```r
###########################################################################
###########################################################################
######################### Mixture of normals ##############################
###########################################################################
###########################################################################


###### Clear environment and load libraries
rm(list = ls())
library(coda)
library(DirichletReg)

################# Gibbs Sampler Function #################
fit_mixture <- function(Y,K,n_iter,burn_in,thin){
  #Y is the data
  #K in the number of clusters
  #n_iter is the number of iterations
  #burn_in is the number of burn-in

  ###Data summaries
  n <- nrow(Y)

  ###Hyperparameters for the priors
  mu_0 <- 0
  gamma_0_sq <- 100
  nu_0 <- 2
  sigma_0_sq <- 100
  alpha <- rep(1,K)

  ###Initialize
  mu <- rep(0,K)
  #mu <- seq(-3,3,length.out=K)*sd(Y) + mean(Y); mu <- mu[order(mu,decreasing=F)]
  sigma_sq <- var(Y)
  lambda <- rep(1/K,K)
  post_prob_z <- matrix(0,n,K)

  ###Set null matrices to save samples
  MU <- matrix(0,nrow=n_iter,ncol=K)
  SIGMA_SQ <- matrix(nrow=n_iter, ncol=1)
  Z_MAT <- matrix(0,nrow=n_iter,ncol=n)
  LAMBDA <- matrix(0,nrow=n_iter,ncol=K)

  ###Start Gibbs
  iter_seq <- seq(1,(n_iter+burn_in),by=500)
  for(s in 1:(n_iter+burn_in)){


    #Update z_i, the mixture i.d. for each observation
    for(k in 1:K){
      post_prob_z[,k] <- lambda[k]*dnorm(Y,mu[k],sqrt(sigma_sq))
    }
    post_prob_z <- post_prob_z/matrix(rowSums(post_prob_z),nrow=n,ncol=K)
    Ran_unif_z <- runif(nrow(post_prob_z))
    cumul_z <- post_prob_z%*%upper.tri(diag(ncol(post_prob_z)),diag=TRUE)
    Z <- rowSums(Ran_unif_z>cumul_z) + 1L
    #a slightly efficient way to sample z_1...z_n at once.


    #Update mu for each mixture component
    n_k <- c(table(factor(Z,levels=c(1:K))))
    for(k in 1:K){
      if(n_k[k]!=0){
        y_bar_k <- mean(Y[which(Z==k),])
        gamma_k_n_sq <- 1/(n_k[k]/sigma_sq + 1/gamma_0_sq)
        mu_k_n <- gamma_k_n_sq*(n_k[k]*y_bar_k/sigma_sq + mu_0/gamma_0_sq)
        mu[k] <- rnorm(1,mu_k_n,sqrt(gamma_k_n_sq))
      }
    }
    #mu <- mu[order(mu,decreasing=F)]
    #Ad-hoc trick to help with label switching


    #Update sigma^2
    nu_n <- nu_0 + n
    nu_n_sigma_n_sq <- nu_0*sigma_0_sq + sum((Y-mu[Z])^2)
    sigma_sq <- 1/rgamma(1,nu_n/2,nu_n_sigma_n_sq/2)


    #Sample lambda
    lambda <- rdirichlet(1, (alpha + n_k))


    #Print iteration info
    if(sum(s==iter_seq)==1){
      cat("Iteration =", s,"\t"," ",
          "K =",formatC(length(unique(Z)), width=2, flag=" "),"\t",
          "mu =",mu,"\t",
          "lambda =",lambda,"\n",sep = " ")
    } else if(s==(n_iter+burn_in)){
      cat("Iteration =", s,"\t"," ",
          "K =",formatC(length(unique(Z)), width=2, flag=" "),"\t",
          "mu =",mu,"\t",
          "lambda =",lambda,"\n",sep = " ")
    }


    #Save results
    if(s > burn_in){
      MU[(s-burn_in),] <- mu
      SIGMA_SQ[(s-burn_in)] <- sigma_sq
      Z_MAT[(s-burn_in),] <- Z
      LAMBDA[(s-burn_in),] <- lambda
    }
  }


  #Return results
  list(MU=MU,SIGMA_SQ=SIGMA_SQ,Z=Z_MAT,LAMBDA=LAMBDA)
}
################# End of Gibbs Sampler Function #################


################# Fit to simulated data #################
#First generate data
set.seed(1234)
Z_true <- sample(1:3,500,replace=T,prob=c(0.55,0.30,0.15))
mu_true <- c(-5,3,9)
Y <- matrix(rnorm(length(Z_true),mu_true[Z_true],2.5),ncol=1)
plot(density(Y))

#Now fit model back to simulated data
K <- 3
n_iter <- 10000
burn_in <- 0.3*n_iter
thin <- 1
model <- fit_mixture(Y,K,n_iter,burn_in,thin)


#MCMC summary and diagnostics
plot(mcmc(model$SIGMA_SQ))
plot(mcmc(model$MU))

plot(model$MU[,1],type="l",ylim=c(-10,15),col="red4")
lines(model$MU[,2],col="blue4")
lines(model$MU[,3],col="green4")
legend("topright", c('Cluster 1','Cluster 2','Cluster 3'), lwd=2, lty=1.5,
       col=c('red4',"blue4","green4"))

plot(model$LAMBDA[,1],type="l",ylim=c(0,1),col="red4")
lines(model$LAMBDA[,2],col="blue4")
lines(model$LAMBDA[,3],col="green4")
legend("topright", c('Cluster 1','Cluster 2','Cluster 3'), lwd=2, lty=1.5,
       col=c('red4',"blue4","green4"))

mean(model$LAMBDA[,1])
mean(model$LAMBDA[,2])
mean(model$LAMBDA[,3])
```