Bayesian Regression and the Metropolis Algorithm

Math 392

Simple Linear Regression with Gaussian Errors

Let Y be a r.v., X be fixed, and $\theta = \{\beta_0, \beta_1, \sigma^2\}$ are parameters.

$$Y|X, \theta \sim N(\beta_0 + \beta_1 x, \sigma^2)$$

Frequentists ask

What sort of $\hat{\theta}$ would we get under hypothetical resampling?

Bayesians ask

What is our sum knowledge of θ based on the data and prior information?

Prior Considerations

- What is the support of θ?
- Flat or mounded?
- Conjugate?
  - $\sigma^2 \sim InvChisq()$
  - $\beta|\sigma^2 \sim N()$
- Correlated?

Let's use:

$$
\begin{aligned}
\beta_0 &\sim Unif(-10, 10) \\
\beta_1 &\sim Normal(\mu = 0, \tau^2 = 25) \\
\sigma^2 &\sim Unif(0, 10)
\end{aligned}
$$

Your turn

Please write out the expression for the full joint distribution.

Calculating the Posterior

Full Joint

$$
\begin{aligned}
f(Y, \beta_0, \beta_1, \sigma^2) &= f(Y|\beta_0, \beta_1, \sigma^2) f(\beta_0) f(\beta_1) f(\sigma^2) \\
&= \left[\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{1}{2\sigma}(Y_i - \beta_0 - \beta_1 x)^2} \right] \left[\frac{1}{20} \right] \left[\frac{1}{\sqrt{2\pi\tau^2}} e^{\frac{1}{2\sigma}(\beta_1)^2} \right] \left[\frac{1}{10} \right]
\end{aligned}
$$

Full Conditional

$$
f(\beta_0, \beta_1, \sigma^2 | Y) = c f(Y, \beta_0, \beta_1, \sigma^2)
$$

Metropolis Algorithm

Let $f(\theta)$ be a target density that you wish to sample from. Let $J(\theta|\theta_i, \tau^2)$ be a jumping distribution that is symmetric: $J(\theta_a|\theta_b) = J(\theta_b|\theta_a)$.

1. Select an initial value θ_0 s.t. $f(\theta_0) > 0$
2. For $i = 1, 2, \dots$
   a) Sample a *proposal* θ_* from $J(\theta_*|\theta_{i-1})$
   b) Calculate the ratio of densities

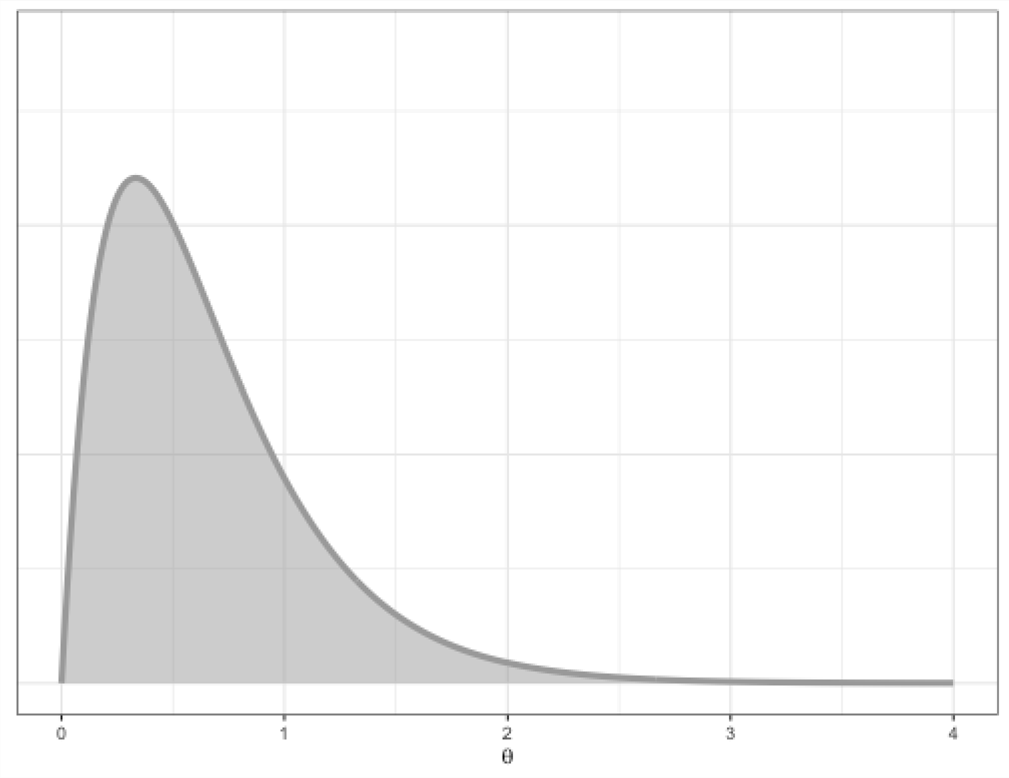
$$r = \frac{f(\theta_*)}{f(\theta_{i-1})}$$

3. Set:

$$\theta_i = \begin{cases} \theta_* & \text{with probability } min(r, 1) \\ \theta_{i-1} & \text{otherwise} \end{cases}$$

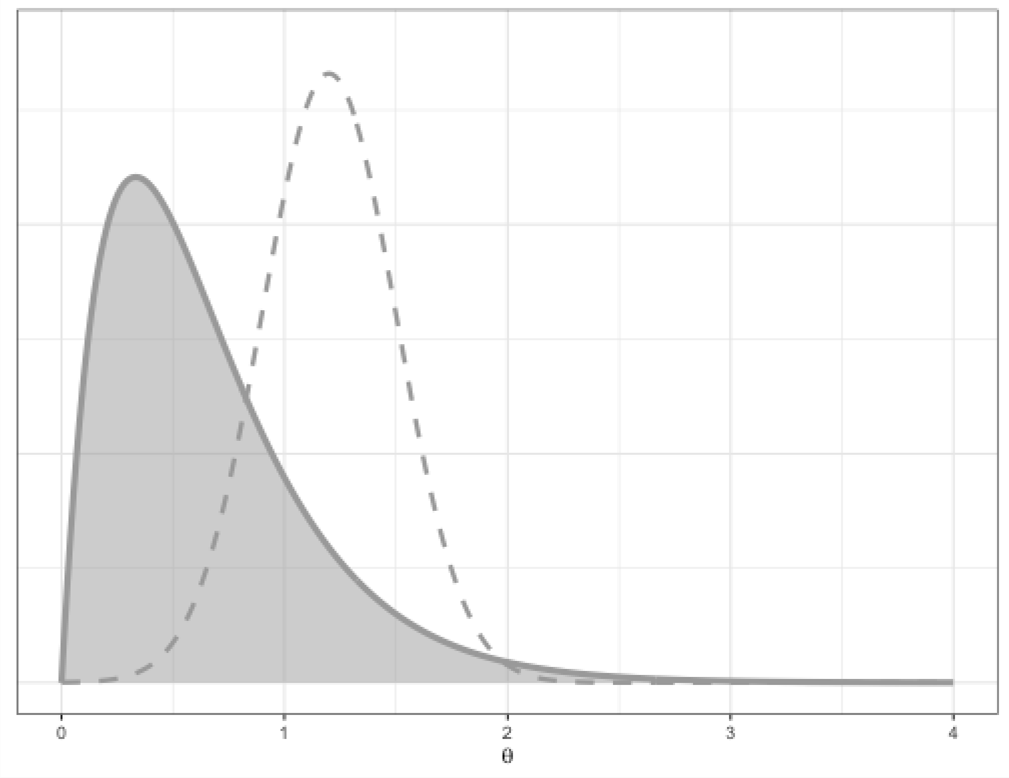
Example: Sampling from the Gamma

$$\theta \sim Gamma(\alpha = 2, \beta = 3)$$



Proposal Distribution

$$J(\theta|\theta_i, \tau^2) \sim N(\theta_0, \tau^2 = 3^2)$$



Metropolis Algorithm

1. Select an initial value θ_0 s.t. $f(\theta_0) > 0$
2. For $i = 1, 2, \ldots$
   a) Sample a *proposal* θ_* from $J(\theta_*|\theta_{i-1})$
   b) Calculate the ratio of densities

$$r = \frac{f(\theta_*)}{f(\theta_{i-1})}$$

3. Set:

$$\theta_i = \begin{cases} \theta_* & \text{with probability } min(r, 1) \\ \theta_{i-1} & \text{otherwise} \end{cases}$$

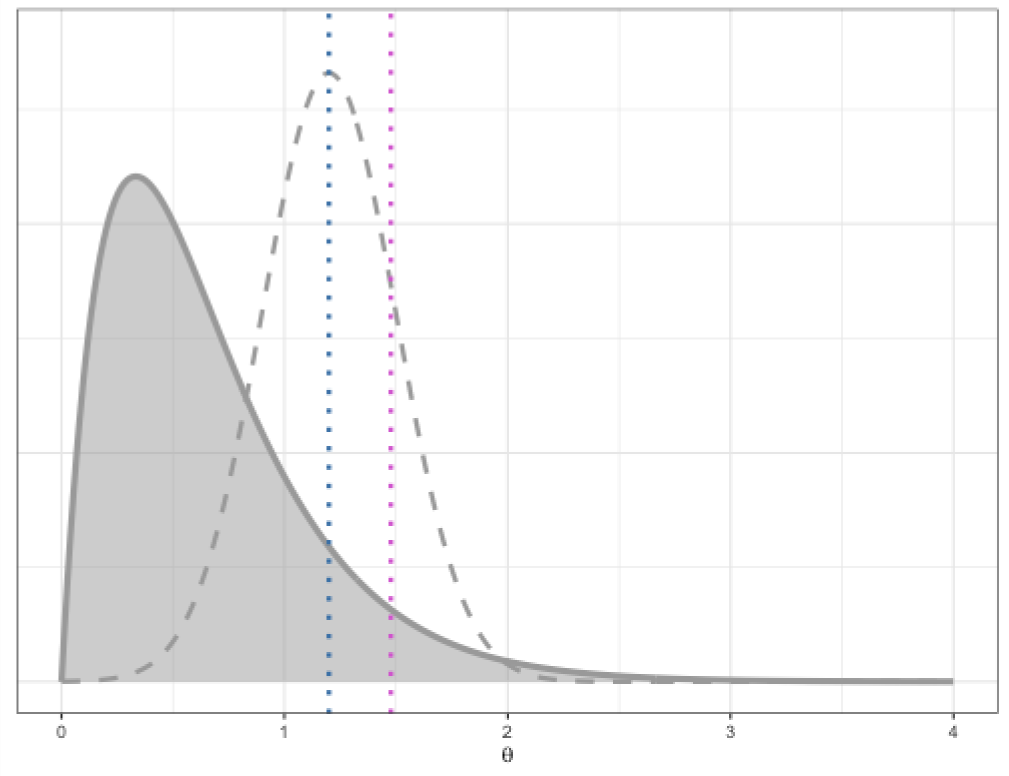
Initialize θ_0

```r
theta_0 <- 1.2
```

A modest proposal

```r
theta_star <- rnorm(1, theta_0, .3)
```

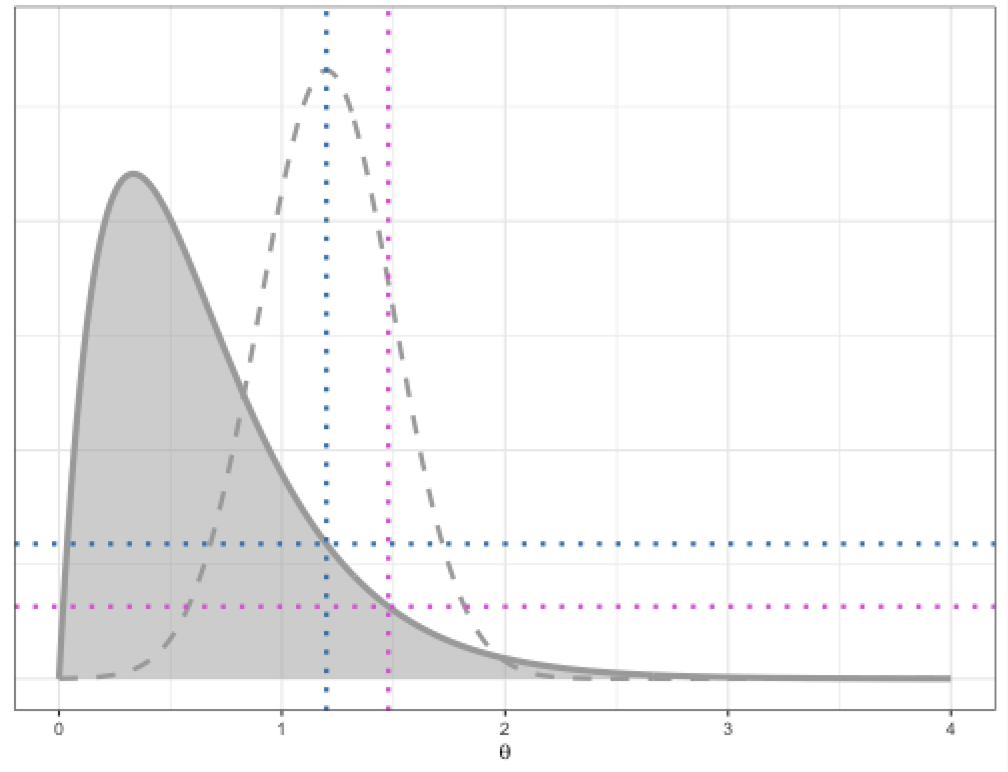
```
## [1] 1.477938
```



Calculate the ratio

```
r <- dgamma(theta_star, 2, 3)/dgamma(theta_0, 2, 3)
```

```
## [1] 0.5350007
```



Accept?

```
runif(1) < min(r, 1)
```

```
## [1] FALSE
```

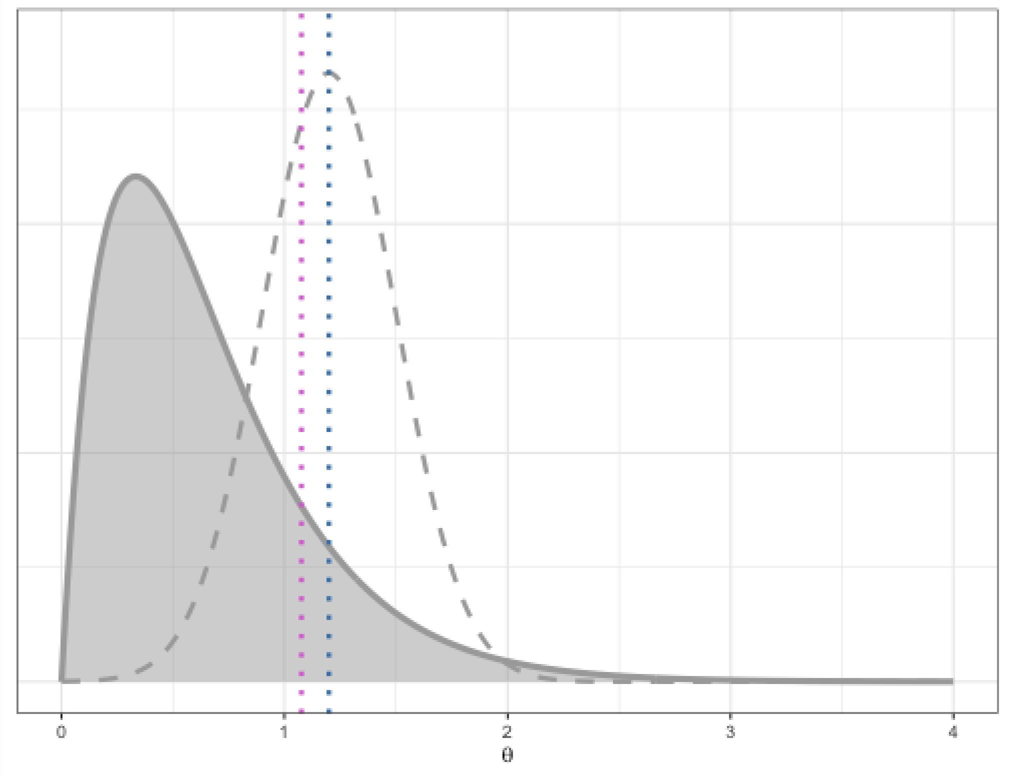
So we set the new center of the jumping distribution to the previous value:

```
theta_1 <- theta_0
```

A second proposal

```
theta_star <- rnorm(1, theta_1, .3)
```

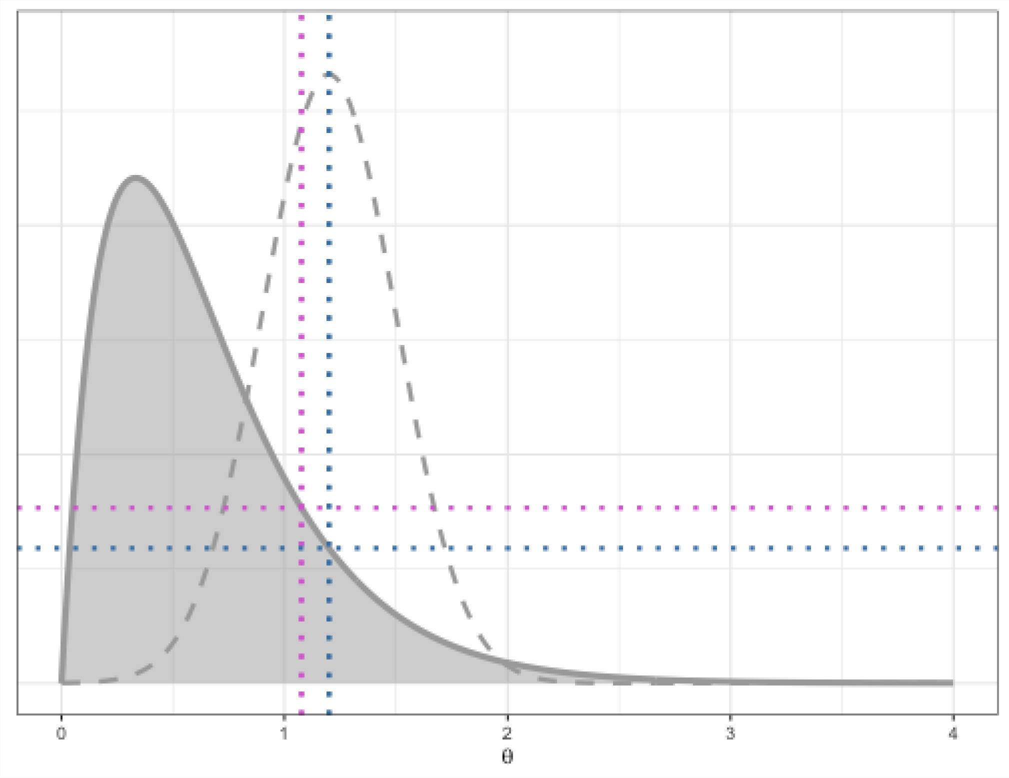
```
## [1] 1.076791
```



Calculate the ratio

```
r <- dgamma(theta_star, 2, 3)/dgamma(theta_1, 2, 3)
```

```
## [1] 1.298604
```



Accept?

```
runif(1) < min(r, 1)
```

```
## [1] TRUE
```

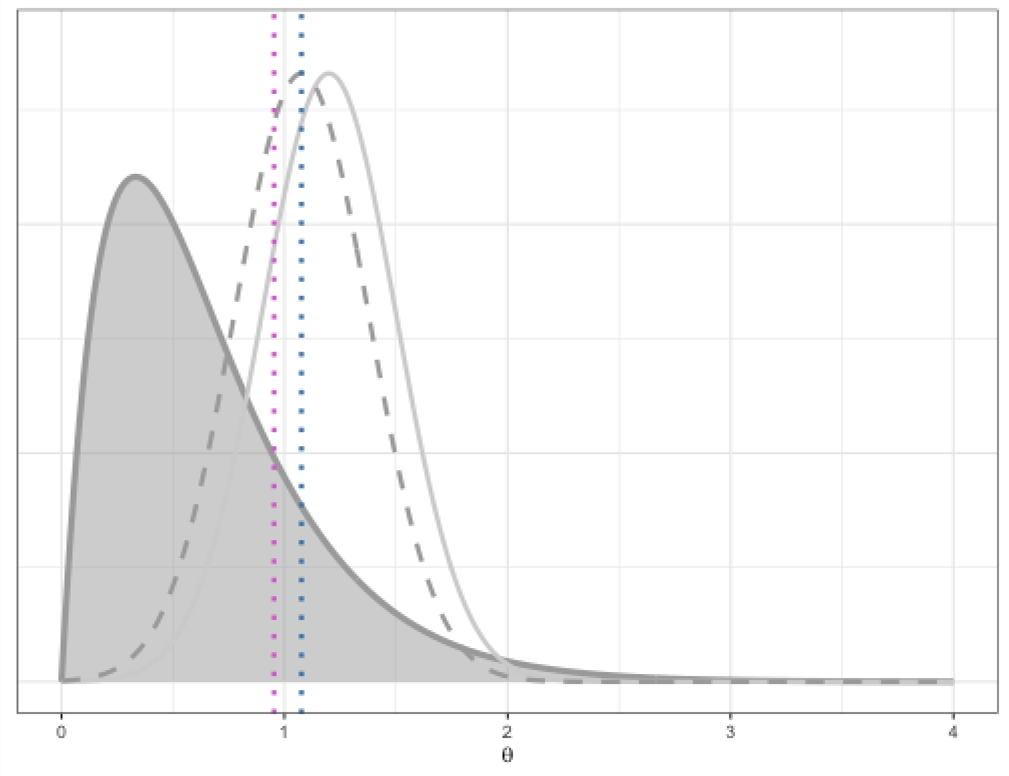
So we set the new center of the jumping distribution to the proposed value:

```
theta_2 <- theta_star
```

A third proposal

```r
theta_star <- rnorm(1, theta_2, .3)
```

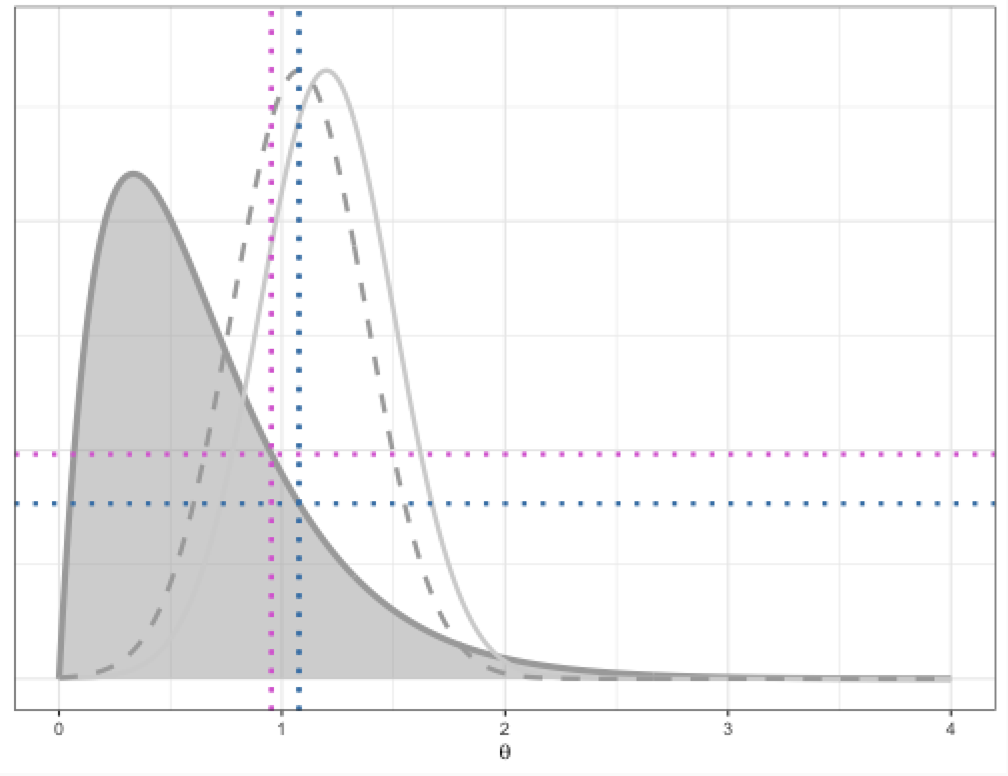
```
## [1] 0.9535826
```



Calculate the ratio

```
r <- dgamma(theta_star, 2, 3)/dgamma(theta_2, 2, 3)
```

```
## [1] 1.281603
```



Accept?

```
runif(1) < min(r, 1)
```

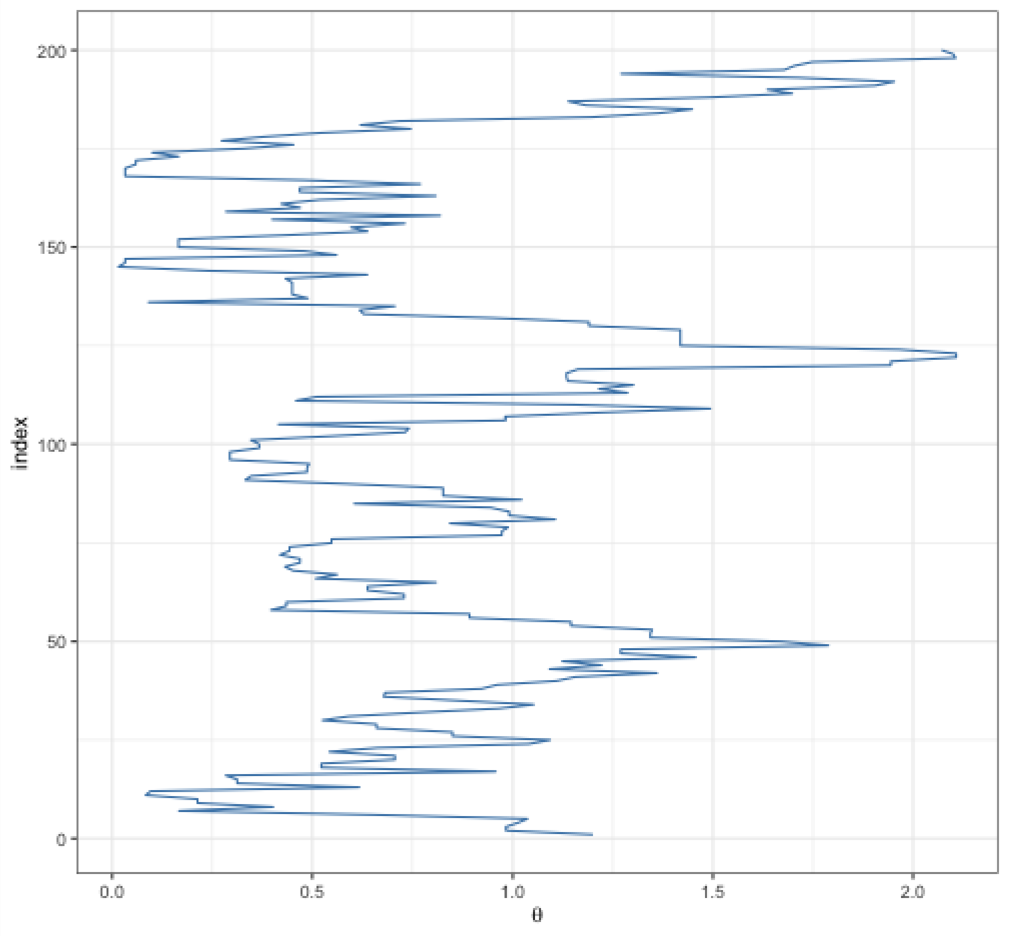
```
## [1] TRUE
```

Iterated algorithm

```
theta_0 <- 1.2
tau <- .3
it <- 50000
chain <- rep(NA, it + 1)
chain[1] <- theta_0
for (i in 1:it) {
  proposal <- rnorm(1, chain[i], tau)
  p_move <- min(dgamma(proposal, 2, 3)/
                  dgamma(chain[i], 2, 3),
                1)
  chain[i + 1] <- ifelse(runif(1) < p_move,
                         proposal,
                         chain[i])
}
head(chain)
```

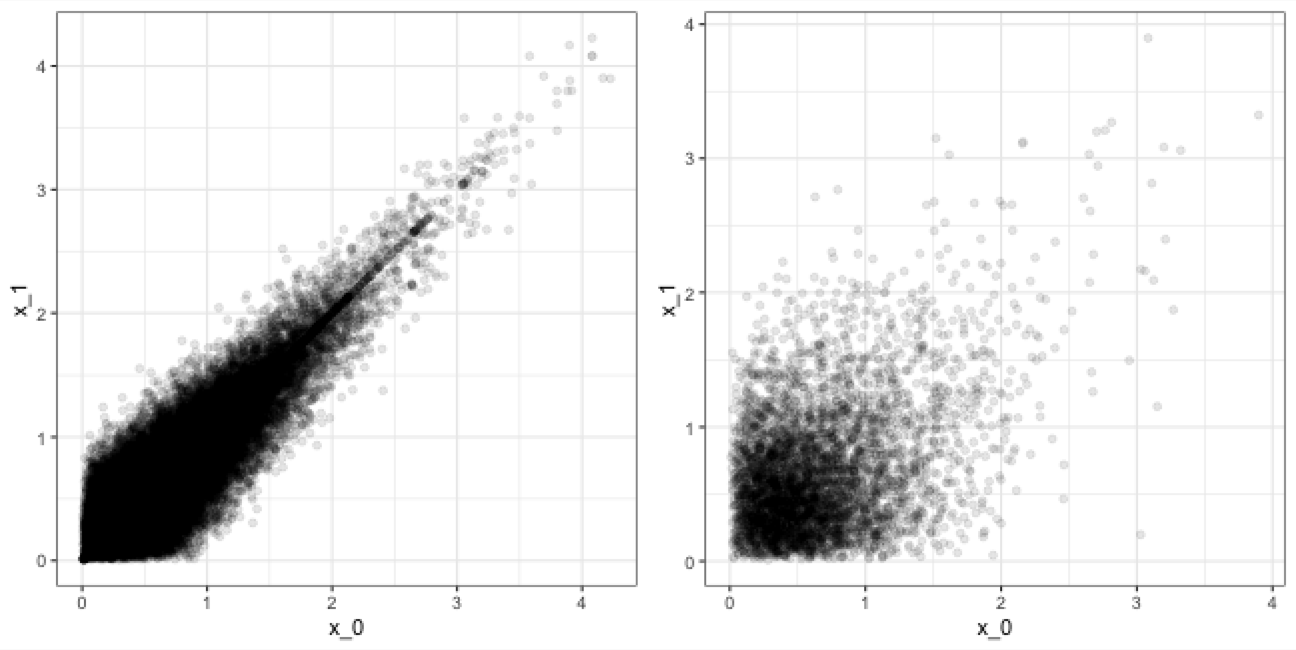
```
## [1] 1.2000000 0.9841212 0.9841212 1.0130421 1.0364991 0.6582159
```

The burn-in period

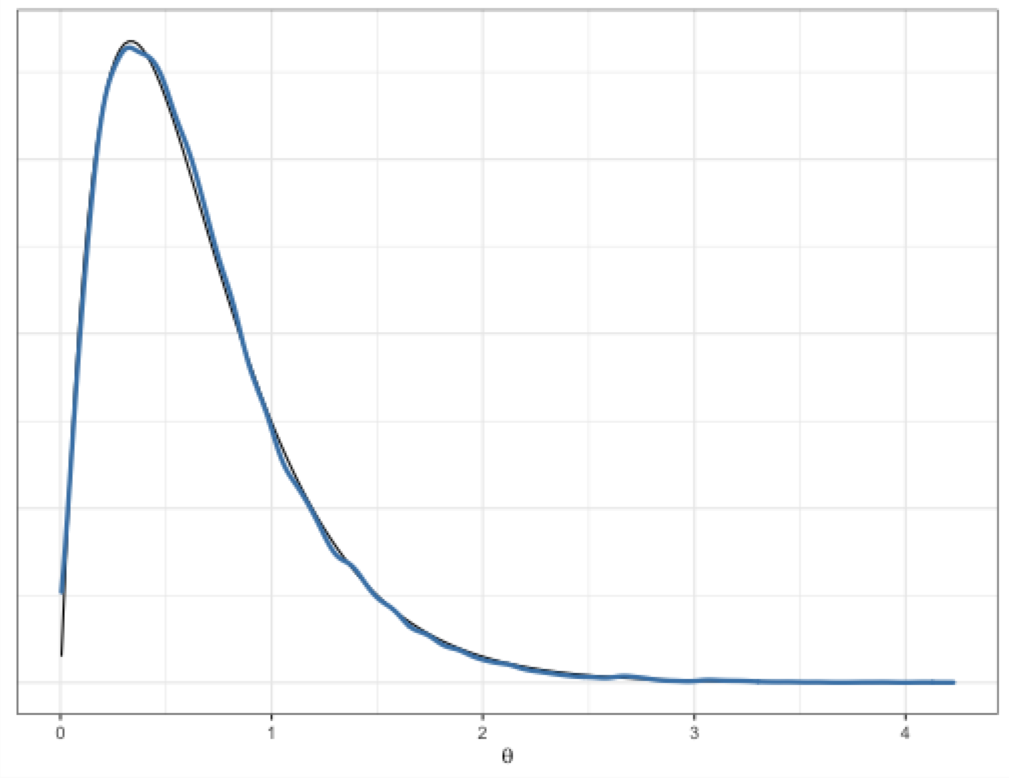


Thinning

There is strong auto-correlation, so we decimate.



Distribution of samples



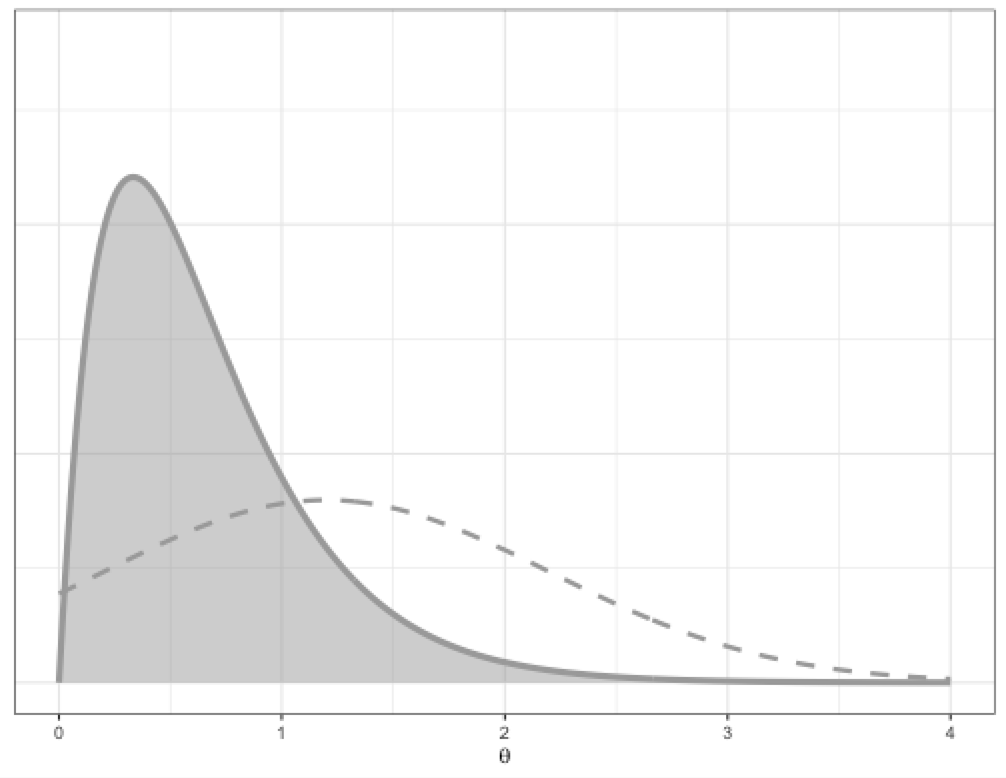
Acceptance rate

```
(acceptance <- 1 - mean(duplicated(chain[-(1:burn_in)])))
```

```
## [1] 0.7503833
```

- Recommended acceptance rate is 30%-40% - why?
- How can we adjust the acceptance rate?

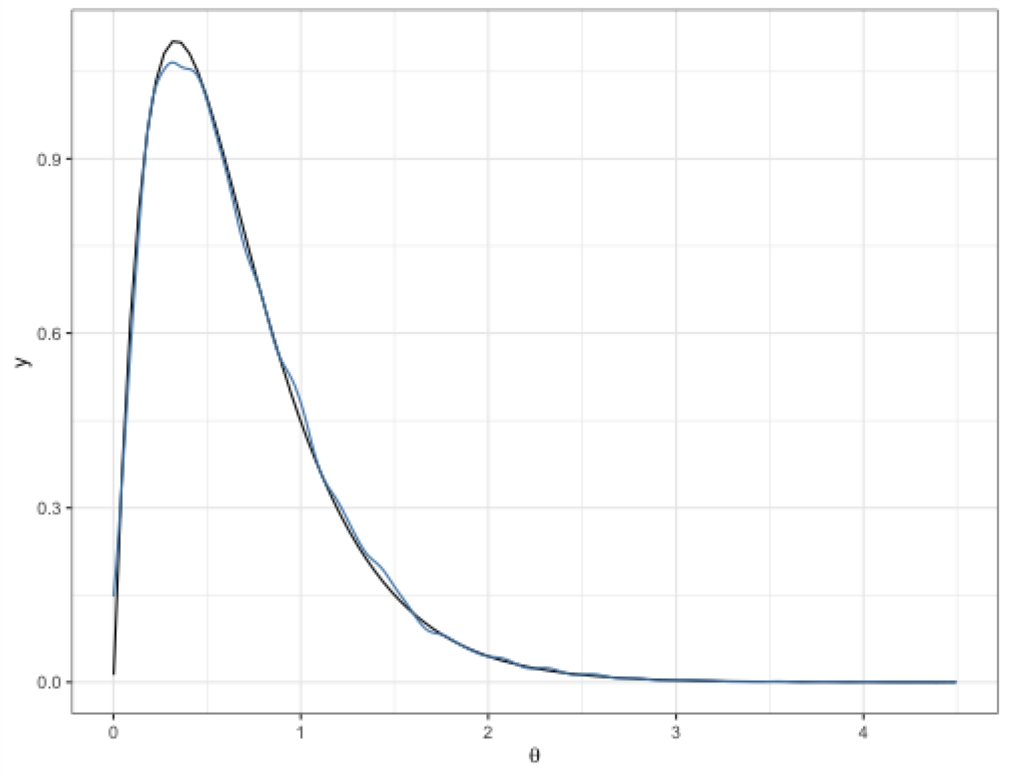
High variance jump



New MC chain

```
theta_0 <- 1.2
tau <- 1
it <- 50000
chain <- rep(NA, it + 1)
chain[1] <- theta_0
for (i in 1:it) {
  proposal <- rnorm(1, chain[i], tau)
  p_move <- min(dgamma(proposal, 2, 3)/
                  dgamma(chain[i], 2, 3),
                1)
  chain[i + 1] <- ifelse(runif(1) < p_move,
                         proposal,
                         chain[i])
}
head(chain)
```

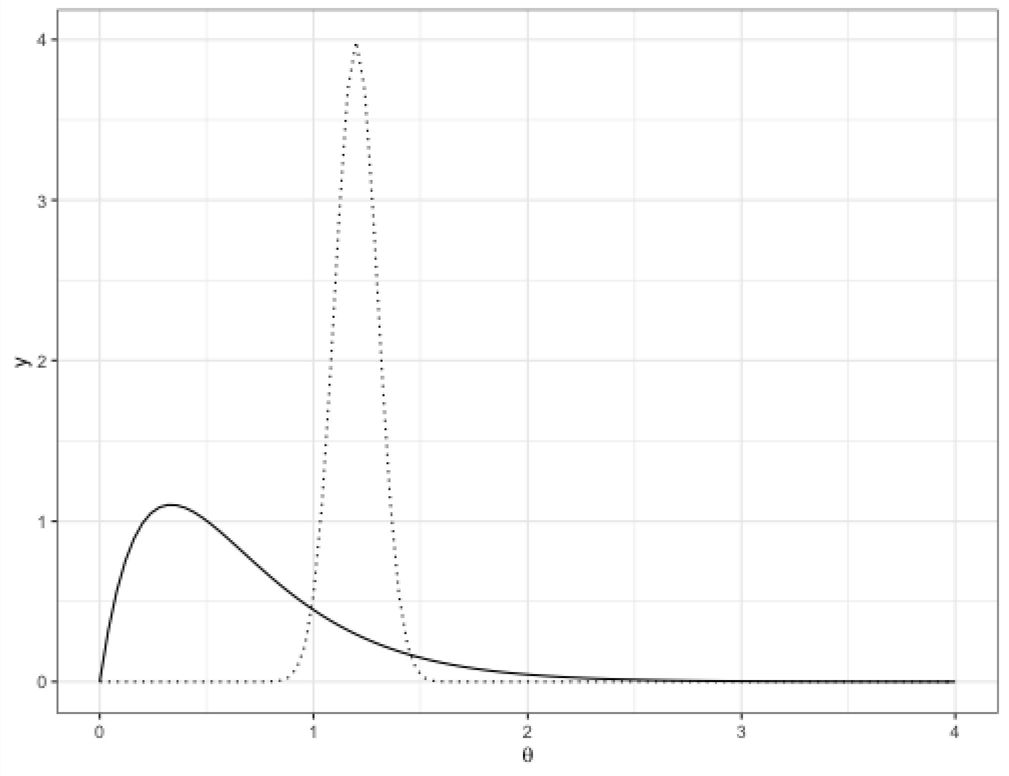
```
## [1] 1.200000 1.636454 1.636454 1.636454 1.014004 1.014004
```

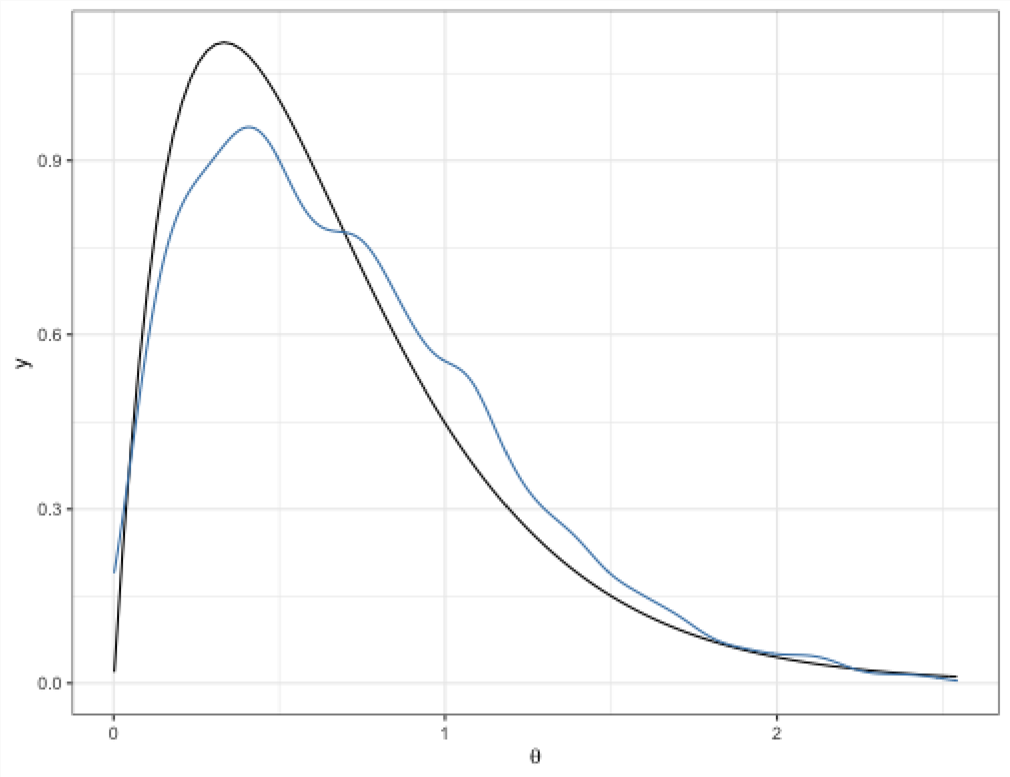



```r
(acceptance <- 1 - mean(duplicated(chain[-(1:burn_in)])))
```

```
## [1] 0.4101242
```

Low variance jump





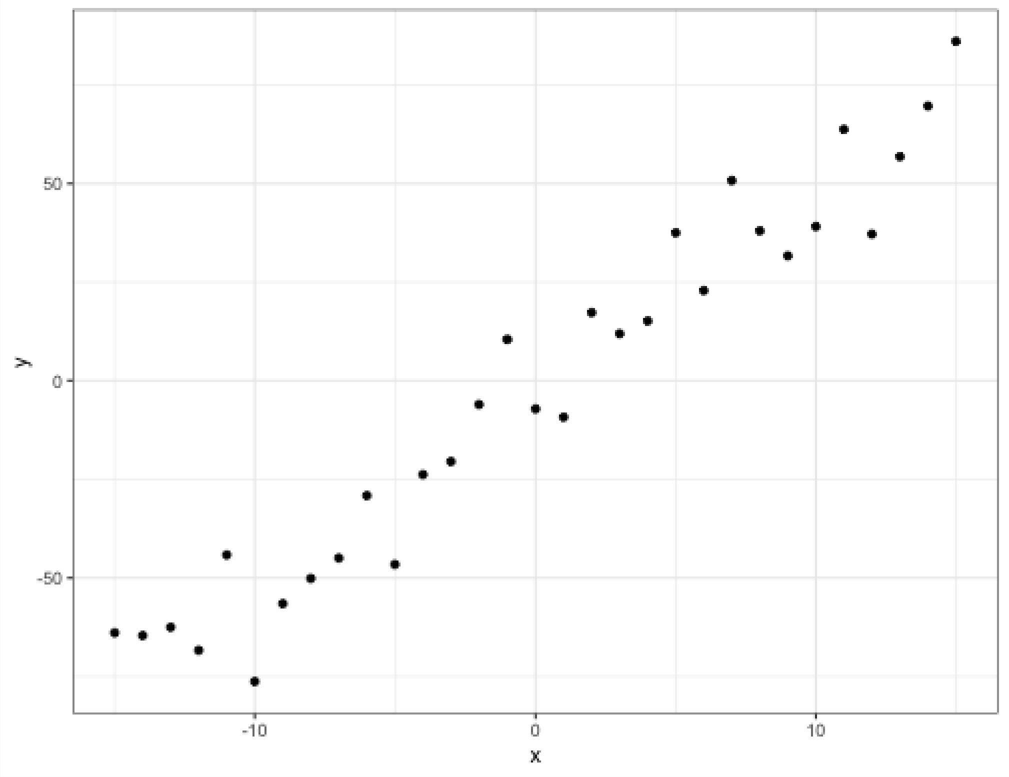

```
(acceptance <- 1 - mean(duplicated(chain[-(1:burn_in)])))
```

```
## [1] 0.9082184
```

Bayesian Regression

Begin by generating data.

```
set.seed(79)
B0 <- 0
B1 <- 5
sigma <- 10
n <- 31
x <- (-(n-1)/2):((n-1)/2)
y <-  B0 + B1 * x + rnorm(n, mean = 0, sd = sigma)
```


50
0
-50
y
-10
0
10
x

The Likelihood

```
likelihood <- function(theta) {
    B0 <- theta[1]
    B1 <- theta[2]
    sigma <- theta[3]
    y_fit <- B0 + B1 * x
    logLik_vec <- dnorm(y,
                        mean = y_fit,
                        sd = sigma,
                        log = T)
    sum(logLik_vec)
}
```

The Prior

```
prior <- function(theta) {
    B0 <- theta[1]
    B1 <- theta[2]
    sigma <- theta[3]
    B0_prior <- dnorm(B0, sd = 5, log = T)
    B1_prior <- dunif(B1, min = 0, max = 10, log = T)
    sigma_prior <- dunif(sigma, min = 0, max = 30, log = T)
    B0_prior + B1_prior + sigma_prior
}
```

The Posterior

```
posterior <- function(theta) {
   likelihood(theta) + prior(theta)
}
```

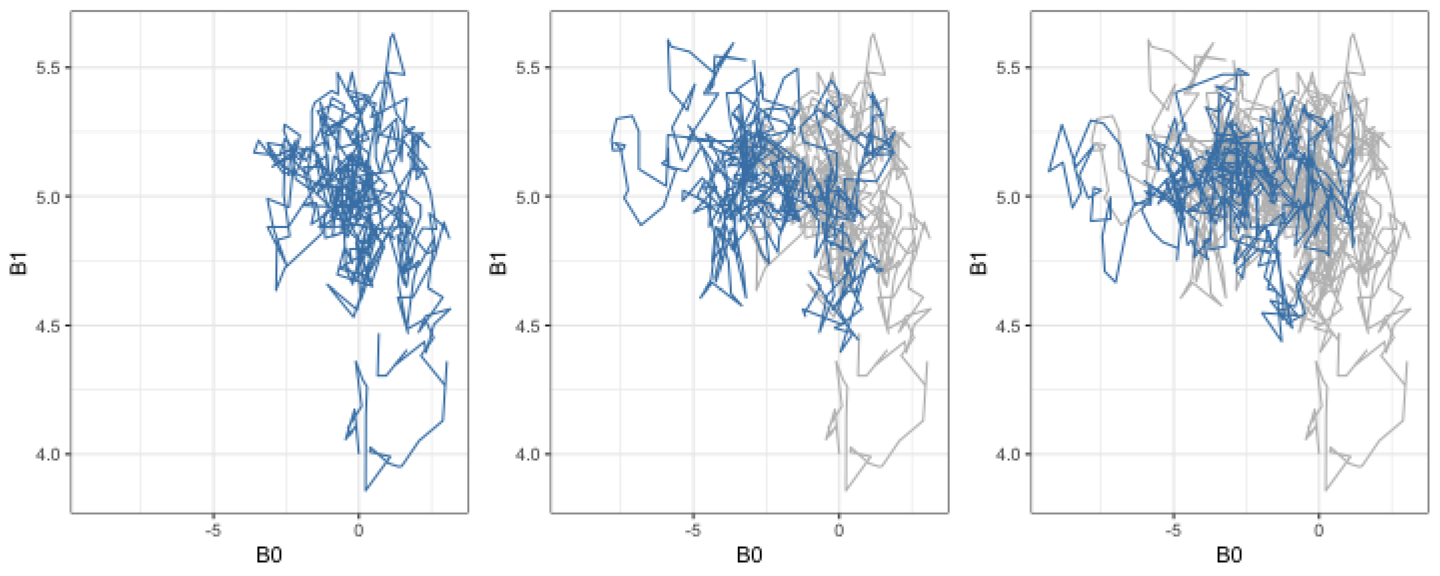
Why are we using logs of everything? Why don't we care about the constant of proportionality?

Metropolis Algorithm

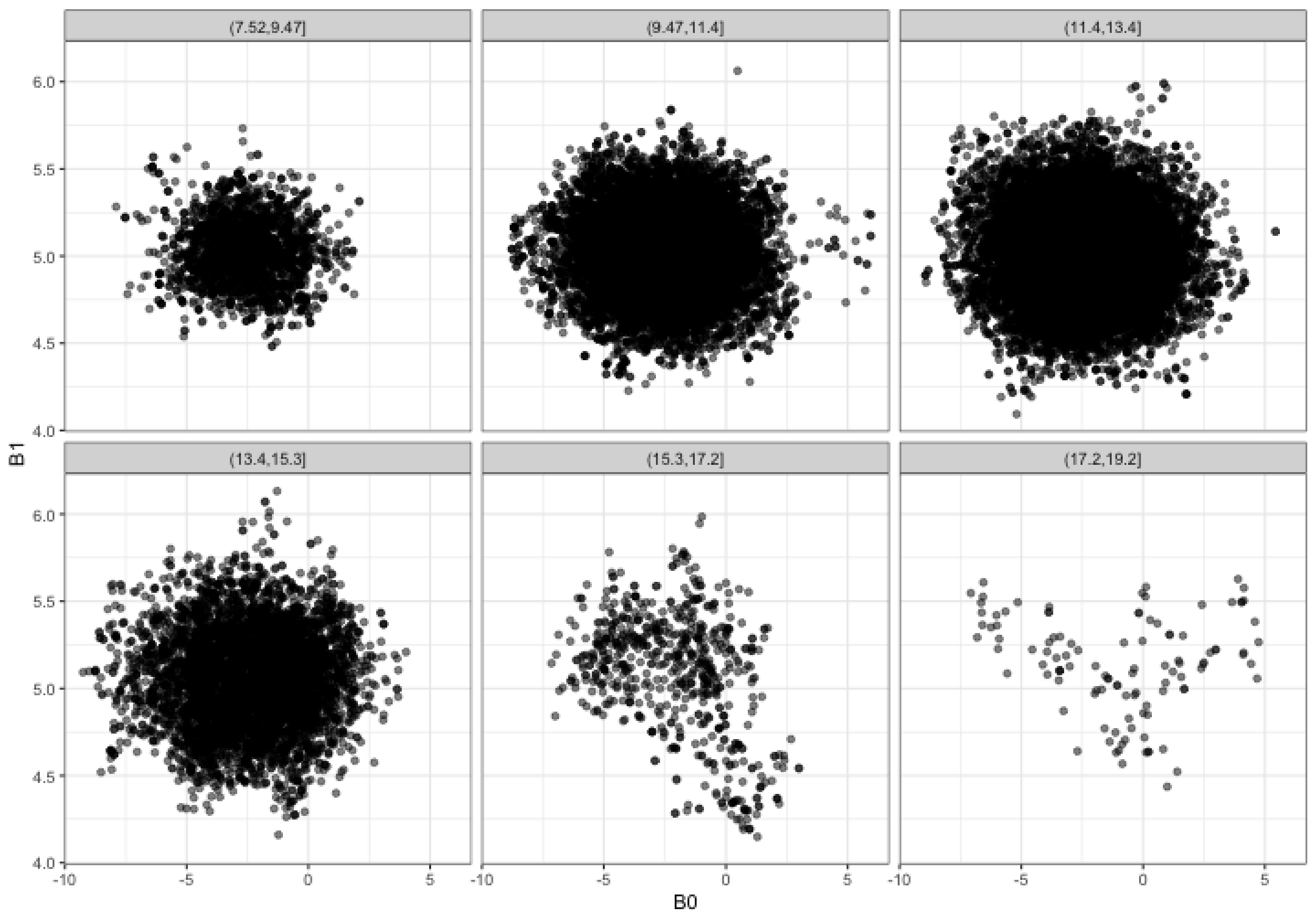
```
it <- 50000
chain <- matrix(rep(NA, (it + 1) * 3), ncol = 3)
theta_0 <- c(0, 4, 10)
chain[1, ] <- theta_0
for (i in 1:it){
  proposal <- rnorm(3, mean = chain[i, ],
                       sd = c(0.5,0.1,0.3))
  p_move <- exp(posterior(proposal) - posterior(chain[i, ]))
  if (runif(1) < p_move) {
    chain[i + 1, ] <- proposal
  } else {
    chain[i + 1, ] <- chain[i, ]
  }
}
head(chain)
```

```
##            [,1]     [,2]      [,3]
## [1,]  0.0000000 4.000000 10.000000
## [2,] -0.1481424 4.173897 10.086451
## [3,] -0.4571575 4.057407  9.837978
## [4,] -0.1109865 4.109123 10.150943
## [5,] -0.2945380 4.075292 10.446455
```

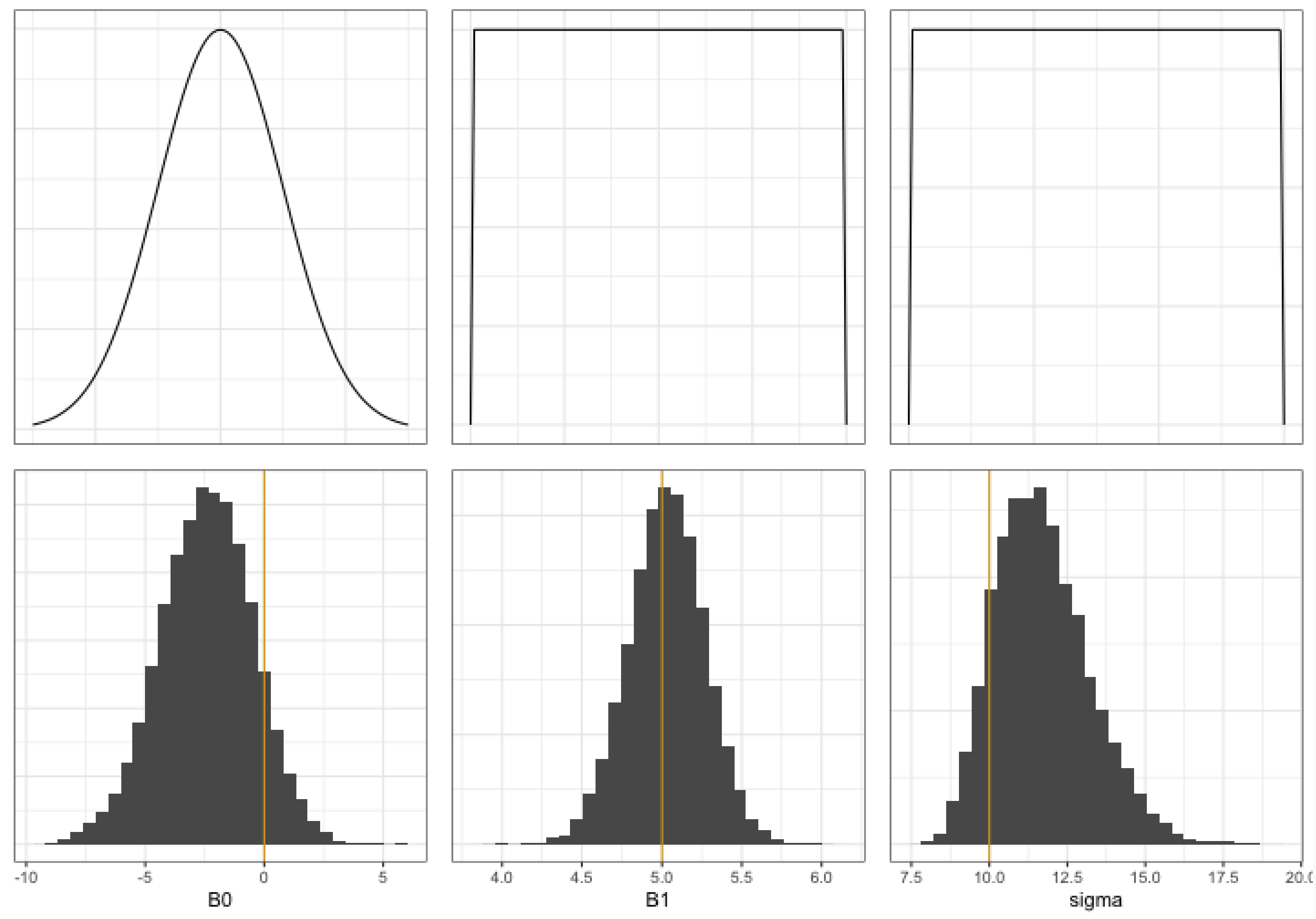
Trace chain



Sigma vs Betas



From Prior to Posterior



Bayesian Point Estimates

There are several options for turning the posterior distribution of the parameters into point estimates of the coefficients. We'll use the mean.

```
(B0_bayes <- mean(chain$B0))
```

```
## [1] -2.423809
```

```
(B1_bayes <- mean(chain$B1))
```

```
## [1] 5.028589
```

```
(sigma_bayes <- mean(chain$sigma))
```

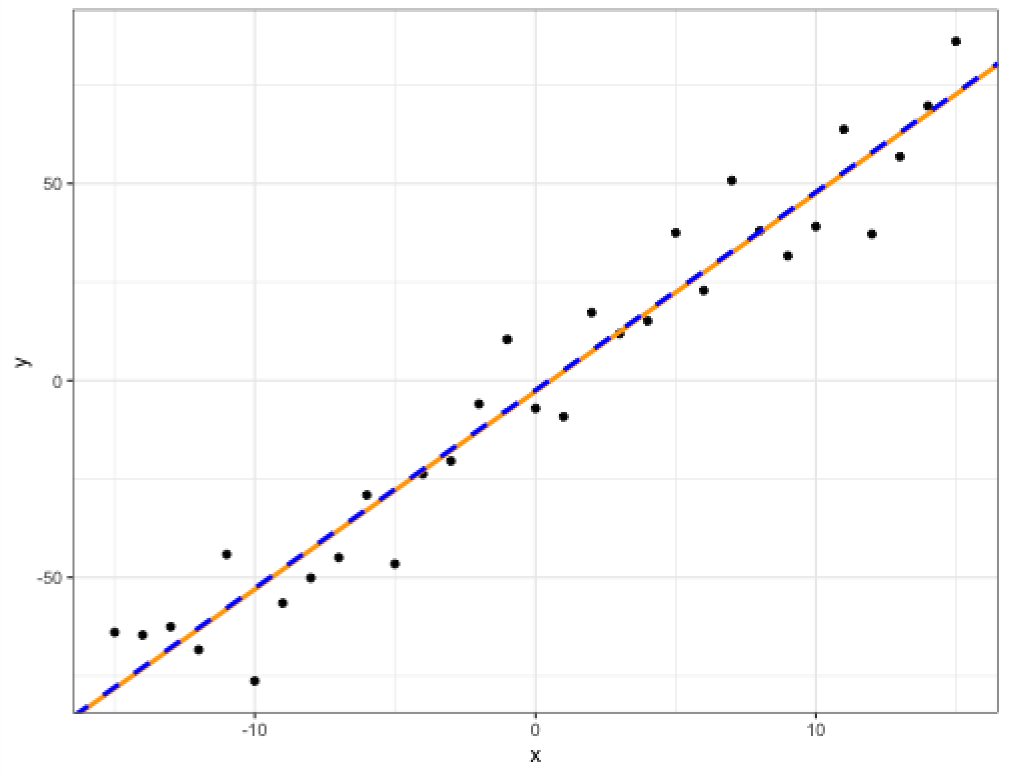
```
## [1] 11.68731
```

We can compare those to the maximum likelihood / least squares estimates.

```r
df <- data.frame(x, y)
m1 <- lm(y ~ x, df)
coef(m1)
```

```
## (Intercept)           x
##   -2.749816    5.028168
```

Two approaches



Intervals on β_1

Confidence Interval

```
confint(m1, parm = 2)
```

```
##       2.5 %   97.5 %
## x 4.559266 5.497069
```

Credible Interval

```
quantile(chain$B1, c(.025, .975))
```

```
##     2.5%    97.5%
## 4.554244 5.491358
```