# 1. MCMC Overview

In data analysis, as in many other aspects of (an admittedly sad) life, we are often faced with having to do high-dimensional integrals over likelihood functions. An example would be to calculate the expectation of a model parameter $\langle p_i \rangle = \int p_i \mathcal{L}(p) \mathrm{d}^n p / \int \mathcal{L}(p) \mathrm{d}^n p$. Similarly, we can calculate uncertainties on parameters by calculating their variances since $\mathrm{Var}(p_i) = < p_i^2 > - < p_i >^2$. If the likelihood is reasonably well described by a Gaussian, a Taylor expansion around the peak (found via something like Levenberg-Marquardt) can work very well.

For broad classes of problems, though, this can fail, and fail spectacularly. A classic example is a $\chi^2$ surface with multiple minima. LM is unlikely to notice the multiple minima, and its reported error bars will be derived from the curvature around the minimum it found. If you blindly use those error bars, you can underestimate your uncertainties by orders of magnitude. Evaluating the integrals on a grid of points rapidly becomes intractable, and so we will need another technique.

A very robust (and easy to code) tool that can handle complicated likelihood surfaces is Markov Chain Monte Carlo (MCMC). The downside is that it's hundreds of times slower than Newton's method or LM, and that's on a good day. The inspiration for MCMC comes from thermodynamics. I'll go through the motivation, but don't worry if you don't follow all the details. You'll still be able to happily and usefully use MCMC.

# 2. MCMC Derivation

We'll start with the seemingly unrelated problem of a two-state system with populations $n_1$ and $n_2$, and transition probabilities $p_{12}$ for a particle in state 1 to transition to state 2, and $p_{21}$ for the reverse. The time evolution of the system is then

$$\frac{\mathrm{d}n_1}{\mathrm{d}t} = -n_1 p_{12} + n_2 p_{21}$$

and similar for state 2. If we let the system come into equilibrium, then the derivative must be equal to zero, so we have

$$\frac{n_1}{n_2} = \frac{p_{21}}{p_{12}} \tag{1}$$

That is, the ratio of the population of the two states is equal to the ratio of their transition probabilities. We could carry out a simulation of this with a single particle. At every time step, we roll a die, and based on our roll and on the transition probabilities, we decide if the particle jumps states or not. If we wait long enough, we know that the fraction of the time the particle spends in state 1 divided by the fraction of the time it spends in state 2 is just the ratio of $p_{21}$ to $p_{12}$. A fact that will become crucial is that the absolute magnitudes of the transition proabilities don't matter - it is just the ratio. If the probabilities are very low, we will have to wait a very long time for the particle's behavior to approach the average, but if we wait long enough, it will.

How does this extend to higher numbers of states? Well, we can convert our two state system into a matrix equation:

$$\begin{pmatrix} \dot{n_1} \\ \dot{n_2} \end{pmatrix} = \begin{pmatrix} -p_{12} & p_{21} \\ p_{12} & -p_{21} \end{pmatrix} \begin{pmatrix} n_1 \\ n_2 \end{pmatrix} \tag{2}$$

In steady state, this must equal zero. We can now build up a many-state transition matrix by adding together all the two-state matrices. As long as the phase space densities are correct (*i.e.* we set $n_1, n_2, n_3...$ correctly), then every two-state matrix will give zero when we multiply by the phase space density, and so *every* possible sum of two-state matrices will give zero. As a more concrete example, let's take a three state system and explicitly write down all the transition matrices:

$$\left[ \alpha_{12} \begin{pmatrix} -p_{12} & p_{21} & 0 \\ p_{12} & -p_{21} & 0 \\ 0 & 0 & 0 \end{pmatrix} + \alpha_{13} \begin{pmatrix} -p_{13} & 0 & p_{31} \\ 0 & 0 & 0 \\ p_{13} & 0 & -p_{31} \end{pmatrix} + \alpha_{23} \begin{pmatrix} 0 & 0 & 0 \\ 0 & -p_{23} & p_{32} \\ 0 & p_{23} & -p_{32} \end{pmatrix} \right] \begin{pmatrix} n_1 \\ n_2 \\ n_3 \end{pmatrix} \tag{3}$$

As long as Equation 1 holds for each pair of states, then this must be equal to zero for *any* values of $\alpha_{ij}$. You can think of the $\alpha$'s as mixing rates between the states. Large values of $\alpha$ mean that particles are likely to bounce back and forth between those two states quickly, while small values of $\alpha$ mean particles will only slowly transition between the two states. Note also that there is nothing special about the number of states. In particular, we can reproduce continuous distributions by taking the limit as the number of states approaches infinity.

With these results in hand, we are now ready to make the key leap. We can simulate a system by starting in a state, picking a new state the system *might* jump to, and then deciding if the system actually makes the jump or not. By convention, if the phase space density is higher at the trial point, we always take the jump. If it is lower, we *sometimes* take the jump, with probability $\frac{n_i}{n_j}$. If we wait long enough, then the the path our system took in our simulation (called a Markov chain) will have its observed phase space density converge to the underlying average (ergodicity). We can then measure any quantity we desire by averaging over the path of the particle.

How we pick trial state $j$ when starting at state $i$ is called the trial distribution. Crucially, the distribution of states in the Markov chain doesn't rely on the trial distribution. A better trial distribution means our Markov chain converges *faster*, but it will still converge even with a poor trial distribution. The only formal constraint is that the probability for state $i$ to attempt to transition to state $j$ has to be the same for state $j$ to transition to state $i$ (and that the chance to transition is non-zero - obviously, we can't wall off sections of parameter space).

How do Markov chains relate to parameter fitting? Let's say we want to calculate the expectation of a model parameter. If we know how to calculate the likelihood of a given set of model parameters, we can write down the expectation of model parameter $m_i$:

$$\langle m_i \rangle = \int m_i \mathcal{L}(m) \mathrm{d}^n m$$

We could in principle just calculate this integral, but these sorts of integrals can rapidly become intractable. Let's say we have 10 parameters, and want to do a numerical integration going from $-3\sigma$ to $3\sigma$ in steps of $0.2\sigma$. In that case, each parameter can have one of 31 possible values, so to brute force this integral we would need to evaluate the likelihood at $31^{10}$ different points in parameer space. That works out to over 800 trillion points. If each likelihood took 1 microsecond,

we would need 25 years just to do this integral! Now let's say we had a Markov chain - we could take the expectation of $m_i$ just by averaging the values of $m_i$ in the chain. If the Markov chain converges at the level we care about in many fewer steps, then we can calculate parameter expectations, or indeed *any* statistical quantity, very quickly by averaging over samples in the chain.

This then is the power of using Markov chains to carry out Monte Carlo integrations over complicated likelihood surfaces (hence the name of this technique: Markov Chain Monte Carlo or MCMC). With a good trial distribution, MCMC will converge faster, but even with a poor trial distribution, it will eventually converge. The number of chain samples required to converge tends to grow only slowly with increasing numbers of parameters, and so MCMC is useful for high dimensional problems. They are particularly when the likelihood surface is non-Gaussian, since methods based on Taylor expansions can give wildly misleading results.

We can now write down the steps to carry out MCMC:
Step -1: Decide on a trial distribution. It must be symmetric, but otherwise can be (nearly) anything.
Step 0: Pick a random point in parameter space and calculate its likelihood. This is your current location.
Step 1: Take a trial step, drawn from the trial distribution.
Step 2: Evaluate the likelihood at the trial location.
Step 3: If the new likelihood is larger, move to the new location with probability 1. If the new likelihood is smaller, move to the new location with probability $\mathcal{L}_{new}/\mathcal{L}_{old}$ (you'll need to generate a random number for this).
Step 4: Save your current location, and go back to step 0.
Keep going until you're converged.

## 3. Convergence

We have repeatedly mentioned that Markov chains must be "long enough", or "converged". What does that mean, and how will we know when we are converged? The qualitative answer is that we are converged when the difference we expect between any quantitty we care about and the result from an infinitely long Markov chain is "small enough". Convergence criteria can vary wildly depending on which wuantities the user cares about and what errors they are willing to tolerate. Because a particle spends only a tiny fraction of its time in regions of very low likelihood, measuring the $5\sigma$ error on a parameter can take drastically longer than measuring its mean.

An important concept in convergence is the idea of an *independent sample*. If I know the state of my system at time $t$, I know that the state at time $t + dt$ must not have changed much if $dt$ is small. For sufficiently large $dt$, knowing the state at time $t$ tells you nothing about the state at $t + dt$. Loosely speaking, there is a minimum value of $dt$ for which samples are independent, and convergence often comes down to knowing how many independent samples we have. Take the case of measuring the mean of a parameter to $0.1\sigma$. The variance of each individual independent sample would be $\sigma^2$, and so we could get to $0.1\sigma$ uncertainty with 100 independent samples. In contrast, a Gaussian-distributed particle would spend about $10^{-6}$ of its time at $5\sigma$ or greater, so brute-force

we would need millions of independent samples to get enough $5\sigma$ entries in the Markov chain to say anything useful about the distribution there.

One way of checking convergence is through the Gelman-Rubin test. It relies on the fact that the scatter in the mean of a distribution is smaller than the scatter of individual realizations of that distribution. For a single Markov chain, the error in the mean of a paramter is roughly the per-sample error divided by the square root of the number of independent samples. This doesn't help us much if we only have one Markov chain, but take the case of having multiple, independent Markov chains. We can get an estimate of the error in the mean by comparing the means of the different chains. The number of independent samples is then something like the mean of the variances within chains divided by the variance of the differences between chains of their means. One usually runs several different chains (often in the range of 5-10) starting from different, random points, and stops when the scatter of the means across chains is smaller than some factor (often $\sim$ 0.01-0.02) times the average of the within-chain scatters. Gelman-Rubin says to periodically check this ratio and stop when it is small enough. It is generally wise to pick starting points for the chains with more scatter than the chains themselves have, to reduce the risk of believing you are converged when you're not.

The Gelman-Rubin test works well in the world where you have many chains. The question naturally arises: "Can we estimate convergence from a single chain?" Happily, the answer remains yes. If we have uncorrelated samples, then if we took the Fourier transform of the parameter values in those samples, we would see white noise (since the Fourier transform of white noise is white noise). If the samples are correlated, we would see that there is more noise on large scales (where the samples bounce around) than on small scales (since nearby samples are closely related). When we Fourier transform the parameters, we can square and smooth, and typically see a bend in the smoothed transform (referred to as a *power spectrum*). If you take the index of the power spectrum where the slope breaks, that indicates roughly how many independent samples you have (in fact it's slightly pessimistic, because there is *some* information in the correlated region, but you won't be too far off taking the break position). Examples of unconverged and converge Markov chains and their corresponding power spectra are shown in Figures 1 and 2.

### 3.1. Quantitative Estimate of Independent Samples

We can use the power spectrum picture to be more quantitative about convergence. If we estimate the mean of a distribution by averaging over a bunch of independent, identically-distributed data points then the uncertainty in the mean is just $frac1\sqrt{n}$ times the uncertainty in a single data point. We can estimate the number of independent samples by squaring the ratio of the single-point uncertainty to the mean uncertainty, equivalent to the ratio of their variances.

To work out the variance based on the power spectrum, let's start by taking a 2-element data vector with entries $(d_1, d_2)$, and associated noise matrix N where $N_{ij} = \langle d_i d_j \rangle$. What is the variance of $\alpha d_1 + \beta d_2$? For simplicity, we'll assume the means are zero, so we just need

$$\left\langle (\alpha d_1 + \beta d_2)^2 \right\rangle = \left\langle \alpha^2 d_1^2 + 2\alpha\beta d_1 d_2 + \beta^2 d_2^2 \right\rangle = \alpha^2 N_{11} + 2\alpha\beta N_{12} + \beta^2 N_{22}$$

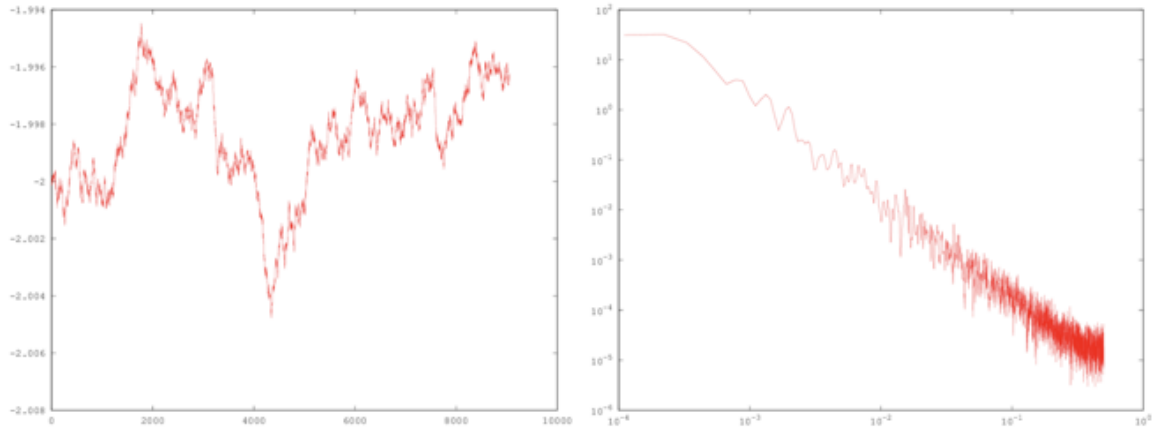

Fig. 1.— Left: An unconverged Markov chain. If someone bet you your life you knew what the long-term average of this was going to end up being, would you take that bet? Right: The power spectrum (smoothed Fourier transform) of the Markov chain in the left panel. Note that power keeps going up as you go to larger scales (move towards the left). There's no indication that the spectrum is levelling off.

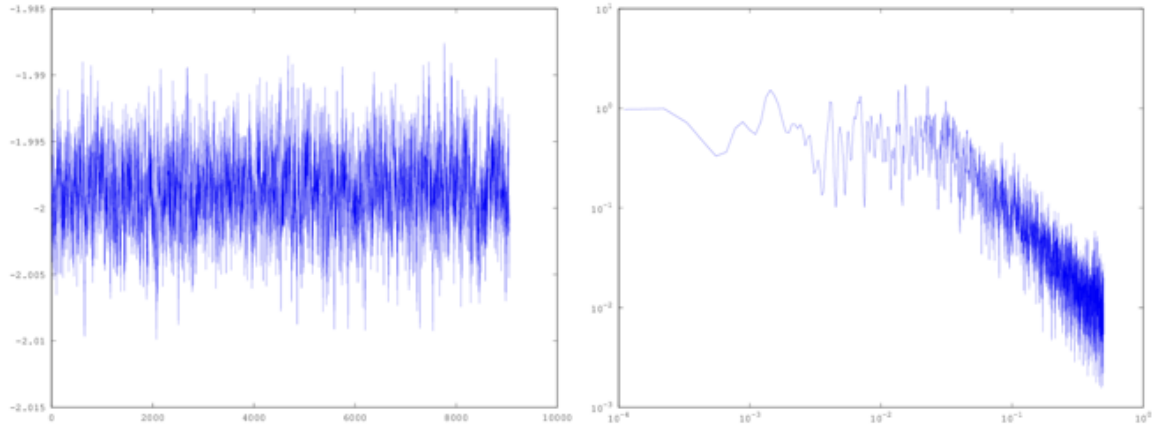

Fig. 2.— Left: An converged Markov chain. If someone bet you your life about the mean of this distribution, would you take it this time? Right: the power spectrum of this Markov chain. The power flattens off in the left half of the spectrum, indicating convergence. The x-axis has been scaled to go to 1, and the knee is at about 0.03, so we get an independent sample roughly every 30 entries in the chain. Since the chain is 9000 samples long (from the x-axis of the left panel), we have about $0.03 \times 9000 \sim 300$ independent samples. We should have good estimates of the $1-\sigma$ errors, reasonable estiamtes of the $2-\sigma$ errors, and nothing to say about $5-\sigma$ errors from this chain.

If we define the vector of weights to be $w \equiv (\alpha, \beta)$ then this is exactly

$$w^T \mathrm{N} w \tag{4}$$

If we assume our chain is stationary (which it had better be if it's converged), then we know it's diagonal in Fourier space, and we can write the noise as $\mathrm{F}^\dagger \tilde{\mathrm{N}} \mathrm{F}$ where F is the Fourier transform

operator, and $\tilde{\mathrm{N}}$ is the power spectrum/diagonal noise. If we want our noises to come out properly normalized we do need to be a little careful, since the usual definition of the FT has a $1/n$ on the inverse but nothing on the forward transform, so we'll probably want to divide by $\sqrt{n}$.

What's the uncertainty of the mean? Well, the mean of our chain samples is just the average across them, or $\frac{1}{n}\sum d_i$. We can express this using the weight vector by setting the entries to $1/n$. Pulling out the $1/n$, we have $w = \frac{1}{n}(1, 1, 1, 1, ..., 1)$. If we plug that into the noise, then we pick up the Fourier transform of $w$. The standard FT of a vector of ones is $n$ if $k = 0$ and zero otherwise. The factor of $n$ cancels the $1/n$ coefficient, and we're left with a delta function. Now, we actually are using $1/\sqrt{n}$ for our normalization factor to keep the noise factorization symmetric, so our final normalized vector is $(\frac{1}{\sqrt{n}}, 0, 0, ...)$. Apply that on both sides to the diagonal power spectrum, and we're left with the variance of the mean is $\tilde{N}_{0,0}/n$, the $k = 0$ element of our diagonal power spectrum divided by the number of data points. Of course, in our definition of the FT we introduced a $1/\sqrt{n}$, so when we calculate the square of the FT, we pick up a $1/n$. We can go from the usual FT to the normalized with this $1/n$, so if we start with the usual DFT, our estimate of the variance on the mean is the $k = 0$ power spectrum from the usual DFT divided by $n^2$. We never actually get to measure the $k = 0$ component directly because we don't know the true mean, but we do know the power spectrum of a converged chain is flat so we can estimate the $k = 0$ value by averaging over the flat part of the power spectrum.

Now that we have the estimate of the mean variance, let's also estimate the single-point variance using the power spectrum. We could, of course, just look at the scatter within the chain, but it's illustrative to go via the power spectrum. We redo the exact same calculation, but since now we want the uncertainty on a single point, our weight vector is all zeros except for a single 1. It doesn't matter which entry we put the one in (since the chain should be stationary), so for simplicity let's put it in the first entry and so the weights become a delta function. The unnormalized DFT of that is just a vector of ones, and when we put that on both sides of $\tilde{\mathcal{N}}$, we're left with the sum of the power spectrum. We pick up the same $1/n$ factor from switching to the normalized DFT operator, and the same additional $1/n$ if we want to start with the standard DFT. We're left with the single-sample variance being the sum of the power spectrum divided by $n^2$.

Now that we have the single-sample variance and the variance on the mean, we can take the ratio to get the effective number of independent samples. The factors of $1/n^2$ cancel, and we're left simply with

$$n_{ind} = \sum F(k)^2 / < F(k=0)^2 >$$

where $F(k)$ is the DFT of the chain. This is exactly the estimate we had before if the power spectrum was flat out to the knee, and then immediately dropped to zero. We now see that the true answer is somewhat larger than our initial estimate, with the exact factor set by the slope of the power spectrum past the knee.

A pretty reasonable assumption about the power spectrum is that on short scales the chain is a random walk. If your steps are small then the likelihood won't change appreciably, so given a current sample, the next one will just be the current one plus a random step. The power spectrum of a random walk is $k^{-2}$, so the overall power spectrum of a chain should be something like $\frac{a}{1+(k/k_{knee})^2}$ where $a$ is the per-sample variance, and $k_{knee}$ is the knee frequency. This is a Lorentzian, and the

sum over the power spectrum, assuming $k$ goes to infinity, will be $\frac{\pi}{2} a k_{knee}$. Since $F(k=0) = a$, then the number of independent samples is something like $\frac{\pi}{2} k_{knee}$. After all this work, we're left with the remarkably simple conclusion that a reasonable guess for the number of independent samples is about 1.5 times the knee frequency, since $\pi/2 = 1.57....$

## 4. Importance Sampling

For large problems, generating well-converged Markov chains can often be a pretty large computational task. Very often, you may want to make small changes to the likelihood (say from including a bit of extra data, or including prior knowledge of parameters). Starting from scratch to include the new information would be painful, so one might naturally ask if there's a way to handle this case without having to re-run a full chain? Happily, the answer is yes! If I have a likelihood function $\mathcal{L}(p)$ and want to adjust the likelihood to be some new function $\mathcal{L}'(p)$, then I know the ratio of the new phase-space density to the old one is just $\mathcal{L}'(p)/\mathcal{L}(p)$. If our likelihood has taken the form of $\chi^2$, this is just $\exp(-\frac{1}{2}\delta\chi^2$. If I have a well-converged chain for
$mathcalL(p)$, then I can just adjust the phase-space density in that chain by $\exp(-\frac{1}{2}\delta\chi^2$, and so I can average over the new distribution simply by taking a weighted average over the old distribution. One needs to be careful when doing this, as if the parameters shift by too much (typically $1-\sigma$ or so), then the old chain will only poorly sample some regions that are important for $\mathcal{L}'(p)$. This technique is called *importance sampling*, and is very handy for quickly getting new parameter estimates/errors when only slightly changing a problem. Note that to do this, you'll need to keep track of the likelihood of each chain sample. You should be doing this anyways.

Notice that when we importance sample, the way it can break down is if the new parameter space has important regions that fall outside the old parameter space. If the new parameter space is a subset of the old space, then we are much less likely to have issues. We still may want longer chains, since we are in effect throwing away samples, but we aren't going to miss anything major. With this in mind, we can use importance sampling to get to those elusive $5-\sigma$ errors. Say we divide $\chi^2$ by 5 and run a Markov chain. Regions that used to be pretty much disallowed are now going to be well sampled by this new chain. Since thermodynamically, dividing $\chi^2$ by a factor is equivalent to increasing the temperature by the same factor, these chains are usually referred to as having high temperatures. After running a high-temperature chain, we can now importance sample it down to the actual temperature we want (for numerical overflow/underflow purposes, you might want to pick a new/old reference $\chi^2$ and importance sample based on scaling the difference between the sample $\chi^2$'s and the reference value. While you'll probably want to run a longer-than-usual chain initially, this technique will get to large-$\sigma$ error bars far more efficiently than trying to brute-force an unheated chain.

Since people often find this confusing, let's walk through the details of how to run and importance-sample a chain for $5-\sigma$ errors. If we're looking for $5-\sigma$, we probably want to run with our errors multiplied by 5, or $\chi^2$ divided by 25. So now we run our chain and get our samples and their associated $\chi^2$ values, but we'll call this $\chi'^2$, wich is the true $\chi^2$ divided by 25. The chain has a density in parameter space given by $\chi'^2$, but we actually want the density given by $\chi^2$. We can importance sample to get there - since a factor of $\chi^2/25$ is already baked into the chain,

we need to account for the other $24\chi^2/25$. That means weighting each point by $\exp(-24\chi^2/25/2)$, but since we already have $\chi'^2 = \chi^2/25$ handy from the chain we ran we can use $w = \exp(-24\chi'^2/2)$ for our weights.

We can now do any operation with the chain samples, as long as we weight each sample by its weight $w$. We could for instance estimate the mean of parameter $j$ with $\sum_i w_i s_{i,j} / \sum_i w_i$ where $w_i$ is the weight of the $i^{th}$ sample and $s_{i,j}$ is the value for parameter $j$ in sample $i$. While this works, if you're after the mean then running a high-temperature chain is rather silly. The purpose of this high-T chain is instead to set limits on $p_j$ that you'd expect to exceed once in a million times. For a vanilla, enormously long chain this is easy to do. You find the value for $p_j$ where one in a million samples is larger, and that's your upper limit. For the high-T chain, though, the weight is the equivalent density of samples, so you want to find the value of $p_j$ where a part in a million of the *weighted* samples is larger than your threshold. You sort the samples/weights based on $p_j$, and find the value where $\sum_{p_j > thresh} w_i = 10^{-6} \sum w_i$.

## 5. MC

A big part of the reason MCMC can be slow is that adjacent samples are typically highly correlated. If we had a way of generating independent samples, we could get convergence much more quickly. There are several ways of doing this (*e.g.* Hamiltonian Monte Carlo), but one simple way that works in many situations is plain-old Monte Carlo. Consider the case where our likelihood is perfectly described by a Gaussian with some covariance matrix C around a set of maximum likelihood parameters $p_0$:

$$\mathcal{L} = \exp\left(-\frac{1}{2}(p-p_0)^T \mathrm{C}^{-1}(p-p_0)\right)$$

If we were to run a chain, we know the density of chain samples would be proportional to this likelihood. Of course, we *also* know how to generate samples with density proportional to this likelihood. We just generate samples with covariance C centered on $p_0$. The difference is that if we generate the samples directly, every single sample is independent, while we know that in a chain, only a small fraction of the samples are independent. For starters, fastest convergence on a chain is when the acceptance fraction is ∼20-25%, so *at best* a quarter of the chain samples are independent. A more typical number is 10% or less. If we're clever, then we might be able to speed up our analysis by an order of magnitude. This can be very useful!

Of course, it's not very useful to sample if our likelihood surface is actually Gaussian. We know how to integrate that analytically, so we're wasting our time by sampling. However, if the surface is "mostly" Gaussian, then we can pretend it's Gaussian and then correct. We know that the sample density should be proportional to the likelihood, so we can define a correction factor between the density of points we have from sampling a Gaussian, and the density of points we should have had from our true likelihood. Given a Gaussian approximation with best-fit parameters and associated covariance matrix, we can sample points randomly from the covariance matrix and we know the associated Gaussian density is $\exp -\frac{1}{2}\delta_p^T \mathrm{C}^{-1}\delta_p$, where $\delta_p$ is the difference between our sample point and the nominal best-fit parameters, and C is our parameter covariance matrix. If we sample from a Gaussian this is the density of sample points. The density we *should* have gotten is the exact

likelihood, which we can write as $\frac{\mathcal{L}_{exact}}{\mathcal{L}_{gauss}}\mathcal{L}_{gauss}$. The upstairs factor of $\mathcal{L}_{gauss}$ is taken care of by our sampling density, so we want to re-weight each point by $\frac{\mathcal{L}_{exact}}{\mathcal{L}_{gauss}}$. Then we average over our samples just as we did for an importance-sampled chain, using these updated weights.

As with MCMC, this MC sampling will always work if we have enough samples. How do we guess if we're converged? Well, if the true likelihood is close to Gaussian, we'll see that the weights are close to one. In that case, every sample is independent, and you can estimate uncertainties based on that. If some of the weights are much less than one, that's telling you that a region the Gaussian approximation thinks is OK is actually ruled out by the true likelihood. We've wasted a bit of effort sampling there, but we're still in pretty good shape. The dangerous case is when we see some weights that are very much larger than one. That means there's a region of parameter space where we expected to be ruled out (because the Gaussian probability is very low), but in fact this region is consistent with the data. Because we sample low-probability regions very coarsely due to their low weights, the high-weight region is poorly sampled but should have been more carefully sampled. If your MC pipeline is going to break, this is the clearest warning flag you have a problem. In that case, you may have to go back and re-run a standard chain.

For the homework case of the Planck likelihood, I find that the weights are well-behaved for a standard temperature MC chain, and that convergence is 5-15 times faster than an MCMC. If you know your likelihood is reasonably well behaved, you can save a lot by not actually running a chain! Of course, if you find out that you should have run a chain, you're usually stuck starting over again from square one. It's so much faster to run a non-Markov chain, though, that in many cases it's worth a shot.

## 6. More Complicated Surfaces

A quite common case is that we have multiple islands of allowed parameter space. We've talked about sampling from a Gaussian for our trial MCMC steps, but we aren't required to do that. For a chain to converge, we require that the probability of trying to move to state 2 when we're at state 1 is the same as trying to move to state 1 from state 2. This is called detailed balance, and is the condition for a Markov chain to be unbiased. Symmetric Gaussians are one sampling distribution that's guaranteed to satisfy detailed balance, but there's absolutely no reason we have to use them.

Consider the case where you know you have two likelihood islands separated by some shift in parameters $\delta_p$. If the ridge in $\chi^2$ is very high between the two islands, it will take a chain a very long time to make the jump if we use a Gaussian sampler based on the curvature near the global minimum. Let's say you managed to estimate $\delta_p$, though - say by running Netwon's method near each peak. I could then say "90% of the time, take a usual step. 10% of the time, take a step $\pm\delta_p$", where the sign on $\delta_p$ is chosen randomly. Since this sampling distribution is symmetric, it will satisfy detailed balance. 90% of the time, we sample as usual, but now there's a substantially non-zero chance of jumping between the two islands. Now our sampler will run happily, sampling both islands while also efficiently sampling within each island.

Incidentally, the original Page Rank google algorithm did exactly this. It was a Markov chain

where it randomly followed links on the web to see which web pages most people pointed to. To avoid getting stuck in closed loops of, say, internal files (or later, of SEO optimizers trying to trick google) google's web crawler Markov chains would occasionally randomly jump to a new web page and see where they went. After running their web crawler chains long enough, they could see which web pages were trusted by others by seeing where the chains spend the most time[1].

[1]Thanks to Veritasium for this interesting tidbit. I recommend their recent video on Markov chains for some historical context as well