An Introduction to Markov Chain Monte Carlo

Galin L. Jones

School of Statistics
University of Minnesota

August 7, 2012

Motivating Example

Suppose we want to calculate an integral

$$\int_0^\infty \frac{dx}{(x+1)^{2.3}[\log(x+3)]^2}$$

which we can approximate numerically in R:

```
> integrand<-function(w){
+  1/(((w+1)^(2.3))*(log(w+3))^2)}
> integrate(integrand, lower=0, upper=Inf)
0.419317 with absolute error < 7e-05
```

Numerical integration works well in low dimensions but we want a way of approximating integrals in much more complicated situations.

Motivating Example

Note

$$\int_0^\infty \frac{dx}{(x+1)^{2.3}[\log(x+3)]^2} = \int_0^\infty \frac{f(x)}{f(x)} \frac{dx}{(x+1)^{2.3}[\log(x+3)]^2}$$

Let $X \sim \text{Gamma}(3/2, 1)$ and f be its density function so that the integral we want to calculate is

$$\int_0^\infty \frac{\sqrt{\pi}\, e^x}{2\sqrt{x}(x+1)^{2.3}[\log(x+3)]^2} \frac{2\sqrt{x}}{e^x\sqrt{\pi}} dx \ .$$

This doesn't look easier but it is the key to using *Monte Carlo* since we can now write the integral as an expectation

$$E\left[\frac{\sqrt{\pi}\, e^X}{2\sqrt{X}(X+1)^{2.3}[\log(X+3)]^2}\right]$$

Monte Carlo

The only generally applicable method for approximating integrals is *Monte Carlo Integration*.

Suppose $X_1, \ldots, X_n$ are iid Gamma$(3/2, 1)$. Then we can approximate our integral with

$$E\left[\frac{\sqrt{\pi}\, e^X}{2\sqrt{X}(X+1)^{2.3}[\log(X+3)]^2}\right] \approx \frac{1}{n}\sum_{i=1}^{n} \frac{\sqrt{\pi}\, e^{x_i}}{2\sqrt{x_i}(x_i+1)^{2.3}[\log(x_i+3)]^2}$$

as long as n is "large."

Monte Carlo in R

```
> set.seed(528)
> nsim <- 1e4
> x <- rgamma(nsim, 3/2, 1)
> g.hat <- sqrt(pi)*exp(x) / (2 * sqrt(x) *
   (x+1)^2.3 * (log(x+3))^2)
> mu.hat <- mean(g.hat)
> mu.hat
[1] 0.4183406
```

Recall that numerical integration gave the result as `0.419317`

Monte Carlo in R

An important point about Monte Carlo methods is that different runs will give different estimates.

```
> x <- rgamma(nsim, 3/2, 1)
> g.hat <- sqrt(pi)*exp(x) / (2 * sqrt(x) *
   (x+1)^2.3 * (log(x+3))^2)
> mu.hat <- mean(g.hat)
> mu.hat
[1] 0.4234863
```

If the simulation size is sufficiently large the estimates shouldn't differ by much.

Monte Carlo in R

Lets simplify the problem a little. Suppose we want to estimate $E(X)$ where $X \sim \text{Gamma}(3/2, 1)$. That is, we want to calculate

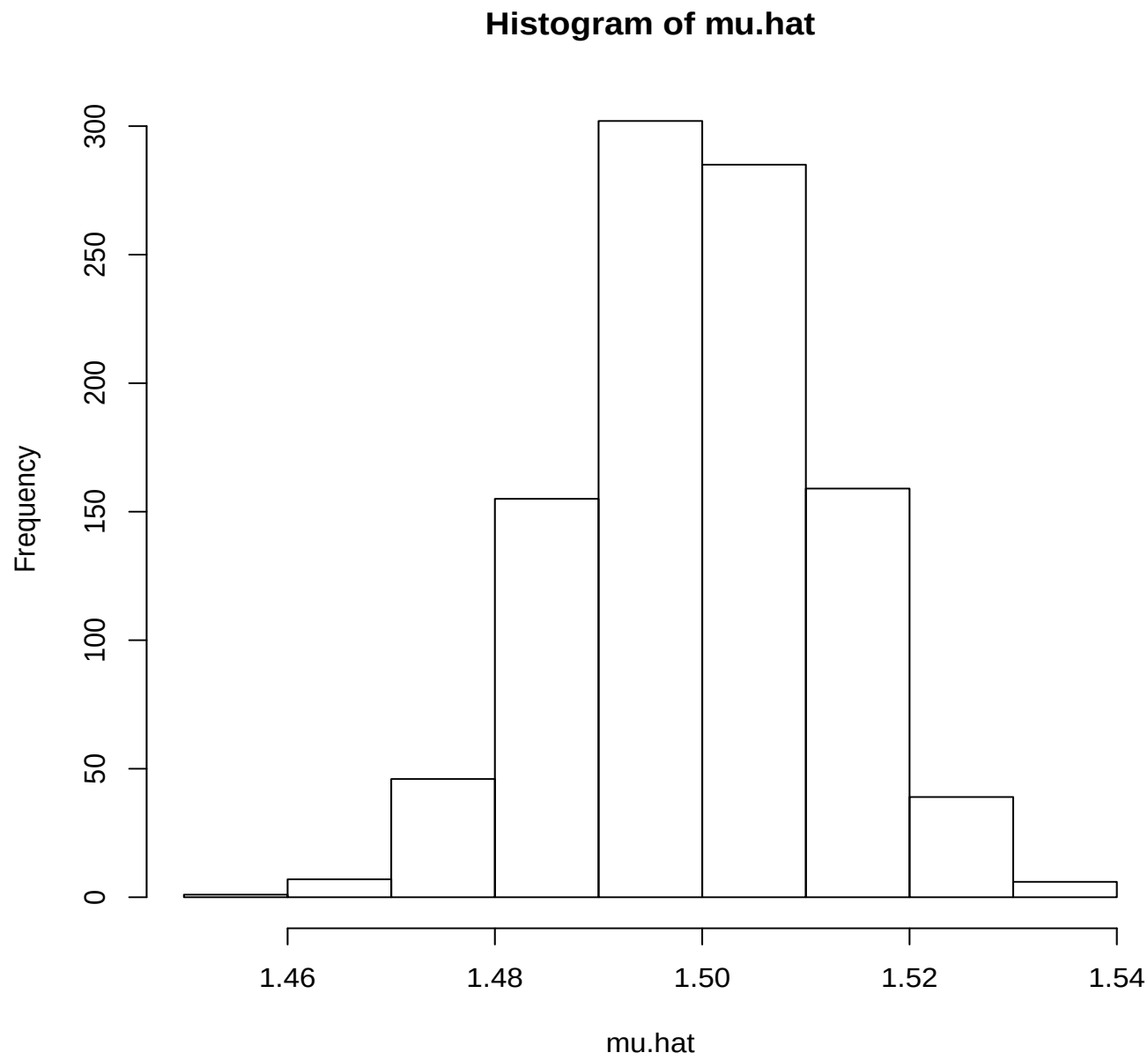
$$\int_0^\infty x \frac{2\sqrt{x}}{e^x \sqrt{\pi}} dx = \frac{3}{2} .$$

```
> nsim<-1e4
> mean(rgamma(nsim, 3/2,1))
[1] 1.474619
> mean(rgamma(nsim, 3/2,1))
[1] 1.491769
> mean(rgamma(nsim, 3/2,1))
[1] 1.501089
```

Monte Carlo in R

```
set.seed(528)
nsim<-1e4
nrep<-1e3
iter<-seq(length=nrep)
mu.hat<-rep(NA, nrep)
for(i in iter){
    x<-rgamma(nsim, 3/2, 1)
    mu.hat[i]<- mean(x)
    }
hist(mu.hat)
```

Monte Carlo in R



Monte Carlo in R

For a single run of length `nsim`, a 95% confidence interval is given by $(1.468, 1.517)$ since

```
> x<-rgamma(nsim, 3/2, 1)
> mu.hat<-mean(x)
> mu.hat
[1] 1.492684
> mu.hat - 2 * sd(x) / sqrt(nsim)
[1] 1.468285
> mu.hat + 2 * sd(x) / sqrt(nsim)
[1] 1.517084
```

Markov Chain Monte Carlo

GOFMC is great when it works but is still generally limited to low dimensional problems.

Markov chain Monte Carlo (MCMC) is the only completely general method for estimating integrals.

Even so, we will work only with our simple one dimensional example.

MCMC in R

The Metropolis-Hastings algorithm is implemented in the `mcmc` R package.

```
> library(mcmc)
> lu<-function(w){dgamma(w, 3/2, 1, log=TRUE)}
> out<-metrop(lu, init=1, 1e4)
> x<-out$batch
> mean(x)
[1] 1.501914
```

Fundamental MCMC Algorithm

Metropolis-Hastings

Recall that f is a Gamma$(3/2, 1)$ density.

Suppose the current state is $X_k = x$. Then

1. $X^* \sim \mathsf{N}(x, 1)$
2. $$\alpha(x, x^*) = 1 \ \wedge \ \frac{f(x^*)}{f(x)} = 1 \ \wedge \ \left(\frac{x^*}{x}\right)^{1/2} e^{x - x^*}$$
3. $$X_{k+1} = \begin{cases} x^* & \text{w. p. } \alpha(x, x^*) \\ x & \text{w. p. } 1 - \alpha(x, x^*) \end{cases}$$

MCMC in Practice

The result of an MCMC simulation is a realization of a Markov chain

$$X_1, X_2, X_3, \ldots$$

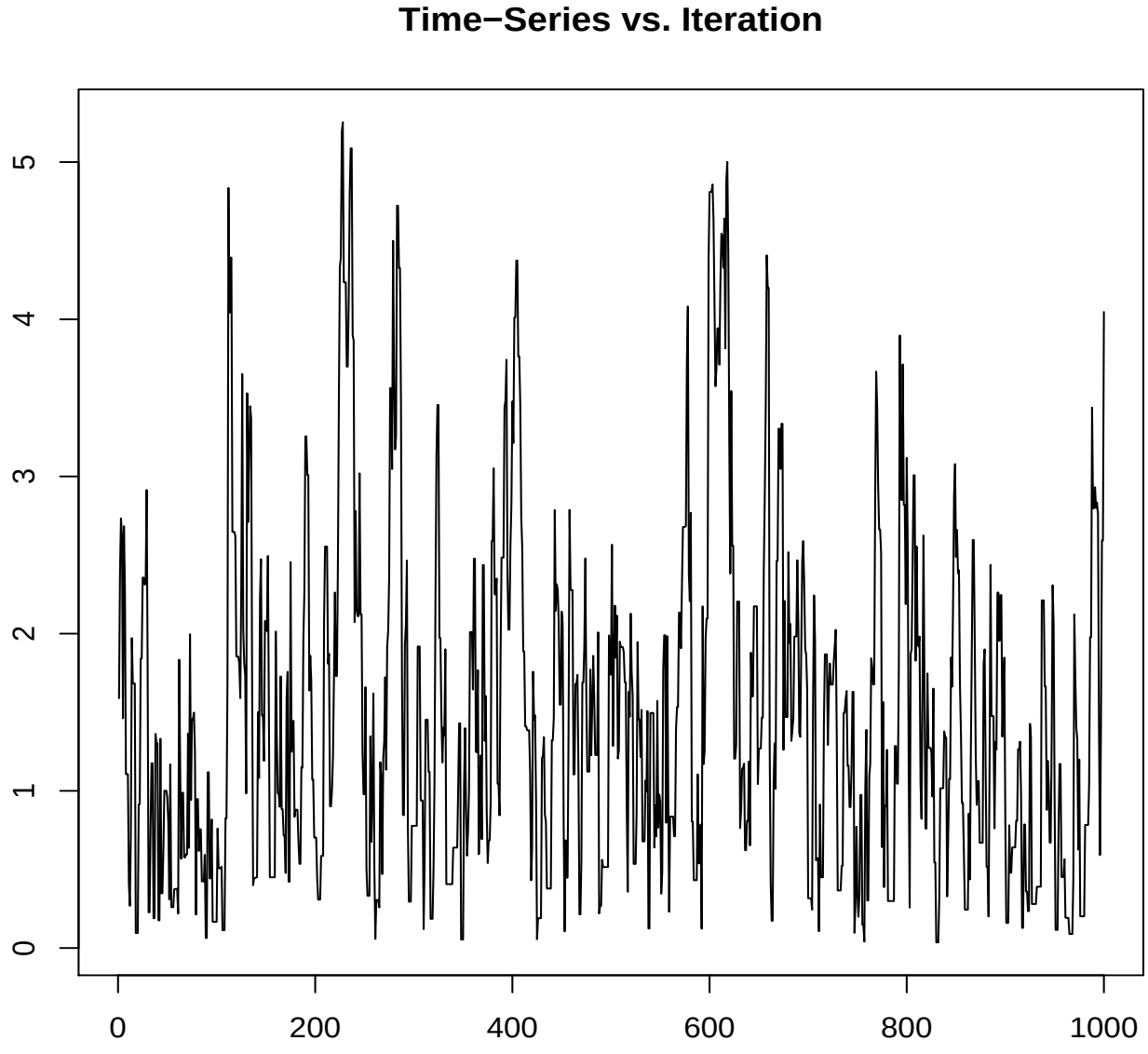
This has several consequences.

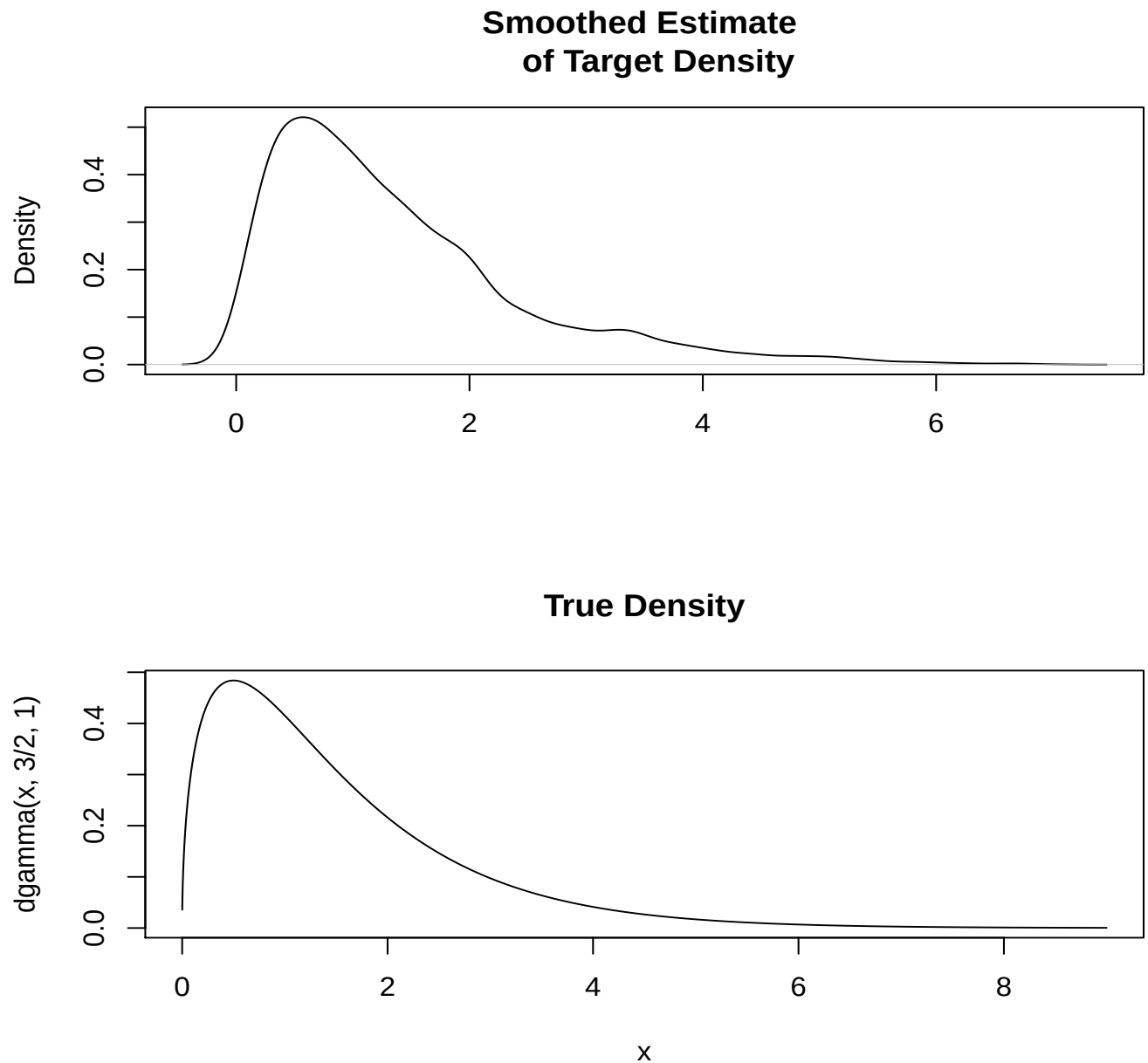
- Each $X_i \sim F_i$.
- F_i is not the target distribution but for very large i it is "close" to the target.
- The observations are positively correlated.
- Much larger simulation sizes are needed as compared to GOFMC.

MCMC in R

The following R code will produce a time series plot of the first 1000 iterations of the simulation and plots to compare the estimated density with the target density.

```
> ts.plot(out$batch[0:1e3], main="Time-Series vs.
   Iteration", xlab="", ylab="", xlim=c(0, 1e3))
> par(mfrow=c(2,1))
> plot(density(out$batch), main="Smoothed Estimate
   of Target Density")
> x<-1:9e3
> x<-x/1e3
> plot(x,dgamma(x,3/2,1),main ="True Density", type="l")
```


Time−Series vs. Iteration
0
1
2
3
4
5
0
200
400
600
800
1000


Smoothed Estimate
of Target Density
Density
0.0
0.2
0.4
0
2
4
6
True Density
dgamma(x, 3/2, 1)
0.0
0.2
0.4
0
2
4
6
8
x

Fundamental MCMC Algorithm

Let f be an arbitrary density function.

Metropolis-Hastings

Suppose the current state is $X_k = x$. Then

1. $X^* \sim \mathrm{N}(x, \sigma^2)$
2. $$\alpha(x, x^*) = 1 \ \wedge \ \frac{f(x^*)}{f(x)}$$
3. $$X_{k+1} = \begin{cases} x^* & \text{w. p. } \alpha(x, x^*) \\ x & \text{w. p. } 1 - \alpha(x, x^*) \end{cases}$$

Effective MH boils down to the choice of σ^2.

Where do we go from here?

- There are many variants on Metropolis-Hastings–in fact, infinitely many–how can we know we have a good algorithm?
- Can we estimate the error in our approximation?
- The output of an MCMC simulation is random–how large should the simulation be?

Finding a good algorithm–choosing σ^2

This is a Goldilock's problem–we need σ^2 to be not too large and not too small.

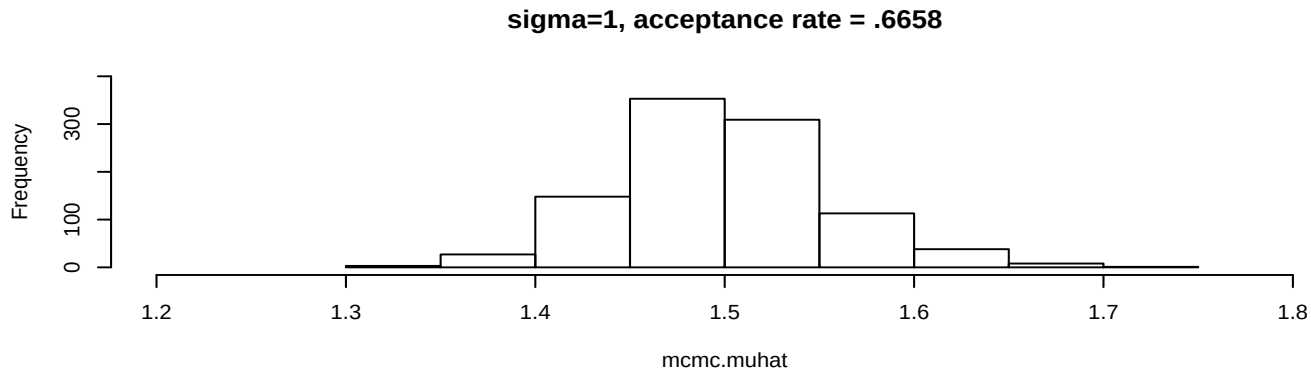
There are some theoretical results that suggest the acceptance rate should be somewhere around 20% to 50%.

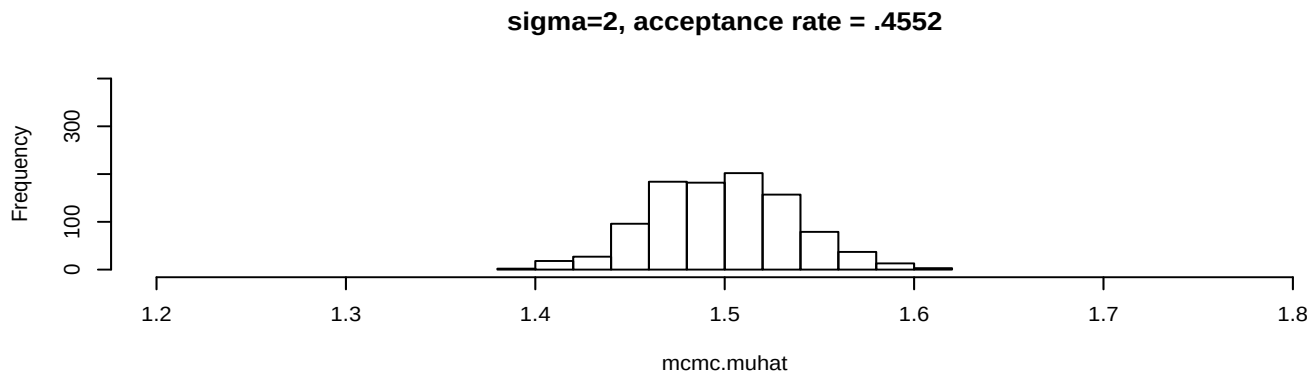
Recall our simple problem. Suppose we want to estimate $E(X)$ where $X \sim \text{Gamma}(3/2, 1)$. That is,

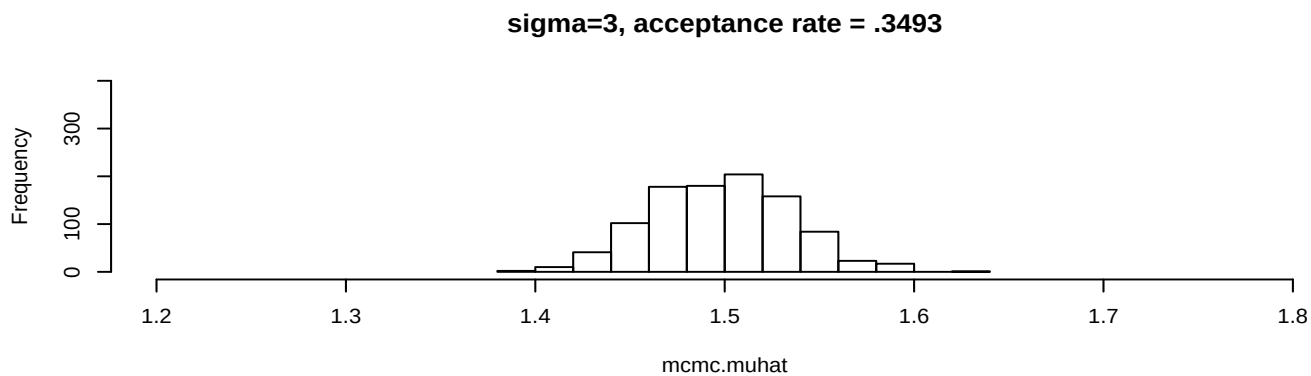
$$\int_0^\infty x \frac{2\sqrt{x}}{e^x \sqrt{\pi}} dx = \frac{3}{2} .$$

We will consider Metropolis-Hastings with $\sigma^2 = 1, 4, 9$.

Finding a good algorithm–choosing σ^2







Monte Carlo standard error

MCMC methods simulate *dependent* sequences. This means that the sample standard deviation is NOT a valid way of assessing the error in our estimate of $E(X)$.

Just as with GOFMC, we want to be able to construct interval estimators of $E(X)$

$$\bar{x}_n \pm t_* SE$$

From the histograms on the previous page it seems reasonable to conclude that the sample mean will be approximately normally distributed. So estimating the error boils down to estimating the variance of that asymptotic normal distribution. Batch means is one technique for doing this.

The `mcmcse` package will produce SE for us. It uses the method of batch means.

Monte Carlo standard error

```
library(mcmc)
library(mcmcse)

set.seed(528)
nsim<-1e6
lu<-function(w){dgamma(w, 3/2, 1, log=TRUE)}
out<-metrop(lu, init=1, nsim, scale=3)
mcse(out$batch)
$est
[1] 1.498239

$se
[1] 0.003559984
```

Monte Carlo standard error

```
> out1<-mcse(out$batch)
> out1$est - 2 * out1$se
[1] 1.491119
> out1$est + 2 * out1$se
[1] 1.505359
```

An approximate 95% confidence interval for $E(X)$ is given by $(1.491, 1.505)$. Recall that the truth is E(X)=1.5.

Terminating the simulation

MCMC typically requires a larger simulation effort than GOFMC. In my previous example I used 1e4 for GOFMC and 1e6 for MCMC.

Q: In MCMC, when should we terminate the simulation? When has the computer run long enough?

A: When the Monte Carlo standard error is sufficiently small. Equivalently, when the width of the confidence interval is sufficiently narrow.

Terminating the simulation

nsim	$\bar{x}_{\text{nsim}}$	MCSE	95% CI	Width
1e3	1.499	0.098	(1.302, 1.695)	0.393
1e4	1.444	0.031	(1.384, 1.505)	0.121
1e5	1.505	0.010	(1.484, 1.526)	0.042
1e6	1.498	0.004	(1.491, 1.505)	0.014

More simulation does not necessarily mean you have a better estimate.

Precision of estimates increases as nsim increases.

Without the MCSE we would have no idea about the quality of our estimates.

Further Reading / References

Flegal JM, Haran M, and Jones GL. "Markov chain Monte Carlo: Can we trust the third significant figure?" *Statistical Science*

Flegal JM and Jones GL "Implementing MCMC: Estimating with Confidence" *Handbook of Markov Chain Monte Carlo*

Geyer CJ, "Introduction to Markov chain Monte Carlo" *Handbook of Markov Chain Monte Carlo*

`mcmc`
http://cran.r-project.org/web/packages/mcmc/index.html

`mcmcse`
http://cran.r-project.org/web/packages/mcmcse/index.html