

Introduction to using simulation for inference: the rejection sampler

Paul Hewson

December 11, 2009

1 A rejection sampler

As an introduction to the use of simulation in inference we shall look at the rejection sampler. The basic idea of a rejection sampler is:

- We wish to simulate variables from some density $f(x)$, but don't know how to do this.
- On the other hand, we do know how to simulate variables from $g(x)$. $g(x)$ must clearly have the same support as $f(x)$.
- So we simulate from $g(x)$ and throw away (reject) variables in such a way as to leave us with variables that look as if they came from $f(x)$.

We shall illustrate this with an example based on simulating from the Beta density:

$$f(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1} (1-x)^{b-1}. \quad (1)$$

To be specific, let's say that that we wish to simulate from a $Beta(2.7, 6.3)$ density so $f(x) \sim Beta(2.7, 6.3)$. Most computer software has some routine for generating pseudo-random numbers following a standard uniform distribution, so we shall take $g(x) \sim U(0, 1)$.

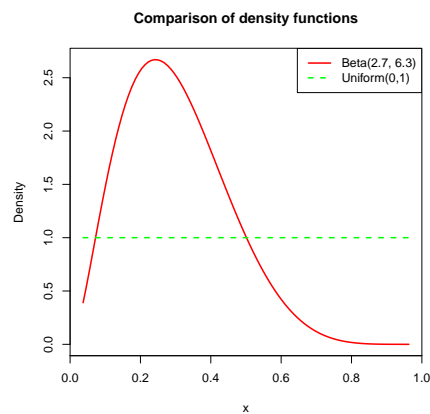


Figure 1: Comparison of Beta(2.7,6.3) density and Uniform(0,1) density

First, a sketch of the densities we are working with is given in figure 1.

- You should be able to see that if we simulate from a Uniform density and reject variables to match the shape of the $Beta(2.7, 6.3)$ density we have a problem.
- Obviously the two densities have the same support, but for part of the range (approximately from $x = 0.8$ to $x = 5.6$) the density of $Beta(2.7, 6.3)$ exceeds the density of the $Uniform(0, 1)$.

What we need to do therefore is to reject variables in the proportion $\frac{f(x)}{Mg(x)}$ where M is a constant chosen to ensure that $Mg(x)$ is greater than $f(x)$ everywhere. If our candidate density is 1 everywhere, and we know that the mode of the $Beta(2.7, 6.3)$ density is $\frac{a-1}{a+b-2} = 0.2429$ we can evaluate the density given in formula (1) and see that it is 2.669. If we round this and take $M = 2.7$ we should be able to run the simulation.

- In order to make these rejection decisions in approximately the correct proportion, we generate another random variable.
- For every rejection sampler algorithm, this will be $U(0, 1)$.
- If this “make a decision” random variable is less than the proportion of densities $\frac{f(x)}{Mg(x)}$ we accept the proposed variate x , if not we reject it.

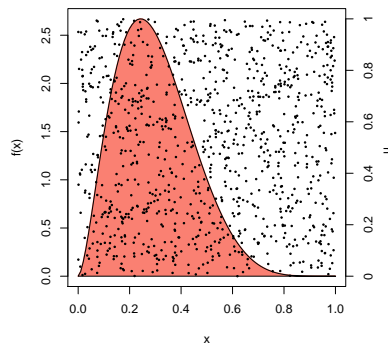


Figure 2: Illustration of rejection sampling with beta distribution. Scaled density and envelope (approx. 1 in 3 points accepted)

The standard illustration for what we are trying to do is given in figure 2. The idea is that the density of the Beta distribution is illustrated on the conventional y axis. We simulate possible values for x , which are illustrated on the x axis. The “decision variable” u is plotted on the second y axis to the right. The idea is that only when u is less than the ratio $\frac{f(x)}{Mg(x)}$ do we accept x as a sample from our target density $f(x)$. In this way, we should accept only the points in the shaded area.

Here is the **algorithm**:

Rejection Sampler

1. Simulate x^* from $g(x)$ (here $x^* \sim U(0,1)$)
2. Simulate u^* from $U[0,1]$
3. Accept x^* if $u^* \leq f(x^*)/Mg(x^*)$ (here $f(x)$ is given in equation (1) with $a = 2.7$ and $b = 6.3$, $M = 2.7$, and $g(x) = 1$)
4. Continue

We have supplied a sample piece of python script that does this (`rejectbeta.py`). It is not the best piece of python script ever written, but it does seem to work correctly. If you find it easier to write your own rejection sampler entirely on your own please do (and please ask if you want any help with anything). You should find that around 30% of the values you generate are accepted, so you need to generate at least 1000 values to leave you with enough variates to confirm they have the correct shape. However, whilst it is a useful starting point, we're not too interested in this sampler. What we want to do is to use this general approach to simulate from the posterior distribution for a Poisson rate parameter λ .

2 Rejection sampler for a simple posterior simulation

We can adapt this basic idea to simulate from a posterior distribution.

For now we shall take the prior distribution to be $f(\lambda) = \text{Gamma}(a,r)$.

Rejection Sampler

1. Simulate λ^* from the prior distribution $p(\lambda)$ - a $\text{Gamma}(a,r)$ density
2. Simulate u^* from $U[0,1]$
3. Accept λ^* if $u^* \leq \frac{L(\lambda^*)p(\lambda)}{L(\hat{\lambda})p(\lambda)}$, this simplifies to $u^* \leq \frac{L(\lambda^*)}{L(\hat{\lambda})}$
4. Continue

We shall calculate the likelihood ratio $\frac{L(\lambda^*)}{L(\hat{\lambda})}$ by performing the calculation $\exp \left\{ \log L(\lambda^*) - \log L(\hat{\lambda}) \right\}$. In order to do this, we shall take the log-likelihood to be:

$$\log L(\lambda) = \sum_{i=1}^n x_i \log(\lambda) - n\lambda$$

(you might notice in this case it is entirely safe to ignore the constants $\sum_{i=1}^n x_i!$ as they cancel in numerator and denominator)

Here are some answers to questions you might have about this algorithm:

- Our proposal density $g(x)$ is the prior density, and we are simulating candidate values of λ , which we denote by λ^* .
- Instead of $f(x)/Mg(x)$, we shall evaluate the ratio of the density of the posterior density given our λ^* to the posterior given $\hat{\lambda}$, the maximum likelihood estimate of λ .
- Note that the prior density in both numerator and denominator cancel.

2.1 Exercise

1. Create a rejection sampler algorithm to simulate from the posterior density of a Poisson rate parameter λ given $x = (3, 2, 4, 3, 2, 3, 2)$.
2. Simulate a posterior for these data using your algorithm and a $Gamma(2, 1)$ prior density for λ . You would anticipate that around 40% of your values are accepted, and should perhaps aim to generate 10000 realisations of λ^* so that you have a posterior simulation containing around 4000 values.
3. Plot your prior density (a histogram of all your λ^* variables) and your posterior density (a histogram of all your *accepted* λ^* variables).

One such simulation produced the plot in figure 3.

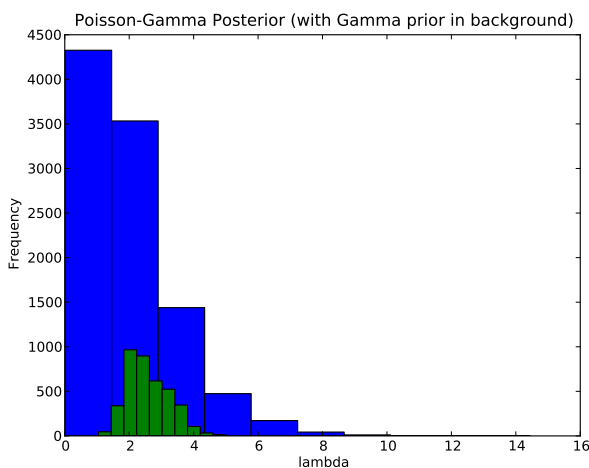


Figure 3: Simulation of prior (blue) and posterior (green) densities for Poisson rate parameter

2.2 Summarising the results

One key attraction of using a rejection sampler is that we simulate a quantity of pseudo-data, and can readily extract simple statistical summaries in order to report our posterior distribution. In this case we can also check that the values of the simulated posterior appear sensible using the analytical results you produced for Homework 2.

You should have a simulation containing perhaps 4000 variables.

1. Use the inbuilt functions in Python to calculate the posterior mean m_p , variance s_p^2 and approximate 95% credible interval for this posterior ($m_p \pm 1.96s_p$). Check these results against those obtained from the analytical formulae.
2. Calculate the 0.025, 0.5 and 0.975 quantiles. The 0.5 quantile is also known as the median. The 0.025 and 0.975 quantiles define the 95% Credible Interval. I couldn't find any inbuilt function to calculate the quantiles in python. I might be wrong/out of date/both so it's worth checking the stat module. If I'm correct, you might find the `quantile.py` script, written by Ernesto P. Adorio and linked from my homepage useful.

- `import quantile` is my naïve way of making this function available

- I stored my posterior simulations as an object called `out`, so `CIlower = quantile.quantile(out, q=0.025)` is my equally naïve way of calculating the $q = 0.025$ quantile (i.e., the lower limit of the 95% posterior credible interval)
3. Check your acceptance rate. If you simulated B values of the prior distribution, and accepted $nsim$ (`nsim = len(out)`) values in your posterior then the acceptance rate is $nsim/B$. If your acceptance rate is too low (less than 10%) you might worry that your prior is too non-informative, if it is too high (greater than 60%) you might worry that it is too informative.

3 Changing the prior

So far we could have done this analytically, and hopefully you have checked your simulation results with analytical results to see that everything is in order. But now we see why simulation is an important technique. We are going to use a different prior, one that is not conjugate and one for which we have no analytical results.

3.1 Include a log-Normal prior

We are going to simulate values from the log-Normal distribution. The Normal distribution is useful in many ways, but $x \sim Normal(\mu, \sigma^2)$ is defined for $-\infty < x < \infty$. We need to transform this and use y such that $0 < y < \infty$. This is very conveniently achieved using $y = \exp(x)$. This can then form a suitable prior for $0 < \lambda < \infty$.

We can directly simulate from a lognormal density using `random.lognormal(muLN, sigma2LN, B)` in python. Although this sounds like a simple transformation of the Normal distribution you should note that the mean of the log-Normal distribution is $\exp(\mu_{LN} + \frac{1}{2}\sigma_{LN}^2)$ and the variance is $\exp(2\mu_{LN}) \exp(\sigma_{LN}^2(\exp(\sigma_{LN}^2) - 1))$. You maybe need to select values such as $\mu_{LN} = \log(2)$ and $\sigma_{LN}^2 = 1$ to start with.

3.2 Exercise

1. Alter and run the rejection sampler to simulate from the posterior of the Poisson rate parameter λ for data $x = (3, 2, 4, 3, 2, 3, 2)$ assuming a log-Normal prior with a range of values for σ_{LN}^2 and μ_{LN}
2. Carefully note the effect this has on your posterior in terms of the mean, median and credible interval. Carefully note the acceptance rate and consider plotting a histogram of the posterior.

4 Footnote

I hope this has introduced you to some ideas about the way in which we can use simulation in order to carry out Bayesian inference. Although you may find this example rather simple, simulation is a very common approach in Bayesian inference, and we have demonstrated several of the key principles in this exercise.