FMM - Rcpp

2023-06-09

Updated Note (6/10/2023)

- Instead of calculating the reallocation probability for the empty clusters by using the marginal distribution, I plug in μ_k and σ^2_k (sample from prior), into the likelihood.

Note

- For the Rcpp code, the code is on Github. (branch: `test`)
- I will use the same datasets as in the previous report. (FMM - R; 6/8/2023)
- Based on the comment, I will run 10,000 iterations in total, but I will let the first 7,500 iterations as a burn-in.

Model

The derivation for the posterior parameters is in `derive_fmm.jpeg`.

$$\begin{aligned} Y_i|c_i = k, \mu, \sigma^2 &\sim \text{N}\left(\mu_k, \sigma^2_k\right) \\ \mu_k &\sim \text{N}\left(\mu_0, \sigma^2_0\right) \\ \sigma^2_k &\sim \text{Inv-Gamma}\left(a, b\right) \\ c_i|\mathbf{w}_i &\sim \text{Multinomial}\left(1, \mathbf{w}_i\right) \\ \mathbf{w}_i &\sim \text{Dirichlet}\left(\xi_1, \xi_2, \cdots, \xi_K\right) \end{aligned}$$

Hyperparameters

According to the model, all clusters will have the same hyperparameters $\left(\mu_0, \sigma^2_0, a, b\right)$. To use the noninformative prior, I will let $\mu_0 = 0$, $\sigma^2_0 = 100$, $a = b = 1$. Also, I will let $\xi_1 = \xi_2 = \cdots = \xi_K = 1$.

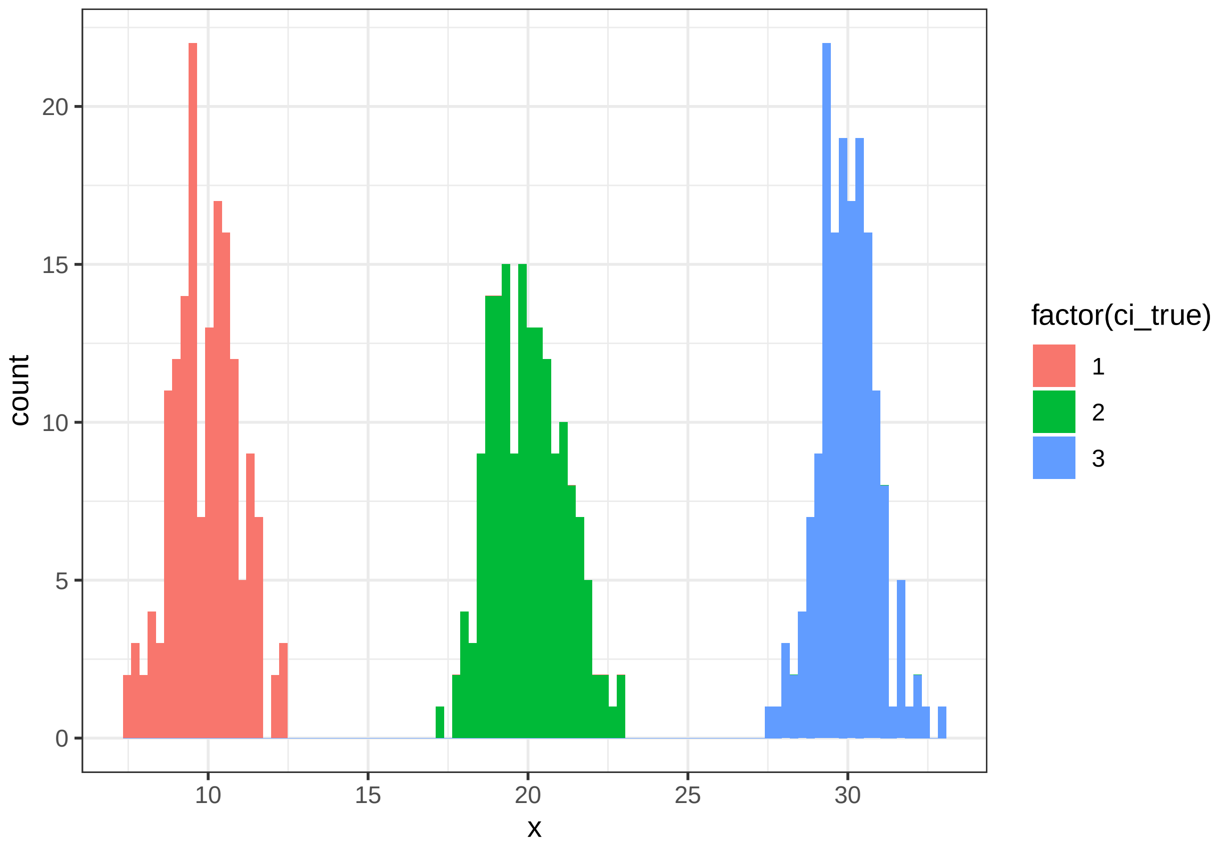
Analysis

For each cases, I will run the model for the one simulated dataset first. Followed by run the model parallel to see that the model provides the stable result or not.

(1) This is the scenario that we discuss during the yesterday's meeting.

```r
### Data Simulation: (1)
set.seed(1843)
N <- 500
K <- 3
ci_true <- sample(1:K, N, replace = TRUE)
```

```r
dat_sim <- rnorm(N, c(10, 20, 30)[ci_true], 1)
ggplot(data.frame(x = dat_sim, ci_true), aes(x = x, fill = factor(ci_true))) +
  geom_histogram(bins = 100) +
  theme_bw()
```



Below is the result from the model.

```r
### Run the model: (1)
test_result <- fmm_rcpp(iter = 10000, y = dat_sim, K_max = K,
                        a0 = 1, b0 = 1, mu0 = 0, s20 = 100, xi0 = 1,
                        ci_init = rep(0, N))

### salso result: (1)
table(salso(test_result$assign_mat[-c(1:7500), ], maxNClusters = K), ci_true)
```

```
##    ci_true
##       1   2   3
##   1   0 170   0
##   2   0   0 166
##   3 164   0   0
```

The result looks good. The posterior mean for each cluster also look reasonable.

```
apply(test_result$mu[-c(1:7500), ], 2, mean)
```

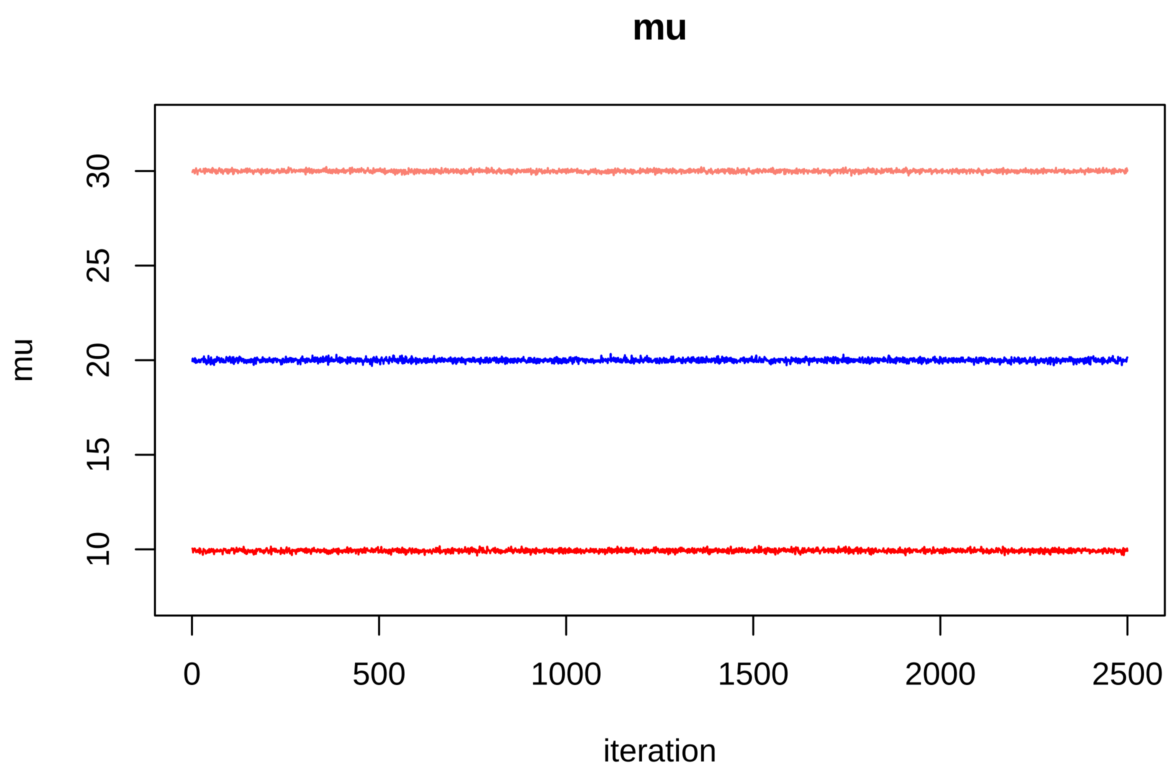
```
## [1]  9.927253 19.993228 29.997072
```

```
apply(test_result$sigma2[-c(1:7500), ], 2, mean)
```

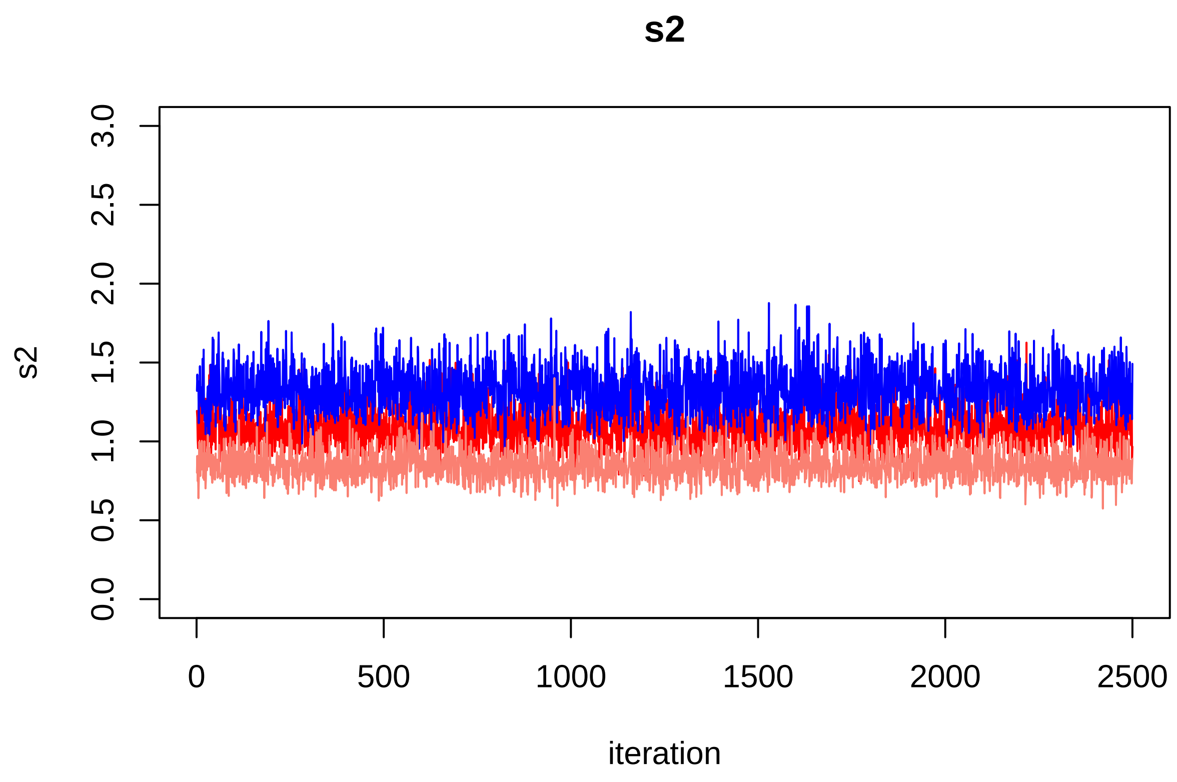
```
## [1] 1.0915394 1.3306977 0.8456725
```

The trace plot for all parameters are converges.

```
plot(test_result$mu[-c(1:7500), 1], type = "l", ylim = c(7.5, 32.5),
     col = "red", main = "mu", ylab = "mu", xlab = "iteration")
lines(1:2500, test_result$mu[-c(1:7500), 2], col = "blue")
lines(1:2500, test_result$mu[-c(1:7500), 3], col = "salmon")
```



```
plot(test_result$sigma2[-c(1:7500), 1], type = "l", ylim = c(0, 3),
     col = "red", main = "s2", ylab = "s2", xlab = "iteration")
lines(1:2500, test_result$sigma2[-c(1:7500), 2], col = "blue")
lines(1:2500, test_result$sigma2[-c(1:7500), 3], col = "salmon")
```



Then, I run the model on 10 datasets.

```r
set.seed(352)
registerDoParallel(detectCores() - 1)
list_result <- foreach(i = 1:10) %dorng%{
  N <- 500
  K <- 3
  ci_true <- sample(1:K, N, replace = TRUE)
  dat_sim <- rnorm(N, c(10, 20, 30)[ci_true], 1)
  test_result <- fmm_rcpp(iter = 10000, y = dat_sim, K_max = K,
                          a0 = 1, b0 = 1, mu0 = 0, s20 = 100, xi0 = 1,
                          ci_init = rep(0, N))
  return(list(clus_assign = test_result$assign_mat, ci_true = ci_true))
}
stopImplicitCluster()
```

The model did a perfect job.

```r
jac_vec <- rep(NA, 10)
for(i in 1:10){
  ci_assign <- as.numeric(salso(list_result[[i]]$clus_assign[-c(1:7500), ],
                              maxNClusters = K))
  jac_vec[i] <- mclustcomp(ci_assign, list_result[[i]]$ci_true, "jaccard")$score
}

mean(jac_vec)
```

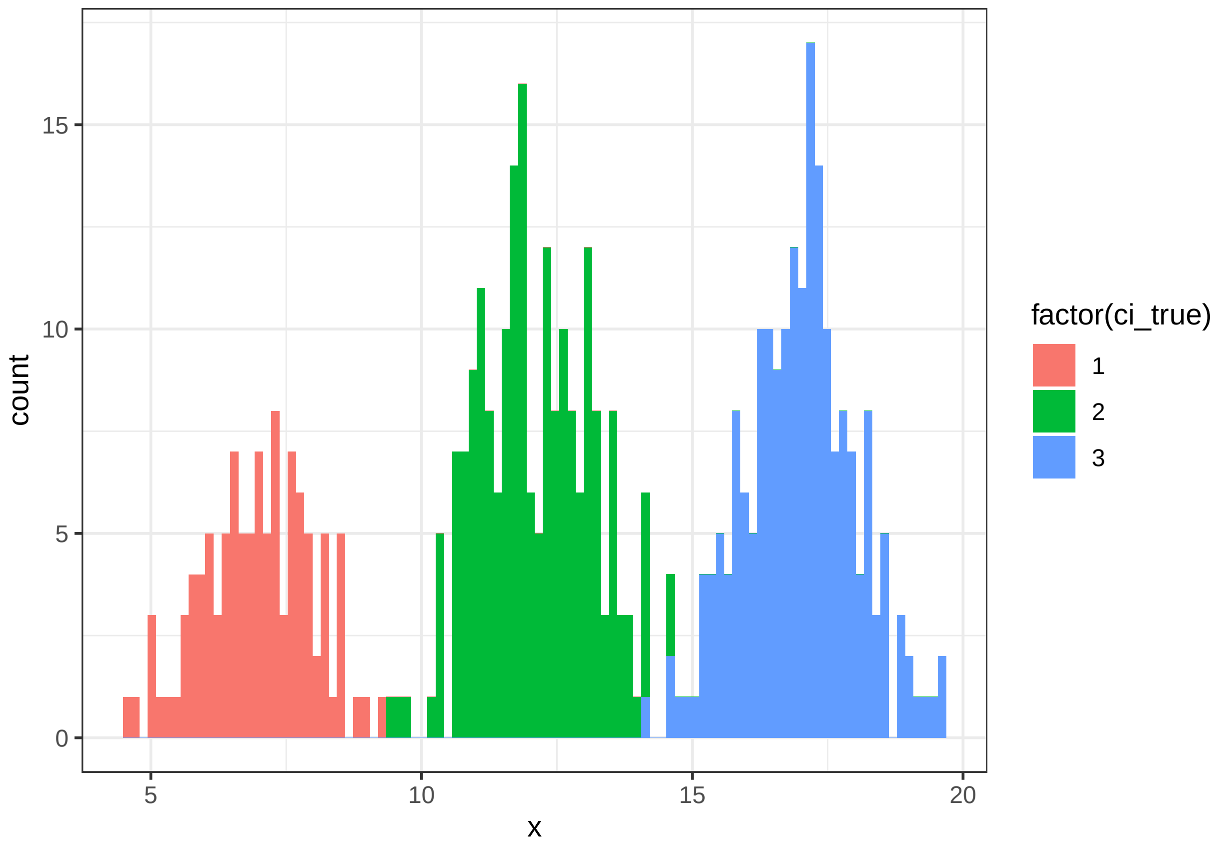
```
## [1] 1
```

```
sd(jac_vec)
```

```
## [1] 0
```

(2) For this case, we will have three (almost) separated clusters. The proportion for each group is 0.25, 0.35, and 0.4

```
### Data Simulation: (2)
set.seed(12441)
N <- 500
K <- 3
ci_true <- sample(1:K, N, replace = TRUE, prob = c(0.25, 0.35, 0.4))
dat_sim <- rnorm(N, c(7, 12, 17)[ci_true], 1)
ggplot(data.frame(x = dat_sim, ci_true), aes(x = x, fill = factor(ci_true))) +
  geom_histogram(bins = 100) +
  theme_bw()
```



```
### Run the model: (2)
test_result <- fmm_rcpp(iter = 10000, y = dat_sim, K_max = K,
                        a0 = 1, b0 = 1, mu0 = 0, s20 = 100, xi0 = 1,
                        ci_init = rep(0, N))
```

```
### salso result: (2)
table(salso(test_result$assign_mat[-c(1:7500), ], maxNClusters = K), ci_true)
```

```
##    ci_true
##       1   2   3
##   1   0 195   1
##   2   0   1 196
##   3 106   1   0
```

The result look good enough to me, and it is similar to the R version.

```
apply(test_result$mu[-c(1:7500), ], 2, mean)
```

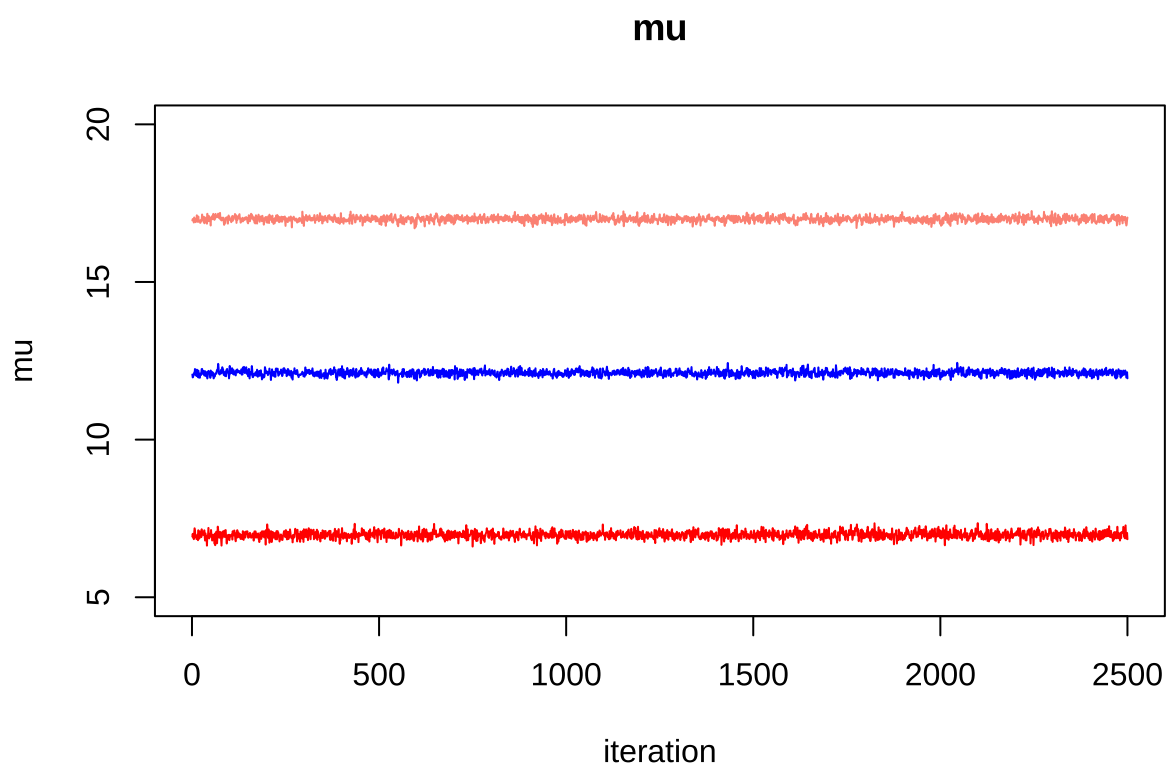
```
## [1]  6.977327 12.115565 16.996549
```

```
apply(test_result$sigma2[-c(1:7500), ], 2, mean)
```

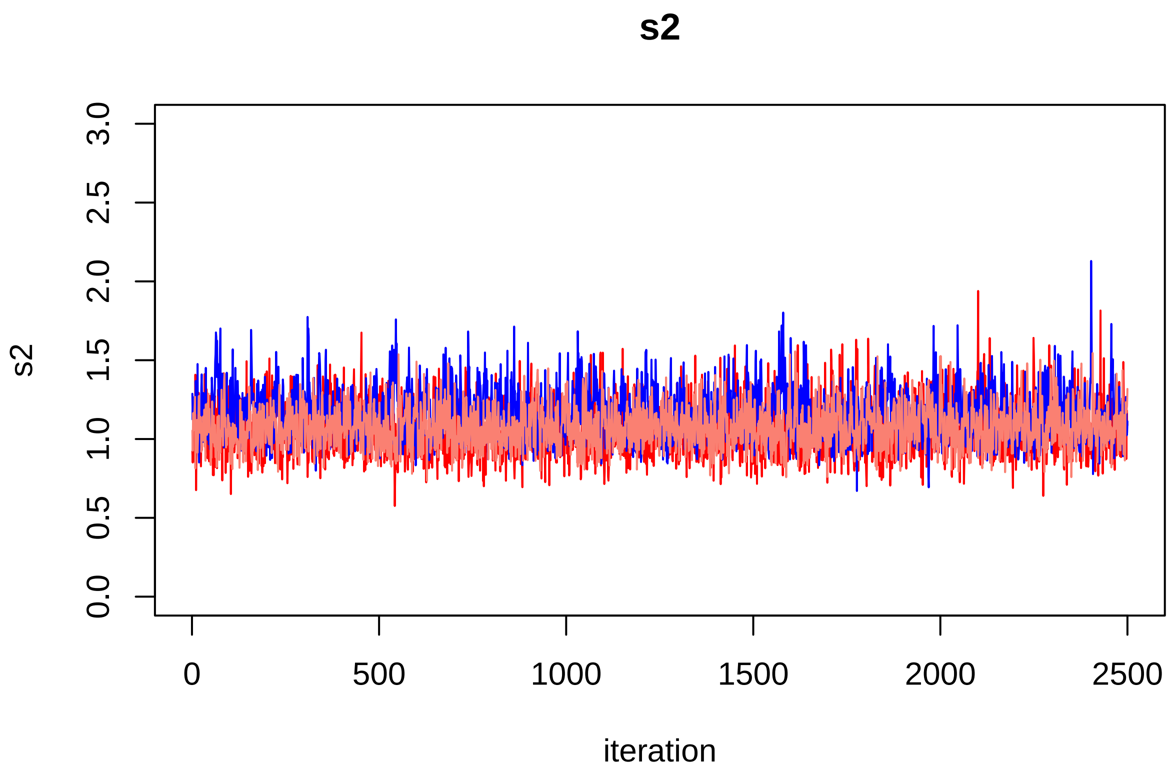
```
## [1] 1.055883 1.163668 1.071564
```

The traceplots show that the parameters are converged.

```
plot(test_result$mu[-c(1:7500), 1], type = "l", ylim = c(5, 20),
     col = "red", main = "mu", ylab = "mu", xlab = "iteration")
lines(1:2500, test_result$mu[-c(1:7500), 2], col = "blue")
lines(1:2500, test_result$mu[-c(1:7500), 3], col = "salmon")
```



```
plot(test_result$sigma2[-c(1:7500), 1], type = "l", ylim = c(0, 3),
     col = "red", main = "s2", ylab = "s2", xlab = "iteration")
lines(1:2500, test_result$sigma2[-c(1:7500), 2], col = "blue")
lines(1:2500, test_result$sigma2[-c(1:7500), 3], col = "salmon")
```



Then, I run the model on 10 datasets.

```
set.seed(352)
registerDoParallel(detectCores() - 1)
list_result <- foreach(i = 1:10) %dorng%{
  N <- 500
  K <- 3
  ci_true <- sample(1:K, N, replace = TRUE, prob = c(0.25, 0.35, 0.4))
  dat_sim <- rnorm(N, c(7, 12, 17)[ci_true], 1)
  test_result <- fmm_rcpp(iter = 10000, y = dat_sim, K_max = K,
                          a0 = 1, b0 = 1, mu0 = 0, s20 = 100, xi0 = 1,
                          ci_init = rep(0, N))
  return(list(clus_assign = test_result$assign_mat, ci_true = ci_true))
}
stopImplicitCluster()
```

The result looks fine as there are some observations that are close to the other clusters.

```r
jac_vec <- rep(NA, 10)
for(i in 1:10){
  ci_assign <- as.numeric(salso(list_result[[i]]$clus_assign[-c(1:7500), ],
                                maxNClusters = K))
  jac_vec[i] <- mclustcomp(ci_assign, list_result[[i]]$ci_true, "jaccard")$score
}

mean(jac_vec)
```

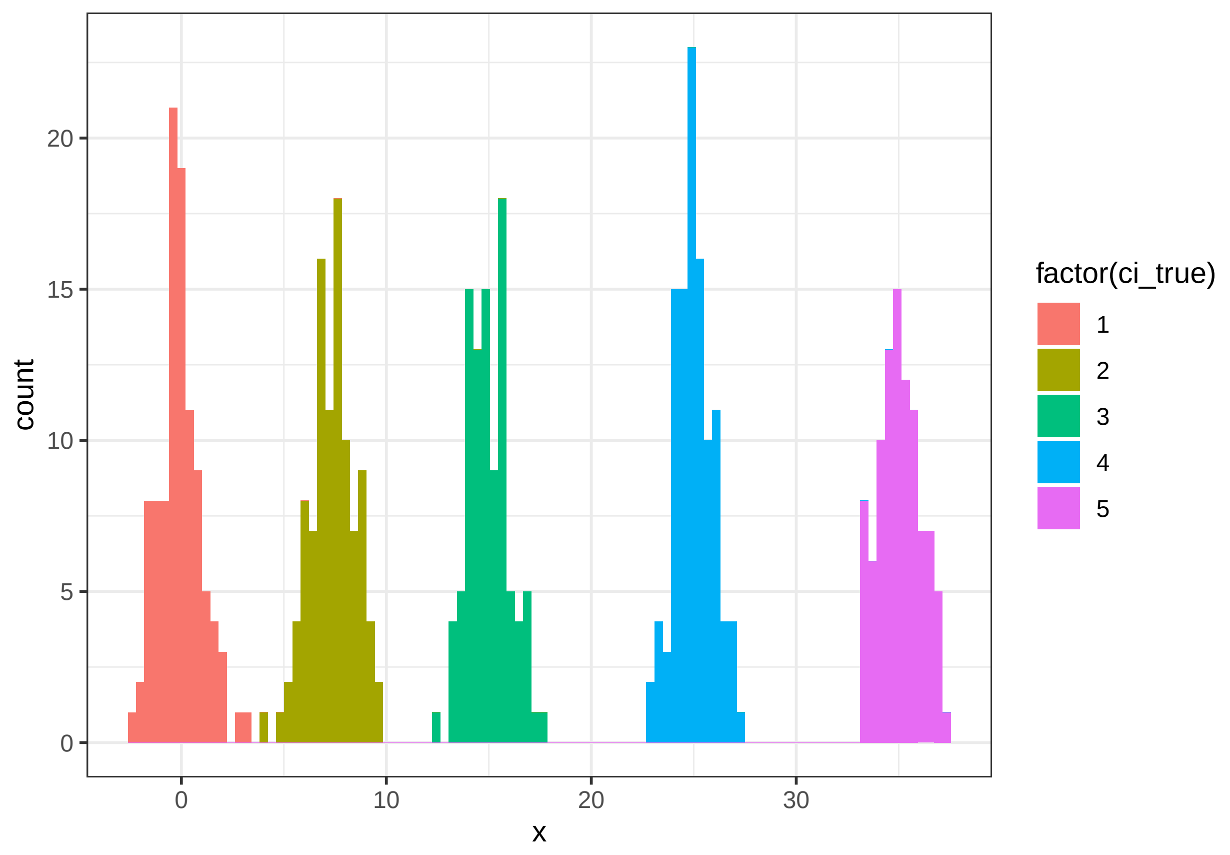
```
## [1] 0.964536
```

```r
sd(jac_vec)
```

```
## [1] 0.01496955
```

(3) For this case, we will have five separated clusters.

```r
### Data Simulation: (3)
set.seed(12441)
N <- 500
K <- 5
ci_true <- sample(1:K, N, replace = TRUE)
dat_sim <- rnorm(N, c(0, 7.5, 15, 25, 35)[ci_true], 1)
ggplot(data.frame(x = dat_sim, ci_true), aes(x = x, fill = factor(ci_true))) +
  geom_histogram(bins = 100) +
  theme_bw()
```

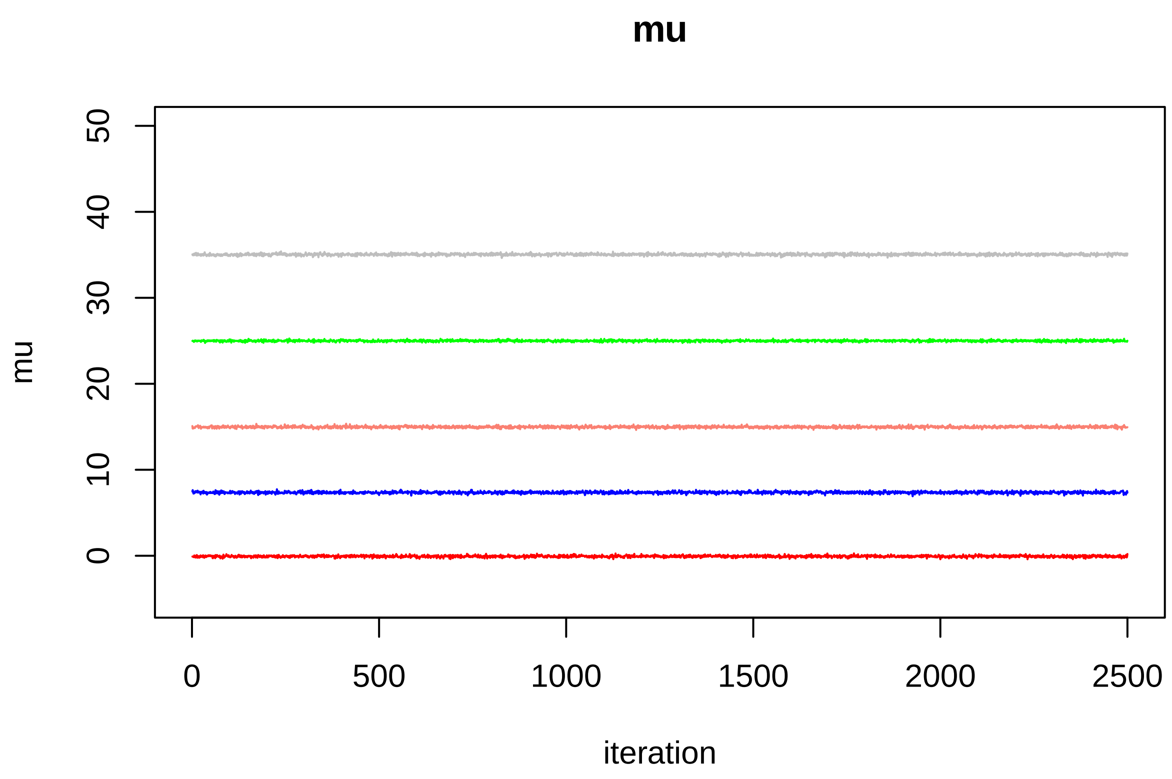


```
### Run the model: (3)
test_result <- fmm_rcpp(iter = 10000, y = dat_sim, K_max = K,
                        a0 = 1, b0 = 1, mu0 = 0, s20 = 100, xi0 = 1,
                        ci_init = rep(0, N))

### salso result: (3)
table(salso(test_result$assign_mat[-c(1:7500), ], maxNClusters = K), ci_true)
```

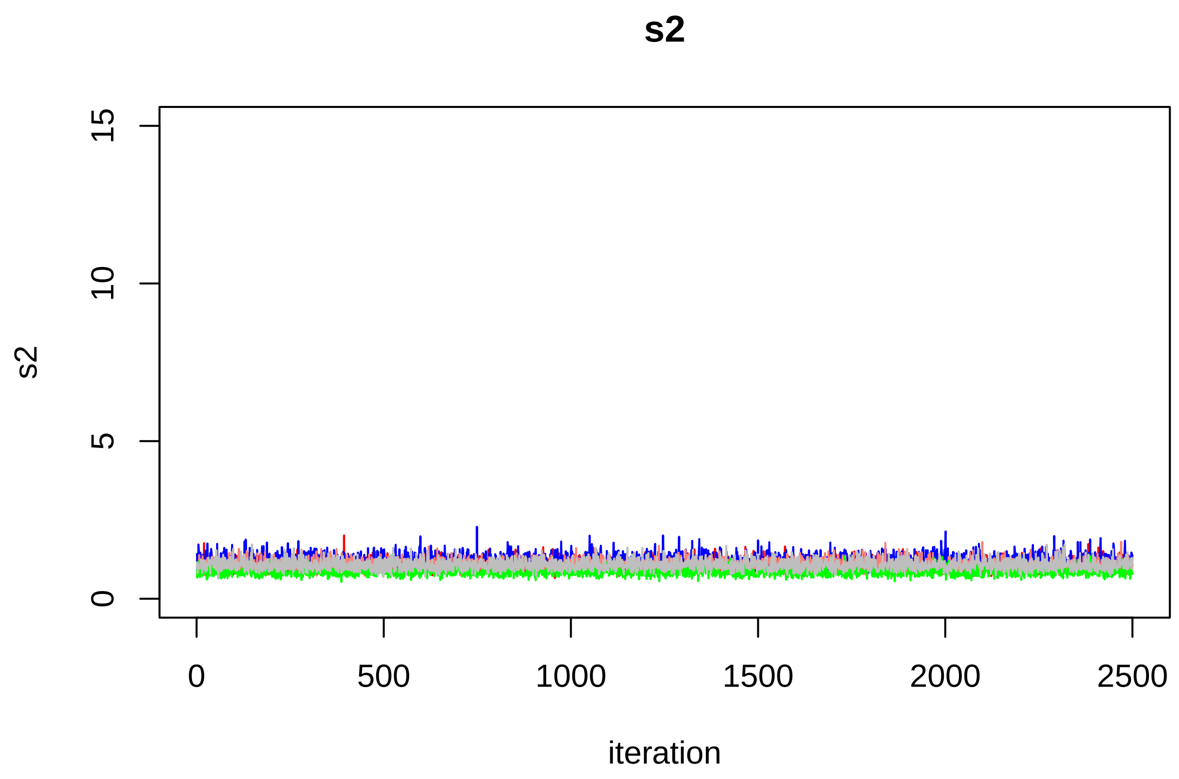
```
##    ci_true
##       1   2   3   4   5
##   1   0   0   0 108   0
##   2   0 100   0   0   0
##   3   0   0   0   0  95
##   4 101   0   0   0   0
##   5   0   0  96   0   0
```

The model performs great. The traceplots also show that the parameters are converged.

```
plot(test_result$mu[-c(1:7500), 1], type = "l", ylim = c(-5, 50),
     col = "red", main = "mu", ylab = "mu", xlab = "iteration")
lines(1:2500, test_result$mu[-c(1:7500), 2], col = "blue")
lines(1:2500, test_result$mu[-c(1:7500), 3], col = "salmon")
lines(1:2500, test_result$mu[-c(1:7500), 4], col = "green")
lines(1:2500, test_result$mu[-c(1:7500), 5], col = "grey")
```



```
plot(test_result$sigma2[-c(1:7500), 1], type = "l", ylim = c(0, 15),
     col = "red", main = "s2", ylab = "s2", xlab = "iteration")
lines(1:2500, test_result$sigma2[-c(1:7500), 2], col = "blue")
lines(1:2500, test_result$sigma2[-c(1:7500), 3], col = "salmon")
lines(1:2500, test_result$sigma2[-c(1:7500), 4], col = "green")
lines(1:2500, test_result$sigma2[-c(1:7500), 5], col = "grey")
```



Then, I run the model on 10 datasets.

```
set.seed(352)
registerDoParallel(detectCores() - 1)
list_result <- foreach(i = 1:10) %dorng%{
  N <- 500
  K <- 5
  ci_true <- sample(1:K, N, replace = TRUE)
  dat_sim <- rnorm(N, c(0, 7.5, 15, 25, 35)[ci_true], 1)
  test_result <- fmm_rcpp(iter = 10000, y = dat_sim, K_max = K,
                          a0 = 1, b0 = 1, mu0 = 0, s20 = 100, xi0 = 1,
                          ci_init = rep(0, N))
  return(list(clus_assign = test_result$assign_mat, ci_true = ci_true,
              mu = test_result$mu))
}
stopImplicitCluster()
```

```
jac_vec <- rep(NA, 10)
for(i in 1:10){
  ci_assign <- as.numeric(salso(list_result[[i]]$clus_assign[-c(1:7500), ],
                                maxNClusters = K))
  jac_vec[i] <- mclustcomp(ci_assign, list_result[[i]]$ci_true, "jaccard")$score
}

mean(jac_vec)
```

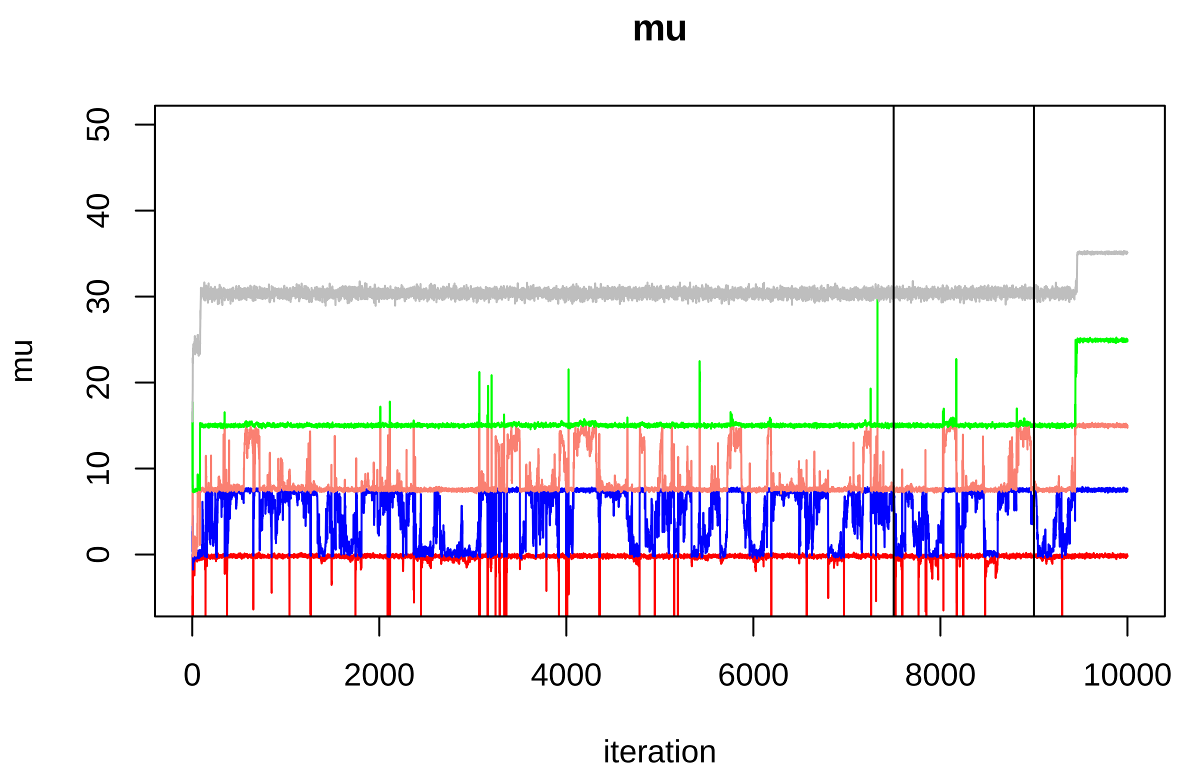
```
## [1] 0.9700336
```

```
sd(jac_vec)
```

```
## [1] 0.094762
```

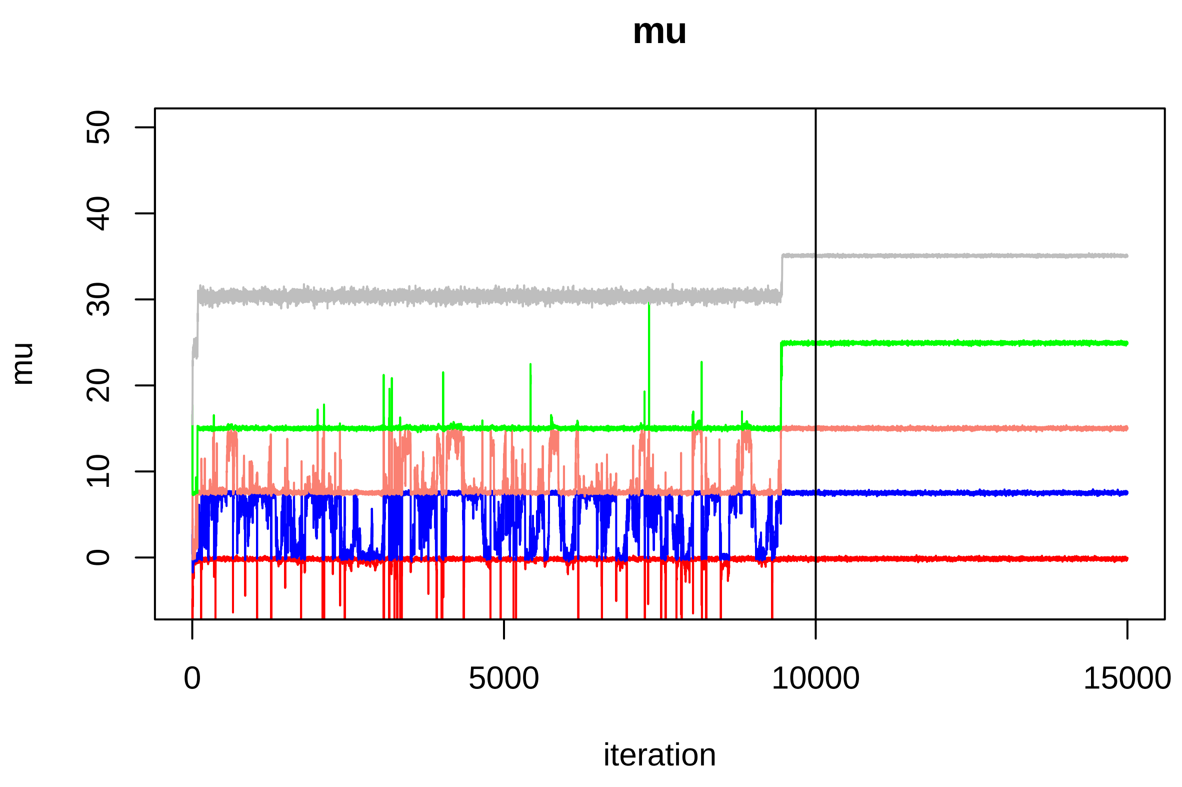
I noticed that there is one chain that might not be converged. Based on the trace plot below, I decide to increase the number of iteration to 15,000 iterations and let the first 10,000 iterations as a burn-in.

```
plot(list_result[[2]]$mu[, 1], type = "l", ylim = c(-5, 50),
     col = "red", main = "mu", ylab = "mu", xlab = "iteration")
lines(1:10000, list_result[[2]]$mu[, 2], col = "blue")
lines(1:10000, list_result[[2]]$mu[, 3], col = "salmon")
lines(1:10000, list_result[[2]]$mu[, 4], col = "green")
lines(1:10000, list_result[[2]]$mu[, 5], col = "grey")

abline(v = c(7500, 9000))
```



```
set.seed(352)
registerDoParallel(detectCores() - 1)
list_result <- foreach(i = 1:10) %dorng%{
  N <- 500
  K <- 5
  ci_true <- sample(1:K, N, replace = TRUE)
  dat_sim <- rnorm(N, c(0, 7.5, 15, 25, 35)[ci_true], 1)
  test_result <- fmm_rcpp(iter = 15000, y = dat_sim, K_max = K,
                          a0 = 1, b0 = 1, mu0 = 0, s20 = 100, xi0 = 1,
                          ci_init = rep(0, N))
  return(list(clus_assign = test_result$assign_mat, ci_true = ci_true,
              mu = test_result$mu))
}
stopImplicitCluster()
```

```
plot(list_result[[2]]$mu[, 1], type = "l", ylim = c(-5, 50),
     col = "red", main = "mu", ylab = "mu", xlab = "iteration")
lines(1:15000, list_result[[2]]$mu[, 2], col = "blue")
lines(1:15000, list_result[[2]]$mu[, 3], col = "salmon")
lines(1:15000, list_result[[2]]$mu[, 4], col = "green")
lines(1:15000, list_result[[2]]$mu[, 5], col = "grey")

abline(v = 10000)
```



The result look much better when I increase the number of iterations and burn-ins.

```
jac_vec <- rep(NA, 10)
for(i in 1:10){
  ci_assign <- as.numeric(salso(list_result[[i]]$clus_assign[-c(1:10000), ],
                                maxNClusters = K))
  jac_vec[i] <- mclustcomp(ci_assign, list_result[[i]]$ci_true, "jaccard")$score
}

mean(jac_vec)
```

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

```
sd(jac_vec)
```

```
## [1] 0
```