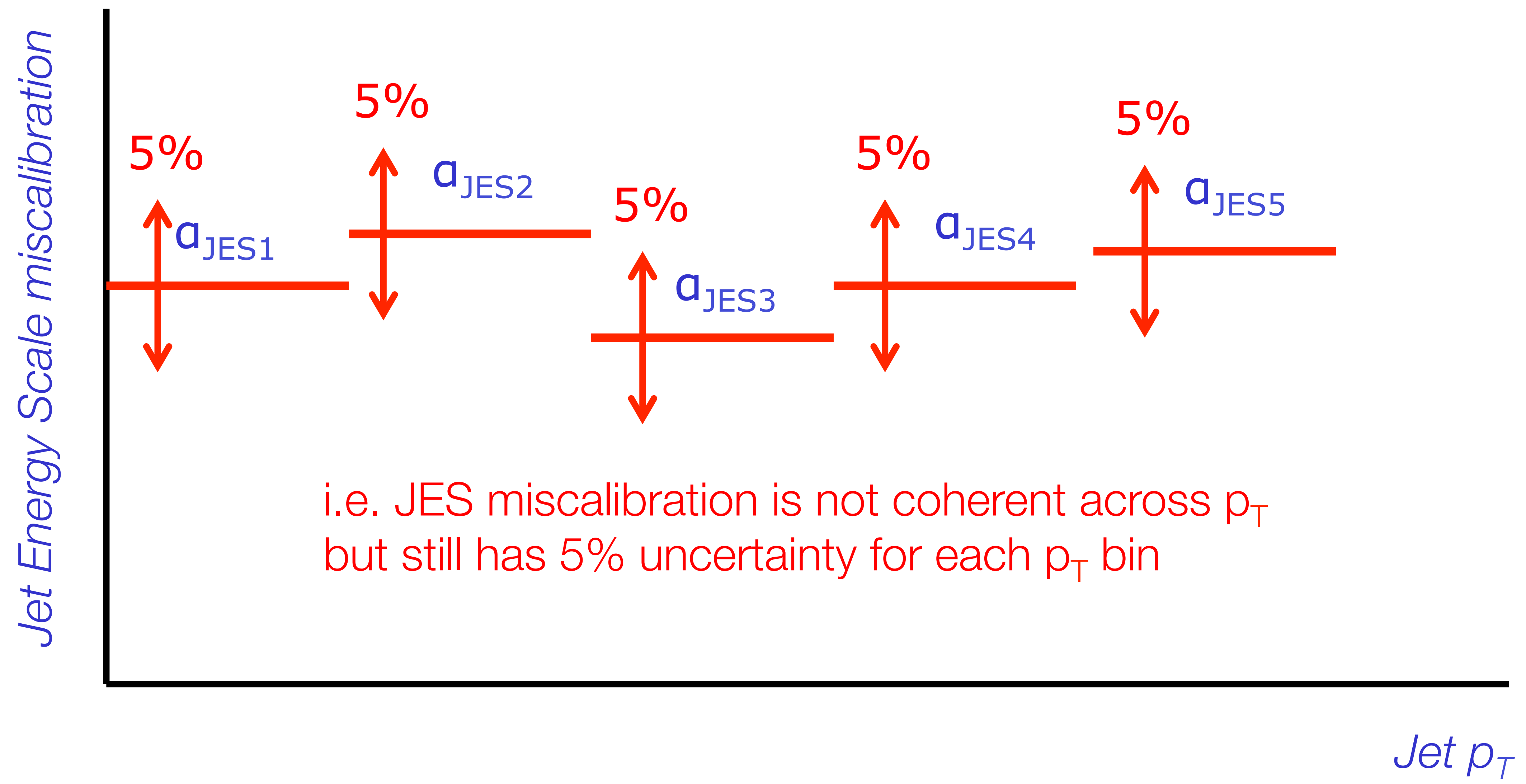
Overconstraining NP

From W. Verkerke:

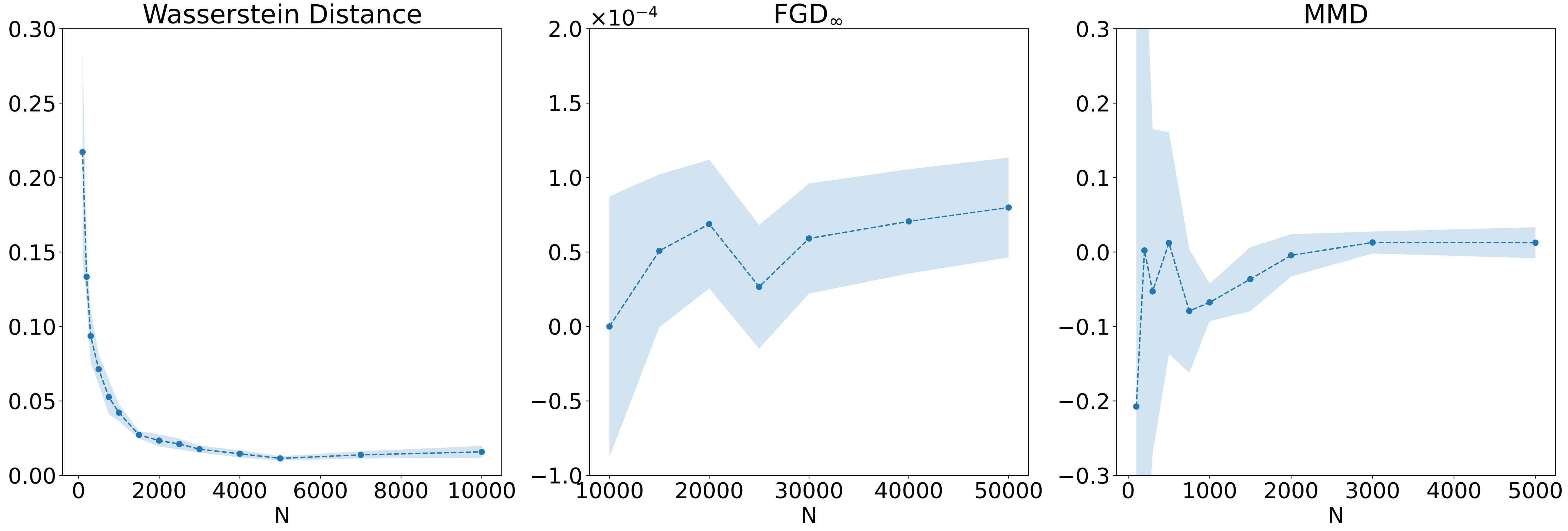
Our modelling of NPs might be over-simplified

- If you assume one NP – chances are that your physics Likelihood will exploit this oversimplified JES model to overconstrain JES for high p_T jets!





Gaussian Study



- FGD_∞, MMD unbiased
- W too expensive for large N

Metric	Truth	Shift μ_x by 1σ	Shift μ_x by 0.1σ	Zero covariance	Multiply (co)variances by 10	Divide (co)variances by 10	Mixture of Two Gaussians 1	Mixture of Two Gaussians 2
Wasserstein	0.016 ± 0.004	1.14 ± 0.02	0.043 ± 0.008	0.077 ± 0.006	9.8 ± 0.1	0.97 ± 0.01	$\mathbf{0.036 \pm 0.003}$	$\mathbf{0.191 \pm 0.005}$
FGD$_\infty$ $\times 10^3$	0.08 ± 0.03	$\mathbf{1011 \pm 1}$	$\mathbf{11.0 \pm 0.1}$	$\mathbf{32.3 \pm 0.2}$	9400 ± 8	$\mathbf{935.1 \pm 0.7}$	0.07 ± 0.03	0.03 ± 0.03
MMD	0.01 ± 0.02	16.4 ± 0.9	0.07 ± 0.04	0.40 ± 0.08	$\mathbf{19k \pm 1k}$	4.3 ± 0.1	0.06 ± 0.02	0.35 ± 0.03
Precision	0.972 ± 0.005	0.91 ± 0.01	0.976 ± 0.004	0.969 ± 0.006	0.34 ± 0.01	1.0 ± 0.0	0.975 ± 0.003	0.9976 ± 0.0007
Recall	0.997 ± 0.001	0.992 ± 0.003	0.997 ± 0.001	0.9976 ± 0.0006	0.998 ± 0.001	0.58 ± 0.02	0.996 ± 0.001	0.9970 ± 0.0009
Density	3.23 ± 0.06	2.48 ± 0.08	3.19 ± 0.07	3.1 ± 0.1	0.60 ± 0.02	5.7 ± 0.3	2.99 ± 0.09	0.989 ± 0.009
Coverage	0.876 ± 0.002	0.780 ± 0.006	0.872 ± 0.005	0.872 ± 0.004	0.60 ± 0.01	0.406 ± 0.008	0.871 ± 0.002	0.956 ± 0.006

FGD_∞ most promising (with caveats)

Physics study with jets

Kansal et al, 2022

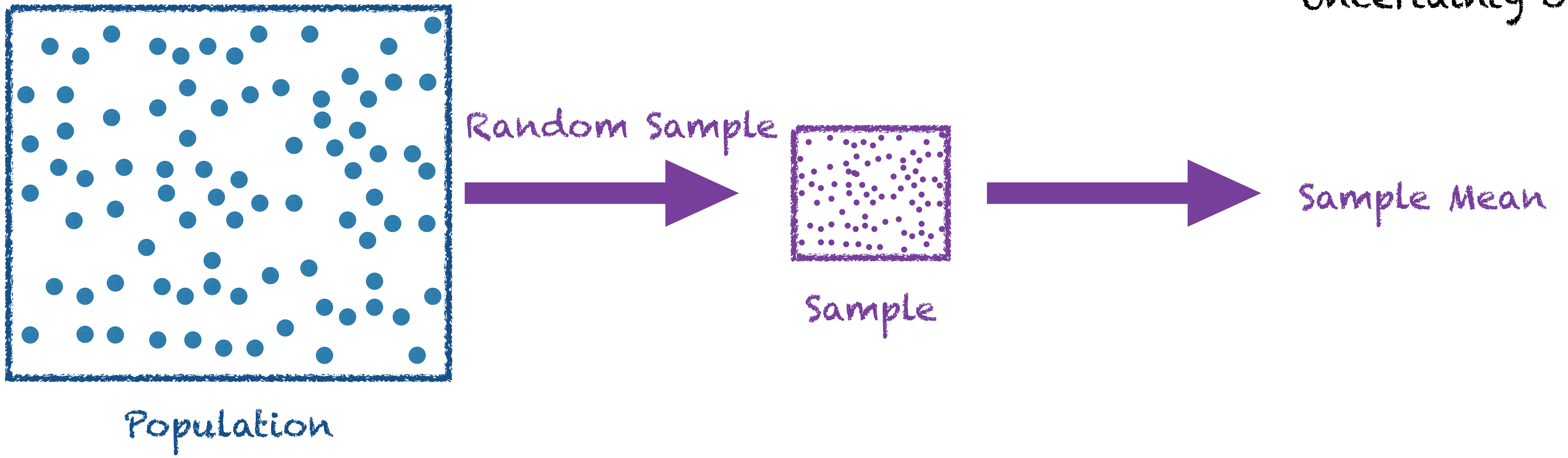
Metric	Truth	Smeared	Shifted	Removing tail	Particle features smeared	Particle η^{rel} smeared	Particle $p_{\mathrm{T}}^{\mathrm{rel}}$ smeared	Particle $p_{\mathrm{T}}^{\mathrm{rel}}$ shifted
$W_1^M \times 10^3$	0.28 ± 0.05	2.1 ± 0.2	6.0 ± 0.3	0.6 ± 0.2	1.7 ± 0.2	0.9 ± 0.3	0.5 ± 0.2	5.8 ± 0.2
Wasserstein EFP	0.02 ± 0.01	0.09 ± 0.05	0.10 ± 0.02	0.016 ± 0.007	0.19 ± 0.08	0.03 ± 0.01	0.03 ± 0.02	0.06 ± 0.02
FGD_∞ EFP $\times 10^3$	0.01 ± 0.02	$\mathbf{21.5 \pm 0.3}$	$\mathbf{26.8 \pm 0.3}$	$\mathbf{2.31 \pm 0.07}$	23.4 ± 0.3	$\mathbf{3.59 \pm 0.09}$	2.29 ± 0.05	28.9 ± 0.2
MMD EFP $\times 10^3$	-0.006 ± 0.005	0.17 ± 0.06	0.9 ± 0.1	0.03 ± 0.02	0.35 ± 0.09	0.08 ± 0.05	0.01 ± 0.02	1.8 ± 0.1
Precision EFP	0.9 ± 0.1	0.94 ± 0.04	0.978 ± 0.005	0.88 ± 0.08	0.7 ± 0.1	0.94 ± 0.06	0.7 ± 0.1	0.79 ± 0.09
Recall EFP	0.9 ± 0.1	0.88 ± 0.07	0.97 ± 0.01	0.92 ± 0.06	0.83 ± 0.05	0.92 ± 0.07	0.8 ± 0.1	0.8 ± 0.1
Wasserstein PN	1.65 ± 0.06	1.7 ± 0.1	2.4 ± 0.4	1.71 ± 0.08	4.5 ± 0.1	1.79 ± 0.05	4.0 ± 0.4	7.6 ± 0.2
FGD_∞ PN $\times 10^3$	0.8 ± 0.7	40 ± 2	193 ± 9	5.0 ± 0.9	$\mathbf{1250 \pm 10}$	20 ± 1	$\mathbf{1230 \pm 10}$	$\mathbf{3640 \pm 10}$
MMD PN $\times 10^3$	-2 ± 2	4 ± 8	80 ± 10	-1 ± 4	500 ± 100	3 ± 2	560 ± 60	1100 ± 40
Precision PN	0.68 ± 0.07	0.64 ± 0.04	0.71 ± 0.06	0.73 ± 0.03	0.09 ± 0.04	0.75 ± 0.08	0.08 ± 0.04	0.39 ± 0.08
Recall PN	0.70 ± 0.05	0.61 ± 0.04	0.61 ± 0.08	0.73 ± 0.06	0.014 ± 0.009	0.7 ± 0.1	0.01 ± 0.01	0.57 ± 0.09
Classifier LLF AUC	0.50	0.52	0.54	0.50	0.97	0.81	0.93	0.99
Classifier HLF AUC	0.50	0.53	0.55	0.50	0.84	0.64	0.74	0.92

- FGD_∞ on EFPs does quite well in these tests
- Would be interesting to see it used and stress tested !

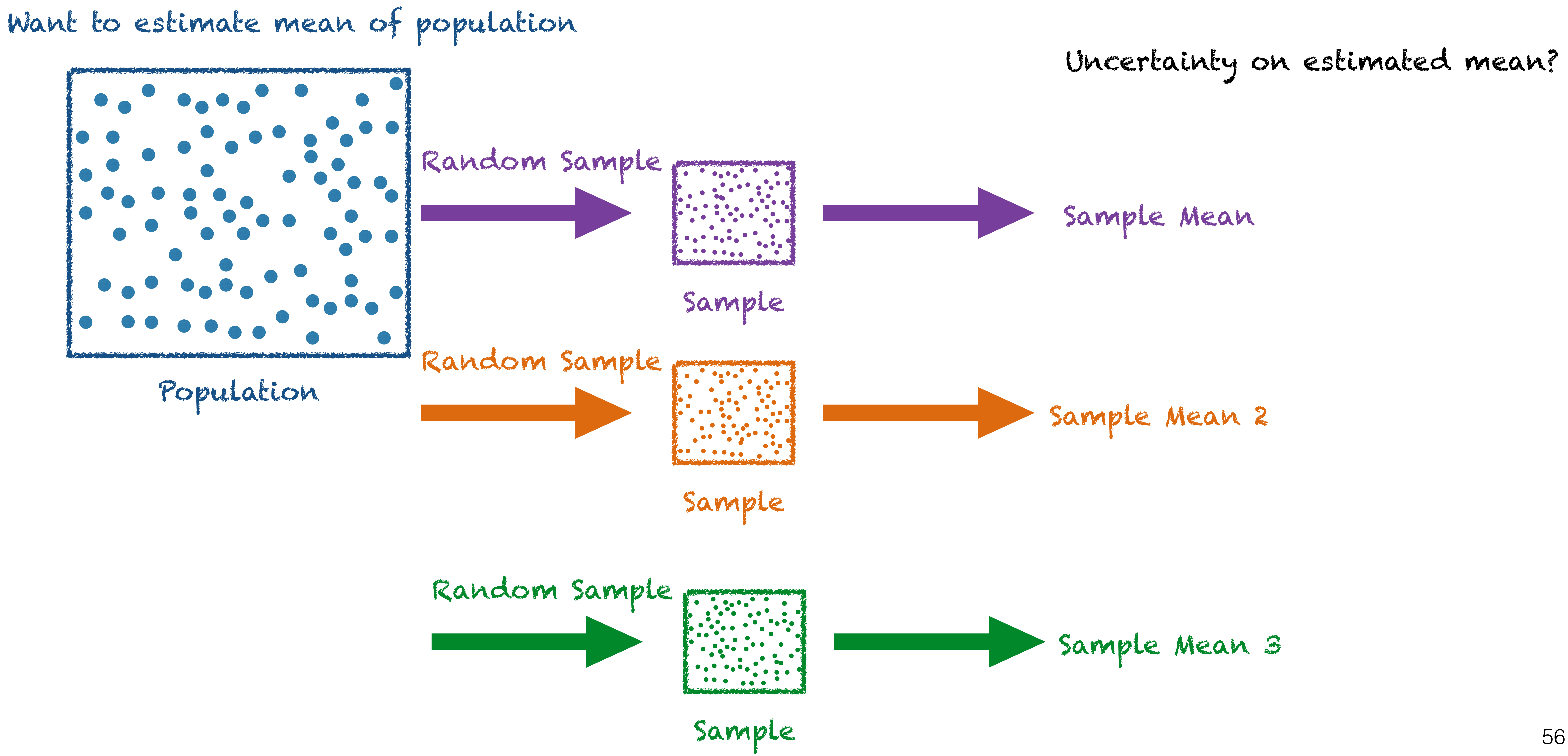
Estimating the variance on mean: Ideal Scenario

Want to estimate mean of population

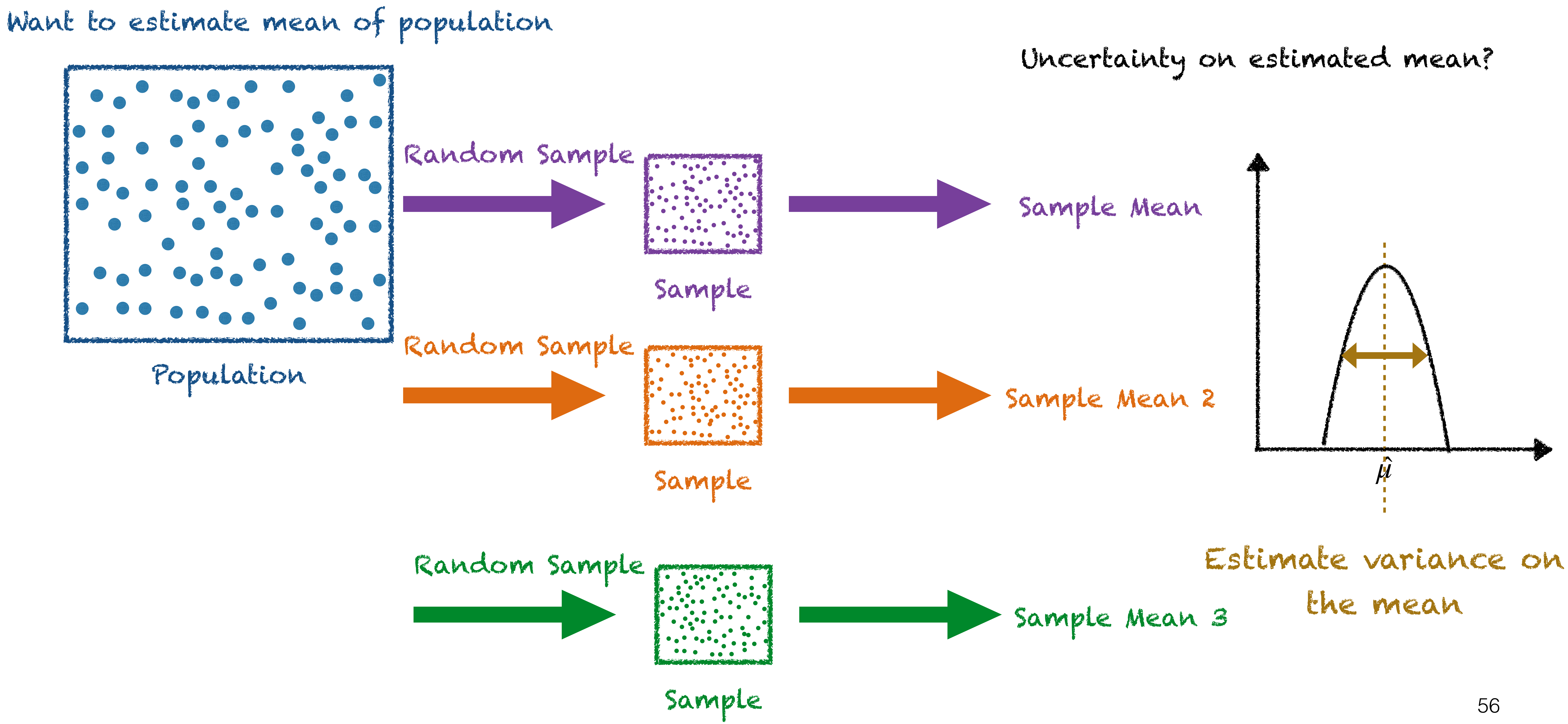
Uncertainty on estimated mean?



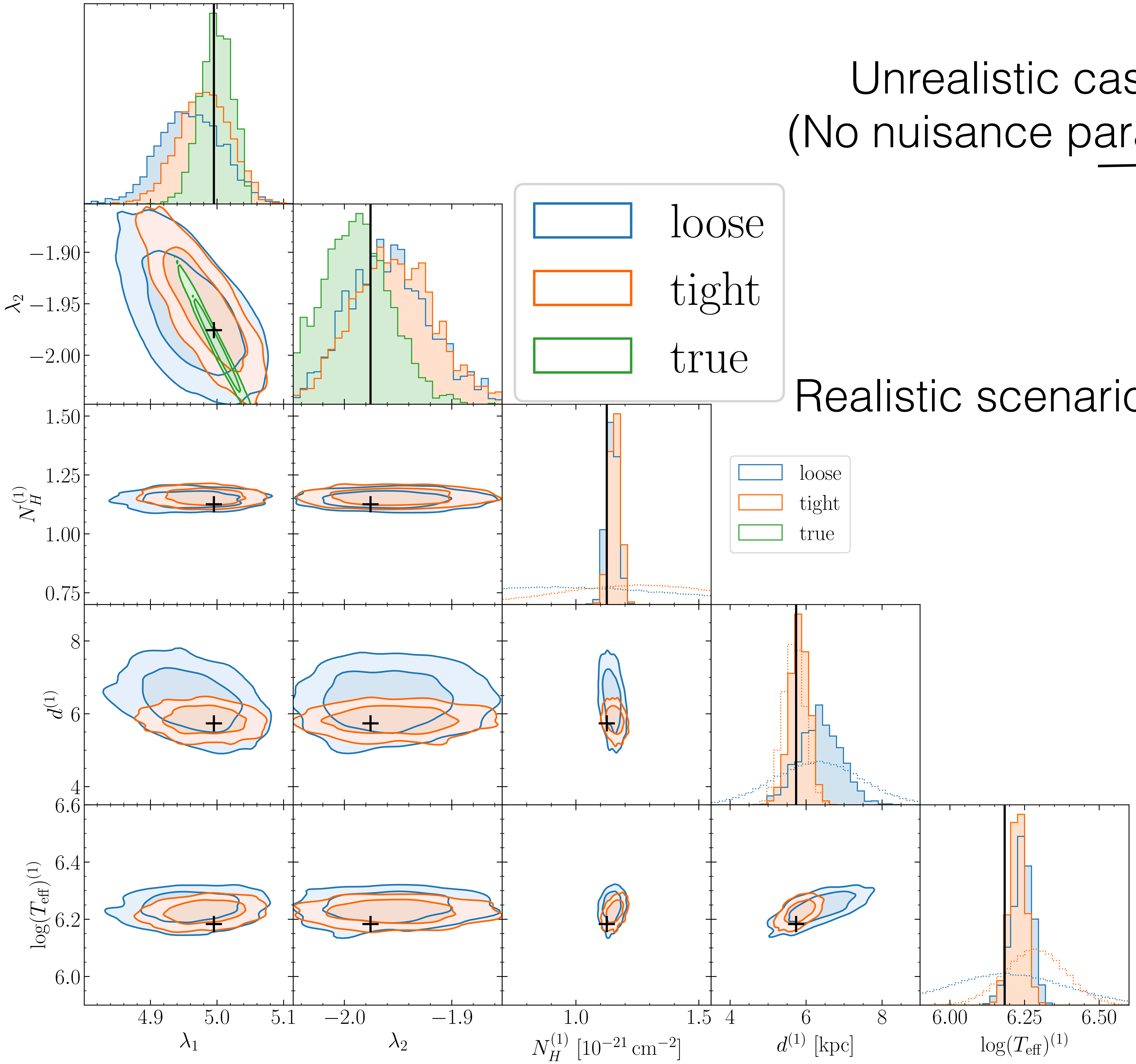
Estimating the variance on mean: Ideal Scenario



Estimating the variance on mean: Ideal Scenario



SBI for Neutron Stars:
Beats all previous methods in sensitivity and interpretability



Unrealistic case
(No nuisance params)

Realistic scenarios:

		$\lambda_{1,\rm pred} - \lambda_{1,\rm truth}$		$\lambda_{2,\rm pred} - \lambda_{2,\rm truth}$		Combined
$p(\nu)$	Method	μ	σ	μ	σ	$\sigma_{\rm tot}$
true	ML-Likelihood$_{\rm EOS}$	-0.02	0.066	**0.01**	0.070	0.096
	NN(Spectra)	-0.02	0.066	**0.01**	0.075	0.099
	NN(M, R via XSPEC)	-0.03	0.065	**0.01**	**0.055**	**0.085**
	NLE	**0.00**	**0.056**	**-0.01**	0.070	0.090
tight	ML-Likelihood$_{\rm EOS}$	-0.02	0.078	0.03	0.081	0.112
	NN(Spectra)	0.02	0.085	-0.02	0.077	0.115
	NN(M, R via XSPEC)	-0.03	0.081	**0.01**	**0.056**	0.098
	NLE	**0.00**	**0.066**	-0.02	0.071	**0.097**
loose	ML-Likelihood$_{\rm EOS}$	-0.04	0.089	0.03	0.081	0.120
	NN(Spectra)	-0.03	0.131	**-0.01**	0.078	0.152
	NN(M, R via XSPEC)	-0.03	0.123	**0.01**	**0.058**	0.136
	NLE	**0.00**	**0.085**	**-0.01**	0.074	**0.113**

[arxiv:2403.00287](https://arxiv.org/abs/2403.00287): Len Brandes, Chirag Modi, Aishik Ghosh, et al.