Modelling the Duration of Wait Times between Old Faithful Geyser's Eruptions: a Bayesian Bimodal Normal Mixture Model approximated by Monte-Carlo Markov Chain

Edbert Jao

2023-12-13

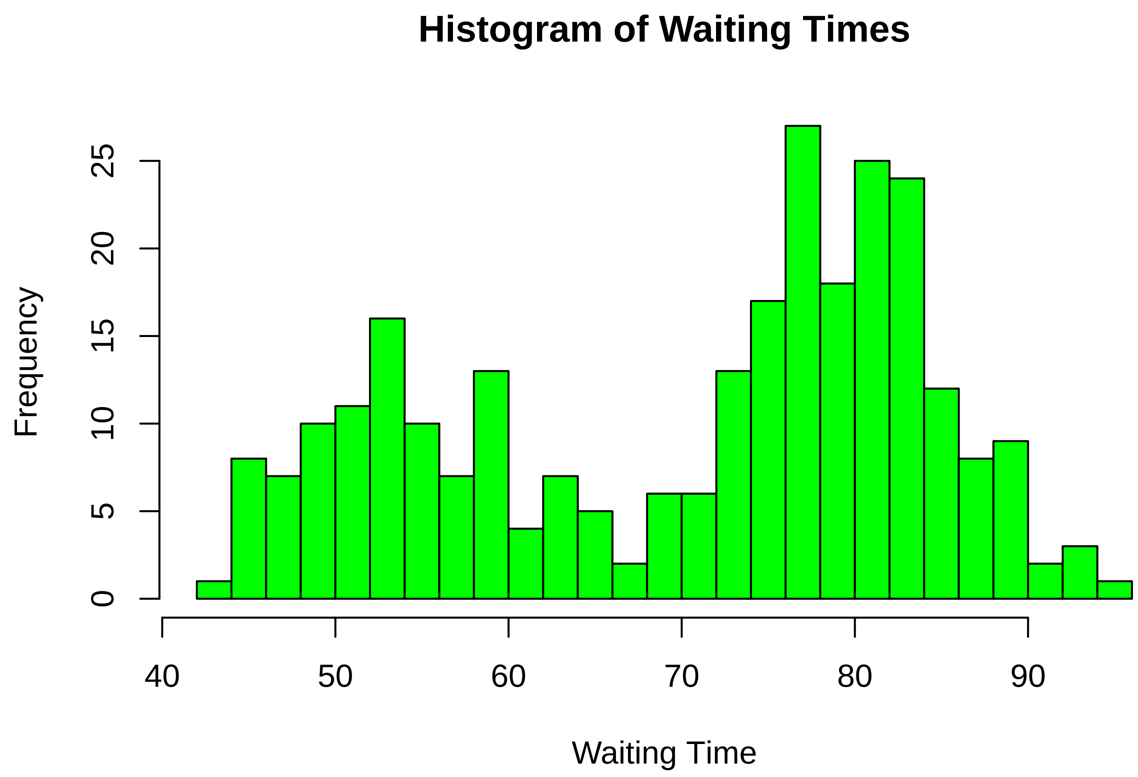
In this project, I seek to model the duration of wait times between Old Faithful Geyser's Eruptions. Data on wait times between these eruptions are available from R directly.

Exploratory Data Analysis:

```
data(faithful);
summary(faithful);
```

```
##    eruptions        waiting
##  Min.   :1.600   Min.   :43.0
##  1st Qu.:2.163   1st Qu.:58.0
##  Median :4.000   Median :76.0
##  Mean   :3.488   Mean   :70.9
##  3rd Qu.:4.454   3rd Qu.:82.0
##  Max.   :5.100   Max.   :96.0
```

```
hist(faithful$waiting, breaks = 20,
     main="Histogram of Waiting Times",
     xlab="Waiting Time", ylab="Frequency", col="green");
```



Proposed Model and Numerical Approximation Method to Fit Model Parameters

The wait times appear to follow a bimodal normal mixture distribution. I propose the following model:

$$X|\{p, \mu_1, \mu_2, \sigma_1, \sigma_2\} = \begin{cases} \mathbb{N}(\mu_1, \sigma_1^2) \text{ with probability } p \\ \mathbb{N}(\mu_2, \sigma_2^2) \text{ with probability } 1-p \end{cases}$$

with prior distributions:

- $p \sim Unif(0, 1)$
- $\mu_1, \mu_2 \sim Cauchy(0, 1)$
- $\sigma_1, \sigma_2 \sim Gamma(0.1, 100)$

Jointly, the parameters are denoted as: $\theta = (p, \mu_1, \mu_2, \sigma_1, \sigma_2)$

The model presented is a bimodal Normal Mixture model because it contains 2 distinct normal distributions. Each normal distribution's mean parameter is independently sampled from a prior $Cauchy(0, 1)$ distribution and each normal distribution's variance parameter is sampled from a prior $Gamma(0.1, 100)$ distribution. The probability p of drawing a sample from one of the normal distributions as opposed to the other is sampled from a prior $Uniform(0, 1)$ distribution. Given that each normal curve's standard deviation appears similar, it is appropriate to sample standard deviation from the same prior $Gamma(0.1, 100)$ distribution. Independent draws of mean parameters from the same prior $Cauchy(0, 1)$ distribution will most likely result in distinct means that the two normal distributions will respectively center on.

There is one particular weakness for this model. Draws from the Cauchy distribution can be negative. It is impossible for a mean waiting time to be negative so it may be problematic to sample the 2 mean parameters, which must be non-negative, from the Cauchy distribution.

To fit my model with the data, I must derive the posterior probabilities of my parameters. I denote the observed waiting times of "faithful" as $X = (X_1, ..., X_n)$. I assume that $X_1, ..., X_n$ are independent and identically distributed.

To fit my model, I need to solve for the parameters of the posterior distribution $(p, \mu_1, \mu_2, \sigma_1, \sigma_2)|X$. It is not feasible to analytically derive the posterior distribution $(p, \mu_1, \mu_2, \sigma_1, \sigma_2)|X$. Then, the parameters of the posterior distribution must be approximated. I choose to use Gibbs Sampling, a Markov Chain Monte Carlo method, because it can approximate all elements of a chain based on their conditional distributions. The chain, in this context, would be $(p, \mu_1, \mu_2, \sigma_1, \sigma_2)$.

However, the marginal distributions of elements of the chain $(p, \mu_1, \mu_2, \sigma_1, \sigma_2)$ conditional on all other elements do not have closed-form solutions. (i.e. the integrals of the probability density functions do not have closed-form solutions.) Thus, it is necessary to numerically approximate samples from these distributions.

To do so, I choose to use a Metropolis-Hastings algorithm with a symmetric random walk proposal distribution.

Thus, the full model is fitted with a Metropolis-Hastings within Gibbs Sampling approach.

Derivations of conditional posterior disributions

By Bayes' Theorem, the conditional posterior distribution of p, derived up to a proportionality factor, is as follows:

- $$\begin{aligned}\pi(p|\mu_1, \mu_2, \sigma_1, \sigma_2, X) = \frac{\pi(p, \mu_1, \mu_2, \sigma_1, \sigma_2, X)}{\pi(\mu_1, \mu_2, \sigma_1, \sigma_2, X)} &= \frac{\pi(X|p, \mu_1, \mu_2, \sigma_1, \sigma_2)\pi(\mu_1)\pi(\mu_2)\pi(\sigma_1)\pi(\sigma_2)\pi(p)}{\pi(X|\mu_1, \mu_2, \sigma_1, \sigma_2)\pi(\mu_1)\pi(\mu_2)\pi(\sigma_1)\pi(\sigma_2)} \\ &= \frac{\pi(X|p, \mu_1, \mu_2, \sigma_1, \sigma_2)\pi(p)}{\pi(X|\mu_1, \mu_2, \sigma_1, \sigma_2)} \propto \pi(X|p, \mu_1, \mu_2, \sigma_1, \sigma_2)\pi(p)\end{aligned}$$

By similar conditioning, the conditional posterior distributions for $\mu_1, \mu_2, \sigma_1, \sigma_2$, can also be derived up to proportionality factors. They are as follows:

- $\pi(\mu_1|p_1, \mu_2, \sigma_1, \sigma_2, X) \propto \pi(X|p, \mu_1, \mu_2, \sigma_1, \sigma_2)\pi(\mu_1)$
- $\pi(\mu_2|p_1, \mu_1, \sigma_1, \sigma_2, X) \propto \pi(X|p, \mu_1, \mu_2, \sigma_1, \sigma_2)\pi(\mu_2)$
- $\pi(\sigma_1|p, \mu_1, \mu_2, \sigma_2, X) \propto \pi(X|p, \mu_1, \mu_2, \sigma_1, \sigma_2)\pi(\sigma_1)$
- $\pi(\sigma_2|p, \mu_1, \mu_2, \sigma_1, X) \propto \pi(X|p, \mu_1, \mu_2, \sigma_1, \sigma_2)\pi(\sigma_2)$

Expanding the conditionalpPosterior pdstributions, we derive:

- $\pi(p|\mu_1, \mu_2, \sigma_1, \sigma_2, X_i) \propto \pi(X_i|p, \mu_1, \mu_2, \sigma_1, \sigma_2)\pi(p) = p\frac{1}{\sigma_1\sqrt{2\pi}}e^{-\frac{(X_i-\mu_1)^2}{2\sigma_1^2}} + (1-p)\frac{1}{\sigma_2\sqrt{2\pi}}e^{-\frac{(X_i-\mu_2)^2}{2\sigma_2^2}}$
- $\pi(\mu_1|p, \mu_2, \sigma_1, \sigma_2, X_i) \propto \pi(X_i|p, \mu_1, \mu_2, \sigma_1, \sigma_2)\pi(\mu_1) = \left(p\frac{1}{\sigma_1\sqrt{2\pi}}e^{-\frac{(X_i-\mu_1)^2}{2\sigma_1^2}} + (1-p)\frac{1}{\sigma_2\sqrt{2\pi}}e^{-\frac{(X_i-\mu_2)^2}{2\sigma_2^2}}\right)\frac{1}{\pi+\pi\mu_1^2}$
- $\pi(\mu_2|p, \mu_1, \sigma_1, \sigma_2, X_i) \propto \pi(X_i|p, \mu_1, \mu_2, \sigma_1, \sigma_2)\pi(\mu_2) = \left(p\frac{1}{\sigma_1\sqrt{2\pi}}e^{-\frac{(X_i-\mu_1)^2}{2\sigma_1^2}} + (1-p)\frac{1}{\sigma_2\sqrt{2\pi}}e^{-\frac{(X_i-\mu_2)^2}{2\sigma_2^2}}\right)\frac{1}{\pi+\pi\mu_2^2}$
- $\pi(\sigma_1|p, \mu_1, \mu_2, \sigma_2, X_i) \propto \pi(X_i|p, \mu_1, \mu_2, \sigma_1, \sigma_2)\pi(\sigma_1)$
$= \left(p\frac{1}{\sigma_1\sqrt{2\pi}}e^{-\frac{(X_i-\mu_1)^2}{2\sigma_1^2}} + (1-p)\frac{1}{\sigma_2\sqrt{2\pi}}e^{-\frac{(X_i-\mu_2)^2}{2\sigma_2^2}}\right)\frac{1}{\Gamma(0.1)100^{0.1}}\sigma_1^{-0.9}e^{-\frac{\sigma_1}{100}}$

$$\propto \left(p\frac{1}{\sigma_1\sqrt{2\pi}}e^{-\frac{(X_i-\mu_1)^2}{2\sigma_1^2}} + (1-p)\frac{1}{\sigma_2\sqrt{2\pi}}e^{-\frac{(X_i-\mu_2)^2}{2\sigma_2^2}}\right)\sigma_1^{-0.9}e^{-\frac{\sigma_1}{100}}$$

- $\pi(\sigma_2|p,\mu_1,\mu_2,\sigma_1,X_i) \propto \pi(X_i|p,\mu_1,\mu_2,\sigma_1,\sigma_2)\pi(\sigma_2)$

$$= \left(p\frac{1}{\sigma_1\sqrt{2\pi}}e^{-\frac{(X_i-\mu_1)^2}{2\sigma_1^2}} + (1-p)\frac{1}{\sigma_2\sqrt{2\pi}}e^{-\frac{(X_i-\mu_2)^2}{2\sigma_2^2}}\right)\frac{1}{\Gamma(0.1)100^{0.1}}\sigma_2^{-0.9}e^{-\frac{\sigma_2}{100}}$$

$$\propto \left(p\frac{1}{\sigma_1\sqrt{2\pi}}e^{-\frac{(X_i-\mu_1)^2}{2\sigma_1^2}} + (1-p)\frac{1}{\sigma_2\sqrt{2\pi}}e^{-\frac{(X_i-\mu_2)^2}{2\sigma_2^2}}\right)\sigma_2^{-0.9}e^{-\frac{\sigma_2}{100}}$$

The conditional posterior joint distributions, given $X=(X_1,...,X_n)$ as data, are thus as follows:

- $\pi(p|\mu_1,\mu_2,\sigma_1,\sigma_2,X) \propto \prod_{i=1}^{n}\left(p\frac{1}{\sigma_1\sqrt{2\pi}}e^{-\frac{(X_i-\mu_1)^2}{2\sigma_1^2}} + (1-p)\frac{1}{\sigma_2\sqrt{2\pi}}e^{-\frac{(X_i-\mu_2)^2}{2\sigma_2^2}}\right)$
- $\pi(\mu_1|p,\mu_2,\sigma_1,\sigma_2,X) \propto \prod_{i=1}^{n}\left(\left[p\frac{1}{\sigma_1\sqrt{2\pi}}e^{-\frac{(X_i-\mu_1)^2}{2\sigma_1^2}} + (1-p)\frac{1}{\sigma_2\sqrt{2\pi}}e^{-\frac{(X_i-\mu_2)^2}{2\sigma_2^2}}\right]\frac{1}{\pi+\pi\mu_1^2}\right)$
- $\pi(\mu_2|p,\mu_2,\sigma_1,\sigma_2,X) \propto \prod_{i=1}^{n}\left(\left[p\frac{1}{\sigma_1\sqrt{2\pi}}e^{-\frac{(X_i-\mu_1)^2}{2\sigma_1^2}} + (1-p)\frac{1}{\sigma_2\sqrt{2\pi}}e^{-\frac{(X_i-\mu_2)^2}{2\sigma_2^2}}\right]\frac{1}{\pi+\pi\mu_2^2}\right)$
- $\pi(\sigma_1|p,\mu_1,\mu_2,\sigma_2,X) \propto \prod_{i=1}^{n}\left(\left[p\frac{1}{\sigma_1\sqrt{2\pi}}e^{-\frac{(X_i-\mu_1)^2}{2\sigma_1^2}} + (1-p)\frac{1}{\sigma_2\sqrt{2\pi}}e^{-\frac{(X_i-\mu_2)^2}{2\sigma_2^2}}\right]\sigma_1^{-0.9}e^{-\frac{\sigma_1}{100}}\right)$
- $\pi(\sigma_2|p,\mu_1,\mu_2,\sigma_2,X) \propto \prod_{i=1}^{n}\left(\left[p\frac{1}{\sigma_1\sqrt{2\pi}}e^{-\frac{(X_i-\mu_1)^2}{2\sigma_1^2}} + (1-p)\frac{1}{\sigma_2\sqrt{2\pi}}e^{-\frac{(X_i-\mu_2)^2}{2\sigma_2^2}}\right]\sigma_2^{-0.9}e^{-\frac{\sigma_2}{100}}\right)$

Algorithm of Numerical Approximation to Fit Model

My Gibbs Sampling algorithm follows:

- Set number of iterations $T > 0$.
- Set initial state $(p^{(0)}, \mu_1^{(0)}, \mu_2^{(0)}, \sigma_1^{(0)}, \sigma_2^{(0)})$ and set $t = 1$.
- While $t < T$,
  - Generate $p^{(t)} \sim \pi(p \mid \mu_1^{(t-1)}, \mu_2^{(t-1)}, \sigma_1^{(t-1)}, \sigma_2^{(t-1)}, X)$ via Metropolis-Hastings algorithm.
  - Generate $\mu_1^{(t)} \sim \pi(\mu_1 \mid p^{(t)}, \mu_2^{(t-1)}, \sigma_1^{(t-1)}, \sigma_2^{(t-1)}, X)$ via Metropolis-Hastings algorithm.
  - Generate $\mu_2^{(t)} \sim \pi(\mu_2 \mid p^{(t)}, \mu_1^{(t-1)}, \sigma_1^{(t-1)}, \sigma_2^{(t-1)}, X)$ via Metropolis-Hastings algorithm.
  - Generate $\sigma_1^{(t)} \sim \pi(\sigma_1 \mid p^{(t)}, \mu_1^{(t-1)}, \mu_2^{(t-1)}, \sigma_2^{(t-1)}, X)$ via Metropolis-Hastings algorithm.
  - Generate $\sigma_2^{(t)} \sim \pi(\sigma_2 \mid p^{(t)}, \mu_1^{(t-1)}, \mu_2^{(t-1)}, \sigma_1^{(t-1)}, X)$ via Metropolis-Hastings algorithm.
  - Save $\left(p^{(t)}, \mu_1^{(t)}, \mu_2^{(t)}, \sigma_1^{(t)}, \sigma_2^{(t)}\right)$.
  - $t = t + 1$
- Output: $\left(p^{(1)}, \mu_1^{(1)}, \mu_2^{(1)}, \sigma_1^{(1)}, \sigma_2^{(1)}\right) \cdots \left(p^{(T)}, \mu_1^{(T)}, \mu_2^{(T)}, \sigma_1^{(T)}, \sigma_2^{(T)}\right)$

My Metropolis-Hastings algorithm uses a symmetric random walk proposal distribution:

- For parameter θ_i of $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)$ and at time t,
  - Generate random walk perturbation $\epsilon \sim Unif(-\delta, \delta)$, where $\delta > 0$ is a globally defined tuning parameter.
  - Set $\alpha = \theta_i^{(t-1)} + \epsilon$ as proposal value.

- Calculate acceptance probability $p = min\left(1, \frac{\pi_{\theta_i}\left(\alpha|\theta_{-i}^{(t-1)}, X\right)}{\pi_{\theta_i}\left(\theta_i^{(t-1)}|\theta_{-i}^{(t-1)}, X\right)}\right)$, where π_{θ_i} denotes the joint posterior distribution of θ_i conditional on data $X = (X_1, ..., X_n)$.
- Generate $u \sim Unif(0, 1)$.
    - If $u \leq p$, set $\theta_i^{(t)} = \alpha$
    - Otherwise, set $\theta_i^{(t)} = \theta_i^{(t-1)}$
- Output: $\theta_i^{(t)}$

Implementation of parameter probability distributions conditional on other parameters

```r
normal_pdf <- function(x, mu, sigma) {
  (1 / (sigma * sqrt(2 * pi))) * exp(-((x - mu)^2 / (2 * sigma^2)))
};

condprob_p = function(p, mu1, mu2, sigma1, sigma2){
  product  = 1;
  for (i in 1:n) {
    product = product * p * normal_pdf(X[i], mu1, sigma1) + (1-p) * normal_pdf(X[i], mu2, sigma2);
  };
  return(product);
};

condprob_mu1 = function(p, mu1, mu2, sigma1, sigma2){
  product  = 1;
  for (i in 1:n) {
    product = product * p * normal_pdf(X[i], mu1, sigma1) + (1-p) * normal_pdf(X[i], mu2, sigma2);
  };
  return(product * dcauchy(mu1, location = 0, scale = 1));
};

condprob_mu2 = function(p, mu1, mu2, sigma1, sigma2){
  product  = 1;
  for (i in 1:n) {
    product = product * p * normal_pdf(X[i], mu1, sigma1) + (1-p) * normal_pdf(X[i], mu2, sigma2);
  };
  return(product * dcauchy(mu2, location = 0, scale = 1));
};

condprob_sigma1 = function(p, mu1, mu2, sigma1, sigma2){
  product  = 1;
  for (i in 1:n) {
    product = product * p * normal_pdf(X[i], mu1, sigma1) + (1-p) * normal_pdf(X[i], mu2, sigma2);
  };
  return(product * dgamma(sigma1, shape = 0.1, rate = 100));
};

condprob_sigma2 = function(p, mu1, mu2, sigma1, sigma2){
  product  = 1;
  for (i in 1:n) {
    product = product * p * normal_pdf(X[i], mu1, sigma1) + (1-p) * normal_pdf(X[i], mu2, sigma2);
  }
  return(product * dgamma(sigma2, shape = 0.1, rate = 100));
};
```

Implementation of Gibbs Sampler with Metropolis Hastings to perform sampling of individual parameters conditional on other parameters

```
faithful_gibbs <- function(p_0, mu1_0, mu2_0, sig1_0, sig2_0, T, X, n, delta) {
  p_chain <- numeric(T);
  mu1_chain <- numeric(T);
  mu2_chain <- numeric(T);
  sig1_chain <- numeric(T);
  sig2_chain <- numeric(T);

  for (i in 1:T) {
   if (i == 1) {
     p_prev <- p_0;
      mu1_prev <- mu1_0;
      mu2_prev <- mu2_0;
      sig1_prev <- sig1_0;
      sig2_prev <- sig2_0;
    } else {
      p_prev <- p_chain[i-1];
      mu1_prev <- mu1_chain[i-1];
      mu2_prev <- mu2_chain[i-1];
      sig1_prev <- sig1_chain[i-1];
      sig2_prev <- sig2_chain[i-1];
    };
    u <- runif(1,0,1);
    e <- runif(1, -delta, delta);

    ### generate next iteration of p
    p_proposal <- p_prev + e;

    accep_prob <- min(1,
                      condprob_p(p_proposal,mu1_prev, mu2_prev, sig1_prev, sig2_prev) /
                        condprob_p(p_prev, mu1_prev, mu2_prev, sig1_prev, sig2_prev));


    if (is.nan(accep_prob) || is.infinite(accep_prob)) {
      accep_prob <- 1;
    } else if (u <= accep_prob) {
      p_current <- p_proposal;
    } else {
      p_current <- p_prev;
    };

    ### generate next iteration of mu_1
```

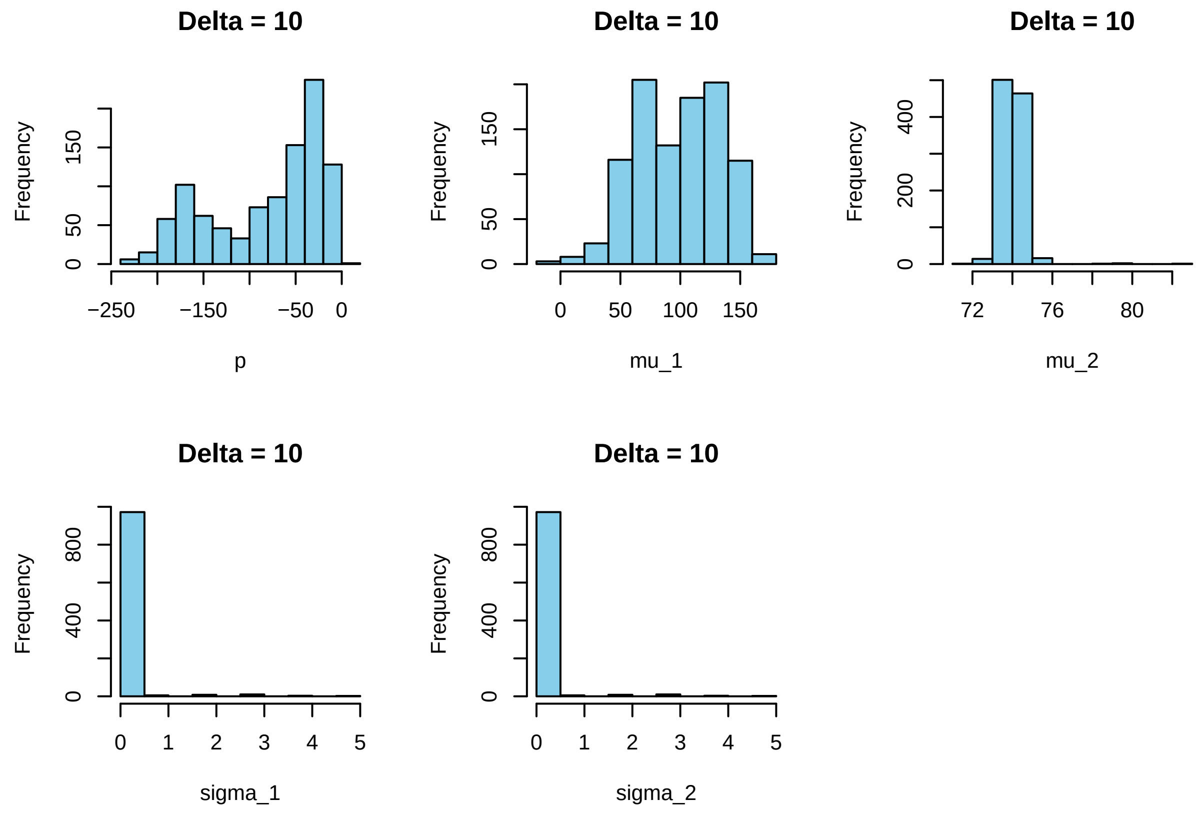
```r
    mu1_proposal <- mu1_prev + e;

    accep_prob <- min(1,
                      condprob_mu1(p_current, mu1_proposal, mu2_prev, sig1_prev, sig2_prev) /
                        condprob_mu2(p_current, mu1_prev, mu2_prev, sig1_prev, sig2_prev));

    if (is.nan(accep_prob) || is.infinite(accep_prob)) {
      accep_prob <- 1;
    } else if (u <= accep_prob) {
      mu1_current <- mu1_proposal;
    } else {
      mu1_current <- mu1_prev;
    };

    ### generate next iteration of mu_2
    mu2_proposal <- mu2_prev + e;

    accep_prob <- min(1,
                      condprob_mu2(p_current, mu1_current, mu2_proposal, sig1_prev, sig2_prev) /
                        condprob_mu2(p_current, mu1_current, mu2_prev, sig1_prev, sig2_prev));

    if (is.nan(accep_prob) || is.infinite(accep_prob)) {
      accep_prob <- 1;
    } else if (u <= accep_prob) {
      mu2_current <- mu2_proposal;
    } else {
      mu2_current <- mu2_prev;
    };

    ### generate next iteration of sigma_1
    sig1_proposal <- sig1_prev + e;

    accep_prob <- min(1,
                      condprob_sigma1(p_current, mu1_current, mu2_current, sig1_proposal, sig2_prev) /
                        condprob_sigma1(p_current, mu1_current, mu2_current, sig1_prev, sig2_prev));

    if (is.nan(accep_prob) || is.infinite(accep_prob)) {
      accep_prob <- 1;
    } else if (u <= accep_prob) {
      sig1_current <- sig1_proposal;
    } else {
      sig1_current <- sig1_prev;
    };

    ### generate next iteration of sigma_2
    sig2_proposal <- sig2_prev + e;

    accep_prob <- min(1,
                      condprob_sigma2(p_current, mu1_current, mu2_current, sig1_current, sig2_proposal)
                        condprob_sigma2(p_current, mu1_current, mu2_current, sig1_current, sig2_prev));

    if (is.nan(accep_prob) || is.infinite(accep_prob)) {
      accep_prob <- 1;
    } else if (u <= accep_prob) {
      sig2_current <- sig2_proposal;
    } else {
      sig2_current <- sig2_prev;
    };

    # Storing Sampled Parameters
    p_chain[i] <- p_current;
    mu1_chain[i] <- mu1_current;
    mu2_chain[i] <- mu2_current;
    sig1_chain[i] <- sig1_current;
    sig2_chain[i] <- sig2_current;
  };
  sample_df <- as.data.frame(cbind(p_chain, mu1_chain, mu2_chain, sig1_chain, sig2_chain));
  colnames(sample_df) <- c("p","mu_1", "mu_2", "sigma_1", "sigma_2");

  return(sample_df);
};

###### END
```

Fitting the model: different results based on perturbation parameter

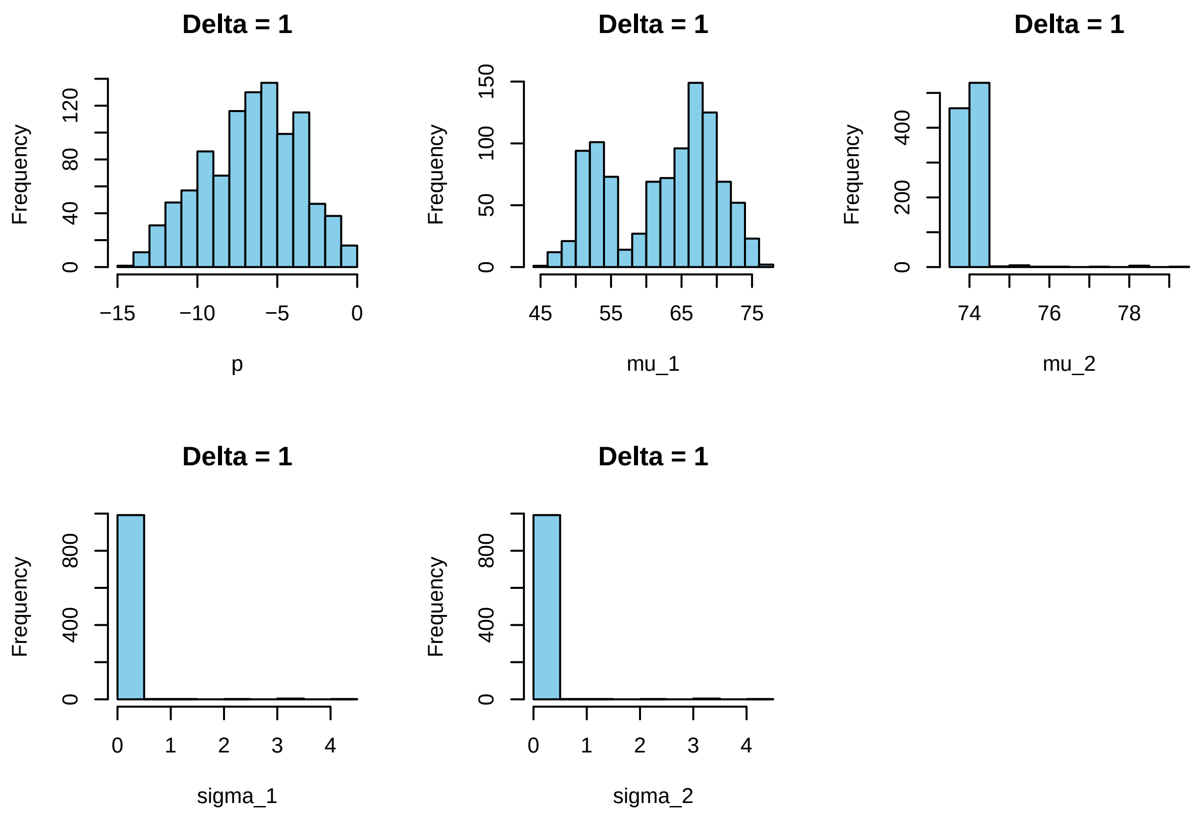
Using number of iterations $T = 1000$, I set initial state $(p^{(0)}, \mu_1^{(0)}, \mu_2^{(0)}, \sigma_1^{(0)}, \sigma_2^{(0)}) = (0.4, 55, 80, 5, 5)$ and compare the outcome of my Metropolis-Hastings within Gibbs Sampling approach under different choices of perturbation parameter: $\delta \in (10, 1, 0.1, 0.01, 0.001)$.

```r
X <- faithful$waiting;
n <- length(X);

T <- 1000;

faithful_simulation1 <- faithful_gibbs(0.4, 55, 80, 5, 5, 1000, X, n, 10);
faithful_simulation2 <- faithful_gibbs(0.4, 55, 80, 5, 5, 1000, X, n, 1);
faithful_simulation3 <- faithful_gibbs(0.4, 55, 80, 5, 5, 1000, X, n, 0.1);
faithful_simulation4 <- faithful_gibbs(0.4, 55, 80, 5, 5, 1000, X, n, 0.01);
faithful_simulation5 <- faithful_gibbs(0.4, 55, 80, 5, 5, 1000, X, n, 0.001);
```

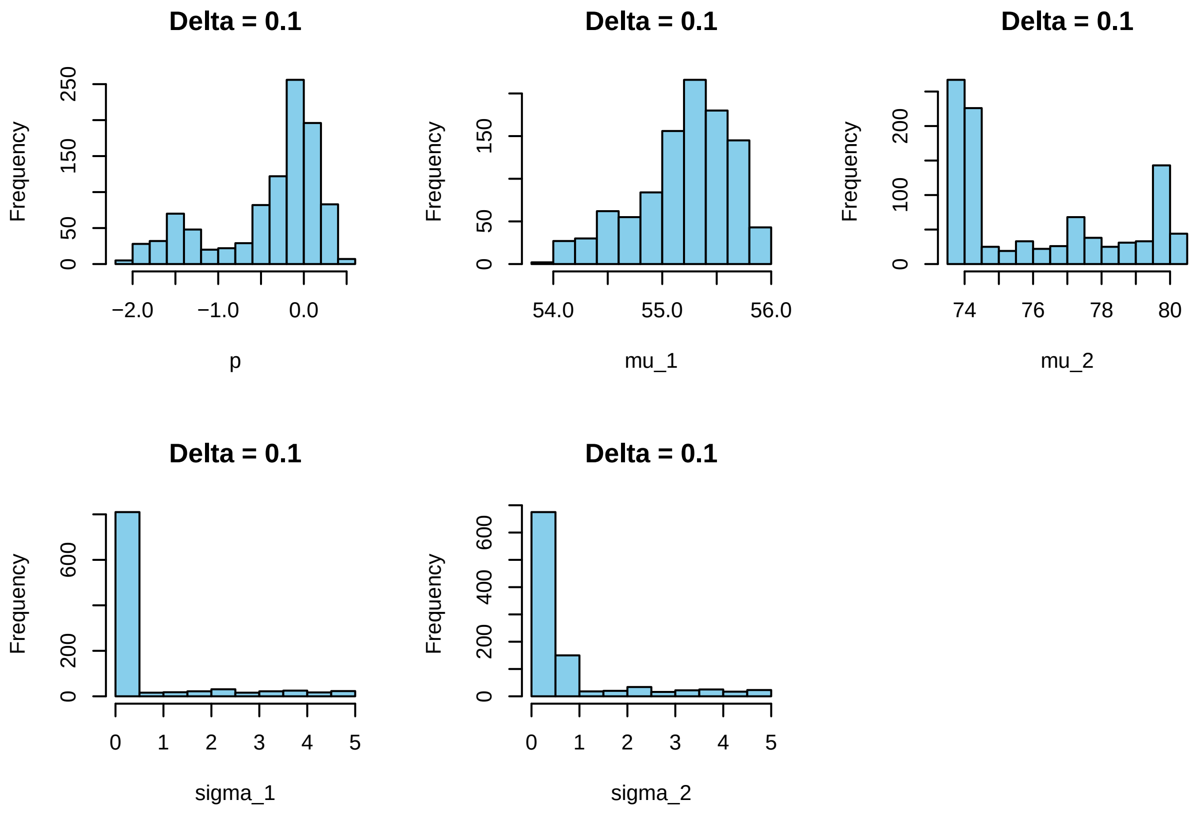
```r
par(mfrow = c(2, 3));

for (i in 1:5) {
  hist(faithful_simulation1[, i],
       main = "Delta = 10",
       xlab = colnames(faithful_simulation1)[i], col = "skyblue", border = "black")
};
```



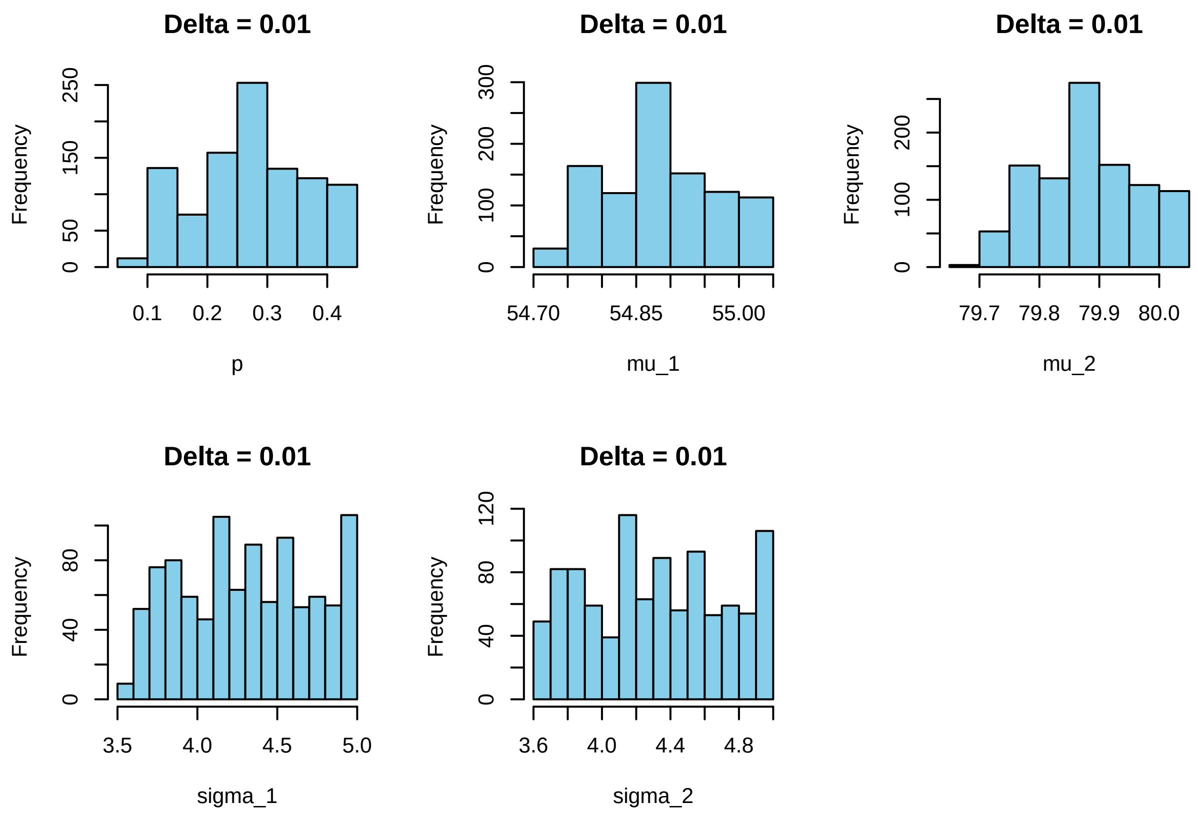
```r
par(mfrow = c(2, 3));

for (i in 1:5) {
  hist(faithful_simulation2[, i],
       main = "Delta = 1",
       xlab = colnames(faithful_simulation1)[i], col = "skyblue", border = "black")
};
```



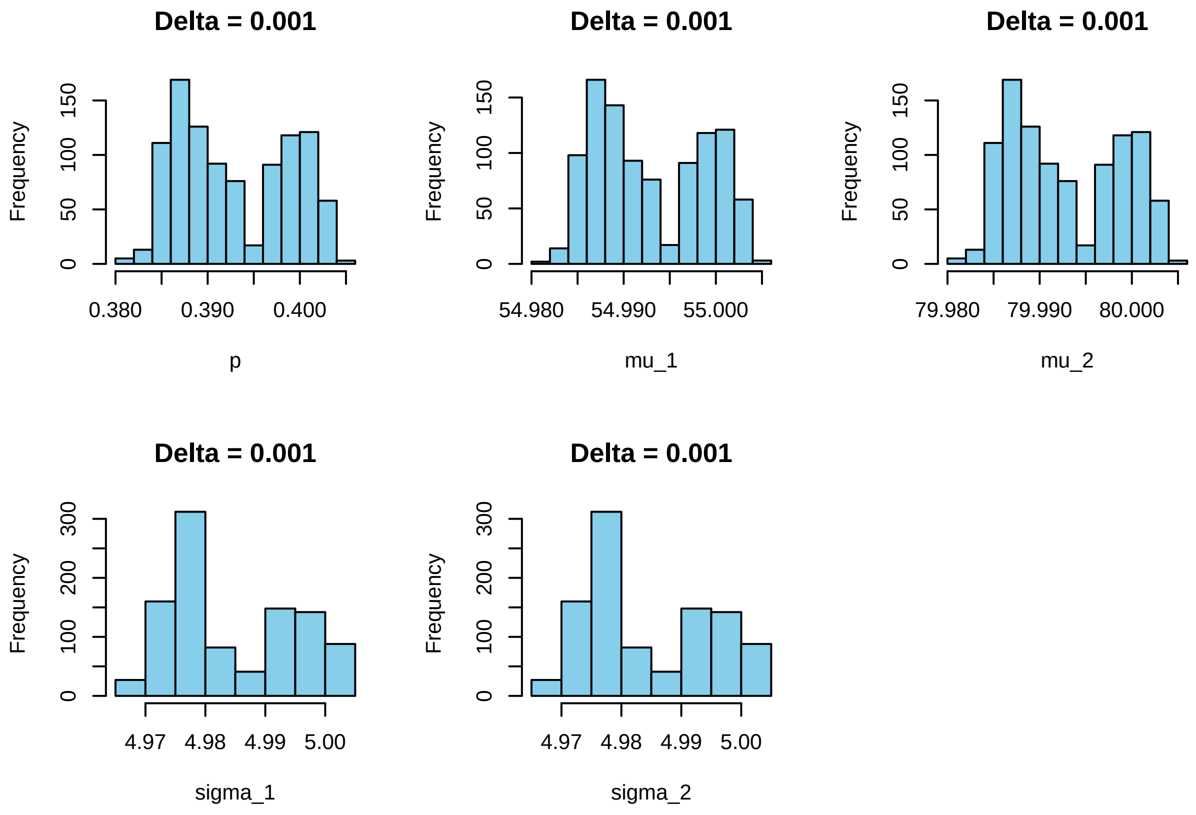
```r
par(mfrow = c(2, 3));

for (i in 1:5) {
  hist(faithful_simulation3[, i],
       main = "Delta = 0.1",
       xlab = colnames(faithful_simulation1)[i], col = "skyblue", border = "black")
};
```



```r
par(mfrow = c(2, 3));

for (i in 1:5) {
  hist(faithful_simulation4[, i],
       main = "Delta = 0.01",
       xlab = colnames(faithful_simulation1)[i], col = "skyblue", border = "black")
};
```



```r
par(mfrow = c(2, 3));

for (i in 1:5) {
  hist(faithful_simulation5[, i],
       main = "Delta = 0.001",
       xlab = colnames(faithful_simulation1)[i], col = "skyblue", border = "black")
};
```



Using the empirical posterior mean of the parameters to fit the model and compare against the observed data:

I select the results of Metropolis-Hastings within Gibbs Sampling implemented with initial state $(p^{(0)}, \mu_1^{(0)}, \mu_2^{(0)}, \sigma_1^{(0)}, \sigma_2^{(0)}) = (0.4, 55, 80, 5, 5)$ and perturbation $\delta = 0.01$ to fit the model $X|\{p, \mu_1, \mu_2, \sigma_1, \sigma_2\}$, i.e. faithful_simulation4.

I use the empirical mean of the parameters to fit the model, generate 300 predicted values from the model, and compare these 300 predicted values against the original observations to see how well this model has fit the observed waiting times.

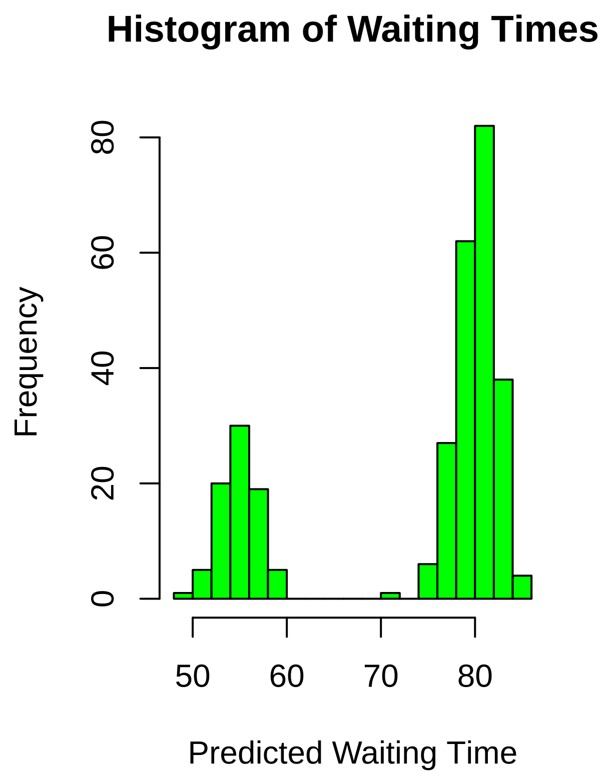
```
# Generate Empirical Posterior Mean
PM <- colMeans(faithful_simulation4);
print(PM);
```

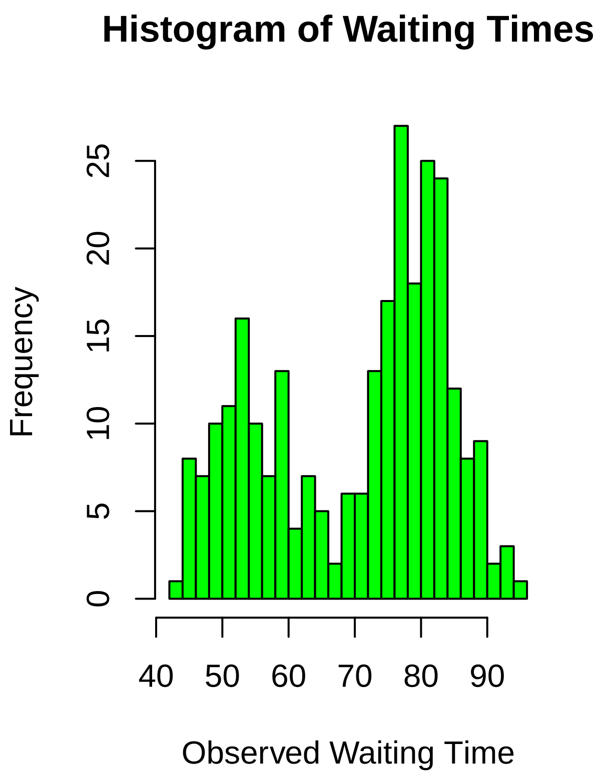
```
##          p       mu_1       mu_2    sigma_1    sigma_2
##  0.2751643 54.8849861 79.8822334  4.3115194  4.3151295
```

```
library(MASS);

sample_from_posterior <- function(PM, n) {
  p <- PM[1];
  mu1 <- PM[2];
```

```r
  mu2 <- PM[3];
  sigma1 <- PM[4];
  sigma2 <- PM[5];

  X <- numeric(n);

  for (i in 1:n) {
    coinflip <- rbinom(1,1,p);
    if (coinflip == 1) {
      X[i] <- rnorm(1, mu1, sqrt(sigma1));
    } else {
      X[i] <- rnorm(1, mu2, sqrt(sigma2));
    };
  };
  return(X);
};
```

```r
par(mfrow = c(1, 2));


hist(sample_from_posterior(PM, 300),
     main = "Histogram of Waiting Times",
     xlab="Predicted Waiting Time", breaks = 20, col = "green");
hist(faithful$waiting, breaks = 20,
     main= "Histogram of Waiting Times",
     xlab="Observed Waiting Time", ylab="Frequency", col="green");
```





The histograms of observed waiting times vs posterior-mean sampled waiting times are similar, with the difference being that the observed waiting times appear to have more variance. The result here suggests that given the model is initialized with appropriate starting parameters and a good $\delta > 0$ tuning parameter, it is possible to develop a model that well-fits the observed data.

A good next step would be to collect more new data on waiting times between eruptions of Old Faithful, then see if the predictions of this fitted model are close to the new observed waiting times.