Introduction to Markov Chain Monte-Carlo

Week 6+

Markov Chain Monte-Carlo

- 804 papers in astro-ph (in arXiv) which have MCMC in abstract. First used in 2000 in Astrophysics (astro-ph/0006401)

- Most widely used sampler is Emcee

  http://dan.iel.fm/emcee/current/

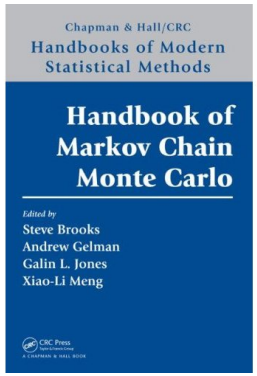
  Front end to Emcee also available at

  http://www.theonlinemcmc.com/ *(Not very robust and flexible and only very recently introduced)*

For R-based MCMC resources see PSU astrostats summer school notes by M. Haran

MCMC Resources

- See the literature on Bayesian analysis (mentioned in Lecture 6 notes)
- We shall follow Trotta's Jan 2017 review on Bayesian(arXiv:1701.01467)
- Another recent review by Hogg et al (arXiv:1710.06068)
- Another review (including history) by Sanjib Sharma (arxiv:1706.01629)
- https://jellis18.github.io/post/2018-01-02-mcmc-part1/
- https://arxiv.org/abs/1909.12313 A conceptual introduction to MCMC methods
- For more advanced stuff :



https://www.coursera.org/learn/statistical-mechanics

http://www.mcmchandbook.net/HandbookTableofContents.html

Basic idea behind MCMC

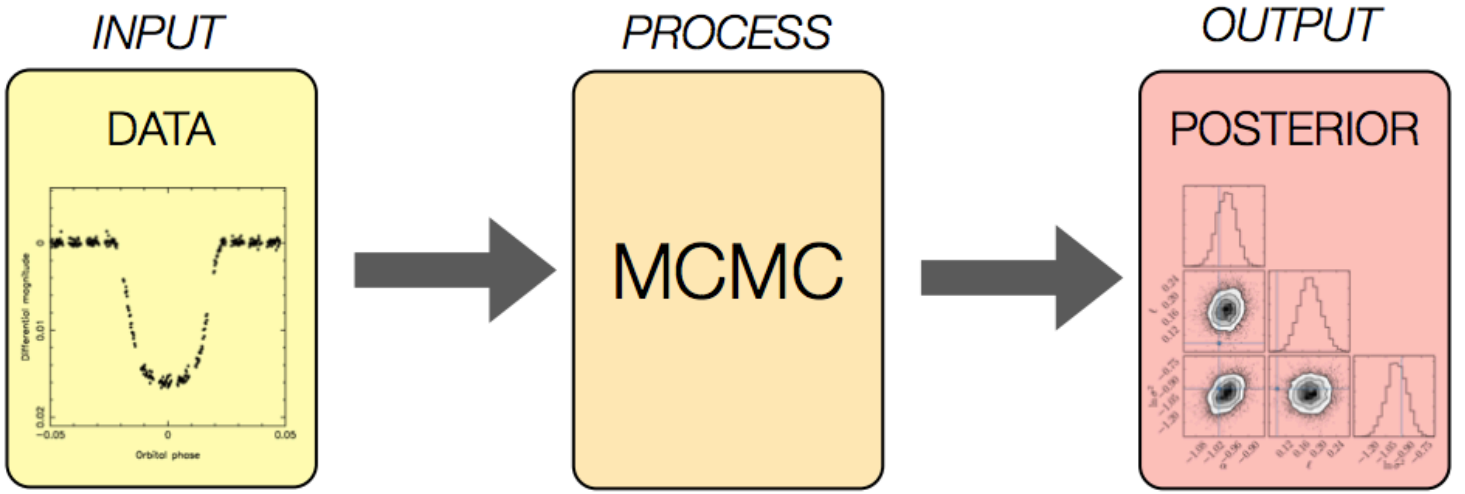
- Let's say we have a probability distribution (possibly in a very high dimension) and we want to sample from or find the expected value of it and doing so analytically is impossible.
- We can do this through a Monte Carlo method known as Markov Chain Monte Carlo:
  - The basic idea is that we perform a ``random walk'' through the probability distribution in question favoring values with higher probabilities.
  - i.e. once we have a starting point, we randomly pick a nearby point and evaluate its probability. If that probability is higher than the point we started on, we move there. Otherwise we simply stay put or move to that point with some probability.
  - The amazing thing is that if we repeat this enough under the right conditions we will hit every point in the space with a frequency proportional to its probability.

General Theory

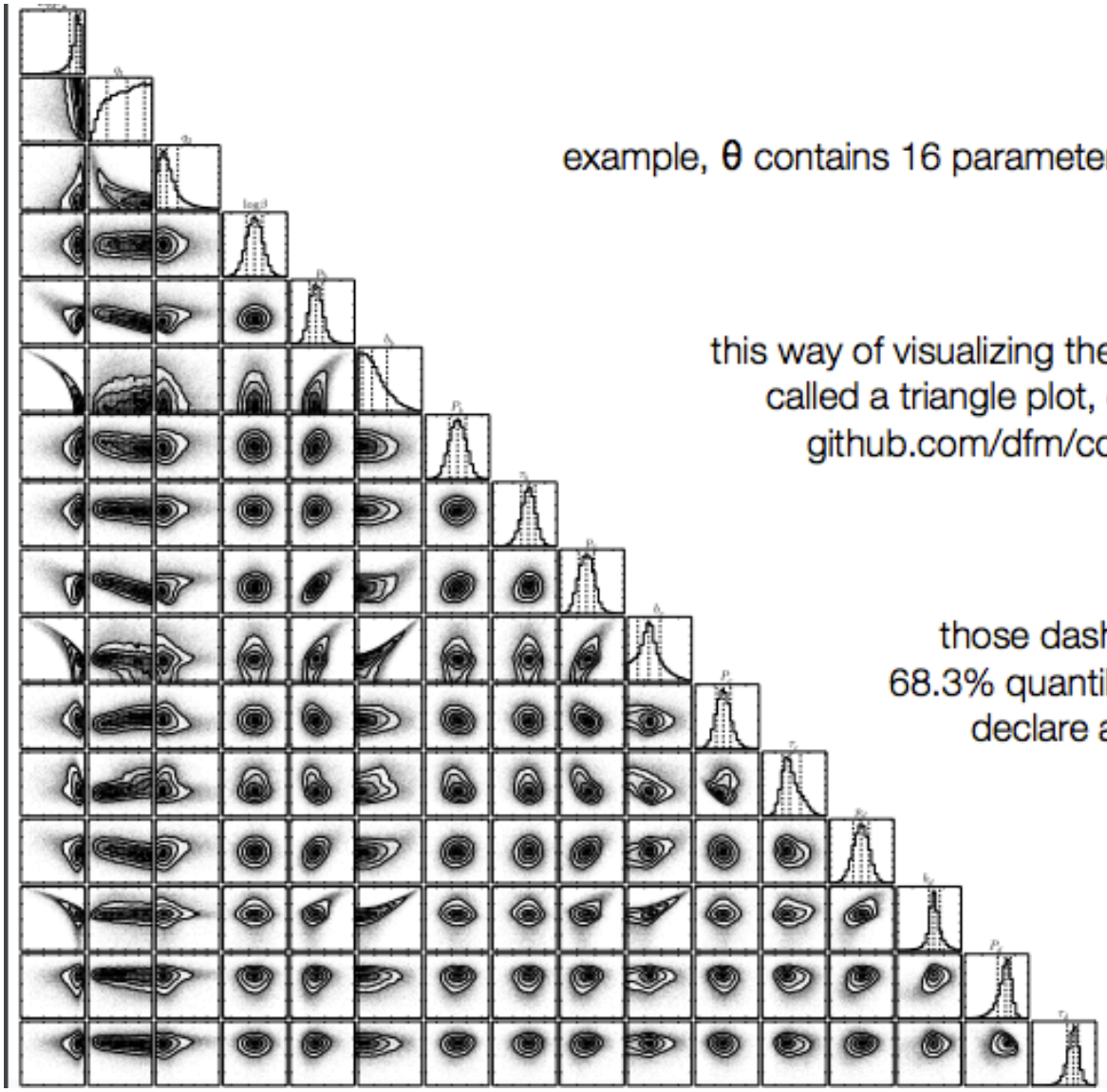
- Purpose of a Markov Chain Monte Carlo algorithm is to construct a sequence of points (``samples") in parameter space (called `` chain"). Density of samples is proportional to the posterior pdf, which allows us to construct a map of the posterior distribution.

- A Markov chain is a sequence of random variables $\{X^0, X^1, \ldots, X^{M-1}\}$ such that the probability of (t+1)-th element depends only on the value of the t-th element. They can be shown to converge to a stationary state , where successive elements of the chain are *samples from target distribution* (mostly the posterior).

What is the product of MCMC



and really by this I mean a set of samples from the posterior



example, θ contains 16 parameters

this way of visualizing the posteriors is called a triangle plot, check out github.com/dfm/corner.py

those dashed lines are the 68.3% quantiles, which we often declare as $\theta_1 = 2.0 \pm 0.1$

Kipping et al. (2016)

- Generation of elements of the chain is *probabilistic* in nature and is described by a *transition probability* $T(\theta^{(t)}, \theta^{(t+1)})$, giving the probability of moving from point $\theta^{(t)}$ to $\theta^{(t+1)}$ in parameter space. Sufficient condition to obtain a Markov chain is that the transition probability satisfy the *detailed balanced conditions.*

$$\frac{T(\theta^{(t)}, \theta^{(t+1)})}{T(\theta^{(t+1)}, \theta^{(t)})} = \frac{p(\theta^{(t+1)}|d)}{p(\theta^{(t)}|d)}$$

Ratio of transition probabilities is inversely proportional to the ratio of the posterior probabilities at the two points.

Posterior mean given by

$$E[\theta] = \int P(\theta|d)\theta d\theta \approx \frac{1}{M}\sum_{t=0}^{M-1}\theta^{(t)}$$

$$E[f(\theta)] \approx \frac{1}{M}\sum_{t=0}^{M-1} f(\theta^{(t)})$$

Calculation of Marginalized Posterior

- 1-dimensional marginalized probability for the j[th] element of θ, θ_j obtained by integrating out all other parameters from the posterior.

$$P(\theta_1|d) = \int P(\theta|d)d\theta_2.....d\theta_n$$

This can be easily calculated from the Markov Chain. Since elements of the Markov Chains are samples from the full posterior P(θ|d), their density reflects the value of the full posterior

Divide the range of θ_j in a series of bins and count the number of samples falling in each bin (ignoring the values for other coordinates)

Same with 2-d posterior.

Calculation of Credible Intervals

- A 1D 2-tail symmetric α % credible region is given by the interval within which α% of the samples are contained, obtained in such a way that a fraction (1- α)/2 of the samples lie outside the interval on either side.

(This is called symmetric credible interval)

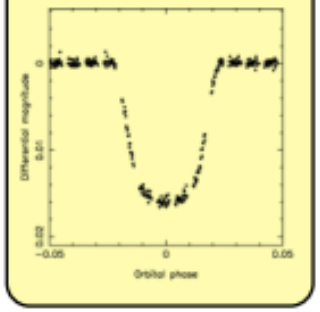
- In case of a upper (lower) limit, we report the value of the quantity below (above) which α% of the samples are to be found.

Role of MCMC Sampler

the sampler "guesses" different θ vectors, calculates the posterior probability of that guess, and then makes small jumps

INPUT

DATA



PROCESS

data

$\mathcal{D}$

model

θ $\mathcal{L}$

sampler

θ π

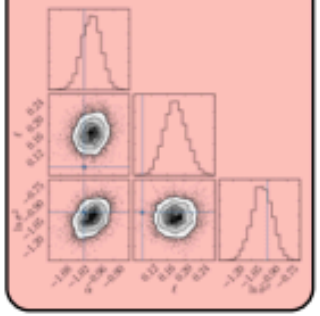
priors

MCMC

actually the point of the sampler is to make intelligent guesses with high posterior probabilities

OUTPUT

"POSTERIOR"



Metropolis-Hastings Algorithm

1. Start from a random point θ_0 having associated posterior probability $p_0 \equiv p(\theta^{(0)}|d)$

2. Propose a candidate θ^c from the proposal distribution $q(\theta_0, \theta^c)$. The proposal distribution might be for example a Gaussian of fixed $\boldsymbol{\sigma}$ centered around the current point. For the Metropolis algorithm, distribution q satisfies the symmetry condition q(x,y) = q(y,x)

3. Calculate the posterior at the candidate point $p_c = p(\theta^c \mid d)$. Accept the candidate point with probability

$$\alpha = min\left(\frac{p_c q(\theta^{(c)}, \theta^{(o)})}{p_o q(\theta^{(o)}, \theta^{(c)})}, 1\right) \quad \Rightarrow \quad \alpha = min\left(\frac{p_c}{p_o}, 1\right)$$ for Metropolis algo.

4. Generate a uniform random number u from the uniform distribution [0,1) and accept the candidate sample if $u < \alpha$ and reject it otherwise.

5. If the candidate point is accepted, add it to the chain and move there. Otherwise stay at the old point (which is counted twice in the chain). Go back to 2.

- if $P_c > P_o$, the candidate is always accepted
- Only the un-normalized posterior is required (since normalization constant drops out of the ratio)
- Metropolis algorithm satisfies the detailed balance condition.
- Affine invariant ensemble samplers evolve a series of `` walkers'' to generate a move with greatly reduced correlation length. This is used in `emcee`
- Other sampling methods such as Gibbs sampling used.

M-H Algorithm described in 2017 Hogg review

The M–H MCMC algorithm requires two inputs. The first is a handle to the function $f(\theta)$ that is the function to be sampled, such that the algorithm can evaluate $f(\theta)$ for any value of the parameters θ. In data-analysis contexts, this function would be the prior $p(\theta)$ times the likelihood $p(D\,|\,\theta)$ evaluated at the observed data D. The second input is a handle to a proposal pdf function $q(\theta'\,|\,\theta)$ that can deliver samples, such that the algorithm can draw a new position θ' in the parameter space given an "old" position θ. This second function must meet a symmetry requirement (detailed balance) we discuss further below. It permits us to random-walk around the parameter space in a fair way.

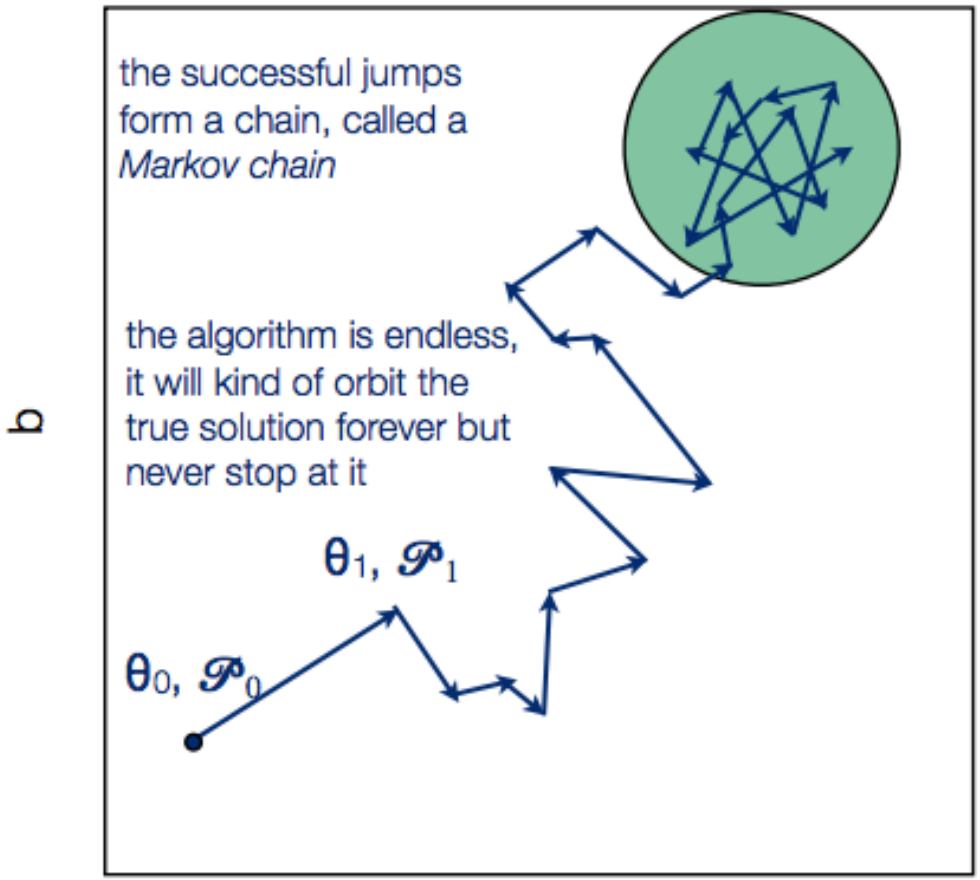
The algorithm[14] is the following: We have generated some set of samples, the most recent of which is θ_k To generate the next sample θ_{k+1} do the following:

- Draw a proposal θ' from the proposal pdf $q(\theta'\,|\,\theta_k)$.
- Draw a random number $0 < r < 1$ from the uniform distribution.
- If $f(\theta')/f(\theta_k) > r$ then $\theta_{k+1} \leftarrow \theta'$; otherwise $\theta_{k+1} \leftarrow \theta_k$.

That is, at each step, either a new proposed position in the parameter space gets accepted into the list of samples or else the previous sample in the parameter space *gets repeated*. The algorithm can be iterated a large number K of times to produce K samples.

Why does this algorithm work? The answer is not absolutely trivial[15], but there are two components to the argument: The first is that the Markov process delivers

arXiv:1802.07683



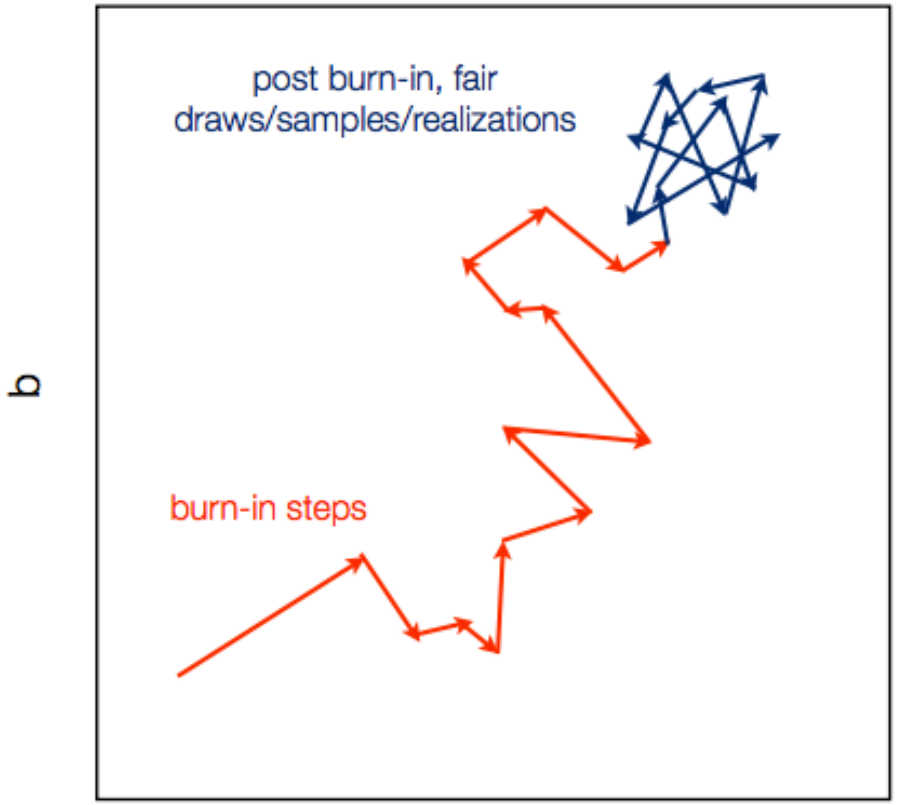
high $\mathscr{P}$ region

METROPOLIS RULE

if $\mathscr{P}_{trial} > \mathscr{P}_i$,
accept the jump, so
$\theta_{i+1} = \theta_{trial}$

if $\mathscr{P}_{trial} < \mathscr{P}_i$,
accept the jump with probability $\mathscr{P}_{trial}/\mathscr{P}_i$

5. keep jumping!
6. after you've done many steps, remove burn-in steps

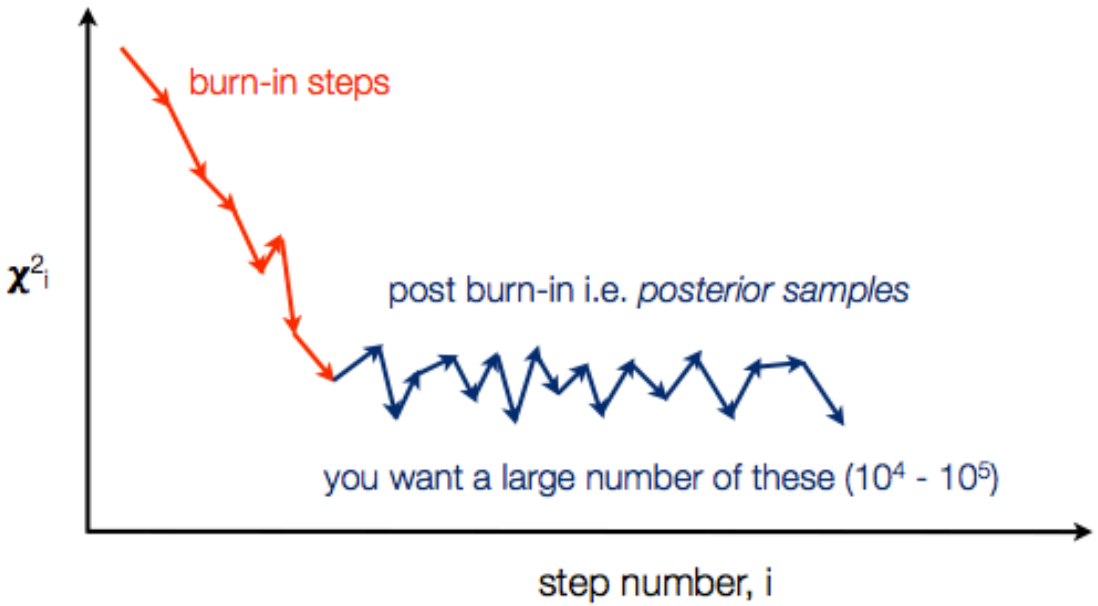


METROPOLIS RULE

if $\mathscr{P}_{trial} > \mathscr{P}_i$,
accept the jump, so
$\theta_{i+1} = \theta_{trial}$

if $\mathscr{P}_{trial} < \mathscr{P}_i$,
accept the jump with
probability $\mathscr{P}_{trial}/\mathscr{P}_i$

burn-in point can be spotted by eye...



Practical and Numerical Issues

- Initial Samples from the Chain must be discarded, since the Markov process is not yet sampling from the equilibrium distribution (so called *burn-in period*)
- Assessment of chain convergence, which aims at establishing whether the Monte Carlo estimate is sufficiently accurate. Some tools include Gelman and Rubin criterion and Raftey/Lewis statistic.
- Make sure it does not get trapped in local minima
- Successive samples in a chain are correlated. Calculate the auto-correlation, which is a good measure of the number of steps required before chain has ``forgotten'' its previous state.

Successive samples in a chain are correlated. Calculate the auto-correlation, which is a good measure of the number of steps required before chain has ``forgotten'' its previous state.

Define $$\hat{\gamma} = \frac{\sum_{i=0}^{M-k} (\theta_i - \bar{\theta})(\theta_{i+k} - \bar{\theta})}{\sum_{i=0}^{M-k} (\theta_i - \bar{\theta})^2}$$

where k is called lag and $\bar{x}$ the sample mean and a plot of $\hat{\gamma}$ versus lag k Is called the auto-correlation function and the value of the lag after which it drops to 0 provides an estimate of the thinning factor *K* required to obtain independent samples from the chain.

- For more information on how to check convergence of MCMC, check out http://astrostatistics.psu.edu/RLectures/diagnosticsMCMC16.pdf

- Subroutine for calculating Gelman-Rubin Convergence Criterion in Emcee http://joergdietrich.github.io/emcee-convergence.html

Tools for Calculating Bayesian Evidence

- https://ccpforge.cse.rl.ac.uk/gf/project/multinest/

based on Nested Sampling algorithm (check out wikipedia)
Details on MultiNest in https://arxiv.org/abs/0809.3437

https://dynesty.readthedocs.io/en/latest/dynamic.html
(see also arXiv:1907.07199 Krishak & SD)

Youtube lectures on nested sampling (at AstroHackWeek 2015@NYU)
https://www.youtube.com/watch?v=5bO2JwsJY1M

Also check out