

# Metropolis Hastings Algorithm

*Michael Rose*

*November 3, 2018*

## Metropolis Algorithm

### Question

Suppose we have data  $X_1, \dots, X_n$  which we believe come from a normal distribution with mean  $\theta$  and variance

1. Suppose we are uncertain about  $\theta$ , and for us  $\theta$  has the following Cauchy Distribution:

$$f(\theta) = \frac{1}{\pi(1+\theta^2)}, -\infty < \theta < \infty$$

This is a special form of the Cauchy called the standard Cauchy Distribution with parameters  $x_0 = 0, \gamma = 1$ .

We will write a Metropolis Hastings Algorithm whose limiting distribution is our posterior distribution:

$$f(x, \theta, \sigma^2 = 1) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-\theta)^2}{2}} \frac{1}{\pi(1+\theta^2)}$$
$$X_i \sim \text{Normal}(\theta, 1)$$
$$\theta \sim \text{Cauchy}(0, 1)$$

First lets simulate some data

```
set.seed(103)
# choose a cauchy sample for mean
theta_x <- rcauchy(1, location = 0, scale = 1)
theta_x

## [1] 0.8060194

# generate samples
data_y <- rnorm(n = 100000, mean = theta_x, sd = 1)
```

Now we can create our metropolis algorithm:

The Metropolis Algorithm calls for  $\theta_{prop}$  to be sampled from a symmetric proposal distribution centered at the current parameter value,  $\theta_{curr}$ . For this task we will use  $\theta_{prop} \sim \text{Normal}(\theta_{curr}, \sigma^2)$ .

The proposal distribution is separate and distinct from either the prior or posterior distribution for the parameter. The proposal distribution's sole purpose is to give candidate parameter values to *try* and

potentially accept as a valid sample from the posterior distribution of  $\theta$ .

- `samples` is the number of samples we want to draw from the posterior distribution and determines the length of the resulting MCMC chain
- `theta_start` gives us a  $\theta$  to start the algorithm
- `sd` is the standard deviation of the proposal distribution

Within the function, we construct a for loop that repeatedly draws  $\theta_{prop}$  from a standard normal proposal distribution (using `rnorm`). It then computes the ratio of Bayes' numerators and carries out the accept / reject logic. We store the results in a vector called `posterior_thetas` which are initialized to NA.

```
metropolis_algo <- function(samples, theta_start, sd){  
  # declarations  
  theta_curr <- theta_start  
  # vector of NAs to store sampled parameters  
  posterior_thetas <- rep(NA, times = samples)  
  
  for (i in 1:samples){  
    # proposal distribution  
    theta_prop <- rnorm(n = 1, mean = theta_curr, sd = sd)  
  
    # if proposed parameter is outside range, set to current value. Else keep proposed value  
    theta_prop <- ifelse((theta_prop < 0 | theta_prop > 2), theta_curr, theta_prop)  
  
    # bayes numerators  
    posterior_prop <- dcauchy(theta_prop, location = 0, scale = 1) *  
      dnorm(data_y, mean = theta_prop, sd = 0.5)  
    posterior_curr <- dcauchy(theta_curr, location = 0, scale = 1) *  
      dnorm(data_y, mean = theta_curr, sd = 0.5)  
  
    # calculate probability of accepting  
    p_accept_theta_prop <- min(posterior_prop / posterior_curr, 1.0)  
  
    rand_unif <- runif(n=1)  
  
    # probabilistically accept proposed theta  
    theta_select <- ifelse(p_accept_theta_prop > rand_unif, theta_prop, theta_curr)
```

```

posterior_thetas[i] <- theta_select

  # reset theta_curr for the next iteration of the loop
  theta_curr <- theta_select
}
return(posterior_thetas)
}

```

Now we can try 100,000 samples with a starting value of 0.9 and a sd for our normal proposal distribution of 0.5

```

set.seed(999)
posterior_thetas <- metropolis_algo(samples = 100000, theta_start = 0.9, sd = 0.5)

```

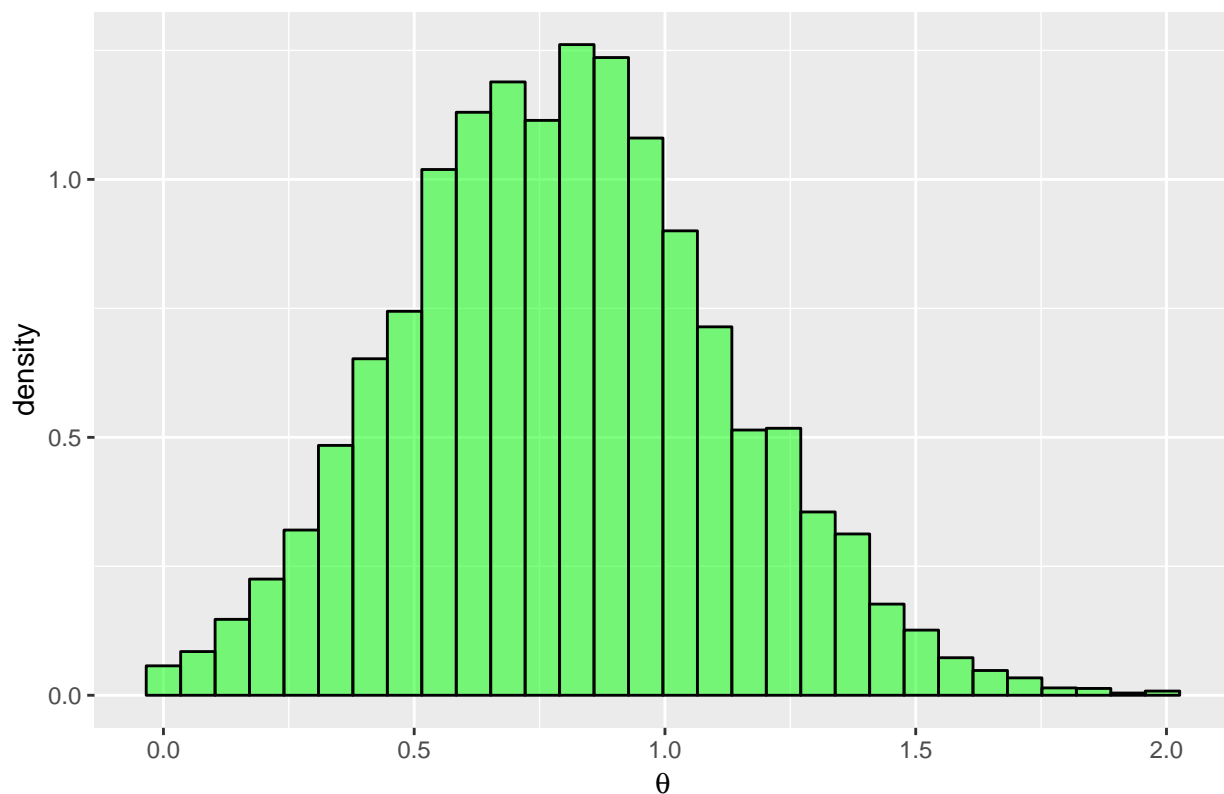
Lets take a look at the kernel density estimate of the posterior.

```

ggplot() + geom_histogram(aes(x = posterior_thetas, y = ..density..), color = "black", fill = "green",
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

```

Given data with  $\theta = 0.8$



# Gibbs Sampler

## Problem

Suppose we have a **Zero-Inflated Poisson Model**. In this model, random data  $X_1, \dots, X_n$  are of the form  $X_i = R_i Y_i$  where  $Y_i \sim \text{Poisson}(\lambda)$  and  $R_i \sim \text{Bernoulli}(p)$ , and our samples are iid.

If given an outcome  $x = (x_1, \dots, x_n)$ , our goal is to estimate  $\lambda$  and  $p$ .

A zero-inflated Poisson model is used when we have a random event containing excess zero-count data in unit time. For example, the number of insurance claims within a population for a certain type of risk would be zero-inflated by those people who have not taken out insurance against the risk and thus are unable to claim.

Using a hierarchical Bayes model:

---

$p \sim \text{Uniform}(0, 1) \mid \text{Prior for } p$

$(\lambda|p) \sim \text{Gamma}(a, b) \mid \text{Prior for } \lambda$

$(r_i|p, \lambda) \sim \text{Bernoulli}(p) \mid \text{Independently (from model above)}$

$(x_i|r, \lambda, p) \sim \text{Poisson}(\lambda r_i) \mid \text{Independently (from model above)}$

Given  $a, b, r = (r_1, \dots, r_n)$ , we have the posterior:

$$f(x, r, \lambda, p) = \frac{b^\alpha \lambda^{\alpha-1} e^{-b\lambda}}{\Gamma(\alpha)} \prod_{i=1}^n \frac{e^{-\lambda r_i} (\lambda r_i)^{x_i}}{x_i!} p^{r_i} (1-p)^{1-r_i}$$

---

We want to use Gibbs sampling to sample from our posterior pdf  $f(\lambda, p, r|x)$ . First though, we must learn the full conditional distributions for  $\lambda$ ,  $p$ , and  $r_i$ .

Given  $f(x, r, \lambda, p)$ , the full conditional densities are all proportional to our joint density as functions of  $\lambda, p, r_i$ .

For instance :

$$\pi(\lambda|p, r, x) \propto \lambda^{\alpha-1} e^{-b\lambda} \prod_{i=1}^n e^{-\lambda r_i} (\lambda)^{x_i} \propto \lambda^{\alpha-1} e^{-b\lambda} e^{-\lambda \sum_i r_i} \lambda^{\sum_i x_i}$$

Where only terms depending on  $\lambda$  are kept. Then

$$\pi(\lambda|p, r, x) \propto \lambda^{\alpha-1 + \sum_i x_i} e^{-\lambda[b + \sum_i r_i]}. \text{ Therefore}$$

$$\lambda|p, r, x \sim \text{Gamma}(a + \sum_i x_i, b + \sum_i r_i)$$

Similarly for  $p$ :

$$\pi(p|\lambda, r, x) \propto \prod_{i=1}^n p^{r_i} (1-p)^{1-r_i} = p^{\sum_i r_i} (1-p)^{n - \sum_i r_i}, \text{ leading to:}$$

$$p|\lambda, r, x \sim \text{Beta}(1 + \sum_i r_i, n + 1 - \sum_i r_i).$$

Similarly for  $r_i$ :

Note that only the  $x_i$ 's for which  $r_i = 1$  need to be simulated. We can write the full conditional density for  $r_i$  as such:

$$\pi(r_i|\lambda, x, p) \propto \prod_{i=1}^n e^{-r_i \lambda} (\lambda r_i) p^{r_i} (1-p)^{1-r_i} \propto \frac{p e^{-\lambda}}{p e^{-\lambda} + (1-p) I\{x_i=0\}}. \text{ Then}$$

$$r_i|\lambda, p, x \sim \text{Bernoulli}\left(\frac{p e^{-\lambda}}{p e^{-\lambda} + (1-p) I\{x_i=0\}}\right).$$

Now, given our 3 full conditionals, we can run our gibbs sampler:

```
# simulate x_i

# num samples
n = 10^5

# given data p
p = 0.3

# given data lambda
lam = 2

# given r values for data
r = as.integer(runif(n) < p)

# create vector
x = rep(0, n)

# constrain to r = 1, sample from poisson into vector
x[r == 1] = rpois(sum(r == 1), lam)

# Gibbs Sampler

# given values
a <- 2
b <- 2

# number of iterations
T <- 10^4

# initial lambda, p
lamb <- pe <- rep(0.5, T)
```

```

# sampler
for (t in 2:T){
  # r full conditional
  r = (x == 0) * (runif(n) < 1 / (1 + (1 - pe[t-1]) / (pe[t-1]*exp(-lamb[t-1])))) + (x>0)
  # lambda full conditional
  lamb[t] = rgamma(1, a + sum(x), b + sum(r))
  # p full conditional
  pe[t] = rbeta(1, 1 + sum(r), n - sum(r) + 1)
}

# plot
lambda_plot <- ggplot() + geom_histogram(aes(x = lamb, y = ..density..), color = "black", fill = "blue")

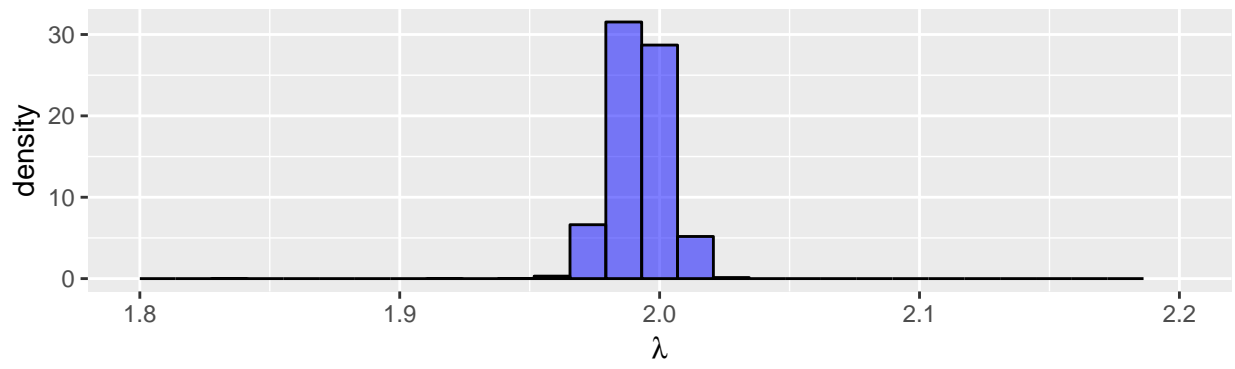
p_plot <- ggplot() + geom_histogram(aes(x = pe, y = ..density..), color = "black", fill = "green", alpha = 0.5)
  ggtitle("Given data with p = 0.3") + xlim(c(0.29, 0.31))

grid.arrange(lambda_plot, p_plot, ncol = 1)

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

```

Given data with  $\lambda = 2$



Given data with  $p = 0.3$

