

Rmd_EsempioPartizioneFissa

Partizione fissa

```
library(GDFMM)
library(ACutils)
```

data generation

Genero i dati per 3 gruppi

```
d = 3          # number of groups
K = 3          # number of global clusters
mu = c(-20,0,20) # vectors of means
sd = c(1,1,1)   # vector of sd
n_j = rep(200, d) # set cardinality of the groups
p = matrix(0, nrow = d, ncol = K) # matrix with components weights

set.seed(124123)
Kgruppo = c()
componenti_gruppo = NULL
data = matrix(NA, nrow = d, ncol = max(n_j)) # d x max(n_j) matrix
cluster = matrix(NA, nrow = d, ncol = max(n_j)) # d x max(n_j) matrix
real_partition = c() # real_partition is a vector of length sum(n_j), it collects all the group means
# values are collected level by level, so first all the values in level 1, then level 2, etc.
# cluster label must always start from 0!

for(j in 1:d){
  Kgruppo[j] = sample(1:K,1) # number of clusters in each level
  componenti_gruppo[[j]] = sample(1:K,Kgruppo[j], replace = F) # choose the components
  p[j,1:Kgruppo[j]] = rep(1/Kgruppo[j], Kgruppo[j]) # set the weights all equals
  appoggio = genera_mix_gas(n = n_j[j], pro = p[j,1:Kgruppo[j]], means = mu[ componenti_gruppo[[j]] ],
                             sds = sd[ componenti_gruppo[[j]] ])

  data[j, 1:n_j[j]] = appoggio$y
  #cluster[j, 1:n_j[j]] = appoggio$clu, #errore, genera_mix_gas usa sempre indici che partono da 1!
  cluster[j, 1:n_j[j]] = unlist(lapply(1:n_j[j], function(h){componenti_gruppo[[j]][appoggio$clu[h]]}))
  real_partition = c(real_partition, cluster[j, 1:n_j[j]])
}
```

La partizione generata così non rispecchia i criteri del sampler. Serve che parta da 0 e che abbia valori contigui. Quindi deve essere 0,1,2,... e non cose tipo 1,2,3... oppure 0,1,3,5,6... Ho fatto una funzione per sistemare questa cosa, si chiama `arrange_partition`. Non la faccio vedere, viene chiamata automaticamente nel sampler.

guardo i dati

Spero che sia il seed giusto, perché se non ci sono formati 3 cluster allocati, può essere che ci siano dei nan

```

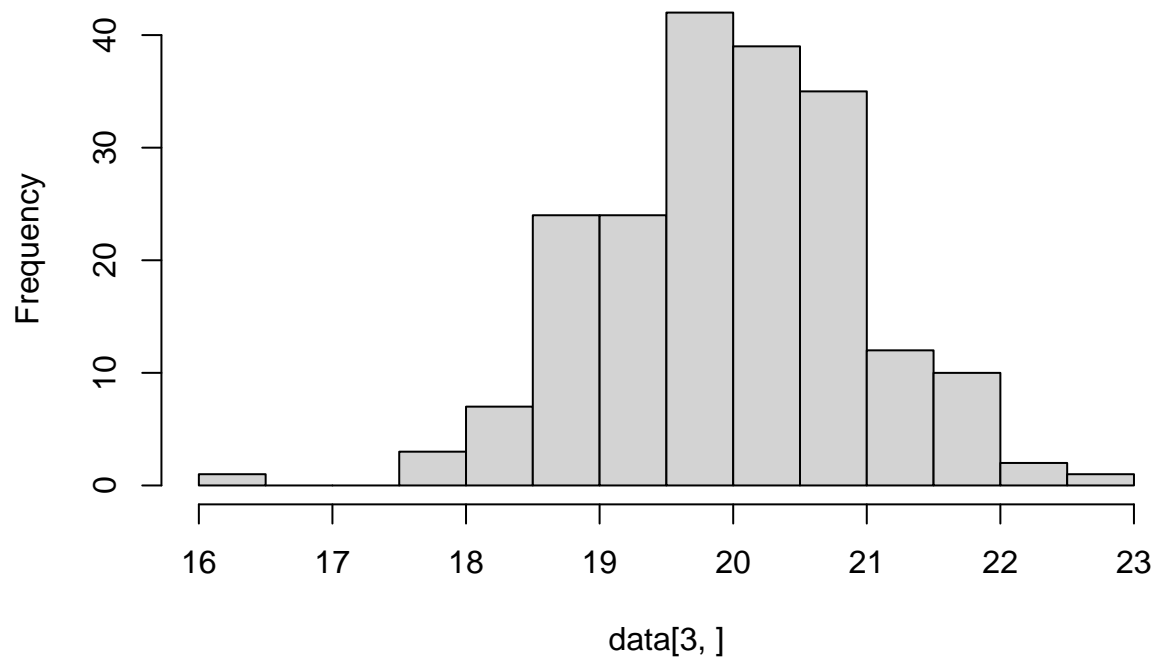
N_m = table(real_partition)

x11();hist(data[1,])
x11();hist(data[2,])
x11();hist(data[3,])

data_level1 = data[cluster==1]

```

Histogram of data[3,]



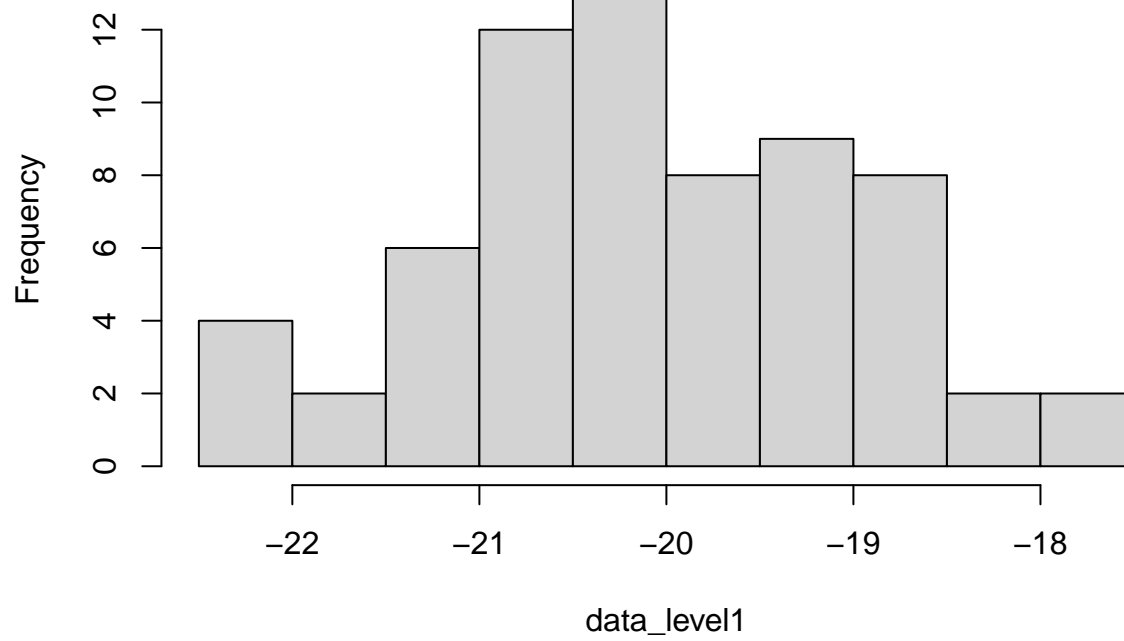
```

data_level2 = data[cluster==2]
data_level3 = data[cluster==3]

mean1 = mean(data_level1); var1 = var(data_level1); x11(); hist(data_level1); N_m1 = length(data_level1)
mean2 = mean(data_level2); var2 = var(data_level2); x11(); hist(data_level2); N_m2 = length(data_level2)

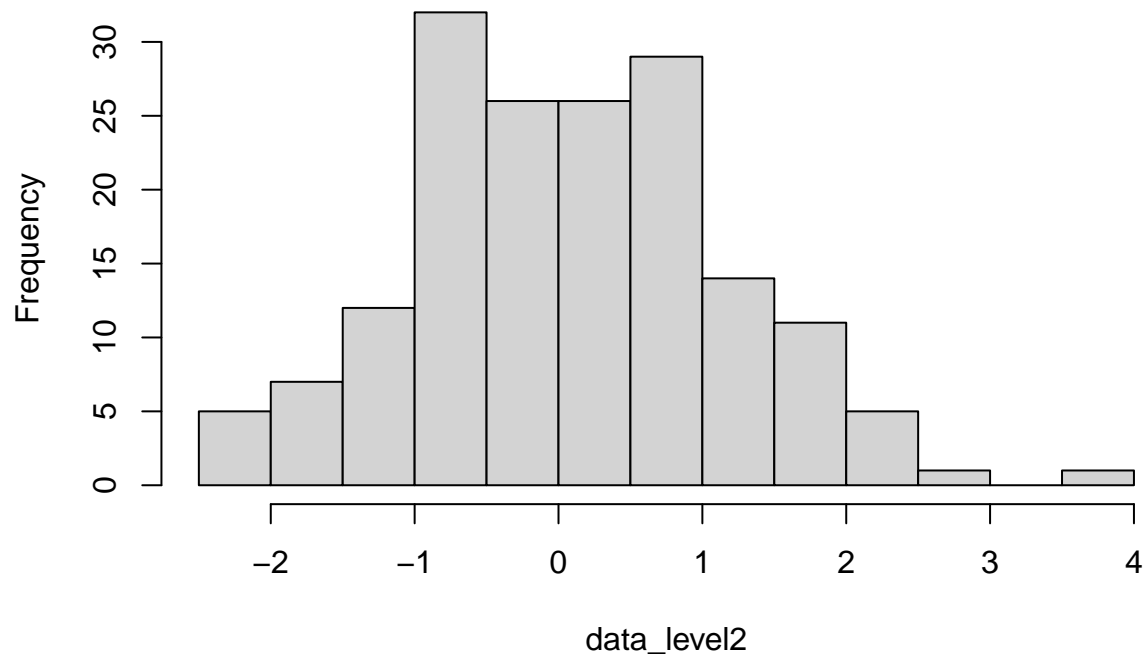
```

Histogram of data_level1



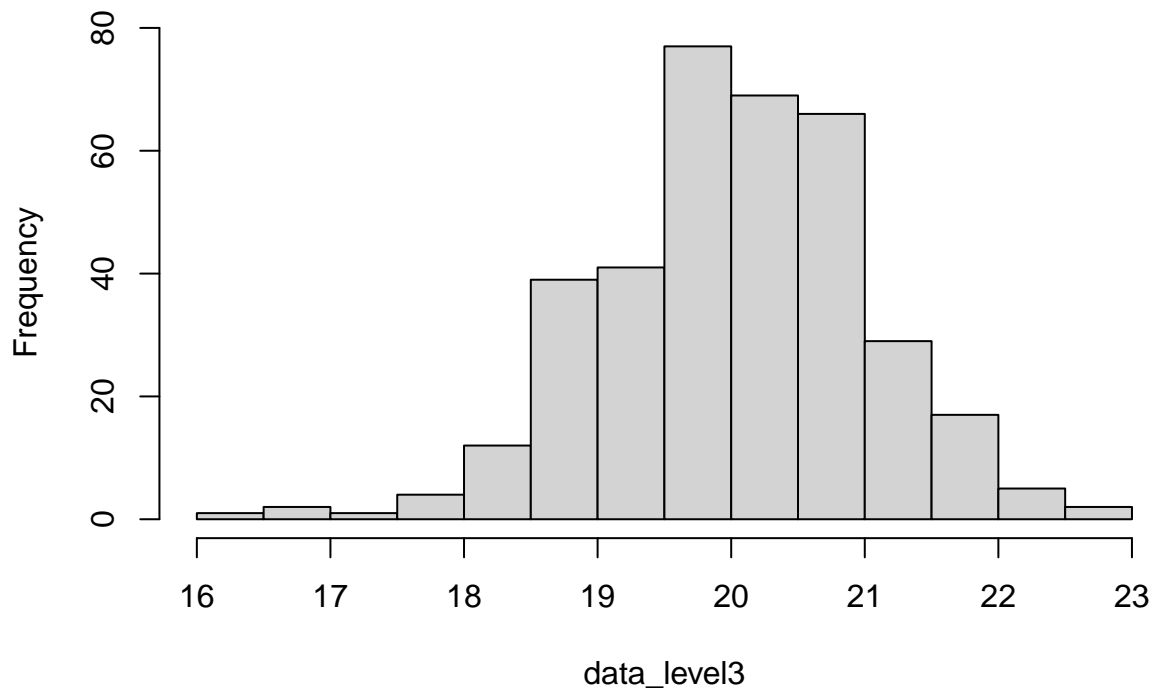
```
mean3 = mean(data_level3); var3 = var(data_level3); x11(); hist(data_level3); N_m3 = length(data_level3)
```

Histogram of data_level2



```
c(N_m1, N_m2, N_m3)
```

Histogram of data_level3



```
## [1] 66 169 365
```

```
c(mean1, mean2, mean3)
```

```
## [1] -20.07729349 0.08255401 20.02558109
```

```
c(var1, var2, var3)
```

```
## [1] 1.168506 1.160008 0.999889
```

Run

```
niter <- 1000
```

```
burnin <- 1000
```

```
thin <- 1
```

```
option<-list("Mstar0" = 3, "Lambda0" = 3, "mu0" = 0, "sigma0" = 1, "gamma0" = 1,
  "Adapt_MH_hyp1" = 0.7, "Adapt_MH_hyp2" = 0.234, "Adapt_MH_power_lim" = 10, "Adapt_MH_var0" = 1,
  "k0" = 1/10, "nu0" = 10, "alpha_gamma" = 1,
  "beta_gamma" = 1, "alpha_lambda" = 1, "beta_lambda" = 1,
  "UpdateU" = T, "UpdateM" = F, "UpdateGamma" = T, "UpdateS" = T,
  "UpdateTau" = T, "UpdateLambda" = F, "partition" = real_partition
)
```

```
#GDFMM = GDFMM_sampler(data, niter, burnin, thin, seed = 123, option = option)
```

```
GDFMM = GDFMM_sampler(data, niter, burnin, thin, seed = 123, FixPartition = T, option = option)
```

```
##
```

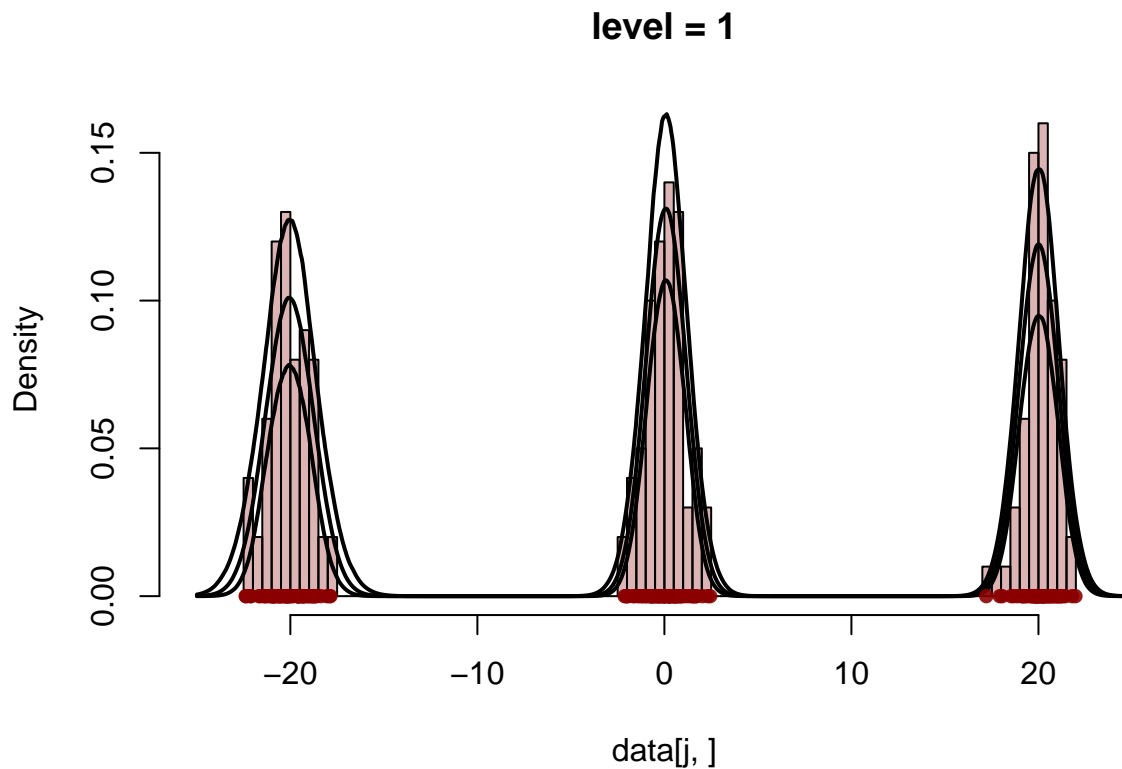
```
## Check that provided partition is well formed. It must start from 0 and all values must be contiguous.
## initialize_Partition with non empty partition_vec
## (K, Mstar, M) = (3,0,3)
## Chiamato initialize_S con gs_engine, mette casuale!
```

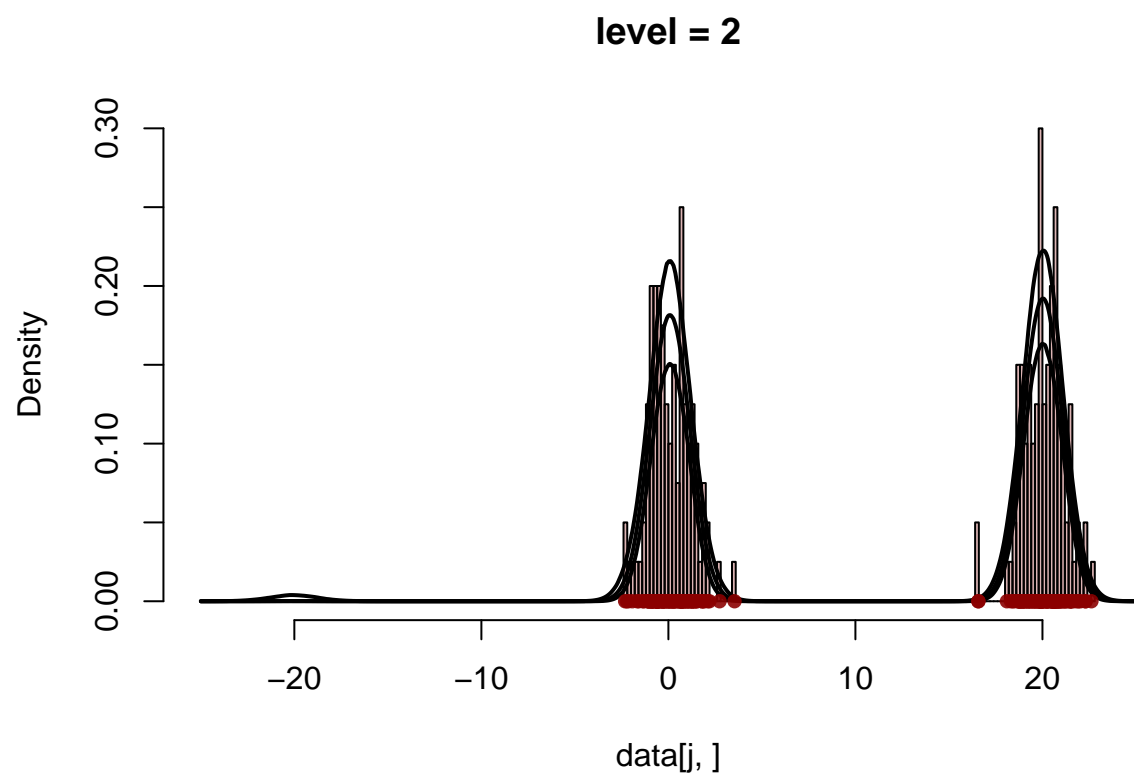
Analisi output - Calcolo predittive

```
l_grid = 1000
grid = seq(-25,25,length.out = l_grid)

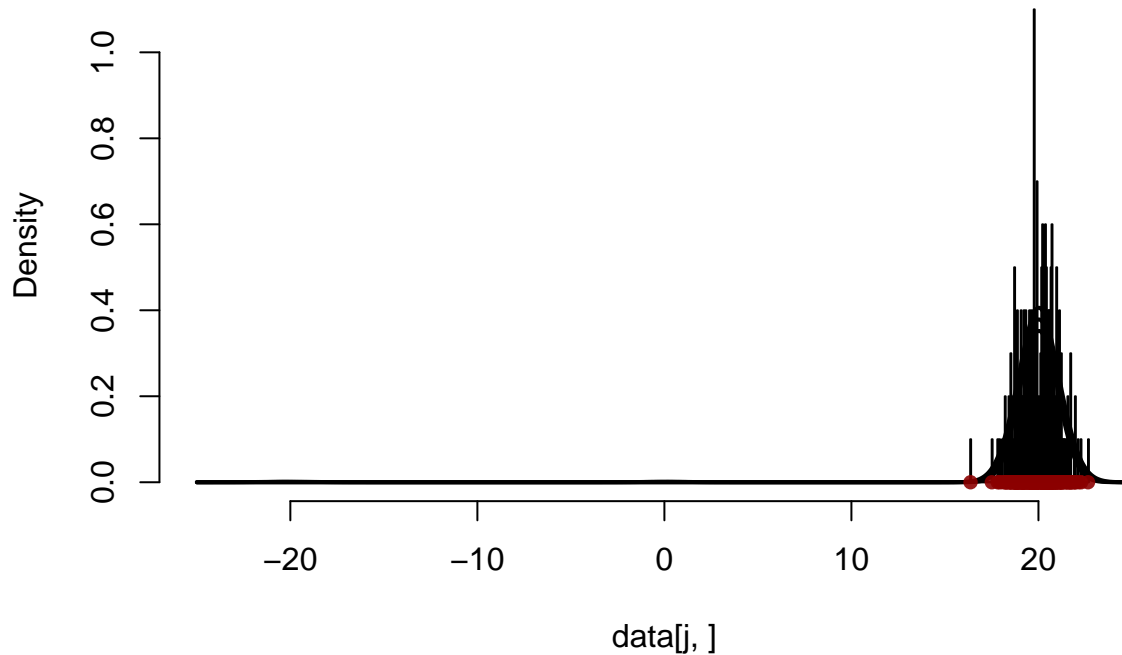
# Predictive in all groups
Pred_all = predictive_all_groups(grid = grid, fit = GDFMM)

for(j in 1:d){
  hist(data[j,], freq = F, breaks = l_grid/10, col = ACutils::t_col("darkred", 70), xlim = range(grid),
       main = paste0("level = ",j))
  matplot(x = grid, y = t(Pred_all[[j]]), type = 'l', col = 'black', lty = 1, lwd = 2, add = T)
  points(x = data[j,], y = rep(0, length(data[j,])), pch = 16, col = ACutils::t_col("darkred", 10))
}
```





level = 3



ripeto per $d = 10$

```
d = 10                                # number of groups
K = 3                                  # number of global clusters
mu = c(-20,0,20)                      # vectors of means
sd = c(1,1,1)                         # vector of sd
n_j = rep(200, d)                     # set cardinality of the groups
p = matrix(0, nrow = d, ncol = K)     # matrix with components weights

set.seed(124123)
Kgruppo = c()
componenti_gruppo = NULL
data = matrix(NA, nrow = d, ncol = max(n_j)) # d x max(n_j) matrix
cluster = matrix(NA, nrow = d, ncol = max(n_j)) # d x max(n_j) matrix
real_partition = c()                   # real_partition is a vector of length sum(n_j), it collects all the group means
                                         # values are collected level by level, so first all the values in level 1, then level 2, etc.
                                         # cluster label must always start from 0!

for(j in 1:d){
  Kgruppo[j] = sample(1:K,1) # number of clusters in each level
  componenti_gruppo[[j]] = sample(1:K,Kgruppo[j], replace = F) # choose the components
  p[j,1:Kgruppo[j]] = rep(1/Kgruppo[j], Kgruppo[j]) # set the weights all equals
  appoggio = genera_mix_gas(n = n_j[j], pro = p[j,1:Kgruppo[j]], means = mu[ componenti_gruppo[[j]] ],
                           sds = sd[ componenti_gruppo[[j]] ] )

  data[j, 1:n_j[j]] = appoggio$y
}
```



```

    #cluster[j, 1:n_j[j]] = appoggio$clu, #errore, genera_mia_gas usa sempre indici che partono da 1!
    cluster[j, 1:n_j[j]] = unlist(lapply(1:n_j[j], function(h){componenti_gruppo[[j]][appoggio$clu[h]]}))
    real_partition = c(real_partition, cluster[j, 1:n_j[j]])
}

niter <- 1000
burnin <- 1000
thin <- 1

option<-list("Mstar0" = 3, "Lambda0" = 3, "mu0" = 0,"sigma0"= 1, "gamma0" = 1,
            "Adapt_MH_hyp1"= 0.7,"Adapt_MH_hyp2"= 0.234, "Adapt_MH_power_lim"=10, "Adapt_MH_var0"=1,
            "k0"= 1/10, "nu0"=10, "alpha_gamma"=1,
            "beta_gamma"=1, "alpha_lambda"=1, "beta_lambda"=1,
            "UpdateU" = T, "UpdateM" = F, "UpdateGamma" = T, "UpdateS" = T,
            "UpdateTau" = T, "UpdateLambda" = F, "partition" = real_partition
)

#GDFMM = GDFMM_sampler(data, niter, burnin, thin, seed = 123, option = option)
GDFMM = GDFMM_sampler(data, niter, burnin, thin, seed = 123, FixPartition = T, option = option)

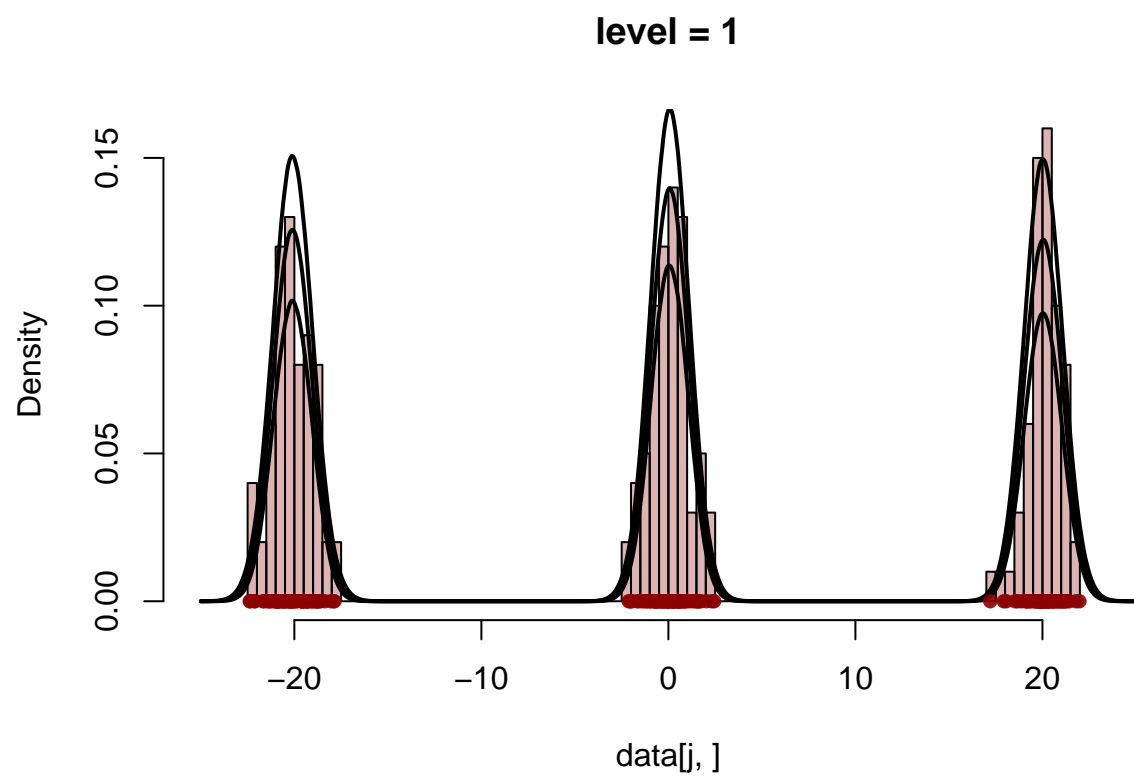
##
## Check that provided partition is well formed. It must start from 0 and all values must be contiguous
## initialize_Partition with non empty partition_vec
## (K, Mstar, M) = (3,0,3)
## Chiamato initialize_S con gs_engine, mette casuale!

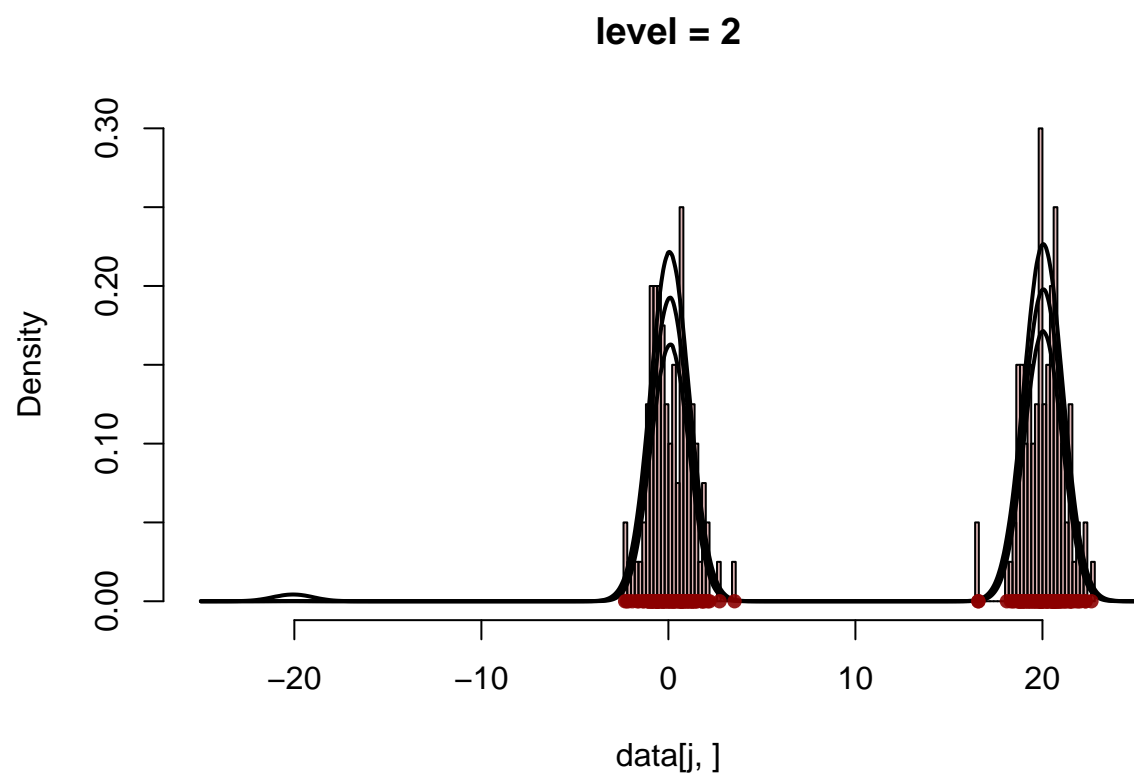
l_grid = 1000
grid = seq(-25,25,length.out = l_grid)

# Predictive in all groups
Pred_all = predictive_all_groups(grid = grid, fit = GDFMM)

for(j in 1:d){
  hist(data[j,], freq = F, breaks = l_grid/10, col = ACutils::t_col("darkred", 70), xlim = range(grid),
       main = paste0("level = ",j))
  matplot(x = grid, y = t(Pred_all[[j]]), type = 'l', col = 'black', lty = 1, lwd = 2, add = T)
  points(x = data[j,], y = rep(0, length(data[j,])), pch = 16, col = ACutils::t_col("darkred", 10))
}

```





level = 3

