



Accelerating LISA inference with GP's: no-pretraining, 100x performance for expensive signals

Jesús Torrado

141th Iberian GW Meeting – 23rd June 2025

Based on [arXiv:2211.02045](https://arxiv.org/abs/2211.02045), [arXiv:2305.19267](https://arxiv.org/abs/2305.19267) and [arXiv:2503.21871](https://arxiv.org/abs/2503.21871)
with J. el Gammal, N. Schöneberg, C. Fidler, R. Buscicchio and G. Nardini

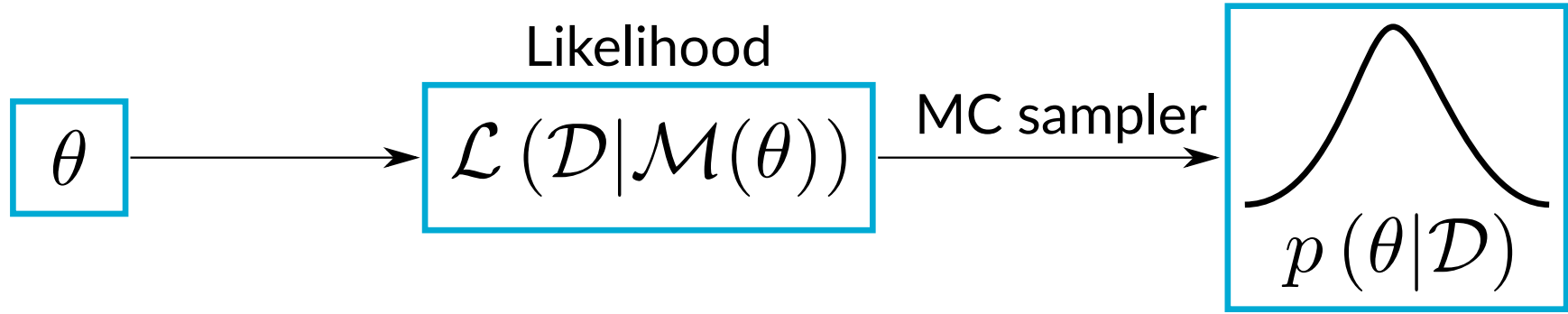


CSIC

CONSEJO SUPERIOR DE INVESTIGACIONES CIENTÍFICAS

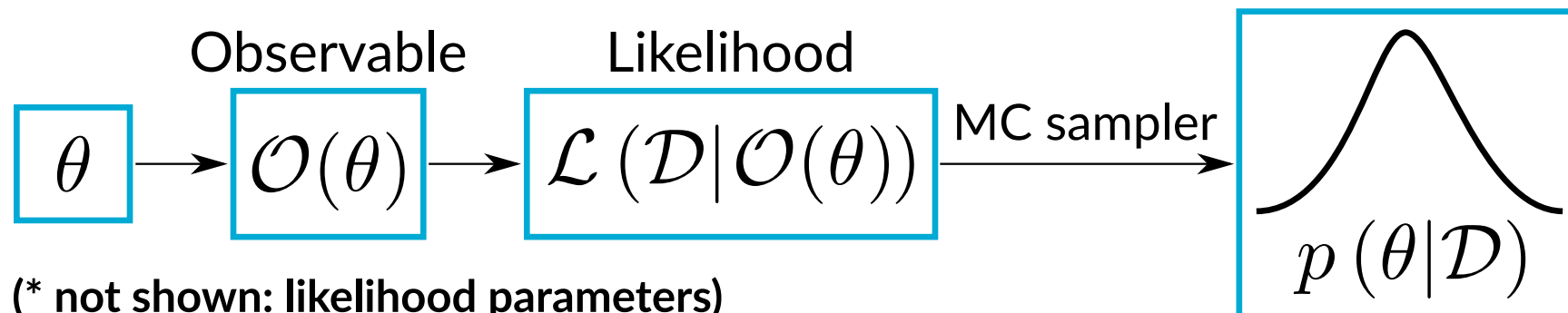
Elements of an inference problem

Likelihood-based



Elements of an inference problem

Likelihood-based with *observable*



- θ : *source parameters* ($m_i, \chi_i, d_L, \tau, \psi \dots$)
- $\mathcal{O}$: *waveform* arriving at the detector
- $\mathcal{L}$: an *experimental* likelihood: LVK, LISA, ET...
- **MC sampler**: MCMC, NS, etc with pyCBC, bilby...

Slow step: calculation of $\mathcal{O}$, and sometimes, if long integr. time, $\mathcal{L}$

Bayesian parameter inference

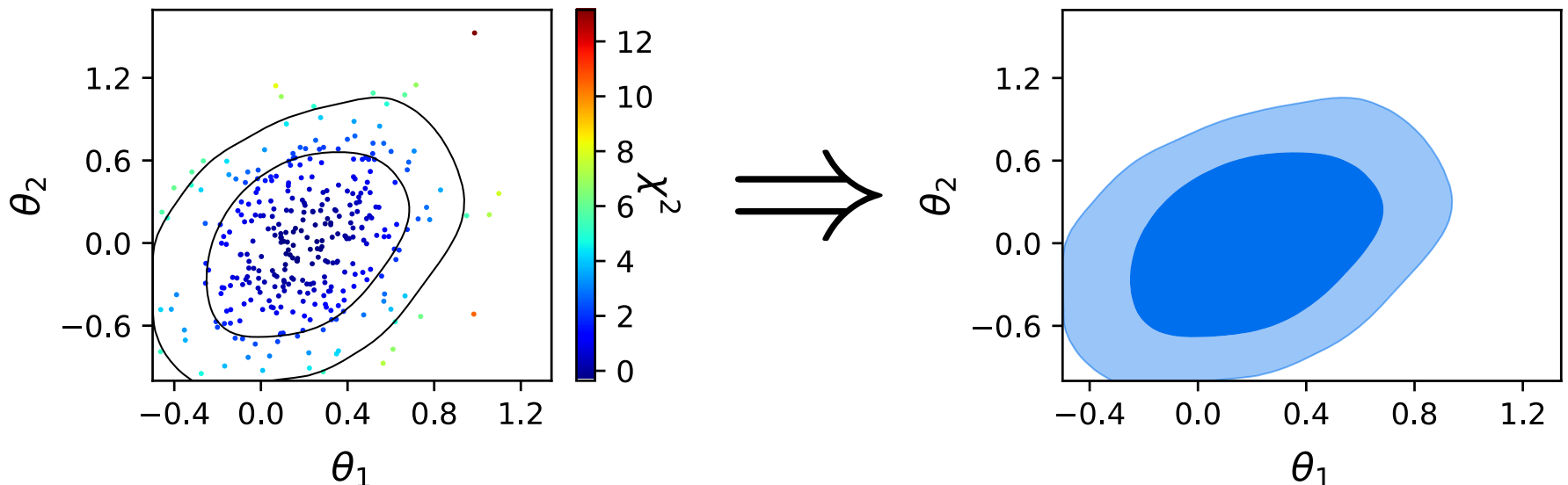
So how do we go about *mapping and characterising the posterior*?

i.e., *where are the maxima are how does probability spread around them?*

This is not trivial for not-so-simple probability distributions.

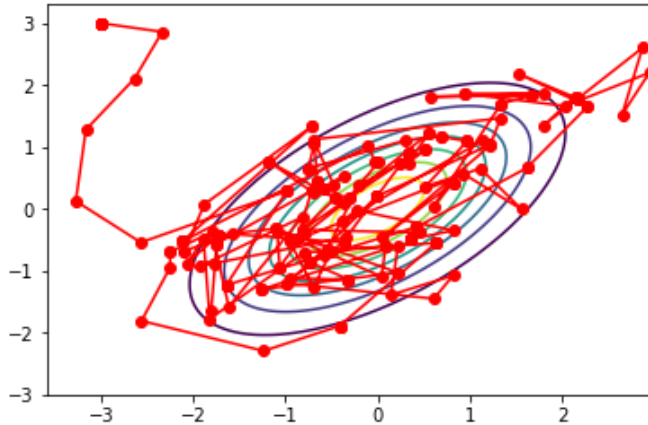
Monte Carlo approach: use algorithm (MCMC, Nested Sampling...) to produce a *sample* from the distribution.

⇒ Estimate **probability** $\propto$ **sample density**



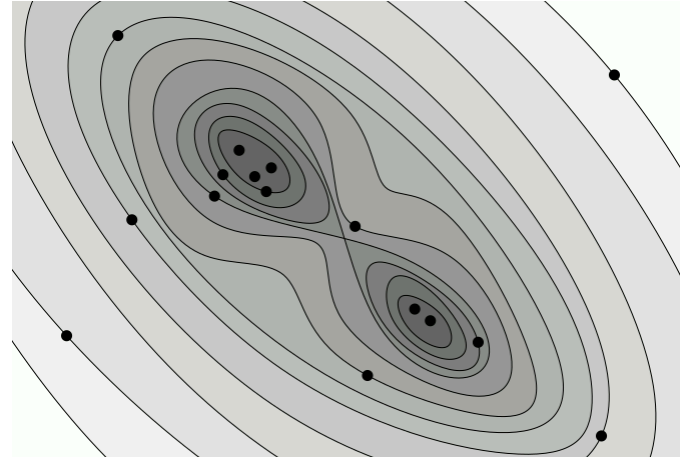
Problems of conventional approaches

MCMC



Img src: <https://relguzman.blogspot.com>

NS



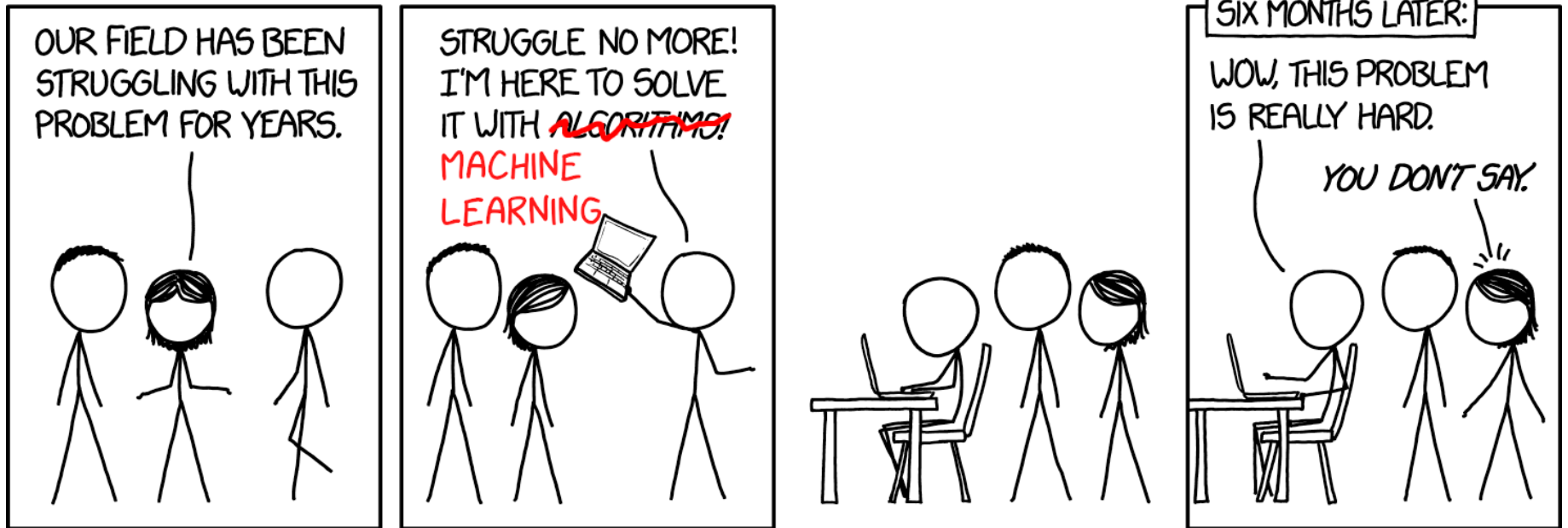
Img src: Will Handley

Estimating probability as **density** $\Rightarrow$ **lots** of samples, $\mathcal{O}(10^5)$ for $d \approx 10$.

++ data $\Rightarrow$ **++** precision $\Rightarrow$ **++** computation time $\Rightarrow$ is MC feasible **??**

Keys to reduce costs at least $\mathcal{O}(10^2)$:

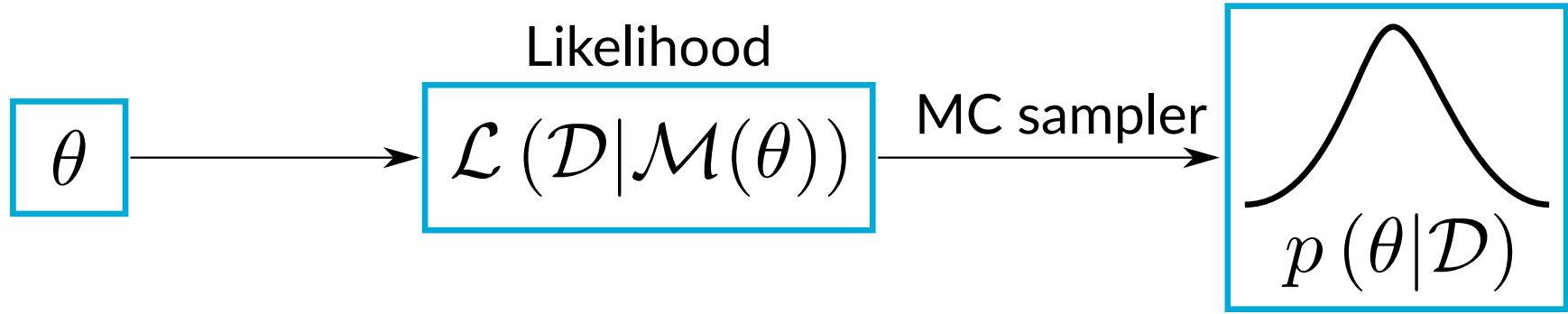
ML-based **emulation**, and heavily **parallelisation** (GPU's)



<https://xkcd.com/1831/>

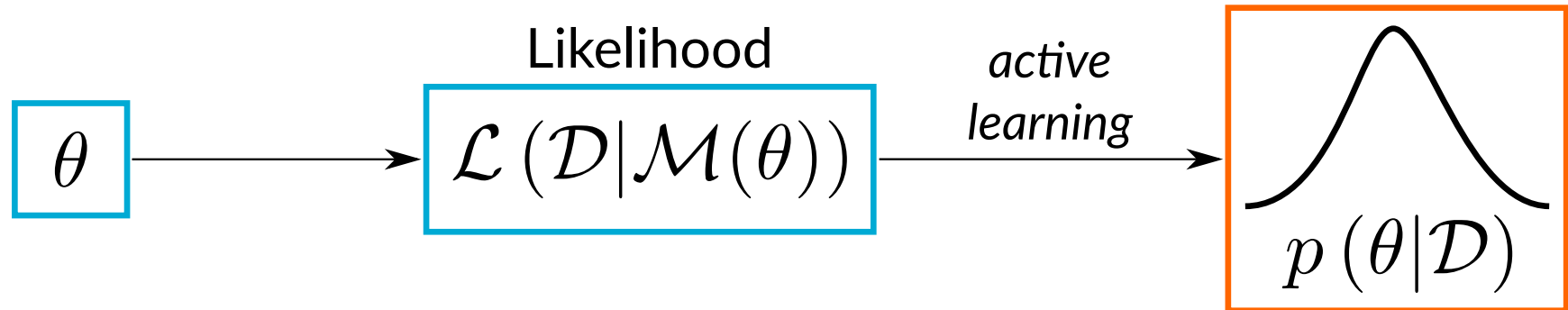
Machine-learning – *surrogate posterior*

Likelihood-based



Machine-learning – *surrogate posterior*

Likelihood-based with **ML-surrogate posterior**

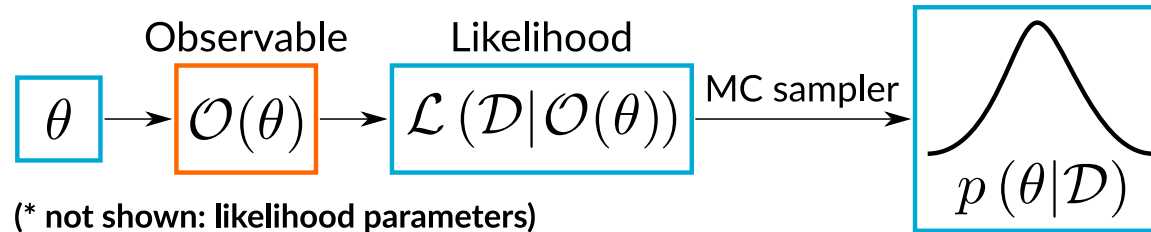


Based on Gaussian Processes: [Osborne et al. '12](#), [Gunter et al. '14](#), [Kandasamy et al. '15](#), [Chai & Garnett et al. '18](#), [Wang et al. '18](#), [Pellejero-Ibáñez et al. '19](#) (cosmo!)

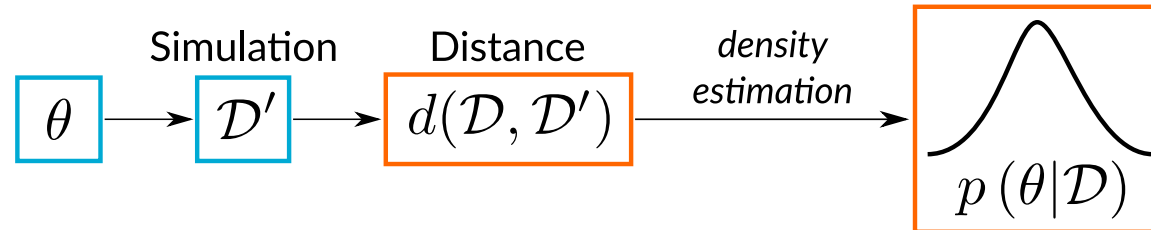
* Alternative: Variational Inference: [Acerbi '18](#), [Rizzato & Sellentin '22](#) (cosmo!)

Machine-learning – *summary*

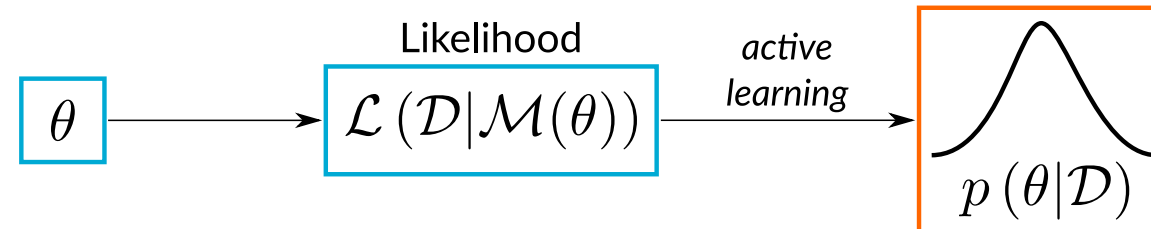
Likelihood-based with **ML-emulated observable**



Simulation-based with **ML-learned data distance & density**

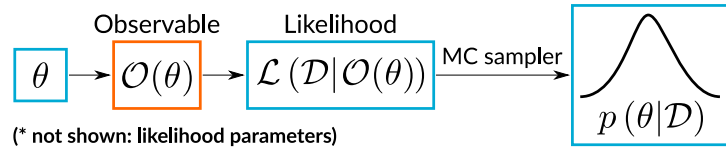


Likelihood-based with **ML-surrogate posterior**



Machine-learning – *adv. and disadv.*

Likelihood-based with ML-emulated observable



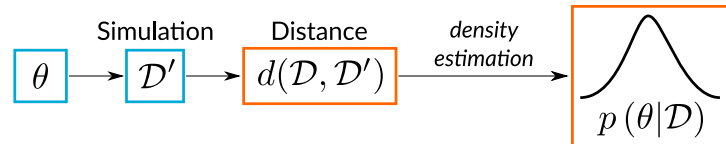
Fast to change data & data-modelling.

Easy to assess **robustness**.

Changing theory needs retraining.

Does not help if likelihood is slow / hard.

Simulation-based with ML-learned data distance & density



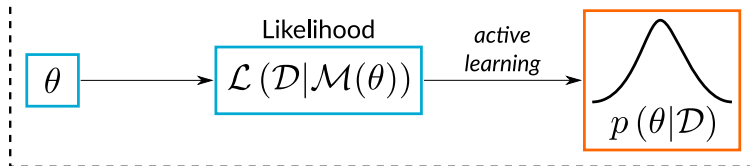
Fast to change data (not data-modelling).

Helps with large data vectors w/o summaries (*).

Changing theory or data-model needs retraining (*).

Hard to assess **robustness**.

Likelihood-based with ML-surrogate posterior

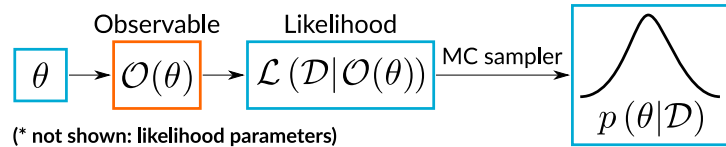


Changing anything in pipeline is "cheap" (on-the-fly training).

Hard to assess **robustness**.

Machine-learning – *adv. and disadv.*

Likelihood-based with **ML-emulated observable**



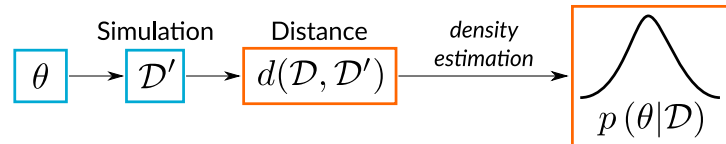
Fast to change data & data-modelling.

Easy to assess **robustness**.

Changing theory needs retraining.

Does not help if likelihood is slow / hard.

Simulation-based with **ML-learned data distance & density**



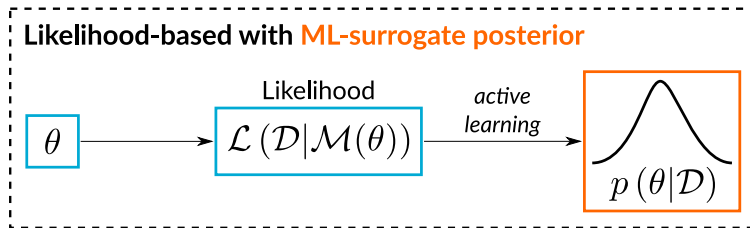
Fast to change data (not data-modelling).

Helps with large data vectors w/o summaries (*).

Changing theory or data-model needs retraining (*).

Hard to assess **robustness**.

Likelihood-based with **ML-surrogate posterior**



Changing anything in pipeline is "cheap" (on-the-fly training).

Hard to assess **robustness**.

The *FUTURE*: a combination of the three:

- **Observable emulators** for quick surveys of different data / data models.
- **Simulation-based** if likelihood hard to get, or good data summaries not clear.
- **Surrogate post.** if nothing else (trained) is available; drop-in MC replacement.

Our approach:

Emulate the "simple" mapping **[parameters $\rightarrow$ posterior]**:

$$\Omega \subset \mathbb{R}^n \longrightarrow L^1, C^{\geq 2} \in \mathbb{R}_+$$

The target function is a **positive** pdf: we model its **log**

GOAL: **drop-in** replacement for traditional MC

WISHLIST:

- No pre-training (reusability would not be a problem!)
- Small overhead: few hyperparameters and few NL calculations
- Runs fast on CPUs (though it could run faster on GPUs)
- Modest memory requirements

$\Rightarrow$ **TOOL: Gaussian Processes** $\sim$ probabilistic interpolators

(Similar approaches: [Chai & Garnett et al. '18](#), [Pellejero-Ibáñez et al. '19](#))

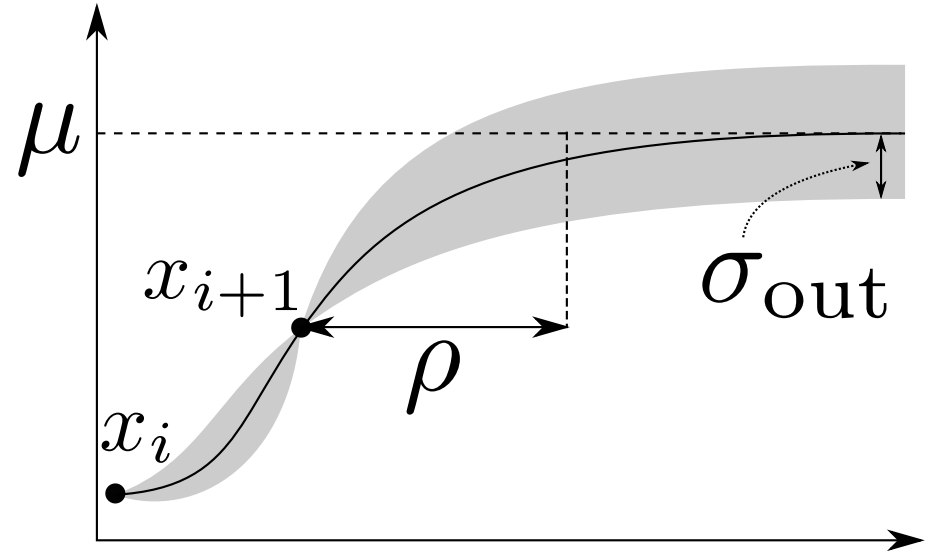
Gaussian Processes – mean, kernel and hyperparams

Rasmussen & Williams '06

$$f(x) \sim \mathcal{GP}(\mu(x), k(x, x'))$$

mean μ : baseline value to which uncorrelated values return.

kernel k : describes the correlation between points. Encodes prior information on f : e.g. smoothness, noise, periodicity...



$$y(x_*) = k(x_*, x_i)(K - \sigma_n^2 I)_{ij}^{-1} y_j$$

$$k(x_1, x_2) = \sigma_{\text{out}}^2 \cdot \mathcal{C}(|x_1 - x_2|; \rho) + \sigma_n^2 \delta_{ij}$$

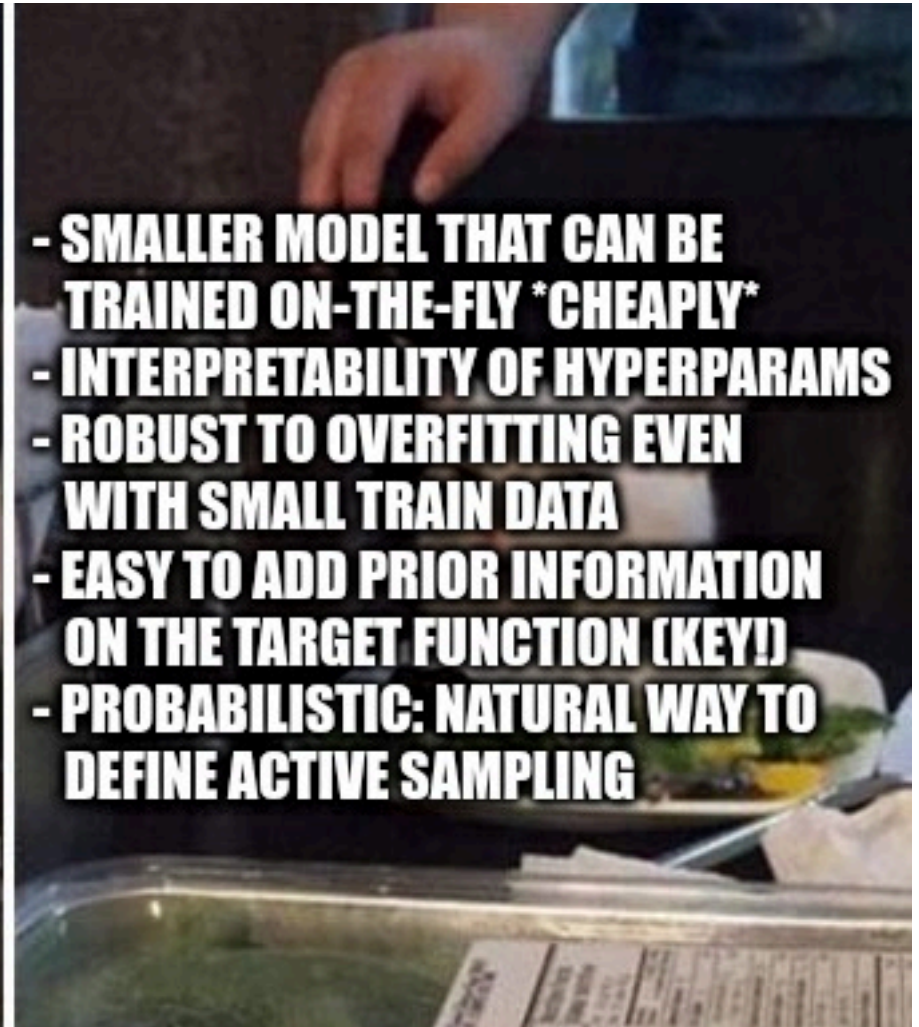
σ_{out}^2 : variance of *uncorrelated* points

ρ : correlation length(s)

σ_n^2 : white noise variance (*)

Why use Gaussian Processes?





Algorithm for surrogate-posterior inference

Modelling:

Choice of GP kernel, variable transformations...

Algorithm:

1. **Acquisition (active sampling)**: search for the next optimal locations.
2. **Evaluation of the true model** at the optimal locations.
3. **Re-fitting** of the GP model: new points and new optimal hyperparams.
4. **Convergence checks**

Profit!

Analytic, differentiable GP model of the posterior to do **inference** with.



I am developing a ML
sampler based on
Gaussian Processes



It will be robust, right?



It will be robust, right?

Robustness – *it's the hard part!*

Usual kernels: **guaranteed convergence** (any func) if sufficient training.

But in a realistic implementation will convergence...

- ... happen fast/early enough?
 - GP costs increase rapidly as $\mathcal{O}(N^3)$
 - GP's break down due to bad-conditioning of kernel matrices
- ... be detected by the convergence criterion when it happens, and not before (false positive) or *too much* later (false negative).
- ... depend on fine-tuning of hyperparameters or initial conditions?
- ... happen reliably for non-gaussian and multimodal pdf's?

In summary: if we feed the algorithm new data or a new model, will we get a result at all, and can we trust that result?

Our focus in [arXiv:2211.02045](#)

Convergence robustness by incorporating prior info

Remember our emulation problem:

$$\Omega \subset \mathbb{R}^n \longrightarrow L^1, C^{\geq 2} \in \mathbb{R}_+$$

L^1 positive continuous function on a compact domain will either:

(a) be approximately constant within the domain (easy!)

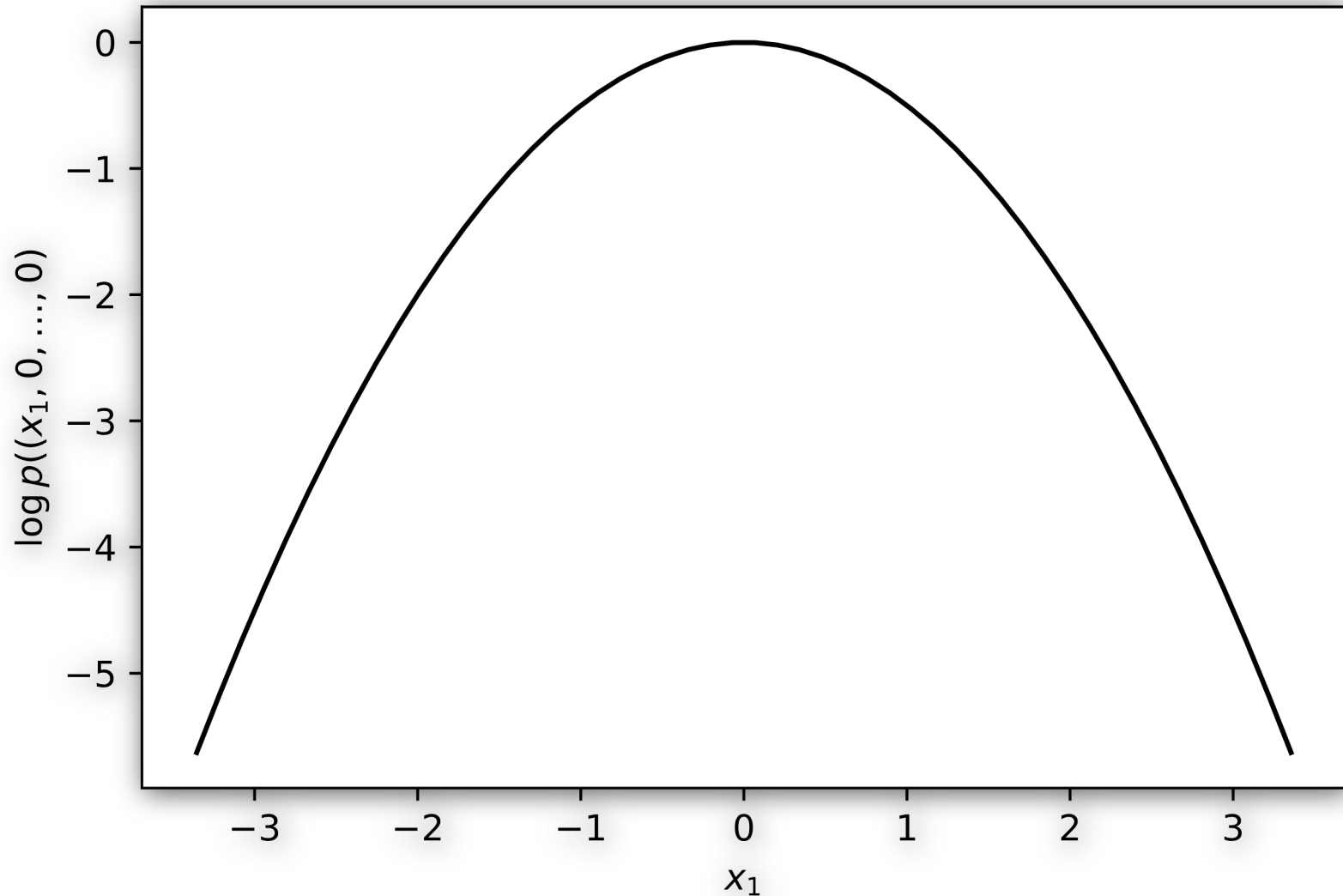
(b) have its mass concentrated around some *mode(s)*

(or a combination of both in different dimensions)

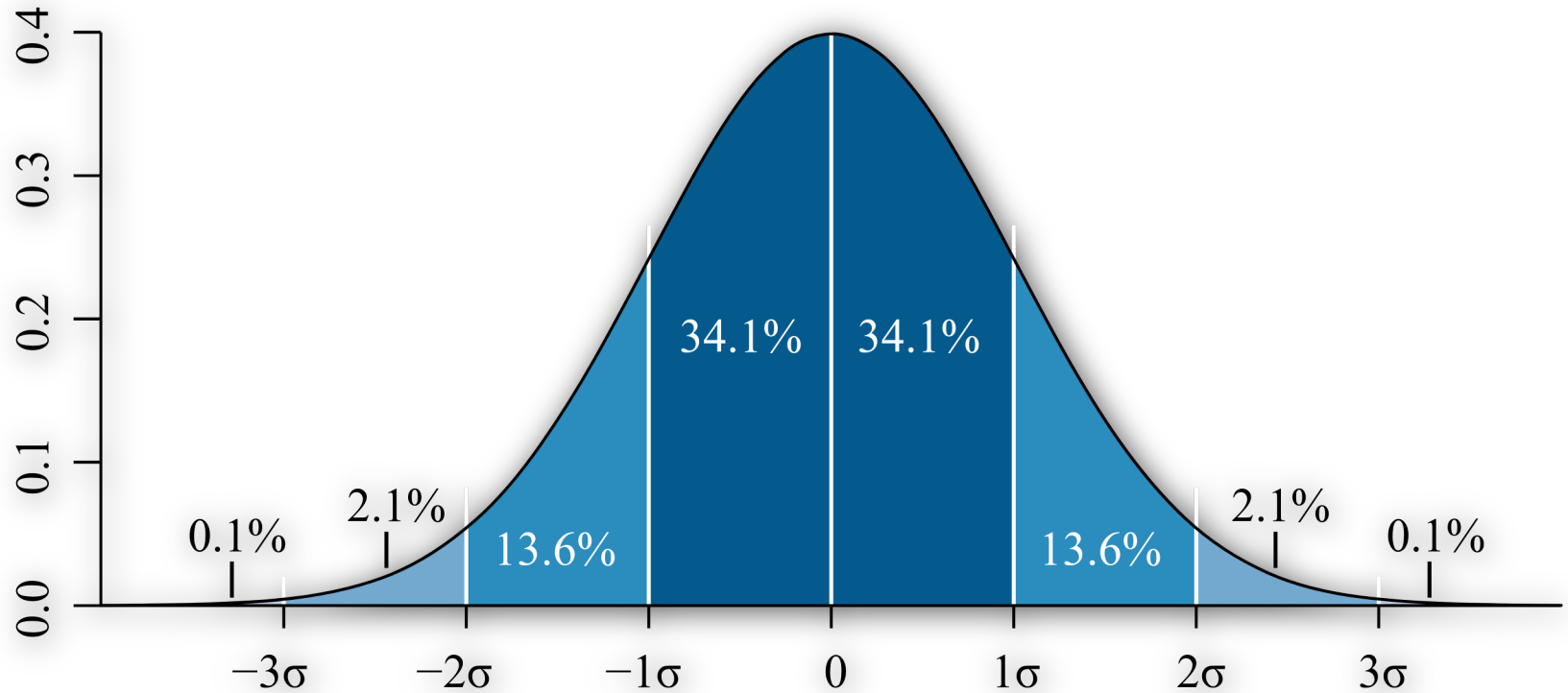
The target are **probability density functions**, so we do not need to map the whole domain correctly, just the part of the domain **that encloses most of the probability mass**.

But... where is the region that encloses most of the *mass*?

i.e. how far down the function we need accurate modelling?



1-d normalised Gaussian pdf



M.W. Toews, CC-BY 2.5 (Wikipedia)

The curse(?) of dimensionality

$$d = 1 \quad \text{---} = \text{---}$$

$$d = 2 \quad \text{---} > \text{---}$$

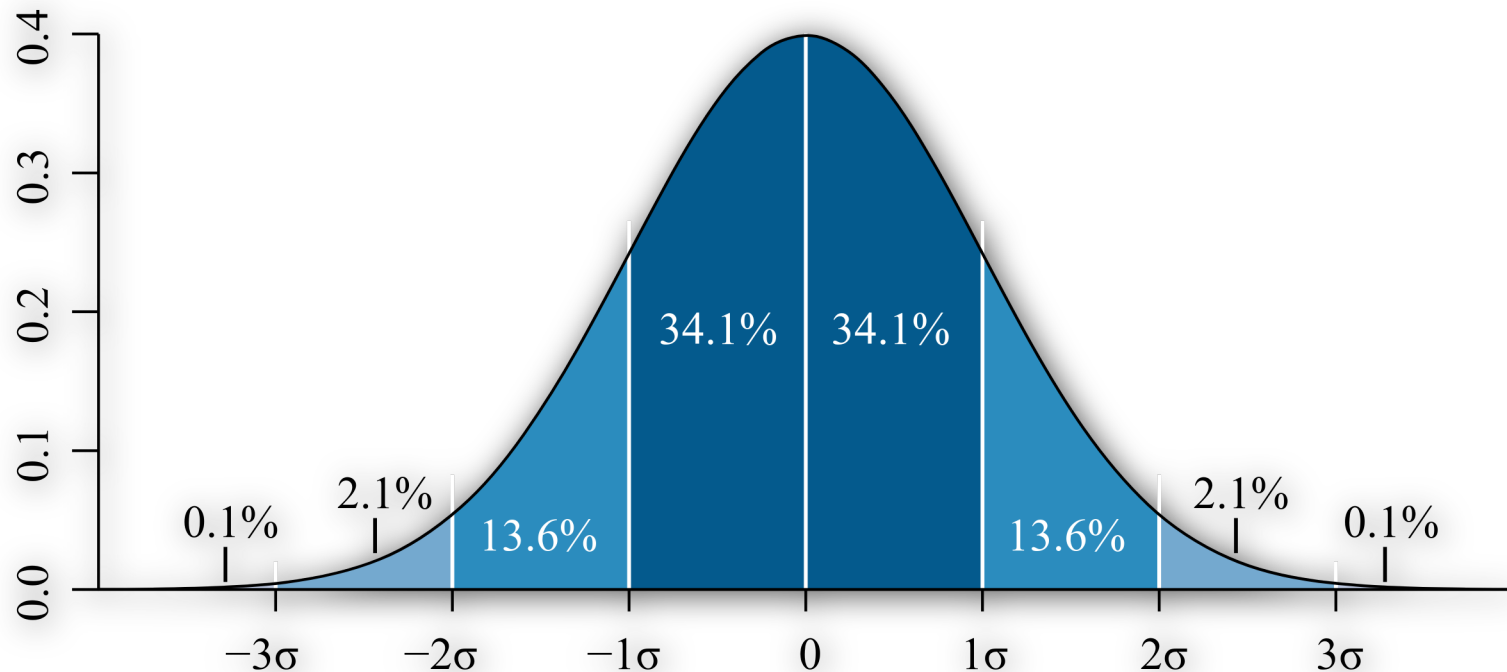
$$d = 3 \quad \text{---} \gg \text{---}$$

$\Rightarrow$ (hyper)volume **concentrates towards the edges**

$\Rightarrow$ the cost of $\approx$ grid-based methods diverges exponentially!

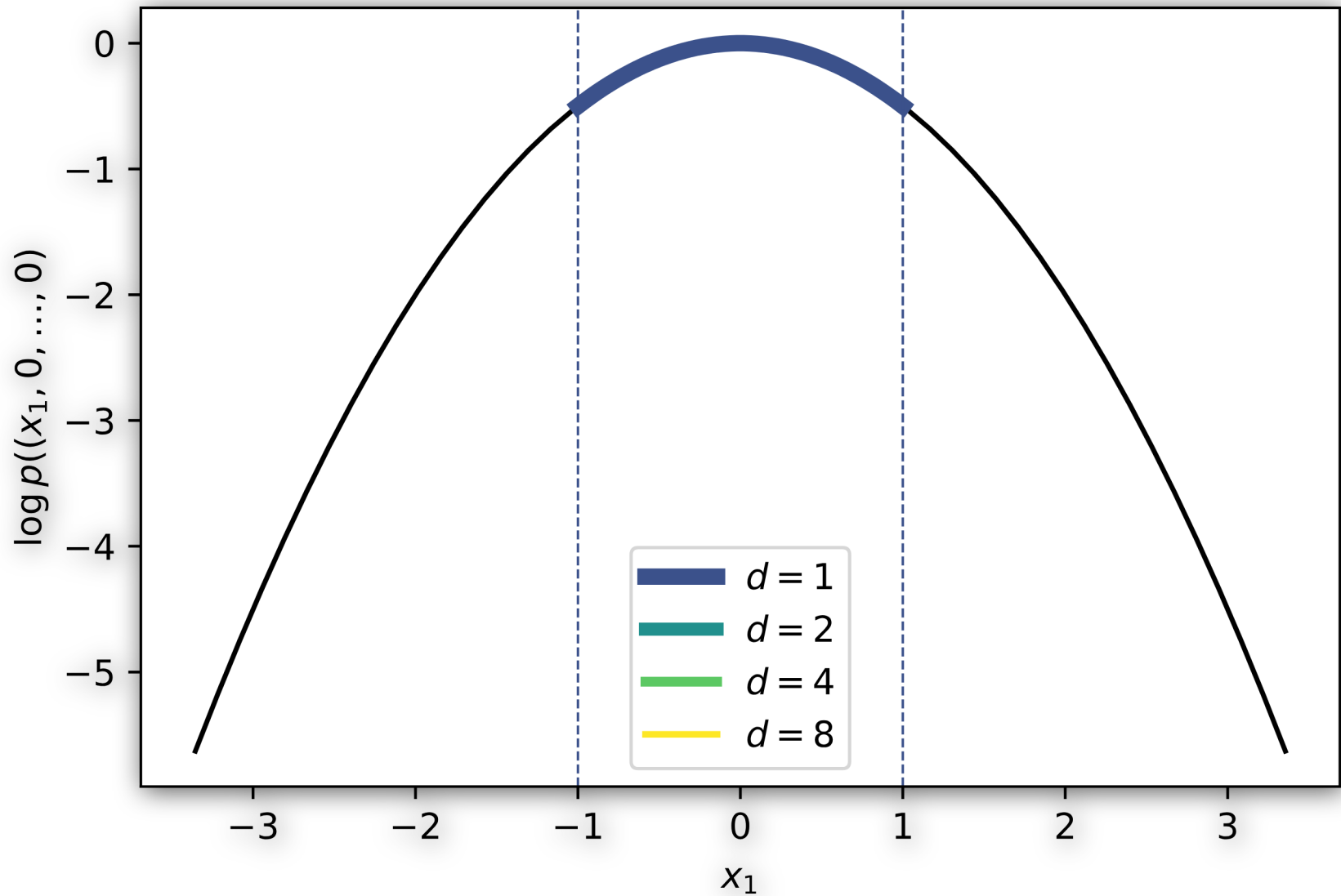
The curse(?) of dimensionality – Bayesian inference

Tails contain more and more of the **hypervolume/probability**!
It is important to map them correctly as dimensionality grows.

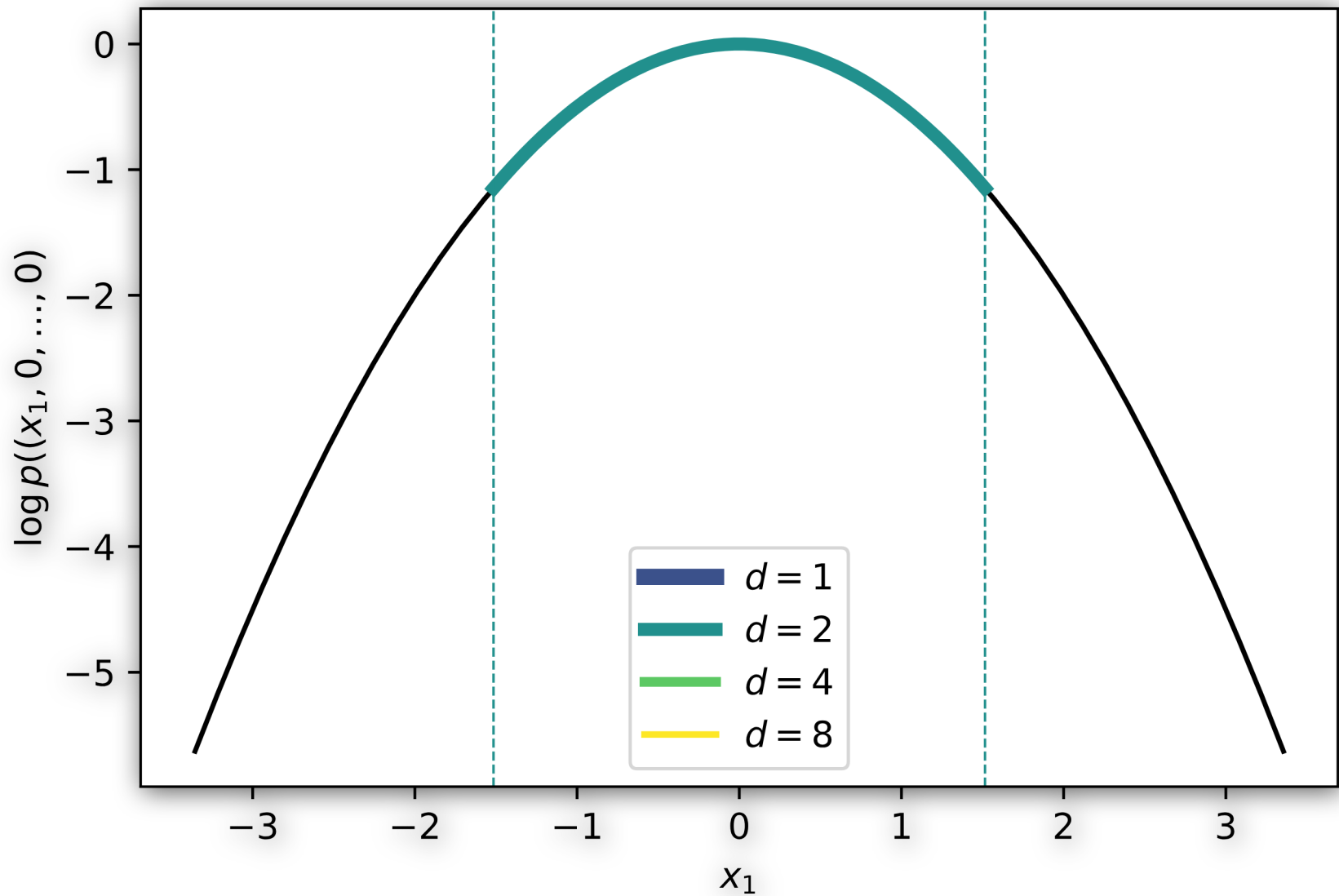


M.W. Toews, CC-BY 2.5 (Wikipedia)

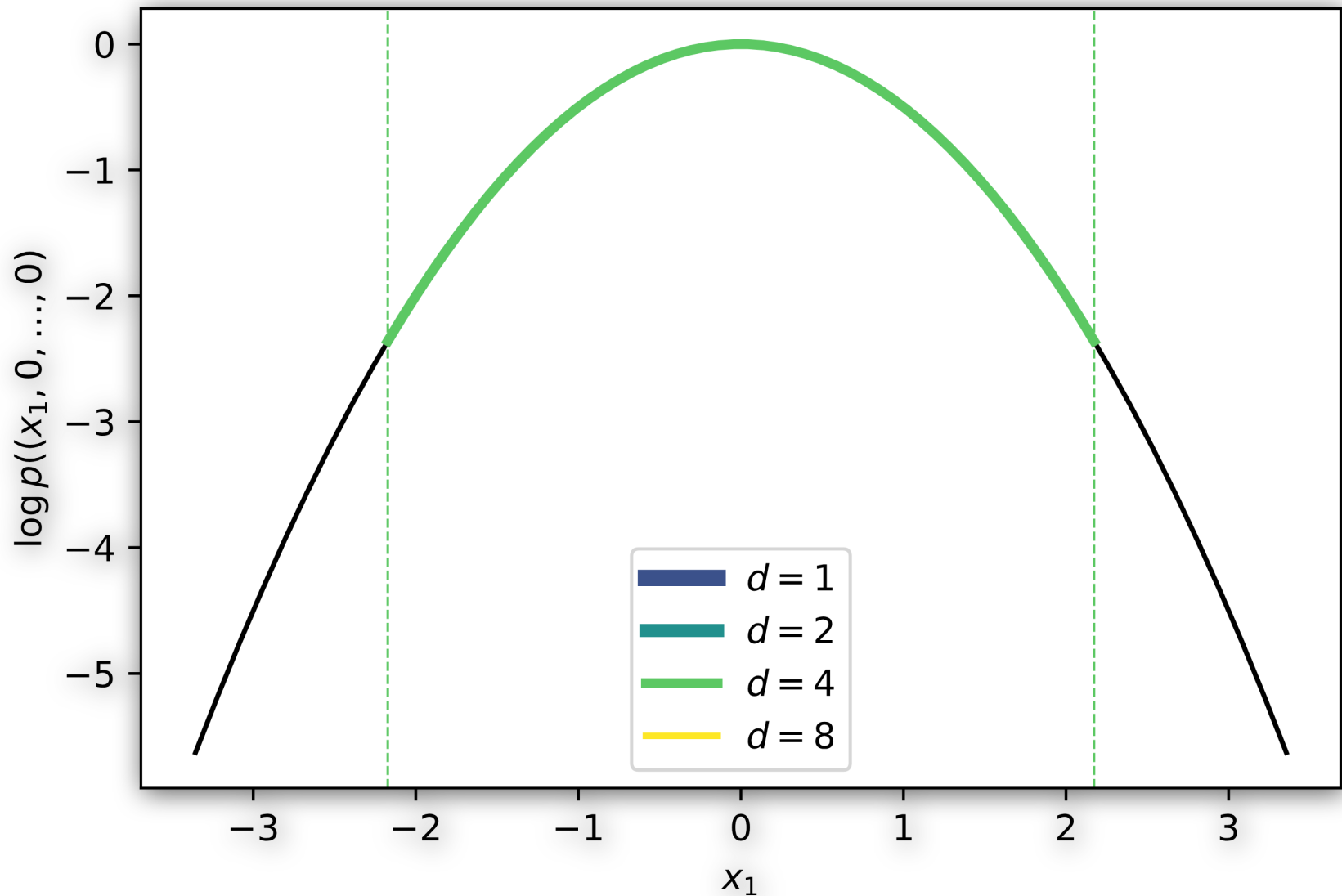
The radius of a 68% (hyper)contour



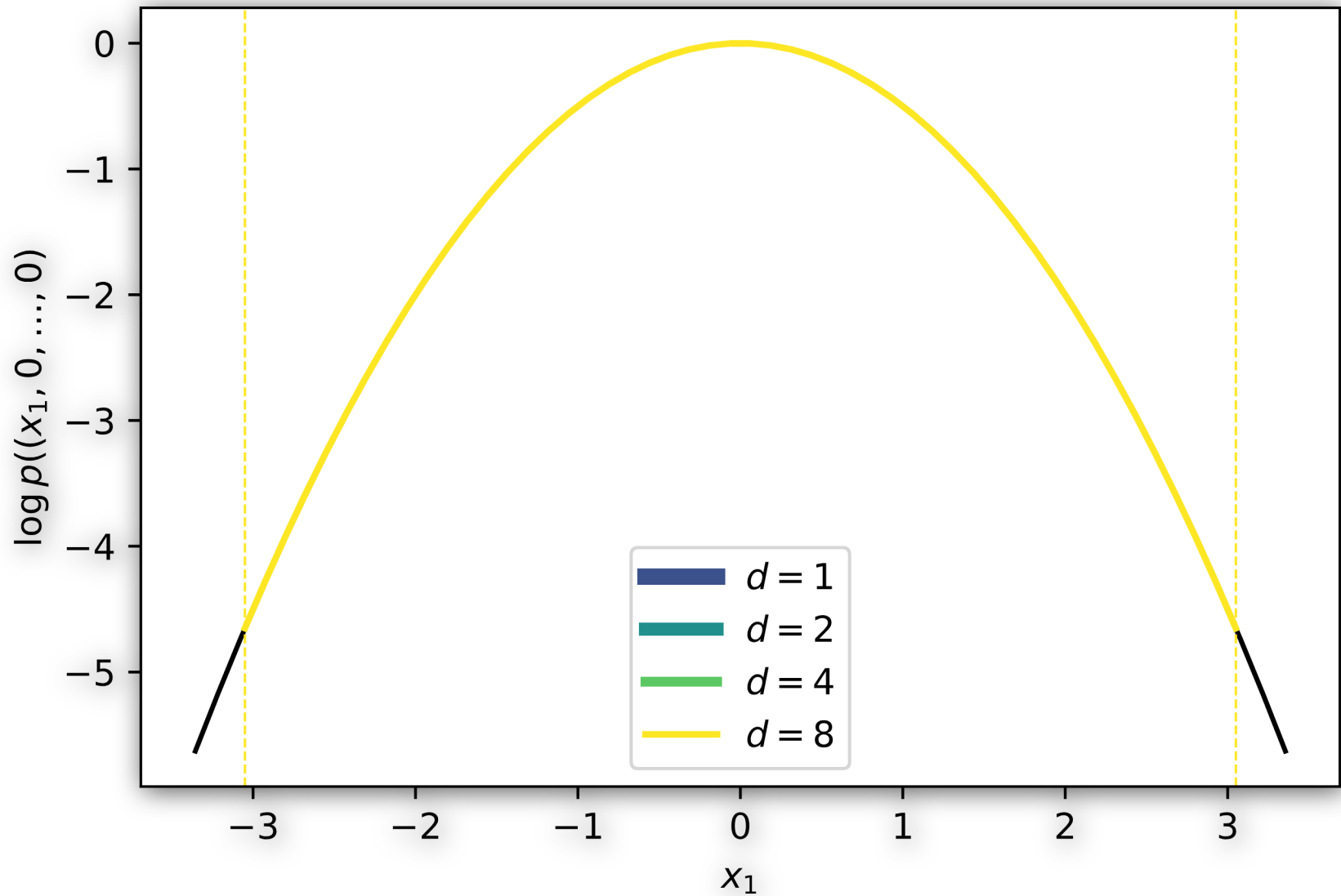
The radius of a 68% (hyper)contour



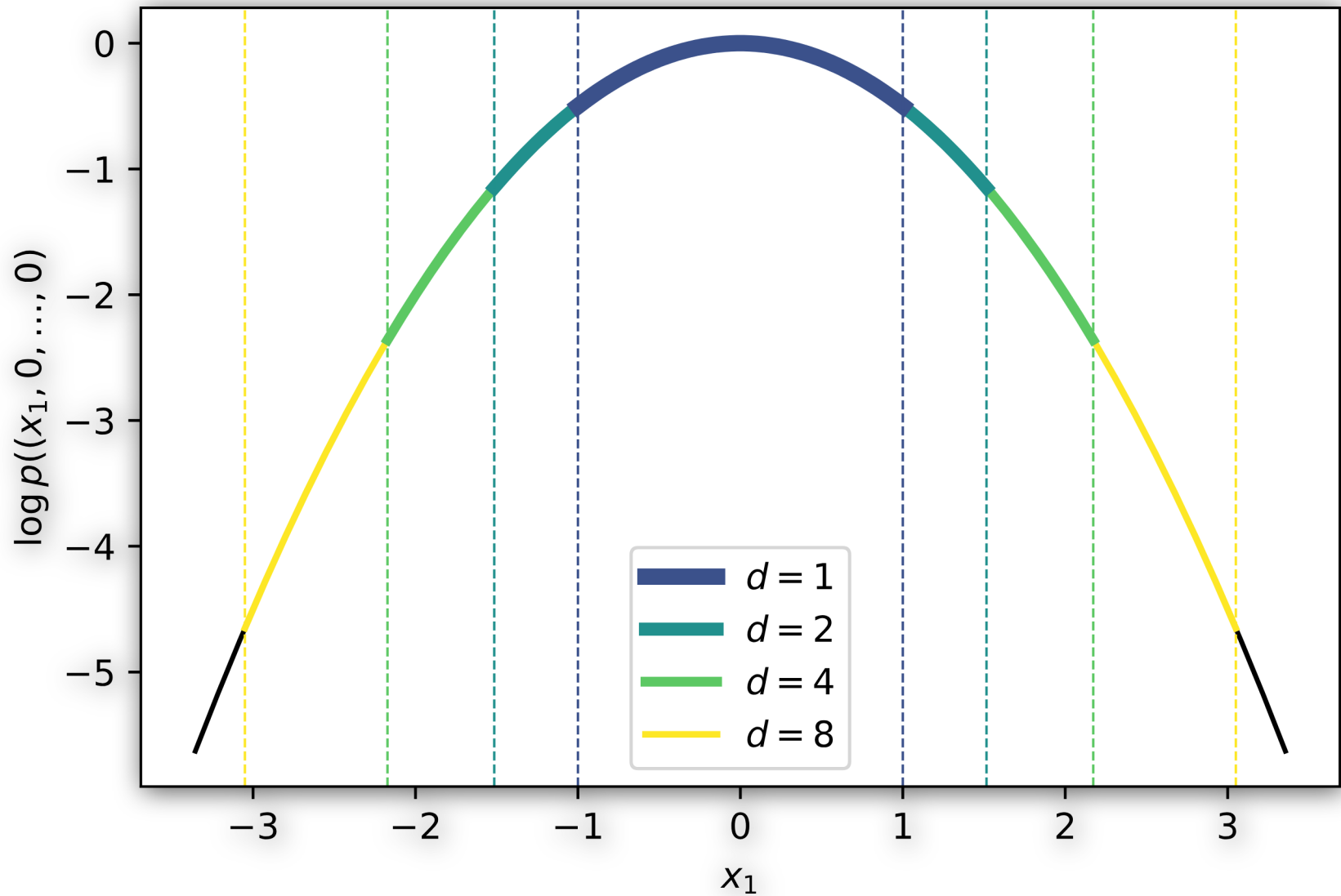
The radius of a 68% (hyper)contour



The radius of a 68% (hyper)contour



The radius of a 68% (hyper)contour



Analytic calculation of dynamic range in a Gaussian

The log-probabilities themselves follow a (1D) χ^2 with d d.o.f.:

$$(x_1, \dots, x_d) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \implies -2 \log p(\boldsymbol{x}) \sim \chi_d^2$$

By definition, for the *cumulative* function of a 1D probability distribution on x , with ϵ being the probability mass under $x > x_0$:

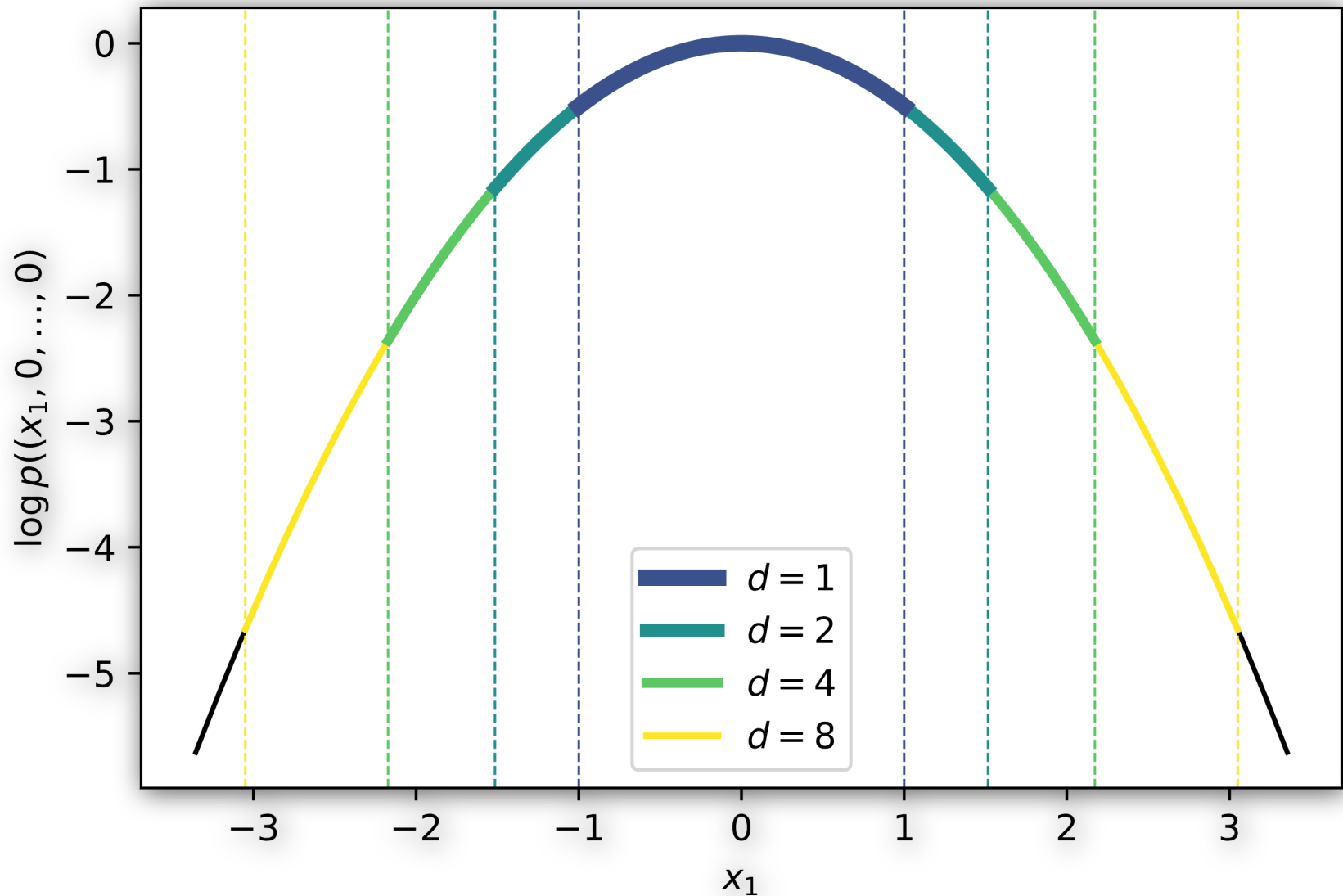
$$1 - \epsilon = F(x_0)$$

So the dynamic range of the target log-p for enclosed prob. mass $1 - \epsilon$:

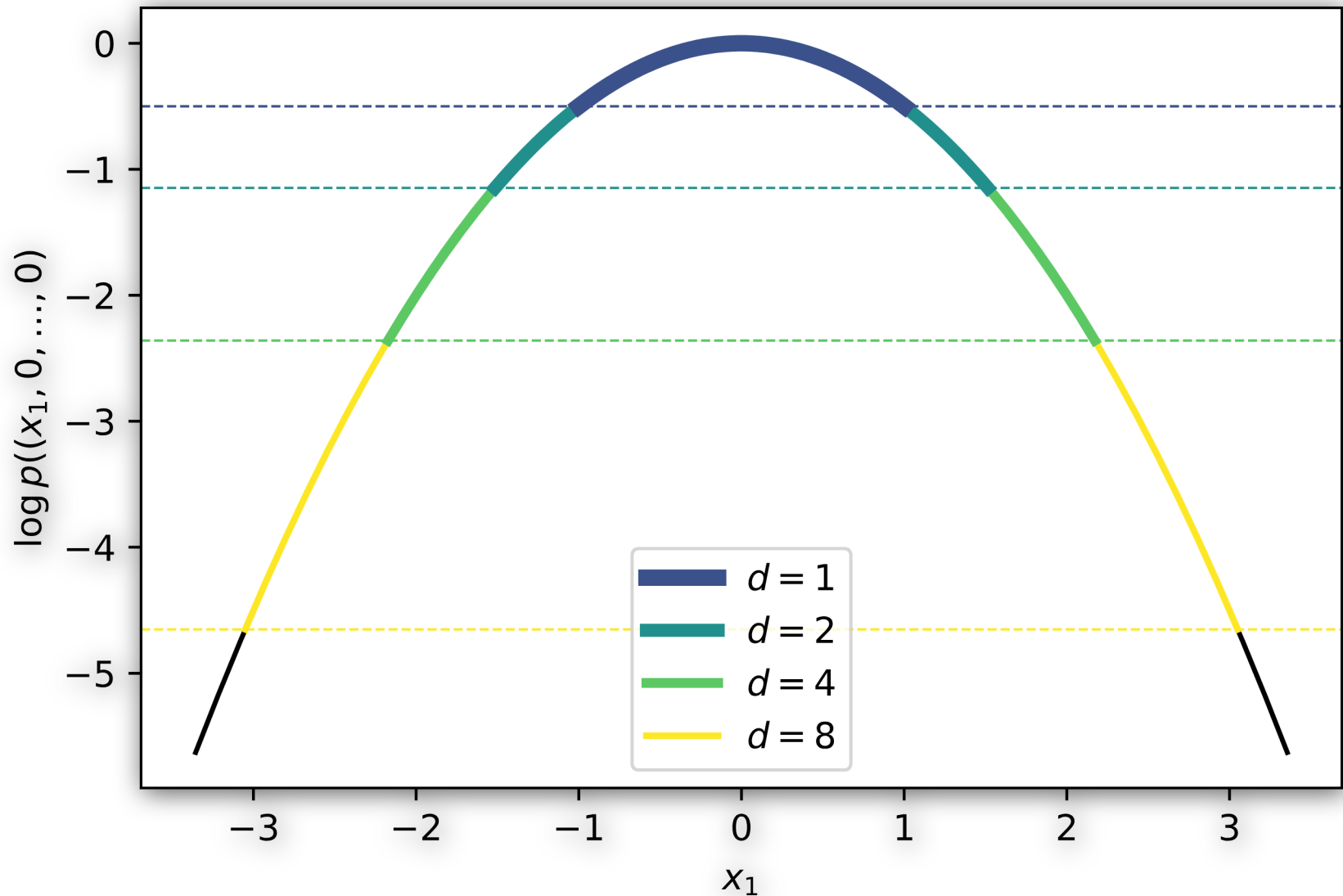
$$\Delta_{1-\epsilon} \log p = -\frac{1}{2} F_d^{-1}(1 - \epsilon)$$

(the maximum that defines the Δ is the maximum in the training set)

The radius of a 68% (hyper)contour



The radius of a 68% (hyper)contour $\Rightarrow$ dynamic range!



We use a Gaussian model to have a proxy on the relevant dynamic range of the target!

- Auto-tuning of local convergence criterion: measure of how **close** the prediction at a proposed sample is, relative to the expected dynamic range.
- Know whether a new training set has a value too low to make it worth it to include it in the model.

... without loss of generality!

- Distributions in the exponential family behave similarly, and the Central Limit Theorem tends to *gaussianize* likelihoods of large data sets.
- **The GP model itself does not assume a Gaussian shape at all!**

Algorithm for surrogate-posterior inference

Modelling:

Choice of GP kernel, variable transformations...

Algorithm:

1. **Acquisition** (**active sampling**): search for the next optimal locations.
2. **Evaluation of the true model** at the optimal locations.
3. **Re-fitting** of the GP model: new points and new optimal hyperparams.
4. **Convergence checks**

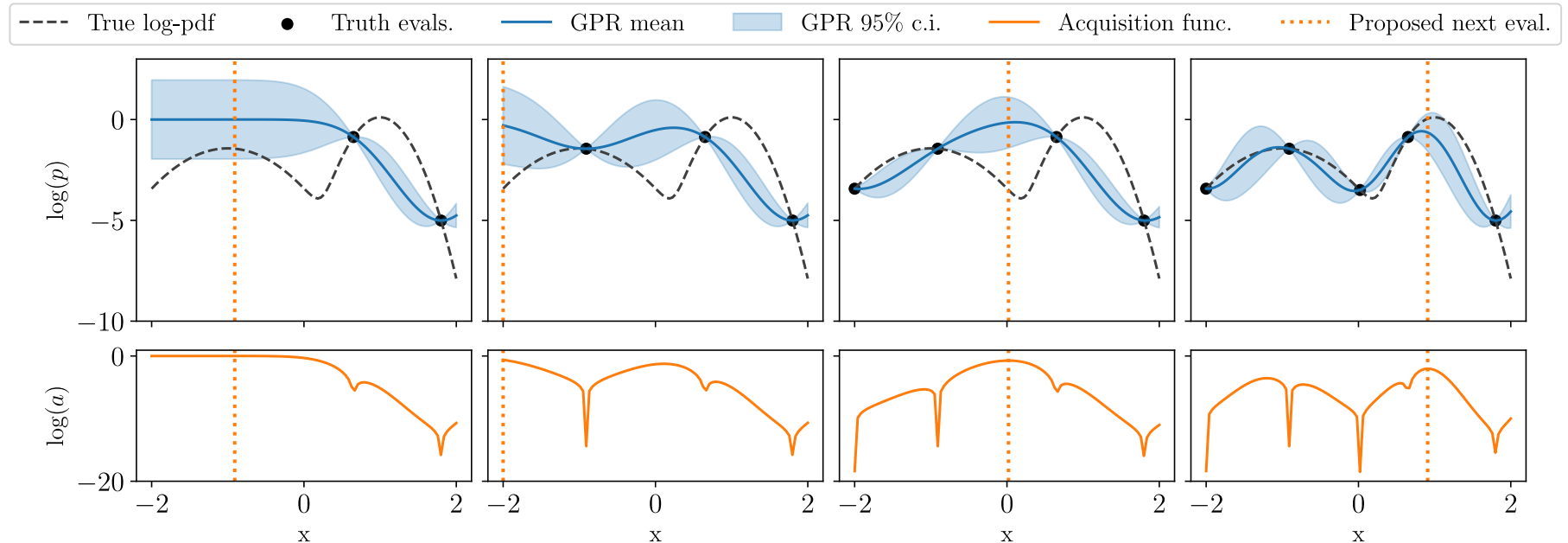
Profit!

Analytic, differentiable GP model of the posterior to do **inference** with.

Active sampling – where to evaluate next?

Key to reducing the number of evaluations:
choosing well where to evaluate next.

Optimal locations are chosen such that they maximise an **acquisition function**, usually defined in terms of how **reduction in local uncertainty** would affect some **global target**.



Active sampling – where to evaluate next?

Key to reducing the number of evaluations:
choosing well where to evaluate next.

Optimal locations are chosen such that they maximise an **acquisition function**, usually defined in terms of how **reduction in local uncertainty** would affect some **global target**.

E.g. when modelling a pdf, reduction in variance of GP evidence

$$\text{target: pdf } p(\theta) \quad \Longrightarrow \quad \text{acq. function: } \Delta \text{Var}(\mathcal{Z} | \log \mathcal{GP})$$

But that is **very expensive** to compute! (involves an integral)

Ideally, use a local measure instead of a global one
but they introduce fine-tuning problems (more on this later...)

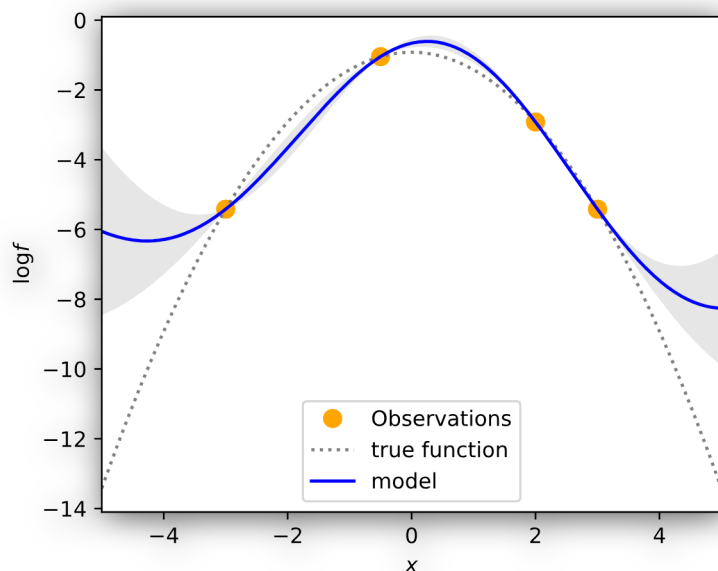
Active sampling – a local function

When modelling the **log**-posterior, the derived posterior uncertainty **depends on the target function value**:

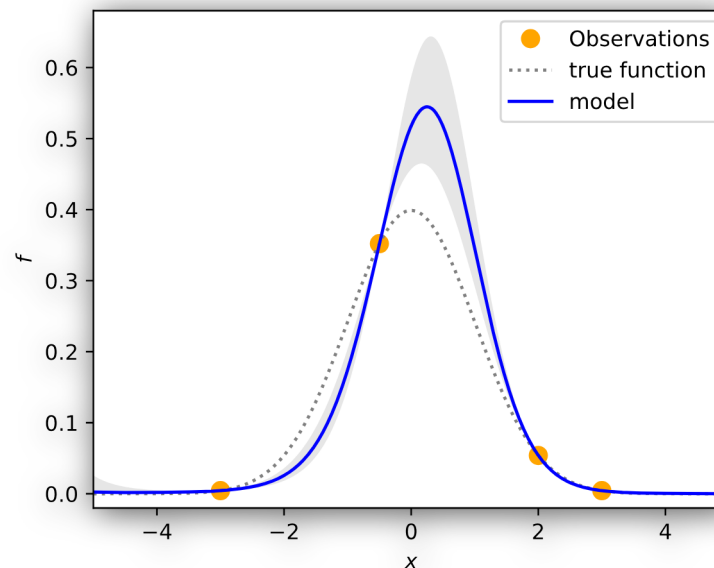
$$\log p \sim \mathcal{GP} \Rightarrow \sigma_p = e^{\mu_{\log p}} (e^{\sigma_{\log p}} - 1) \propto \mu_p$$

Gunter, Osborne et al '14; Chai & Garnett '18

$\mathcal{GP}$ model on $\log p$



$e^{\log p}$



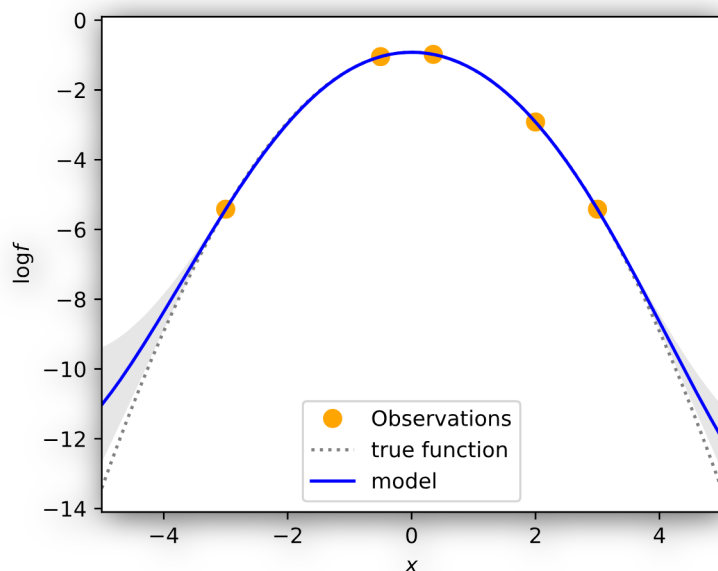
Active sampling – a local function

When modelling the **log**-posterior, the derived posterior uncertainty **depends on the target function value**:

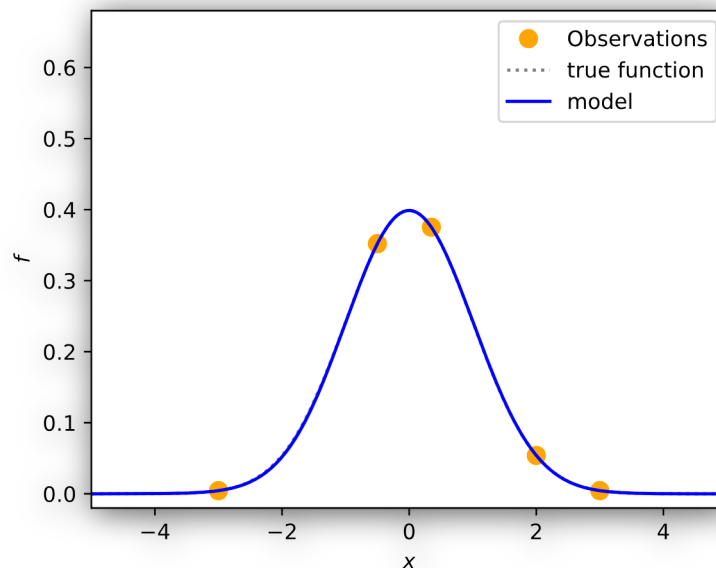
$$\log p \sim \mathcal{GP} \Rightarrow \sigma_p = e^{\mu_{\log p}} (e^{\sigma_{\log p}} - 1) \propto \mu_p$$

Gunter, Osborne et al '14; Chai & Garnett '18

$\mathcal{GP}$ model on $\log p$



$e^{\log p}$



Active sampling – exploration vs exploitation

$$\alpha(x) = e^{\mu_{\log p}(x) \cdot \zeta} \left(e^{\sigma_{\log p}(x)} - 1 \right)$$

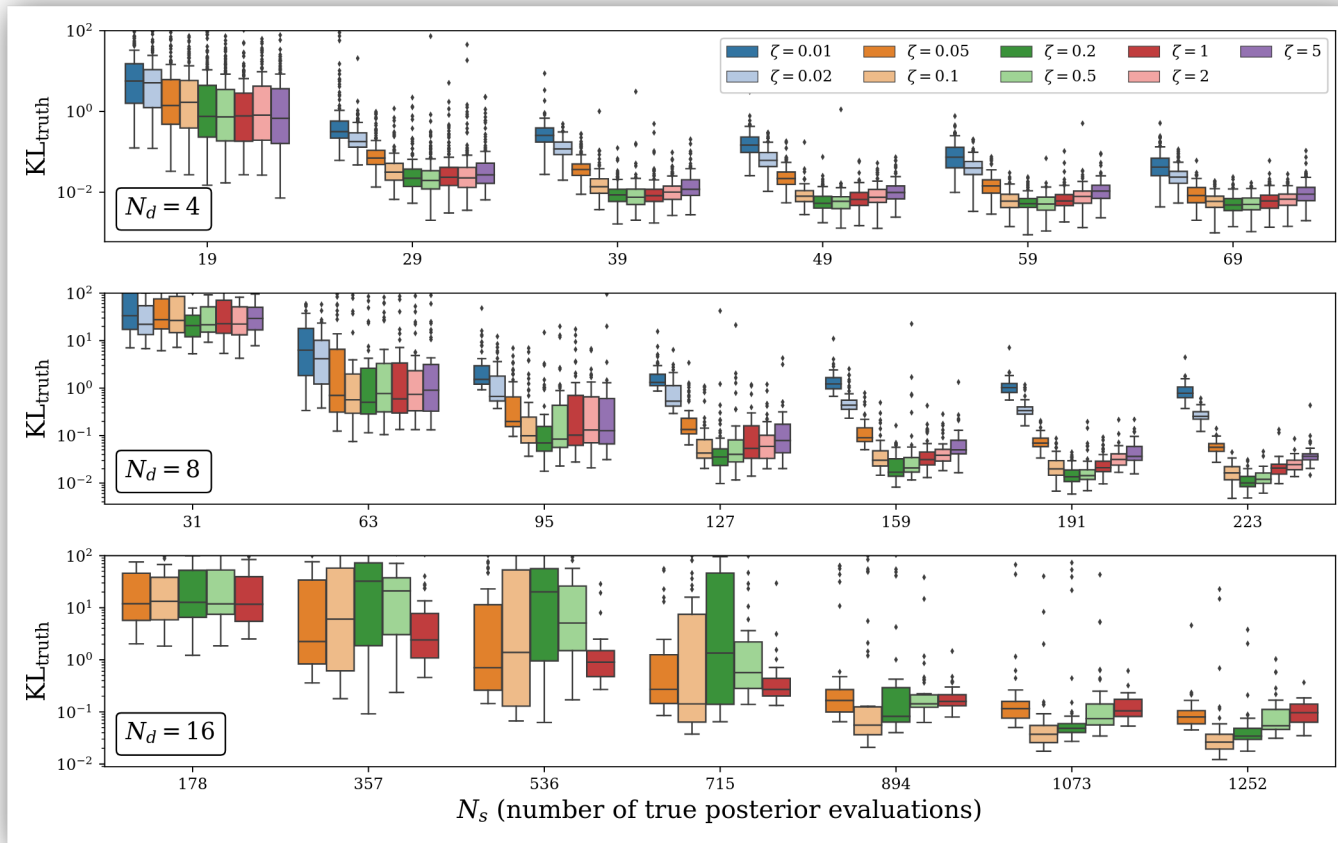
- **Exploitation**: focus on areas of high value of the target.
- **Exploration**: encourages looking far away from training points, where uncertainty is higher.
- **Balance**: regulate or switch between the two.

Usually requires fine-tuning of ζ for efficient convergence!

Our contribution: modelling and automatising this tuning

Acquisition function – empirical fit of ζ

$\zeta \approx d^{-0.85} \Rightarrow$ more **exploratory** for higher dim!



Acquisition function, revisited

We discussed the acquisition function

$$\alpha(x) = e^{\mu_{\log p}(x) \cdot \zeta} \left(e^{\sigma_{\log p}(x)} - 1 \right)$$

We use a linearised version of it: (the factor 2 is a convention)

$$\alpha^{\text{lin}}(x) = e^{\mu_{\log p}(x) \cdot 2\zeta} \sigma_{\log p}(x)$$

Since the **dynamic range of $\log p$ grows with dimensionality**, and the *exploitation* term e^{μ} depends on absolute values of the $\log p$, this **acquisition function(s) becomes greedier and greedier with dimensionality**.

Solution: make $\zeta(d)$ a decreasing function of d .

No obvious way to model that behaviour analytically, as before:
let's find an empirical fit that optimises convergence.

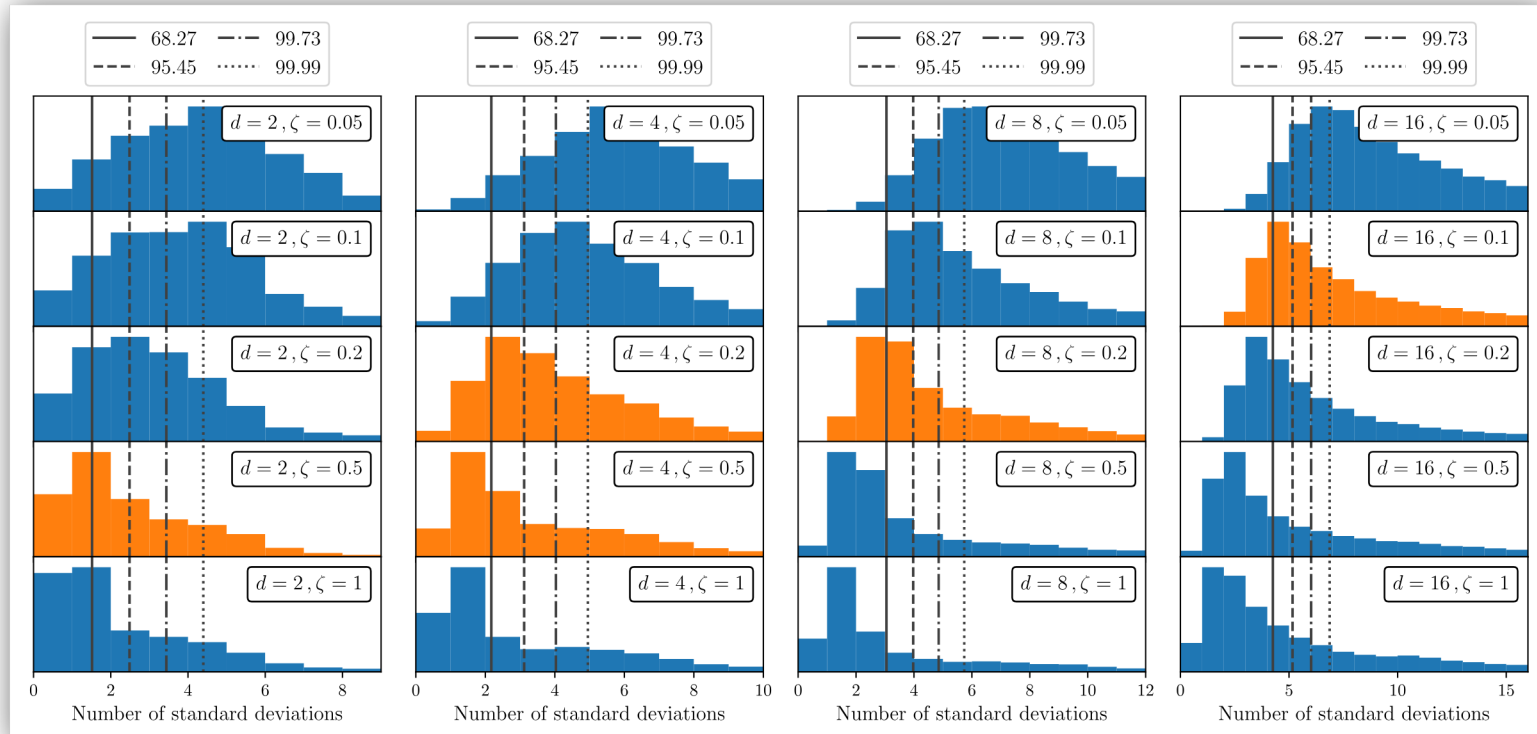
Acquisition function – empirical fit of ζ (II)

Do optimal ζ 's follow consistent hypercontours across dimensionalities?

Acquisition function – empirical fit of ζ (II)

Do optimal ζ 's follow consistent hypercontours across dimensionalities?

Yes, they do! In particular they focus in the 68% one.

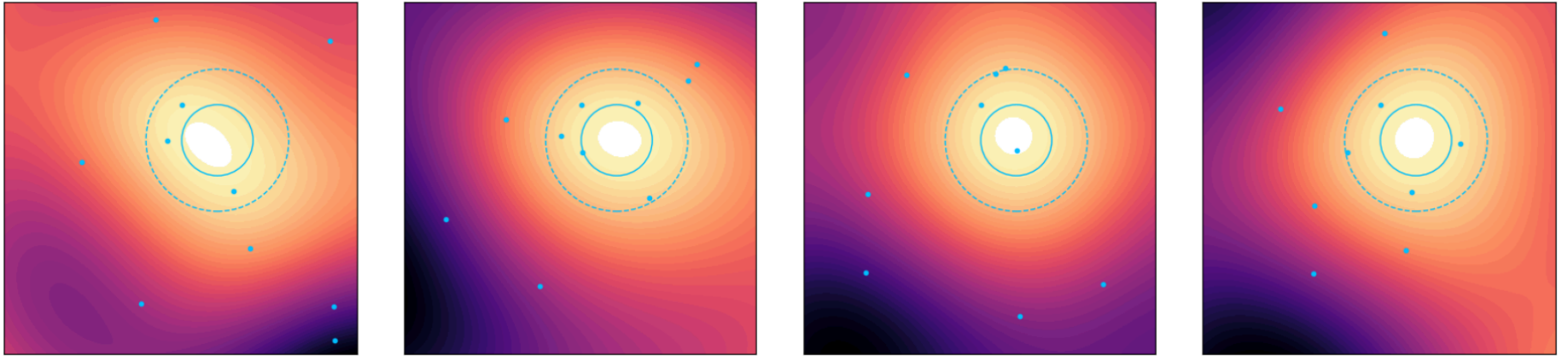


NB: Here we have not used the modelling of the dimensionality dependence presented before. It just chooses to follow it to get better convergence!

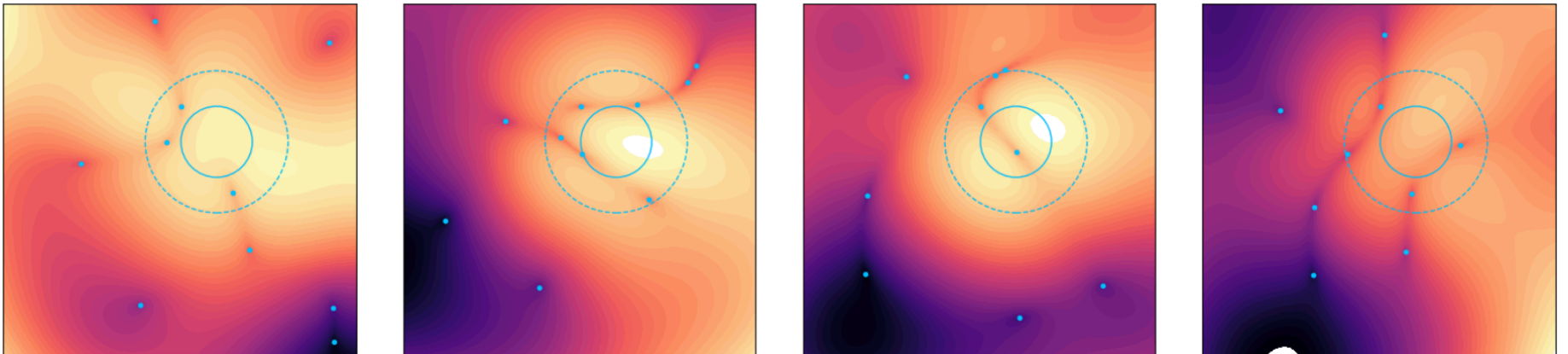
Active sampling – optimisation

Global maxima of **highly multi-modal** function → **Hard!**

Even if the model is nice and simple...



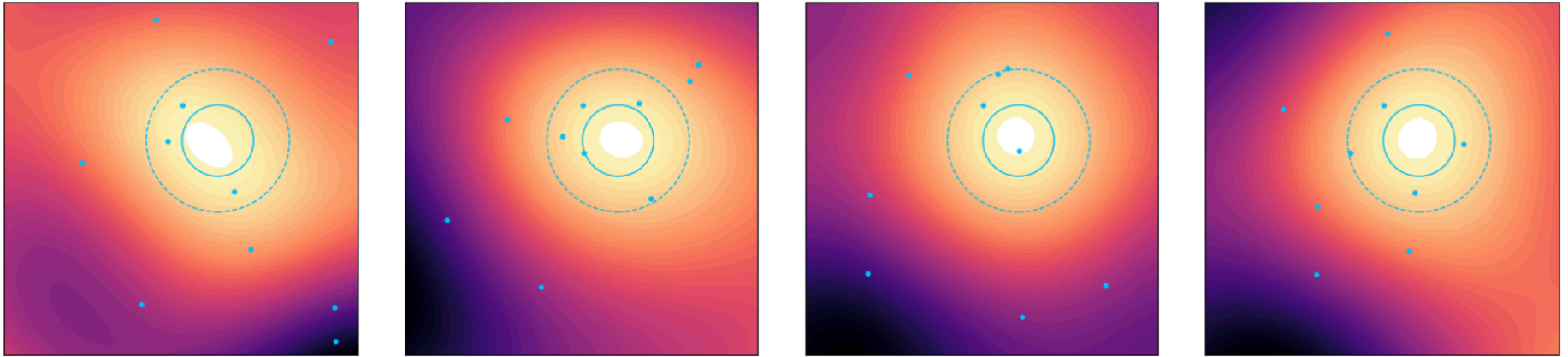
... its acquisition function has many local maxima



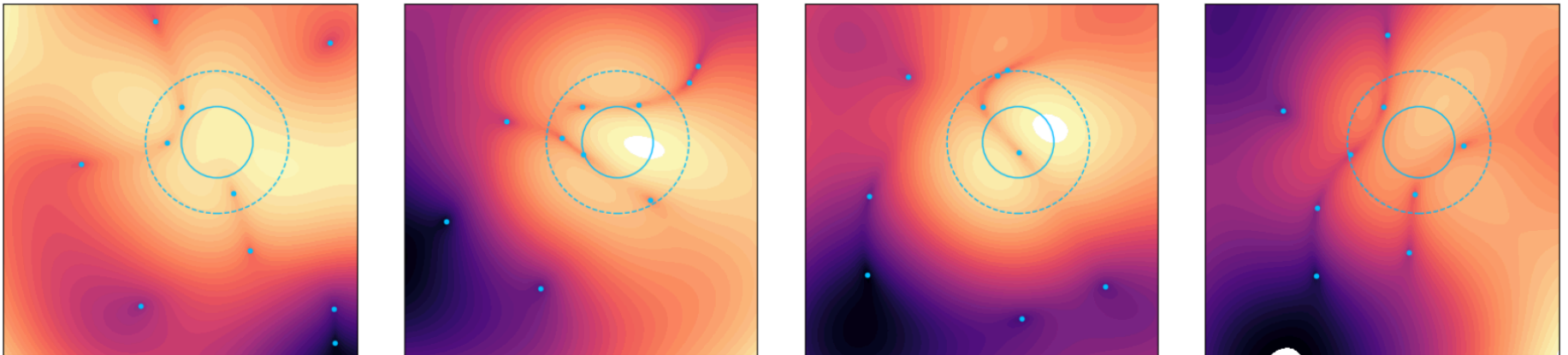
Active sampling – MC sampling!

Global maxima of **highly multi-modal** function $\rightarrow$ **Hard!**

Our contribution: we **MC-sample from here**



... and simply track the acquisition func. and keep its **approximate maxim[a]**



Active sampling – MC sampling! – II

MC algorithm of choice: **Nested sampling**

- Explores the full dynamic range (exploration)
- Convergence can be controlled
- Highly parallelisable

Currently interfaces for:

- PolyChord (way faster, especially with MPI)
- nessai
- UltraNest

Algorithm for surrogate-posterior inference

Modelling:

Choice of GP kernel, variable transformations...

Algorithm:

1. **Acquisition (active sampling)**: search for the next optimal locations.
2. **Evaluation of the true model** at the optimal locations.
3. **Re-fitting** of the GP model: new points and new optimal hyperparams.
4. **Convergence checks**

Profit!

Analytic, differentiable GP model of the posterior to do **inference** with.

Active sampling – parallelisation

Key for the **slow-posteriors**: **parallel** evaluations of the true posterior.

⇒ we need **simultaneous maxima of the acquisition function**.

$$\theta^{(1)}, \theta^{(2)}, \dots \Leftarrow \arg \max_{\theta^{(1)}, \theta^{(2)}, \dots} a(\theta)$$

Simultaneous optimisation is *very* expensive.

Problems with simply taking **multiple local maxima**:

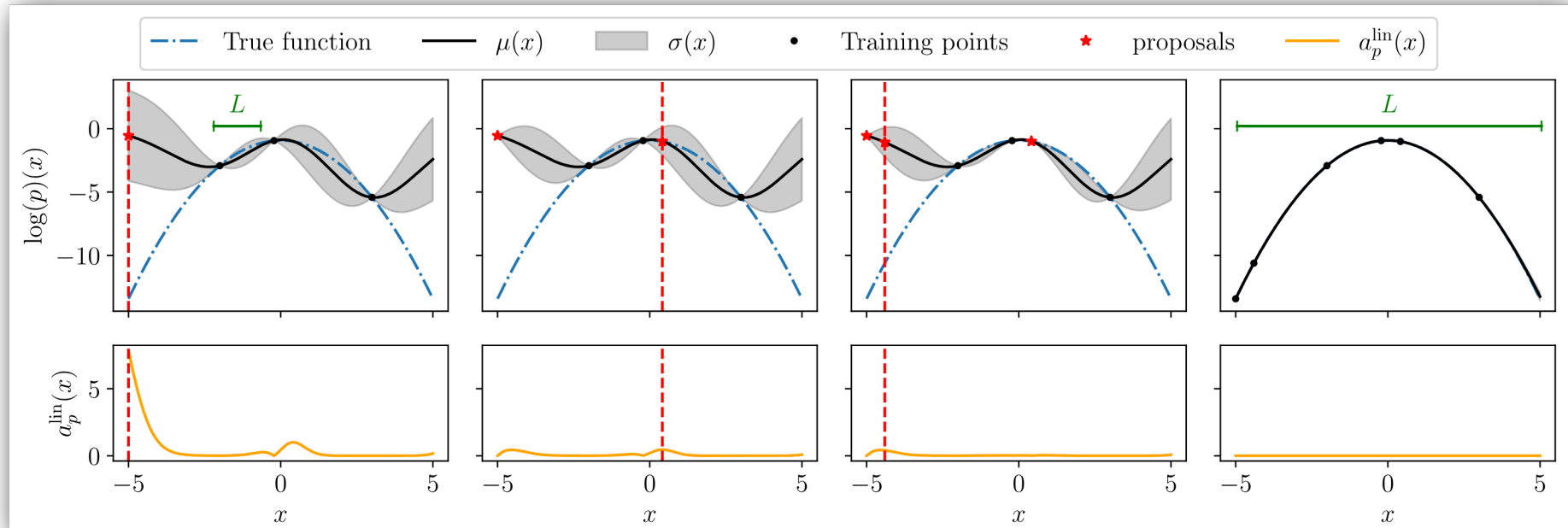
- a) Our maximiser would run into the same maximum more than once
- b) The information in one maximum may be redundant with another one

Active sampling – parallelisation (II)

A typical (but not unique) choice is "**Kriging believer**":

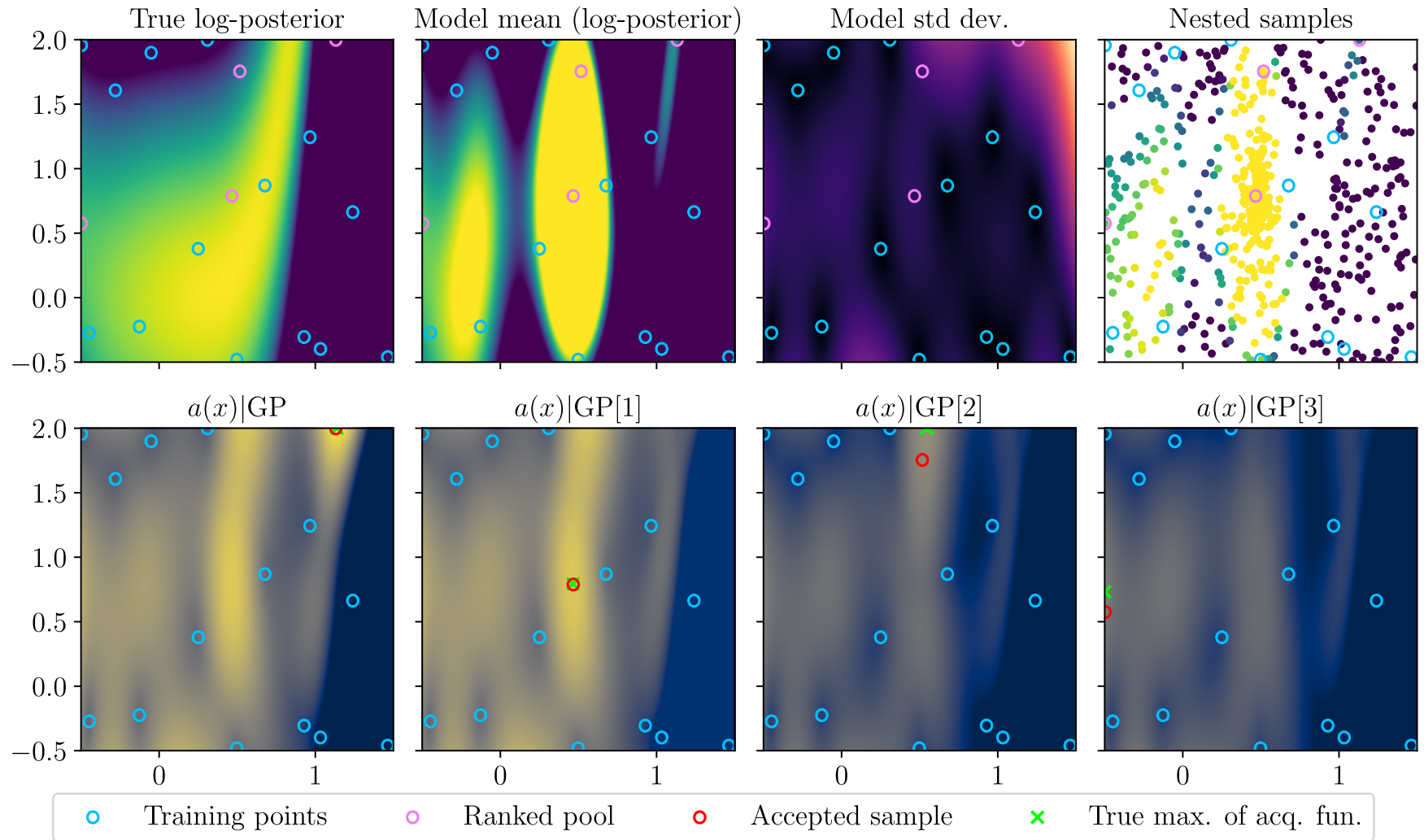
[find global maximum] $\rightarrow$ [add GP-predicted mean there] $\rightarrow$ [repeat]

Works well up to $n_{\text{parallel}} \approx d$.



NB: Adding a training point with the value of the mean GP does not modify the mean GP, but excludes points around the new one.

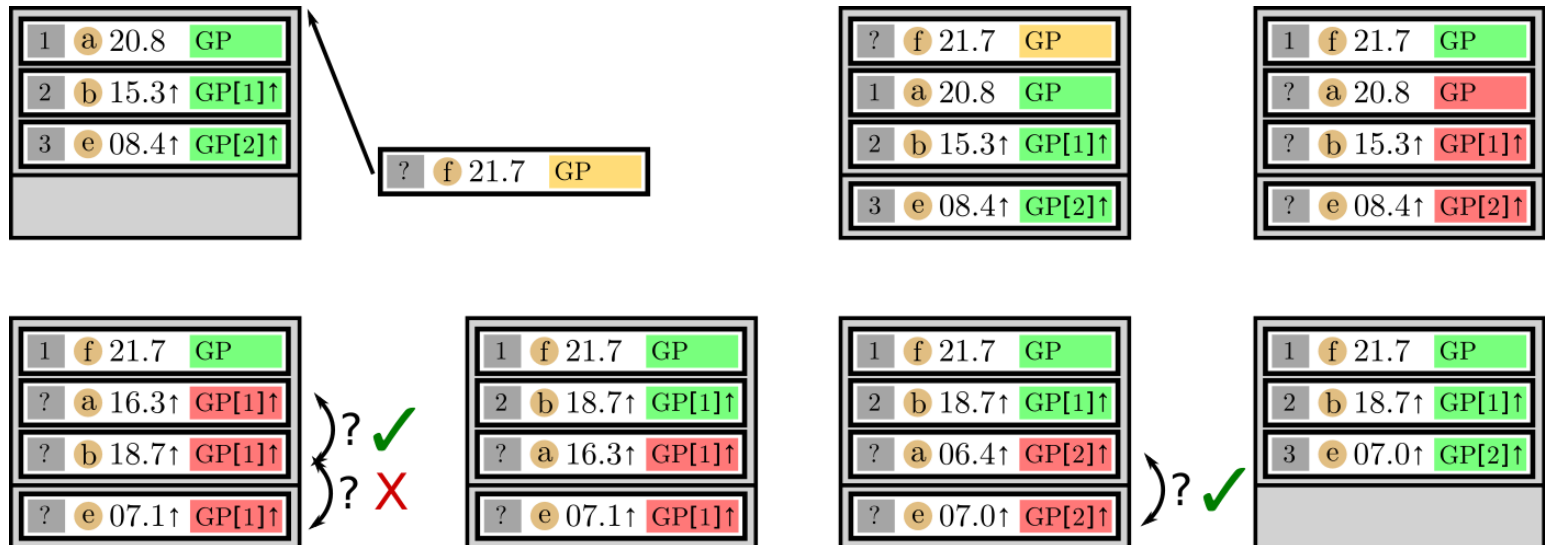
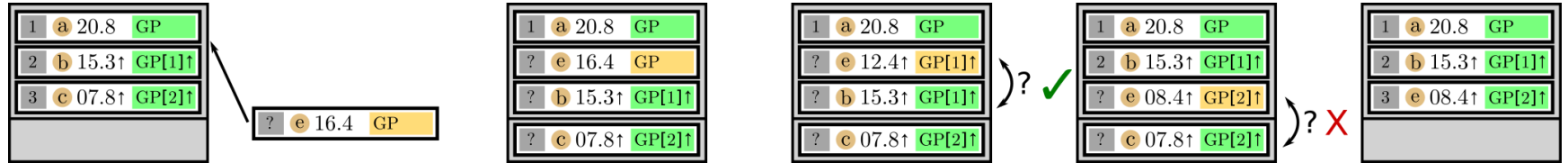
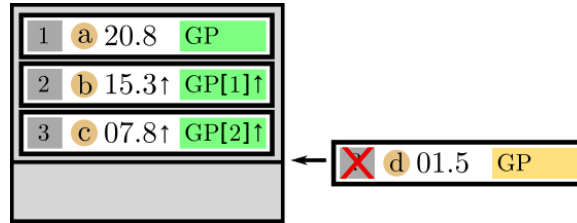
How does it work in practice?



Best points in the last row are **not optimised**: are selected from the nested sample!

Efficiently getting these chain-conditioned candidates requires some secret sauce...

Efficient, parallelisable conditional ranking



Algorithm for surrogate-posterior inference

Modelling:

Choice of GP kernel, variable transformations...

Algorithm:

1. **Acquisition (active sampling)**: search for the next optimal locations.
2. **Evaluation of the true model** at the optimal locations.
3. **Re-fitting** of the GP model: new points and new optimal hyperparams.
4. **Convergence checks**

Profit!

Analytic, differentiable GP model of the posterior to do **inference** with.

Model fitting

Once the **true posterior** is evaluated at the new optimal locations, we need to **re-fit** the GP model.

Re-computing the model (kernel matrix) has a cost $\mathcal{O}(N^3)$.

- If hyperparameters are kept constant, this only needs to be done **once**.
- If they are to be re-fitted, there are many $\mathcal{O}(N^3)$ evaluations.

This is way costlier than active sampling, which only requires GP model predictions that scale as $\mathcal{O}(N^2)$.

Usual strategy: re-fit hyperparams only every m active-sampling steps.

(After a while parameters don't change, in any case.)

Algorithm for surrogate-posterior inference

Modelling:

Choice of GP kernel, variable transformations...

Algorithm:

1. **Acquisition (active sampling)**: search for the next optimal locations.
2. **Evaluation of the true model** at the optimal locations.
3. **Re-fitting** of the GP model: new points and new optimal hyperparams.
4. **Convergence checks**

Profit!

Analytic, differentiable GP model of the posterior to do **inference** with.

Useful convergence criteria

Assessing convergence: look at the **stability** over iterations of a **global measure**; e.g. the **Bayesian evidence** for the case in which we model probability density functions. But global measures are **expensive**!

⇒ Our approach:

- **Local** difference between new evaluations and their expected value
 $|\mu(x^{(i)}) - \log p(x^{(i)})| < \epsilon$ (where ϵ is **dynamical**!)
- **Global** (but cheap) KL divergence from NS used in active learning.

Necessary but not sufficient additional criterion:

- Is the NS exploring some far out region, or are we happy with the one induced by the training set? (protects from **overshoots**)

⇒ **good** failure states: mode not found / not fully explored.

Very rare completely-wrong-but-confident answer.

Convergence criterion, revisited

We mentioned the possibility of using cheap local criteria, such as

$$\left| \mu(x^{(i)}) - \log p(x^{(i)}) \right| < \epsilon$$

Recalling the dependence of the **dynamic range** of $\log p$ with dimensionality, it should be obvious that this criterion becomes **stricter** as dimensionality grows!

We should relax it when the dynamic range of interest is larger (larger d).

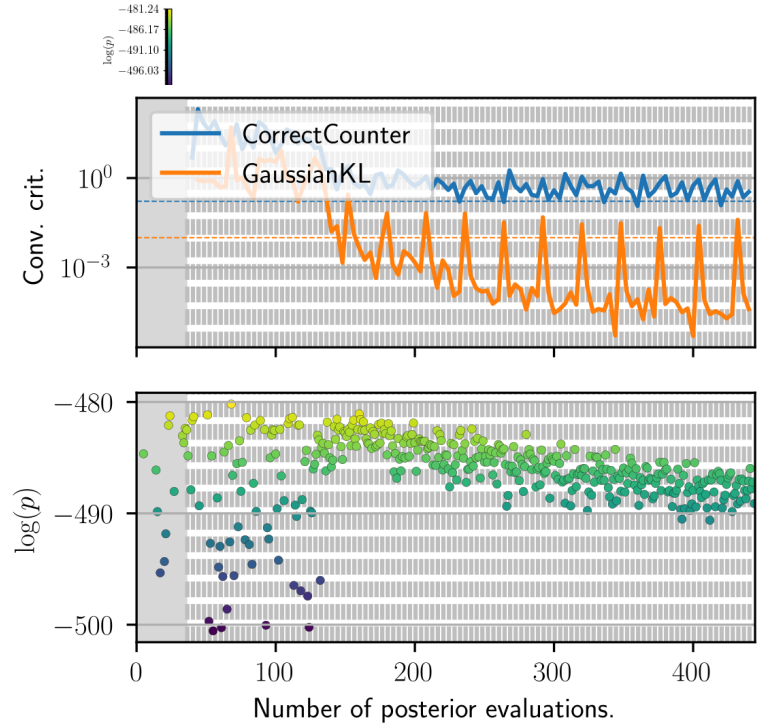
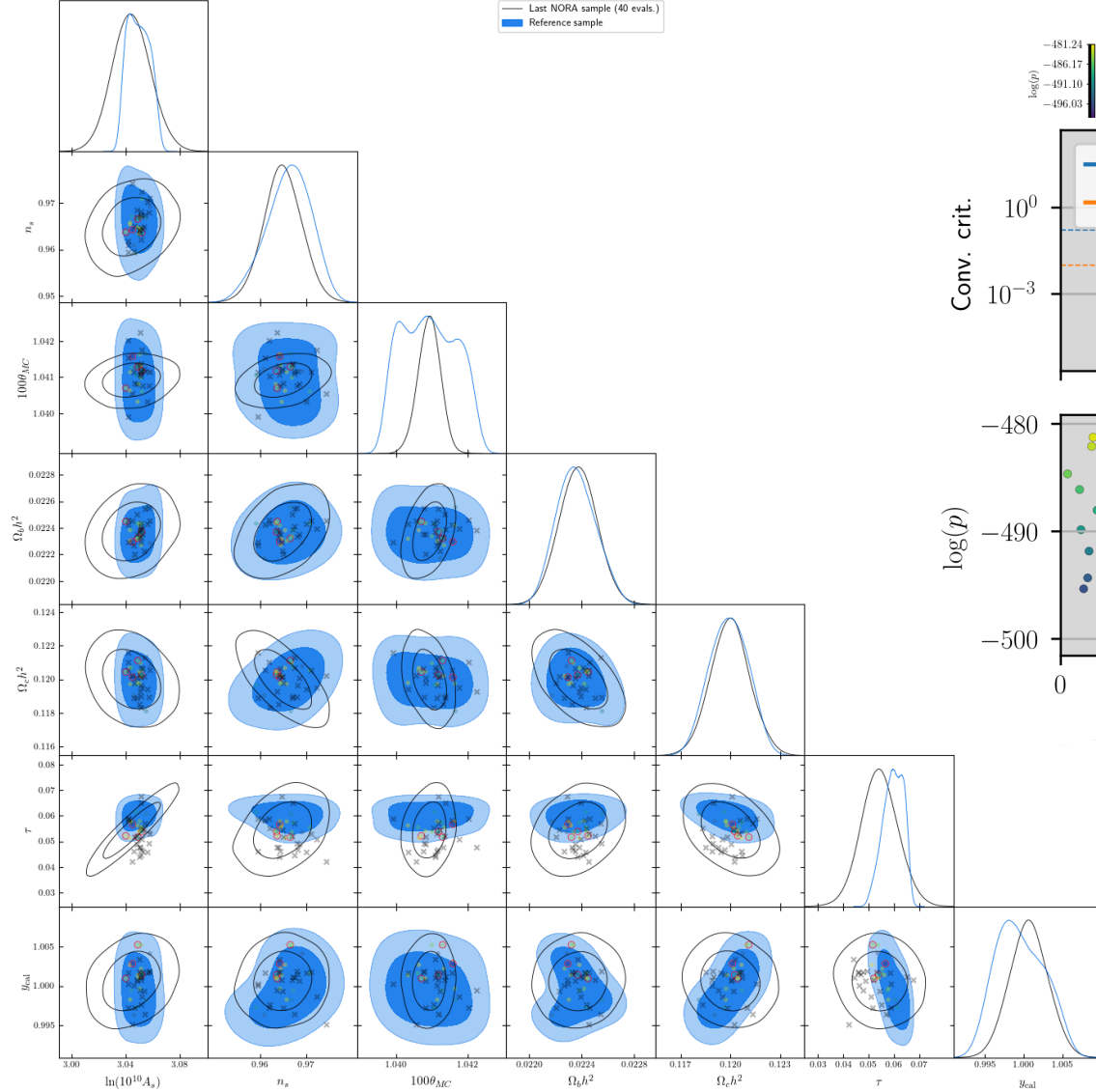
We set ϵ to a small fraction of the $\Delta \log p(68\%)$ in each dimensionality!

Cheap and reliable convergence without further fine-tuning!

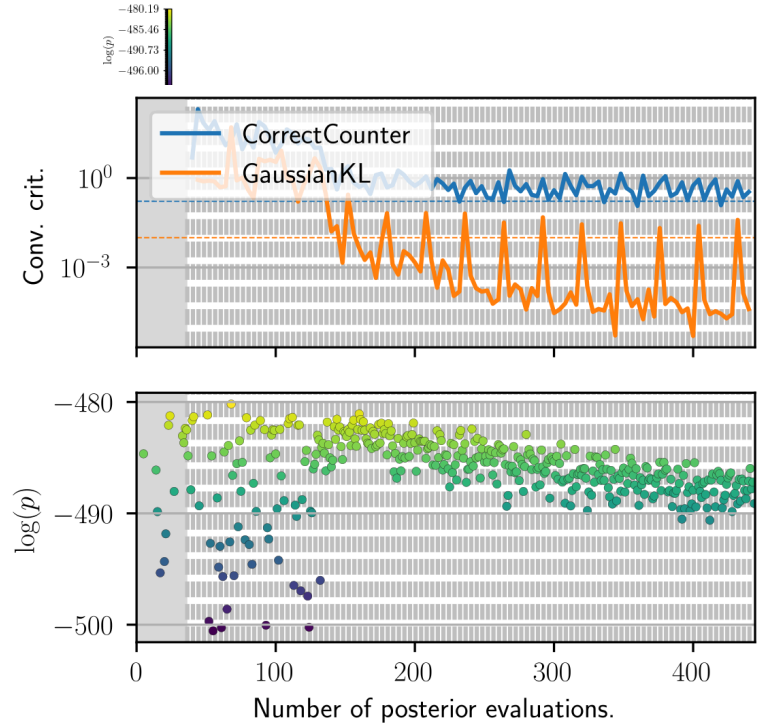
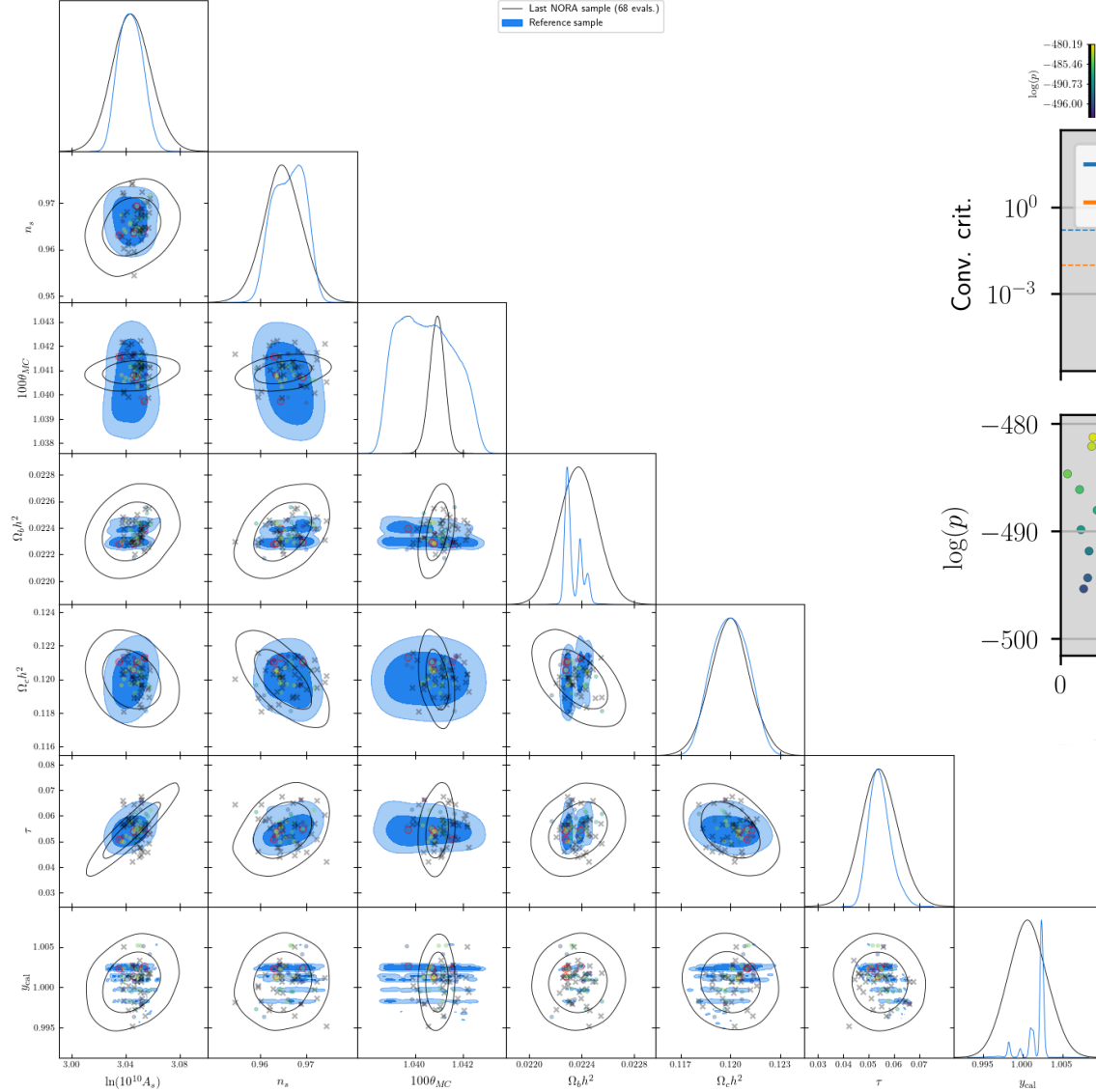
(In practice we have another term in the RHS, $|\mu(x^{(i)}) - y_{\max}| \cdot \epsilon_{\text{rel}}$, with no dim. scaling)

Examples

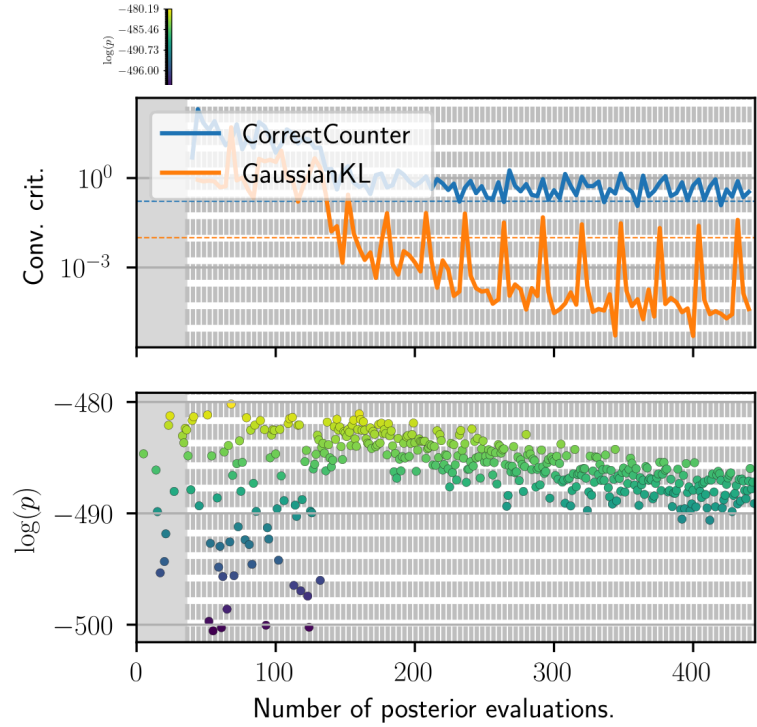
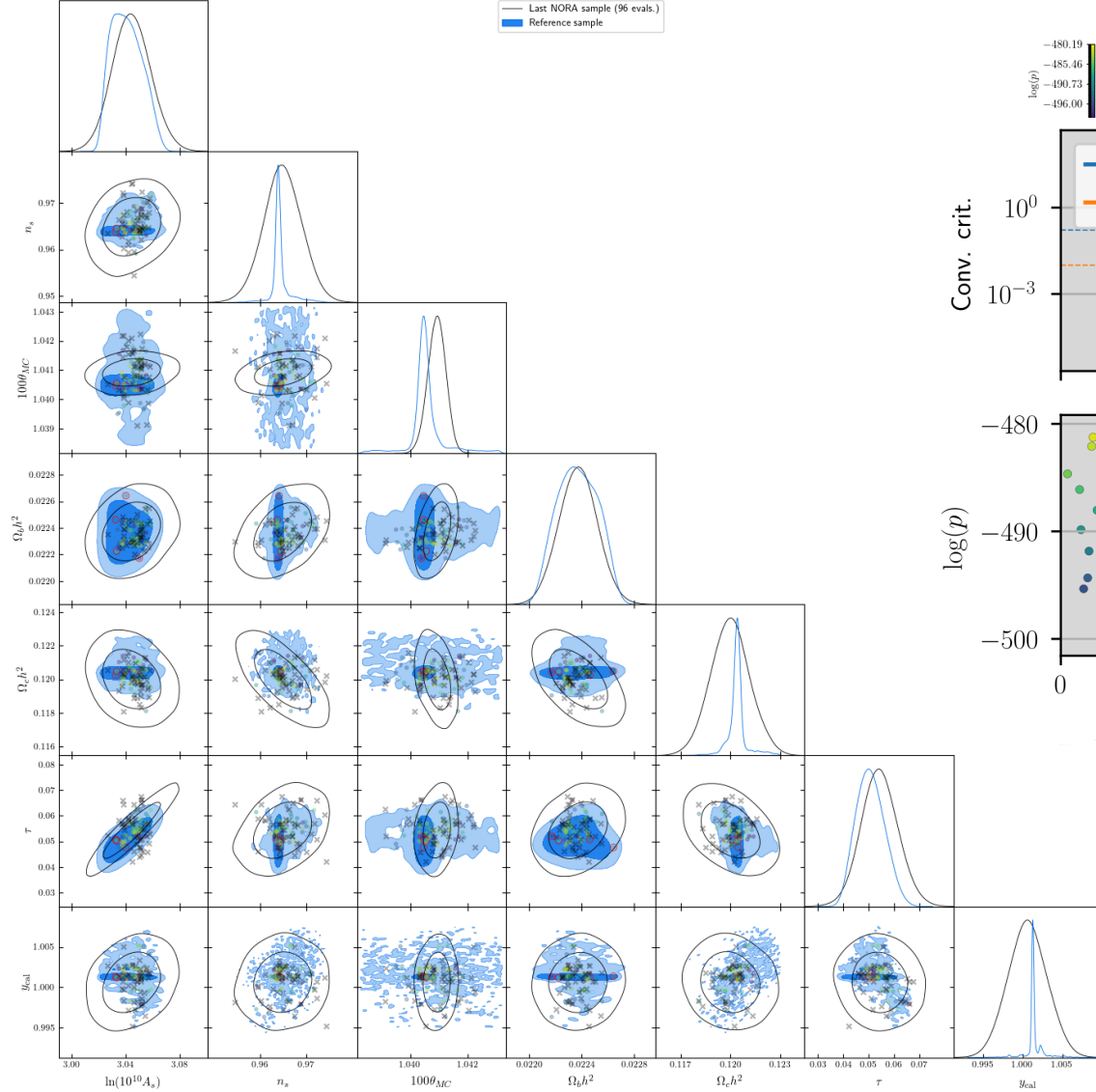
Λ CDM with Planck 2018 Lite: 6 + 1 parameters



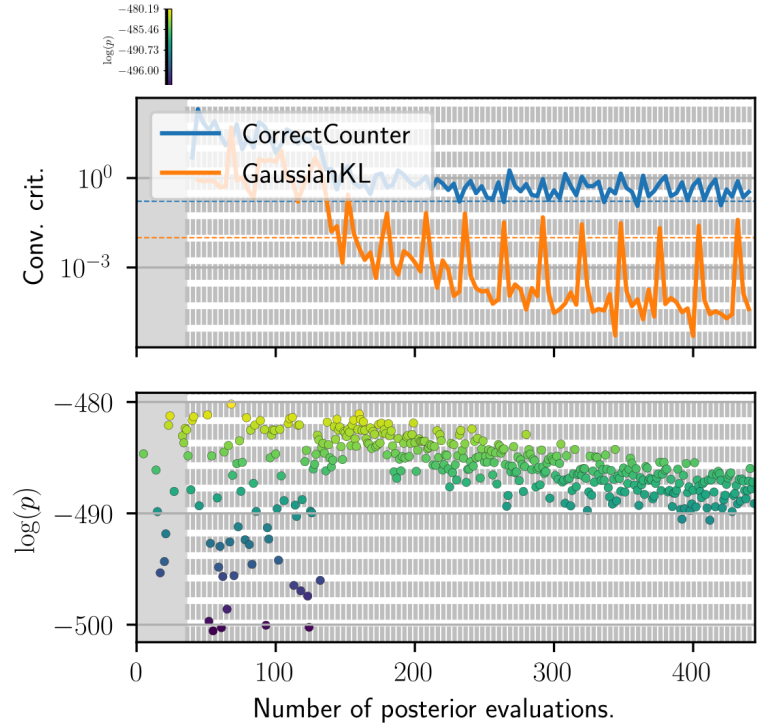
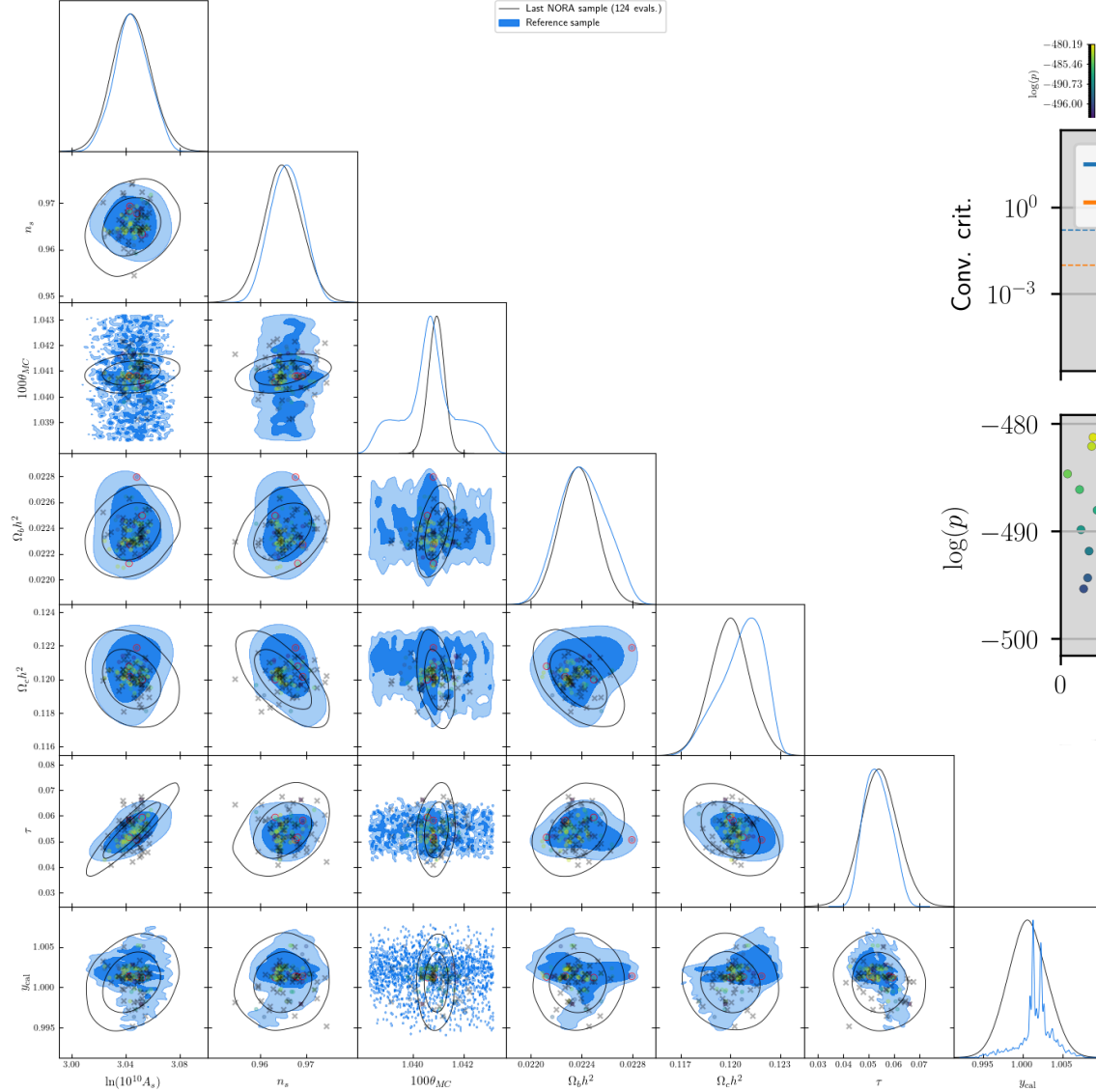
Λ CDM with Planck 2018 Lite: 6 + 1 parameters



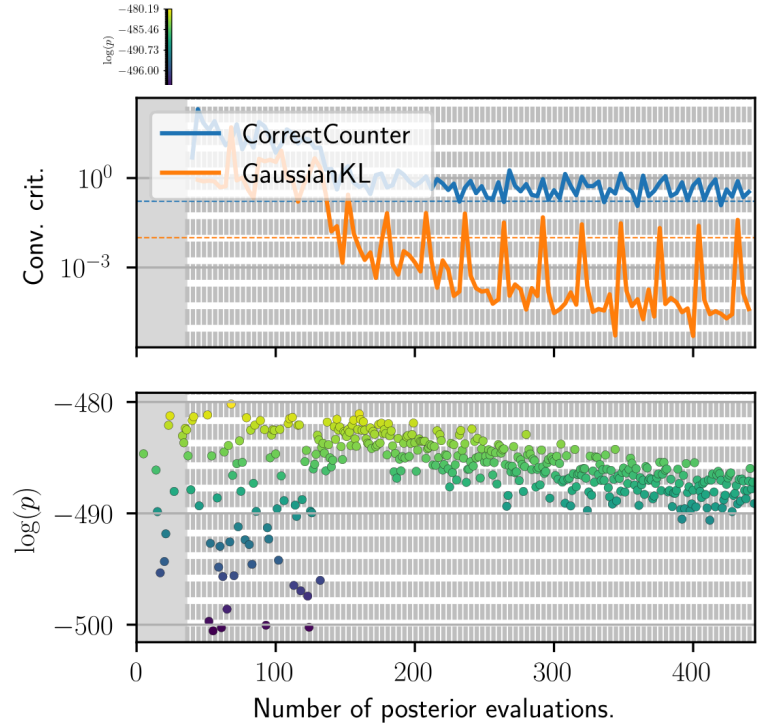
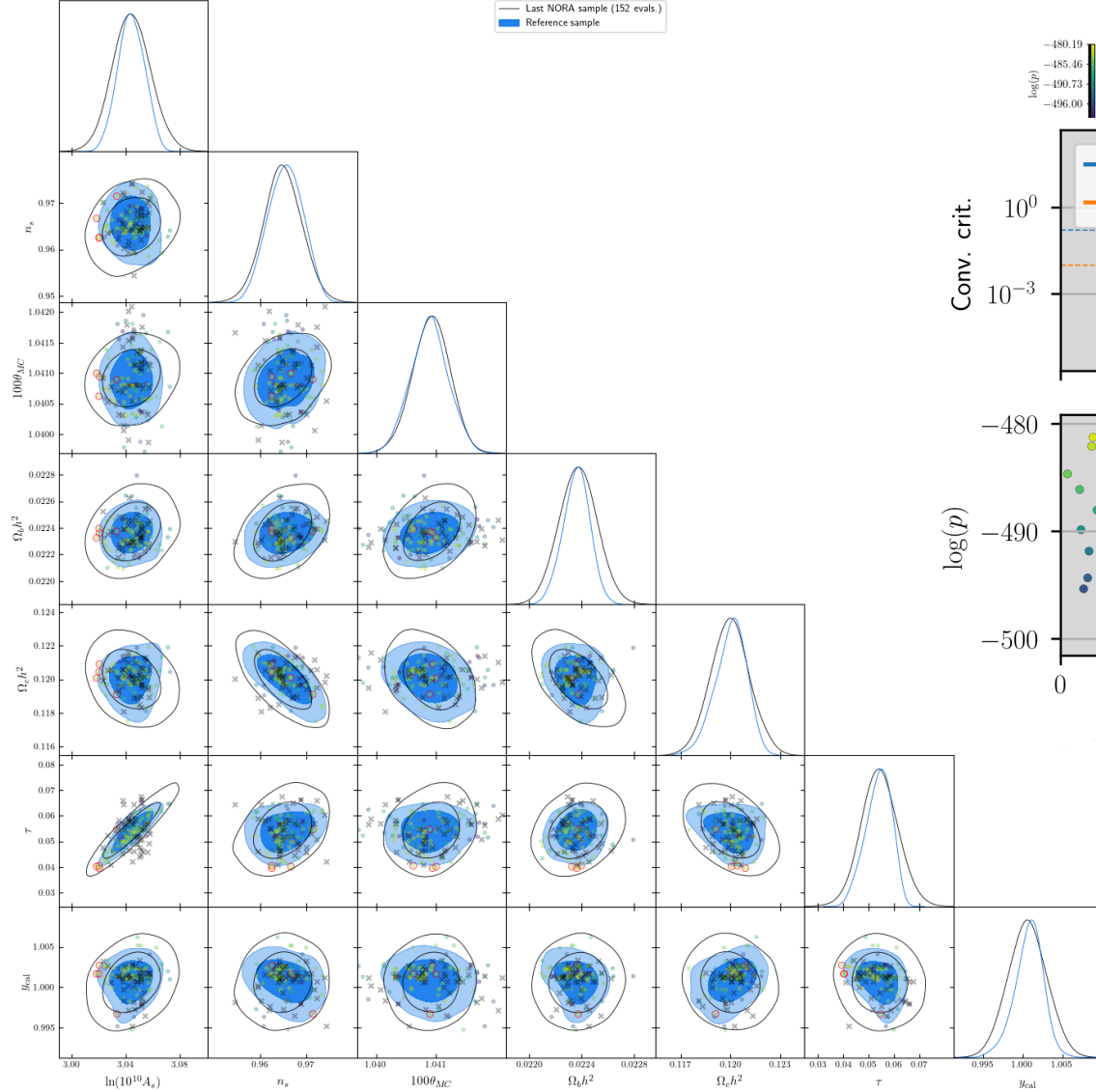
Λ CDM with Planck 2018 Lite: 6 + 1 parameters



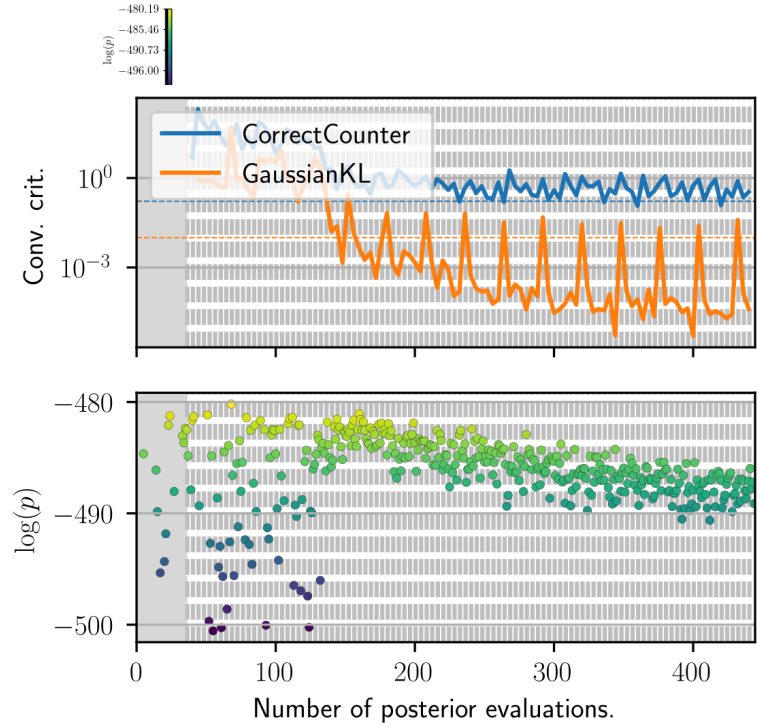
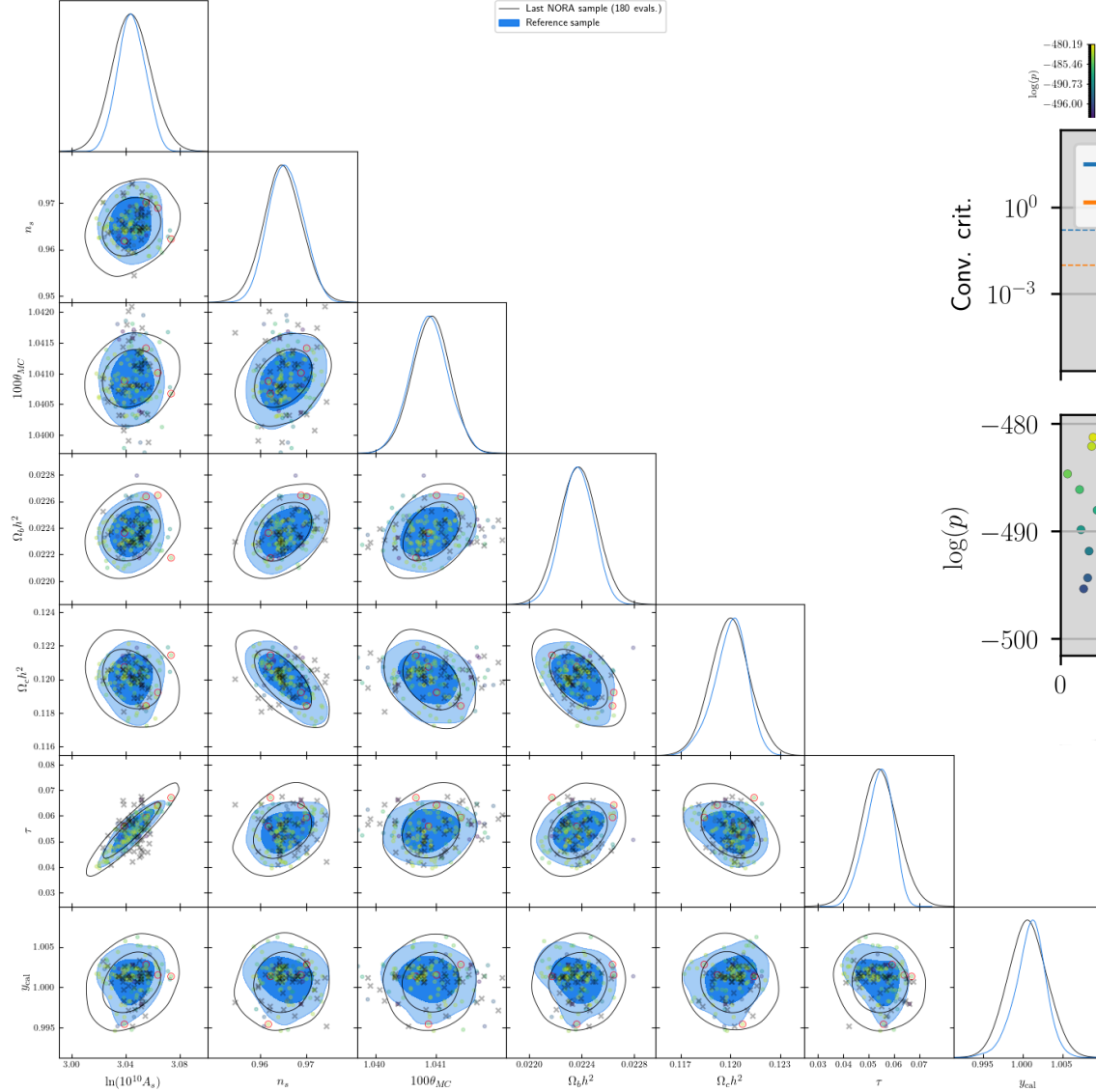
Λ CDM with Planck 2018 Lite: 6 + 1 parameters



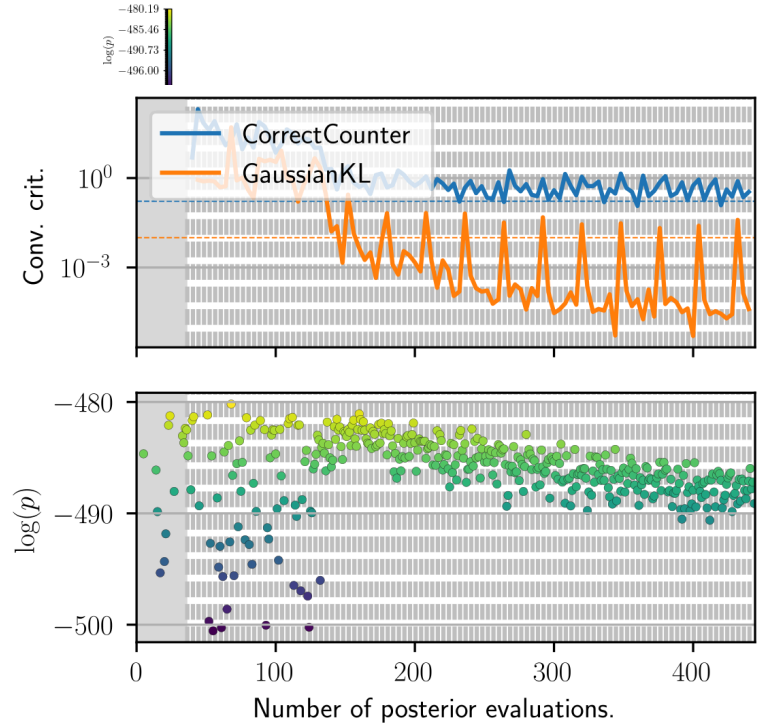
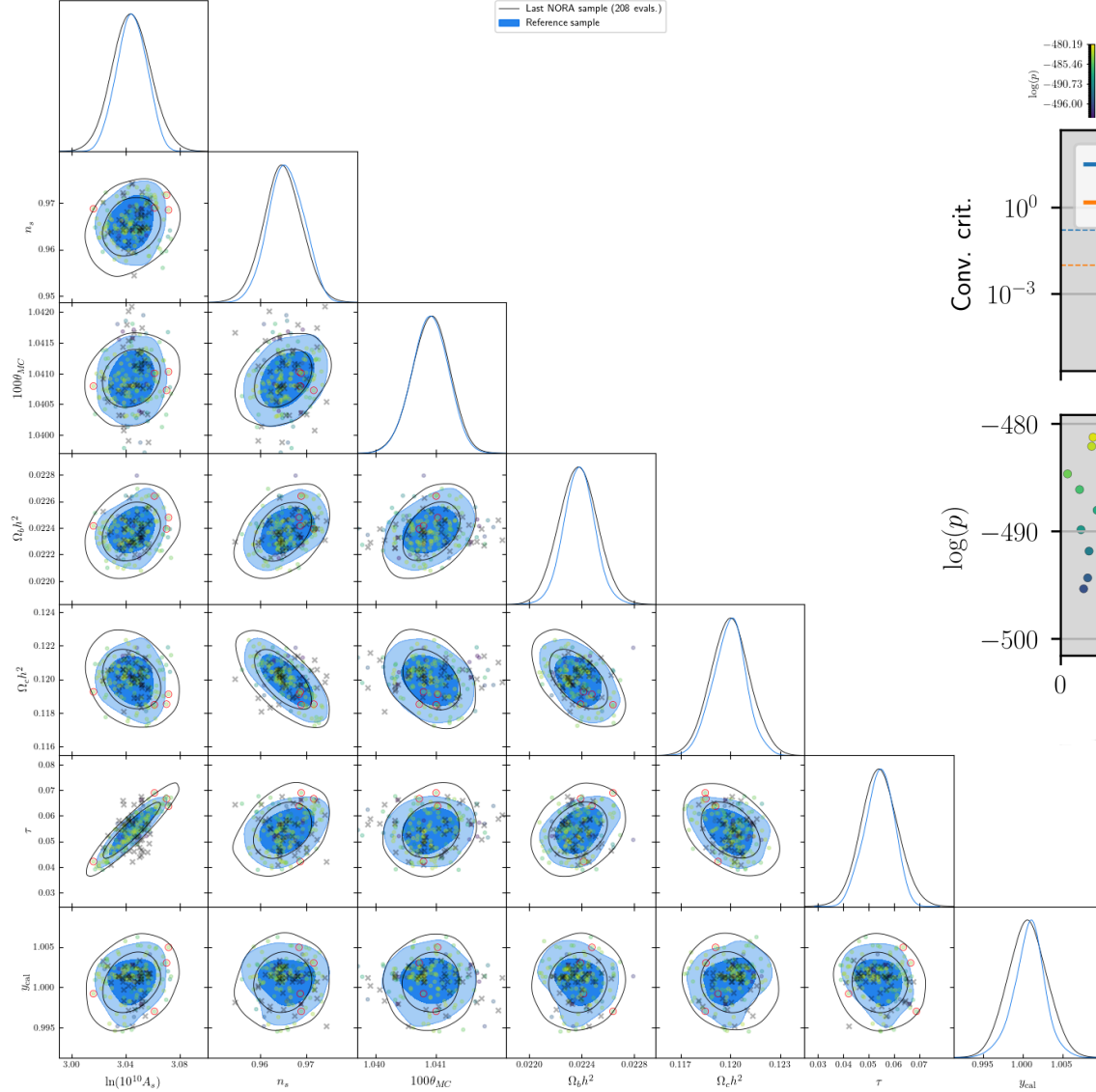
Λ CDM with Planck 2018 Lite: 6 + 1 parameters



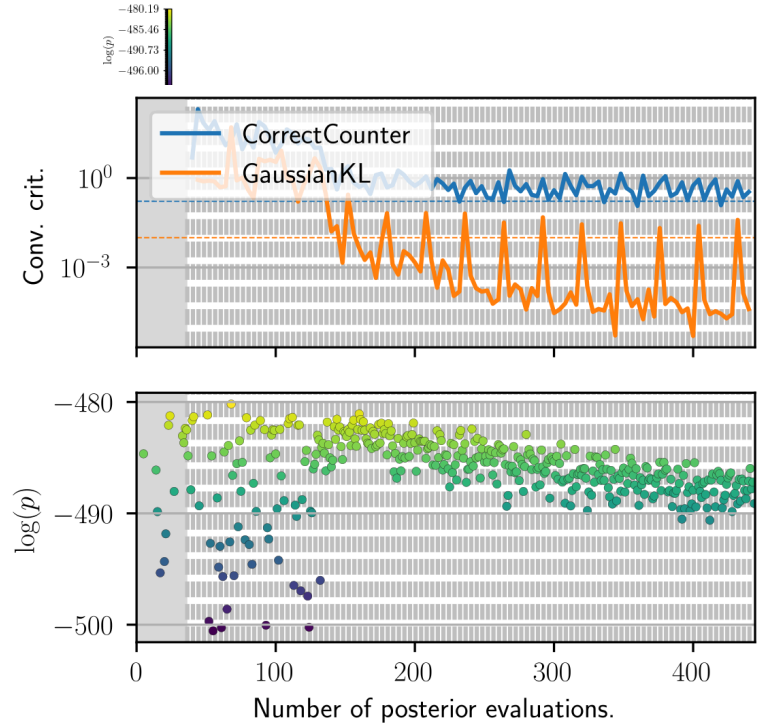
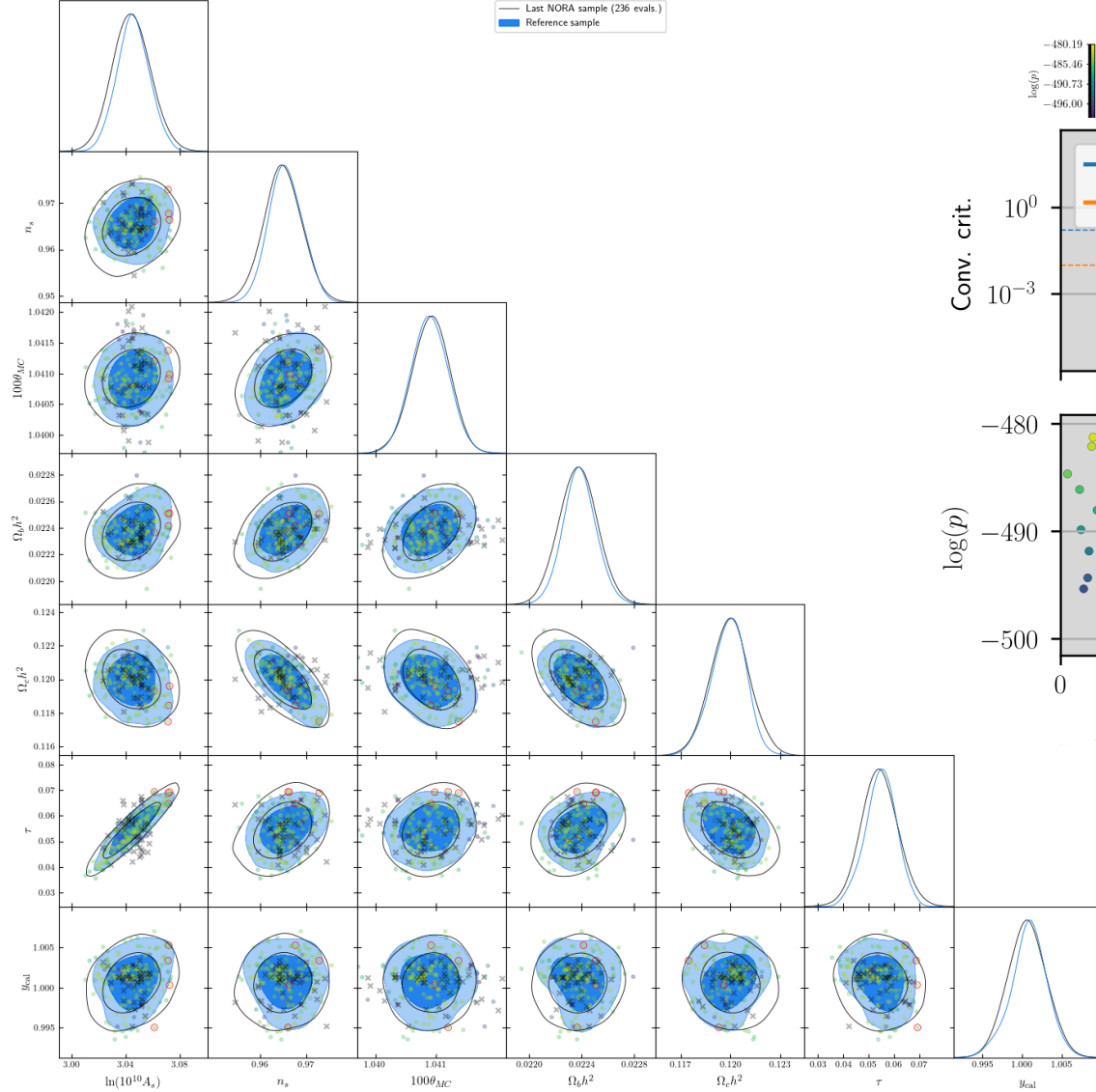
Λ CDM with Planck 2018 Lite: 6 + 1 parameters



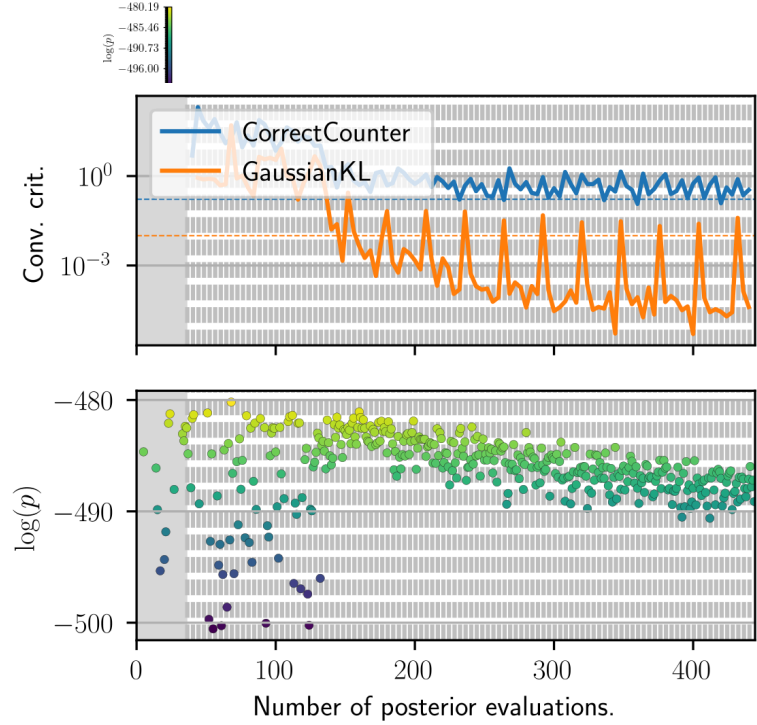
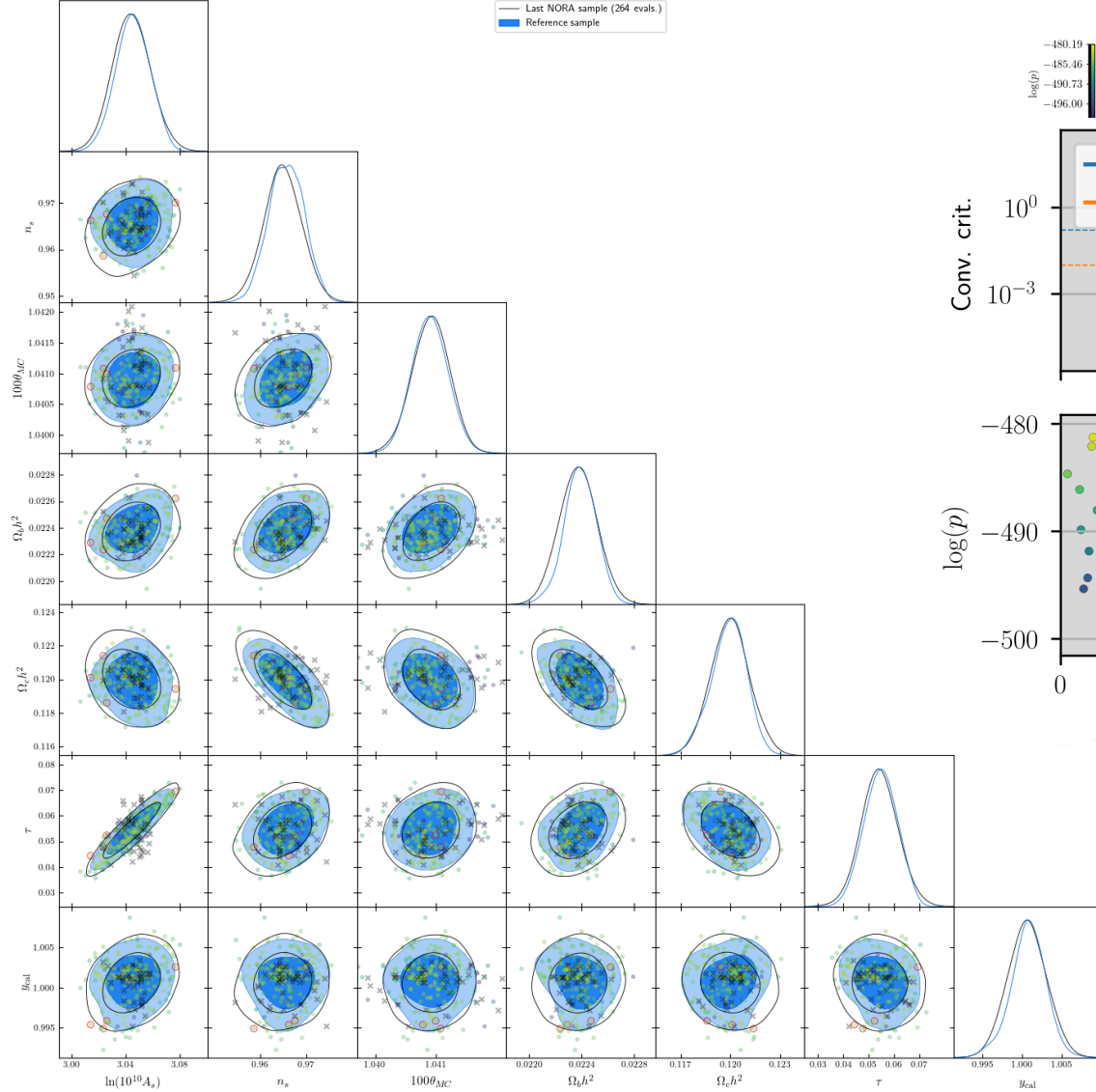
Λ CDM with Planck 2018 Lite: 6 + 1 parameters



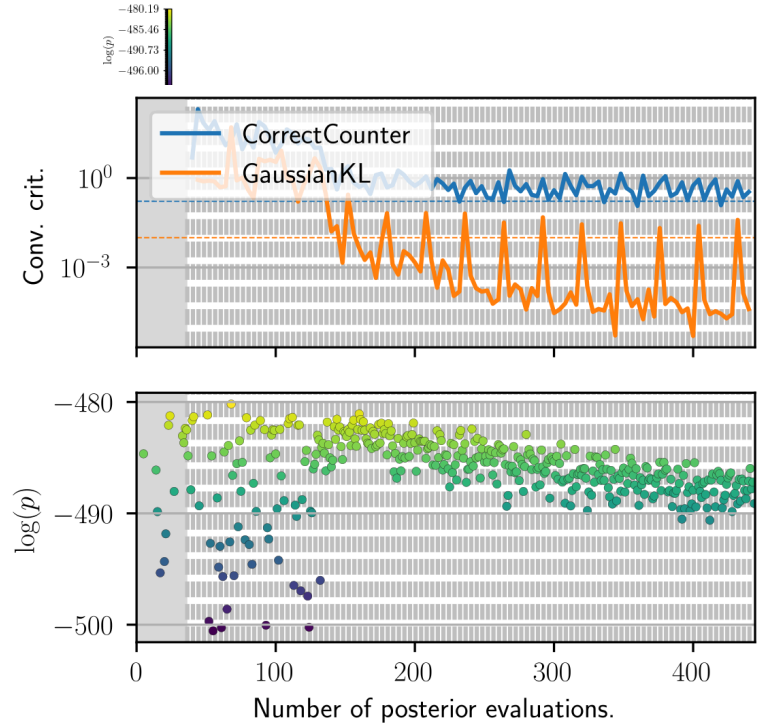
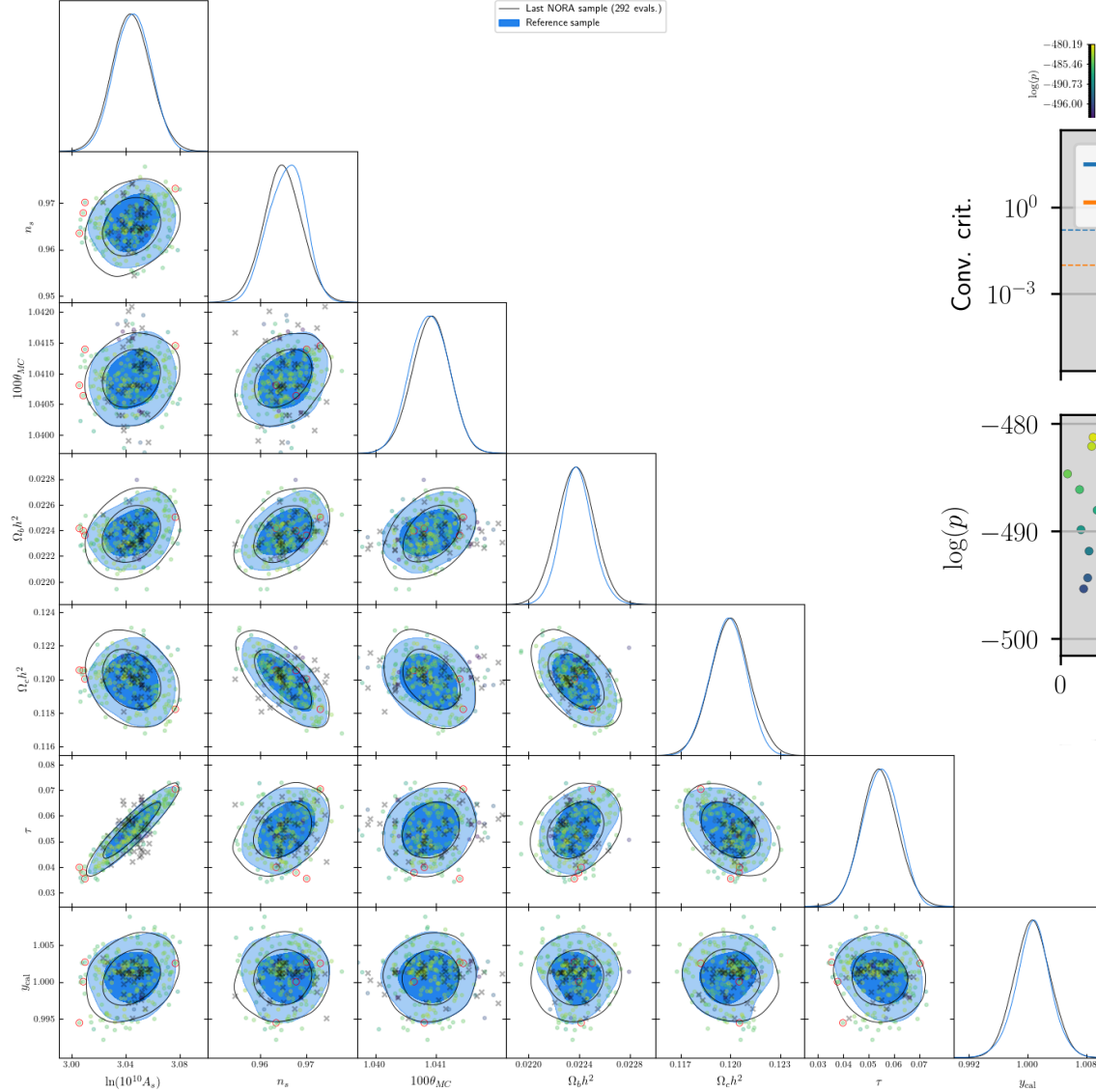
Λ CDM with Planck 2018 Lite: 6 + 1 parameters



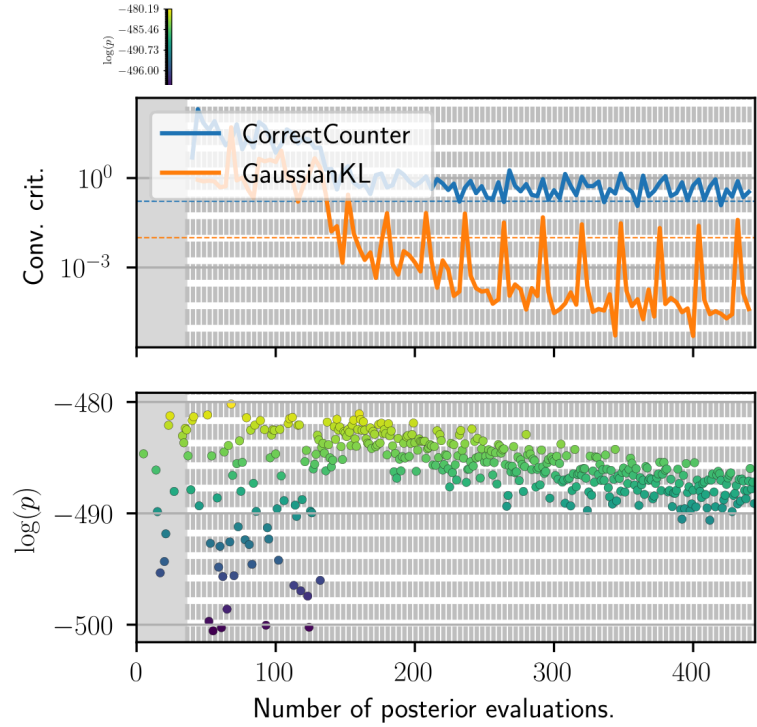
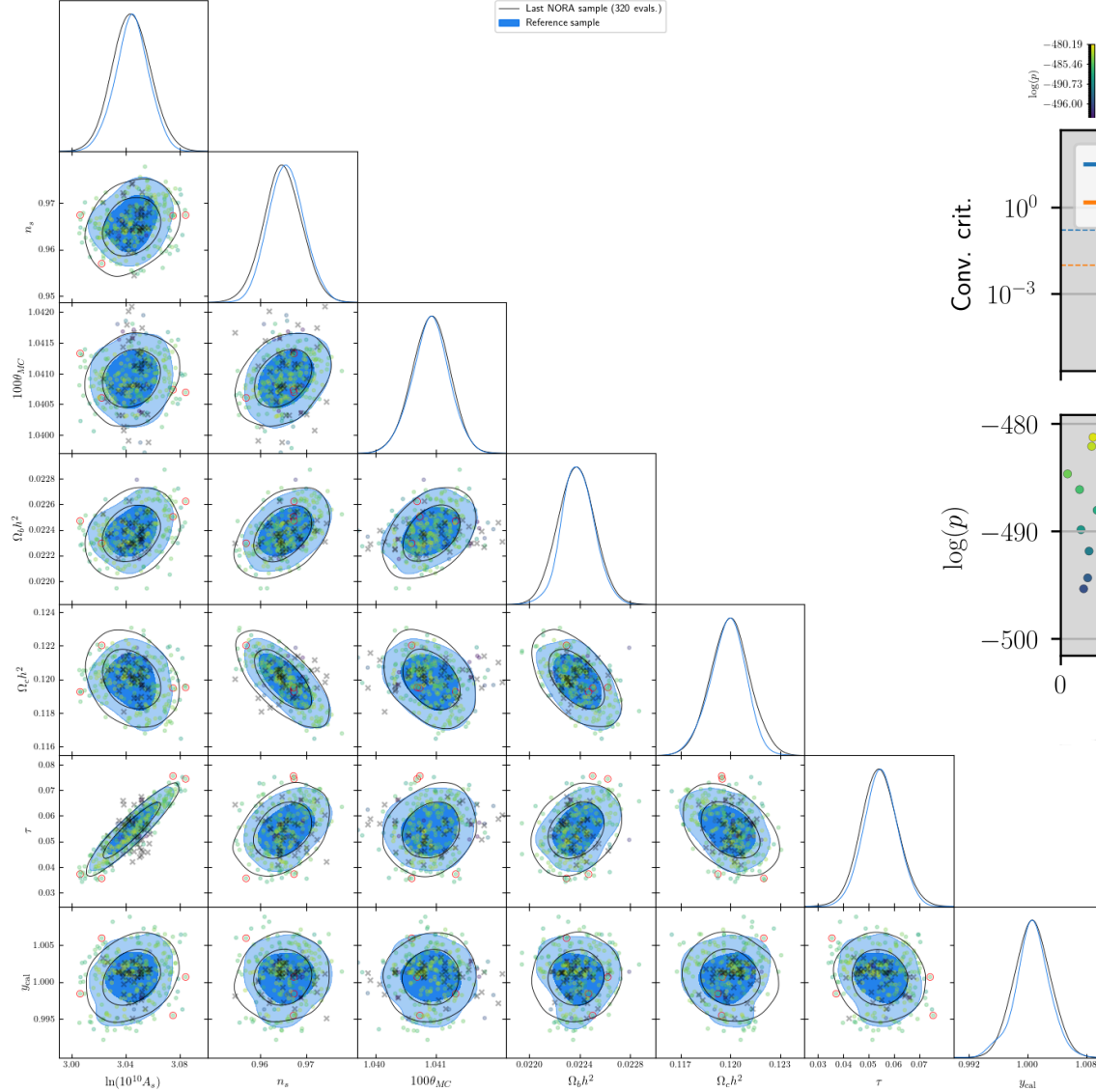
Λ CDM with Planck 2018 Lite: 6 + 1 parameters



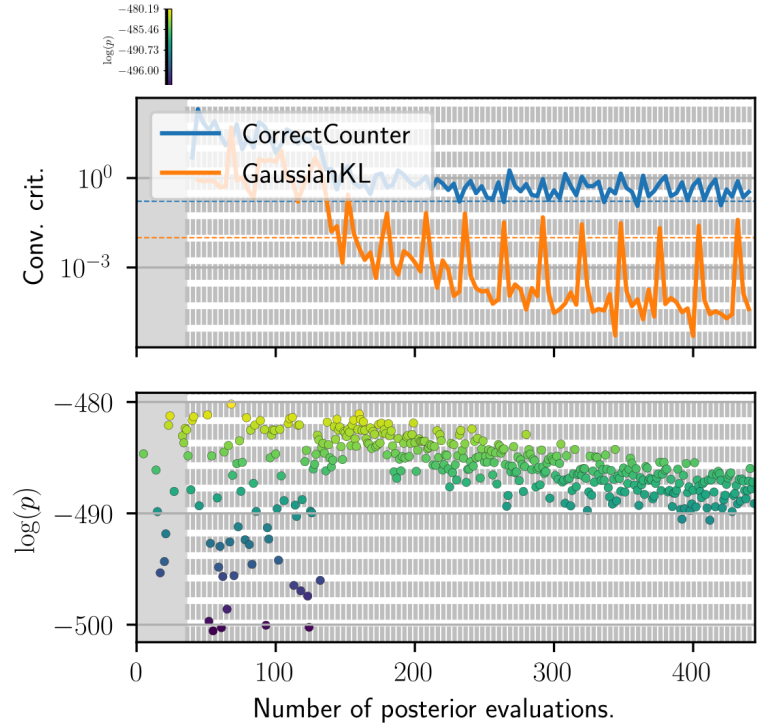
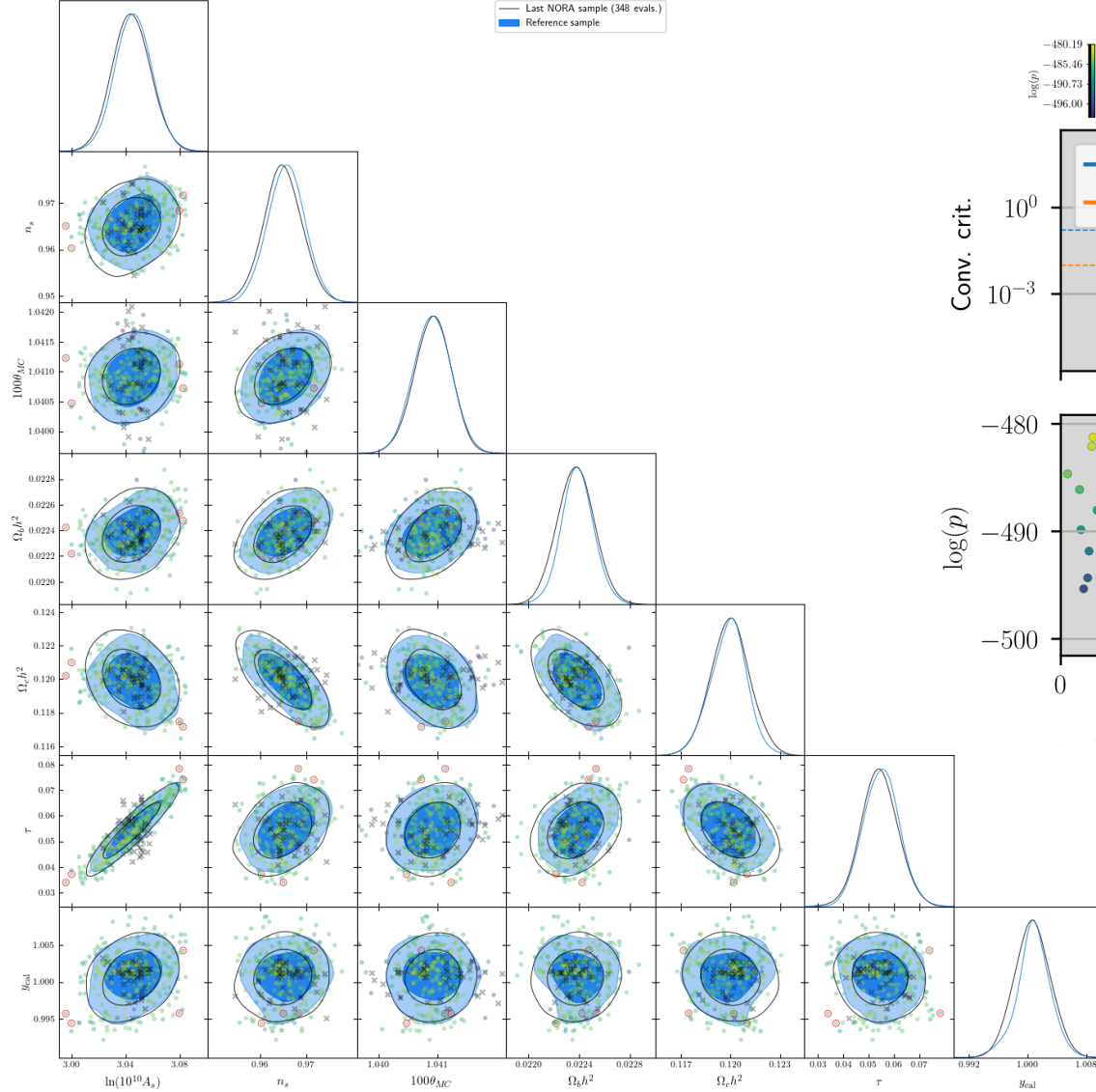
Λ CDM with Planck 2018 Lite: 6 + 1 parameters



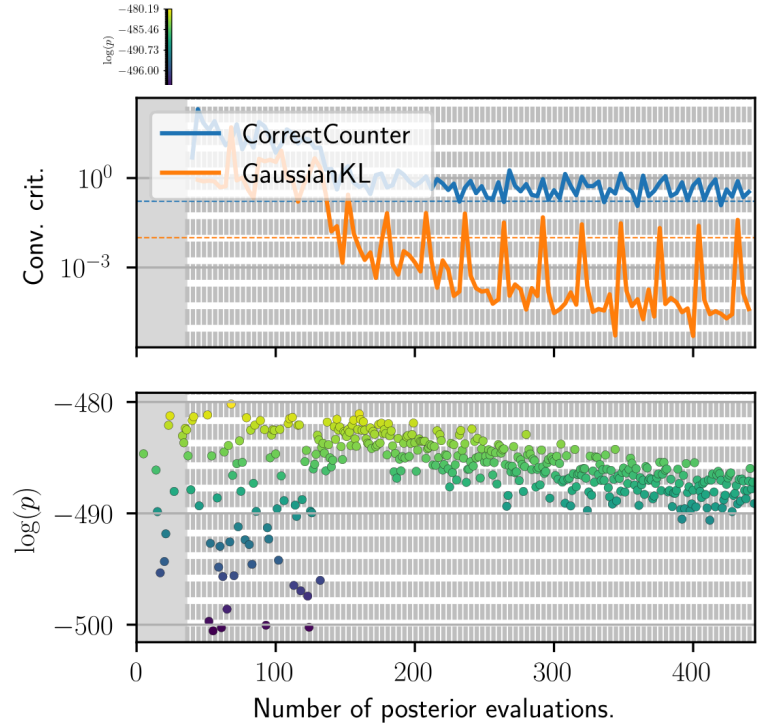
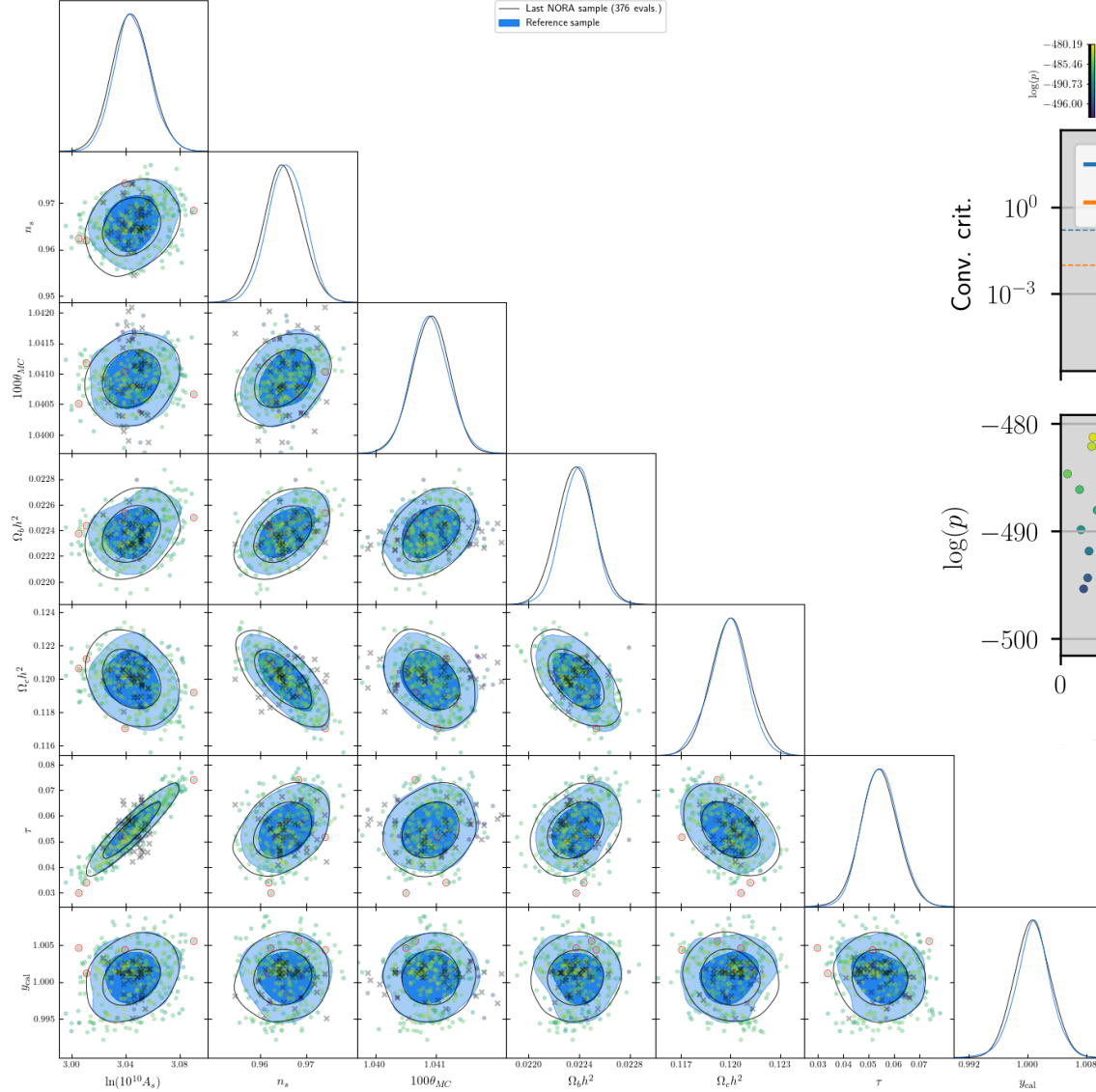
Λ CDM with Planck 2018 Lite: 6 + 1 parameters



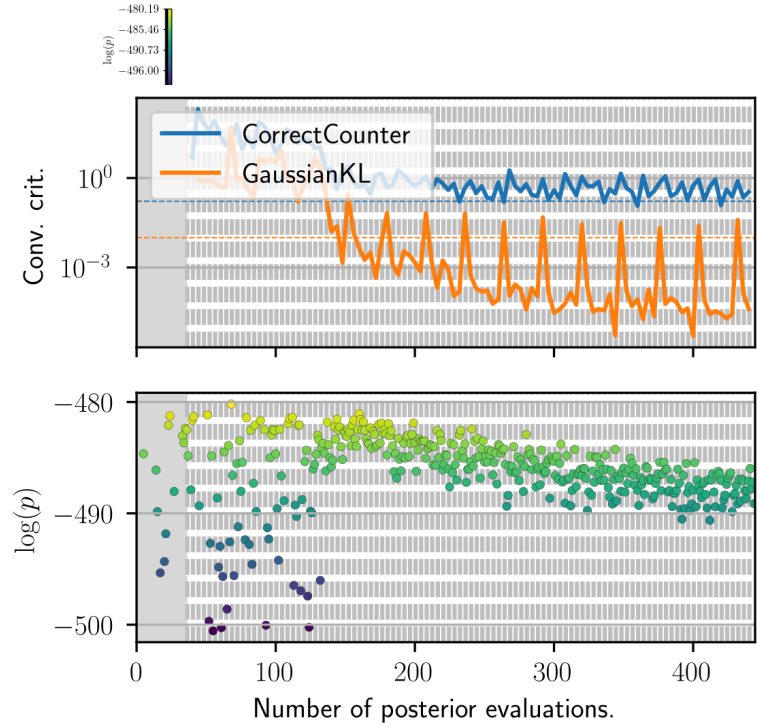
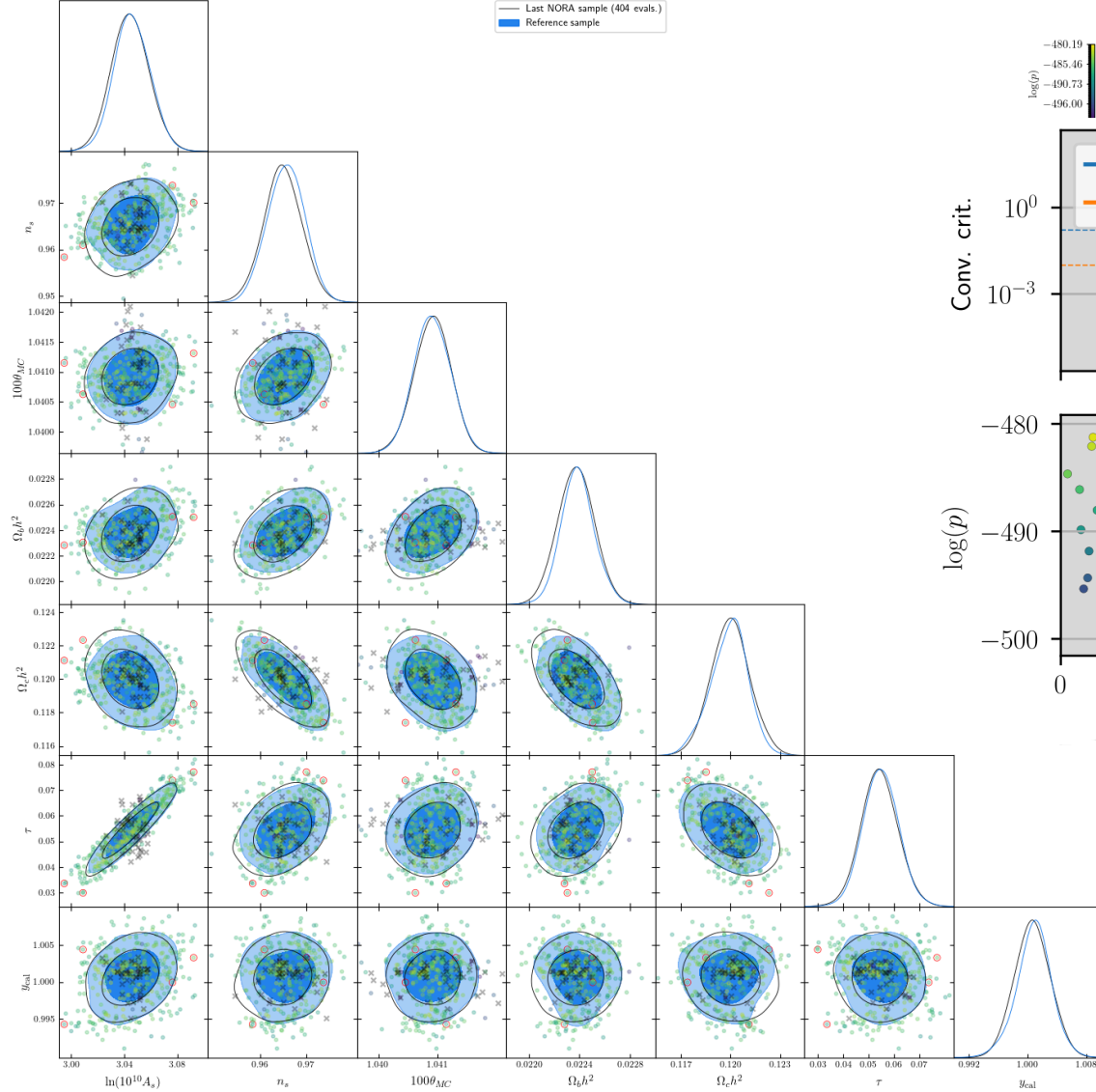
Λ CDM with Planck 2018 Lite: 6 + 1 parameters



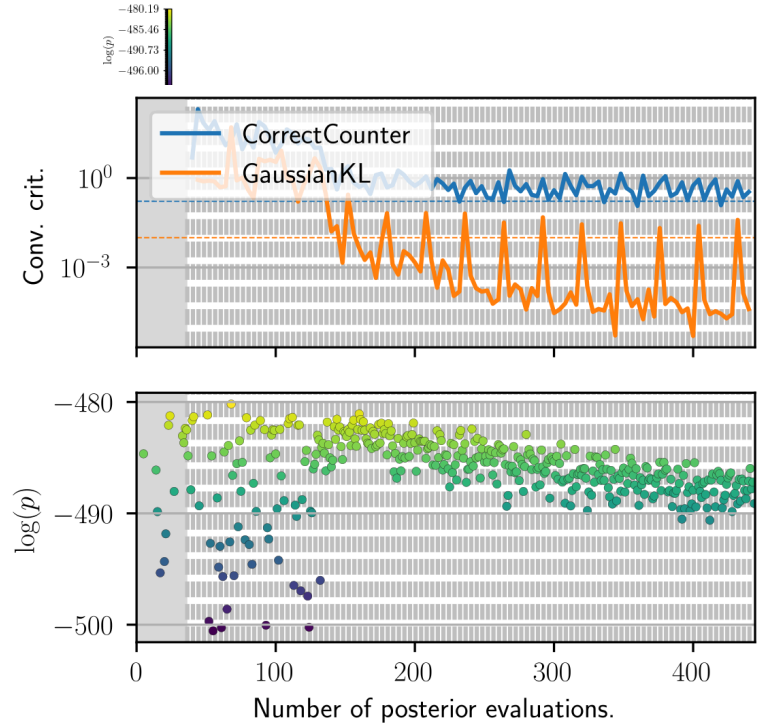
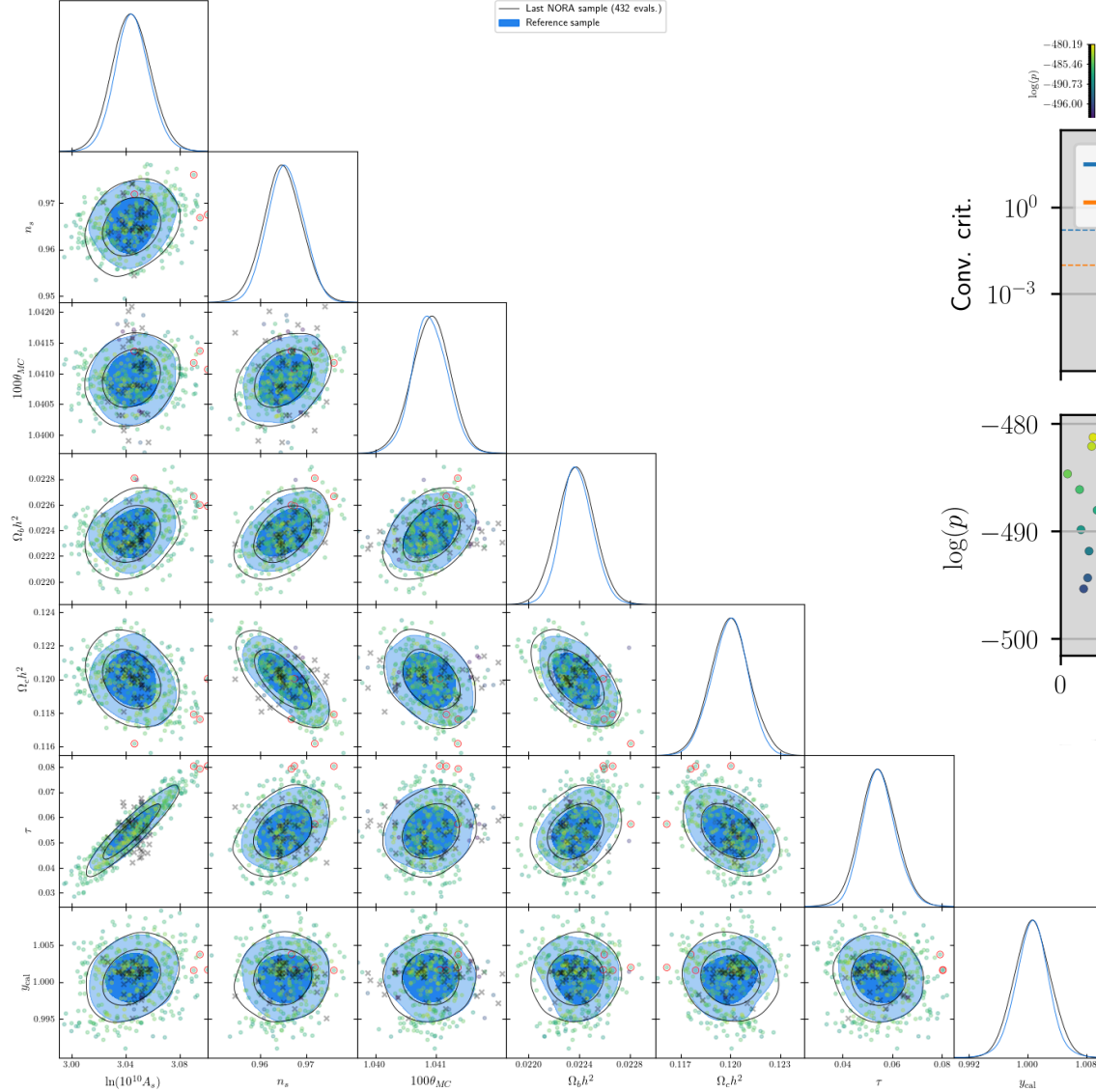
Λ CDM with Planck 2018 Lite: 6 + 1 parameters



Λ CDM with Planck 2018 Lite: 6 + 1 parameters

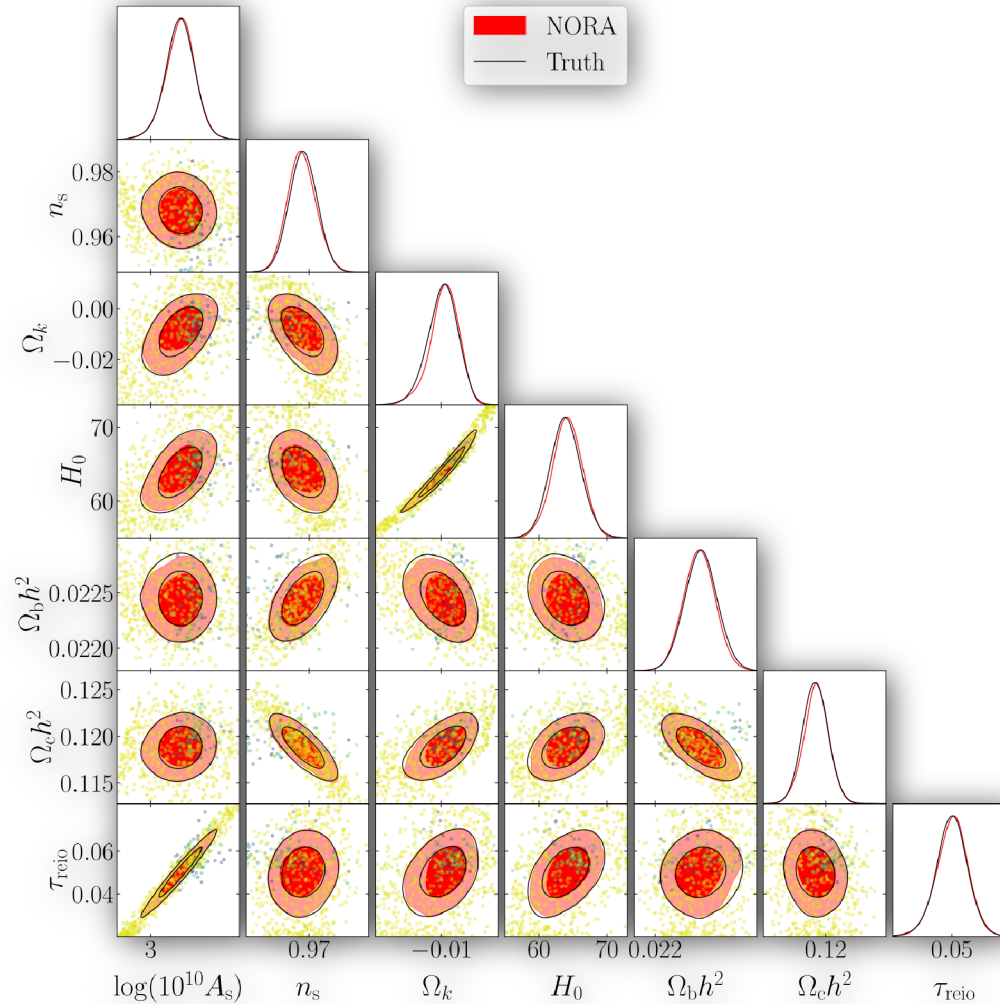


Λ CDM with Planck 2018 Lite: 6 + 1 parameters



More Cosmology – curved degeneracy

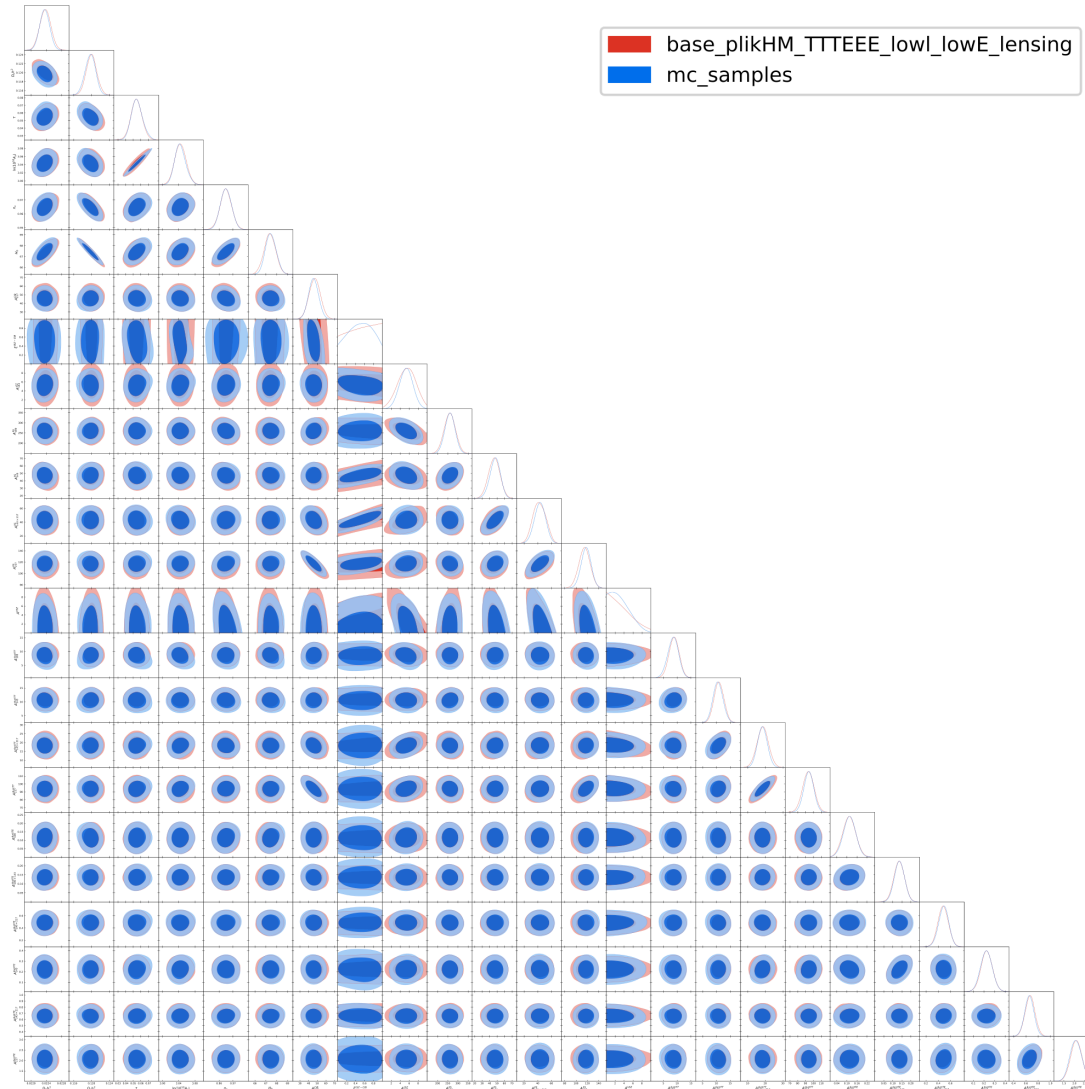
7-parameter Λ CDM+ Ω_k with Planck Lite 2018



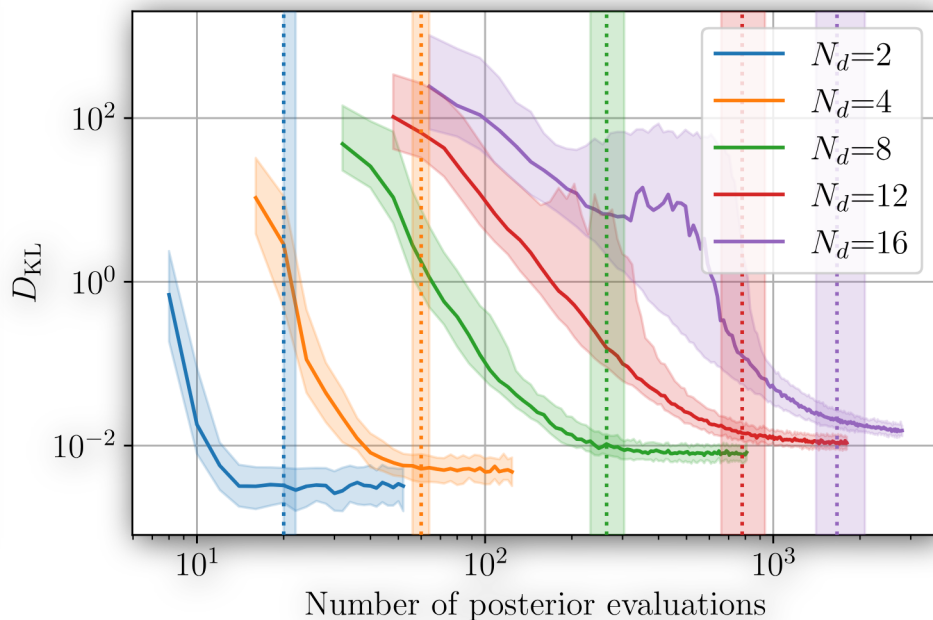
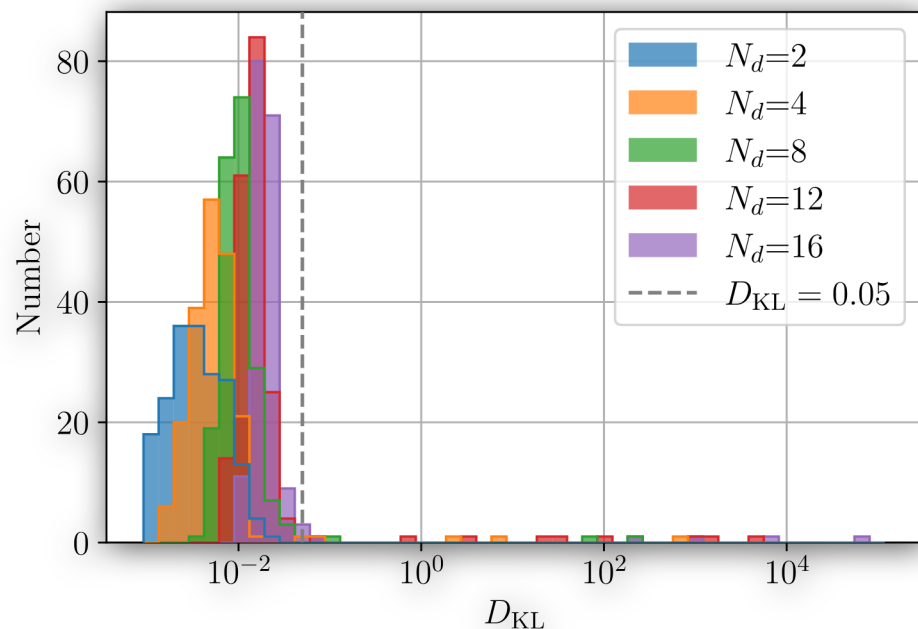
~ 900 evals $\rightarrow \sim 130$ iterations (CAMB/CLASS + small overhead)

More Cosmology – Λ CDM with Planck 2018 FULL

(don't do this!)

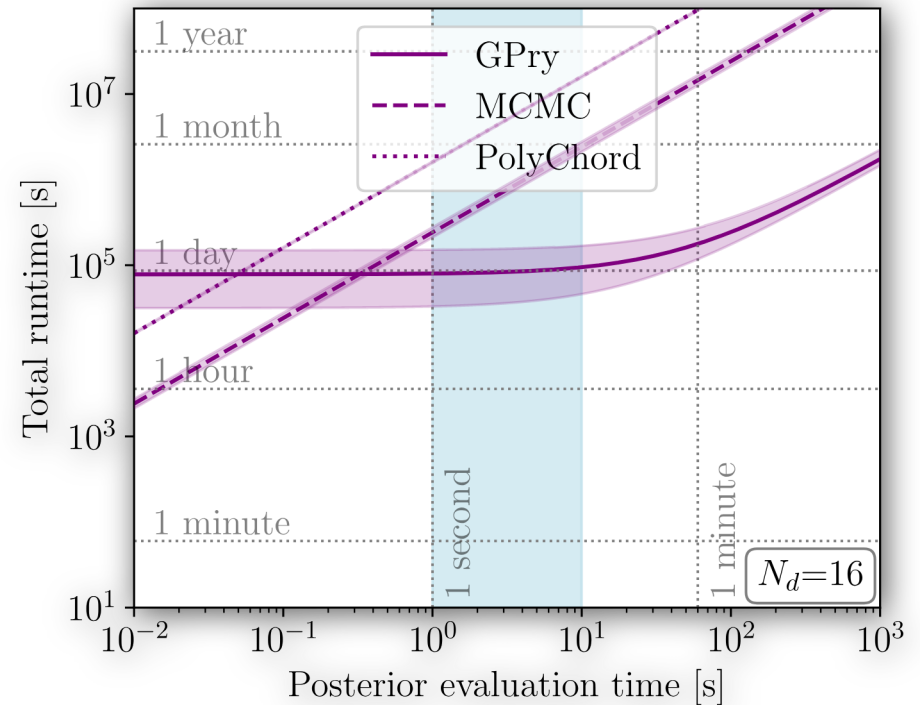
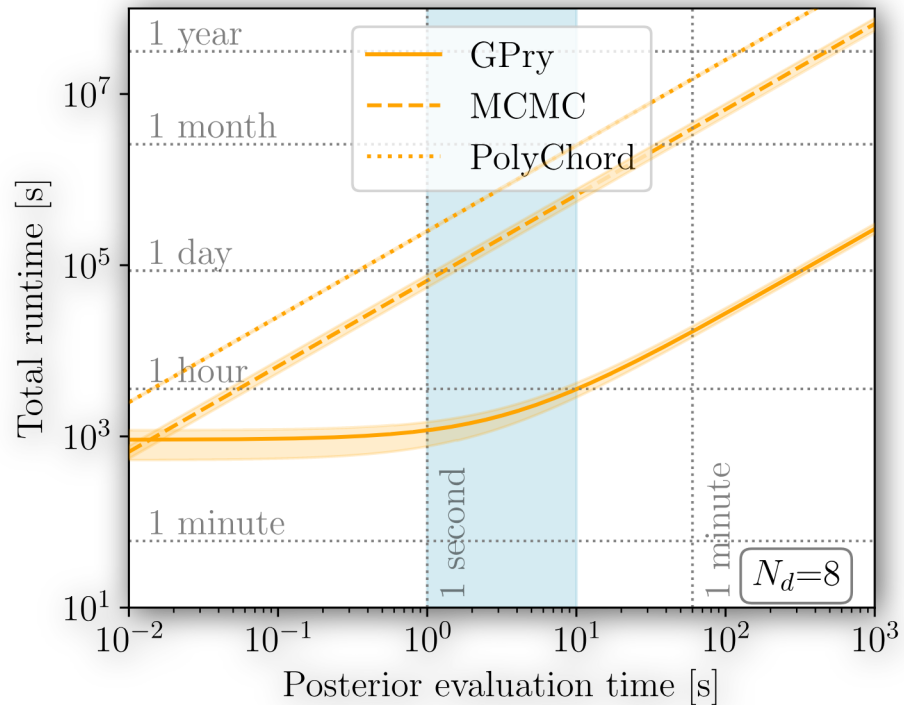


Examples/convergence experiments: Gaussians

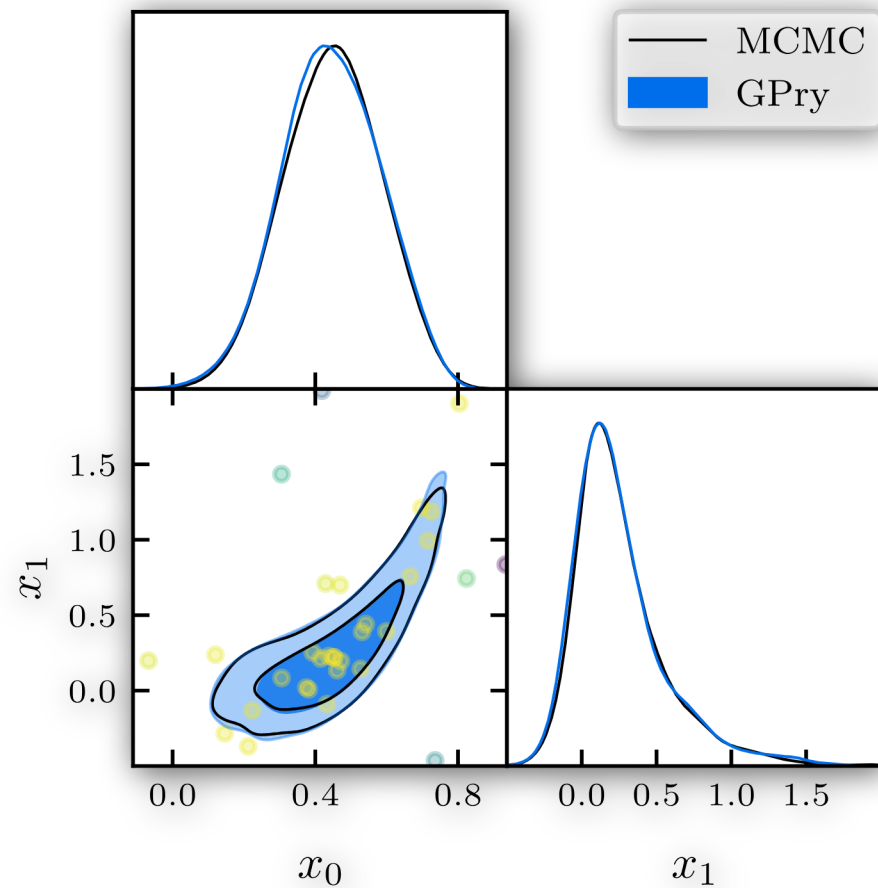
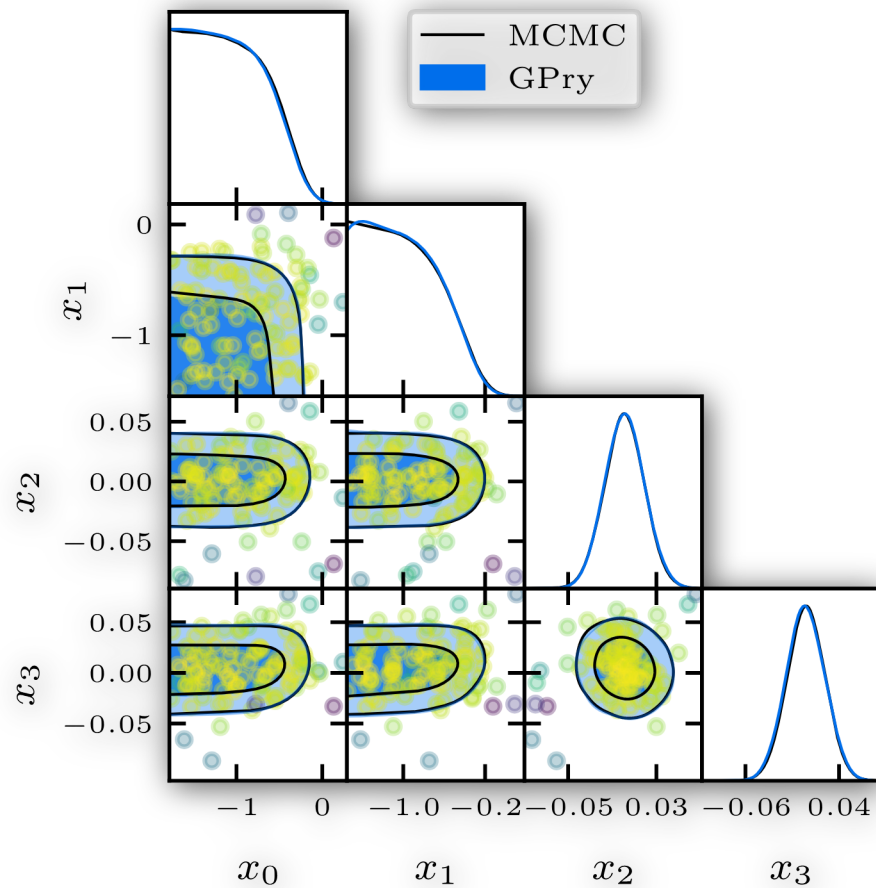


Examples/convergence experiments: Gaussians

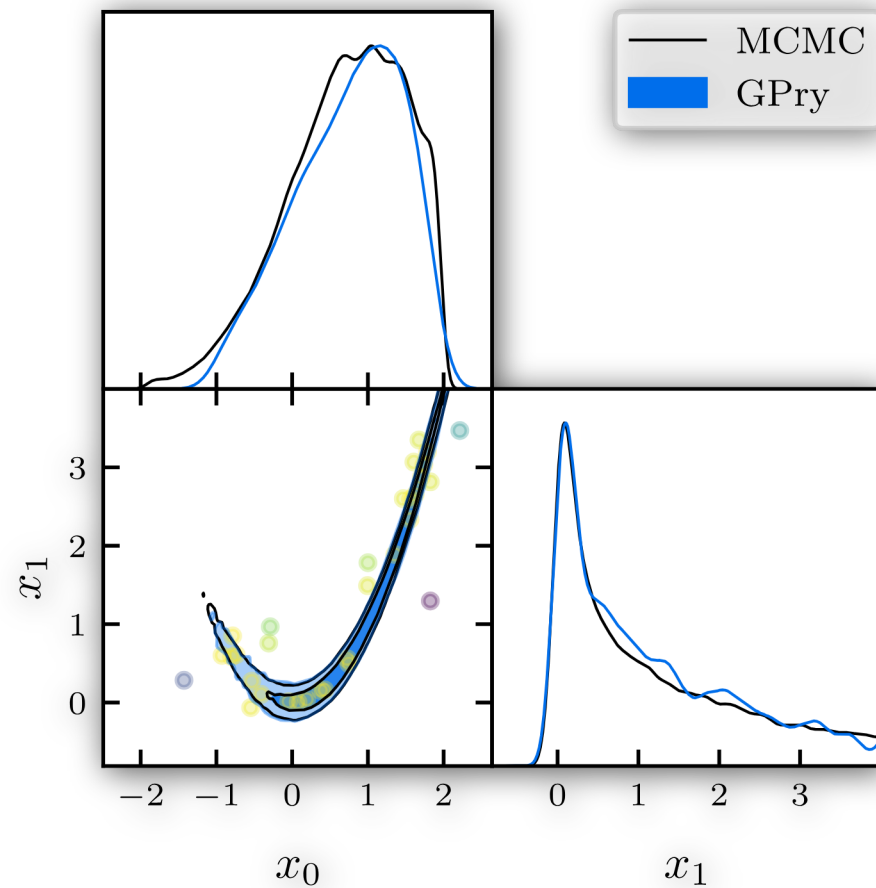
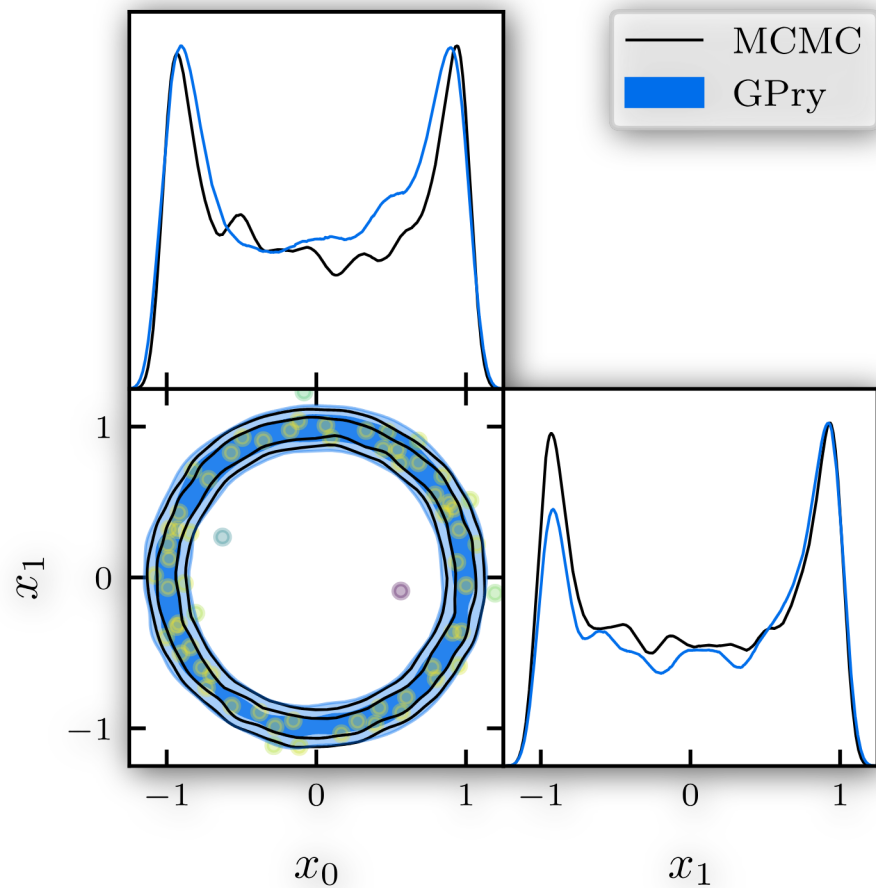
Cost comparison:



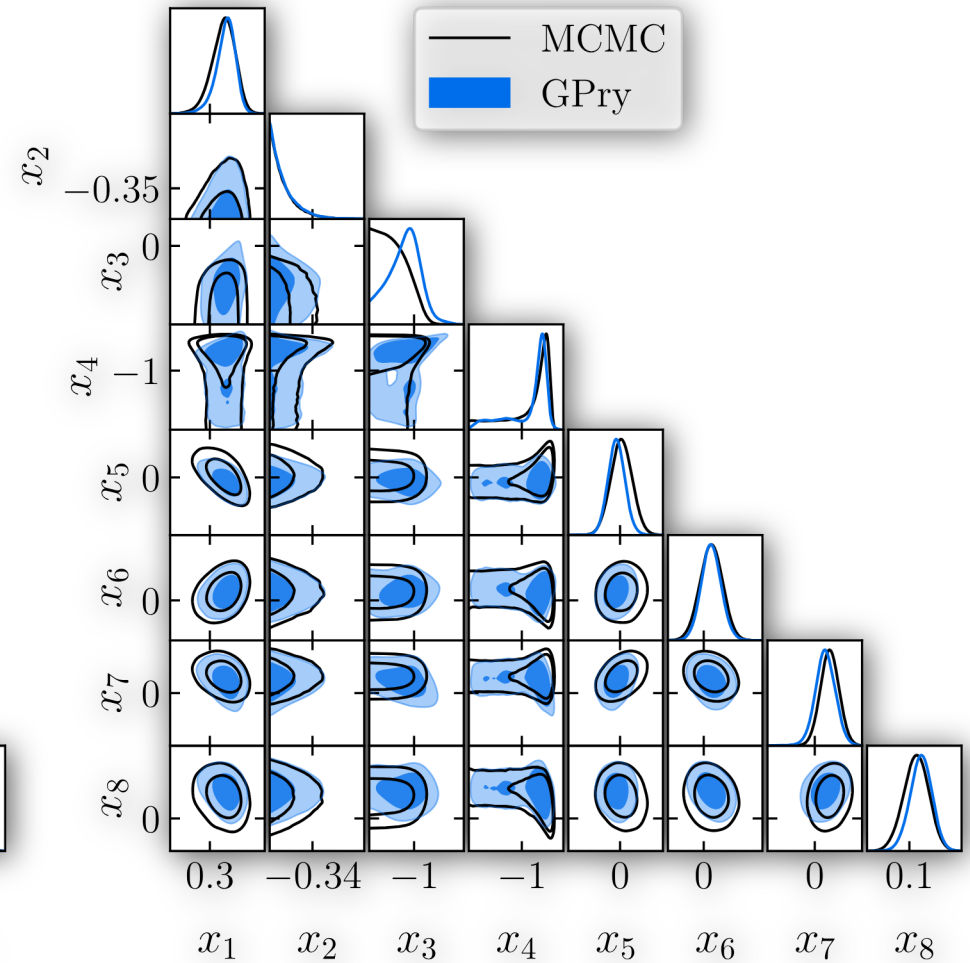
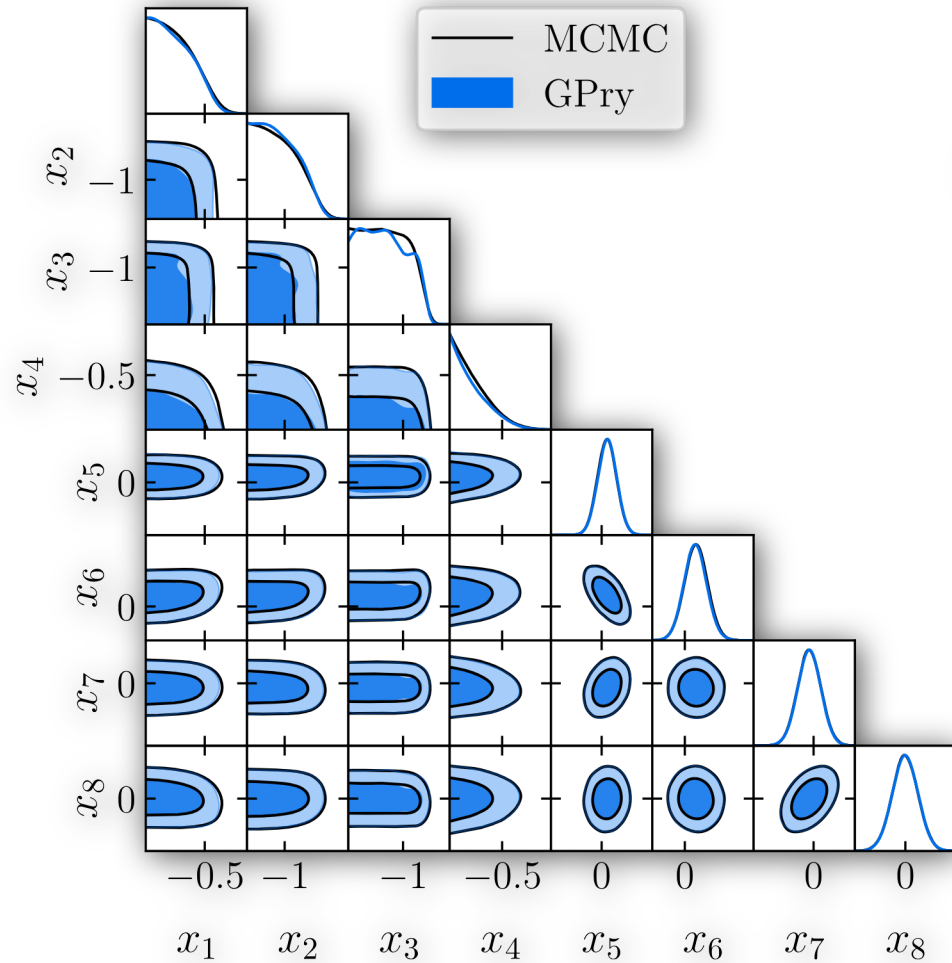
Examples: non-Gaussians



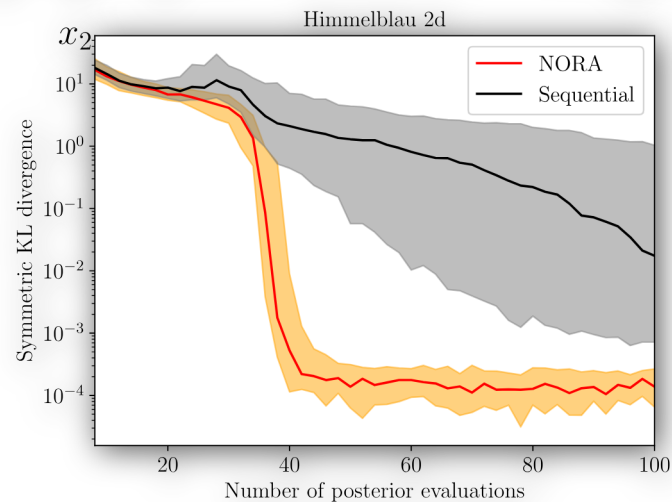
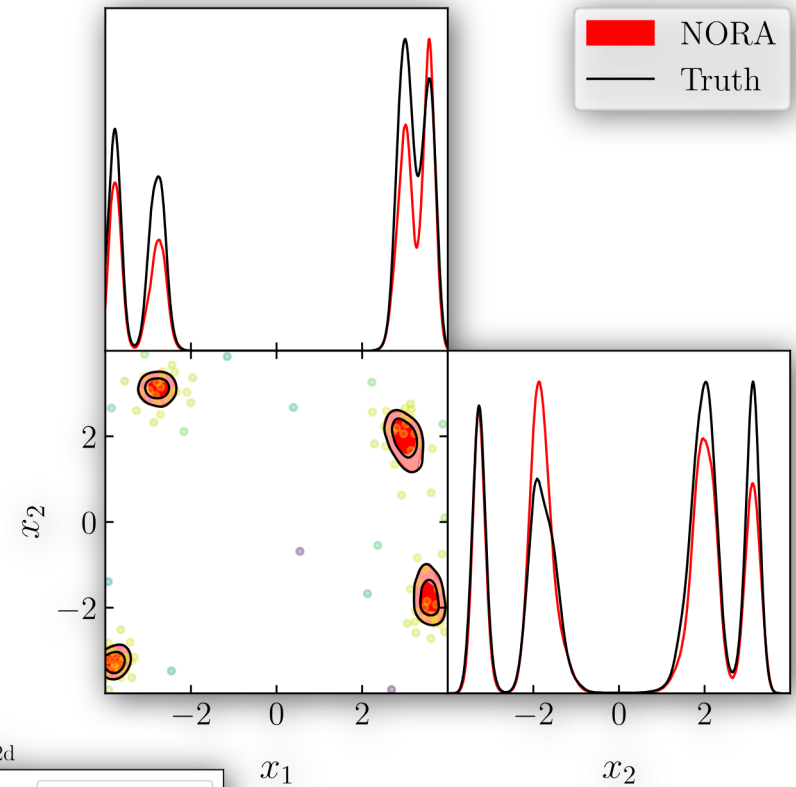
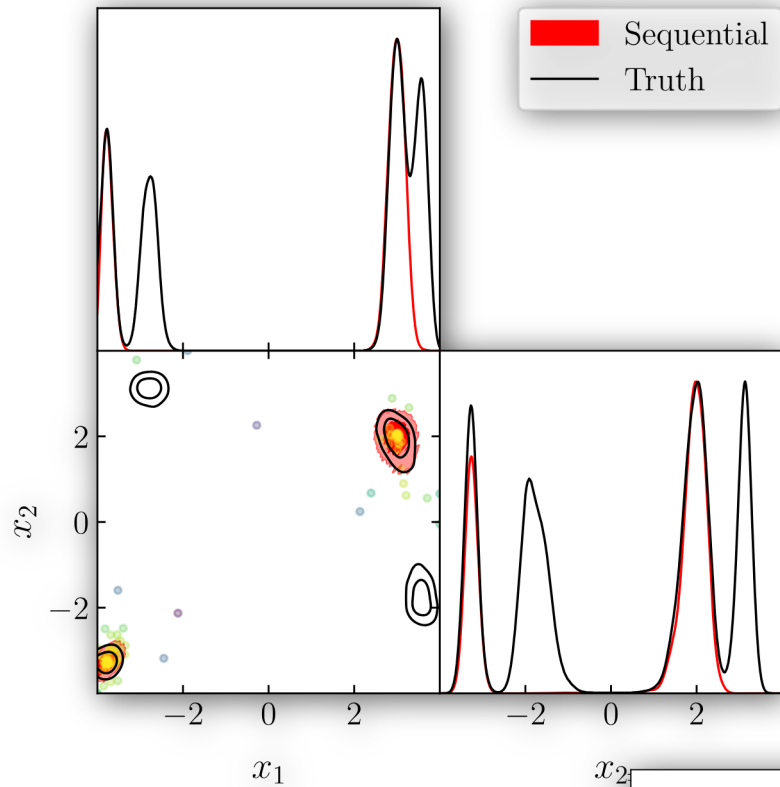
Examples: non-Gaussians (II)



Bad stuff: non-gaussianity in high dim

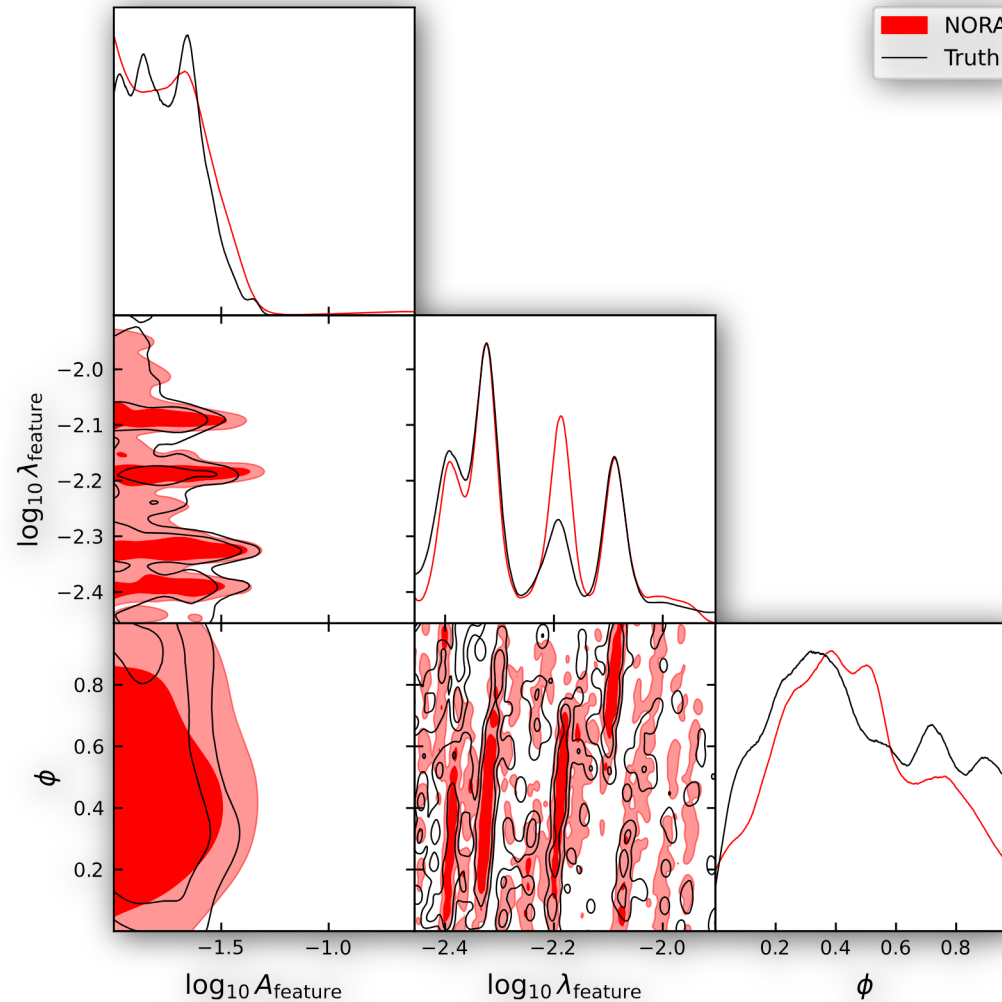


Examples/convergence experiments: Himmelblau



More Cosmology – hard multimodality

3-param primordial $P(k)$ oscillation with fixed Planck 2018 Λ CDM



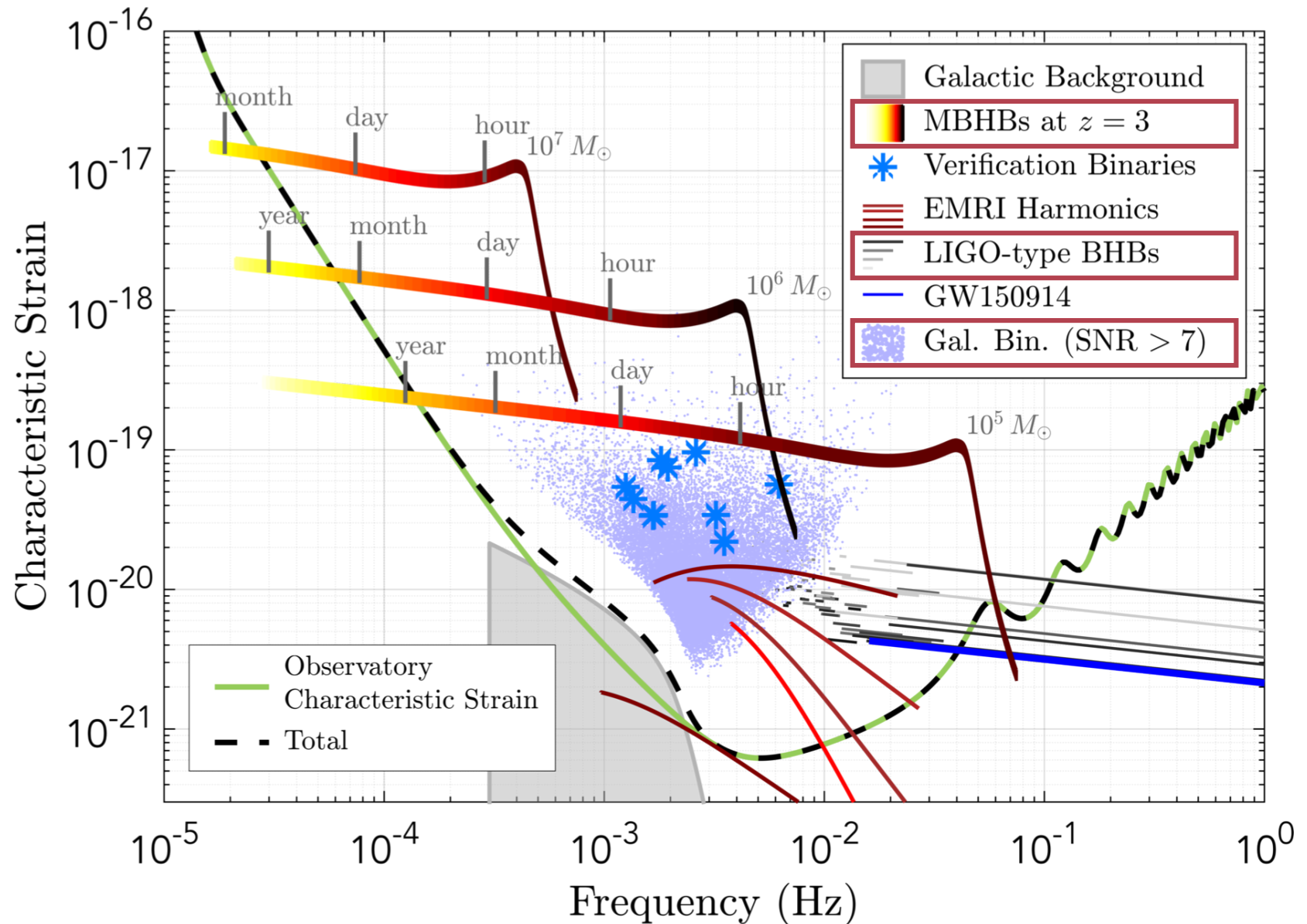
~ 2000 evals $\rightarrow \sim 350$ (batch 6) iterations (CAMB/CLASS + small overhead)

GW with LISA

[arXiv:2503.21871](https://arxiv.org/abs/2503.21871)

with J. el Gammal, R. Buscicchio and G. Nardini

Resolvable signals in LISA (fig from White Book)



A bit more on SMBHBs

- Most massive SM binaries expected to emit GWs (merge) in LISA band
- In fact, loudest transient LISA signals: SNR as large as $\sim 10^3$
- State-of-the-art pheno waveform for inspiral-merger-ringdown (numerical GR calibrated, incl. higher-modes): **IMRPhenomXHM**.
Fast in principle, but **output array is large** (LISA freq. resolution) (*)
- Rapid evolution $\Rightarrow$ broadband $\Rightarrow$ **expensive likelihood calculation** (*)

(* also true for EMRIs!)

- Total eval time per parameter change (waveform+likelihood): $\mathcal{O}(1s)$
- Out test SMBHB: emitting up to 3.5 mHz, merging $\tau_m = 4.138 \times 10^6$ s after LISA switched on.

No time-domain truncation, signal modelled at highest frequency resolution available.

- GW part of the pipeline managed by **BALROG** (to be released!)

Can GPry accelerate inference of individually-resolved sources?

(*Application to SGWB with expensive theoretical models: [El Gammal et al. '25 \(2501.11320\)](#))

Goal: Accurate inference with $\mathcal{O}(10^3)$ W+L evals, low overhead.
(that is: $\mathcal{O}(10^{-2})$ times a typical NS with `nlive` ~ 1000)

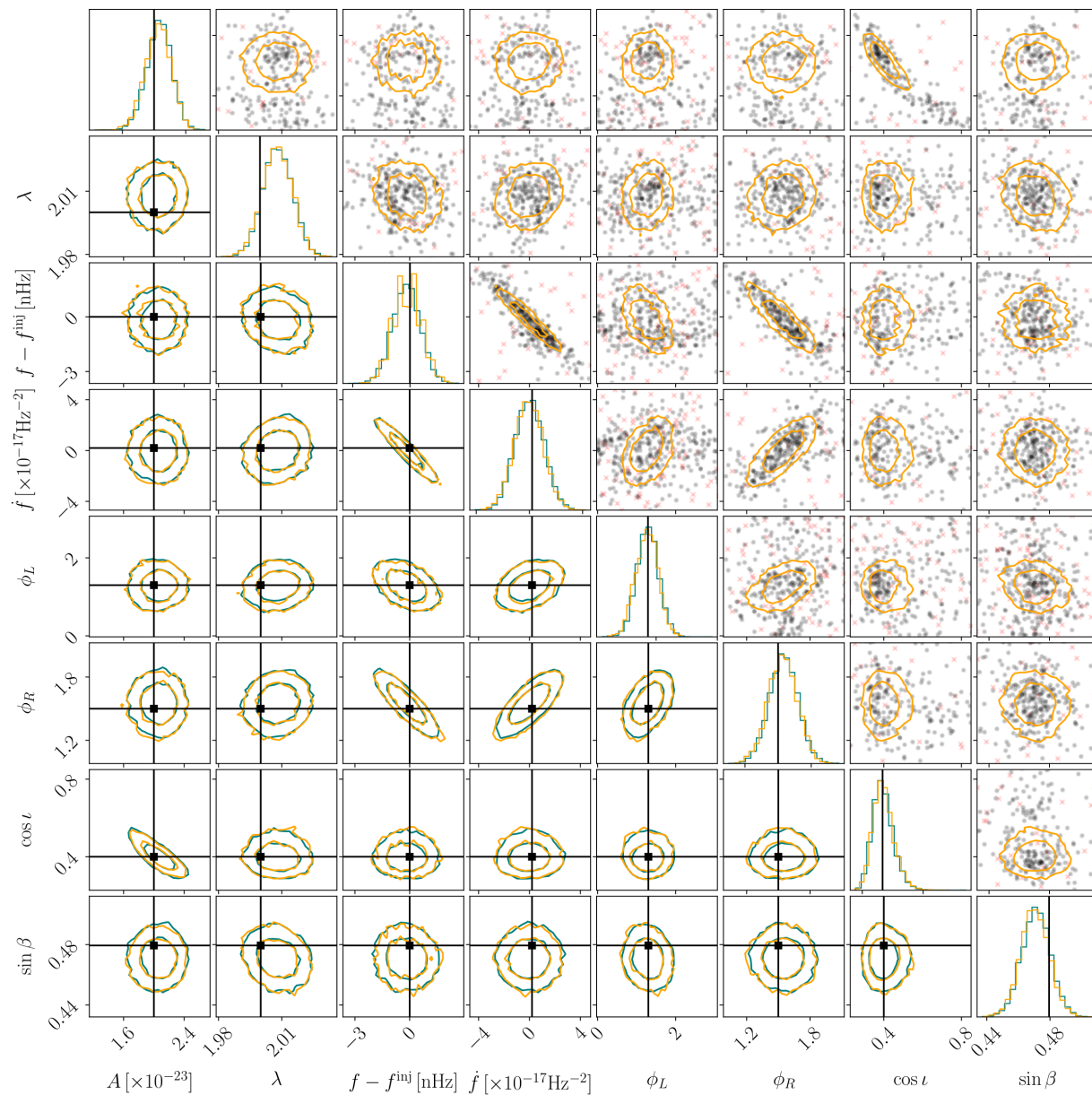
Slower than an amortised approaches, but requires no pre-training
(i.e. useful for "low-latency", novel theoretical models, exotic signals...).

Tests, single source + rand. realisation Gaussian, stationary, known noise:

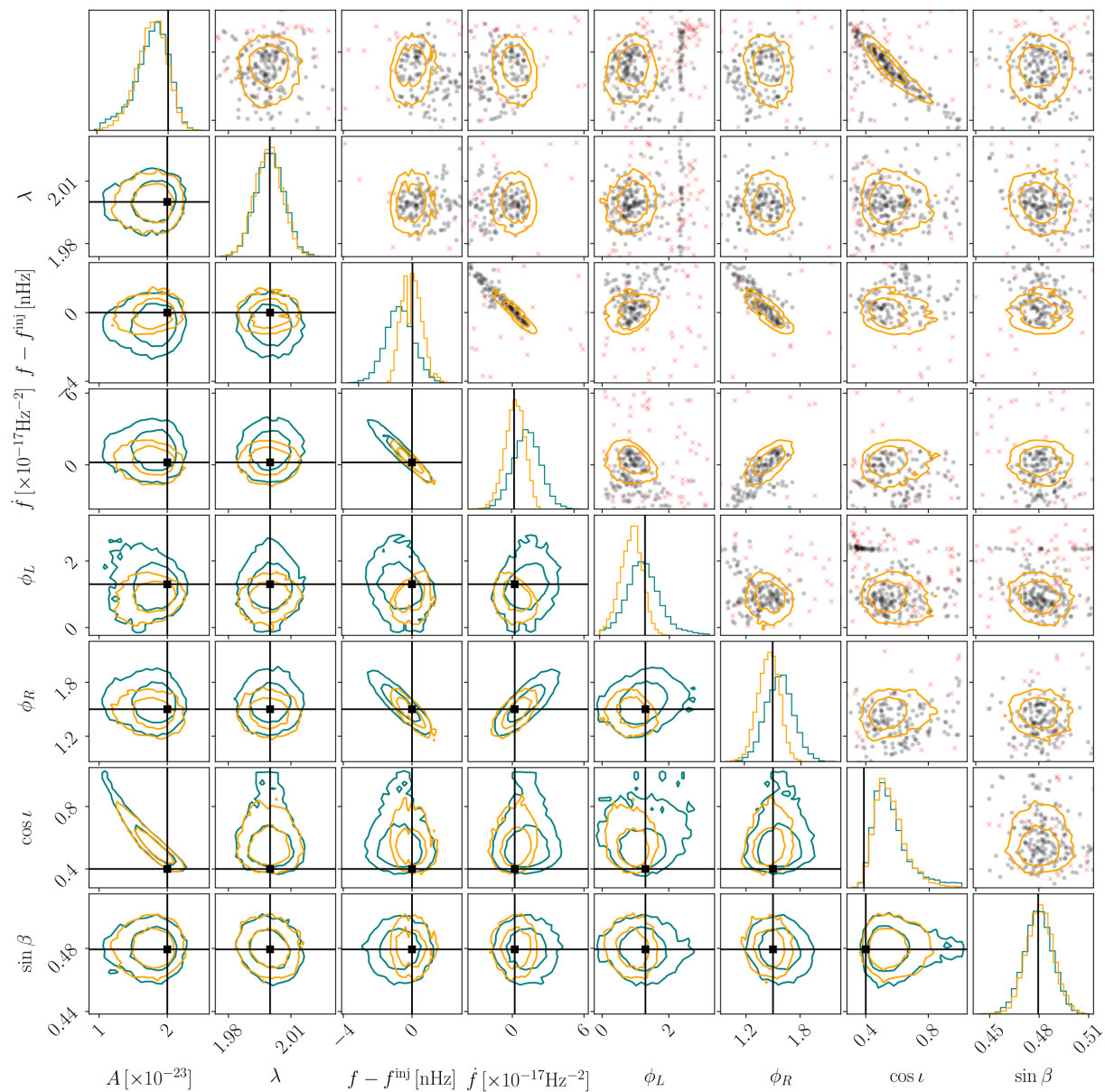
- **DWD**: $\text{SNR} \sim 20$, $t_{\mathcal{L}} \sim 10^{-3}$ s — no expected improvement
- **stBHB**: $\text{SNR} \sim 20$, $t_{\mathcal{L}} \sim 10^{-1}$ s
- **SMBHB**: $\text{SNR} \sim 2000$, $t_{\mathcal{L}} \sim 1$ s — **most promising!!!**

Pre-conditioning: prior cuts at $\sim 10\sigma$ from amortised/approx. methods.

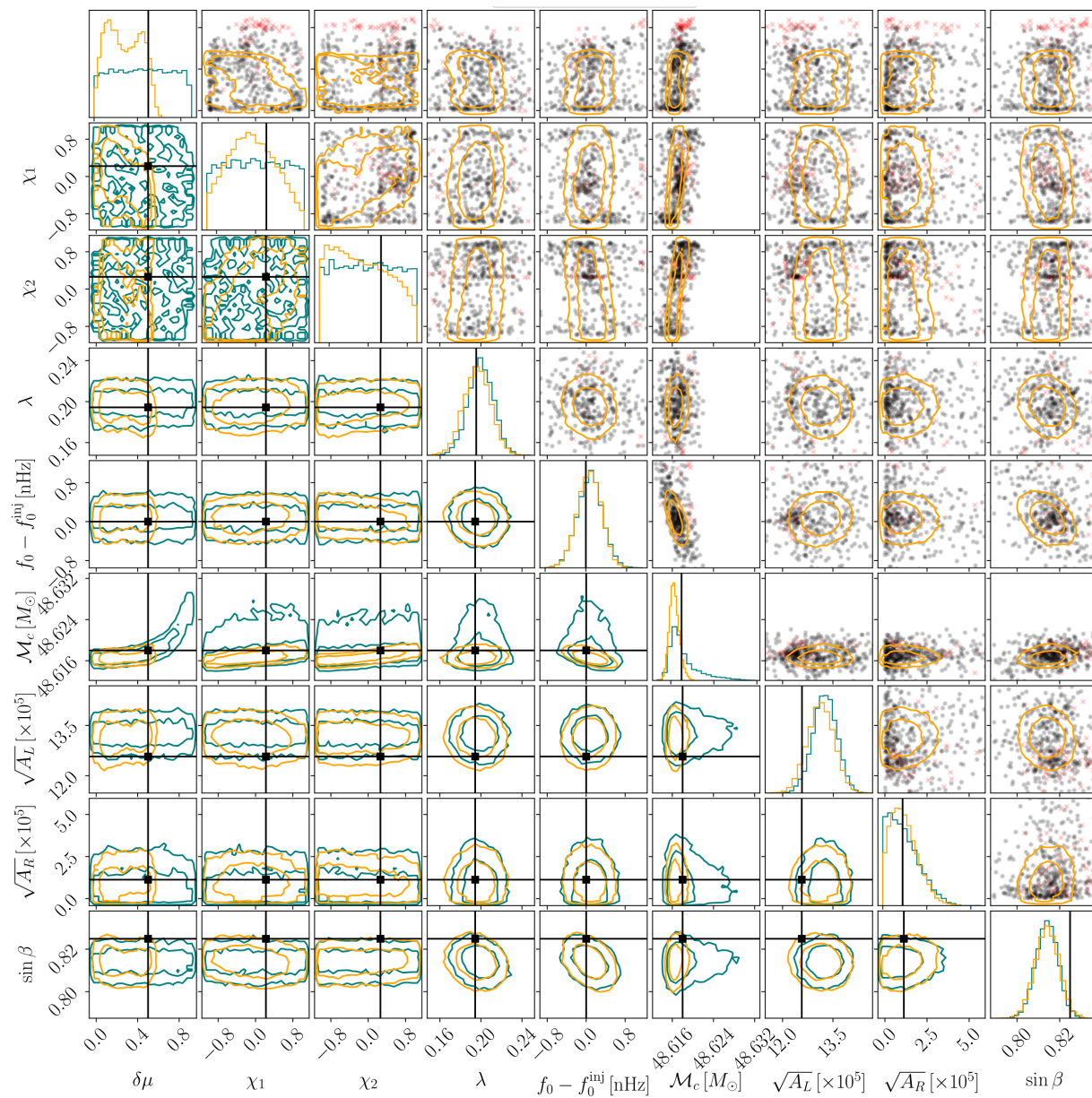
DWD (median $D_{JS} \approx 0.005$) : 500 vs 136 500



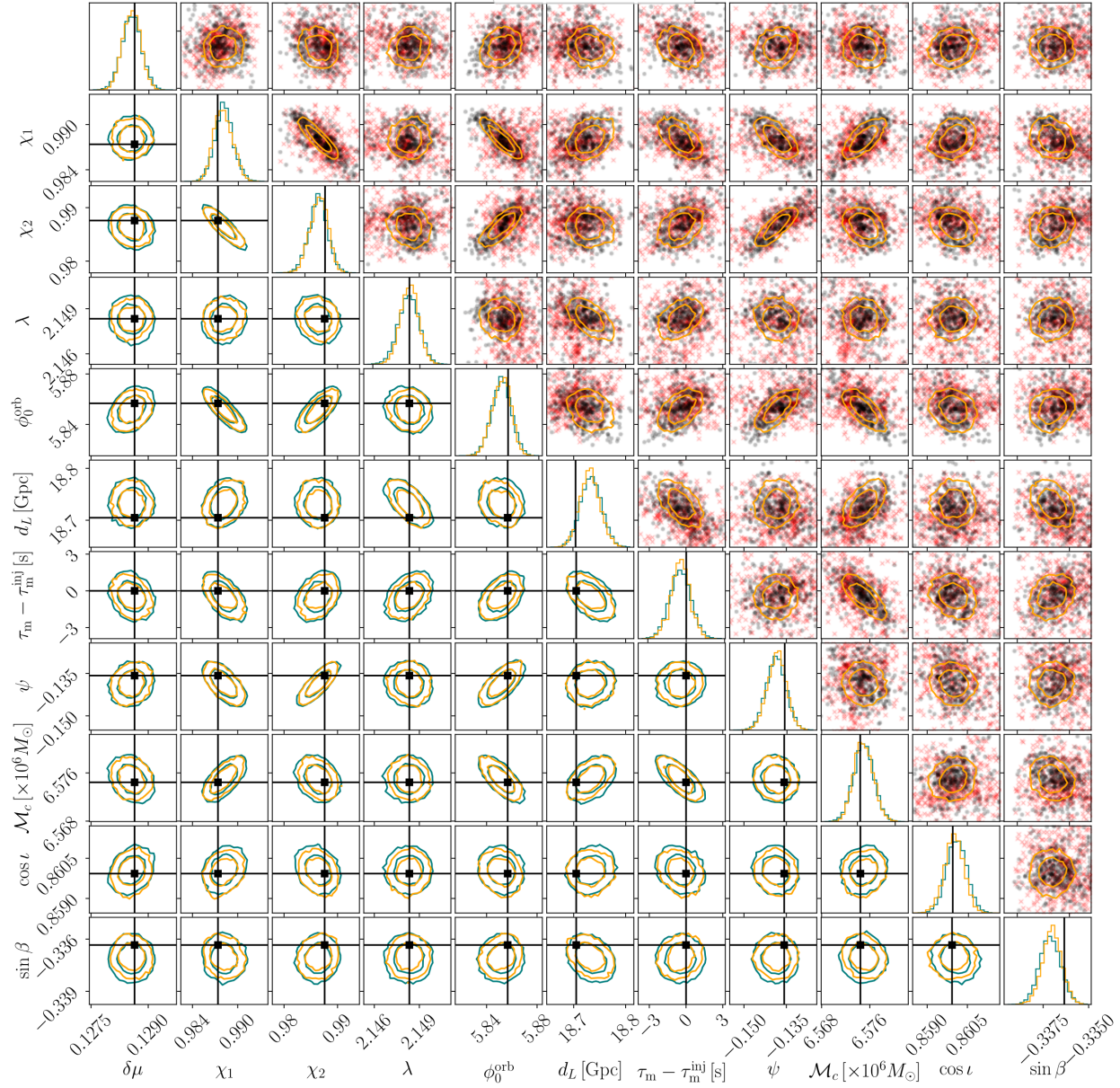
DWD (1/100 outlier! $D_{JS} \approx 0.12$): 500 vs 146 000



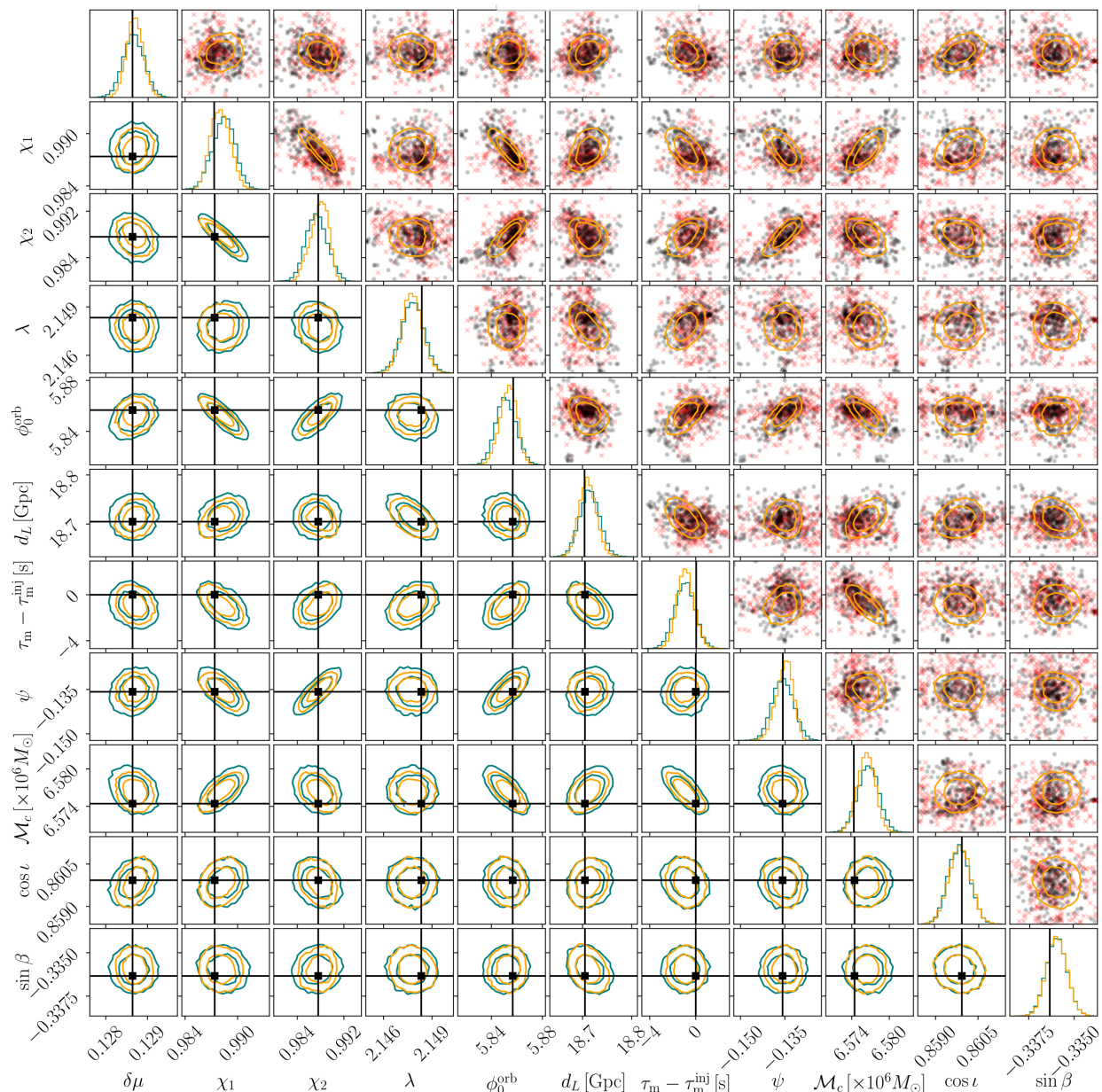
stBHB (median $D_{\text{JS}} \approx 0.19$) : 450 vs 23 484 500



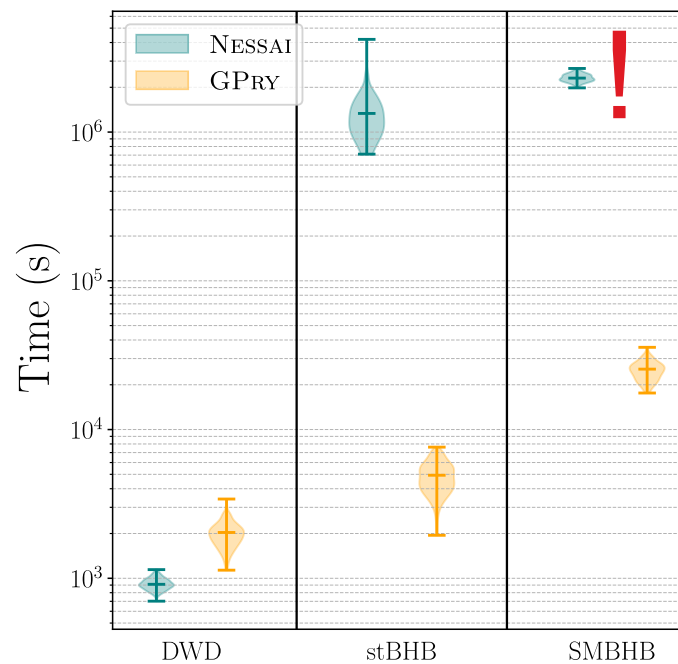
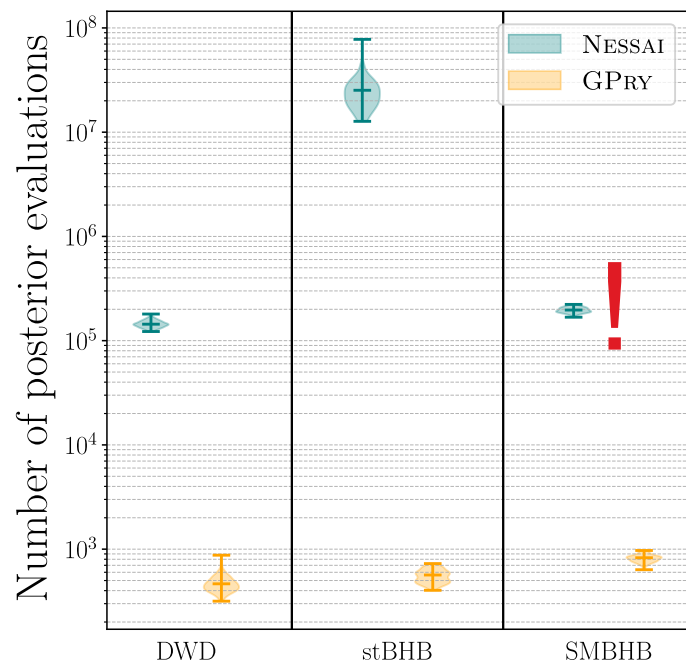
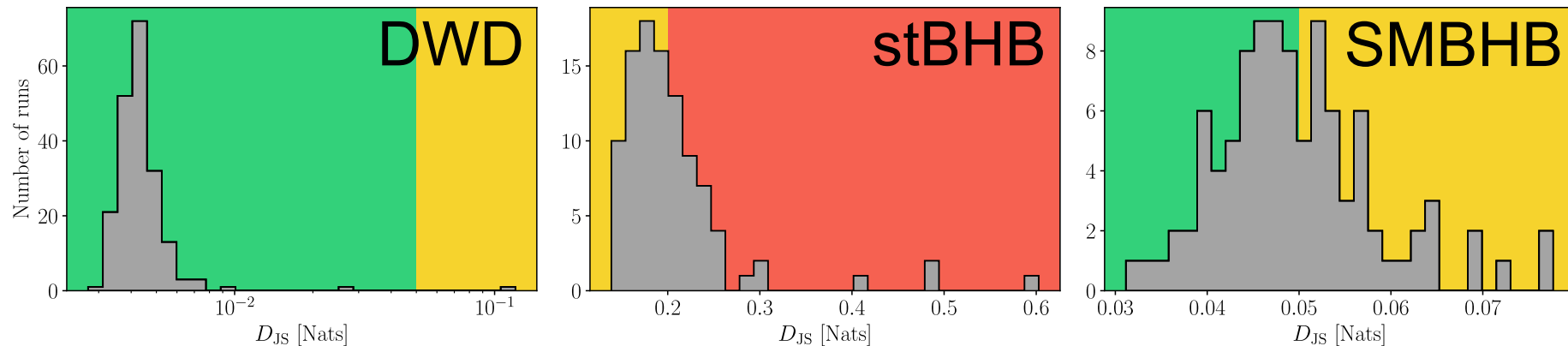
SMBHB (median $D_{JS} \approx 0.05$): 850 vs 207 000 (567 000)



SMBHB (worst $D_{\text{JS}} \approx 0.08$) : 650 vs 198 500 (364 000)



Results overview



Conclusions & outlook

Focus is on **robustness**! Error states: very rare **confident wrong answers**.

Main advances: novel parallelised acquisition, prior modelling of the dynamic range of the GP, SVM classifier for null-likelihood regions.

$\mathcal{O}(> 100)$ **fewer evaluations** (+overhead).

Way faster than MC for posterior evaluation costs of $\gtrsim 1$ s in $d \sim 10$.

WIP: strong **non-Gaussianity** and **multimodality** in $d \gtrsim 10$ (maybe NF trained on synthetic samples produced during active learning). Also overhead reduction (GPU, sparse GPs...)

LISA sources: DWD 👍 (but not cheaper), stBHB 👎, SMBHB 👍 👍 👍

Large potential: new physics (no amortization), other sources (**EMRIs**...)

Give it a try: <https://github.com/jonaseigammal/GPry> & <https://gpri.readthedocs.io>

Thanks!

Bonus slides

A note on likelihood-free approaches

The idea of likelihood-free approaches:

- Create a simulator $\theta \rightarrow \mathcal{D}^{\text{sim}}(\theta)$ (stochastic: phys. perts. and noise)
- Sample from joint $(\theta, \mathcal{D}^{\text{sim}})$ space, and estimate density w.r.t. some **divergence** between $\mathcal{D}^{\text{sim}}$ and the actual data $\mathcal{D}$.

That **density** implicitly defines a **data likelihood** $\mathcal{L}(\mathcal{D} \mid \theta, \mathcal{D}^{\text{sim}})$.

Limitations:

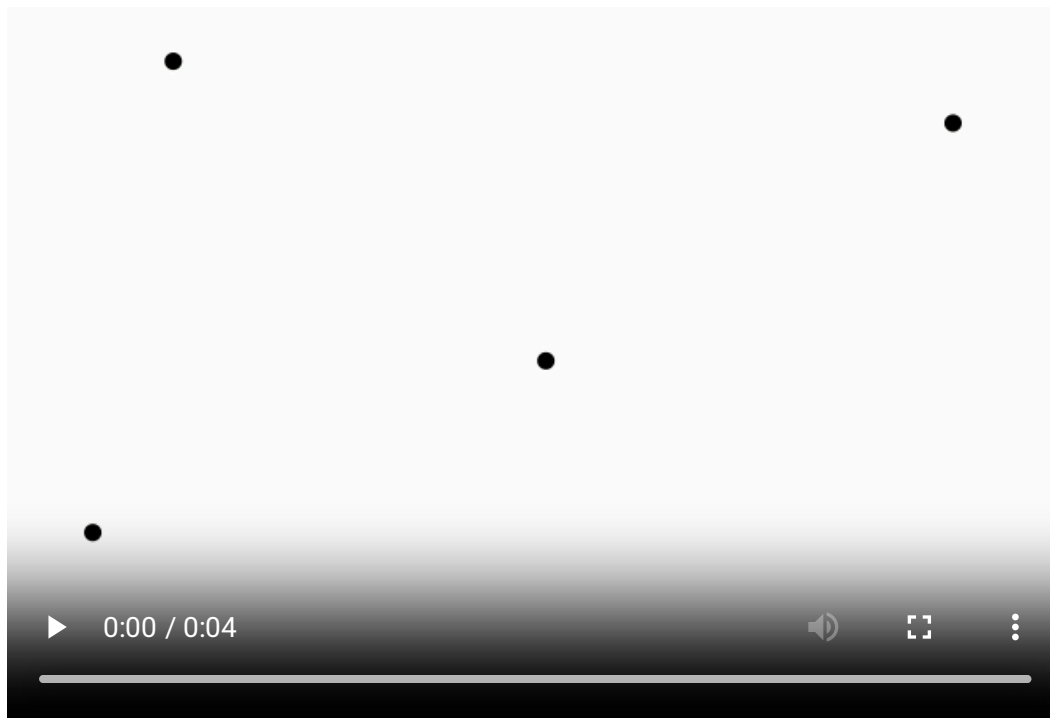
- Low reusability, since **derived likelihoods depend on models**.
- Less statistical control (propagation of stochasticity is simulated)
- Limited discoverability (better than summary statistics, but hard goodness-of-fit assessment)
- Assessing robustness is an open problem.

"Are theory-independent summary statistics an unfortunately necessary intermediate step? Or a useful tool for Physics?"

Nested Sampling

1. Draw n points X_i from the prior.
2. Drop the one with $p_i \stackrel{\text{def}}{=} \min \mathcal{P}(X_i)$.
3. Draw a new one (e.g. from the prior) $\tilde{X}_{i+1}$ with $\mathcal{P}(\tilde{X}_{i+1}) > p_i$.

The set of all dropped points will form a sample of the true distribution!

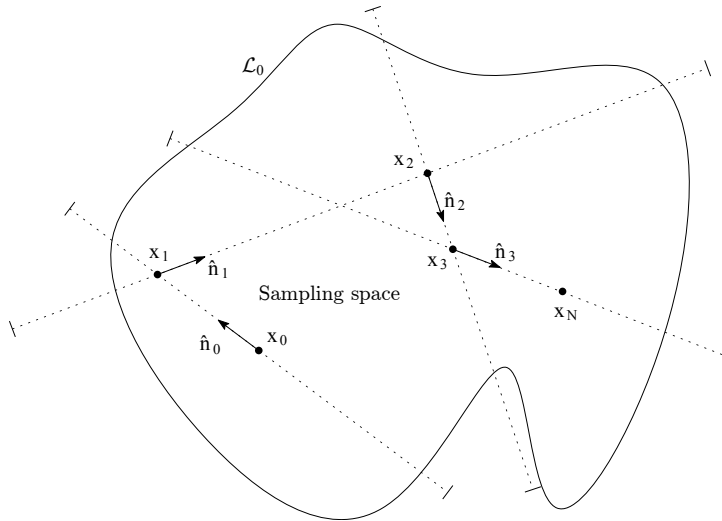


(Gif by Will Handley)

Nested Sampling – II

Step 3 is the hard one! As the iso- $\mathcal{P}$ get smaller and smaller, it's harder and harder to hit them by chance $\rightarrow$ bad scaling with dimensionality.

Naively, this can be ameliorated by computing the approximate covariance ellipsoid at each step, and sampling from them (e.g. [CosmoNest](#) (Mukherjee et al. '05) [MultiNest](#) (Feroz et al. '08)).



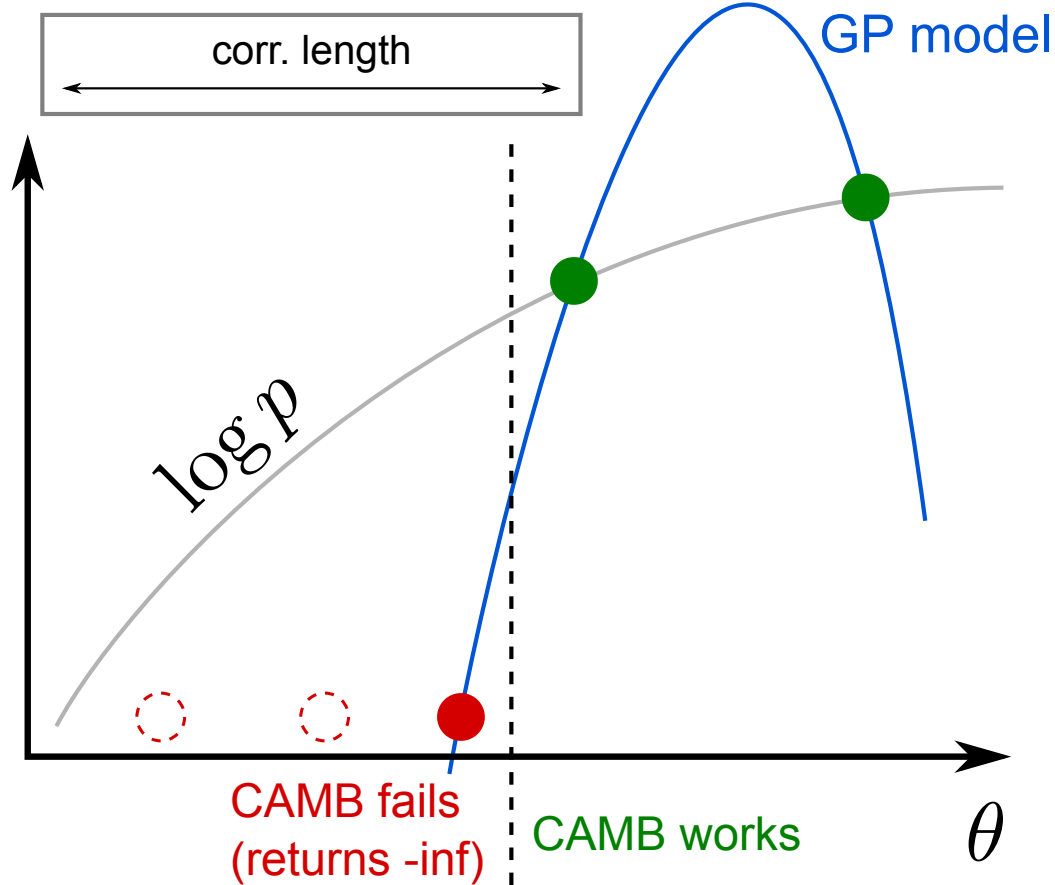
Even better, use a Markov Chain to generate new points from old ones (chains scale better with dimensionality, and current points are close to good new ones).

(Image from [PolyChord](#) (Handley et al. '15))

E.g.: [PolyChord](#) (Handley et al. '15), also implementing *whitening* and *clustering*.

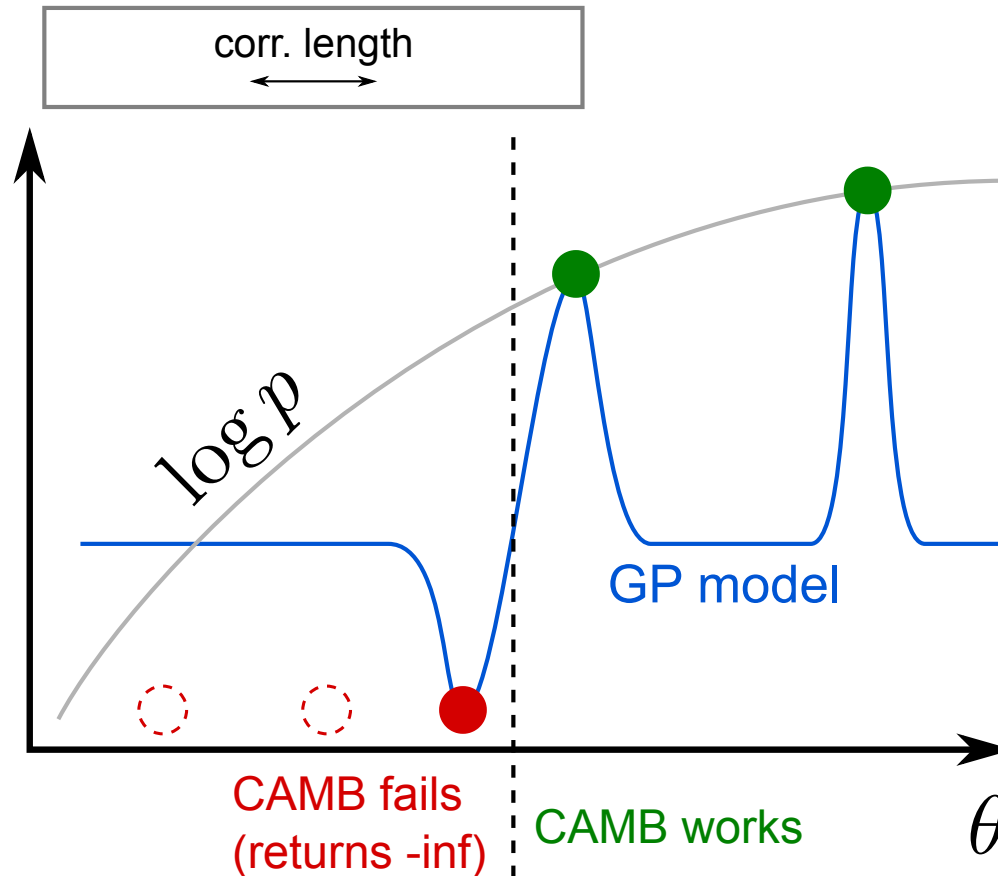
Good dimensionality scaling (MCMC-generation) and good at multimodality (clustering) $\rightarrow$ good *exploratory* tool, *slow-ish* but **highly parallelisable**.

Infinites and extreme values



Likely to either **overshoot** (long corr. length)

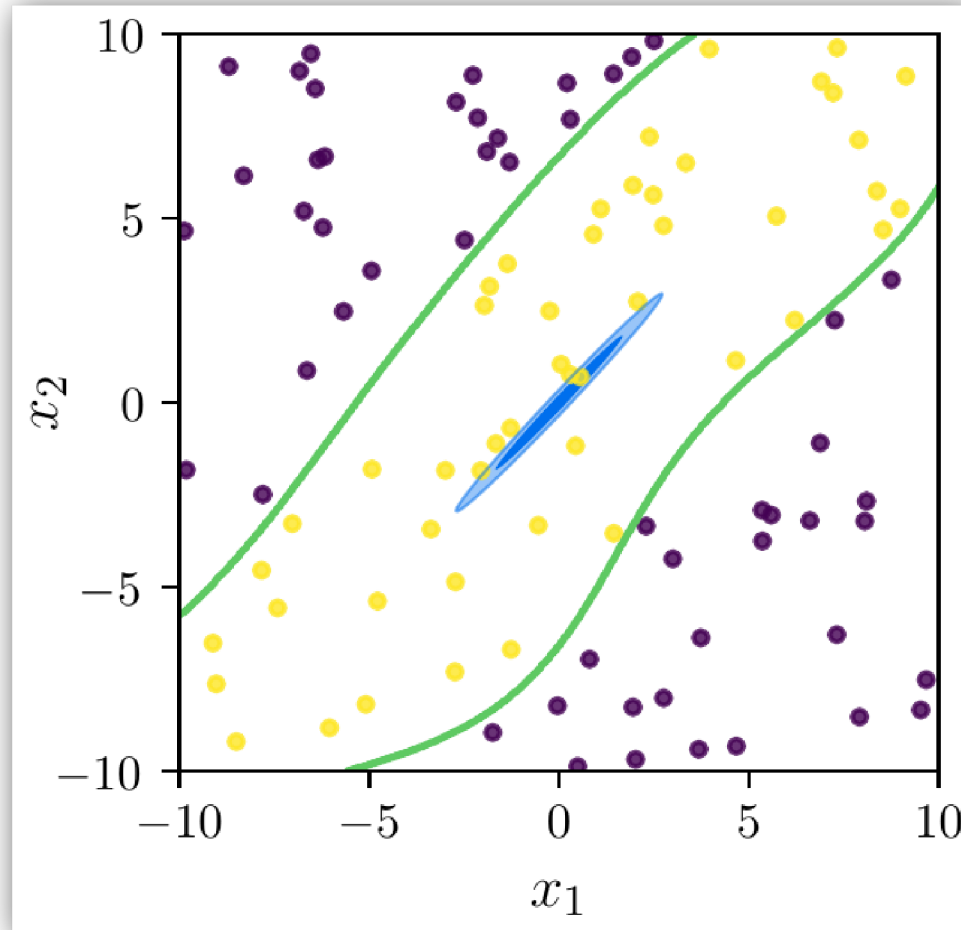
Infinites and extreme values



Likely to either overshoot (long corr. length) or **overfit** (short corr. length).

Regularising the bad points doesn't help too much (and is finicky)

SVM, a cheap classifier to learn which regions to ignore



Our solution to this problem is to simultaneously exclude these *infinities* from the GP, and to use them to divide the parameter space into a *finite* and an *infinite* region using a support vector machine (SVM) classifier [61, 62].⁸ A SVM defines a hyperplane which maximizes the separation between samples with locations $\mathbf{x}_i$ belonging to one of two classes $y \in \{-1, 1\}$. By defining the distance between points through a kernel function $k(\mathbf{x}, \mathbf{x}')$ the separating hyperplane is drawn in a higher-dimensional space which is connected to the sample space by a non-linear transformation. This effectively transforms the separating hyperplane into more complex hypersurfaces which are better suited to the classification problem at hand.

The categorical predictions $\hat{y}(\mathbf{x})$ of the SVM are then given by

$$\hat{y}(\mathbf{x}) = \text{sgn} \left(b + \sum_{i=1}^{N_s} \alpha_i k(\mathbf{x}_i, \mathbf{x}) \right) \quad (3.3)$$

where the hyperparameters b and α_i are optimized in the training procedure.

SVM, a **cheap** classifier to learn which regions to **ignore** (II)

Calibration needs knowledge about the expected dynamical range of the target function, i.e. *how small is too small and w.r.t. what?*

We go back again to our Gaussian example, where we can calculate the dynamical range, and simply choose a threshold in terms of credibility ϵ .

We have found that the ϵ equivalent to 20 1D standard deviations works well, and the correct scaling is guaranteed.

The criterion auto-tunes with dimensionality!

(The SVM will not only ignore $-\infty$ points, but also very-low-value ones. That's OK, and will keep the model cheaper and more numerically robust.)