**[Darren Wilkinson's research blog](http://darrenjw.wordpress.com/)**

Statistics, computing, Bayes, stochastic modelling, systems biology and bioinformatics

« [The last Valencia meeting on Bayesian Statistics and the future of Bayesian computation](http://darrenjw.wordpress.com/2010/06/10/the-last-valencia-meeting-on-bayesian-statistics-and-the-future-of-bayesian-computation/)

[The pseudo-marginal approach to “exact approximate” MCMC algorithms](http://darrenjw.wordpress.com/2010/09/20/the-pseudo-marginal-approach-to-exact-approximate-mcmc-algorithms/) »

**Metropolis-Hastings MCMC algorithms**

**Background**

Over the next few months I’m intending to have a series of posts on recent developments in [MCMC](http://en.wikipedia.org/wiki/Markov_chain_Monte_Carlo) and algorithms of [Metropolis-Hastings](http://en.wikipedia.org/wiki/Metropolis-Hastings_algorithm) type. These posts will assume a basic familiarity with stochastic simulation and [R](http://darrenjw.wordpress.com/2010/01/30/yet-another-introduction-to-r-and-bioconductor/). For reference, I have some old notes on [stochastic simulation and MCMC](http://www.staff.ncl.ac.uk/d.j.wilkinson/teaching/mas451/notes.pdf) from a course I used to teach. There were a set of [computer practicals](http://www.mas.ncl.ac.uk/%7Endjw1/teaching/sim/) associated with the course which focused on understanding the algorithms, mainly using R. Indeed, this post is going to make use of the example from [this old practical](http://www.mas.ncl.ac.uk/%7Endjw1/teaching/sim/metrop/metrop.html). If anything about the rest of this post doesn’t make sense, you might need to review some of this background material. My favourite introductory text book on this subject is Gamerman’s [Markov Chain Monte Carlo](http://www.amazon.com/gp/product/1584885874?ie=UTF8&tag=darrewilkisre-20&linkCode=as2&camp=1789&creative=9325&creativeASIN=1584885874). In this post I’m going to briefly recap the key idea behind the Metropolis-Hastings algorithm, and illustrate how these kinds of algorithms are typically implemented. In order to keep things as simple as possible, I’m going to use R for implementation, but as discussed in a [previous post](http://darrenjw.wordpress.com/2010/04/28/mcmc-programming-in-r-python-java-and-c/), that probably won’t be a good idea for non-trivial applications.

**Metropolis-Hastings**

The motivation is a need to understand a complex probability distribution, *p(x)*. In applications to [Bayesian statistics](http://en.wikipedia.org/wiki/Bayesian_statistics), this distribution is usually a posterior from a Bayesian analysis. A good way to understand an intractable distribution is to simulate realisations from it and study those. However, this often isn’t easy, either. The idea behind MCMC is to simulate a [Markov chain](http://en.wikipedia.org/wiki/Markov_chain) whose equilibrium distribution is *p(x)*. Metropolis-Hastings (M-H) provides a way to correct a fairly arbitrary transition kernel *q(x’|x)* (which typically won’t have *p(x)* as its equilibrium) to give a chain which does have the desired target. In M-H, the transition kernel is used to generate a *proposed* new value, *x’* for the chain, which is then accepted as the new state at random with a particular probability *a(x’|x)=min(1,A)*, where *A = p(x’)q(x|x’)/[p(x)q(x'|x)]*.

If the value is accepted, then the chain moves to the proposed new state, *x’*. If the value is not accepted, the chain still advances to the next step, but the new state is given by the previous state of the chain, *x*.

It is clear that this algorithm is also a Markov chain, and that for *x’ != x*, the transition kernel of this chain is *q(x’|x)a(x’|x)*. A couple of lines of algebra will then confirm that this “corrected” Markov chain satisfies [detailed balance](http://en.wikipedia.org/wiki/Detailed_balance) for the target *p(x)*. Therefore, under some regularity conditions, this chain will have *p(x)* as its equilibrium, and this gives a convenient way of simulating values from this target.

**Special case: the Metropolis method**

It is clear from the form of the acceptance probability that a considerable simplification arises in the case where the proposal kernel is symmetric, having *q(x’|x)=q(x|x’)*. This is useful, as a convenient updating strategy is often to update the current value of *x* by adding an *innovation* sampled from a distribution that is symmetric about zero. This clearly gives a symmetric kernel, which then drops out of the acceptance ratio, to give *A=p(x’)/p(x)*.

**Example**

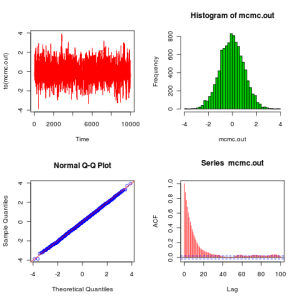
The example we will use for illustration is a Metropolis method for the standard normal distribution using innovations from a *U(-eps,eps)* distribution. Below is a simple R implementation of the algorithm

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | metrop1=function(n=1000,eps=0.5)  {          vec=vector("numeric", n)          x=0          vec[1]=x          for (i in 2:n) {                  innov=runif(1,-eps,eps)                  can=x+innov                  aprob=min(1,dnorm(can)/dnorm(x))                  u=runif(1)                  if (u < aprob)                          x=can                  vec[i]=x          }          vec  } |

First a candidate value (can) is constructed by perturbing the current state of the chain with the uniform innovation. Then the acceptance probability is computed, and the chain is updated appropriately depending on whether the proposed new value is accepted. Note the standard trick of picking an event with probability *p* by checking to see if *u<p*, where *u* is a draw from a *U(0,1)*.

We can execute this code and plot the results with the following peice of code:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | plot.mcmc<-function(mcmc.out)  {      op=par(mfrow=c(2,2))      plot(ts(mcmc.out),col=2)      hist(mcmc.out,30,col=3)      qqnorm(mcmc.out,col=4)      abline(0,1,col=2)      acf(mcmc.out,col=2,lag.max=100)      par(op)  }    metrop.out<-metrop1(10000,1)  plot.mcmc(metrop.out) |

[](http://darrenjw.files.wordpress.com/2010/08/mh-plots.png)

Plots showing convergence of the Markov chain to a N(0,1) equilibrium distribution

From these plots we see that the algorithm seems to be working as expected. Before finishing this post, it is worth explaining how to improve this code to get something which looks a bit more like the kind of code that people would actually write in practice. The next version of the code (below) makes use of the fact that *min(1,A)* is redundant if you are just going to compare to a uniform (since values of the ratio larger than 1 will be accepted with probability 1, as they should), and also that it is unnecessary to recalculate the likelihood of the current value of the chain at every iteration – better to store and re-use the value. That obviously doesn’t make much difference here, but for real problems likelihood computations are often the main computational bottleneck.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | metrop2=function(n=1000,eps=0.5)  {          vec=vector("numeric", n)          x=0          oldlik=dnorm(x)          vec[1]=x          for (i in 2:n) {                  innov=runif(1,-eps,eps)                  can=x+innov                  lik=dnorm(can)                  a=lik/oldlik                  u=runif(1)                  if (u < a) {                          x=can                          oldlik=lik                          }                  vec[i]=x          }          vec  } |

However, this code still has a very serious flaw. It computes likelihoods! For this problem it isn’t a major issue, but in general likelihoods are the product of a very large number of small numbers, and numerical underflow is a constant hazard. For this reason (and others), no-one ever computes likelihoods in code if they can possibly avoid it, but instead log-likelihoods (which are the sum of a large number of reasonably-sized numbers, and therefore numerically very stable). We can use these log-likelihoods to calculate a log-acceptance ratio, which can then be compared to the log of a uniform for the accept-reject decision. We end up with the code below, which now doesn’t look too different to the kind of code one might actually write…

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | metrop3=function(n=1000,eps=0.5)  {          vec=vector("numeric", n)          x=0          oldll=dnorm(x,log=TRUE)          vec[1]=x          for (i in 2:n) {                  can=x+runif(1,-eps,eps)                  loglik=dnorm(can,log=TRUE)                  loga=loglik-oldll                  if (log(runif(1)) < loga) {                          x=can                          oldll=loglik                          }                  vec[i]=x          }          vec  } |

It is easy to verify that all 3 of these implementations behave identically. Incidentally, most statisticians would consider all 3 of the above codes to represent *exactly* the same *algorithm*. They are just slightly different *implementations* of the same algorithm. I mention this because I’ve noticed that computing scientists often have a slightly lower-level view of what constitutes an algorithm, and would sometimes consider the above 3 algorithms to be different.

[About these ads](http://en.wordpress.com/about-these-ads/)

**Share this:**

* [Twitter5](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/?share=twitter&nb=1)
* [Facebook1](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/?share=facebook&nb=1)
* [LinkedIn1](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/?share=linkedin&nb=1)
* [Reddit](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/?share=reddit&nb=1)
* [Google +1](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/?share=google-plus-1&nb=1)

**Like this:**

Tags: [Bayesian](http://darrenjw.wordpress.com/tag/bayesian/), [detailed balance](http://darrenjw.wordpress.com/tag/detailed-balance/), [Hastings](http://darrenjw.wordpress.com/tag/hastings/), [Markov chain](http://darrenjw.wordpress.com/tag/markov-chain/), [MCMC](http://darrenjw.wordpress.com/tag/mcmc/), [Metropolis](http://darrenjw.wordpress.com/tag/metropolis/), [Monte Carlo](http://darrenjw.wordpress.com/tag/monte-carlo/), [programming](http://darrenjw.wordpress.com/tag/programming/), [R](http://darrenjw.wordpress.com/tag/r/), [rstats](http://darrenjw.wordpress.com/tag/rstats/), [statistics](http://darrenjw.wordpress.com/tag/statistics/)

This entry was posted on 15/08/2010 at 20:00 and is filed under [Uncategorized](http://darrenjw.wordpress.com/category/uncategorized/). You can follow any responses to this entry through the [RSS 2.0](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/feed/) feed. You can [leave a response](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/#respond), or [trackback](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/trackback/) from your own site.

**14 Responses to “Metropolis-Hastings MCMC algorithms”**

1. http://0.gravatar.com/avatar/65d5a829d86b2bb5b02e4257edac1b20?s=32&d=identicon&r=G*Jerzy* Says:   
   [16/08/2010 at 13:34](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/#comment-39) | [Reply](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/?replytocom=39#respond)

Thank you so much for this and the earlier MCMC post (comparing R, C, Java and Python)! I’m trying to learn this out of textbooks which explain the theory well… but it is also immensely helpful to hear about numerical-computing pitfalls and see simple examples of “the kind of code one might actually write,” as you put it. I look forward to the rest of this series! Cheers,  
Jerzy

1. http://1.gravatar.com/avatar/a7e48f5bd0f1fdd69bf4317a917e364d?s=32&d=identicon&r=G*Gregor Gorjanc* Says:   
   [17/08/2010 at 17:15](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/#comment-41) | [Reply](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/?replytocom=41#respond)

Nice post! I like the details – majority of books and of course papers do not state such important details. I guess it is assumed that the readership should have know this via some training, but …

1. [*The pseudo-marginal approach to “exact approximate” MCMC algorithms « Darren Wilkinson's research blog*](http://darrenjw.wordpress.com/2010/09/20/the-pseudo-marginal-approach-to-exact-approximate-mcmc-algorithms/) Says:   
   [20/09/2010 at 14:47](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/#comment-73) | [Reply](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/?replytocom=73#respond)

[...] Darren Wilkinson's research blog Statistics, computing, Bayes, stochastic modelling, systems biology and bioinformatics « Metropolis-Hastings MCMC algorithms [...]

1. http://2.gravatar.com/avatar/ba2fde9d1d6e6e540610ac401c6e98d0?s=32&d=identicon&r=G*Heaven's Thunder Hammer* Says:   
   [05/10/2010 at 00:01](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/#comment-92) | [Reply](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/?replytocom=92#respond)

I’m fairly new to MCMC myself. It would be interesting to see how the code gets implemented when one already has some data in hand and a potential model. (i.e. Some kind of mixed model F= p\*guassian1+(1-p)\*guassian2, 50 (or some number “m”) observations, guassian1 and guassian2 “known”, how to find p?)

1. [*Metropolis-Hastings MCMC in R « theoretical ecology*](http://theoreticalecology.wordpress.com/2010/09/17/metropolis-hastings-mcmc-in-r/) Says:   
   [11/10/2010 at 18:22](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/#comment-97) | [Reply](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/?replytocom=97#respond)

[...] which I did to together with Tamara Münkemüller. The sourcecode can be found here. A similar post by Darren Wilkinson is certainly also worth looking [...]

1. [*Getting started with parallel MCMC « Darren Wilkinson's research blog*](http://darrenjw.wordpress.com/2010/12/14/getting-started-with-parallel-mcmc/) Says:   
   [14/12/2010 at 12:35](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/#comment-153) | [Reply](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/?replytocom=153#respond)

[...] algorithm to sample a standard normal using uniform proposals, as discussed in a previous post. Here an independent chain is run on each processor, and the results are written into separate [...]

1. [*MCMC, Monte Carlo likelihood estimation, and the bootstrap particle filter « Darren Wilkinson's research blog*](http://darrenjw.wordpress.com/2011/05/15/mcmc-monte-carlo-likelihood-estimation-and-the-bootstrap-particle-filter/) Says:   
   [15/05/2011 at 21:11](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/#comment-192) | [Reply](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/?replytocom=192#respond)

[...] a previous post I described a generalisation of the Metropolis Hastings MCMC algorithm which uses unbiased Monte Carlo estimates of likelihood in the acceptance ratio, but [...]

1. [*The particle marginal Metropolis-Hastings (PMMH) particle MCMC algorithm « Darren Wilkinson's research blog*](http://darrenjw.wordpress.com/2011/05/17/the-particle-marginal-metropolis-hastings-pmmh-particle-mcmc-algorithm/) Says:   
   [17/05/2011 at 11:46](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/#comment-196) | [Reply](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/?replytocom=196#respond)

[...] we want to target directly, we can use a Metropolis-Hastings scheme with a fairly arbitrary proposal distribution for exploring , where a new is proposed from [...]

1. [*Markov Chain Monte Carlo: What are some good texts on MCMC? - Quora*](http://www.quora.com/Markov-Chain-Monte-Carlo/What-are-some-good-texts-on-MCMC#ans840982) Says:   
   [30/11/2011 at 09:08](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/#comment-362) | [Reply](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/?replytocom=362#respond)

[...] (with code snippets and optimization tips in R, C, C++, Java, Python, Scala) by Darren Wilkinson: <http://darrenjw.wordpress.com/20>…This answer .Please specify the necessary improvements. Edit Link Text Show answer summary [...]

1. [*Catalogue of my first 25 blog posts « Darren Wilkinson's research blog*](http://darrenjw.wordpress.com/2011/12/30/catalogue-of-my-first-25-blog-posts/) Says:   
   [30/12/2011 at 15:44](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/#comment-449) | [Reply](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/?replytocom=449#respond)

[...] Metropolis-Hastings MCMC algorithms: A quick introduction to the Metropolis algorithm, with example code in R, discussing [...]

1. [*Metropolis Hastings MCMC when the proposal and target have differing support « Darren Wilkinson's research blog*](http://darrenjw.wordpress.com/2012/06/04/metropolis-hastings-mcmc-when-the-proposal-and-target-have-differing-support/) Says:   
   [04/06/2012 at 14:46](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/#comment-514) | [Reply](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/?replytocom=514#respond)

[...] often it is desirable to use Metropolis Hastings MCMC for a target distribution which does not have full support (for example, it may correspond to a [...]

1. [*Metropolis Hastings MCMC when the proposal and target have differing support | StatsBlogs.com*](http://www.statsblogs.com/2012/06/04/metropolis-hastings-mcmc-when-the-proposal-and-target-have-differing-support/) Says:   
   [04/06/2012 at 17:03](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/#comment-516) | [Reply](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/?replytocom=516#respond)

[...] often it is desirable to use Metropolis Hastings MCMC for a target distribution which does not have full support (for example, it may correspond to a [...]

1. http://2.gravatar.com/avatar/2a488ad7cbcb21d287ecff448635db36?s=32&d=identicon&r=G*Roisin Mc Allister* Says:   
   [25/06/2012 at 21:19](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/#comment-534) | [Reply](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/?replytocom=534#respond)

what’s the best way to estimate the mean, etc. if you carry out a simple MH algorithm as described? Are the R functions mean() and var() valid in this case? I have found that batchmeans bm() gives a different value for the mean than mean(). I would like to find a value for the mean and a standard error

1. http://0.gravatar.com/blavatar/63af49147209181874ff450965dc41aa?s=32[*Do you \*really\* know how BUGS work? | World Of Piggy*](http://worldofpiggy.com/2013/03/27/do-you-really-know-how-bugs-work/) Says:   
   [27/03/2013 at 20:09](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/#comment-1001) | [Reply](http://darrenjw.wordpress.com/2010/08/15/metropolis-hastings-mcmc-algorithms/?replytocom=1001#respond)

[...] In this case a Metropolis-Hasting step would be required due to the non-standard form of the conditional distribution . A very good article about such an implementation can be found here. [...]

**Leave a Reply**

Top of Form



Bottom of Form

Theme: [Kubrick](http://theme.wordpress.com/themes/kubrick/). [Blog at WordPress.com](http://wordpress.com/?ref=footer).   
[Entries (RSS)](http://darrenjw.wordpress.com/feed/) and [Comments (RSS)](http://darrenjw.wordpress.com/comments/feed/).

[Follow](javascript:void(0))

**Follow “Darren Wilkinson's research blog”**

Top of Form

Get every new post delivered to your Inbox.

Join 78 other followers

Bottom of Form

[Powered by WordPress.com](http://wordpress.com/signup/?ref=lof)