Assignment 10: Evaluation of the Gibbs Sampling Algorithm Using a Real-Life Example

yifan

2025-04-14

Introduction

In class, we learned another type of MCMC algorithm besides the Metropolis-Hastings algorithm: the Gibbs Sampling algorithm. The algorithm can be illustrated as (assuming the dimension of the random variable we want to sample from is p:

Starting with $(X_1^{(0)}, \cdots, X_p^{(0)})$, iterate for $t = 1, 2, \cdots$

- (1) Draw $X_1 \sim f_{X_1|X_{-1}}(\cdot|X_2^{(t-1)}, \cdots, X_p^{(t-1)})$;
  ...
- (j) Draw $X_j \sim f_{X_j|X_{-j}}(\cdot|X_1^{(t)}, \cdots, X_{j-1}^{(t)}, X_{j+1}^{(t-1)}, \cdots, X_p^{(t-1)})$;
  ...
- (p) Draw $X_p \sim f_{X_p|X_{-p}}(\cdot|X_1^{(t)}, \cdots, X_{p-1}^{(t)})$.

It can be seen that this algorithm overcomes the difficulty of sampling from the joint distribution of the variables. Instead, it samples from the full conditionals. In some scenarios, this is a much more trivial task. In this assignment, I will apply this sampling algorithm to an example illustrated in the testbook, and evaluate the performance of the algorithm.

Scenario Discription

Consider the following Poisson Change Model:

The data we observe is: $\mathbf{Y} = (Y_1, \cdots, Y_n)$. We model this data by assuming:

$$\begin{aligned} Y_i &\sim \text{Poi}(\lambda_1) \quad \text{for} \quad i = 1, \dots, M, \\ Y_i &\sim \text{Poi}(\lambda_2) \quad \text{for} \quad i = M+1, \dots, n. \end{aligned} \tag{1}$$

where M,λ_1 and λ_2 are parameters to estimate.

In real life, this model could represent the monthly count of traffic accidents on a road before and after a safety intervention (e.g., speed cameras installed in month M). Before the intervention, the traffic accidents happen in a higher rate λ_1, while after intervention, the accident rate decreases to λ_2.

Statistical Modeling

We apply a Bayesian approach to estimate M, λ_1 and λ_2. A conjugate prior distribution for λ_j is the $Gamma(\alpha_j, \beta_j)$ distribution with density

$$f(\lambda_j) = \frac{1}{\Gamma(\alpha_j)} \lambda_j^{\alpha_j - 1} \beta_j^{\alpha_j} \exp(-\beta_j \lambda_j) \tag{2}$$

Here, $j = 1, 2$ and α_j, β_j are hyperparameters.

We can easily obtain the joint distribution of $Y_1, \cdots, Y_n, \lambda_1, \lambda_2, M$:

$$\begin{aligned} f(y_1, \dots, y_n, \lambda_1, \lambda_2, M) = &\left(\prod_{i=1}^{M} \frac{\exp(-\lambda_1) \lambda_1^{y_i}}{y_i!} \right) \cdot \left(\prod_{i=M+1}^{n} \frac{\exp(-\lambda_2) \lambda_2^{y_i}}{y_i!} \right) \\ &\cdot \frac{1}{\Gamma(\alpha_1)} \lambda_1^{\alpha_1 - 1} \beta_1^{\alpha_1} \exp(-\beta_1 \lambda_1) \cdot \frac{1}{\Gamma(\alpha_2)} \lambda_2^{\alpha_2 - 1} \beta_2^{\alpha_2} \exp(-\beta_2 \lambda_2) \end{aligned} \tag{3}$$

As well as the full conditions:

$$f(\lambda_1 \mid Y_1, \dots, Y_n, M) \propto \lambda_1^{\alpha_1 - 1 + \sum_{i=1}^{M} y_i} \exp(-(\beta_1 + M)\lambda_1) \tag{4}$$

So that

$$\lambda_1 \mid Y_1, \dots, Y_n, M \sim \text{Gamma} \left(\alpha_1 + \sum_{i=1}^{M} y_i, \beta_1 + M \right) \tag{5}$$

Similar results hold for λ_2.

For the prior distribution of M, since we have no prior information available, we employ a uniform distribution on the set $(1, \cdots, n-1)$as its prior distribution. Extract the terms that are relevant to M in equation (3), and we obtain:

$$p(M) \propto \lambda_1^{\sum_{i=1}^{M} y_i} \cdot \lambda_2^{\sum_{i=M+1}^{n} y_i} \cdot \exp\left((\lambda_2 - \lambda_1) \cdot M\right) \tag{6}$$

Since the posterior distributions of the full conditionals of λ_1 and λ_2 follow Gamma distribution, they are easy to sample. Moreover, since the posterior distribution of the full conditional of M is a finite discrete distribution, it is also easy to sample. However, the joint posterior distribution of $(\lambda_1, \lambda_2, M)$ is difficult to sample. By intuition, using Gibbs algorithm to obtain samples from the posterior distribution seems to be a wise choice.

Parameter Selection

Let's consider the traffic accident example mentioned earlier. According to the data published by World Health Organization (WHO), the monthly accident count on urban arterial roads typically ranges between 3 to 8 accidents per kilometer per month (depending on traffic volume and road design). After the installation of speed cameras, effective interventions usually reduce the accident rate by 20-40%. Now we assume that the data we obtained come from a high-risk road segment. Since the expectation of the Poisson distribution is λ, we choose the ground truth λ_1 and λ_2 to be 8 and 5, respectively. Assume that $n = 60$ and $M = 37$, which indicates approximately 3 years of data before the intervention and 2 years of data after the intervention.

Now, we have to estimate the hyperparameters α_j, β_j. Since we don't actually know that we collected data from a dangerous road, we assume that $\mu_1 = 5, \mu_2 = 3$, and $\sigma_j = \mu_j/2$. Since we have $\mu_j = \alpha_j/\beta_j$ and $\sigma_j^2 = \alpha_j/\beta_j^2$, we could choose $\alpha_j = \mu_j^2/\sigma_j^2$ and $\beta_j = \mu_j/\sigma_j^2$. In this way, we obtain: $\alpha_1 = \alpha_2 = 4, \beta_1 = 4/5, \beta_2 = 4/3$.

R Implemetation

First, we have to generate the observed data $\mathbf{Y}$ using our model:

```r
# Set parameters
n <- 60       # Total number of observations
M <- 37       # Change-point location
lambda1 <- 8  # Poisson rate before change-point
lambda2 <- 5  # Poisson rate after change-point

# Generate data
set.seed(123) # Set random seed for reproducibility
Y <- c(rpois(M, lambda1), rpois(n - M, lambda2))

# Print generated data
print(Y)
```

```
##  [1]  6 10  7 11 13  4  8 12  8  8 13  8  9  8  5 12  6  3  7 13 12  9  9 16  9
## [26]  9  8  9  6  5 13 12  9 10  3  8 10  3  4  3  3  4  4  4  3  3  3  5  4  7
## [51]  2  5  7  2  5  3  3  6  8  4
```

First, we create 3 R functions that returns a sample from the posterior full-condition distribution of λ_1, λ_2 and M:

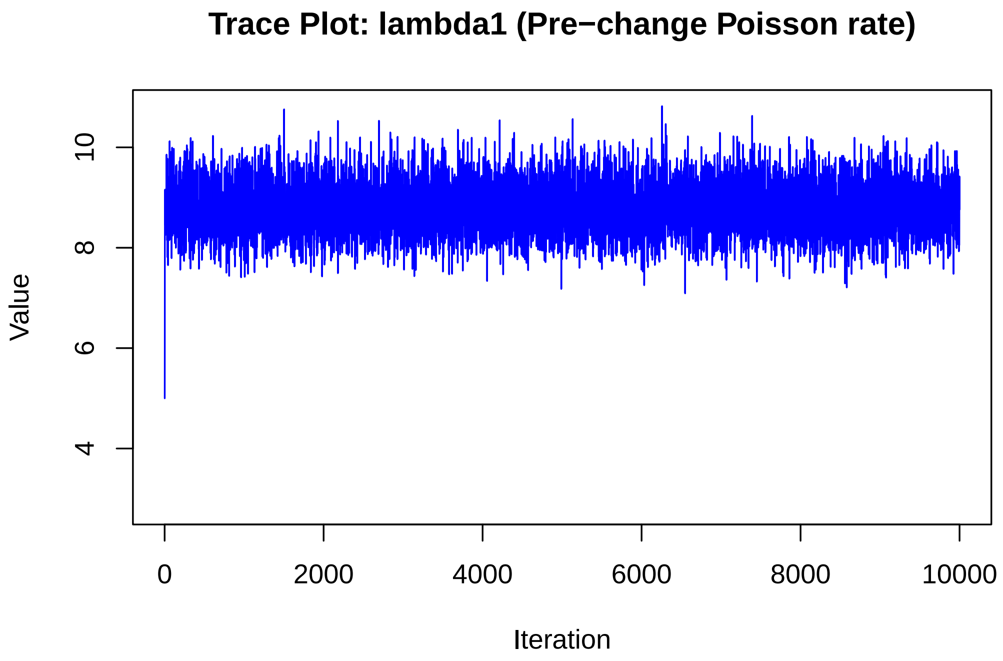
```r
lambda1_posterior <- function(alpha1, beta1, M, Y) {
  # Check input validity
  if (length(Y) < M) {
    stop("Vector Y must have at least M elements")
  }
  if (M <= 0) {
    stop("M must be a positive integer")
  }

  # Calculate posterior parameters
  posterior_alpha <- alpha1 + sum(Y[1:M])
  posterior_beta <- beta1 + M

  # Draw one sample from the posterior Gamma distribution
  rgamma(n = 1, shape = posterior_alpha, rate = posterior_beta)
}

lambda2_posterior <- function(alpha2, beta2, M, Y) {
  # Check input validity
  if (length(Y) < M) {
    stop("Vector Y must have at least M elements")
  }
  if (M <= 0) {
    stop("M must be a positive integer")
  }

  # Calculate posterior parameters
  posterior_alpha <- alpha2 + sum(Y[(M+1):length(Y)])
  posterior_beta <- beta2 + length(Y) - M

  # Draw one sample from the posterior Gamma distribution
  rgamma(n = 1, shape = posterior_alpha, rate = posterior_beta)
}

M_posterior <- function(lambda1, lambda2, Y) {
  n <- length(Y)
  if (n < 2) stop("Data vector Y must have length >= 2")

  # Calculate unnormalized log-probabilities for all possible M
  log_probs <- vapply(1:(n-1), function(M) {
    sum_y1 <- sum(Y[1:M])
    sum_y2 <- sum(Y[(M+1):n])
    sum_y1 * log(lambda1) + sum_y2 * log(lambda2) + (lambda2 - lambda1) * M
  }, numeric(1))

  # Convert to probabilities (with numerical stability)
  probs <- exp(log_probs - max(log_probs))
  probs <- probs / sum(probs)

  # Sample one M value according to these probabilities
  sample(1:(n-1), size = 1, prob = probs)
}
```

Now, we could finally run the Gibbs sampling algorithm. Note that the initial value for λ_1 and λ_2 are chosen as their piors: 5 and 3 respectively. The initial value for M is set to be 1.

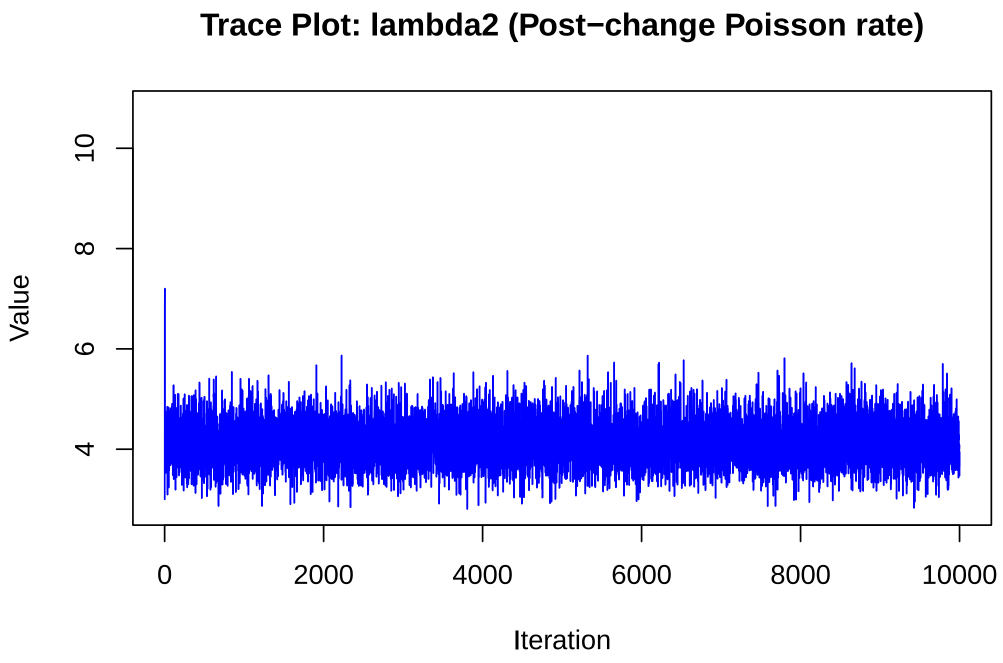
```r
#Set parameters
#Hyperparameters
alpha1 <- 4
alpha2 <- 4
beta1 <- 4/5
beta2 <- 4/3

# Number of iterations
N <- 10000

lambda1_samples <- numeric(N)
lambda2_samples <- numeric(N)
M_samples <- numeric(N)

#Initialization
lambda1_samples[1] <- 5
lambda2_samples[1] <- 3
M_samples[1] <- 1

#Run algorithm
for(i in 2:N){
  lambda1_samples[i] <- lambda1_posterior(alpha1,beta1,M_samples[i-1],Y)
  lambda2_samples[i] <- lambda2_posterior(alpha2,beta2,M_samples[i-1],Y)
  M_samples[i] <- M_posterior(lambda1_samples[i],lambda2_samples[i],Y)
}
```

Next, we plot out the traces of λ_1, λ_2 as well as M:

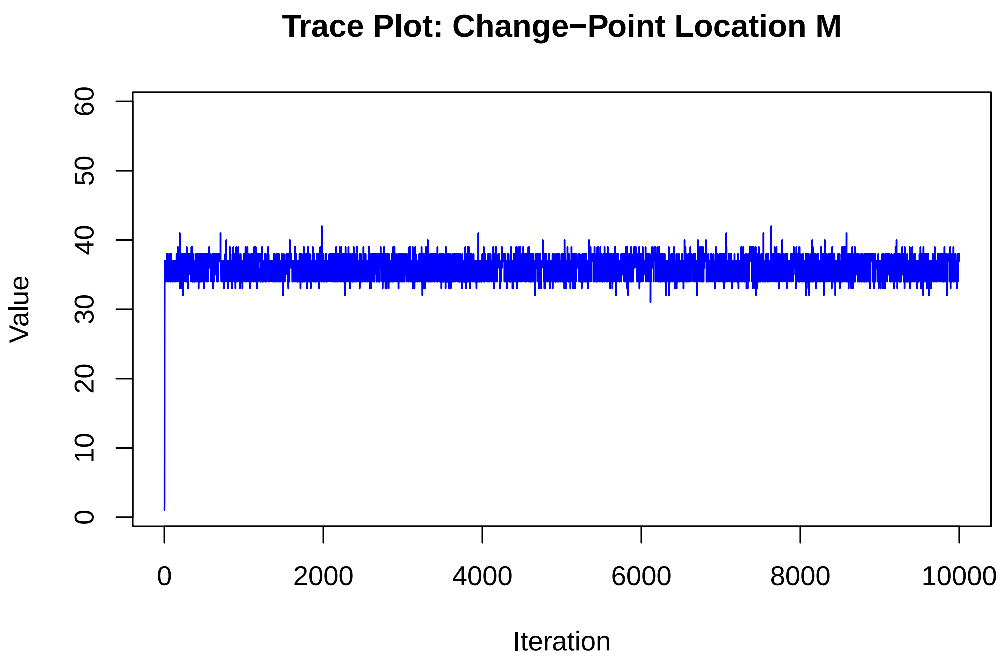
```r
plot(lambda1_samples, type = "l", col = "blue",
     main = "Trace Plot: lambda1 (Pre-change Poisson rate)",
     xlab = "Iteration", ylab = "Value",
     ylim = range(c(lambda1_samples, lambda2_samples)))
```



```r
plot(lambda2_samples, type = "l", col = "blue",
     main = "Trace Plot: lambda2 (Post-change Poisson rate)",
     xlab = "Iteration", ylab = "Value",
     ylim = range(c(lambda1_samples, lambda2_samples)))
```



```r
plot(M_samples, type = "l", col = "blue",
     main = "Trace Plot: Change-Point Location M",
     xlab = "Iteration",
     ylab = "Value",
     ylim = c(1, length(Y)-1))
```



The shapes of those plots are similar to white noises, which justify the performance of the Gibbs sampling algorithm.

After discarding the first 1000 samples, we can find out the posterior mean for λ_1 and λ_2, as well as the posterior mode for M:

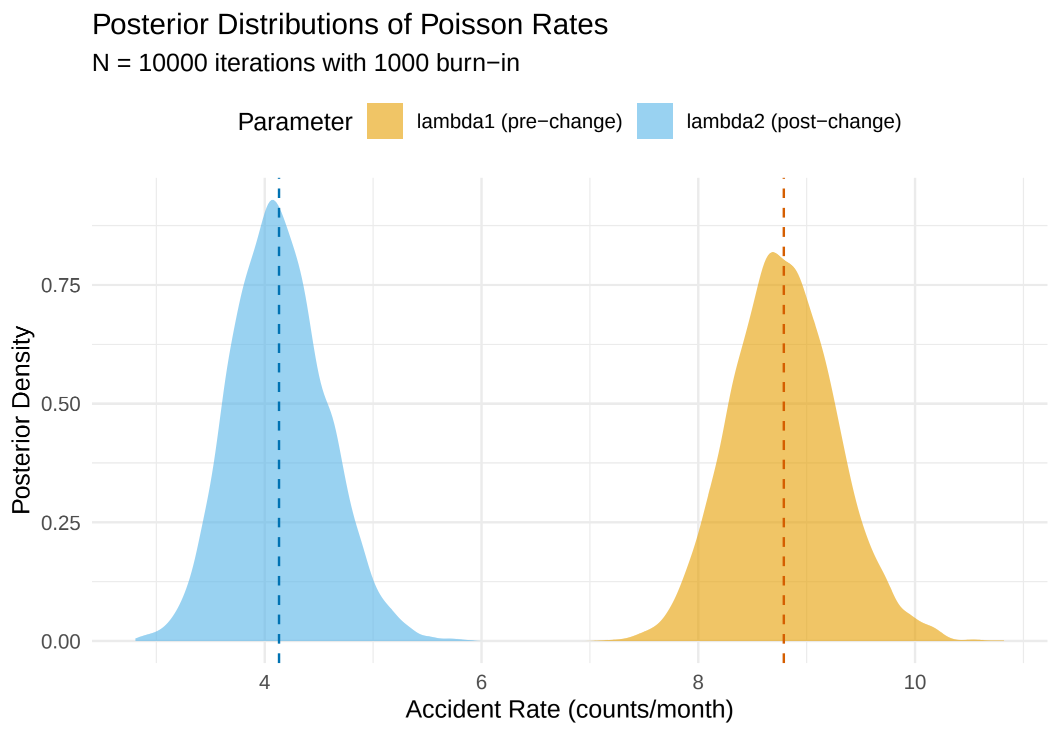
```r
#Function for finding out the mode
get_mode <- function(v) {
  uniq_v <- unique(v)
  uniq_v[which.max(tabulate(match(v, uniq_v)))]
}
# Calculate posterior estimates after burn-in (first 1000 iterations)
post_lambda1 <- mean(lambda1_samples[1001:N])
post_lambda2 <- mean(lambda2_samples[1001:N])
post_mode_M <- get_mode(M_samples[1001:N])

# Print results with context
cat(sprintf(
  "After %d iterations (burn-in=1000):\n- Pre-change rate ( ):
  %.3f accidents/month\n- Post-change rate ( ):
  %.3f accidents/month\n- Most probable change-point: Month %d",
  N, post_lambda1, post_lambda2, post_mode_M
))
```

```
## After 10000 iterations (burn-in=1000):
```

```
## - Pre-change rate ( ):
##   8.789 accidents/month
## - Post-change rate ( ):
##   4.132 accidents/month
## - Most probable change-point: Month 37
```

Moreover, we could fit the posterior distributions of λ_1 and $lambda_2$ using kernel density estimation (more specifically, we utilize Guassian kernel and select bandwidth using Silverman's rule of thumb).

```r
# Create density data frames after burn-in
post_df <- data.frame(
  parameter = rep(c("lambda1 (pre-change)", "lambda2 (post-change)"),
                  each = length(lambda1_samples[1001:N])),
  value = c(lambda1_samples[1001:N], lambda2_samples[1001:N])
)

# Create the plot
ggplot(post_df, aes(x = value, fill = parameter)) +
  geom_density(alpha = 0.6, color = NA) +
  scale_fill_manual(values = c("#E69F00", "#56B4E9")) +
  labs(title = "Posterior Distributions of Poisson Rates",
       subtitle = paste("N =", N, "iterations with 1000 burn-in"),
       x = "Accident Rate (counts/month)",
       y = "Posterior Density",
       fill = "Parameter") +
  theme_minimal() +
  theme(legend.position = "top") +
  geom_vline(data = aggregate(value ~ parameter, post_df, mean),
             aes(xintercept = value, color = parameter),
             linetype = "dashed", show.legend = FALSE) +
  scale_color_manual(values = c("#D55E00", "#0072B2"))
```



These two densities hardly overlap, suggesting significant effect of the intervention.

The posterior distribution of M is easy to obtain since it is discrete (we only display the top-10 values with the highest probabilities:

```
#Extract samples
post_M_samples <- M_samples[1001:N]

# Calculate posterior distribution
posterior_M <- prop.table(table(post_M_samples))

# Convert to dataframe
posterior_M_df <- data.frame(
  M = as.numeric(names(posterior_M)),
  Probability = as.numeric(posterior_M),
  stringsAsFactors = FALSE
)

# Reorder according to probability
posterior_M_df <- posterior_M_df[order(-posterior_M_df$Probability), ]

# Display top-10 values
head(posterior_M_df, 10)
```

```
##     M  Probability
## 7  37 0.7156666667
## 4  34 0.1307777778
## 8  38 0.0715555556
## 6  36 0.0394444444
## 9  39 0.0145555556
## 5  35 0.0130000000
## 3  33 0.0110000000
## 2  32 0.0018888889
## 10 40 0.0013333333
## 11 41 0.0004444444
```

We can see that most of the probability concentrate at the point $M = 37$.

Finally, we could take a step further, and calculate the posterior mean, median, as well as 95% CIs of λ_1 and λ_2 to end this section.

```
cat("Posterior Summaries:\n")
```

```
## Posterior Summaries:
```

```
tapply(post_df$value, post_df$parameter, function(x) {
  c(mean = mean(x),
    median = median(x),
    CI_95 = quantile(x, c(0.025, 0.975)))
})
```

```
## $`lambda1 (pre-change)`
##        mean      median  CI_95.2.5% CI_95.97.5%
##    8.789244    8.777089    7.878800    9.773814
##
## $`lambda2 (post-change)`
##        mean      median  CI_95.2.5% CI_95.97.5%
##    4.131873    4.112360    3.346668    5.024956
```

Discussion

In our previous analysis, we obtained the following posterior estimates:

- $\hat{\lambda_1} = 8.79$ (pre-change rate)
- $\hat{\lambda_2} = 4.13$ (post-change rate)
- $\hat{M} = 37$ (change-point location)

The estimation for the change-point M demonstrates remarkable accuracy, aligning precisely with the simulated ground truth. However, we observe non-negligible biases in the rate parameters:

- $bias(\hat{\lambda_1}) = 0.79$ (9.9% overestimation)
- $bias(\hat{\lambda_2}) = -0.87$ (17.4% underestimation)

A natural question arises whether these biases stem from inherent limitations of the Gibbs sampling algorithm. In this section, we demonstrate that the observed biases do not originate from MCMC approximation errors. Rather, they are primarily attributable to finite sample effects in the observation window.

We note that the average for the data before the intervention and the datas after the intervention are 8.86, 4.13 respectively (which are similar to our aforementioned estimations!)

```r
mean(Y[1:37])
```

```
## [1] 8.864865
```

```r
mean(Y[38:60])
```

```
## [1] 4.130435
```

So that even if we know the ground truth of M is in fact $M = 37$, and we obtain infinite samples from the posterior distribution of λ_1 and λ_2, the posterior means we would get would be:

$$\begin{cases} \tilde{\lambda_1} = \dfrac{\alpha_1 + \sum_{i=1}^{3} 7y_i}{\beta_1 + 37} \\ \tilde{\lambda_2} = \dfrac{\alpha_2 + \sum_{i=38}^{n} y_i}{\beta_2 + n - 37} \end{cases} \tag{7}$$

Which are $\tilde{\lambda_1} = 8.78, \tilde{\lambda_2} = 4.07$.

```r
lambda1_tilde <- (alpha1 + sum(Y[1:37])) / (beta1 + M)
lambda2_tilde <- (alpha2 + sum(Y[38:60])) / (beta2 + n - M)
lambda1_tilde
```

```
## [1] 8.783069
```

```r
lambda2_tilde
```

```
## [1] 4.068493
```

Therefore, we have $bias(\tilde{\lambda_1}) = 0.78$ and $bias(\tilde{\lambda_2}) = 0.93$, which isn't better than our original estimation. The truth is, one cannot achieve a better estimation with only 60 samples. We can imagine that if we are able to obtain a great number of samples, our estimation using the Gibbs sampling algorithm will eventually converge to the ground truth.