Nested Sampling

John Veitch
University of Glasgow

PyCBC Inference workshop, University of Portsmouth
2019-05-14

Contents

- Evidence and model selection
- Nested Sampling algorithm
- Convergence and Errors
- Posterior Samples
- Optimisation and Performance
- Extensions
- Examples

Model Selection

- We often want to compare probability of different models or hypotheses. This is simple if there are no parameters but if models have different parameters, dimensions etc then how to do it?
- Marginalised Likelihood (a.k.a Evidence) is just the *mean likelihood* of a model *A*

$$Z = \int d^n\theta \, p(d|\theta, A)p(\theta|A) = \langle p(d|\theta, A)\rangle_{p(\theta|A)}$$

- “Gold standard” of model selection. Approximated by quick-n-dirty “information criteria” e.g. AIC, BIC, DIC

Model Selection

- To test different hypotheses we can compute the ratio of their probabilities P(A)/P(B), known as the “odds ratio”
- Given some experimental data we can update our relative belief in two models

$$\frac{P(A|d)}{P(B|d)} = \frac{P(A)}{P(B)} \frac{P(d|A)}{P(d|B)}$$

- The evidence ratio is also known as the “Bayes Factor”

$$\frac{P(d|A)}{P(d|B)} = \frac{\int_X p(d,x|A)dx}{\int_Y p(d,y|B)dy}$$

- Allows comparison of models of different parameter spaces X and Y

Evidence & partition function

- Bayesian evidence is analogous to partition function in thermodynamics

$$Z(\beta) = \int e^{-\beta H(x)} d^n x$$

$$\beta \to 1 \qquad\qquad H \to -\log L$$

- "Thermodynamic integration" can compute Z by integrating over inverse temperature β, as in canonical ensemble.

- Nested sampling is similarly inspired by *micro-canonical* ensemble. Density of states g(E) is inverse Laplace transform of $Z(\beta)$.

- Integrate over E instead of integrating over β ?

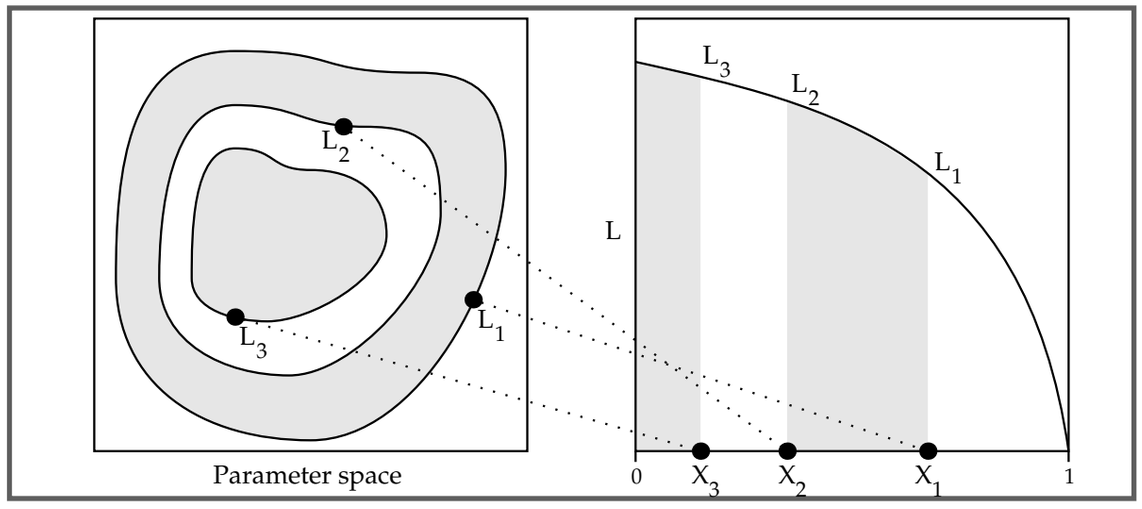
- Skilling pursued this to find the Nested Sampling algorithm (Skilling, "Nested sampling for general bayesian computation", 2004)

Nested Sampling

- Use Nested Sampling to evaluate the integral probabilistically.
- From product rule, get $p(d|\vec{\theta},H,I)p(\vec{\theta}|H,I)d\vec{\theta} = P(d|H,I)p(\vec{\theta}|d,H,I)d\vec{\theta}$
- Write volume element of prior as $dX = p(\vec{\theta}|H,I)d\vec{\theta}$ and note $\int p(\vec{\theta}|d,H,I)d\vec{\theta} = 1$
- Evidence is then given $Z = P(d|H,I) = \int p(d|\vec{\theta},H,I)dX = \int LdX$
- Define $X(\lambda) = \int_{L(\vec{\theta})>\lambda} dX$ as the prior probability mass covering the regions of parameter space with likelihood >L. As L increases, enclosed probability X shrinks from 1 to 0. Using inverse function

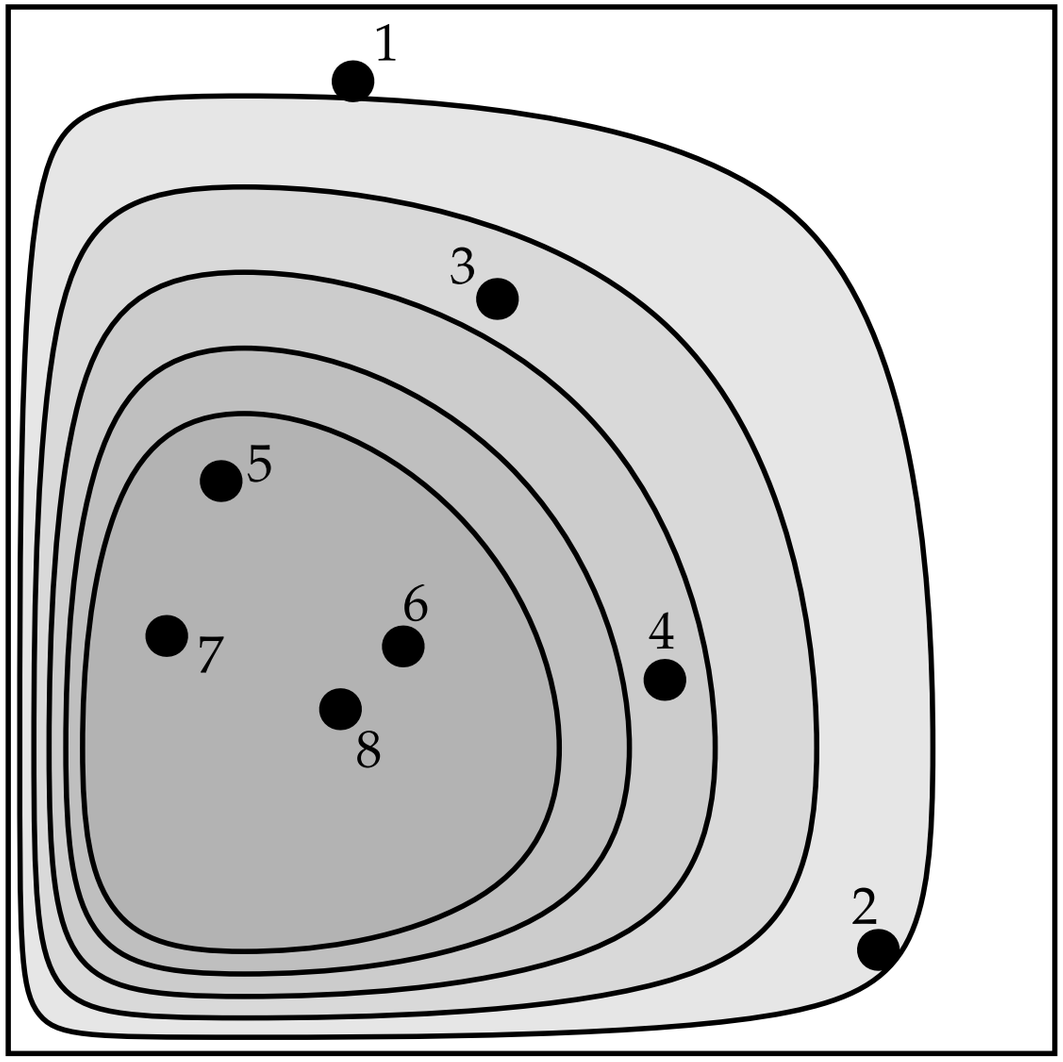
$$L(X) : L(X(\lambda)) = \lambda$$

- Now to do 1-D integral $Z = \int_0^1 L(X)dX$

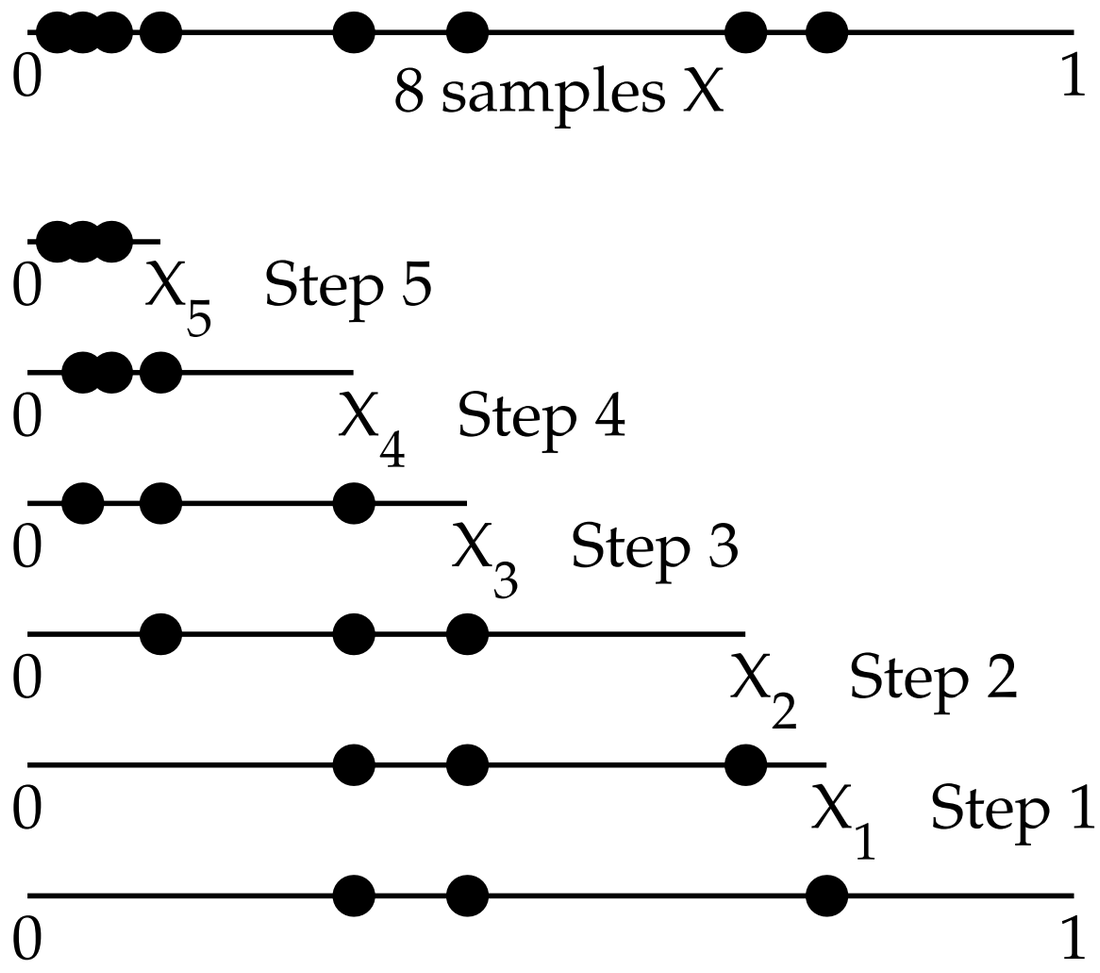


Nested Sampling

- If we had a collection of m points from the prior, we could approximate the integral as Riemann sum $Z = \sum_{i=1}^{m} L_i w_i$ where $w_i = \Delta X$
- But how do we get from θ to X? Sample m points X_i randomly from the parameter space, and calculate their likelihoods L_i, then sort them by L. Lowest L corresponds to highest X.
- At each iteration, replace lowest point (L_{min} , X_{max}) by another sampled from within the limited prior $X(L_{min})$. At each iteration X therefore decreases geometrically, $X_i = t_i X_{i-1}, \qquad < \log t_i >= -m^{-1}$
- To sample the limited prior, design a special MCMC routine.
- Recalling that X_0=1, we can now evaluate the above sum by iterating the process, and find the evidence!



Parameter space



Enclosed prior mass X

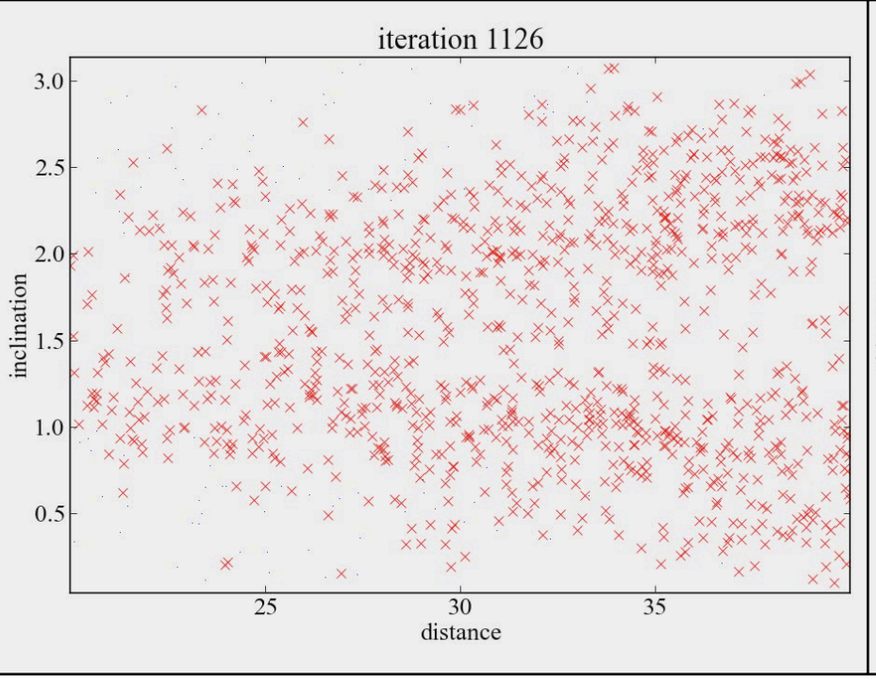
- If we have *N* live points, the enclosed prior volume at iteration k is

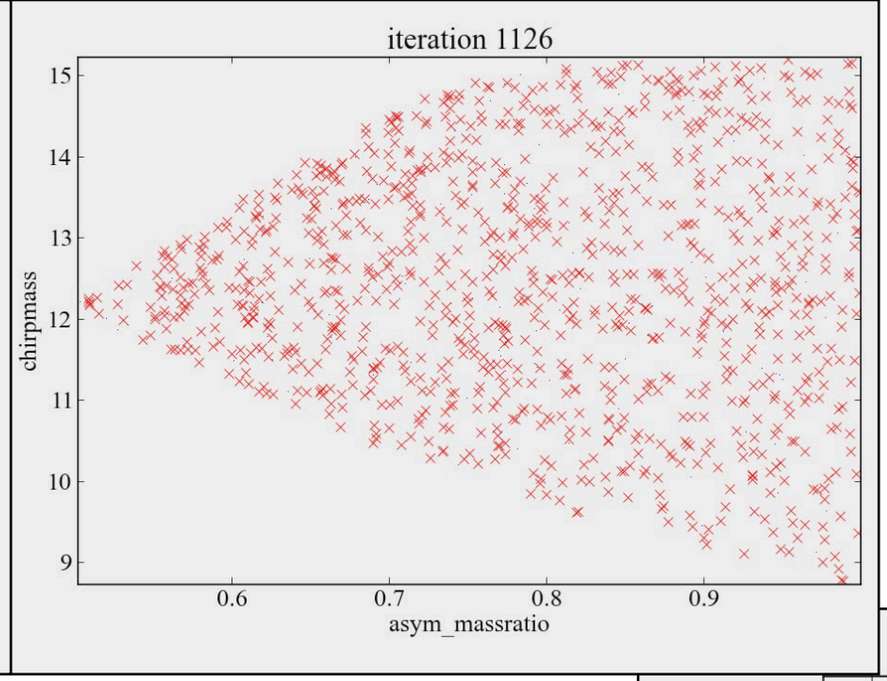
$$X_k = \prod_{i<k} \frac{N-1}{N} = \left(1 - N^{-1}\right)^k \approx \exp\left(-\frac{k}{N}\right)$$

- So no matter how small the posterior we should eventually find it (with sufficiently good sampling!)

Nested Sampling

[Veitch & Vecchio PRD 81 (2010)]

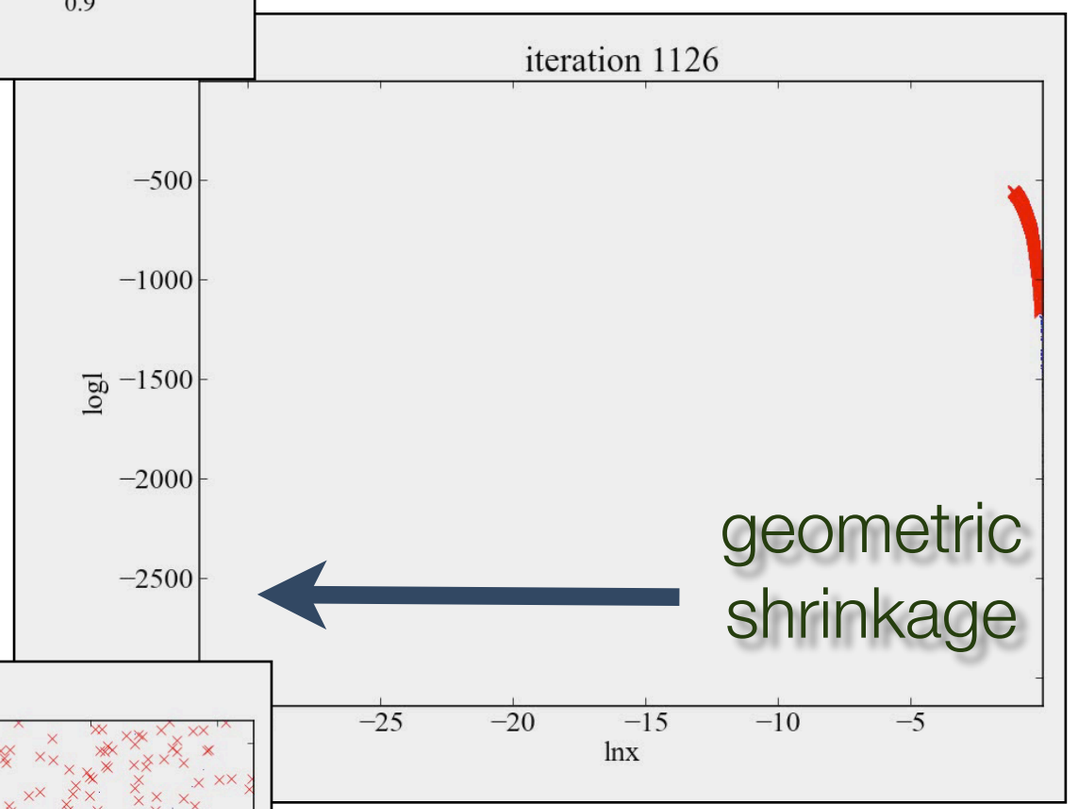




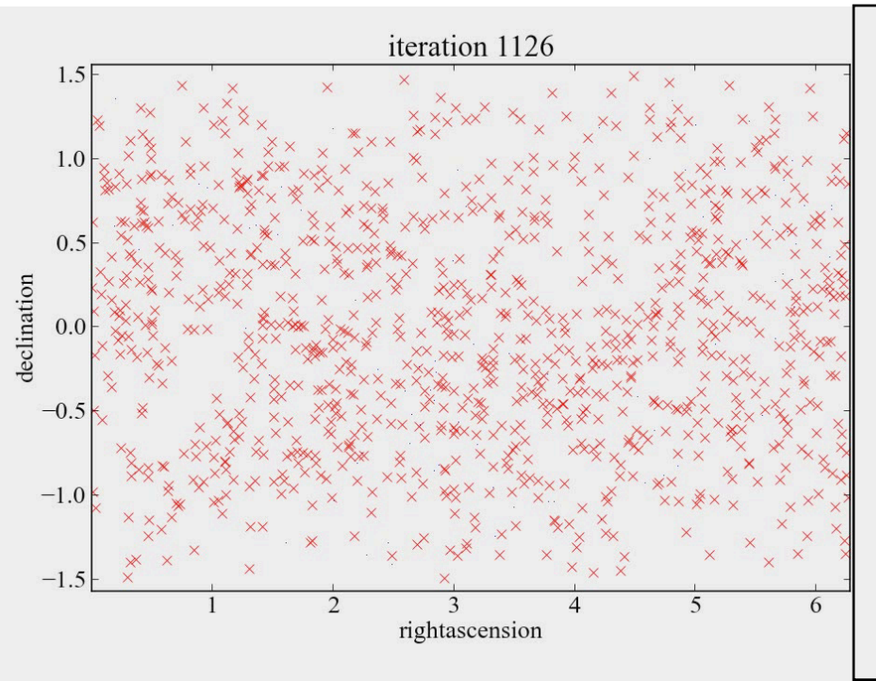
Live points (located inside a contour of constant likelihood)

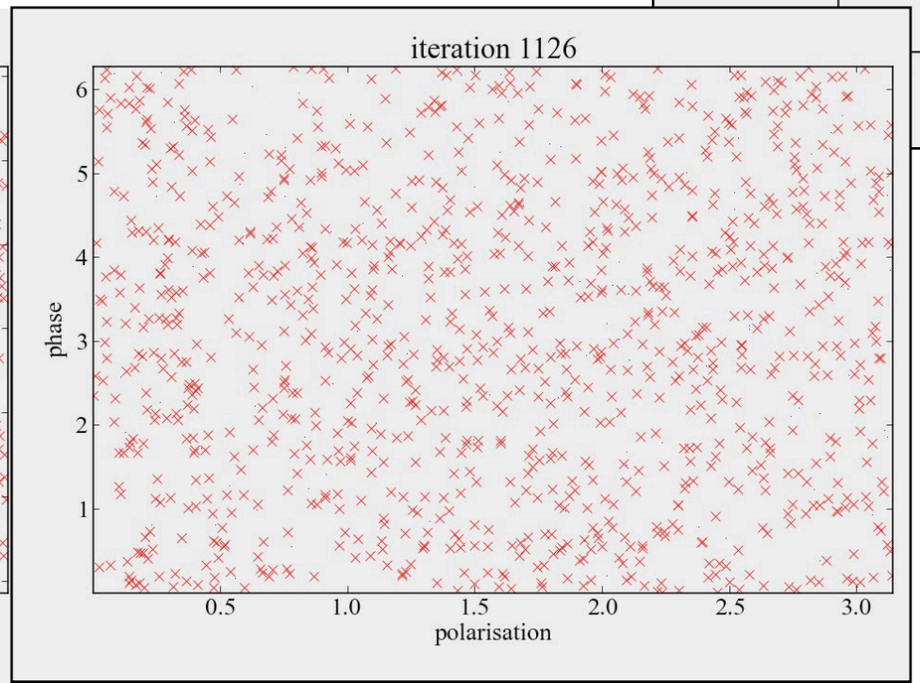
Used points

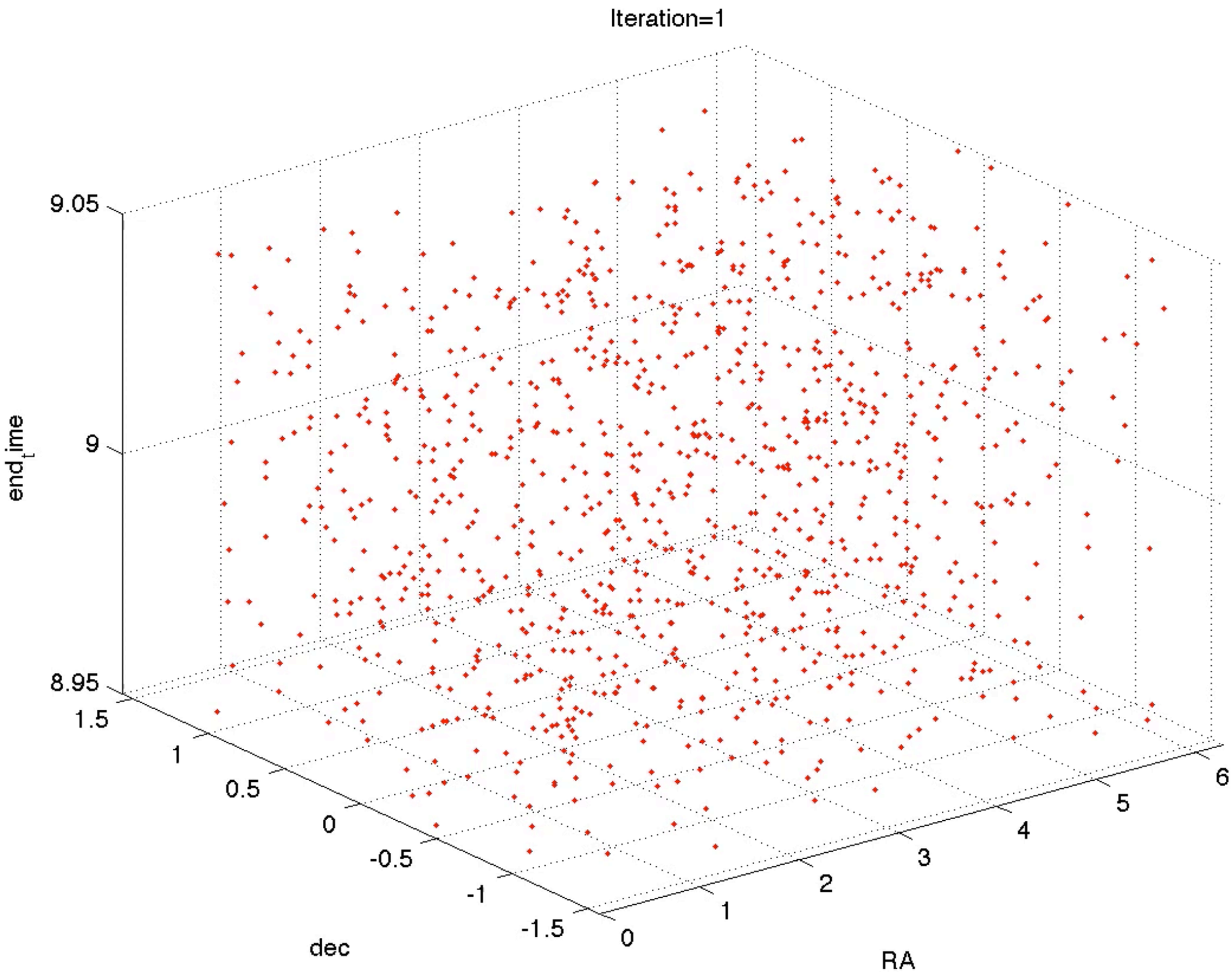
As the contours shrink, replacement samples are drawn from inside the contour using MCMC or rejection sampling



geometric shrinkage






Iteration=1
9.05
9
8.95
end_time
1.5
1
0.5
0
-0.5
-1
-1.5
dec
0
1
2
3
4
5
6
RA

Sampling limited prior

- Original nested sampling paper leaves sampling the limited prior as an exercise for the reader…
- MCMC?
  - LALInferenceNest, CPNest, DNest4, … can use Ensemble sampling using live points for information
  - Can re-use experience & custom proposals from MCMC methods (see Vivien's talk)
- Rejection sampling?
  - MultiNest (clustering of live points + ellipsoid proposal)
- Something fancier?
  - e.g. Hamiltonian Monte Carlo applied to prior ("Galilean Monte Carlo", Skilling 2012)

Information

- Quantify amount of information gained from data *d* as the Kullbeck-Liebler (KL) divergence between the prior and posterior

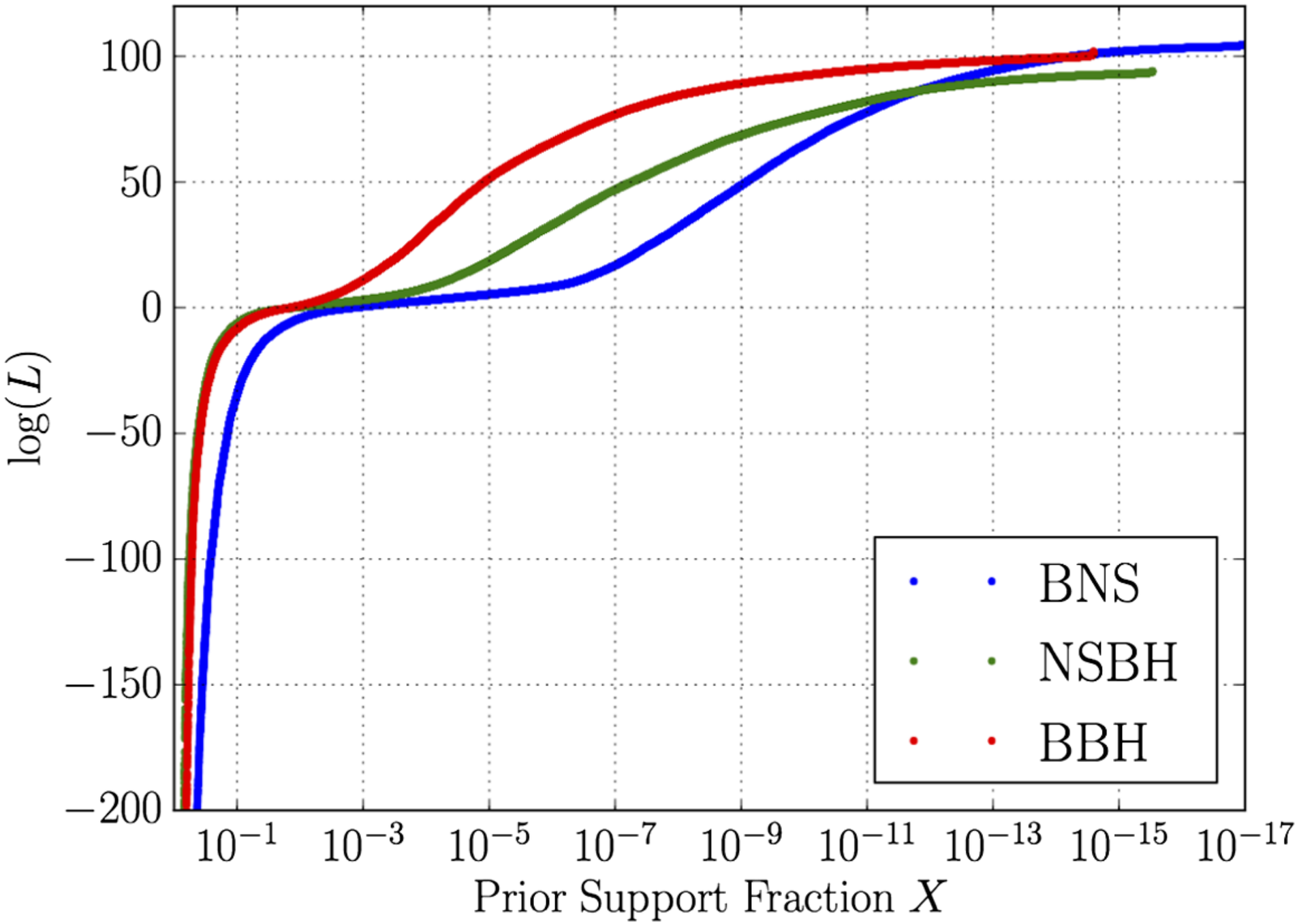
$$H = \int d^n\theta \, p(\theta|d) \log \frac{p(\theta|d)}{p(\theta)}$$

- Units are bits ($\log_2$) or nats ($\log_e$)

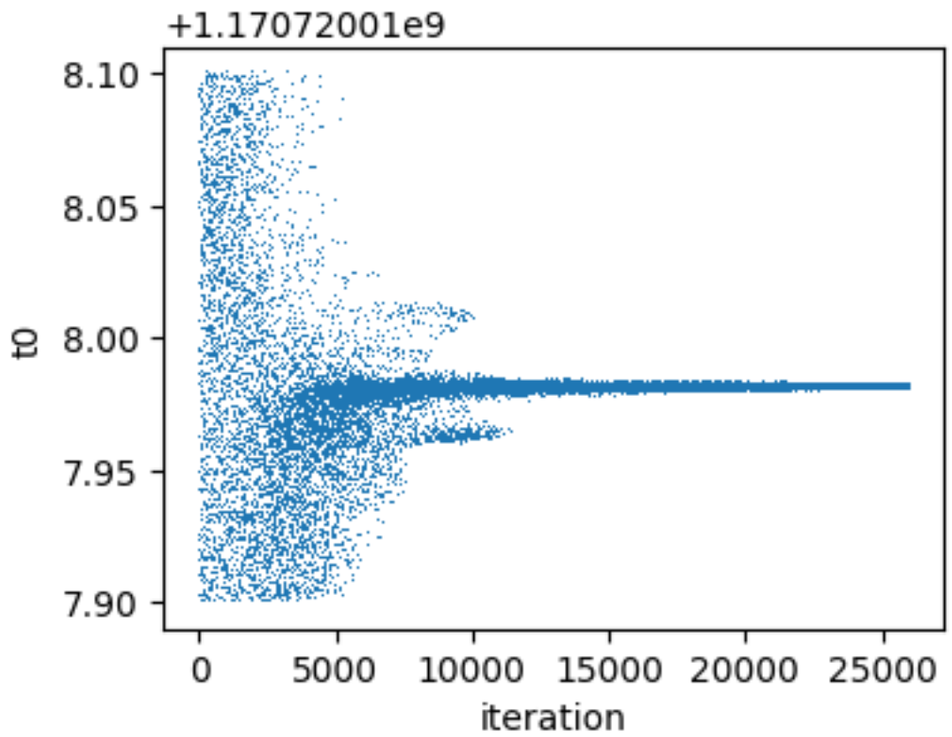
- Measures the shrinkage required to find the posterior!

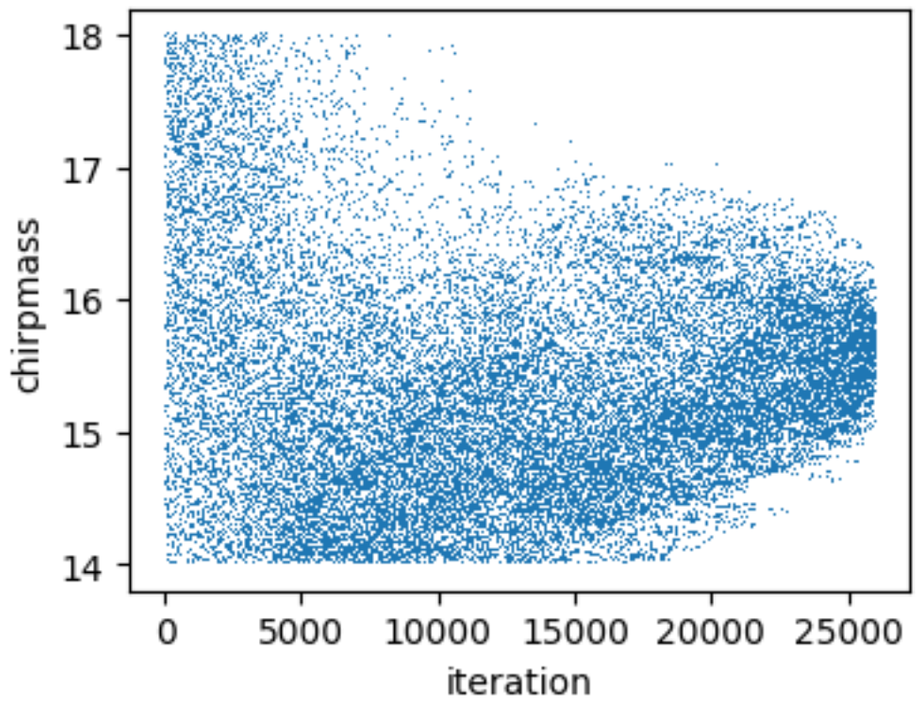
Example CBCs

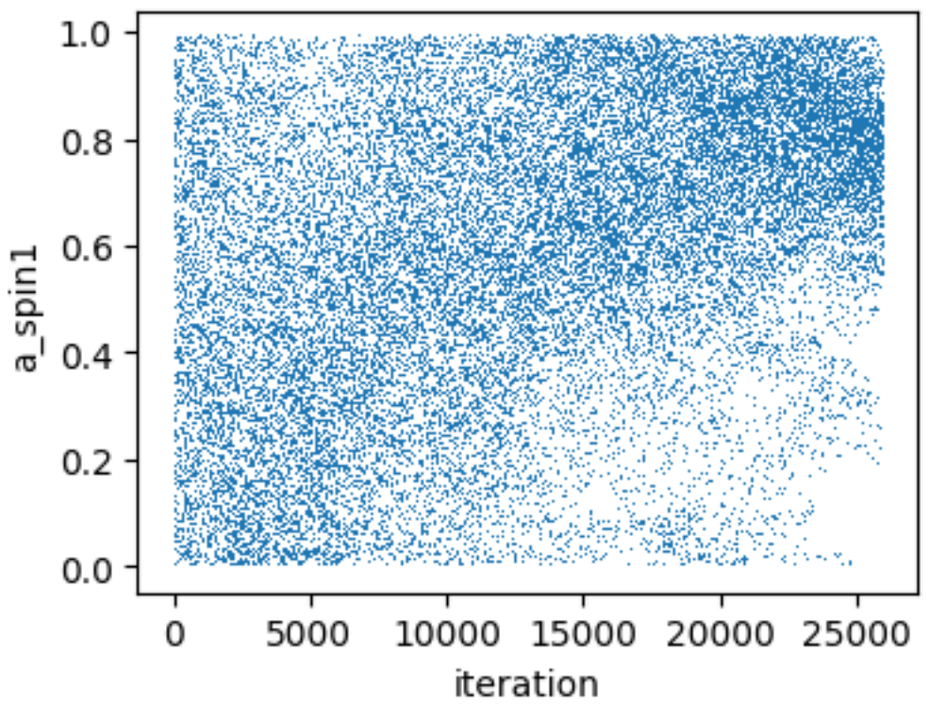
BBH easier to sample than BNS!



Trace plots







trace plots for nested sampling show samples covering to peak

parameters “lock in” earlier if there is more information about them

Convergence Time

- How long does it take to find the likelihood mode?
- Geometric shrinkage proceeds as $X_i \sim \exp\left(-\frac{i}{N}\right)$
- Information tells us that the peak is found at $X_{peak} \sim \exp\left(-H\right)$

- So the peak is reached at iteration $i \sim NH$
- Run time scales with information and number of live points
  - Information scales with ~SNR (at high SNR)

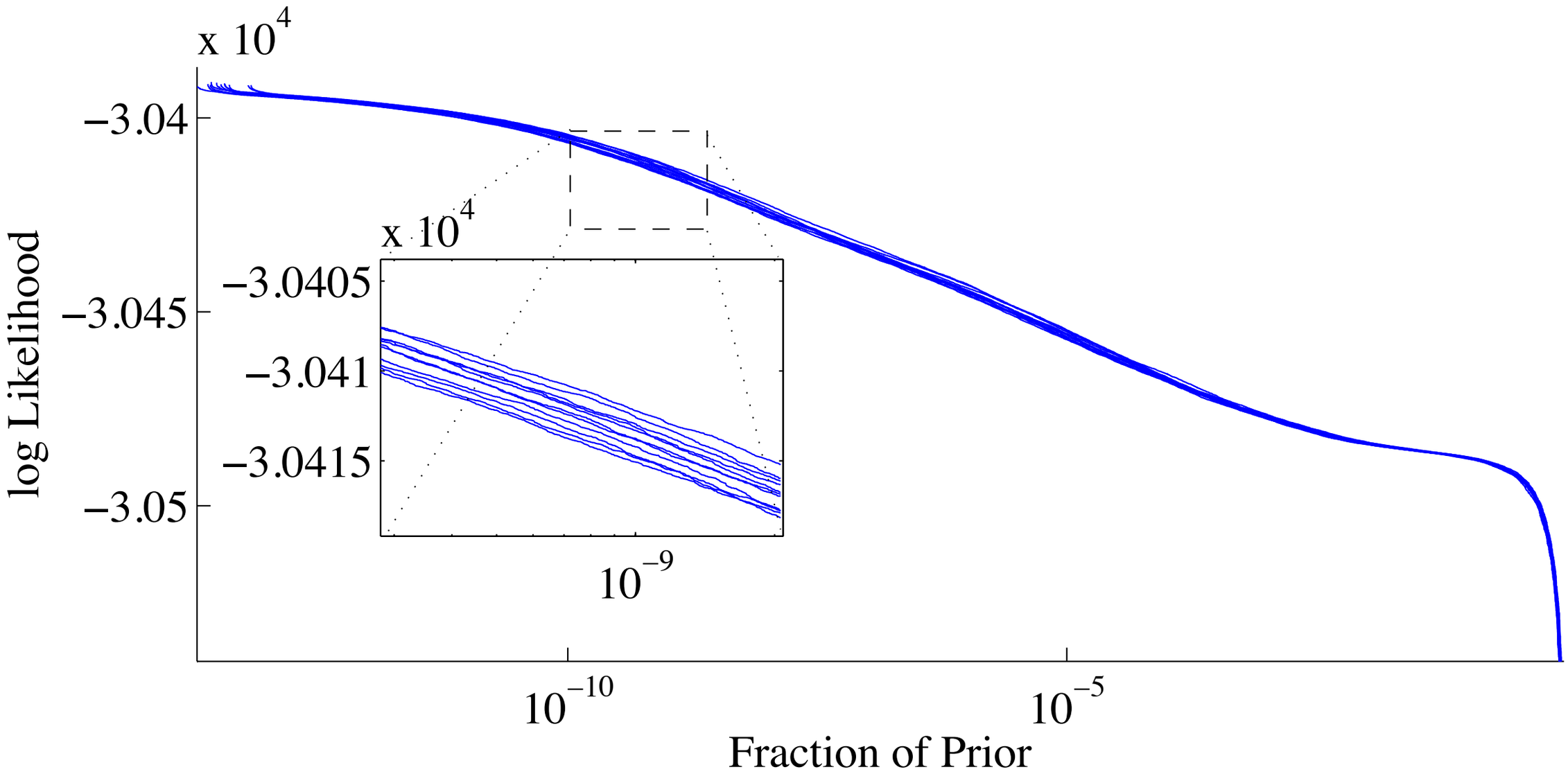
Estimation Error

- Recall that the weight at iteration *i* is $\log X_i \approx -i/N$
- The number of steps required to reach the peak is subject to Poisson errors $\Delta i_{peak} \approx \sqrt{NH}$
- So the error in the estimation of the size of the posterior mode goes like

$$\log X_{peak} \approx -\frac{NH}{N} \pm \frac{\sqrt{NH}}{N} = -H \pm \sqrt{H/N}$$

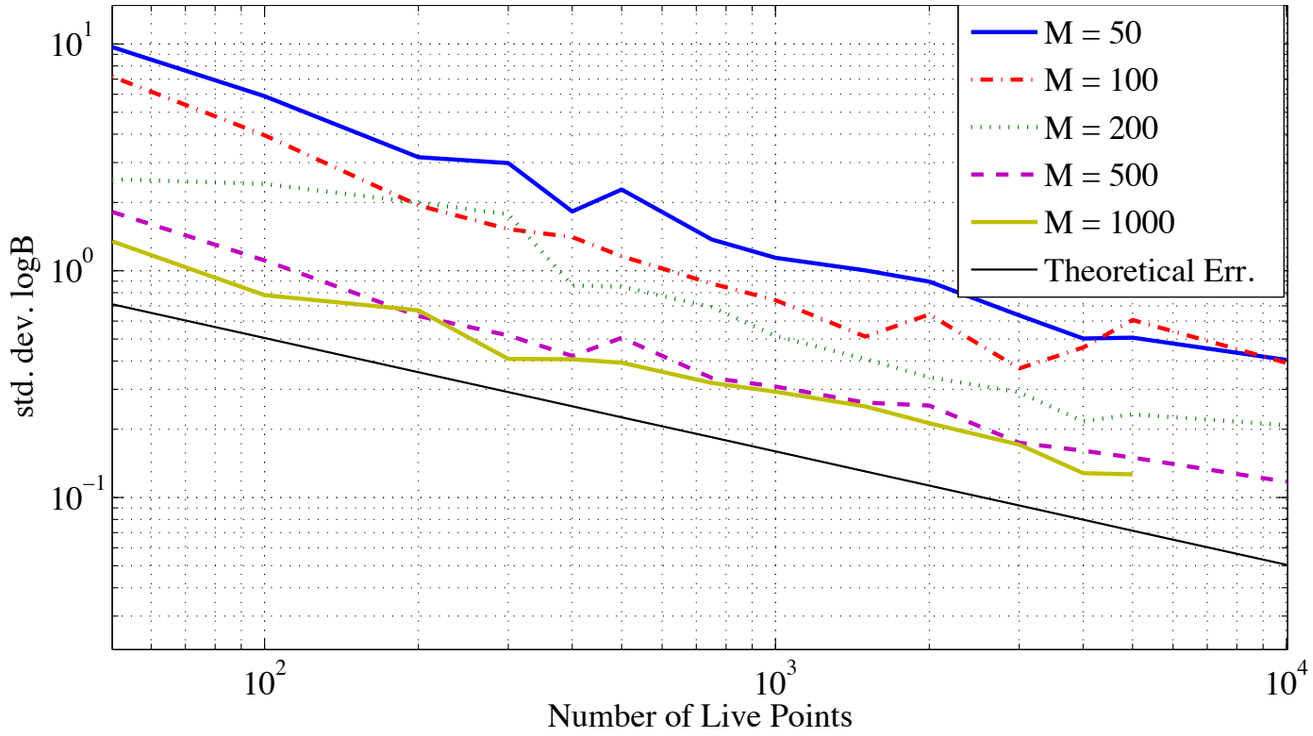
- This translates directly into the fractional error on logZ
- Errors are log-normal with (log) std. dev. $\Delta \log Z \approx \sqrt{H/N}$

Example: Accuracy vs runtime



Estimation Error in Practice

- 50 runs with varying live points N and MCMC steps M
- Compare to theoretical √(H/N)
- Theoretical error not obtained, but increasing MCMC points helps.
- For CBCs, allow up to 5000 MCMC steps



Termination

- At any time we have N samples available. Use these to estimate remaining evidence. $Z'_i = L_{\max} X_i$
- Upper bound on logZ inside current contour is
- Can set condition that upper bound on remaining evidence will not change estimate by >~10%

$$Z'/Z \leq 0.1$$

- LALInference, CPNest use this as a default

$$\log \frac{Z + Z'}{Z} \leq 0.1$$

Posterior Samples

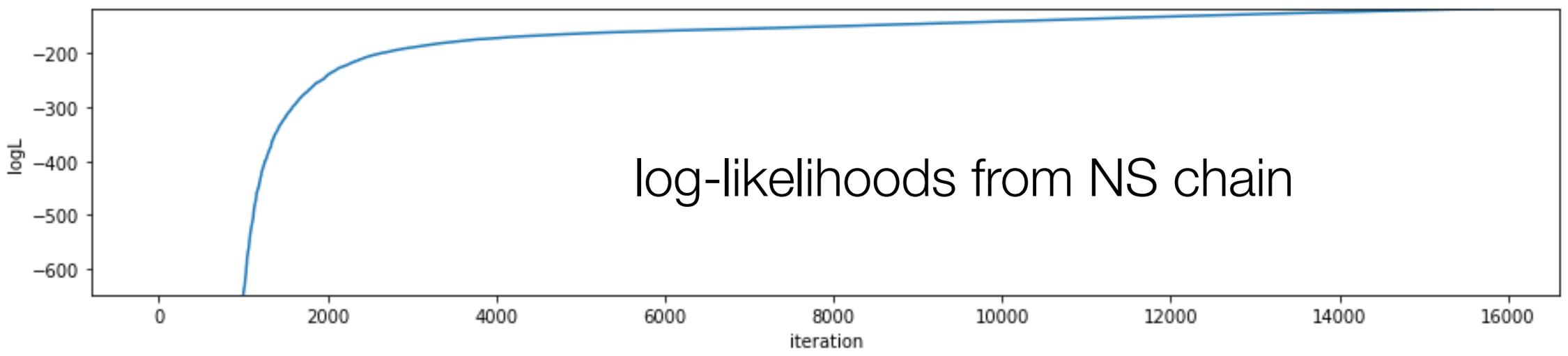
- Nested sampling saves the entire chain and likelihoods sampled at different X positions.

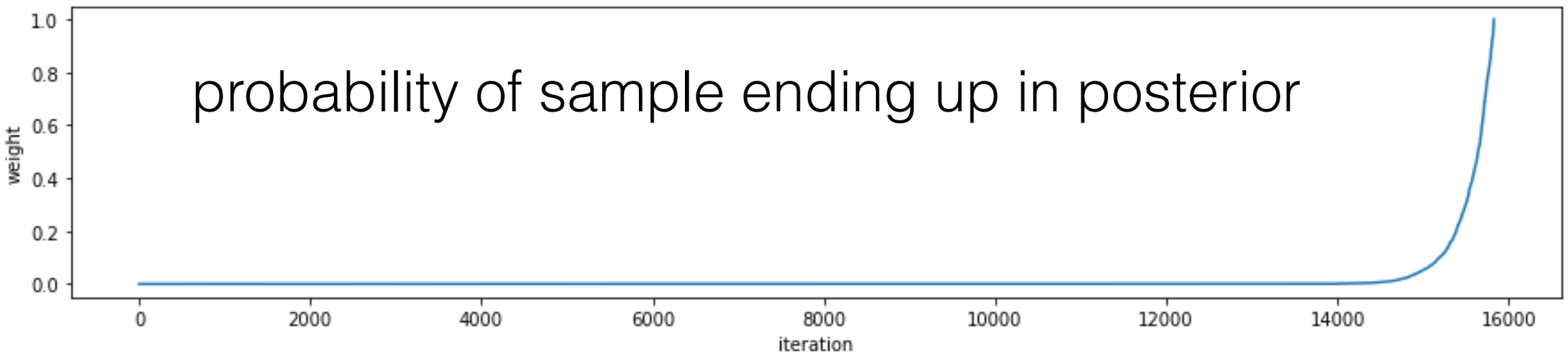
- NS sampling density: $p_{\rm NS}(i) = p(\theta_i|H)/X_i$

- Can obtain posterior samples by importance sampling from chain $p_{\rm pos}(i) = Z^{-1}p(\theta_i|H)L(d|\theta_i,H)$
$= Z^{-1}X_i p_{\rm NS}(i)L(d|\theta_i,H)$

- Weights $\alpha_i = \frac{p_{\rm pos}}{p_{\rm NS}}$
$= Z^{-1}X_i L_i$

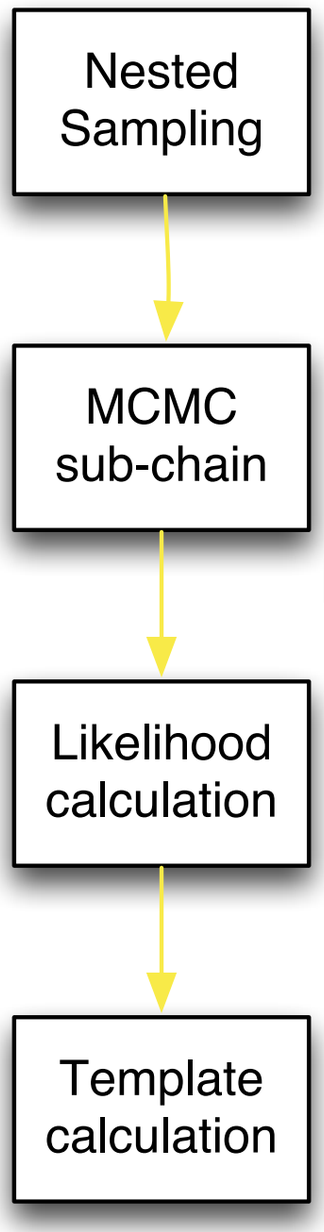
Posterior samples





Nested sampling does not produce posterior samples from an interrupted chain!

Computational Cost



- Run-time is dominated by
  - Template calculation (60-99% run-time)
    - including FFT for time-domain signals (rarely used in practice)
    - scales as length of template x sampling frequency
  - Overlap calculation (remainder of run-time)
    - Once per detector
    - Uses accelerated trigonometry approximation
- Reduce number of likelihoods needed
- Parallelise multiple runs across cores to reduce wall clock time
- Improve computational efficiency of vector ops (SIMD extensions)
- Use of custom hardware? GPUs, Xeon Phi, …

Optimisation Tricks

- LALInferenceNest uses a couple of tricks
  - *Adaptive MCMC length*: Measure autocorrelation length of the prior sampler and adjust during run
  - *Sloppy sampling*: Not every draw from prior has its likelihood evaluated. Allow chains to evolve before testing if they have passed through likelihood contour.
- CPNest currently has chain length adaptation, need to add sloppy sampling.

Implementations and Extensions

- LALInferenceNest (CBC-specific application) - Veitch, Raymond, Farr, …
  - arXiv:1409.7215
- CPNest (parallel python nested sampling) - Del Pozzo & Veitch
  - https://github.com/johnveitch/cpnest
- MultiNest & derivatives (polychord etc) - Feroz
  - https://github.com/farhanferoz/MultiNest
- Diffusive nested sampling (DNest4) - Brewer
  - https://github.com/eggplantbren/DNest4
- Dynamic nested sampling
  - https://github.com/joshspeagle/dynesty

Pitfalls & tips

- Reducing number of live points below 512 is a false economy.
    - Runs may actually take longer as sampling the limited prior becomes more difficult the smaller the number of live points!
- My recommendations for CBC:
    - Live points: 1024 or 2048 (precessing or weird events)
    - max-MCMC: at least 5000
    - Parallel jobs: As many as you can afford
- You will get few x N_{live} posterior samples back (depends on specific problem) per parallel run.

CPNest example

```
10961: n: 690 NS_acc:0.998 S0_acc:0.054 sub_acc:0.025 H: 11.56 logL 225.70730 --> 227.50077 dZ: 17.818 logZ: 213.060 logLmax: 241.84
10962: n: 689 NS_acc:0.998 S0_acc:0.054 sub_acc:0.026 H: 11.56 logL 225.71772 --> 230.92027 dZ: 17.812 logZ: 213.065 logLmax: 241.84
10963: n: 687 NS_acc:0.998 S0_acc:0.054 sub_acc:0.023 H: 11.56 logL 225.71961 --> 238.31826 dZ: 17.805 logZ: 213.070 logLmax: 241.84
10964: n: 686 NS_acc:0.998 S0_acc:0.054 sub_acc:0.025 H: 11.56 logL 225.72177 --> 226.65341 dZ: 17.799 logZ: 213.076 logLmax: 241.84
10965: n: 684 NS_acc:0.998 S0_acc:0.054 sub_acc:0.006 H: 11.56 logL 225.72835 --> 231.46117 dZ: 17.793 logZ: 213.081 logLmax: 241.84
10966: n: 689 NS_acc:0.998 S0_acc:0.054 sub_acc:0.015 H: 11.56 logL 225.73008 --> 227.49128 dZ: 17.786 logZ: 213.087 logLmax: 241.84
10967: n: 689 NS_acc:0.998 S0_acc:0.054 sub_acc:0.022 H: 11.56 logL 225.73363 --> 234.15534 dZ: 17.780 logZ: 213.092 logLmax: 241.84
10968: n: 688 NS_acc:0.998 S0_acc:0.054 sub_acc:0.013 H: 11.56 logL 225.73429 --> 225.88189 dZ: 17.774 logZ: 213.097 logLmax: 241.84
10969: n: 689 NS_acc:0.998 S0_acc:0.054 sub_acc:0.028 H: 11.56 logL 225.74513 --> 226.89530 dZ: 17.767 logZ: 213.103 logLmax: 241.84
10970: n: 687 NS_acc:0.998 S0_acc:0.054 sub_acc:0.007 H: 11.56 logL 225.74953 --> 228.50366 dZ: 17.761 logZ: 213.108 logLmax: 241.84
10971: n: 690 NS_acc:0.998 S0_acc:0.054 sub_acc:0.016 H: 11.56 logL 225.75132 --> 233.34968 dZ: 17.755 logZ: 213.113 logLmax: 241.84
10972: n: 690 NS_acc:0.998 S0_acc:0.054 sub_acc:0.020 H: 11.56 logL 225.75311 --> 228.67873 dZ: 17.748 logZ: 213.119 logLmax: 241.84
```