# A Haskell Perspective on a general perspective on the Metropolis–Hastings kernel

Dominic Steinitz

November 14, 2022

## Todo list



## 1 Introduction

Remarkably (to me at least) all[1] MCMC algorithms can be captured by one general algorithm. At the moment you are expected to know how MCMC works to be able to read what follows. I may add a section introducing MCMC later.

Here's **Algorithm 1** from [1]:

$$
\begin{array}{l}
\mathit{algo1} :: \mathit{Show}\ a \Rightarrow \mathit{Show}\ b \Rightarrow (\mathit{MonadDistribution}\ m, \mathit{Fractional}\ t) \Rightarrow \\
\quad (a, c) \rightarrow (a \rightarrow m\ b) \rightarrow ((a, b) \rightarrow (a, b)) \rightarrow (t \rightarrow \mathit{Double}) \rightarrow ((a, b) \rightarrow t) \rightarrow m\ (a, b) \\
\mathit{algo1}\ (\xi_0, \_)\ \mu_{\xi_0}\ \phi\ a\ \rho = \mathbf{do} \\
\quad \mu_{\xi_{-0}} \leftarrow \mu_{\xi_0}\ \xi_0 \\
\quad \mathbf{let}\ \xi = (\xi_0, \mu_{\xi_{-0}}) \\
\quad \mathbf{let}\ \alpha = a\ \$\ (\rho \circ \phi)\ \xi\ /\ \rho\ \xi \\
\quad u \leftarrow \mathit{random} \\
\quad \mathbf{if}\ u < \alpha \\
\quad\quad \mathbf{then}\ \mathit{return}\ \$\ \phi\ \xi \\
\quad\quad \mathbf{else}\ \mathit{return}\ \xi
\end{array}
$$

Something very similar seems to have been discovered in [3] and [5]. Serendipitously, all three papers call this algorithm 1!.

[1] well almost all

## 2 An Example with an Analytical Solution

In Bayesian statistics we have a prior distribution for the unknown mean which we also take to be normal

$$\mu \sim \mathcal{N}\left(\mu_0, \sigma_0^2\right)$$

and then use a sample

$$x \mid \mu \sim \mathcal{N}\left(\mu, \sigma^2\right)$$

to produce a posterior distribution for it

$$\mu \mid x \sim \mathcal{N}\left(\frac{\sigma_0^2}{\sigma^2 + \sigma_0^2}x + \frac{\sigma^2}{\sigma^2 + \sigma_0^2}\mu_0, \left(\frac{1}{\sigma_0^2} + \frac{1}{\sigma^2}\right)^{-1}\right)$$

If we continue to take samples then the posterior distribution becomes

$$\mu \mid x_1, x_2, \cdots, x_n \sim \mathcal{N}\left(\frac{\sigma_0^2}{\frac{\sigma^2}{n} + \sigma_0^2}\bar{x} + \frac{\sigma^2}{\frac{\sigma^2}{n} + \sigma_0^2}\mu_0, \left(\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}\right)^{-1}\right)$$

Note that if we take $\sigma_0$ to be very large (we have little prior information about the value of $\mu$) then

$$\mu \mid x_1, x_2, \cdots, x_n \sim \mathcal{N}\left(\bar{x}, \left(\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}\right)^{-1}\right)$$

and if we take $n$ to be very large then

$$\mu \mid x_1, x_2, \cdots, x_n \sim \mathcal{N}\left(\bar{x}, \frac{\sigma}{\sqrt{n}}\right)$$

which ties up with the classical estimate.

Let's illustrate this with a few numbers.

$$\mu_0, \sigma_0, \sigma, \sigma_P, z :: \mathit{Floating}\ a \Rightarrow a$$
$$\mu_0 = 0.0$$
$$\sigma_0 = 1.0$$
$$\sigma = 1.0$$
$$\sigma_P = 0.2$$
$$z = 4.0$$
$$\hat{\mu} :: \mathit{Double}$$
$$\hat{\mu} = z * \sigma_0 \uparrow 2 \mathbin{/} (\sigma \uparrow 2 + \sigma_0 \uparrow 2) + \mu_0 * \sigma \uparrow 2 \mathbin{/} (\sigma \uparrow 2 + \sigma_0 \uparrow 2)$$
$$\hat{\sigma} :: \mathit{Double}$$
$$\hat{\sigma} = \mathit{sqrt}\ \$\ \mathit{recip}\ (\mathit{recip}\ \sigma_0 \uparrow 2 + \mathit{recip}\ \sigma \uparrow 2)$$

This gives $\hat{\mu} = 2.0$ and $\hat{\sigma} = 0.7071067811865476$ which is what we would expect: we thought the mean was $\mu_0 = 0.0$ but we have an observation $z = 4.0$ and also the variance is now less.

# 3 Using MCMC

For us, we want the posterior

$$\varpi(\mu) = \frac{1}{Z} \exp \frac{(x-\mu)^2}{2\sigma^2} \exp \frac{(\mu-\mu_0)^2}{2\sigma_0^2}$$

where $x, \mu_0, \sigma$ and $\sigma_0$ are all given but $Z$ is unknown.

## 3.1 Random Walk Metropolis

Let's implement a traditional random walk. Here's the proposal distribution:

$$
\begin{array}{l}
Q :: \mathit{Double} \to \mathit{Double} \to \mathit{Double} \\
Q\ w\ w' = \mathit{exp}\ (-(w - w') \uparrow 2\ /\ (2 * \sigma_P \uparrow 2))
\end{array}
$$

And here's the specification for $\rho$:

$$
\begin{array}{l}
\tilde{\rho} :: (a \to \mathit{Double}) \to (a \to b \to \mathit{Double}) \to (a, b) \to \mathit{Double} \\
\tilde{\rho}\ \varphi\ q\ (w, w') = \varphi\ w * q\ w\ w'
\end{array}
$$

Here's the un-normalised posterior:

$$
\begin{array}{l}
\tilde{\varphi} :: \mathit{Floating}\ a \Rightarrow a \to a \\
\tilde{\varphi}\ \mu = \mathit{exp}\ (-(z - \mu) \uparrow 2\ /\ (2 * \sigma \uparrow 2)) * \mathit{exp}\ (-(\mu - \mu_0) \uparrow 2\ /\ (2 * \sigma_0 \uparrow 2))
\end{array}
$$

We can now use one step of the algorithm and then run it for as many times as we wish:

$$
\begin{array}{l}
\mathit{testRwmOneStep} :: \mathit{MonadDistribution}\ m \Rightarrow (\mathit{Double}, \mathit{Double}) \to m\ (\mathit{Double}, \mathit{Double}) \\
\mathit{testRwmOneStep}\ (\xi_0, \_) = \mathit{algo1}\ (\xi_0, \bot)\ \mu_{\xi_0}\ \phi\ a\ \rho \\
\quad \mathbf{where} \\
\qquad \phi = \lambda(x, y) \to (y, x) \\
\qquad a = \mathit{min}\ 1.0 \\
\qquad \rho = \tilde{\rho}\ \tilde{\varphi}\ Q \\
\qquad \mu_{\xi_0} = \lambda\zeta \to \mathit{normal}\ \zeta\ \sigma_P \\
\mathit{testRwm} :: (\mathit{Eq}\ a, \mathit{Num}\ a, \mathit{MonadDistribution}\ m) \Rightarrow \\
\quad a \to m\ [(\mathit{Double}, \mathit{Double})] \\
\mathit{testRwm}\ n = \mathit{unfoldM}\ f\ (n, (1.0, 0.0\ /\ 0.0)) \\
\quad \mathbf{where} \\
\qquad f\ (0, \_) = \mathit{return}\ \mathit{Nothing} \\
\qquad f\ (m, s) = \mathbf{do}\ x \leftarrow \mathit{testRwmOneStep}\ s \\
\qquad\quad \mathit{return}\ \$\ \mathit{Just}\ (s, (m - 1, x))
\end{array}
$$

And we can see the results in Figure 1. A bit skewed but we didn't burn in and the starting value is 1.0.

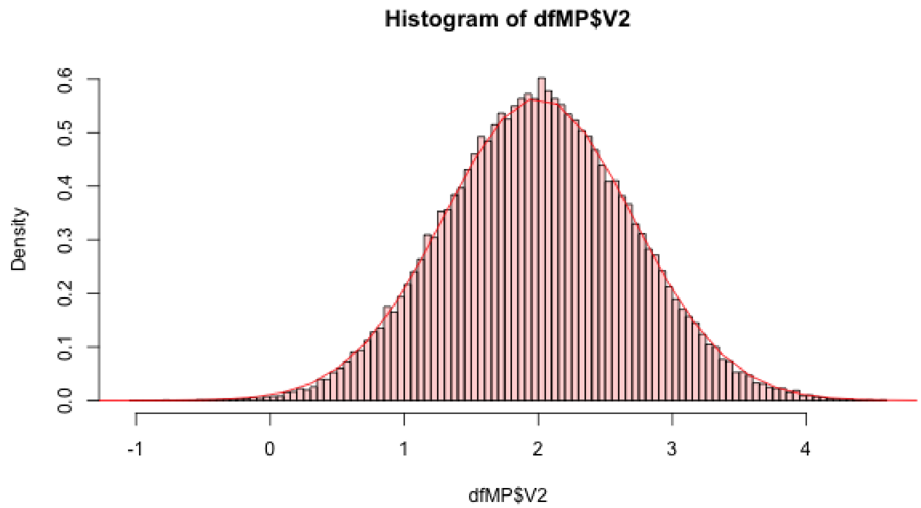


Figure 1: Random Walk Metropolis

### 3.2 Random walk Metropolis ratio

Here's a different algorithm expressed using the generalised approach. The results are in Figure 2.

$$
\begin{aligned}
&\mathit{testMwMrOneStep} :: \mathit{MonadDistribution}\ m \Rightarrow (\mathit{Double}, \mathit{Double}) \to m\ (\mathit{Double}, \mathit{Double}) \\
&\mathit{testMwMrOneStep}\ (\xi_0, \_) = \mathit{algo1}\ (\xi_0, \bot)\ \mu_{\xi_0}\ \phi\ a\ \rho \\
&\quad \mathbf{where} \\
&\qquad \phi = \lambda(x, y) \to (x + y, -y) \\
&\qquad a = \mathit{min}\ 1.0 \\
&\qquad \rho = \tilde{\rho}\ \tilde{\varphi}\ (\backslash\_ \to \backslash\_ \to 1.0) \\
&\qquad \mu_{\xi_0} = \mathit{const}\ (\mathit{quantile}\ (\mathit{normalDistr}\ 0.0\ 1.0) < \$ > \mathit{random}) \\
&\mathit{testMwMr} :: (\mathit{Eq}\ a, \mathit{Num}\ a, \mathit{MonadDistribution}\ m) \Rightarrow \\
&\quad a \to m\ [(\mathit{Double}, \mathit{Double})] \\
&\mathit{testMwMr}\ n = \mathit{unfoldM}\ f\ (n, (1.0, 0.0 / 0.0)) \\
&\quad \mathbf{where} \\
&\qquad f\ (0, \_) = \mathit{return}\ \mathit{Nothing} \\
&\qquad f\ (m, s) = \mathbf{do}\ x \leftarrow \mathit{testMwMrOneStep}\ s \\
&\qquad\quad \mathit{return}\ \$\ \mathit{Just}\ (s, (m - 1, x))
\end{aligned}
$$

### 3.3 What monad-bayes does

Here's our toy problem expressed in monad-bayes:

$$
\begin{aligned}
&\mathit{singleObs} :: (\mathit{MonadDistribution}\ m, \mathit{MonadFactor}\ m) \Rightarrow m\ \mathit{Double} \\
&\mathit{singleObs} = \mathbf{do}
\end{aligned}
$$

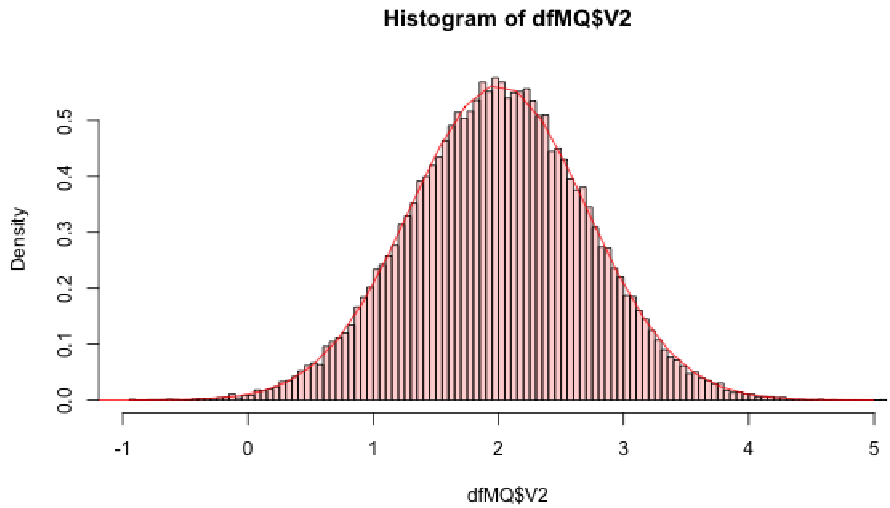


Figure 2: Random walk Metropolis ratio

$$\begin{aligned}
&\mu \leftarrow \mathit{normal}\ \mu_0\ \sigma_0\\
&\mathit{factor}\ \$\ \mathit{normalPdf}\ \mu\ \sigma\ z\\
&\mathit{return}\ \mu
\end{aligned}$$

Here's what I think monad-bayes does with this using the General Perspective. The results are in Figure 3.

$$\begin{aligned}
&\mathit{testMbOneStep} :: \mathit{MonadDistribution}\ m \Rightarrow (\mathit{Double}, \mathit{Double}) \rightarrow m\ (\mathit{Double}, \mathit{Double})\\
&\mathit{testMbOneStep}\ (\xi_0, _) = \mathit{algo1}\ (\xi_0, \bot)\ \mu_{\xi_0}\ \phi\ a\ \rho\\
&\quad \textbf{where}\\
&\qquad \phi = \lambda(x, y) \rightarrow (y, x)\\
&\qquad a = \mathit{min}\ 1.0\\
&\qquad \rho = \tilde{\rho}\ (\lambda\mu \rightarrow \mathit{exp}\ (-(z - \mu) \uparrow 2\ /\ (2 * \sigma \uparrow 2)))\ (\backslash_ \rightarrow \backslash_ \rightarrow 1.0)\\
&\qquad \mu_{\xi_0} = \mathit{const}\ (\mathit{quantile}\ (\mathit{normalDistr}\ \mu_0\ \sigma_0) < \$ > \mathit{random})\\
&\mathit{testMb} :: (\mathit{Eq}\ a, \mathit{Num}\ a, \mathit{MonadDistribution}\ m) \Rightarrow\\
&\quad a \rightarrow m\ [(\mathit{Double}, \mathit{Double})]\\
&\mathit{testMb}\ n = \mathit{unfoldM}\ f\ (n, (1.0, 0.0\ /\ 0.0))\\
&\quad \textbf{where}\\
&\qquad f\ (0, _) = \mathit{return}\ \mathit{Nothing}\\
&\qquad f\ (m, s) = \textbf{do}\ x \leftarrow \mathit{testMbOneStep}\ s\\
&\qquad\quad \mathit{return}\ \$\ \mathit{Just}\ (s, (m - 1, x))
\end{aligned}$$

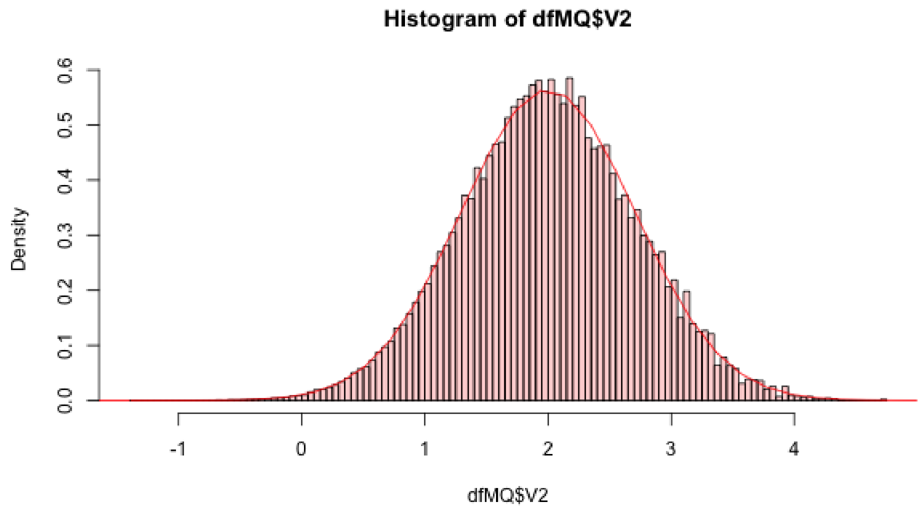


Figure 3: monad-bayes

# 4 Some Mathematical Notes

Suppose we don't know the classical MCMC algorithm. We can derive it from [1]:

$$r\left(z, z^{\prime}\right)=\frac{\varpi\left(z^{\prime}\right) q\left(z^{\prime}, z\right)}{\varpi(z) q\left(z, z^{\prime}\right)}$$

But where does this come from? We define $\mu$:

$$\mu(\mathrm{d} \xi) \triangleq \pi\left(\mathrm{d} \xi_0\right) \mu_{\xi_0}\left(\mathrm{~d} \xi_{-0}\right)$$

$$\mu(\mathrm{d}(z, z^{\prime})) \triangleq \varpi\left(z\right) \mathrm{d} z \; q_z\left(z^{\prime}\right) \mathrm{d} z^{\prime}$$

Let $\mu$ be a finite measure on $(E, \mathscr{E}), \phi: E \rightarrow E$ an involution, let $\lambda \gg \mu$ be a $\sigma$-finite measure satisfying $\lambda \equiv \lambda^\phi$ and let $\rho=\mathrm{d} \mu / \mathrm{d} \lambda$. Then we can take $S=S\left(\mu, \mu^\phi\right)$ to be $S=\{\xi: \rho(\xi) \wedge \rho \circ \phi(\xi)>0\}$ and

$$r(\xi)= \begin{cases}\frac{\rho \circ \phi}{\rho}(\xi) \frac{\mathrm{d} \lambda^\phi}{\mathrm{d} \lambda}(\xi) & \xi \in S, \\ 0 & \text { otherwise }\end{cases}$$

So

$$\rho(z, z^{\prime}) \triangleq \varpi\left(z\right) q_z\left(z^{\prime}\right)$$

and with $\phi(z, z') = (z', z)$ we regain the familiar

$$r(z, z') = \frac{\varpi(z')\, q(z', z)}{\varpi(z) q(z, z')}$$

# 5 Student's T

Let's try running it on Student's T with 5 degrees of freedeom using what I hope the textbook presentation of Metropolis—Hastings. The probability density function (aka the Radon-Nikodym derivative wrt Lebesgue measure) is

$$f(t) = \frac{8}{3\pi\sqrt{5}\left(1 + \frac{t^2}{5}\right)^3}$$

It's traditional to have $q_z(\cdot) \sim \mathcal{N}(z, \sigma_p^2)$ for some given $\sigma_p$.

Here's the density function for Student's T with 5 degrees of freedom. We've defined it in terms of an un-normalised density so that we can pretend we don't know the normalisation constant but still sample from the distribution via MCMC.

```
student5U :: Floating a => a -> a
student5U t = 1 / (1 + t ↑ 2 / 5) ↑ 3
student5 :: Floating a => a -> a
student5 t = student5U t * 8 / (3 * pi * sqrt 5)
```

Again, we can now use one step of the algorithm and then run it for as many times as we wish. We instantiate the algorithm to be a Random Walk Metropolis.

```
testStudentRwmOneStep :: MonadDistribution m =>
  (Double, Double) -> m (Double, Double)
testStudentRwmOneStep (ξ0, _) = algo1 (ξ0, ⊥) (λζ -> normal ζ σP)
  (λ(x, y) -> (y, x)) (min 1.0) (ρ̃ student5U Q)
testStudentRwm :: (Eq a, Num a, MonadDistribution m) =>
  a -> m [(Double, Double)]
testStudentRwm n = unfoldM f (n, (0.0, 0.0 / 0.0))
  where
    f (0, _) = return Nothing
    f (m, s) = do x <- testStudentRwmOneStep s
      return $ Just (s, (m - 1, x))
```

We can also instantiate it with what I think monad-bayes does.

```
testStudentMbOneStep :: MonadDistribution m => (Double, b) -> m (Double, Double)
testStudentMbOneStep (ξ0, _) = algo1 (ξ0, ⊥) (const ((quantile (studentT 5)) <$> random))
```

$$
\begin{array}{l}
\quad (\lambda(x, y) \rightarrow (x + y, -y))\ (min\ 1.0)\ (student5U \circ fst) \\
testStudentMb :: (Eq\ a, Num\ a, MonadDistribution\ m) \Rightarrow \\
\quad a \rightarrow m\ [(Double, Double)] \\
testStudentMb\ n = unfoldM\ f\ (n, (0.0, 0.0\ /\ 0.0)) \\
\quad \mathbf{where} \\
\qquad f\ (0, \_) = return\ Nothing \\
\qquad f\ (m, s) = \mathbf{do}\ x \leftarrow testStudentMbOneStep\ s \\
\qquad\quad return\ \$\ Just\ (s, (m - 1, x))
\end{array}
$$

The results are shown in Figure 4 and Figure 5.

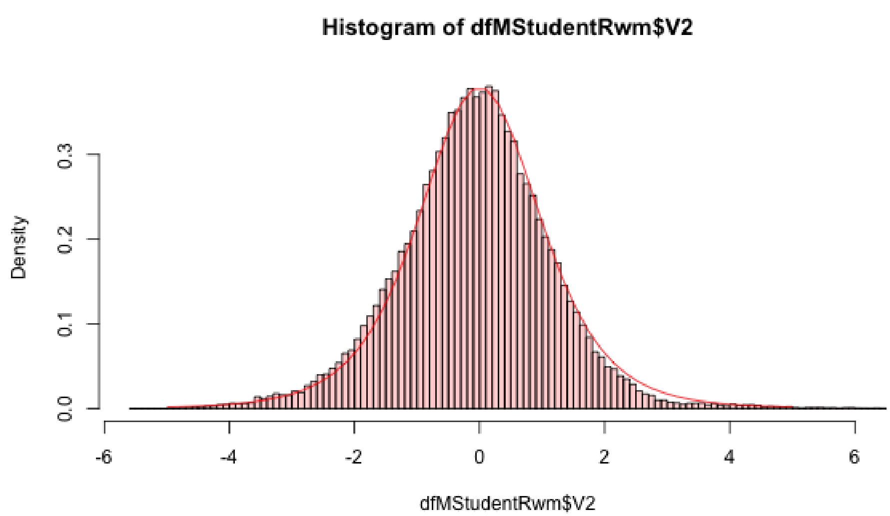


Figure 4: Random Walk Metropolis Student's T 5

# 6 Hamiltonian Monte Carlo

We'd like to put HMC into the same general framework but at the moment, I am having trouble squaring Example 14 in [1] with the algorithm given in [4] (and I haven't even looked in [2]). Here's as far as I got with Student's t-distribution of degree 5. There's something going on with exponentiating the Hamiltonian which I don't understand yet either.

I think I have squared some of this but need to write this down

Here's Student's T again:

$$
f(t) = \frac{8}{3\pi\sqrt{5}\left(1 + \frac{t^2}{5}\right)^3}
$$

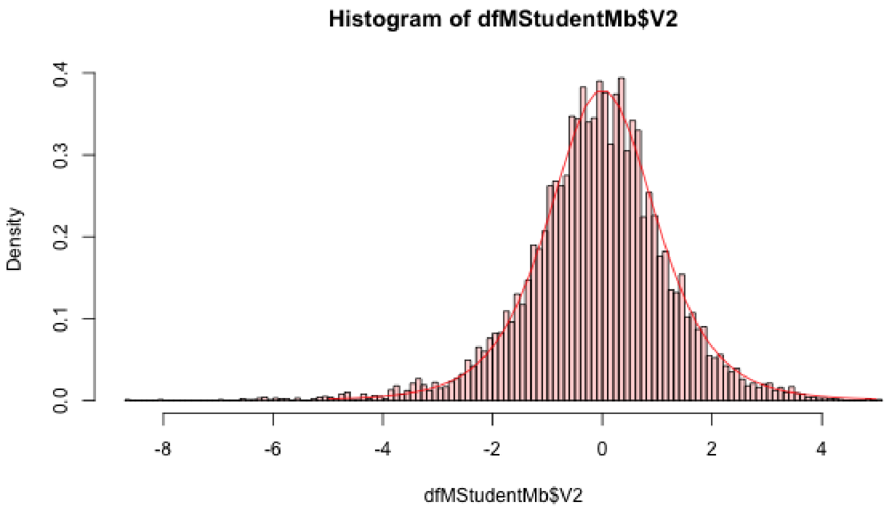


Figure 5: Monad Bayes Student's T 5

Unnormalised:

$$g(t) = \frac{1}{\left(1 + \frac{t^2}{5}\right)^3}$$

And as the potential energy part of the Hamiltonian:

$$U(t) = -\log g(t) = 3\log\left(1 + \frac{r^2}{5}\right)$$

Here's a version of the leapfrog algorithm:

This needs a bit more explanation

```
leapfrog :: Fractional a => a -> Int -> (a -> a) -> (a, a) -> (a, a)
leapfrog epsilon l gradU (qPrev, p) = (q1, p3)
  where
  p' = p - epsilon * gradU qPrev / 2
  f 0 (qOld, pOld) = r
    where
      qNew = qOld + epsilon * pOld
      pNew = pOld
      r = (qNew, pNew)
  f _ (qOld, pOld) = r
    where
```

$$
\begin{aligned}
&\quad\quad \mathit{qNew} = \mathit{qOld} + \mathit{epsilon} * \mathit{pOld} \\
&\quad\quad \mathit{pNew} = \mathit{pOld} - \mathit{epsilon} * \mathit{gradU}\ \mathit{qNew} \\
&\quad\quad r = (\mathit{qNew}, \mathit{pNew}) \\
&\quad (\mathit{q1}, \mathit{p1}) = \mathit{foldr}\ f\ (\mathit{qPrev}, p')\ ([0 \mathinner{..} l - 1]) \\
&\quad \mathit{p2} = \mathit{p1} - \mathit{epsilon} * \mathit{gradU}\ \mathit{q1}\ /\ 2
\end{aligned}
$$

-- Is this necessary?

$$\mathit{p3} = \mathit{negate}\ \mathit{p2}$$

This is the Hamiltonian:

This needs a bit more explanation

$$
\begin{aligned}
&\mathit{rhoHmc} :: \mathit{Floating}\ a \Rightarrow (b \to a) \to (b, a) \to a \\
&\mathit{rhoHmc}\ u\ (q, p) = pU * pK \\
&\quad \textbf{where} \\
&\quad\quad pU = \mathit{recip}\ \$\ u\ q \\
&\quad\quad pK = \mathit{exp}\ \$\ p \uparrow 2\ /\ 2
\end{aligned}
$$

We need the derivative of the potential energy for the leapfrog method. We could use automatic differentiation of course.

$$
\begin{aligned}
&\mathit{gradU} :: \mathit{Fractional}\ a \Rightarrow a \to a \\
&\mathit{gradU}\ r = 3 * (2 * r\ /\ 5)\ /\ (1 + (r \uparrow 2)\ /\ 5) \\
&\mathit{bigU} :: \mathit{Floating}\ a \Rightarrow a \to a \\
&\mathit{bigU} = \mathit{negate} \circ \log \circ \mathit{student5U} \\
&\mathit{gradUAD} :: \mathit{Floating}\ a \Rightarrow a \to a \\
&\mathit{gradUAD}\ w = \textbf{case}\ \mathit{grad}\ (\lambda[x] \to \mathit{bigU}\ x)\ \$\ [w]\ \textbf{of} \\
&\quad [y] \to y \\
&\quad \_ \to \mathit{error}\ \texttt{"Whatever"}
\end{aligned}
$$

And now we can run the sampler. The results are in 6.

$$
\begin{aligned}
&\mathit{eta} :: \mathit{Fractional}\ a \Rightarrow a \\
&\mathit{eta} = 0.3 \\
&\mathit{bigL} :: \mathit{Int} \\
&\mathit{bigL} = 10 \\
&\mathit{testHmcOneStep} :: \mathit{MonadDistribution}\ m \Rightarrow (\mathit{Double}, \mathit{Double}) \to m\ (\mathit{Double}, \mathit{Double}) \\
&\mathit{testHmcOneStep}\ (\xi_0, \_) = \mathit{algo1}\ (\xi_0, \bot)\ \mu_{\xi_0}\ \phi\ a\ \rho \\
&\quad \textbf{where} \\
&\quad\quad \phi = \mathit{leapfrog}\ \mathit{eta}\ \mathit{bigL}\ \mathit{gradU} \\
&\quad\quad a = \mathit{min}\ 1.0 \\
&\quad\quad \rho = \mathit{rhoHmc}\ \mathit{student5U} \\
&\quad\quad \mu_{\xi_0} = \mathit{const}\ \$\ \mathit{normal}\ 0.0\ 1.0 \\
&\mathit{testHmc} :: (\mathit{Eq}\ a, \mathit{MonadDistribution}\ m, \mathit{Num}\ a) \Rightarrow \\
&\quad a \to m\ [(\mathit{Double}, \mathit{Double})] \\
&\mathit{testHmc}\ n = \mathit{unfoldM}\ f\ (n, (0.0, 0.0)) \\
&\quad \textbf{where}
\end{aligned}
$$

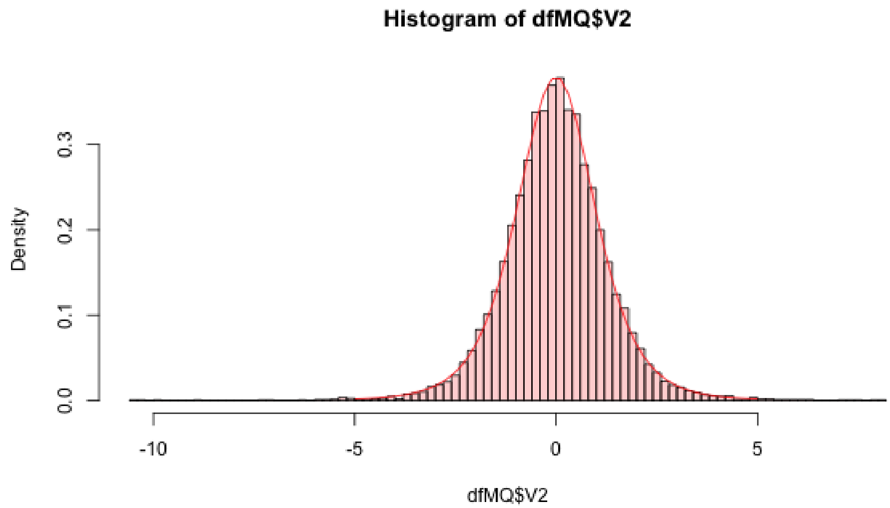


Figure 6: Hamiltonian Monte Carlo Student's T 5

$$
\begin{aligned}
&f\ (0, \_) = \mathit{return}\ \mathit{Nothing} \\
&f\ (m, s) = \mathbf{do}\ a \leftarrow \mathit{testHmcOneStep}\ s \\
&\quad \mathit{return}\ \$\ \mathit{Just}\ (s, (m - 1, a))
\end{aligned}
$$

It should make no difference what we return in the momentum position; yet it does? Maybe not but at least investigate it.

# 7 Gen

Gen is a probabilistic programming language. I've taken the example from [3] and converted it to use monad-bayes.

It's not clear this is a useful section but who knows?

$$
\begin{aligned}
&\mathit{genEg} :: \mathit{MonadDistribution}\ m \Rightarrow \mathit{Int} \rightarrow m\ [\mathit{Double}] \\
&\mathit{genEg}\ n = \mathbf{do} \\
&\quad k \leftarrow (+1) <\$> \mathit{poisson}\ 1.0 \\
&\quad \mathit{means} \leftarrow \mathit{replicate}\ k <\$> \mathit{normal}\ 0.0\ 10.0 \\
&\quad \mathit{gammas} \leftarrow \mathit{replicate}\ k <\$> \mathit{gamma}\ 1.0\ 10.0 \\
&\quad \mathbf{let}\ \mathit{invGammas} = \mathit{map}\ \mathit{recip}\ \mathit{gammas} \\
&\quad \mathit{weights} \leftarrow \mathit{dirichlet}\ (V.\mathit{replicate}\ k\ 2.0) \\
&\quad \mathit{replicate}\ n <\$> (\mathit{categorical}\ \mathit{weights} \ggg\!= \lambda i \rightarrow \mathit{normal}\ (\mathit{means}\ !!\ i)\ (\mathit{invGammas}\ !!\ i))
\end{aligned}
$$

# 8 Bibliography

# References


[1] Christophe Andrieu, Anthony Lee, and Sam Livingstone. A general perspective on the metropolis-hastings kernel. *arXiv*, December 2020. 1, 6, 8

[2] M. J. Betancourt, Simon Byrne, Samuel Livingstone, and Mark Girolami. The geometric foundations of hamiltonian monte carlo, 2014. 8

[3] Marco Cusumano-Towner, Alexander K Lew, and Vikash K Mansinghka. Automating involutive mcmc using probabilistic and differentiable programming. *arXiv preprint arXiv:2007.09871*, 2020. 1, 11

[4] Radford M. Neal. Mcmc using hamiltonian dynamics. *arXiv: Computation*, pages 139–188, 2011. 8

[5] Kirill Neklyudov, Max Welling, Evgenii Egorov, and Dmitry Vetrov. Involutive mcmc: a unifying framework. In *International Conference on Machine Learning*, pages 7273–7282. PMLR, 2020. 1