FMM - R

2023-06-08

The code for the function is in `fmm_R.R`.

```
### Import the function from the other file
source("/Users/kevin-imac/Desktop/Github - Repo/ClusterNormal/Other/fmm_R.R")
```

Model

Below is the model for our code. The derivation for the posterior parameters is in `derive_fmm.jpeg`.

$$
\begin{aligned}
Y_i|c_i = k, \mu, \sigma^2 &\sim \text{N}\left(\mu_k, \sigma_k^2\right) \\
\mu_k &\sim \text{N}\left(\mu_0, \sigma_0^2\right) \\
\sigma_k^2 &\sim \text{Inv-Gamma}\left(a, b\right) \\
c_i|\mathbf{w}_i &\sim \text{Multinomial}\left(1, \mathbf{w}_i\right) \\
\mathbf{w}_i &\sim \text{Dirichlet}\left(\xi_1, \xi_2, \cdots, \xi_K\right)
\end{aligned}
$$

Hyperparameters

According to the model, all clusters will have the same hyperparameters $\left(\mu_0, \sigma_0^2, a, b\right)$. To use the noninformative prior, I will let $\mu_0 = 0$, $\sigma_0^2 = 100$, $a = b = 1$. Also, I will let $\xi_1 = \xi_2 = \cdots = \xi_K = 1$.

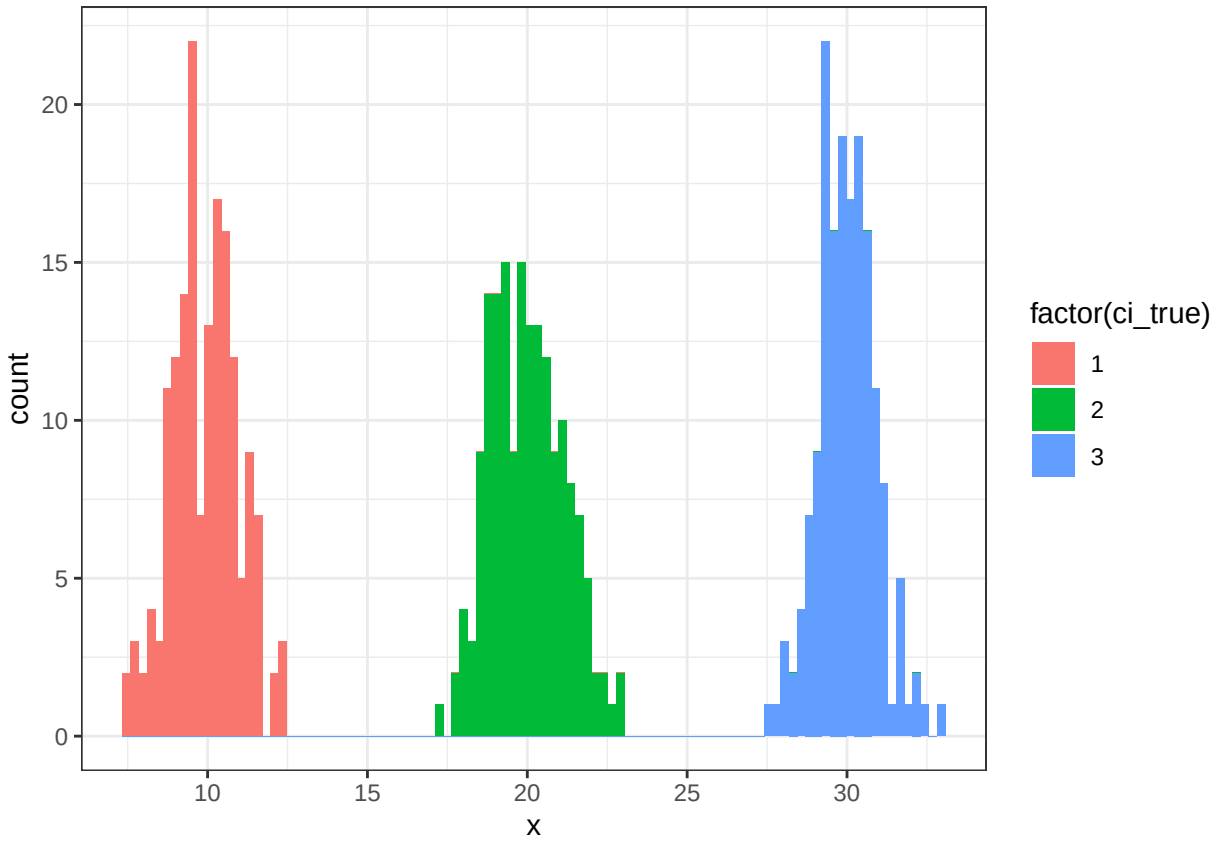
I will run the model for 3,000 iterations and let the first 1,000 iteration as a burn-in.

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 today's meeting.

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

```
### Run the model: (1)
test_result <- fmm_model_R(iter = 3000, dat = dat_sim, K_max = K,
                           a0 = 1, b0 = 1, mu0 = 0, s20 = 100, xi0 = 1,
                           ci_init = rep(1, N))

### salso result: (1)
table(salso(test_result$assign_mat[-c(1:1000), ], 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_mat[-c(1:1000), ], 2, mean)
```

```
## [1]  9.925367 19.996182 29.996162
```

```
apply(test_result$s2_mat[-c(1:1000), ], 2, mean)
```

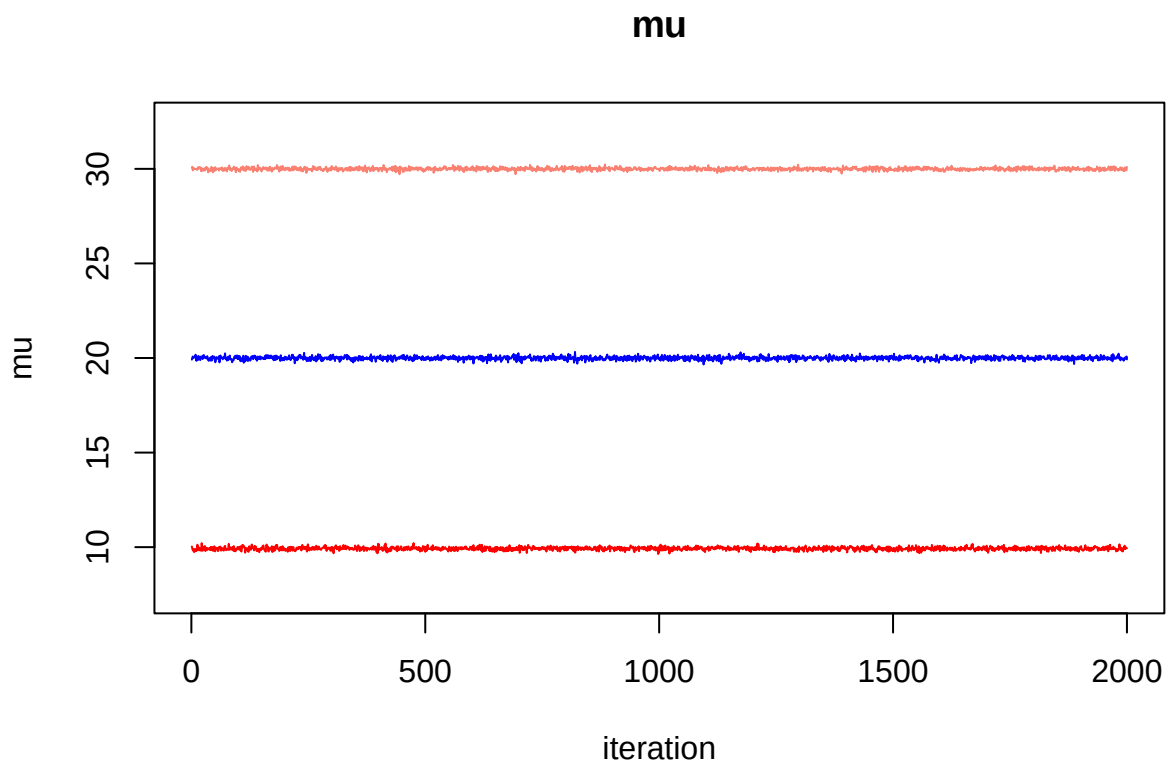
```
## [1] 1.0895816 1.3324600 0.8473314
```

```r
apply(test_result$mixing_mat[-c(1:1000), ], 2, mean)
```

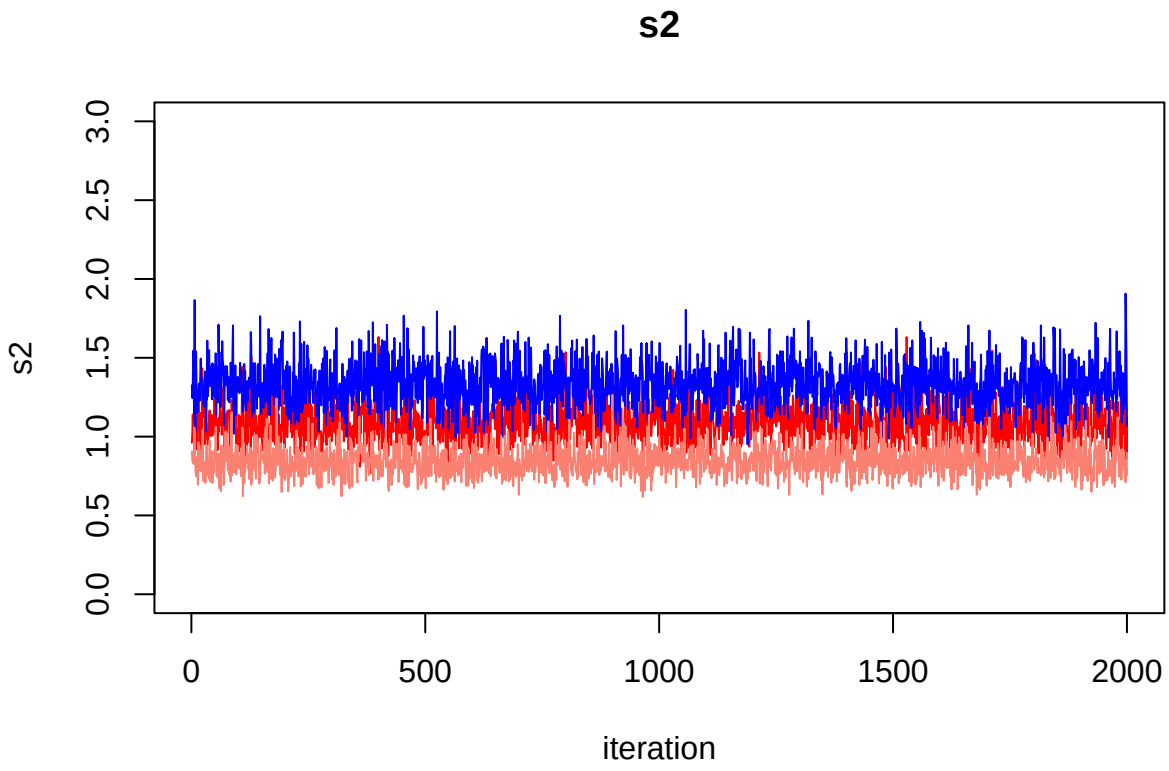
```
## [1] 0.3288842 0.3395668 0.3315491
```

The trace plot for all parameters are converges.

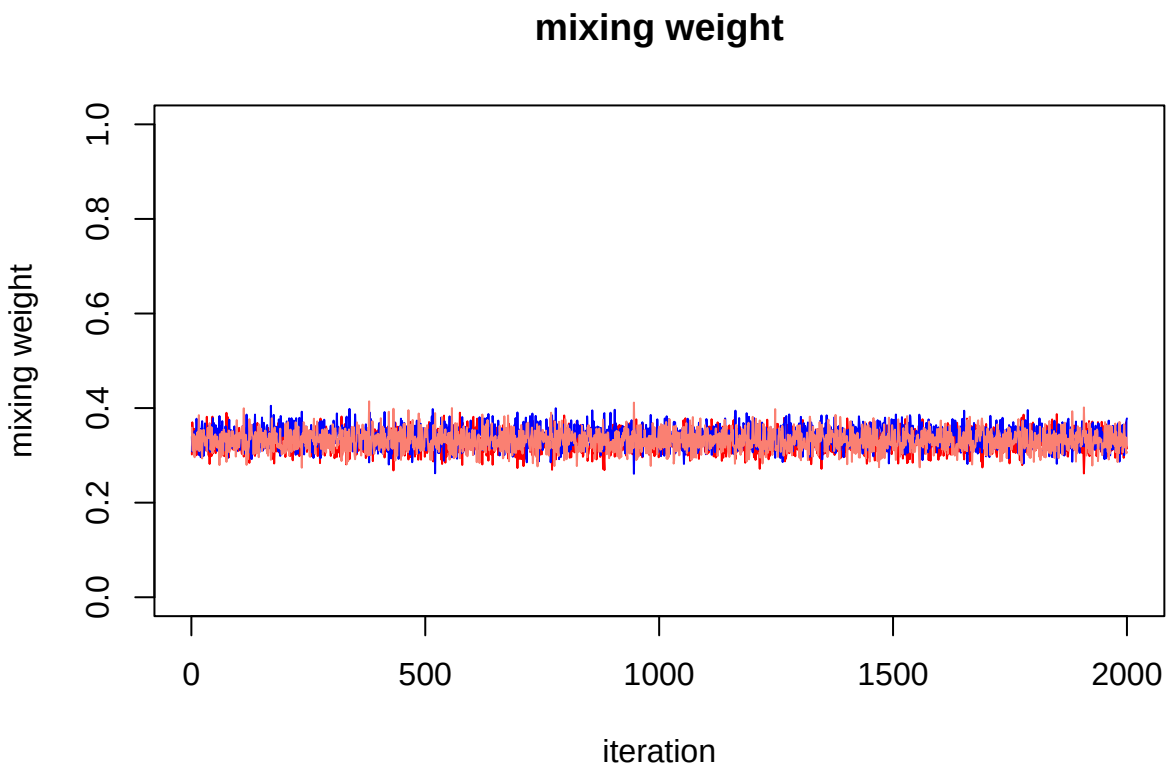
```r
plot(test_result$mu_mat[-c(1:1000), 1], type = "l", ylim = c(7.5, 32.5),
     col = "red", main = "mu", ylab = "mu", xlab = "iteration")
lines(1:2000, test_result$mu_mat[-c(1:1000), 2], col = "blue")
lines(1:2000, test_result$mu_mat[-c(1:1000), 3], col = "salmon")
```




```r
plot(test_result$s2_mat[-c(1:1000), 1], type = "l", ylim = c(0, 3),
     col = "red", main = "s2", ylab = "s2", xlab = "iteration")
lines(1:2000, test_result$s2_mat[-c(1:1000), 2], col = "blue")
lines(1:2000, test_result$s2_mat[-c(1:1000), 3], col = "salmon")
```




```
plot(test_result$mixing_mat[-c(1:1000), 1], type = "l", ylim = c(0, 1),
     col = "red", main = "mixing weight", ylab = "mixing weight",
     xlab = "iteration")
lines(1:2000, test_result$mixing_mat[-c(1:1000), 2], col = "blue")
lines(1:2000, test_result$mixing_mat[-c(1:1000), 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)
  dat_sim <- rnorm(N, c(10, 20, 30)[ci_true], 1)
  test_result <- fmm_model_R(iter = 3000, dat = dat_sim, K_max = K,
                             a0 = 1, b0 = 1, mu0 = 0, s20 = 100, xi0 = 1,
                             ci_init = rep(1, N))
  return(list(clus_assign = test_result$assign_mat, ci_true = ci_true))
}
stopImplicitCluster()
```

We might notice that the jaccard is not exactly 1. (which we expect to see that)

```
jac_vec <- rep(NA, 10)
for(i in 1:10){
  ci_assign <- as.numeric(salso(list_result[[i]]$clus_assign[-c(1:1000), ],
                                maxNClusters = K))
  jac_vec[i] <- mclustcomp(ci_assign, list_result[[i]]$ci_true, "jaccard")$score
}

mean(jac_vec)
```

```
## [1] 0.9592865
```

```
sd(jac_vec)
```

```
## [1] 0.1287474
```

So, I have checked it and notice that the parallel #3 does not perform well. So, I increase the number of the burn-in. The result shows that it is good. (jaccard = 1)

```
jac_vec
```

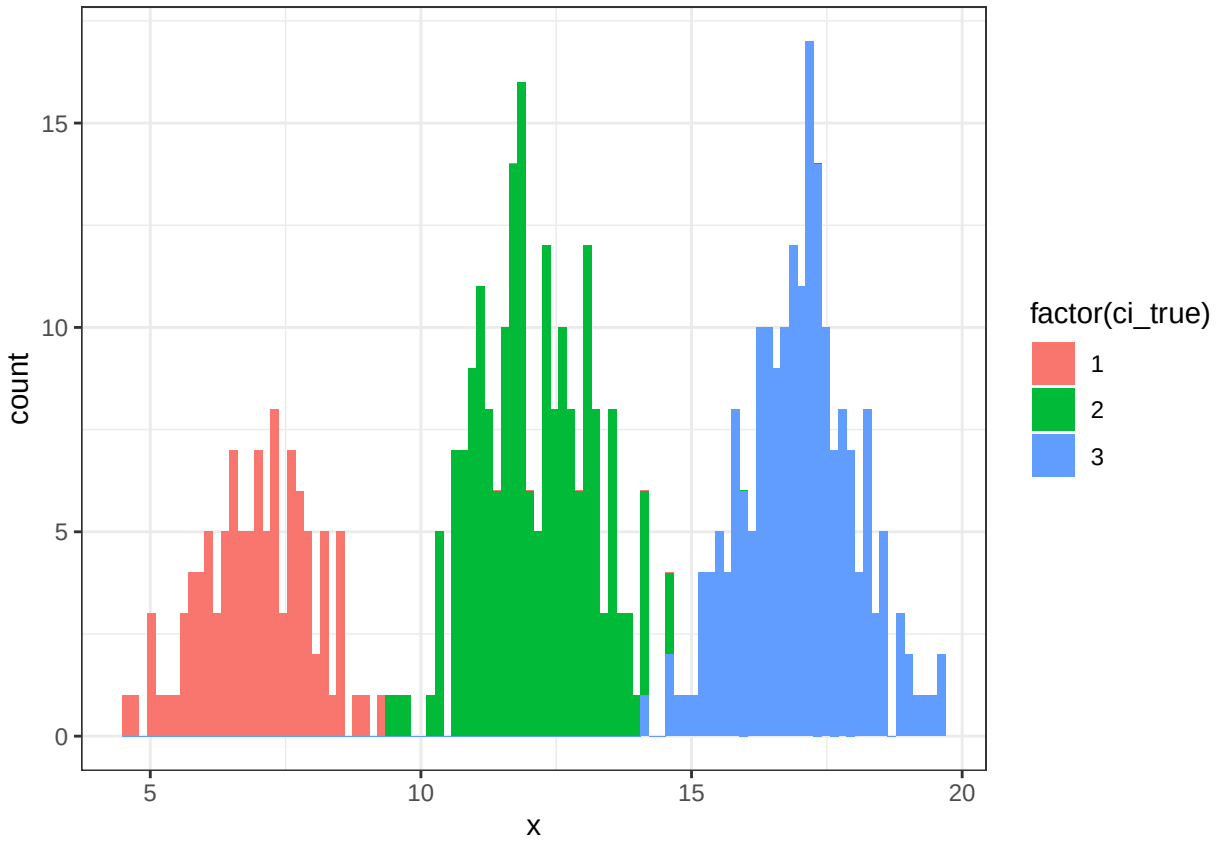
```
##  [1] 1.0000000 1.0000000 0.5928651 1.0000000 1.0000000 1.0000000 1.0000000
##  [8] 1.0000000 1.0000000 1.0000000
```

```
ci_assign <- as.numeric(salso(list_result[[3]]$clus_assign[-c(1:2000), ],
                              maxNClusters = K))
mclustcomp(ci_assign, list_result[[3]]$ci_true, "jaccard")$score
```

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

(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_model_R(iter = 3000, dat = dat_sim, K_max = K,
                           a0 = 1, b0 = 1, mu0 = 0, s20 = 100, xi0 = 1,
                           ci_init = rep(1, N))

### salso result: (2)
table(salso(test_result$assign_mat[-c(1:1000), ], maxNClusters = K), ci_true)
```

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

The result look good enough to me.

```
apply(test_result$mu_mat[-c(1:1000), ], 2, mean)
```

```
## [1]  6.974709 12.116605 16.999115
```

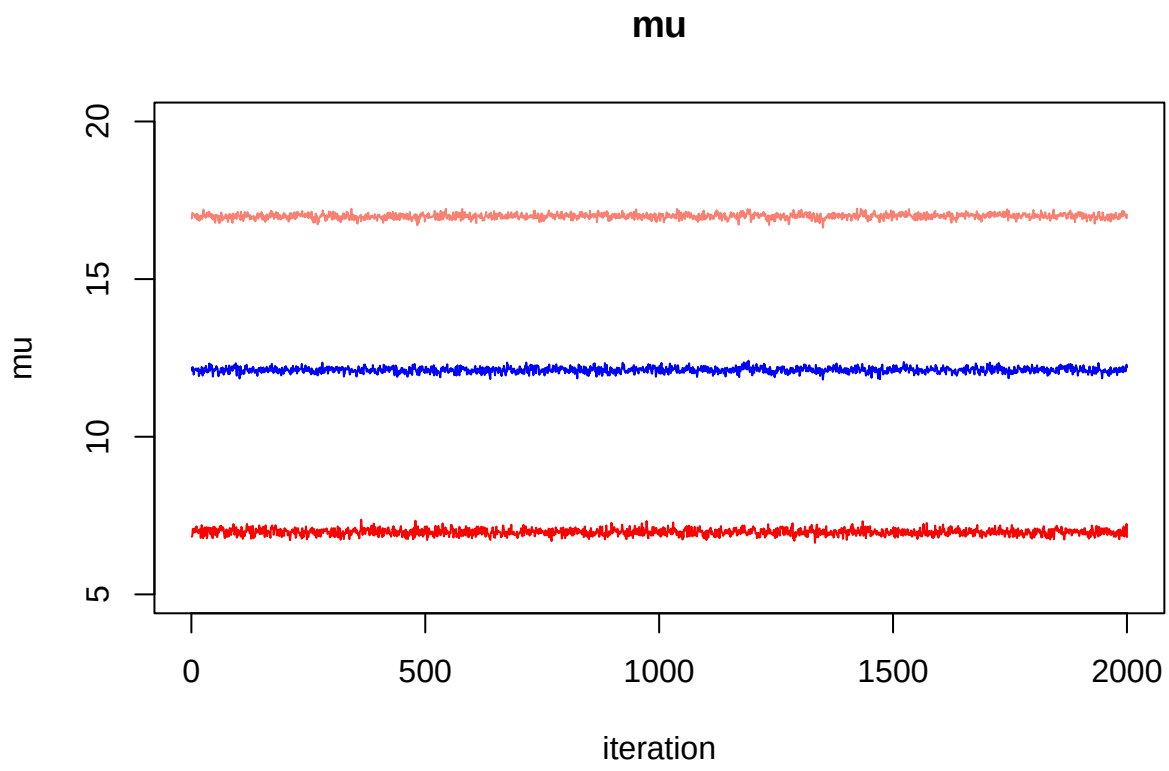
```
apply(test_result$s2_mat[-c(1:1000), ], 2, mean)
```

```
## [1] 1.046775 1.181949 1.058080
```

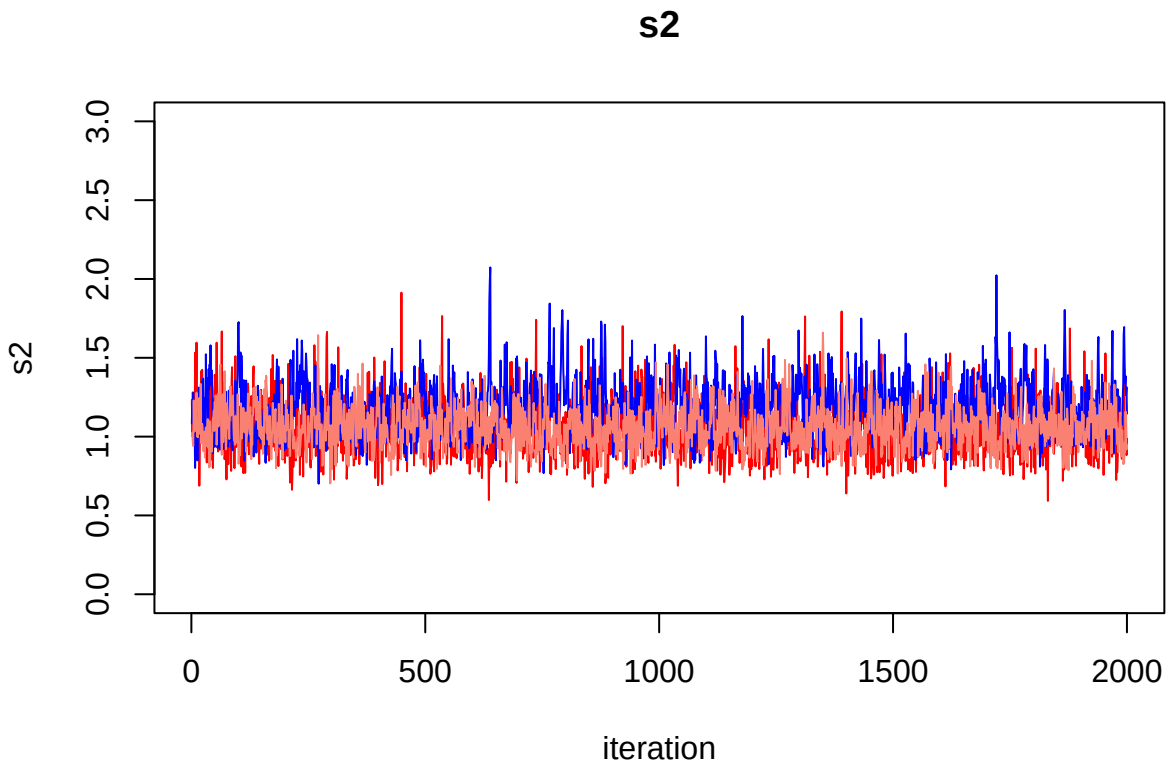
```
apply(test_result$mixing_mat[-c(1:1000), ], 2, mean)
```

```
## [1] 0.2129618 0.3958207 0.3912175
```

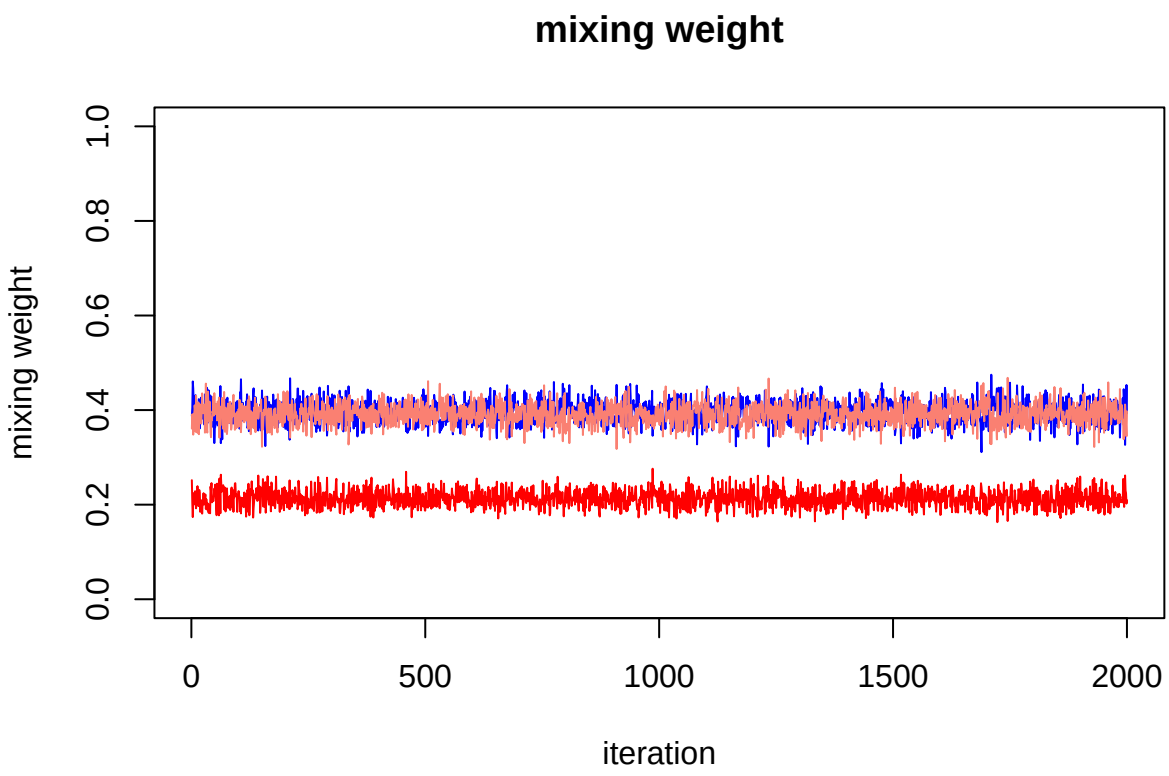
```
plot(test_result$mu_mat[-c(1:1000), 1], type = "l", ylim = c(5, 20),
     col = "red", main = "mu", ylab = "mu", xlab = "iteration")
lines(1:2000, test_result$mu_mat[-c(1:1000), 2], col = "blue")
lines(1:2000, test_result$mu_mat[-c(1:1000), 3], col = "salmon")
```




```
plot(test_result$s2_mat[-c(1:1000), 1], type = "l", ylim = c(0, 3),
     col = "red", main = "s2", ylab = "s2", xlab = "iteration")
lines(1:2000, test_result$s2_mat[-c(1:1000), 2], col = "blue")
lines(1:2000, test_result$s2_mat[-c(1:1000), 3], col = "salmon")
```




```
plot(test_result$mixing_mat[-c(1:1000), 1], type = "l", ylim = c(0, 1),
     col = "red", main = "mixing weight", ylab = "mixing weight",
     xlab = "iteration")
lines(1:2000, test_result$mixing_mat[-c(1:1000), 2], col = "blue")
lines(1:2000, test_result$mixing_mat[-c(1:1000), 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_model_R(iter = 3000, dat = dat_sim, K_max = K,
                             a0 = 1, b0 = 1, mu0 = 0, s20 = 100, xi0 = 1,
                             ci_init = rep(1, N))
  return(list(clus_assign = test_result$assign_mat, ci_true = ci_true))
}
stopImplicitCluster()
```

The result looks fine.

```
jac_vec <- rep(NA, 10)
for(i in 1:10){
  ci_assign <- as.numeric(salso(list_result[[i]]$clus_assign[-c(1:1000), ],
                                maxNClusters = K))
  jac_vec[i] <- mclustcomp(ci_assign, list_result[[i]]$ci_true, "jaccard")$score
}

mean(jac_vec)
```

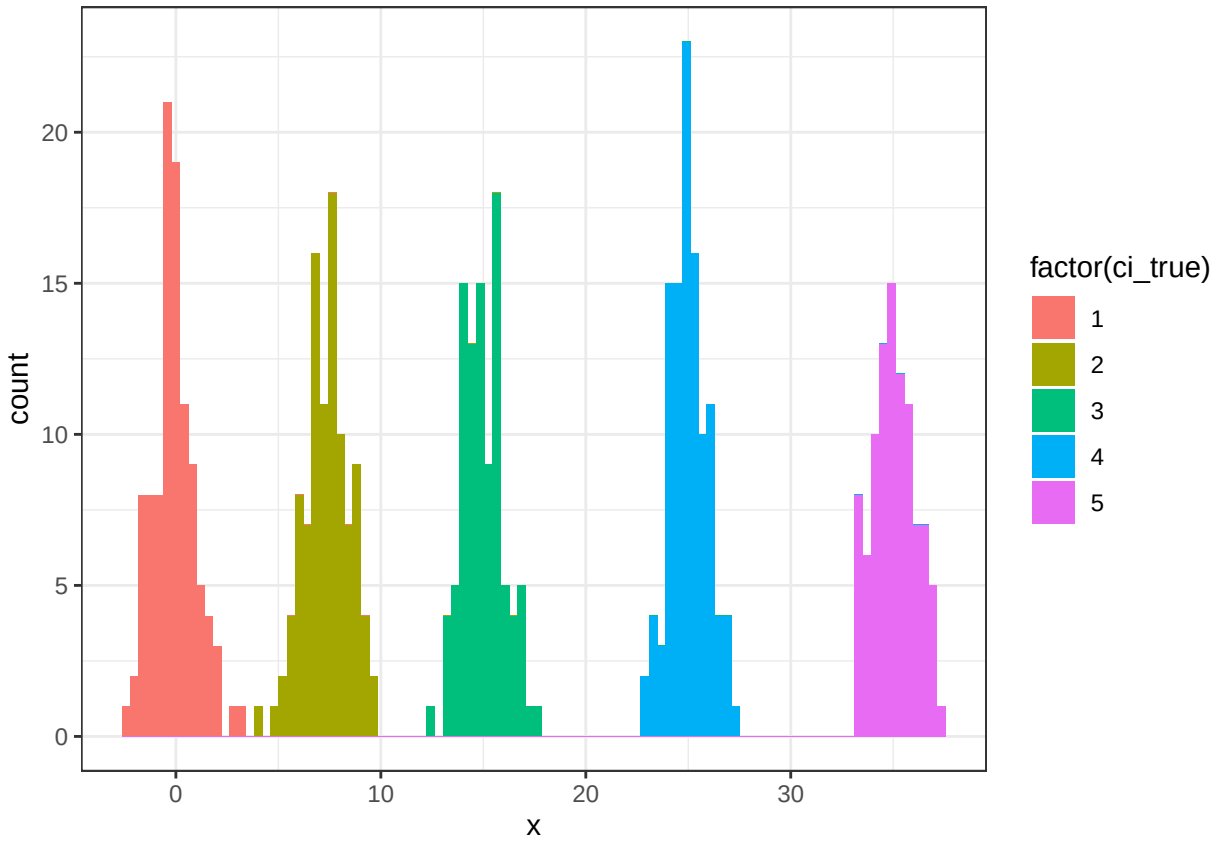
```
## [1] 0.9637099
```

```
sd(jac_vec)
```

```
## [1] 0.01374614
```

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

```
### 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()
```



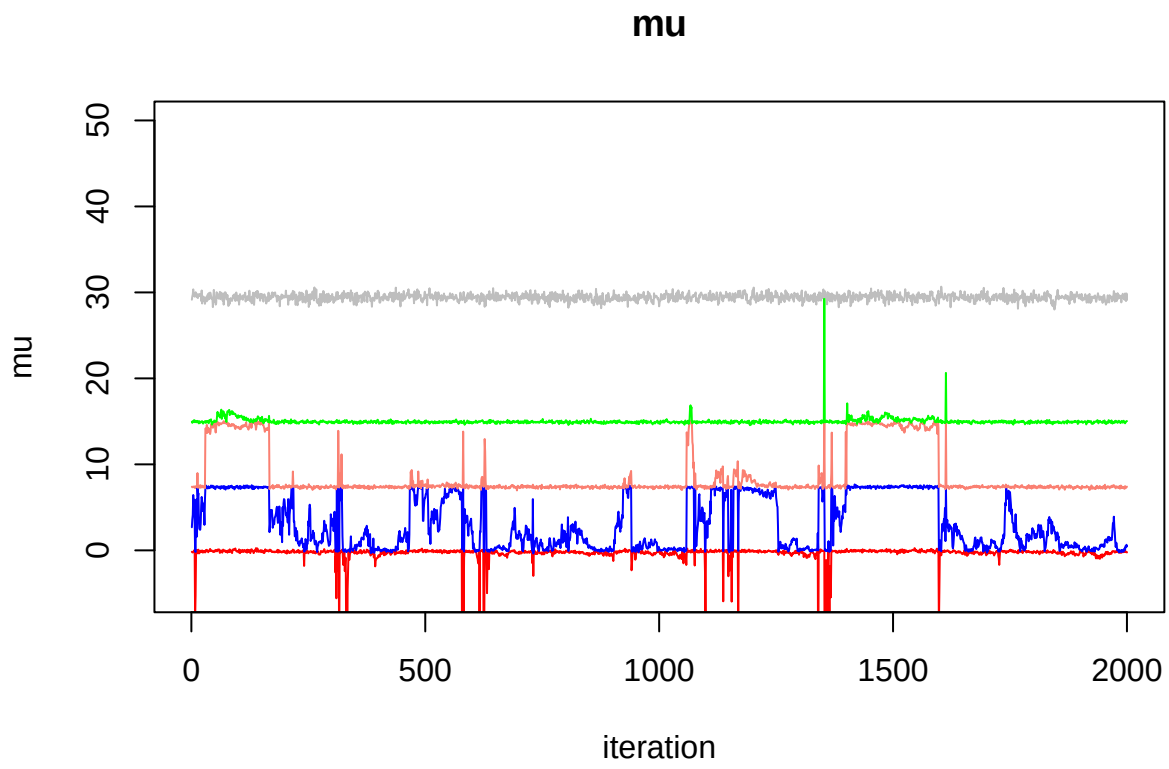

```
### Run the model: (3)
test_result <- fmm_model_R(iter = 3000, dat = dat_sim, K_max = K,
                           a0 = 1, b0 = 1, mu0 = 0, s20 = 100, xi0 = 1,
                           ci_init = rep(1, N))

### salso result: (3)
table(salso(test_result$assign_mat[-c(1:1000), ], maxNClusters = K), ci_true)
```

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

We notice that the performance is not good. So, I decide to take a look at the trace plot.

```
plot(test_result$mu_mat[-c(1:1000), 1], type = "l", ylim = c(-5, 50),
     col = "red", main = "mu", ylab = "mu", xlab = "iteration")
lines(1:2000, test_result$mu_mat[-c(1:1000), 2], col = "blue")
lines(1:2000, test_result$mu_mat[-c(1:1000), 3], col = "salmon")
lines(1:2000, test_result$mu_mat[-c(1:1000), 4], col = "green")
lines(1:2000, test_result$mu_mat[-c(1:1000), 5], col = "grey")
```



The traceplot shows that it does not converge, so I decide to run the model again, but with 5000 iterations and let the first 3000 as a burn-in.

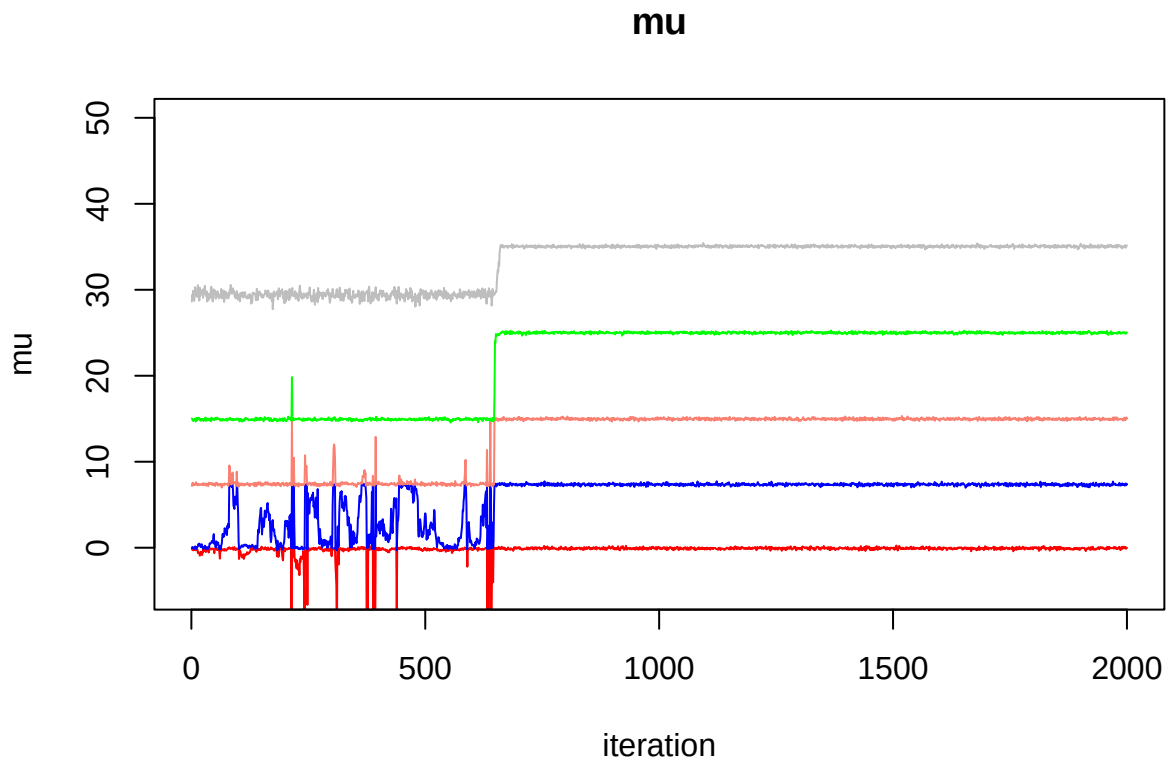
```
set.seed(2345)
### Run the model: (3)
test_result <- fmm_model_R(iter = 5000, dat = dat_sim, K_max = K,
                           a0 = 1, b0 = 1, mu0 = 0, s20 = 100, xi0 = 1,
                           ci_init = rep(1, N))
```

```
### salso result: (3)
table(salso(test_result$assign_mat[-c(1:3000), ], 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
```

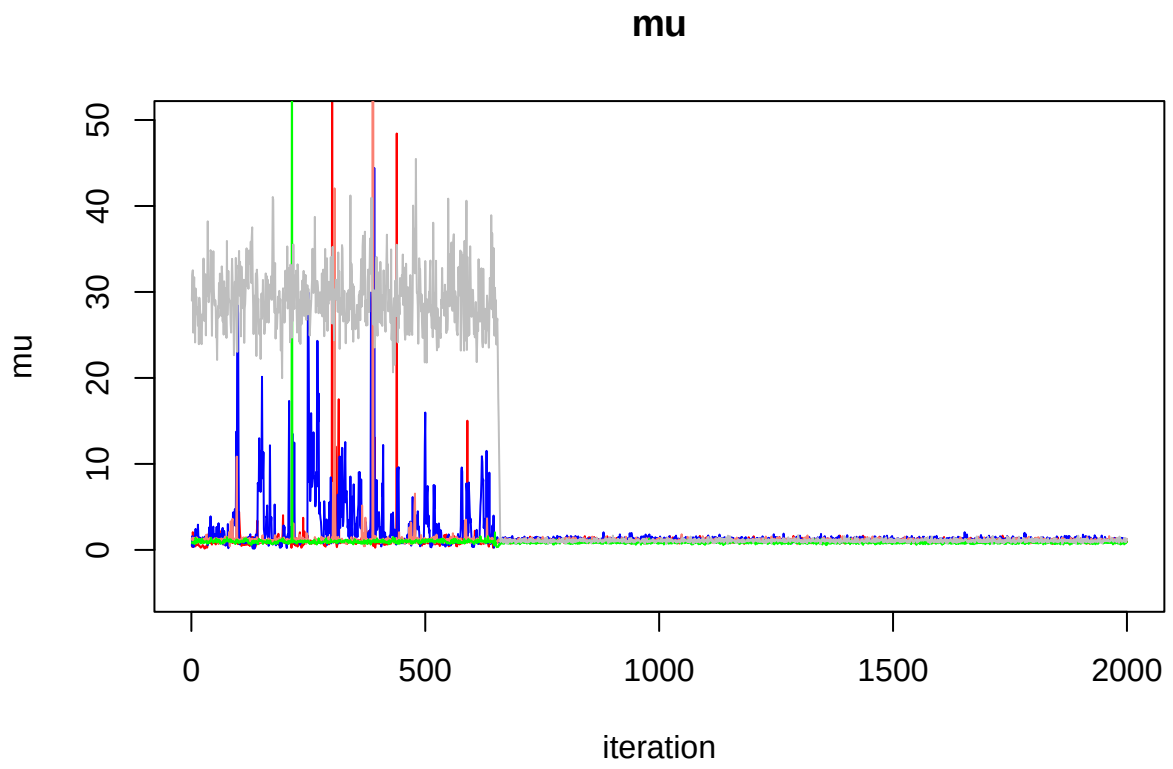
The model performs great.

```
plot(test_result$mu_mat[-c(1:3000), 1], type = "l", ylim = c(-5, 50),
     col = "red", main = "mu", ylab = "mu", xlab = "iteration")
lines(1:2000, test_result$mu_mat[-c(1:3000), 2], col = "blue")
lines(1:2000, test_result$mu_mat[-c(1:3000), 3], col = "salmon")
lines(1:2000, test_result$mu_mat[-c(1:3000), 4], col = "green")
lines(1:2000, test_result$mu_mat[-c(1:3000), 5], col = "grey")
```

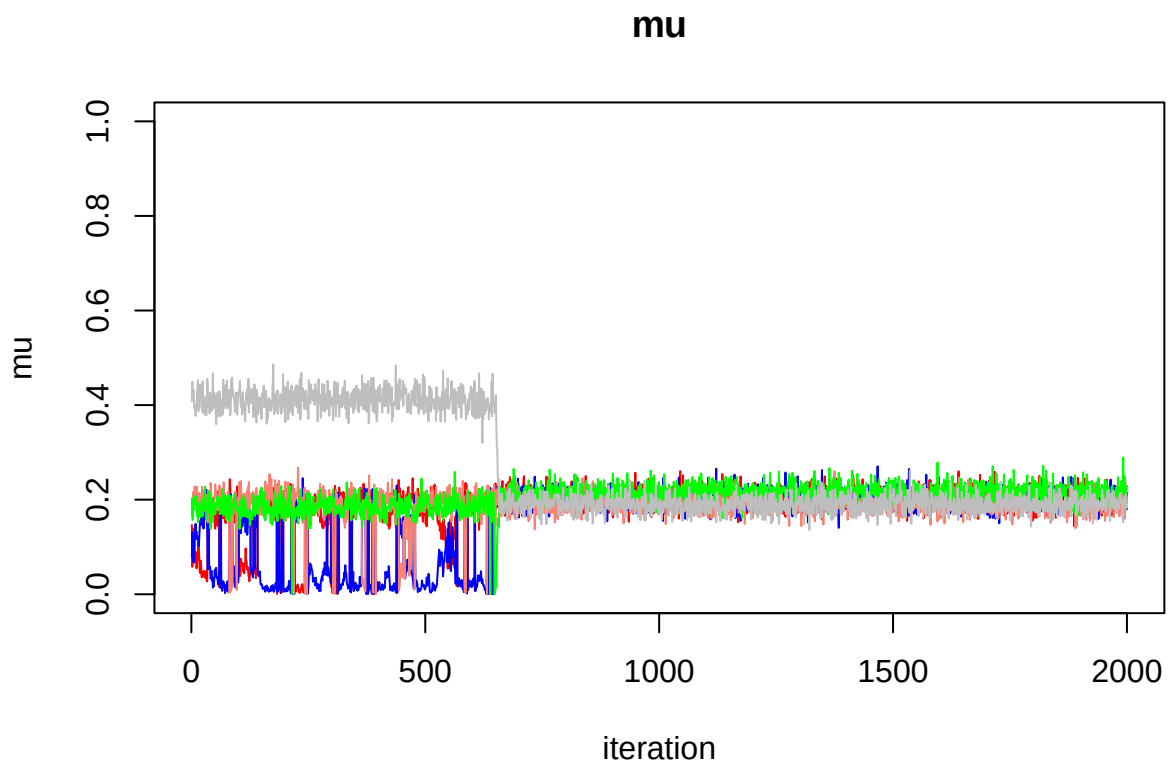



```
plot(test_result$s2_mat[-c(1:3000), 1], type = "l", ylim = c(-5, 50),
     col = "red", main = "mu", ylab = "mu", xlab = "iteration")
lines(1:2000, test_result$s2_mat[-c(1:3000), 2], col = "blue")
lines(1:2000, test_result$s2_mat[-c(1:3000), 3], col = "salmon")
```

```
lines(1:2000, test_result$s2_mat[-c(1:3000), 4], col = "green")
lines(1:2000, test_result$s2_mat[-c(1:3000), 5], col = "grey")
```




```
plot(test_result$mixing_mat[-c(1:3000), 1], type = "l", ylim = c(0, 1),
     col = "red", main = "mu", ylab = "mu", xlab = "iteration")
lines(1:2000, test_result$mixing_mat[-c(1:3000), 2], col = "blue")
lines(1:2000, test_result$mixing_mat[-c(1:3000), 3], col = "salmon")
lines(1:2000, test_result$mixing_mat[-c(1:3000), 4], col = "green")
lines(1:2000, test_result$mixing_mat[-c(1:3000), 5], col = "grey")
```



We might notice that the model took loger than the 3000 iterations to reach the convergence.

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_model_R(iter = 5000, dat = dat_sim, K_max = K,
                             a0 = 1, b0 = 1, mu0 = 0, s20 = 100, xi0 = 1,
                             ci_init = rep(1, N))
  return(list(clus_assign = test_result$assign_mat, ci_true = ci_true))
}
stopImplicitCluster()
```

```
jac_vec <- rep(NA, 10)
for(i in 1:10){
  ci_assign <- as.numeric(salso(list_result[[i]]$clus_assign[-c(1:4000), ],
                                maxNClusters = K))
  jac_vec[i] <- mclustcomp(ci_assign, list_result[[i]]$ci_true, "jaccard")$score
}

mean(jac_vec)
```

```
## [1] 0.9463703
```

```
sd(jac_vec)
```

```
## [1] 0.1132365
```