FMM

2023-05-29

The source code can be found in Github under the 'debugging' branch.

User-Defined functions

```
### Function: Summary Quantities "Mean (SD)".
bal_quan <- function(num_vec, rounding = 4){
  mean_val <- round(mean(num_vec), 4)
  sd_val <- round(sd(num_vec), 4)
  paste0(mean_val, " (", sd_val, ")")
}

### Function: Summary from the result
summary_para <- function(result_list){
  ### Collect the data
  n_cluster_vec <- rep(NA, n_para)
  time_vec <- rep(NA, n_para)
  clus_quality <- matrix(NA, ncol = 3, nrow = n_para)

  for(i in 1:n_para){
    n_cluster_vec[i] <- result_list[[i]]$n_cluster
    time_vec[i] <- result_list[[i]]$time
    clus_quality[i, ] <- result_model[[i]]$clus_measure[c(1, 5, 22), 2]
  }

  data.frame(n_cluster = bal_quan(n_cluster_vec), time = bal_quan(time_vec)) %>%
    data.frame(t(apply(clus_quality, 2, bal_quan))) %>%
    kbl(col.names = c("# cluster", "time", "Adjusted Rand", "Jaccard", "VI"))

}

### Function: Calculate mean and variance
mean_var <- function(num_vec){
  c(mean(num_vec), var(num_vec))
}
```

Overall Settings

I will run the model for 5,000 iterations for all cases while using the first 2,500 iterations as a burn-in. Also, I will run the model for 10 data sets parallel for each case.

```
iter <- 5000
burn_in <- 2500
overall_seed <- 31807
n_para <- 10
```

Part I: EM and DP Algorithm

```
set.seed(overall_seed)
ci_true <- sample(1:5, 500, replace = TRUE)
dat <- rnorm(500, c(-20, -10, 0, 10, 20)[ci_true])

k <- 5
mu_init <- rep(0, k)
sigma_init <- sqrt(1/rgamma(k, 1, 1))
EM_alg <- normalmixEM(scale(dat), mu = mu_init, sigma = sigma_init, maxit = 5000)
```

```
## number of iterations= 1422
```

```
EM_result <-t(apply(EM_alg$posterior, 1, rmultinom, n = 1, size = 1))
table(apply(EM_result, 1, which.max), ci_true)
```

```
##    ci_true
##       1   2   3   4   5
##   1   0   0   0   0  99
##   2 102 101   0   0   0
##   3   0   0   0 100   0
##   4   0   0  85   0   0
##   5   0   0  13   0   0
```

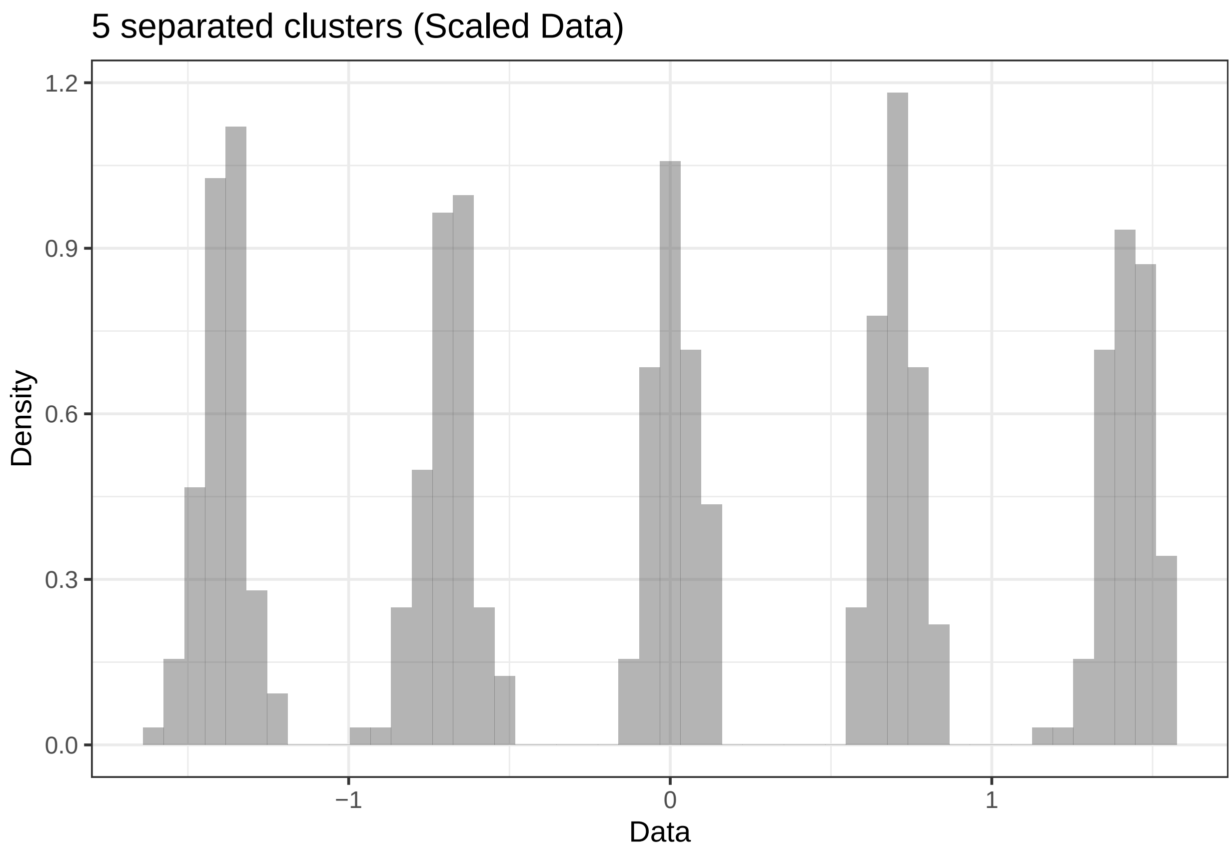
```
set.seed(overall_seed)
registerDoParallel(detectCores() - 1)
overall_start <- Sys.time()
result_model <- foreach(i = 1:n_para) %dorng%{

  ### Data Simulation
  ci_true <- sample(1:2, 500, replace = TRUE)
  dat <- rnorm(500, c(-5, 5)[ci_true])

  ### Run the model
  K_max <- 5
  start_time <- Sys.time()
  mu_init <- rep(0, K_max)
  sigma_init <- sqrt(1/rgamma(K_max, 1, 1))
  EM_alg <- normalmixEM(scale(dat), mu = mu_init, sigma = sigma_init, maxit = 5000)
  EM_result <-t(apply(EM_alg$posterior, 1, rmultinom, n = 1, size = 1))
  total_time <- difftime(Sys.time(), start_time, units = "secs")

  clus_assign <- apply(EM_result, 1, which.max)

  return(list(time = as.numeric(total_time),
```

```
              clus_assign = clus_assign, ci_true = ci_true,
              n_cluster = length(unique(clus_assign)),
              clus_measure = mclustcomp(clus_assign, ci_true)))
}
stopImplicitCluster()
```

# cluster	time	Adjusted Rand	Jaccard	VI
5 (0)	0.8431 (0.5442)	0.6523 (0.2316)	0.6708 (0.1749)	0.9775 (0.785)

Part II: 5 Separated Clusters

Below is the plot for the standardized data for five separated clusters.



I will change the value for ξ ($\xi = 1, 0.1, 0.01, 0.001$) while keeping the other variables to be fixed. ($\mu = 0, a_\sigma = b_\sigma = \lambda = 1, K_{\max} = 10$)

$\xi = 1$

# cluster	time	Adjusted Rand	Jaccard	VI
2.7 (0.6749)	22.118 (1.5423)	0.535 (0.1636)	0.5055 (0.1277)	0.9763 (0.3579)

$\xi = 0.1$

# cluster	time	Adjusted Rand	Jaccard	VI
3 (0.6667)	19.0102 (1.2064)	0.6163 (0.1493)	0.5686 (0.1235)	0.8083 (0.3307)

$\xi = 0.01$

# cluster	time	Adjusted Rand	Jaccard	VI
2.9 (0.3162)	17.6851 (1.0259)	0.6013 (0.0804)	0.5474 (0.0574)	0.8383 (0.1781)

$\xi = 0.001$

# cluster	time	Adjusted Rand	Jaccard	VI
2.9 (0.3162)	17.1974 (1.0663)	0.6013 (0.0804)	0.5474 (0.0574)	0.8383 (0.1781)