Markov Chain Monte Carlo (MCMC) applications in astronomy

DAWGI guest lecture 28/01/2021

Peter Scicluna

This lecture is accompanied by an example sheet and problem set in a Jupyter notebook.
You can download the notebook from:
It is intended to provide a reminder of the topics discussed in the lecture, but you may also find it useful to refer to it in parallel for some sections.
I recommend using Google Colab to complete the problem set in the cloud - to do so, upload the notebook to … and run it there.

Important reminder: Bayesian inference

Frequentist inference optimises something like the likelihood $P\left(D|M\right)$ but this gives opposite of what we want $P\left(M|D\right)$

From Bayes' theorem we can convert from one thing to the other

$$P\left(M|D\right) = \frac{P(D|M)P(M)}{P(D)}$$

Some jargon:

$P\left(D|M\right)$ - the *Likelihood*;
$P\left(M|D\right)$ - the *Posterior*, what we want to infer;
$P\left(M\right)$ - the *Prior*, encapsulates existing knowledge about the problem;
$P\left(D\right)$ - the *Evidence*, often treated as a normalisation term, encapsulates the average likelihood.

Important reminder: Bayesian inference

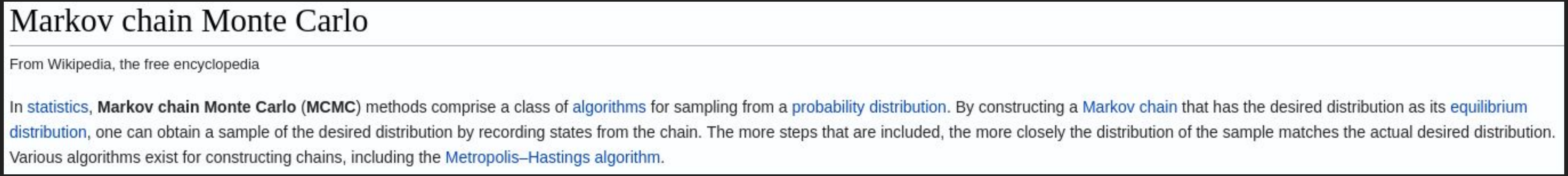
So we want to find models that maximise the posterior, and to understand the shape of the posterior.

Some other important jargon:

~~Confidence interval~~ Credible interval: region that contains N% of the probability

Support: another term for the prior volume (domain where function is non-zero). But this also has another meaning for MCMC - how much of the prior volume is explored during the run.

What is MCMC?



Markov chain Monte Carlo

From Wikipedia, the free encyclopedia

In statistics, **Markov chain Monte Carlo** (**MCMC**) methods comprise a class of algorithms for sampling from a probability distribution. By constructing a Markov chain that has the desired distribution as its equilibrium distribution, one can obtain a sample of the desired distribution by recording states from the chain. The more steps that are included, the more closely the distribution of the sample matches the actual desired distribution. Various algorithms exist for constructing chains, including the Metropolis–Hastings algorithm.

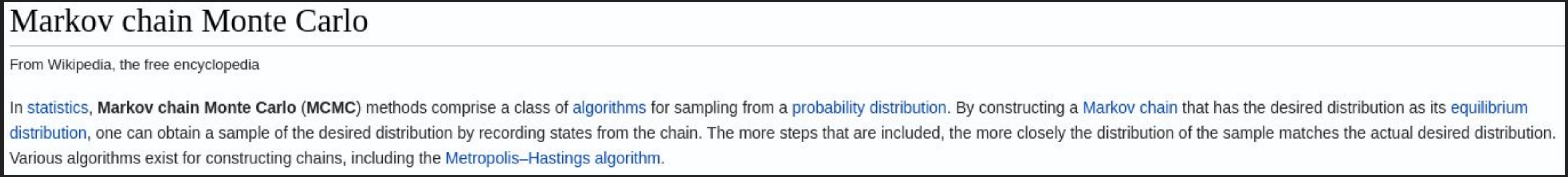
Skip the theory - what does it do and how can we use it for inference?

MCMC takes pseudo-random steps through a space to produce samples from a probability distribution, and can therefore be used to integrate the distribution

If the space is the parameter space for a physical model, we can therefore integrate the posterior to:

- Understand shape of the distribution
- Estimate moments of the distribution, e.g. expectation (mean/median), (co-)variances

What is MCMC?



Markov chain Monte Carlo

From Wikipedia, the free encyclopedia

In statistics, **Markov chain Monte Carlo** (**MCMC**) methods comprise a class of algorithms for sampling from a probability distribution. By constructing a Markov chain that has the desired distribution as its equilibrium distribution, one can obtain a sample of the desired distribution by recording states from the chain. The more steps that are included, the more closely the distribution of the sample matches the actual desired distribution. Various algorithms exist for constructing chains, including the Metropolis–Hastings algorithm.

However, MCMC doesn’t care about absolute normalisation, just relative differences. Therefore, it is only capable of integrating something which is *proportional to* the target distribution.

MCMC takes pseudo-random steps, but because each new step depends on the previous one, steps are *correlated*.

Why is MCMC useful?

Sample distributions - what’s better, $x = 5 \pm 1$ or [plot goes here]?

Why is MCMC useful?

Sample distributions - what’s better, $x = 5 \pm 1$ or [plot goes here]?

Understand shape of parameter space (posterior distribution!)

Why is MCMC useful?

Sample distributions - what's better, $x = 5 \pm 1$ or [plot goes here]?

Understand shape of parameter space (posterior distribution!)

Correlations between parameters

Why is MCMC useful?

Sample distributions - what’s better, $x = 5 \pm 1$ or [plot goes here]?

Understand shape of parameter space (posterior distribution!)

Correlations between parameters

Marginalisation! - handle extra parameters that you’re not interested in.

Why is MCMC useful?

Sample distributions - what’s better, $x = 5 \pm 1$ or [plot goes here]?

Understand shape of parameter space (posterior distribution!)

Correlations between parameters

Marginalisation! - handle extra parameters that you’re not interested in.

- Integrate over plausible prior distribution of a parameter
- →Propagate uncertainty to posteriors of other parameters

Different MCMC approaches

Algorithm	Pros	Cons	Example implementations
Metropolis-Hastings	Simple to implement	*Very* sensitive to step size, poor support	PyMCMC
Slice sampling	Automatically adjusts step size, improved support	Distribution must be evaluable	Zeus
Affine-invariant MCMC	Embarrassingly parallel, good for parameters with very different ranges, handles covariant distributions well	Sensitive to start point, struggles with multimodality, non-optimal support	emcee
Hamiltonian Monte Carlo (HMC)	Rapid convergence, good with high dimensionality	More overhead per step, usually needs gradients	PyMC3
No U-turn Sampler (NUTS)	Rapid convergence, good with high dimensionality	More overhead per step, usually needs gradients	STAN, PyMC3
Parallel Tempering	Embarrassingly parallel,	Sensitive to hyperparameters	ptemcee

Some related approaches

Algorithm	Pros	Cons	Example implementations
Genetic algorithms	Population/ensemble sampling efficient, embarrassingly parallel	Sensitive to tunable hyperparameters, poor scaling	geneticalgorithm
Simulated Annealing	Good for finding approximate global solution	Similar problems to MH, non-trivial to parallelise	simanneal, scipy optimize.dual_annealing
Sequential Monte Carlo	Minimises function evaluations, embarrassingly parallel	Few points explored	SMCPy, particles, QInfer, PySMC
Particle swarm optimisation	Good for large solution spaces, no gradients required	Sensitive to hyperparameters, premature convergence	PySwarms
Nested Sampling	Computes evidence! Very good support, always finds global maximum in infinite time, handles multimodal posteriors well, fewer likelihood evaluations, no correlations between samples	More overhead per likelihood call, always does full global optimisation (can’t speed things up if we know roughly where to start from)	Nestle, MultiNest, UltraNest, Dynesty

How can we interpret the output?

Are results converged:

Trace plots

Autocorrelation

RG statistic

Is model appropriate:

Posterior predictive checks!

Prior sensitivity!

What are the results?

Corner plots! Summary statistics!

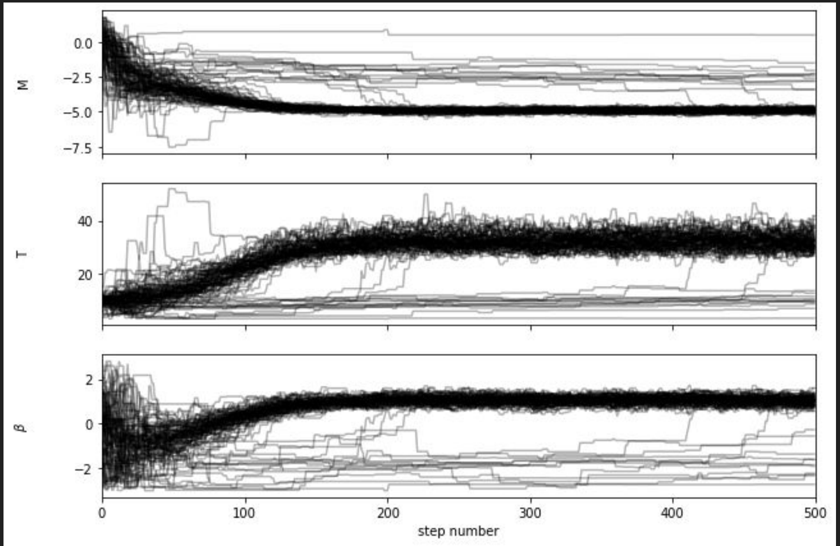
How can we interpret the output?

First question - are the chains converged?

How can we interpret the output?

First question - are the chains converged? Impossible to know for sure!

First step: Visual inspection!



This is called a *trace plot*!

Things to note: The first part of the plot indicates the time when the chains are not yet in equilibrium with the posterior. This phase is called *burn in*.

How can we interpret the output?

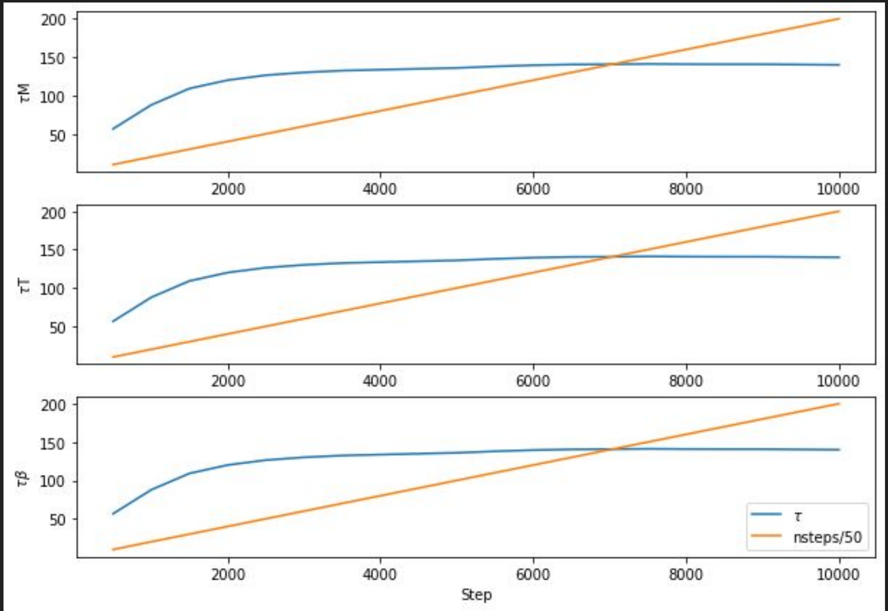
First question - are the chains converged? Impossible to know for sure!

Second step: Statistics! By design, consecutive samples are correlated.

Key questions: How far apart to samples have to be to be ~independent? Do we have enough independent samples to draw reasonable conclusions?

Autocorrelation time! If chain lengths are $\geq 10 \times \tau_{cor}$ then they are probably okay.

Beware! Estimating τ_{cor} is very tricky! Short chains might appear to be long enough but hide some long-period correlations!



How can we interpret the output?

First question - are the chains converged? Impossible to know for sure!

Second step: Statistics! By design, consecutive samples are correlated.

Key questions: How far apart to samples have to be to be ~independent? Do we have enough independent samples to draw reasonable conclusions?

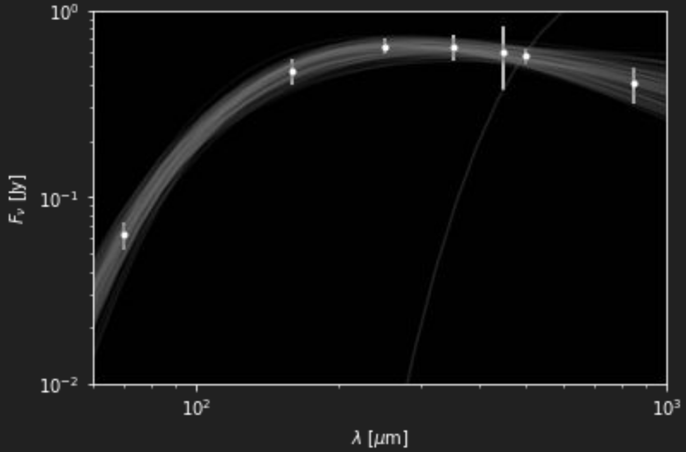
Rubin-Gelman statistic:

How can we interpret the output?

Second question - Does our model match the data well?

We’re Bayesian now - no statistic for absolute quality of fit, only better or worse than something else

Once again - Visual inspection!



These are one form of *posterior predictive checks,* an **essential step** in evaluating any MCMC.

Equivalent to inspecting residuals

How can we interpret the output?

Now we can ask, what are our results?

Summary statistics: point estimate and measure of credible interval $\Theta_1 = x\pm^{\delta x_{hi}}_{\delta x_{lo}}$

Point estimate: Mean, mode, or median of the chains? Or the *maximum a posteriori* (MAP) solution?

- Mode is difficult to determine, depends too much on user choices
- MAP should be similar to mode, but MCMC can only ever find approximate MAP
- Mean can be misleading for heavy-tailed distributions
- Median is well defined and always close to the region with the highest probability density.

Often worth reporting two of these numbers, most commonly the MAP and the median of the chains.

Choice is also linked to how you define the credible interval …

How can we interpret the output?

Now we can ask, what are our results?

Summary statistics: point estimate and measure of credible interval $\Theta_1 = x \pm^{\delta x_{hi}}_{\delta x_{lo}}$

Credible interval: Choice is also linked to how you define the point estimate!

We want to know the region which contains 68.3% (1 sigma) of the total probability mass

- Not unique!
- Some common choices:
    a. Symmetrical region around point estimate that contains *at least* 68.3%
    b. Highest-density region, i.e. narrowest region (in terms of parameter) that contains *exactly* 68.3%
    c. Central 68.3% i.e. 84.15% - 15.85% region (or 84th - 16th percentiles)
    d. …
- Median is always inside all choices, *but mean might not be*! (for options b and c)
- See Andrae et al. (2010) for more discussion of this and related issues.

How can we interpret the output?

What was the key thing that MCMC could tell us about?

How can we interpret the output?

What was the key thing that MCMC could tell us about?

The shape of the posterior distribution!

How can we interpret the output?

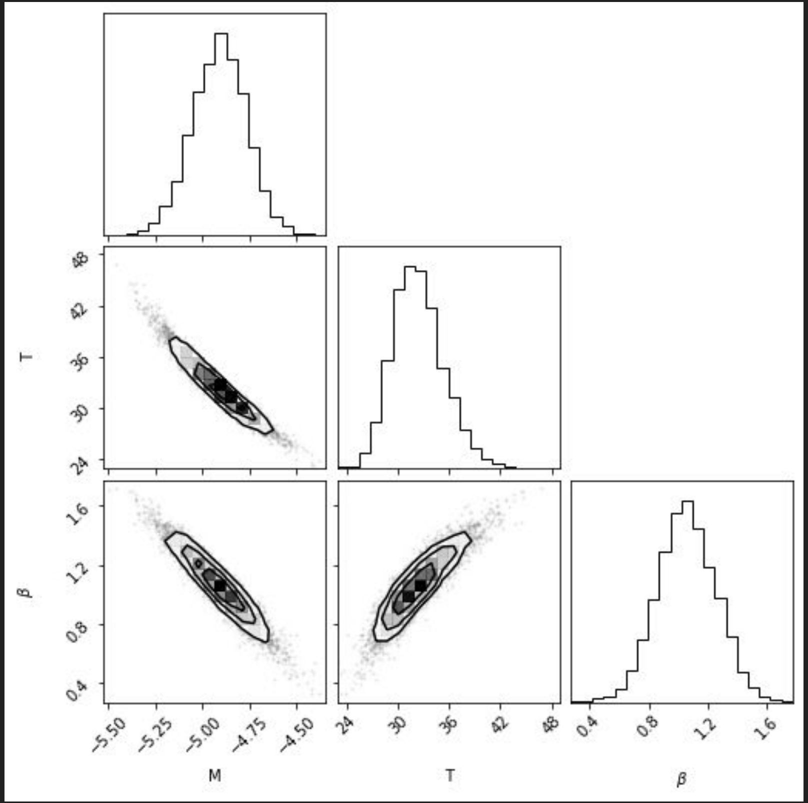
What was the key thing that MCMC could tell us about?
The shape of the posterior distribution!

Again, *visualisation is essential!*

- Are the parameter distributions monomodal?
- Are there correlations between parameters?
  - Plot the *marginal distributions*, often using a *corner plot*

Can also summarise these with numpy.cov() to estimate covariance matrix for parameters

Characterising distribution complex, but important to try different approaches.



How can we interpret the output?

One more thing we should always check:

Does our choice of prior unnecessarily affect the results?

This is known as *prior sensitivity testing*.

Basic idea:

- Re-run the MCMC with different choices for the prior
- Compare results
- If results change significantly (compared to the credible interval!) then the choice of prior matters
- Need to think carefully about the choice
  - Consider conjugate priors
  - Carefully consider available information on which to base the prior choice

Some demonstrations

Please see the example sheet:

Key take-home messages

- MCMC very powerful technique for inference
    - Characterise probability distributions for parameters of interest
    - Marginalise out nuisance parameters
- Lots of existing code in python, widely applied in astronomy
- Important to check convergence, but this is non-trivial
- Always do posterior predictive checks
- See examples and problem sheets for application

Tomorrow:
How to determine which of the models you tried reproduces the data the best?