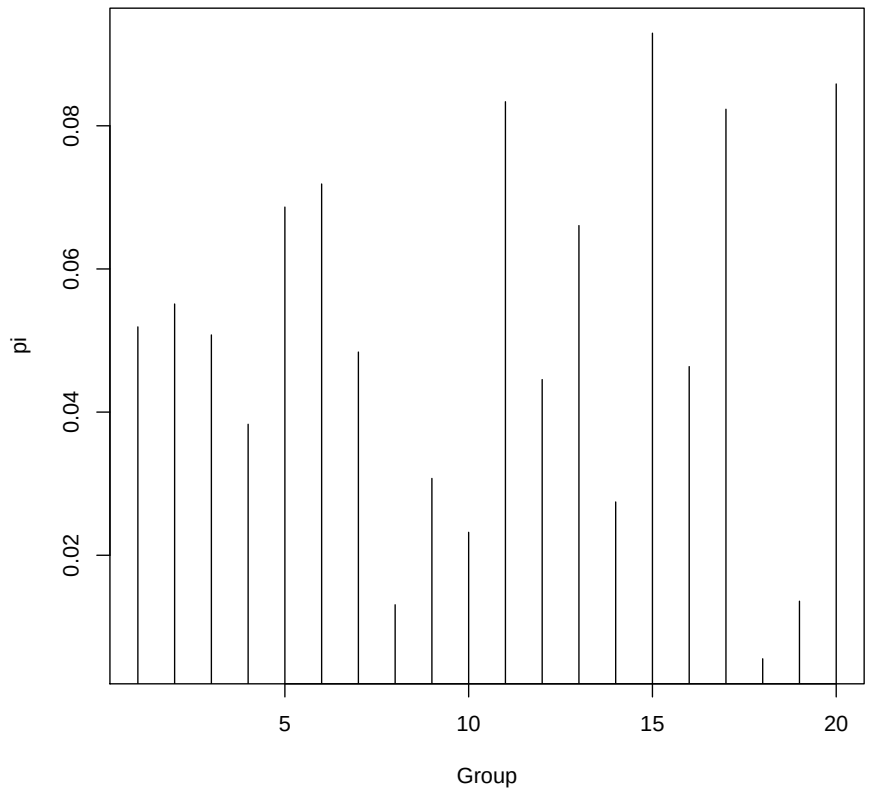
```r
n <- 500
y<- as.numeric()
# Record the group that each point belongs to
ind<- sample(c(1,2,3),n,prob=c(0.1,0.5,0.4),replace=TRUE)
for(i in 1:n){
  if(ind[i]==1)
y[i] <- rnorm(1,mean=-1,sd=0.2)
  if(ind[i]==2)
y[i] <- rnorm(1,mean=0,sd=0.1)
 if(ind[i]==3)
y[i] <- rnorm(1,mean=1,sd=0.4)}
```

```r
 den_y <- density(y)
x <- seq(-2,2,by=0.01)
den_real <- 0.1*dnorm(x,mean=-1,sd=0.2)+0.5*dnorm(x,mean=0,sd=0.1)+0.4*dnorm(x,mean=1,sd=0.4
```

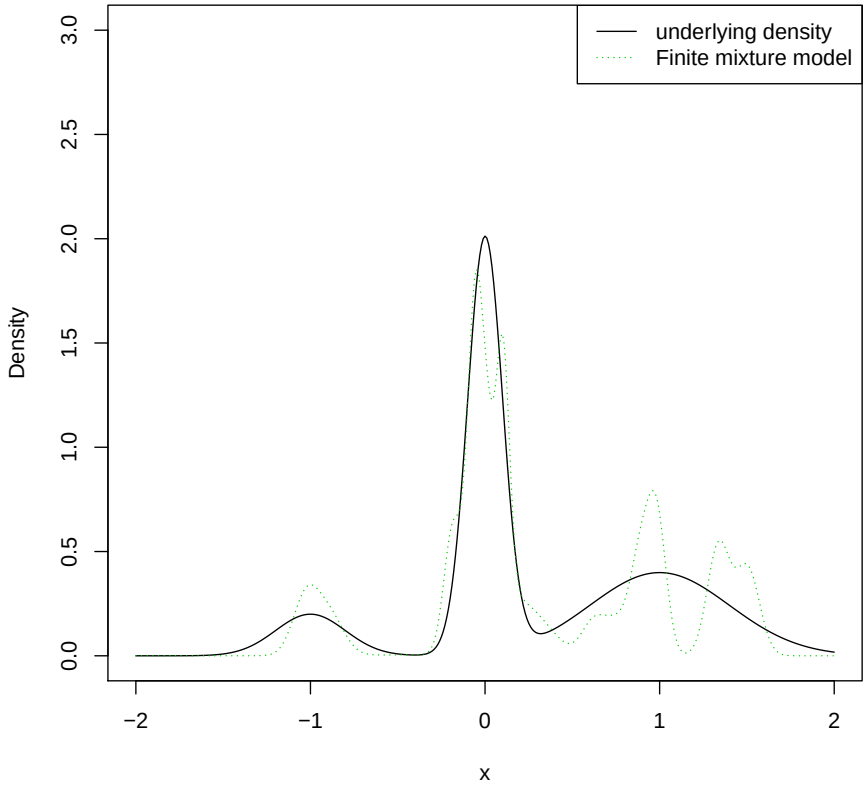
Finite Mixtur model

```r
k <- 20
#Number of mixtuer groups
alpha <- 1
mu0 <- 0
a <- alpha/k
kappa <- tau_a <- tau_b <- 1
#Initialization of the iteration
#Initialize the weight by assigning equal weights to each mixture
pi <- rep(1/k,k)

#Generate mean and variance for each cluster
tau <- 1/rgamma(k, shape=tau_a,rate=tau_b)
vari<-tau*kappa
mu <- rnorm(k,mean=mu0,sd=sqrt(vari))

#2000 iterations
num <- 2000
for(i in 1:num){
  #1. update the mixture group each data point y_i belongs to
  z<-as.numeric()
  #Records the cluster of the data point
  for(j in 1:n){
    w <- pi*dnorm(y[j],mean=mu,sd=sqrt(vari))/(sum(pi*dnorm(y[j],mean=mu,sd=sqrt(vari))))
    z[j] <- sample(1:k, size=1, prob=w) }

  #2. update mu and tau for each mixture (from 1 to k)
  nk <- table(z)
```

```
  #Record the size of each cluster
  empty <- k-length(nk)
  #Record the number of empty clusters
  nk <- c(nk, rep(0, empty))
  kappa_hat <- 1/(1/kappa+nk)
  y_bar <-c(unlist(lapply(split(y,z),mean)), rep(0,empty))
  mu_hat <- kappa_hat*(mu0/kappa+nk*y_bar)
  alpha_hat <- tau_a+nk/2
  cy <- (unlist(split(y,z))-rep(y_bar,nk))^2

  #Group variance, the clusters with size 0 will be assigned with variance 0
  cvar <- c(unlist(lapply(split(cy,rep(c(1:k),nk)),sum)),rep(0,empty))
  beta_hat <- tau_b+1/2*(cvar+(nk/(1+kappa*nk)*(y_bar-mu0)^2))

  #Generating new mu and tau
  tau <- 1/rgamma(k,shape=alpha_hat, rate=beta_hat)
  vari <- tau*kappa_hat
  mu <- rnorm(k,mean=mu_hat, sd=sqrt(vari))

  #3. update pi by generating new pi value from new dirichlet distribution
  cw <- rgamma(k,shape=a+nk,rate=1)
  pi <- cw/(sum(cw)) }


x_hat1 <- as.numeric()
for(i in 1:length(x))
x_hat1[i] <- sum(pi*dnorm(x[i],mean=mu,sd=sqrt(vari)))
plot(c(1:20),pi,type="h",xlab="Group")
```



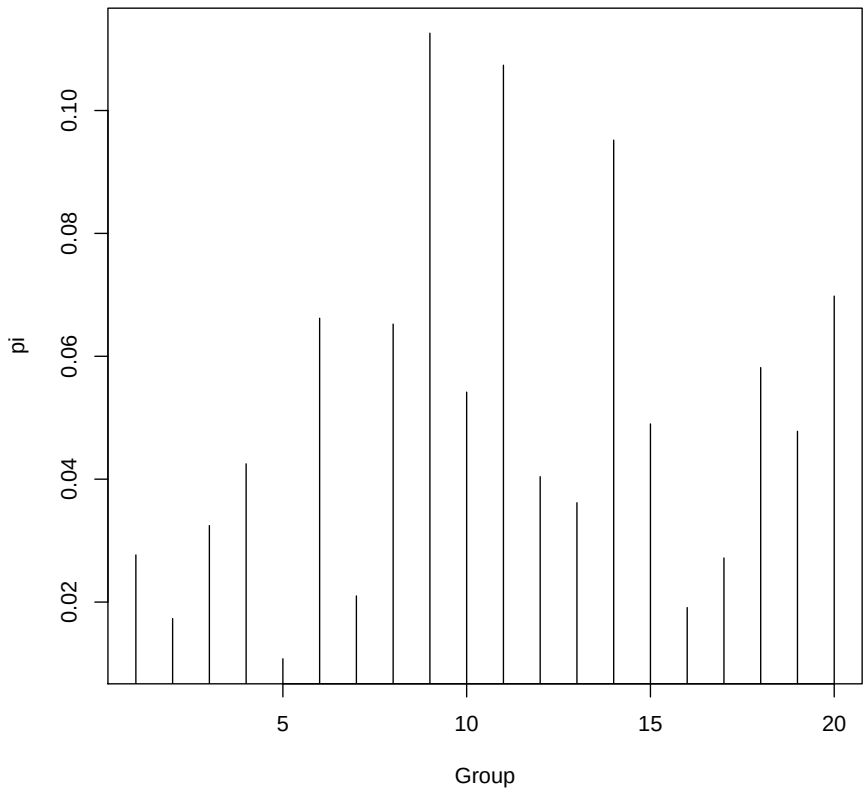

```
plot(x,den_real,type="l",ylim=c(0,3),ylab="Density")
lines(x,x_hat1,col=3,lty=3)
legend("topright",c("underlying density","Finite mixture model"),col=c(1,3),lty=c(1,3))
```



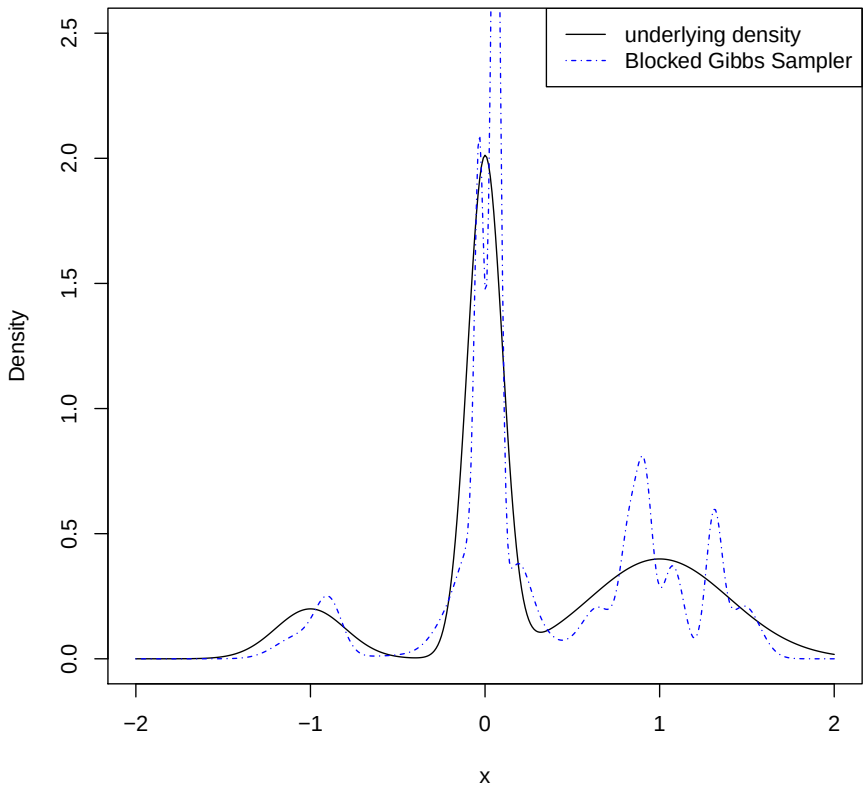
(d) Blocked Gibbs sampler

```r
#Maximum number of clusters
N <- 20

#Stick-breaking process
weight <- rbeta(N-1,shape1=1,shape2=alpha)
pi <- c(weight[1],weight[2:(N-1)]*cumprod((1-weight[1:(N-2)])))
pi <- c(pi, 1-sum(pi))

#Initialize the parameter for each group by generating it using prior dist.
tau <- 1/rgamma(N, shape=tau_a,rate=tau_b)
vari<-tau*kappa
mu <- rnorm(N,mean=mu0,sd=sqrt(vari))

#2000 iterations
num <- 2000
```

```r
for(i in 1:num){
  #1. update the mixture group
  z<-as.numeric()
  for(j in 1:n){
    w <- pi*dnorm(y[j],mean=mu,sd=sqrt(vari))/(sum(pi*dnorm(y[j],mean=mu,sd=sqrt(vari))))
    z[j] <- sample(1:N, size=1, prob=w) }

  #2. Update stick-breaking weight
  nk <-table(z)
  empty <- k-length(nk)
  #Record the number of empty clusters
  nk <- c(nk, rep(0, empty))
  shape1 <- 1+nk[1:N-1]
  shape2 <- alpha+length(y)-cumsum(nk[1:N-1])
  weight <- rbeta(N-1, shape1=shape1, shape2=shape2)
  pi <- c(weight[1],weight[2:(N-1)]*cumprod((1-weight[1:(N-2)])))
  pi <- c(pi, 1-sum(pi))

  #3. update mu and tau
  kappa_hat <- 1/(1/kappa+nk)
  y_bar <-c(unlist(lapply(split(y,z),mean)), rep(0,empty))
  mu_hat <- kappa_hat*(mu0/kappa+nk*y_bar)
  alpha_hat <- tau_a+nk/2
  cy <- (unlist(split(y,z))-rep(y_bar,nk))^2
  cvar <- c(unlist(lapply(split(cy,rep(c(1:N),nk)),sum)),rep(0,empty)) #Group variance
  beta_hat <- tau_b+1/2*(cvar+(nk/(1+kappa*nk)*(y_bar-mu0)^2))
  tau <- 1/rgamma(N,shape=alpha_hat, rate=beta_hat)
  vari <- tau*kappa_hat
  mu <- rnorm(k,mean=mu_hat, sd=sqrt(vari))
}

x_hat2 <- as.numeric()
for(i in 1:length(x))
x_hat2[i] <- sum(pi*dnorm(x[i],mean=mu,sd=sqrt(vari)))
plot(c(1:20),pi,type="h",xlab="Group")
```




```
plot(x,den_real,type="l",ylim=c(0,2.5),ylab="Density")
lines(x,x_hat2,col=4,lty=4)
legend("topright",c("underlying density","Blocked Gibbs Sampler"),col=c(1,4),lty=c(1,4))
```



Dirichlet process mixtures Initialization:
the number of clusters and param- eters for each cluster

```
num <- 1 #number of clusters
S <- as.numeric()
#Record the cluster number that the data belongs
#Random generate the mean and variance for the first data point
tau <- as.numeric()
vari <- as.numeric()
mu <- as.numeric()
tau[1] <- 1/rgamma(1, shape=tau_a,rate=tau_b)
vari[1] <- tau[1]*kappa
mu[1] <- rnorm(1,mean=mu0,sd=sqrt(vari[1]))
S[1]<-1
#Assign the first data point to the first cluster group

for(i in 2:length(y)){
```

```
#Records the size of each cluster
nk <- apply(outer(c(1:num),S,"=="),1,sum)
#Weight for existing groups
weight <- sum(1/(alpha+i-1)*nk)
Is_new <- c(weight,1-weight)
new <- sample(0:1,size=1,prob=Is_new)
if(new ==1){
  num <- num+1
  S[i] <- num
  tau_new <- 1/rgamma(1, shape=tau_a,rate=tau_b)
  vari_new <- tau_new*kappa
  mu_new <- rnorm(1,mean=mu0,sd=sqrt(vari_new))
  tau <- c(tau,tau_new)
  vari <- c(vari, vari_new)
  mu <- c(mu, mu_new)
}
if(new == 0 ){
  pi <- nk/(i-1)
  w <- pi*dnorm(y[i],mean=mu,sd=sqrt(vari))/(sum(pi*dnorm(y[i],mean=mu,sd=sqrt(vari))))
  S[i] <- sample(1:num, size=1, prob=w)
} }
```

Iteration

```
niter <- 2000 #2000 iterations

iter <- 1
while(iter <= niter){
#1. Update the allocation of clusters
for(i in 1:length(y)){

  #Collect the number of existing clusters
  exist <- unique(S)[order(unique(S))]
  #num records the number of clusters in each iteration
  num <- length(exist)
  #Rename the clusters form 1 (erase the empty clusters)
  S <- outer(S, exist,"==")%*%c(1:num)
  S <- as.vector(S)
  #nk records the number of data points in each group
  nk <- table(S)
  weight<-alpha/(alpha+length(y)-1)
  Is_new <- c(1-weight,weight)
  #Generating a new cluster with possiblility
  alpha/(alpha+n-1)
  new <- sample(0:1,size=1,prob=Is_new)
```

```
  if(new == 1){
  #Adding a new cluster only if the cluster choosen has size greater than 1
    if(nk[S[i]]!=1){
      #Assign the data point to a new cluster
      S[i] <- num+1
      tau_new <- 1/rgamma(1, shape=tau_a,rate=tau_b)
      vari_new <- tau_new*kappa
      mu_new <- rnorm(1,mean=mu0,sd=sqrt(vari_new))
      tau <- c(tau,tau_new)
      vari <- c(vari,vari_new)
      mu <- c(mu, mu_new)}}
  if(new == 0){
    nk[S[i]] <- nk[S[i]]-1
    #Record the previous cluster of point i
    prev <- S[i]
    pi <- nk/(length(y)-1)
    w <- pi*dnorm(y[i],mean=mu,sd=sqrt(vari))/(sum(pi*dnorm(y[i],mean=mu,sd=sqrt(vari))))
    S[i] <- sample(1:num, size=1, prob=w)
    if(nk[prev]==0){
      #Empty the respect parameters if the previous cluster is empty after change
      tau <- tau[-prev]
      vari <- vari[-prev]
      mu <- mu[-prev]}}}
  #Unpdate the parameter within each cluster
  nk <- table(S)
  num <- length(nk)

  kappa_hat <- 1/(1/kappa+nk)
  y_bar <-unlist(lapply(split(y,S),mean))
  mu_hat <- kappa_hat*(mu0/kappa+nk*y_bar)
  alpha_hat <- tau_a+nk/2
  cy <- (unlist(split(y,S))-rep(y_bar,nk))^2
  cvar <- c(unlist(lapply(split(cy,rep(c(1:num),nk)),sum))) #Group variance
  beta_hat <- tau_b+1/2*(cvar+(nk/(1+kappa*nk)*(y_bar-mu0)^2))

  tau <- 1/rgamma(num,shape=alpha_hat, rate=beta_hat)
  vari <- tau*kappa_hat
  mu <- rnorm(num,mean=mu_hat, sd=sqrt(vari))
  iter <- iter+1
}

x_hat3 <- as.numeric()
for(i in 1:length(x))
x_hat3[i] <- sum(pi*dnorm(x[i],mean=mu,sd=sqrt(vari)))
plot(x,den_real,type="l",ylim=range(x_hat3),ylab="Density")
```

```
lines(x,x_hat3,col=5,lty=5)
legend("topright",c("density approximation","Dirichlet Process Mixtures"),col=c(1,5),lty=c(1
```