# Home-made MCMC

If you do a search in ADS, you will find that the ratio between the number of times "MCMC" and "Bayesian" are mentioned in refereed papers is about 0.5. A quick conclusion (probably wrong) is that half of the papers doing Bayesian analysis use an MCMC. In any case, it's fair to say that MCMC is widely used, so our first exercise will be to write our own implementation of the Metropolis-Hastings algorithm and use it to sample from a simple distribution.

Lets suppose we have a distribution $p(\theta)$ which, for most purposes, will be some posterior distribution $p(\theta|D, I)$. The goal is to obtain samples from this distribution: a set of points $\theta_i$ which have the correct proportions. We start with the simplest case in which $\theta$ corresponts to only one parameter.

The MCMC algorithm will allows us to obtain these samples and the only thing we need is to be able to *evaluate* the probability density function of the distribution $p(\theta)$.

How does the algorithm work? Assume we have an initial sample $\theta_i$. We then a pick a proposed value for $\theta_{i+1}$ from a *proposal distribution* $q(\theta_{i+1}|\theta_i)$ which we choose so that it is easy to sample from. In practice, this proposal distribution can have almost any form. A second step is to decide whether or not to accept the candidate $\theta_{i+1}$. We do so on the basis of the value

$$r = \frac{p(\theta_{i+1})}{p(\theta_i)} \frac{q(\theta_i|\theta_{i+1})}{q(\theta_{i+1}|\theta_i)} \tag{1}$$

which is called the Metropolis ratio. If $r \geq 1$, we accept $\theta_{i+1}$. If $r < 1$, we accept $\theta_{i+1}$ with probability $r$. If the proposed value is rejected, we set $\theta_{i+1} = \theta_i$. Rinse, repeat.
This is much easier to understand in pseudo-code:

1. Choose initial value $\theta_0$. Set $i = 0$
2. Repeat
   (a) Obtain new candidate $\theta_{i+1}$ from $q(\theta_{i+1}|\theta_i)$
   (b) Calculate $r$
   (c) Sample a variable $u$ from a uniform distribution $\mathcal{U}(0, 1)$
   (d) If $u \leq r$ accept $\theta_{i+1}$ otherwise set $\theta_{i+1} = \theta_i$; Increment $i$.

If we run this algorithm for $n$ steps, we end up with samples $\{\theta_i\}_{i=1}^n$ which follow the distribution $p(\theta)$.

**Exercise 1:** Implement the MCMC algorithm in `Python` to obtain samples from a Gaussian distribution $\mathcal{N}(0, 1)$. You can use a Gaussian distribution as the proposal distribution (don't get confused!) Run the algorithm for 1000 iterations and plot the histogram of the samples. Also plot the samples as a function of iteration. Your plots should look like Figure 1.

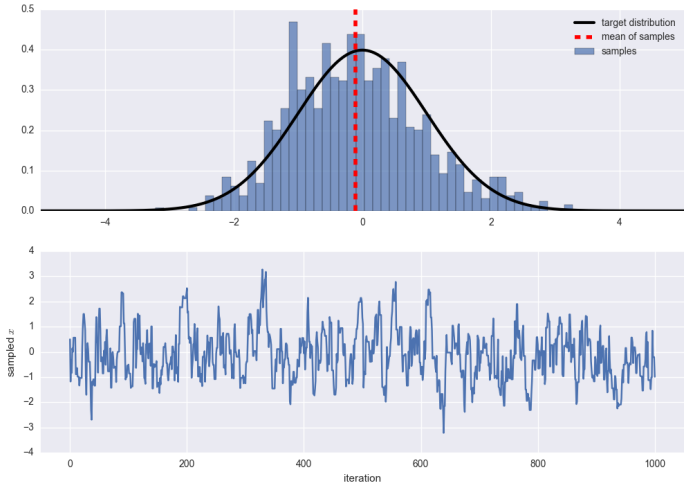


Figure 1.— Partial solution to Exercise 1: *Top:* Histogram of the MCMC samples. *Bottom:* Sampled value vs iteration number.

**Exercise 2:** Transform the MCMC algorithm from Exercise 1 into a (very inefficient) optimization algorithm. What are the differences?

**Exercise 3:** Repeat Exercise 1 but change the variance of the proposal distribution to 0.1 and 10. Calculate the *acceptance rate*, the number of accepted steps divided by the total number of steps, for the different values of the proposal variance. What seems to be the optimal value?

**Exercise 4:** Repeat Exercise 1 but try sampling from a different distribution. Use the distributions implemented in `Scipy`. Does anything change if you try sampling from a Beta distribution?