

Rmd_Introducing_nu

Partizione fissa

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

Let us consider $S_{jm} \sim \text{Gamma}(\gamma_j, 1)$, $S'_{jm} \sim \text{Gamma}(\gamma_j, \nu)$. Hence, by developping some computations we have $S'_{jm} = \frac{1}{\nu} S_{jm}$. Then U is associated to S and U' to S' . We have that $U' = \nu U$. Regarding the Laplace transform, we have $\psi_{S'}(U'_j) = \psi_S(\frac{U'_j}{\nu}) = \psi_S(U_j)$.

Fixed Partition

Genero i dati per 10 gruppi. Ecco un elenco di cose che ho notato: Se $\nu > 1$, U e S esplodono, nel senso che vengono proprio infinito.

Inizialmente, mi sembra di capire che c'è un valore di ν da cui si vedono dei miglioramenti. Poi, ho notato che in effetti quando $\nu = 1$ c'è qualcosa che non va perché mixxa solo se γ_j sono tenuti fissi ad un valore piccolo. Ma se si mettono aleatori, non va bene. Poi c'è una zona di transizione (in questo esempio, per $\nu \approx 0.7$) in cui l'effetto dipende molto dal valore iniziale γ_0 . Poi se ν diventa sufficientemente piccolo, allora si stabilizza sempre attorno allo stesso valore.

Domanda, al momento a noi non piace il fatto che M^* rimanga sempre 0 e ci piace quando invece si ottiene una certa aleatorietà. Però si attesta sempre attorno ad un valore che è circa 20. Perché? Da dove esce questo valore? Va bene?

Genero i dati (d=10)

```
d = 10                # number of groups
K = 3                 # number of global clusters
mu = c(-5,0,1)       # 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$y
cluster[j, 1:n_j[j]] = unlist(lapply(1:n_j[j], function(h){componenti_gruppo[[j]][appoggio$clu[h]]}))
real_partition = c(real_partition, cluster[j, 1:n_j[j]])
}

N_m = table(real_partition)

data_level1 = data[cluster==1]
data_level2 = data[cluster==2]
data_level3 = data[cluster==3]

```

$\nu = 1$, γ updated

```

niter <- 5000
burnin <- 1
thin <- 1

option<-set_options("nu" = 1, "Mstar0" = 3, "Lambda0" = 3, "mu0" = 0, "sigma0" = 1, "gamma0" = 0.01,
                    "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" = T, "UpdateGamma" = T, "UpdateS" = T,
                    "UpdateTau" = T, "UpdateLambda" = T, "partition" = real_partition
)

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
## GibbsSampler.cpp - Initialization
## initialize_Partition with non empty partition_vec
## Watch out modification: Mstar is not set to zero but to Mstar0
## (K, Mstar, M) = (3,3,6)

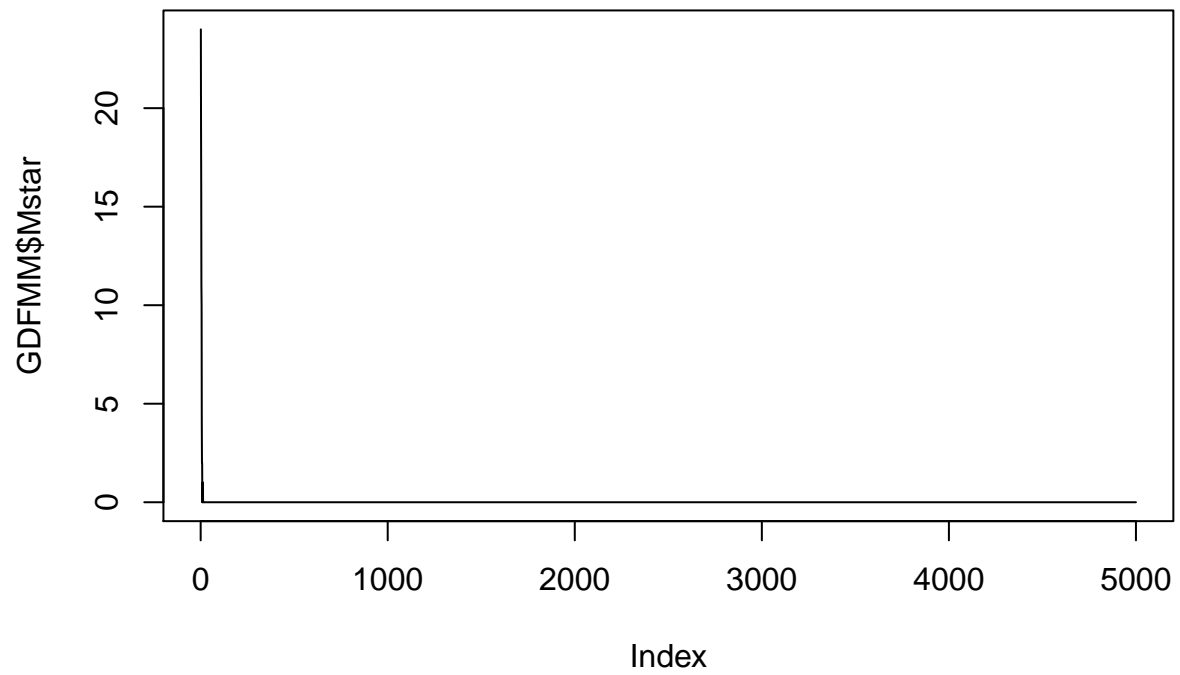
#Mstar
summary(GDFMM$Mstar)

##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      0.000   0.000   0.000   0.018   0.000   24.000

plot(GDFMM$Mstar, type = 'l', main = "Mstar")

```

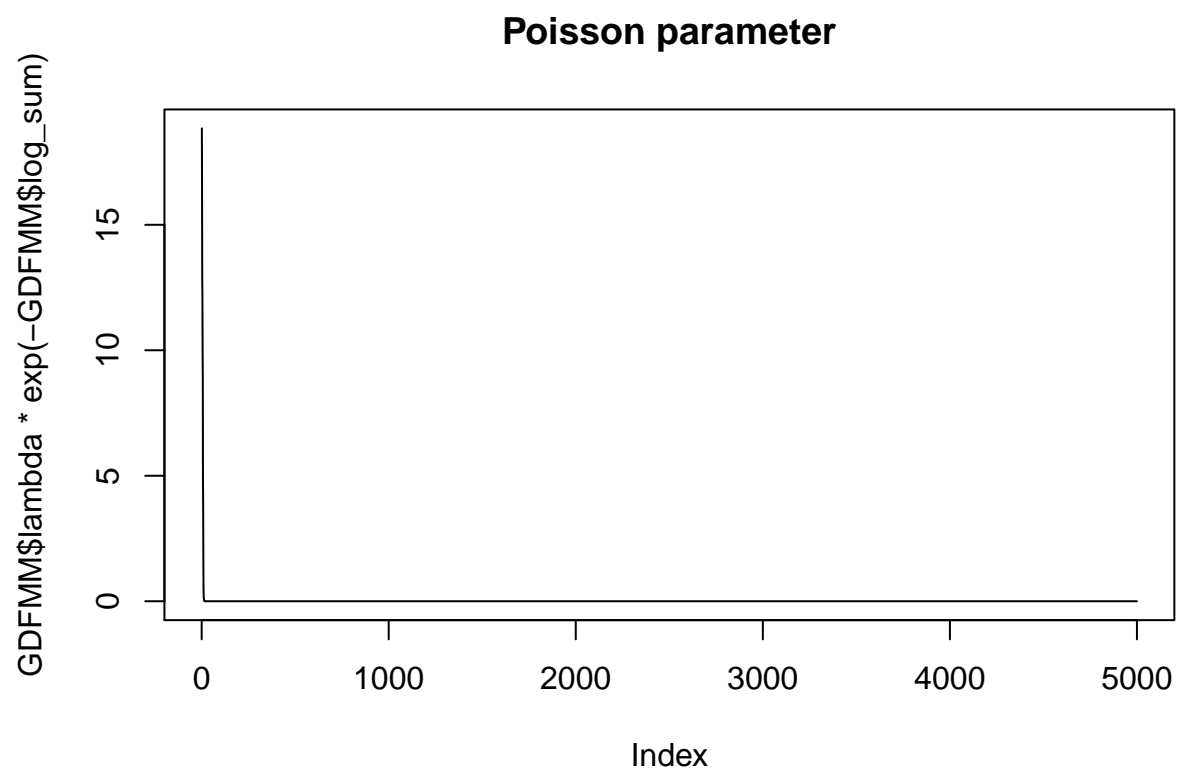
Mstar



```
#Parametro Poisson:  $\lambda * \exp(-\log\_sum)$   
summary(GDFMM$lambda*exp(-GDFMM$log_sum))
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.  
## 0.00000 0.00000 0.00000 0.01818 0.00000 18.84239
```

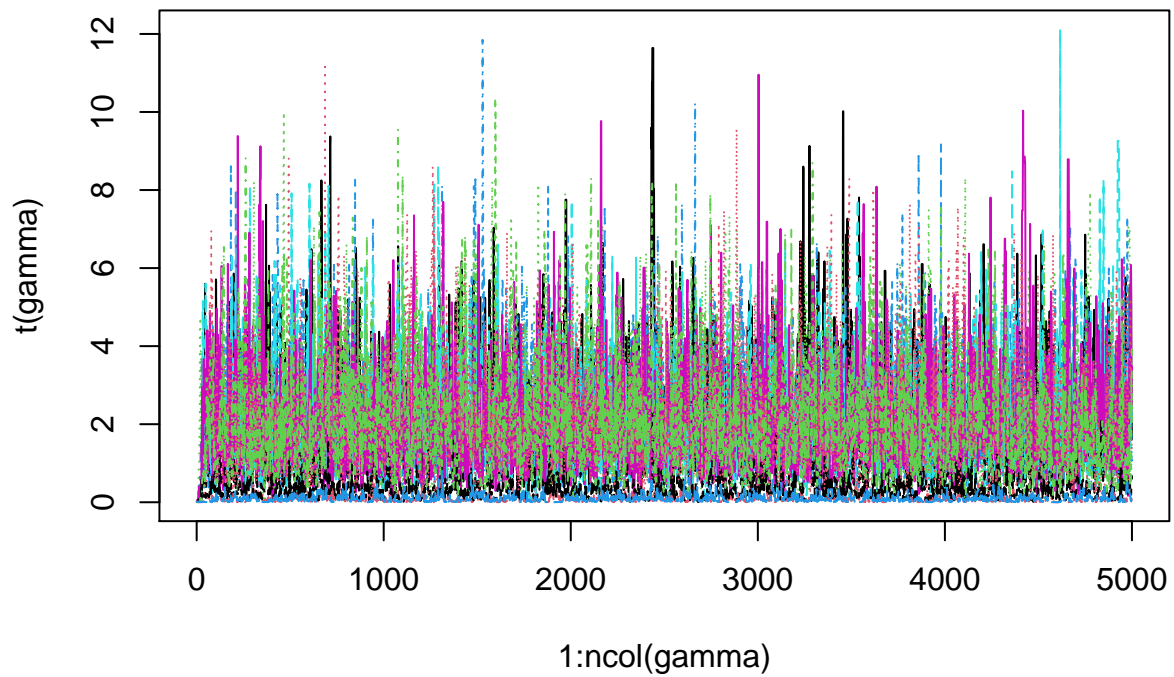
```
plot(GDFMM$lambda*exp(-GDFMM$log_sum), type = 'l', main = "Poisson parameter")
```



```
#gamma.s
gamma = GDFMM$gamma
post_mean_gamma = rowMeans(gamma)
post_mean_gamma

## [1] 2.10846938 0.08286548 1.99944185 2.07910758 2.07147721 2.11472585
## [7] 0.34364325 2.07025438 2.05035759 0.08875989

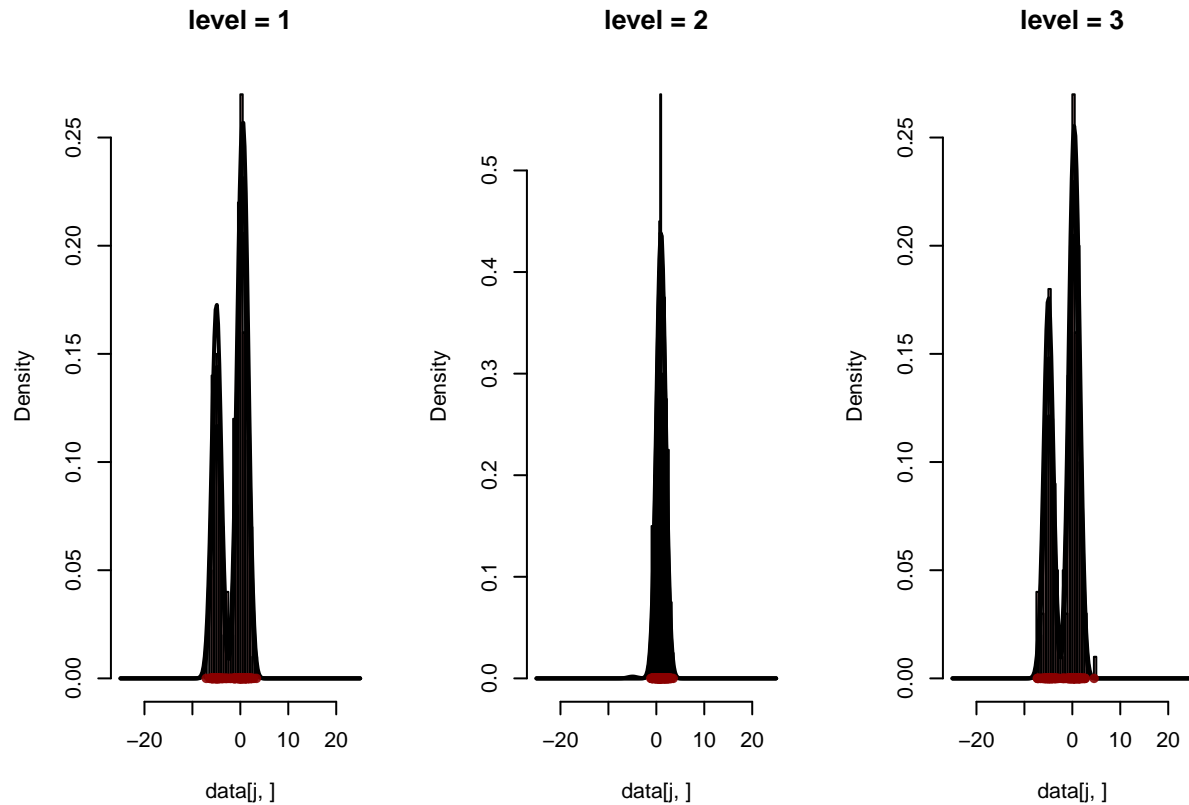
matplot(x = 1:ncol(gamma), y = t(gamma), type = 'l')
```



```
# Predictive -----

l_grid = 200
grid = seq(-25,25,length.out = l_grid)

# Predictive in all groups
Pred_all = predictive_all_groups(grid = grid, fit = GDFMM)
par(mfrow = c(1,3))
for(j in 1:3){
  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))
}
```



$\nu = 1$, γ fixed and small (0.01)

```
niter <- 5000
burnin <- 1
thin <- 1

option<-set_options("nu" = 1, "Mstar0" = 3, "Lambda0" = 3, "mu0" = 0, "sigma0" = 1, "gamma0" = 0.01,
  "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" = T, "UpdateGamma" = F, "UpdateS" = T,
  "UpdateTau" = T, "UpdateLambda" = T, "partition" = real_partition
)

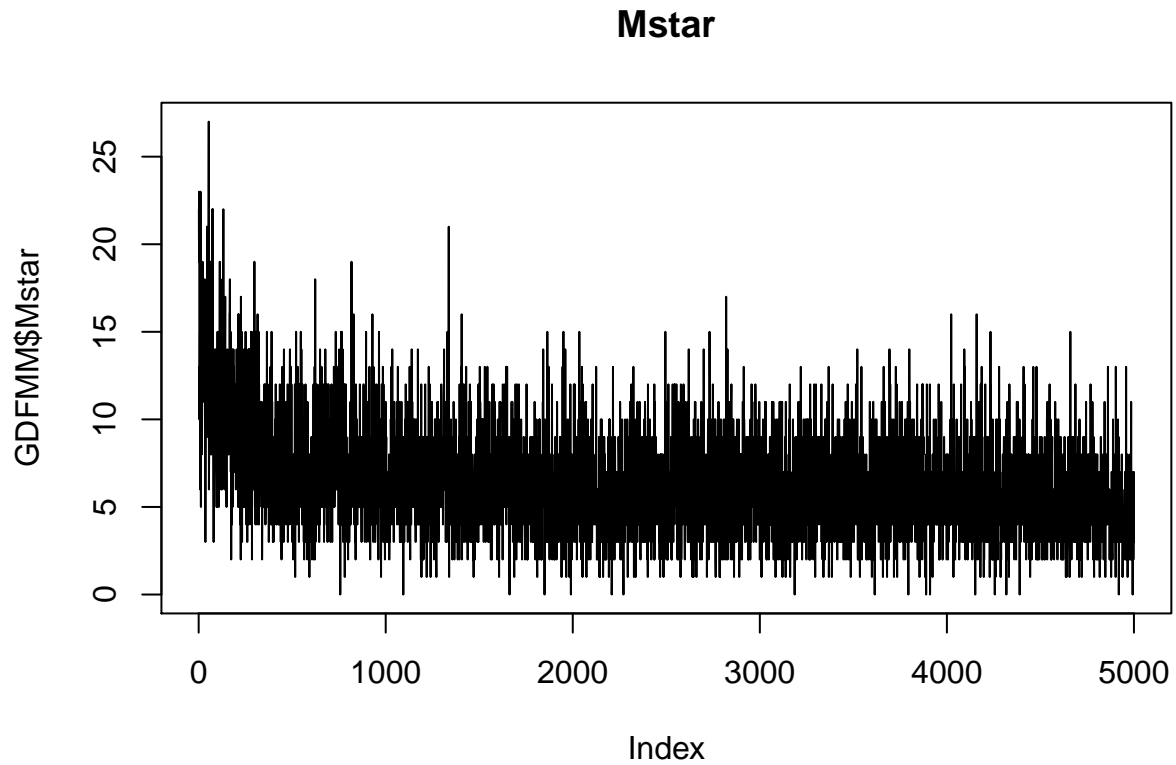
GDFMM = GDFMM_sampler(data, niter, burnin, thin, seed = 123, FixPartition = T, option = option)

##
## Check that provided partition is well formed. It must start from 0 and all values must be contiguous
## GibbsSampler.cpp - Initialization
## initialize_Partition with non empty partition_vec
## Watch out modification: Mstar is not set to zero but to Mstar0
## (K, Mstar, M) = (3,3,6)

#Mstar
summary(GDFMM$Mstar)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.000   4.000   6.000   6.467   8.000  27.000
```

```
plot(GDFMM$Mstar, type = 'l', main = "Mstar")
```

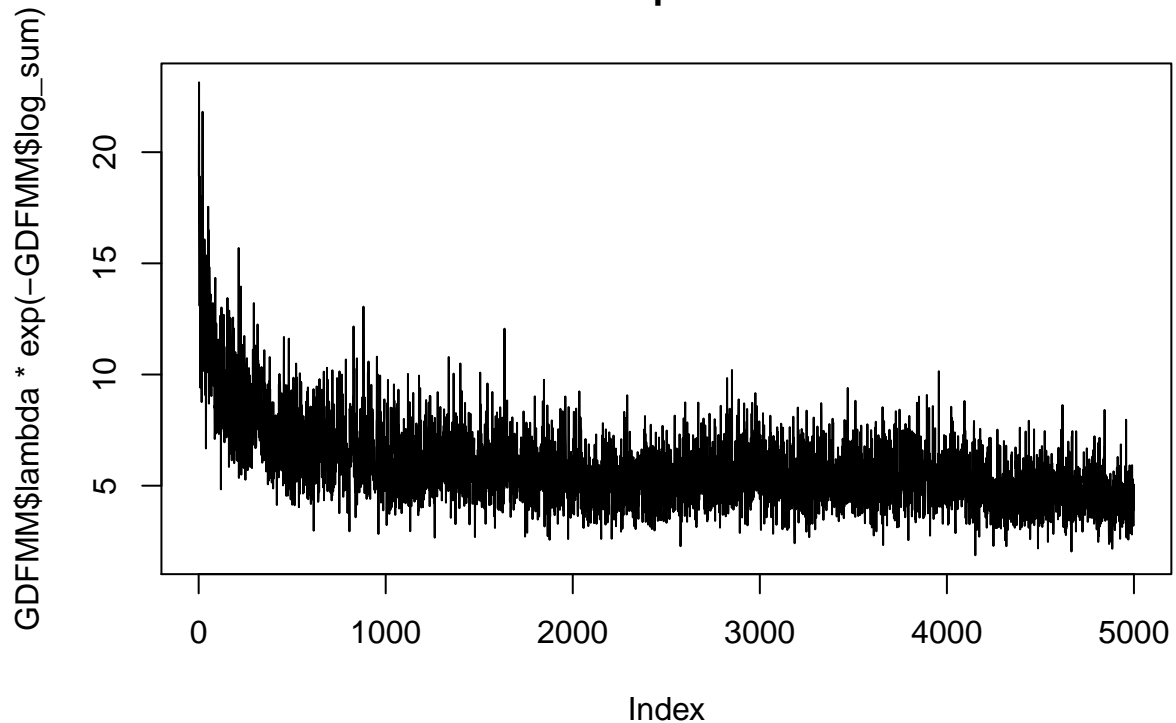


```
#Parametro Poisson: lambda * exp(-log_sum)
summary(GDFMM$lambda*exp(-GDFMM$log_sum))
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.873   4.639   5.537   5.857   6.658  23.140
```

```
plot(GDFMM$lambda*exp(-GDFMM$log_sum), type = 'l', main = "Poisson parameter")
```

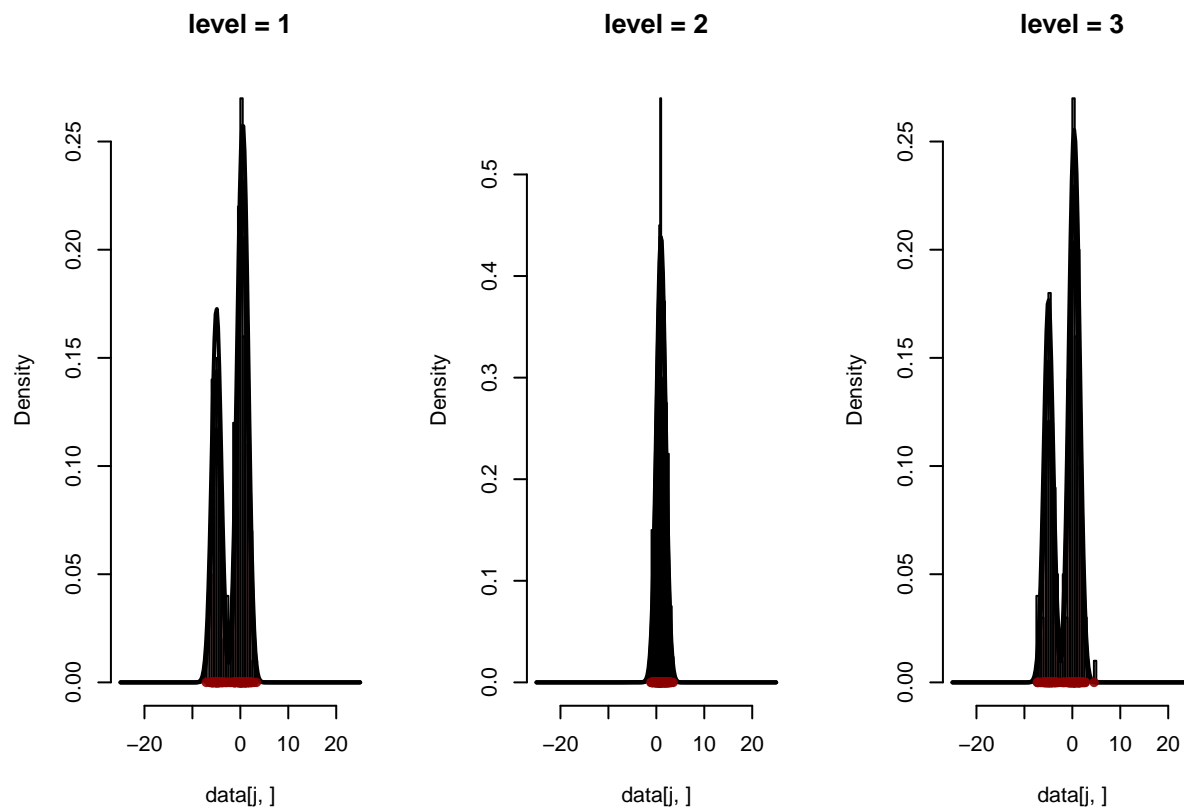
Poisson parameter



```
#gammas
gamma = GDFMM$gamma
post_mean_gamma = rowMeans(gamma)
# Predictive -----

l_grid = 200
grid = seq(-25,25,length.out = l_grid)

# Predictive in all groups
Pred_all = predictive_all_groups(grid = grid, fit = GDFMM)
par(mfrow = c(1,3))
for(j in 1:3){
  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))
}
```

$\nu = 0.75$, $\gamma_0 = 1$

Da qui in poi, fisso solo il valore iniziale e poi gamma lo lascio variare.

```
niter <- 10000
burnin <- 1
thin <- 1

option<-set_options("nu" = 0.75, "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" = T, "UpdateGamma" = T, "UpdateS" = T,
  "UpdateTau" = T, "UpdateLambda" = T, "partition" = real_partition
)

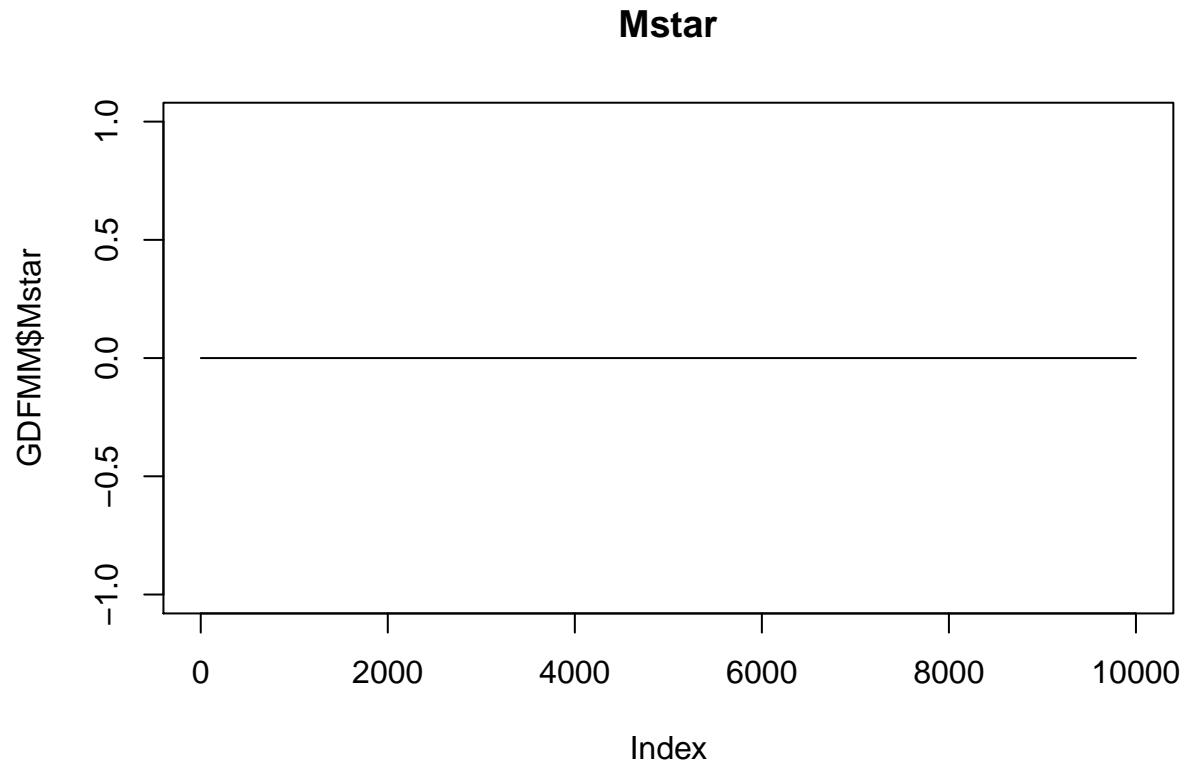
GDFMM = GDFMM_sampler(data, niter, burnin, thin, seed = 123, FixPartition = T, option = option)

##
## Check that provided partition is well formed. It must start from 0 and all values must be contiguous
## GibbsSampler.cpp - Initialization
## initialize_Partition with non empty partition_vec
## Watch out modification: Mstar is not set to zero but to Mstar0
## (K, Mstar, M) = (3,3,6)

#Mstar
summary(GDFMM$Mstar)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##         0         0         0         0         0         0
```

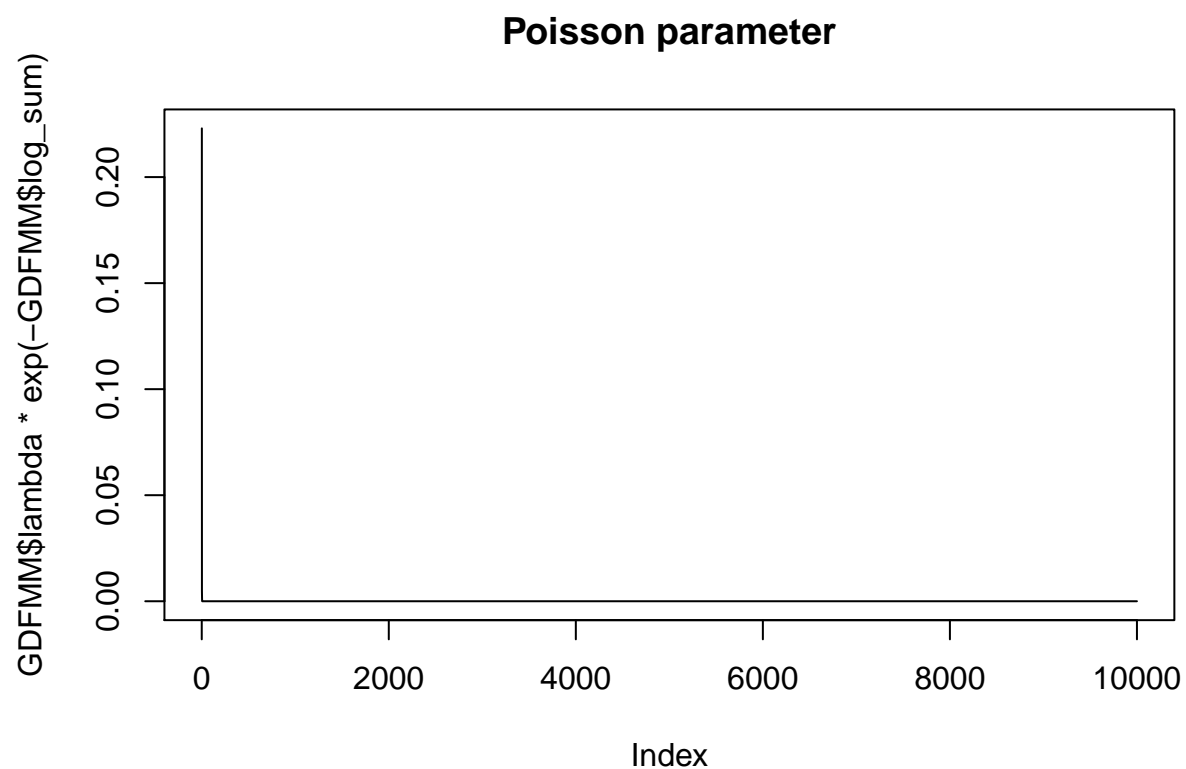
```
plot(GDFMM$Mstar, type = 'l', main = "Mstar")
```



```
#Parametro Poisson: lambda * exp(-log_sum)
summary(GDFMM$lambda*exp(-GDFMM$log_sum))
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.000e+00 0.000e+00 0.000e+00 2.295e-05 0.000e+00 2.230e-01
```

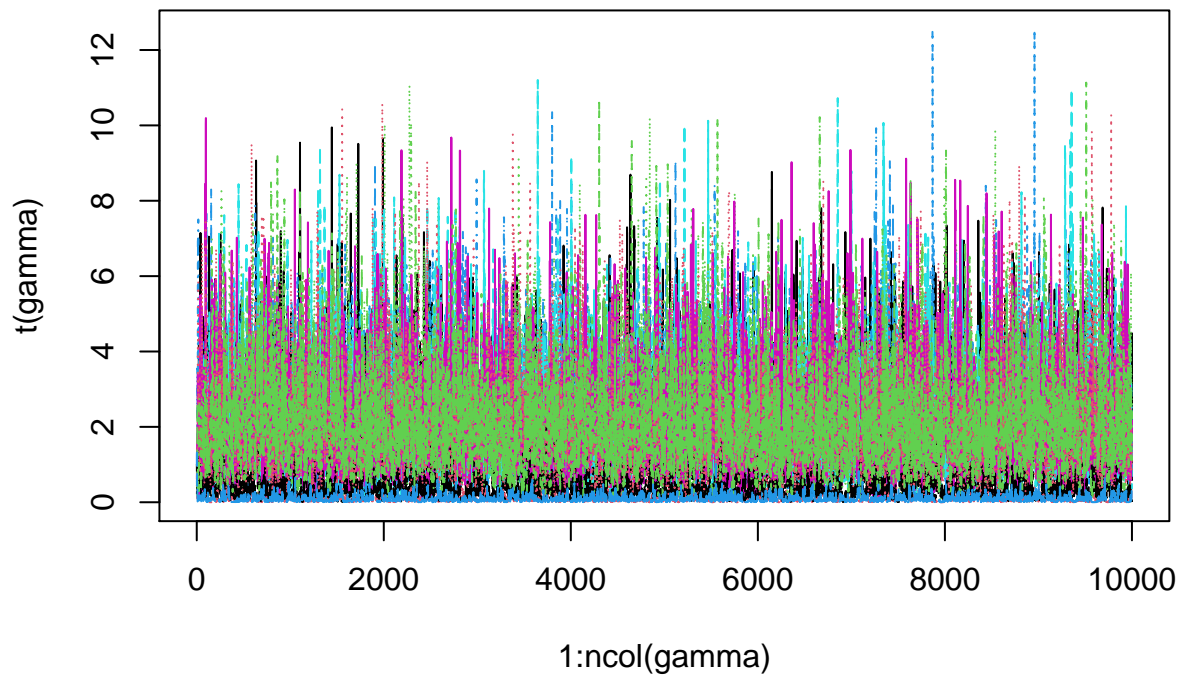
```
plot(GDFMM$lambda*exp(-GDFMM$log_sum), type = 'l', main = "Poisson parameter")
```



```
#gamma.s
gamma = GDFMM$gamma
post_mean_gamma = rowMeans(gamma)
post_mean_gamma

## [1] 2.07539249 0.09212319 2.06284642 2.07099712 2.16693369 2.14944435
## [7] 0.34978824 2.09361469 2.13996121 0.09097763

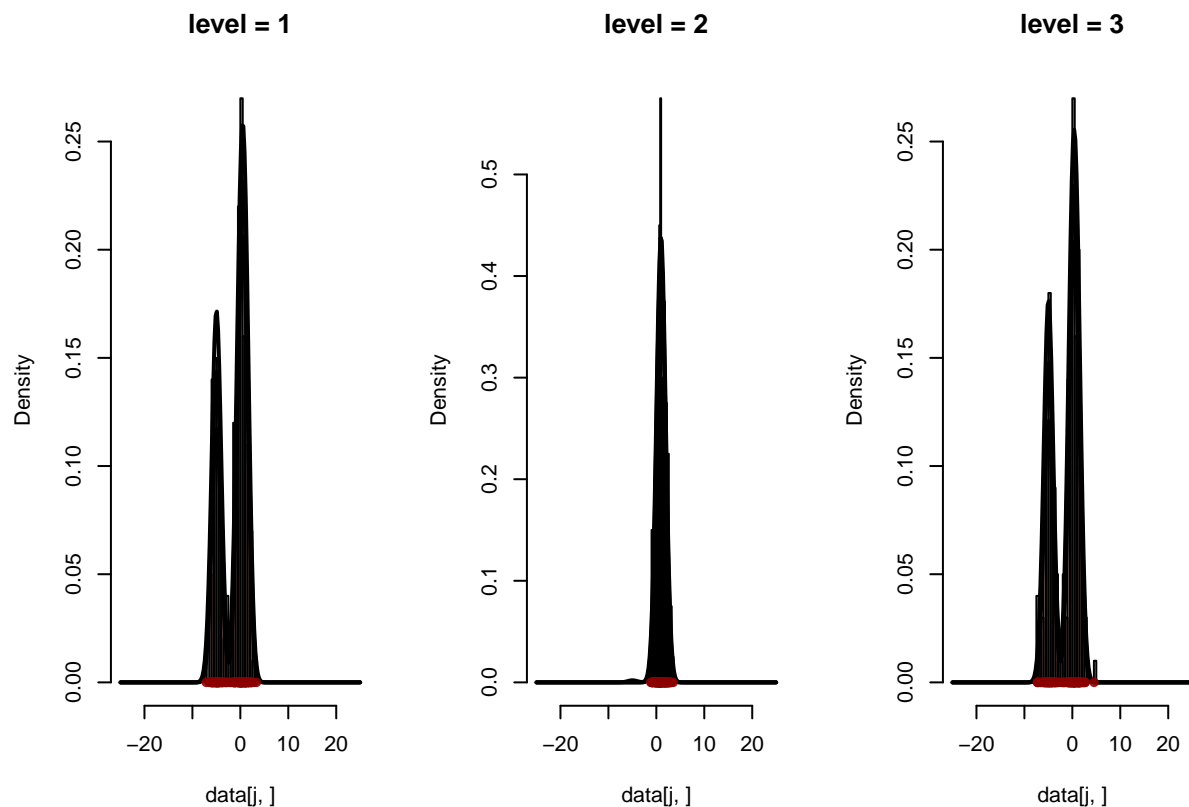
matplot(x = 1:ncol(gamma), y = t(gamma), type = 'l')
```



```
# Predictive -----

l_grid = 200
grid = seq(-25,25,length.out = l_grid)

# Predictive in all groups
Pred_all = predictive_all_groups(grid = grid, fit = GDFMM)
par(mfrow = c(1,3))
for(j in 1:3){
  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))
}
```



Questa è stranissima.

$\nu = 0.75$, $\gamma_0 = 0.1$

```
niter <- 10000
burnin <- 1
thin <- 1

option<-set_options("nu" = 0.75, "Mstar0" = 3, "Lambda0" = 3, "mu0" = 0, "sigma0" = 1, "gamma0" = 0.1,
  "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" = T, "UpdateGamma" = T, "UpdateS" = T,
  "UpdateTau" = T, "UpdateLambda" = T, "partition" = real_partition
)

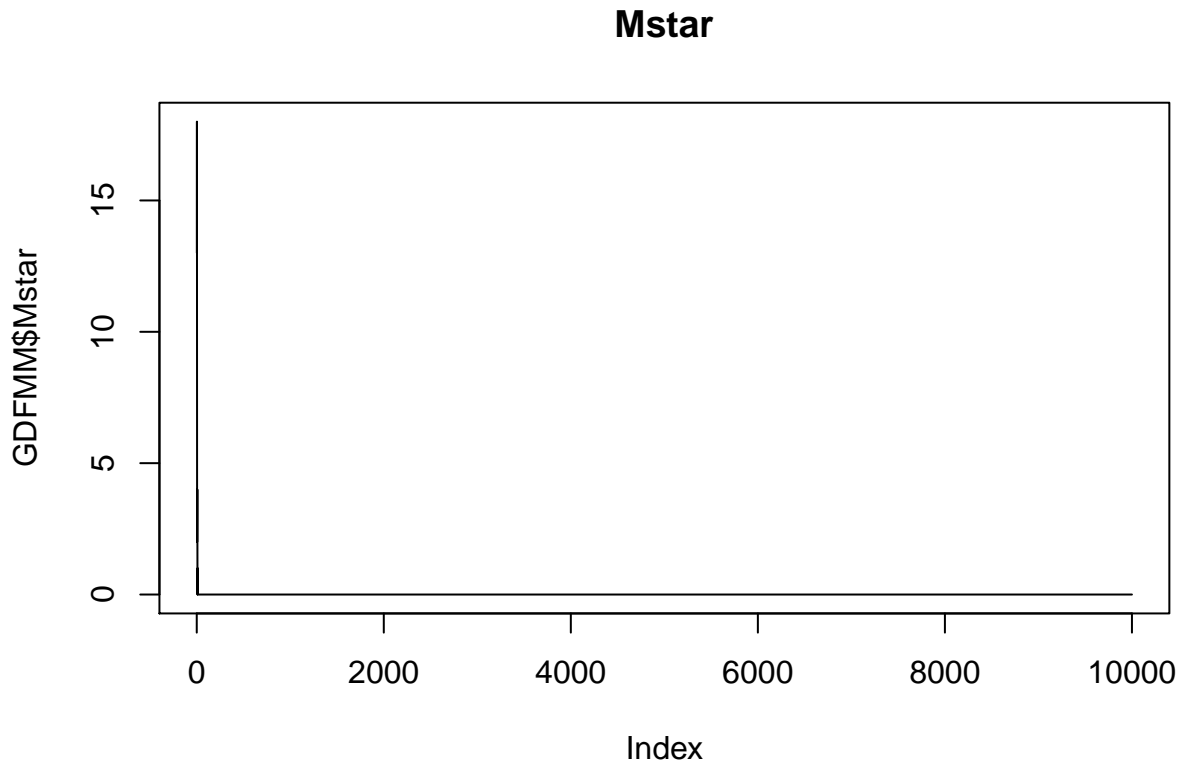
GDFMM = GDFMM_sampler(data, niter, burnin, thin, seed = 123, FixPartition = T, option = option)

##
## Check that provided partition is well formed. It must start from 0 and all values must be contiguous
## GibbsSampler.cpp - Initialization
## initialize_Partition with non empty partition_vec
## Watch out modification: Mstar is not set to zero but to Mstar0
## (K, Mstar, M) = (3,3,6)

#Mstar
summary(GDFMM$Mstar)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.0000 0.0000 0.0000 0.0052 0.0000 18.0000
```

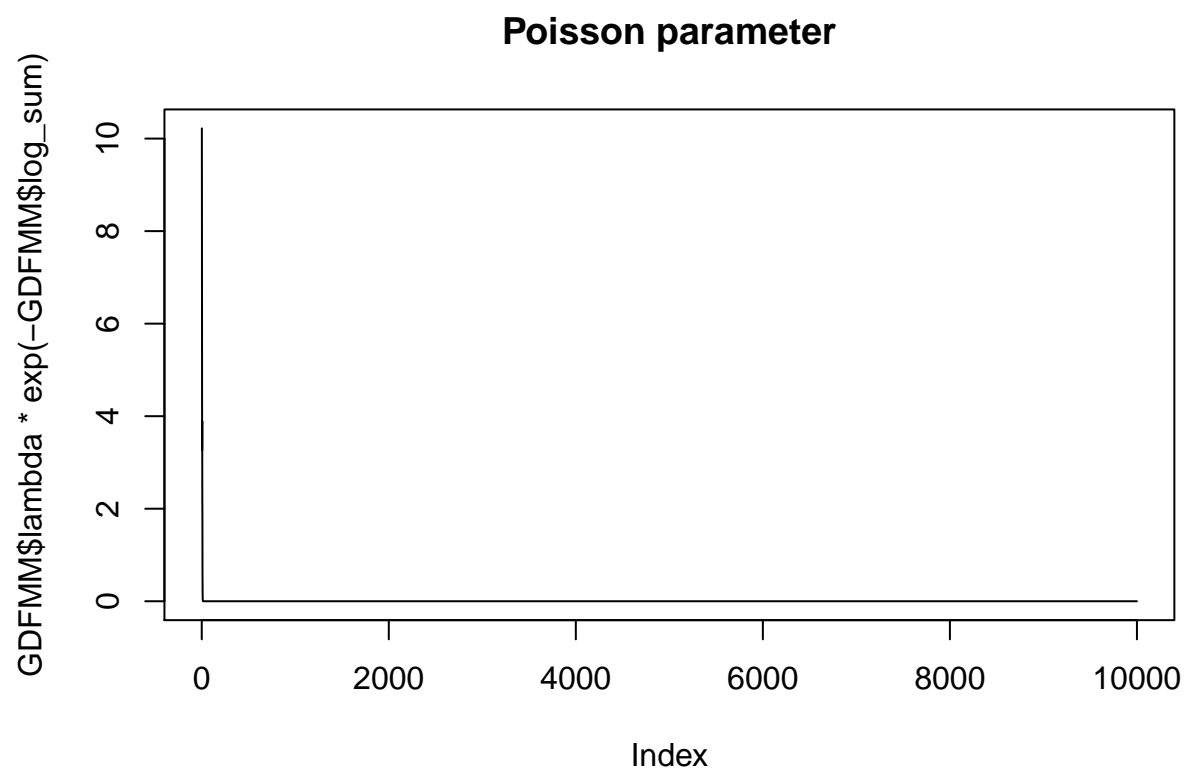
```
plot(GDFMM$Mstar, type = 'l', main = "Mstar")
```



```
#Parametro Poisson: lambda * exp(-log_sum)
summary(GDFMM$lambda*exp(-GDFMM$log_sum))
```

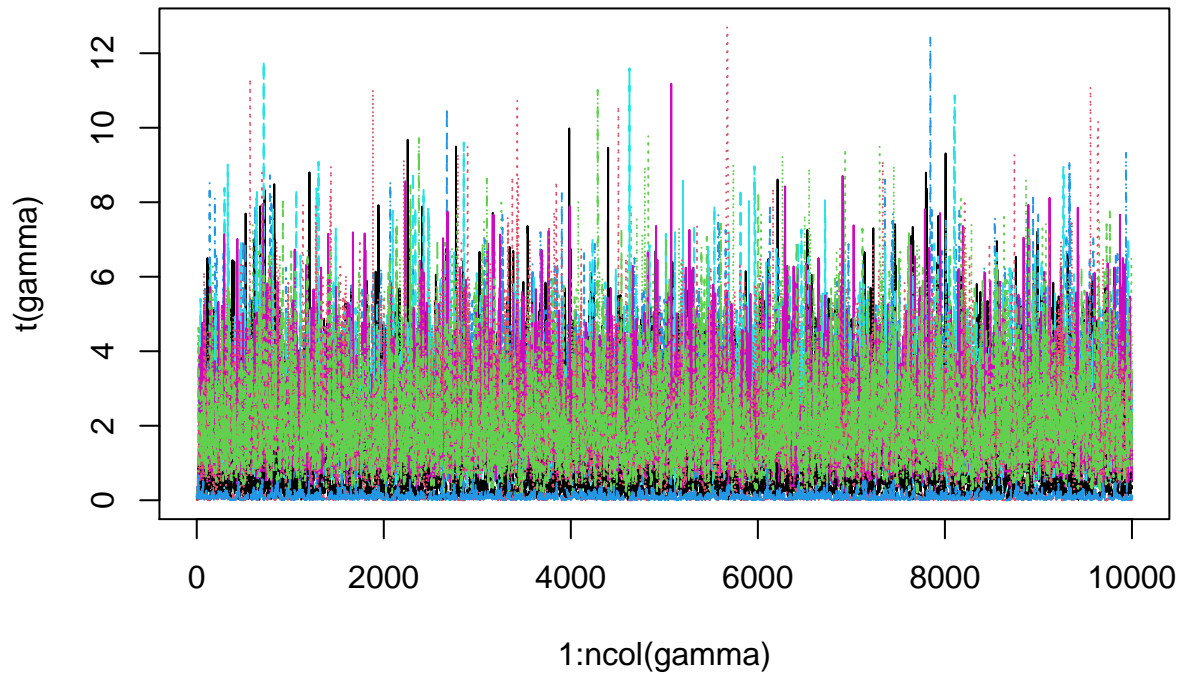
```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.000000 0.000000 0.000000 0.003666 0.000000 10.220921
```

```
plot(GDFMM$lambda*exp(-GDFMM$log_sum), type = 'l', main = "Poisson parameter")
```



```
#gamma.s
gamma = GDFMM$gamma
post_mean_gamma = rowMeans(gamma)
post_mean_gamma

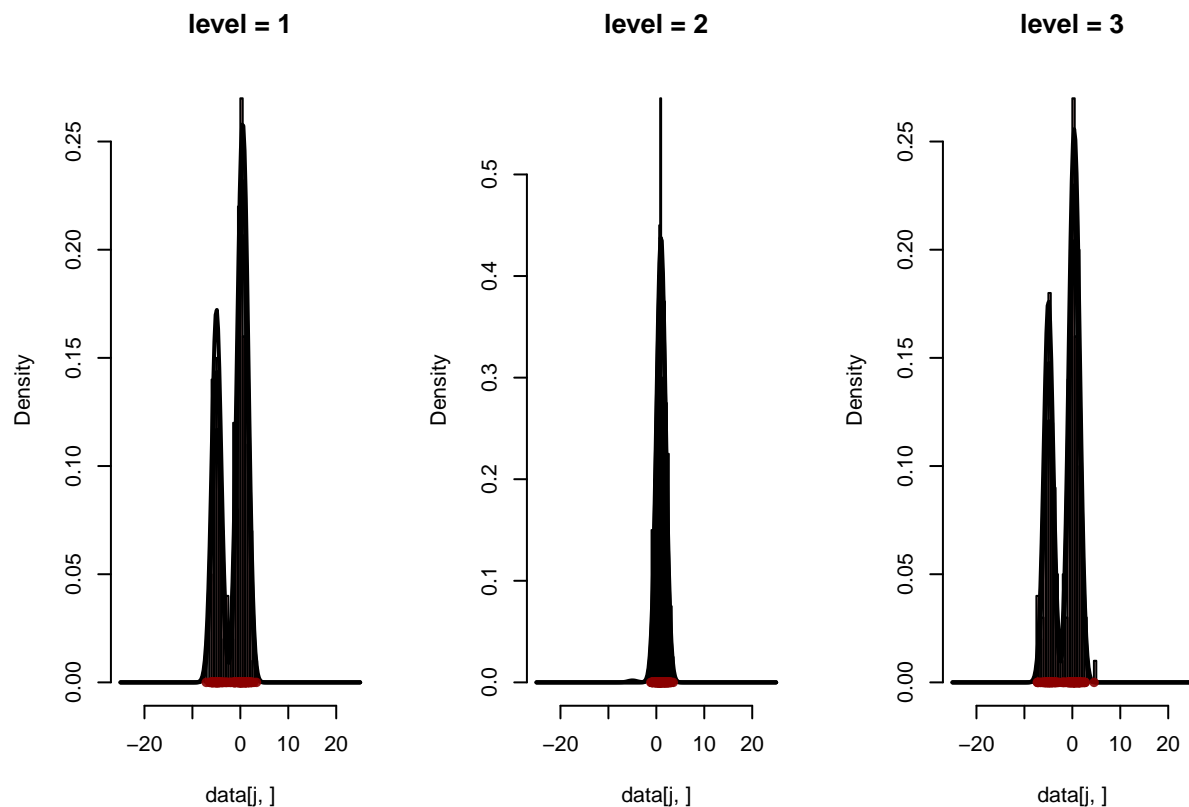
## [1] 2.08499563 0.08602215 2.00318368 2.11991611 2.14839426 2.07585864
## [7] 0.34647215 2.08222724 2.00194267 0.08816258
matplot(x = 1:ncol(gamma), y = t(gamma), type = 'l')
```



```
# Predictive -----

l_grid = 200
grid = seq(-25,25,length.out = l_grid)

# Predictive in all groups
Pred_all = predictive_all_groups(grid = grid, fit = GDFMM)
par(mfrow = c(1,3))
for(j in 1:3){
  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))
}
```

Questa mixxa pochissimo

$\nu = 0.70$, $\gamma_0 = 10$

```
niter <- 10000
burnin <- 1
thin <- 1

option<-set_options("nu" = 0.7, "Mstar0" = 3, "Lambda0" = 3, "mu0" = 0, "sigma0" = 1, "gamma0" = 10,
  "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" = T, "UpdateGamma" = T, "UpdateS" = T,
  "UpdateTau" = T, "UpdateLambda" = T, "partition" = real_partition
)

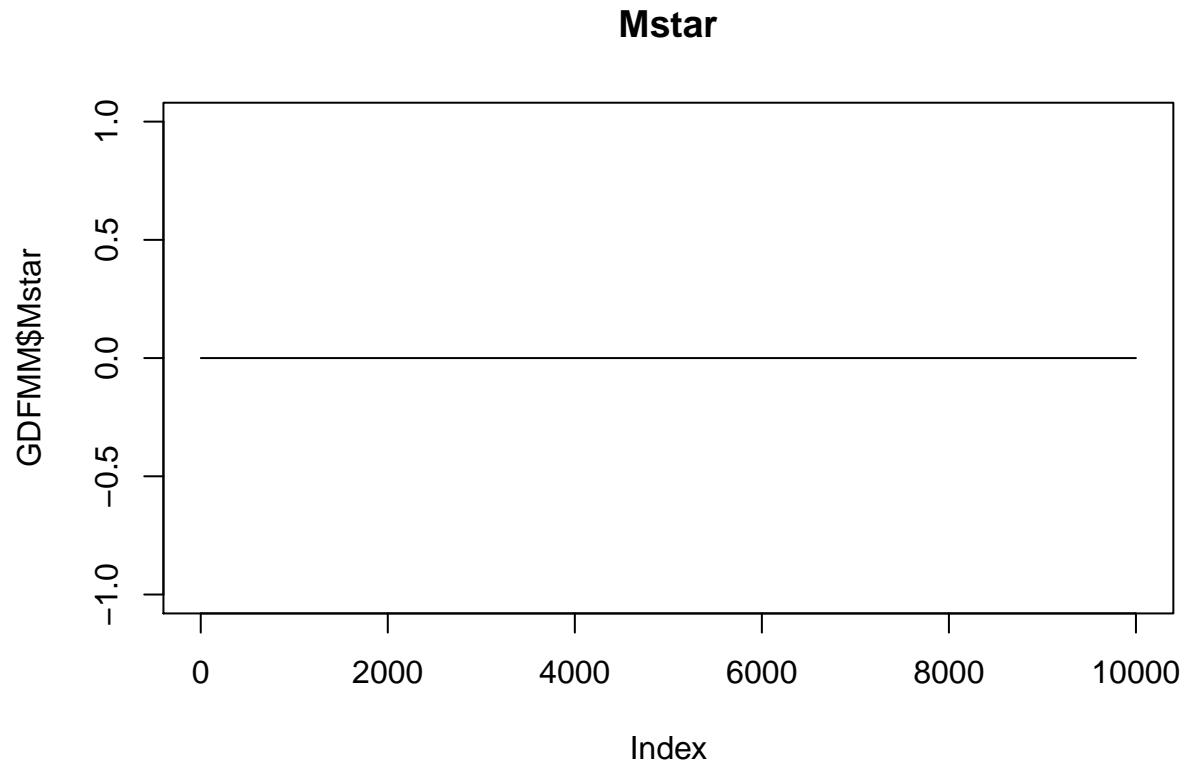
GDFMM = GDFMM_sampler(data, niter, burnin, thin, seed = 123, FixPartition = T, option = option)

##
## Check that provided partition is well formed. It must start from 0 and all values must be contiguous
## GibbsSampler.cpp - Initialization
## initialize_Partition with non empty partition_vec
## Watch out modification: Mstar is not set to zero but to Mstar0
## (K, Mstar, M) = (3,3,6)

#Mstar
summary(GDFMM$Mstar)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0       0       0       0       0       0
```

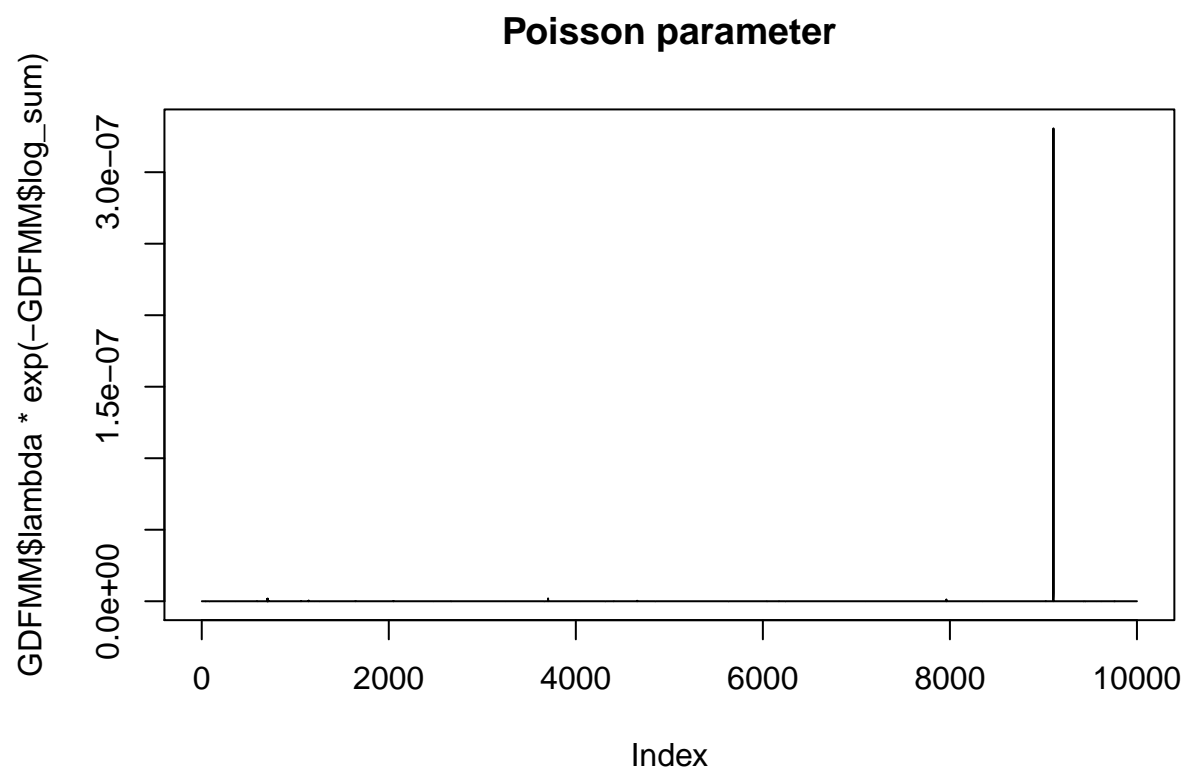
```
plot(GDFMM$Mstar, type = 'l', main = "Mstar")
```



```
#Parametro Poisson: lambda * exp(-log_sum)
summary(GDFMM$lambda*exp(-GDFMM$log_sum))
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.000e+00 0.000e+00 0.000e+00 3.520e-11 0.000e+00 3.307e-07
```

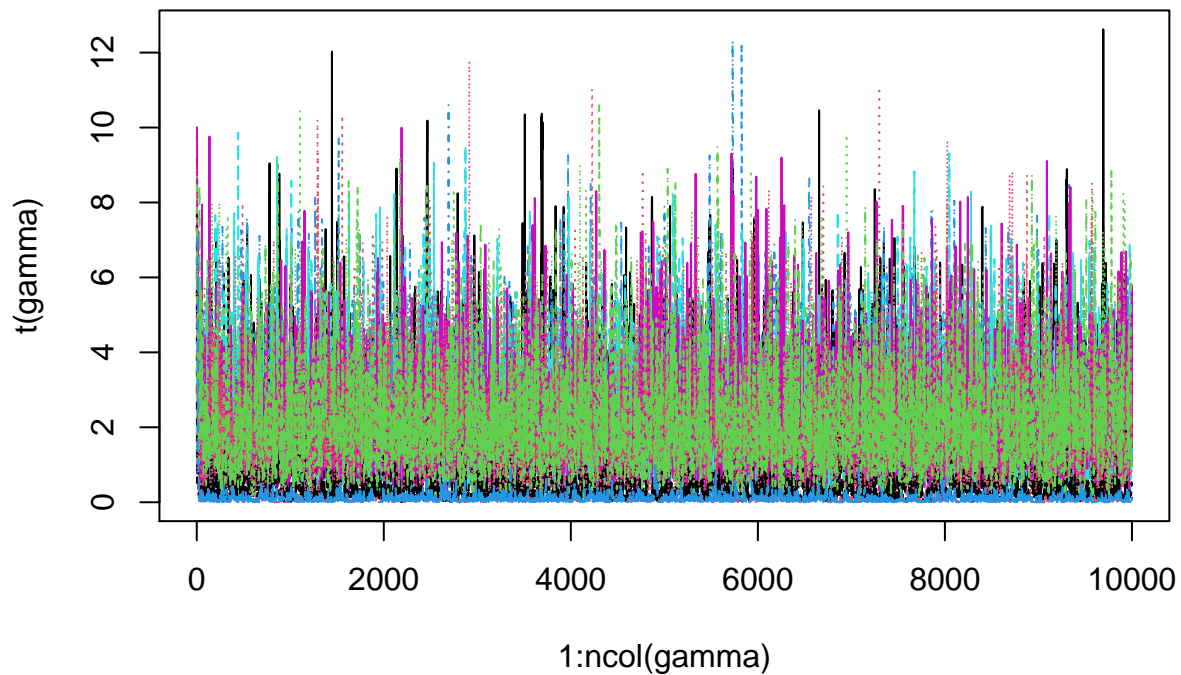
```
plot(GDFMM$lambda*exp(-GDFMM$log_sum), type = 'l', main = "Poisson parameter")
```



```
#gamma.s
gamma = GDFMM$gamma
post_mean_gamma = rowMeans(gamma)
post_mean_gamma

## [1] 2.12448792 0.09386491 2.02875061 2.07250106 2.17891584 2.11400138
## [7] 0.35511539 2.09538812 2.13078240 0.10407124

matplot(x = 1:ncol(gamma), y = t(gamma), type = 'l')
```



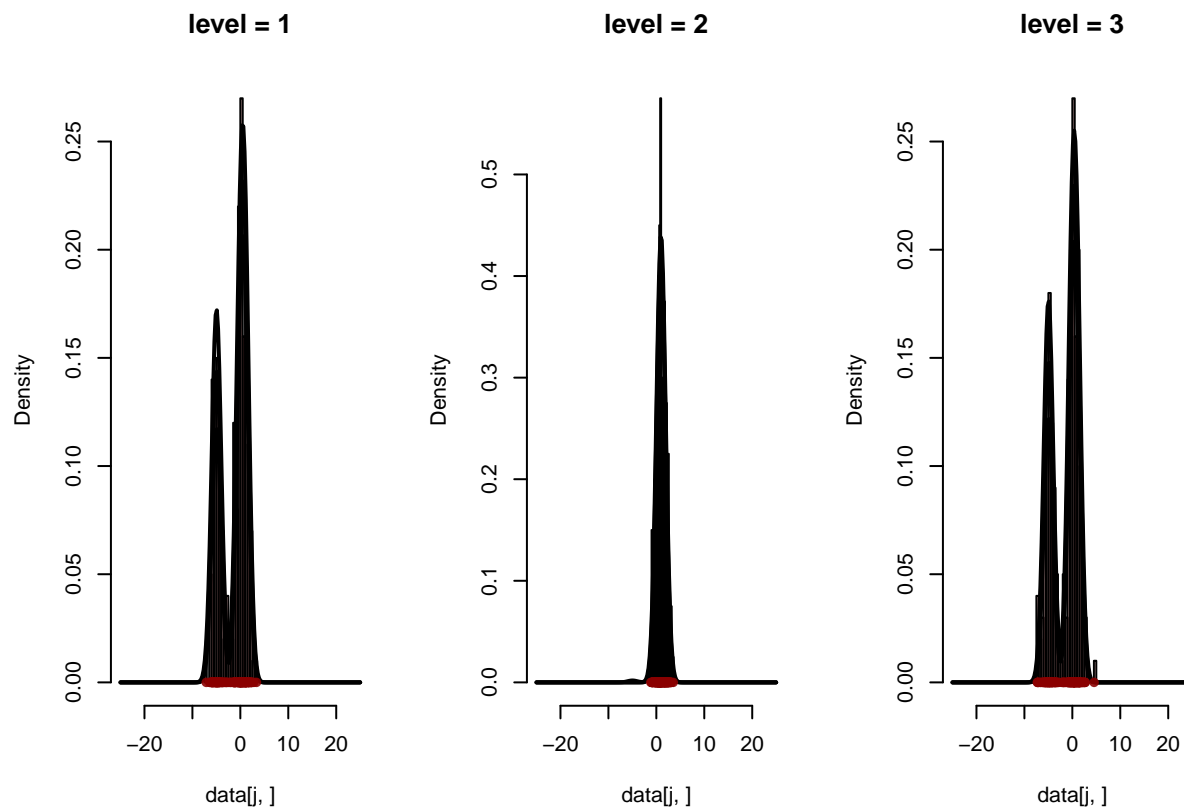
```

# Predictive -----

l_grid = 200
grid = seq(-25,25,length.out = l_grid)

# Predictive in all groups
Pred_all = predictive_all_groups(grid = grid, fit = GDFMM)
par(mfrow = c(1,3))
for(j in 1:3){
  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))
}

```



Dopo poche iterazioni, si assesta attorno a 15 o 20.

$$\nu = 0.70, \gamma_0 = 1$$

```
niter <- 10000
burnin <- 1
thin <- 1

option<-set_options("nu" = 0.7, "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" = T, "UpdateGamma" = T, "UpdateS" = T,
  "UpdateTau" = T, "UpdateLambda" = T, "partition" = real_partition
)

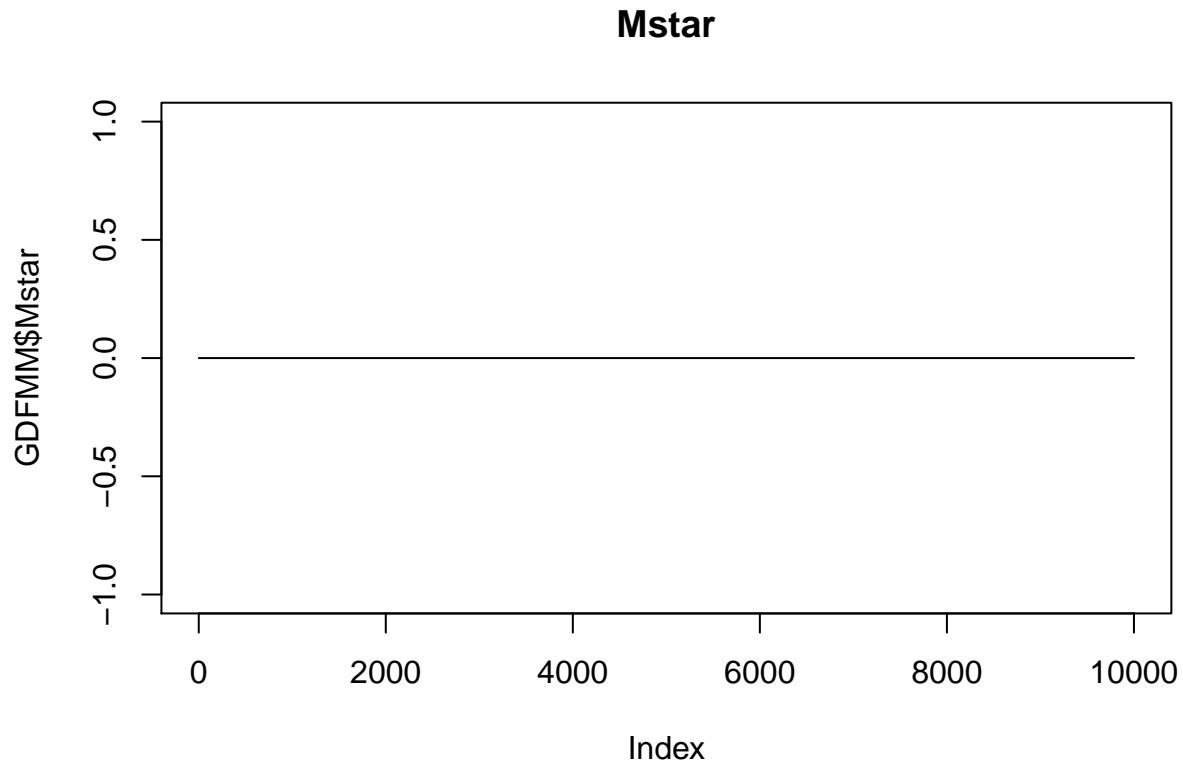
GDFMM = GDFMM_sampler(data, niter, burnin, thin, seed = 123, FixPartition = T, option = option)

##
## Check that provided partition is well formed. It must start from 0 and all values must be contiguous
## GibbsSampler.cpp - Initialization
## initialize_Partition with non empty partition_vec
## Watch out modification: Mstar is not set to zero but to Mstar0
## (K, Mstar, M) = (3,3,6)

#Mstar
summary(GDFMM$Mstar)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0       0       0       0       0       0
```

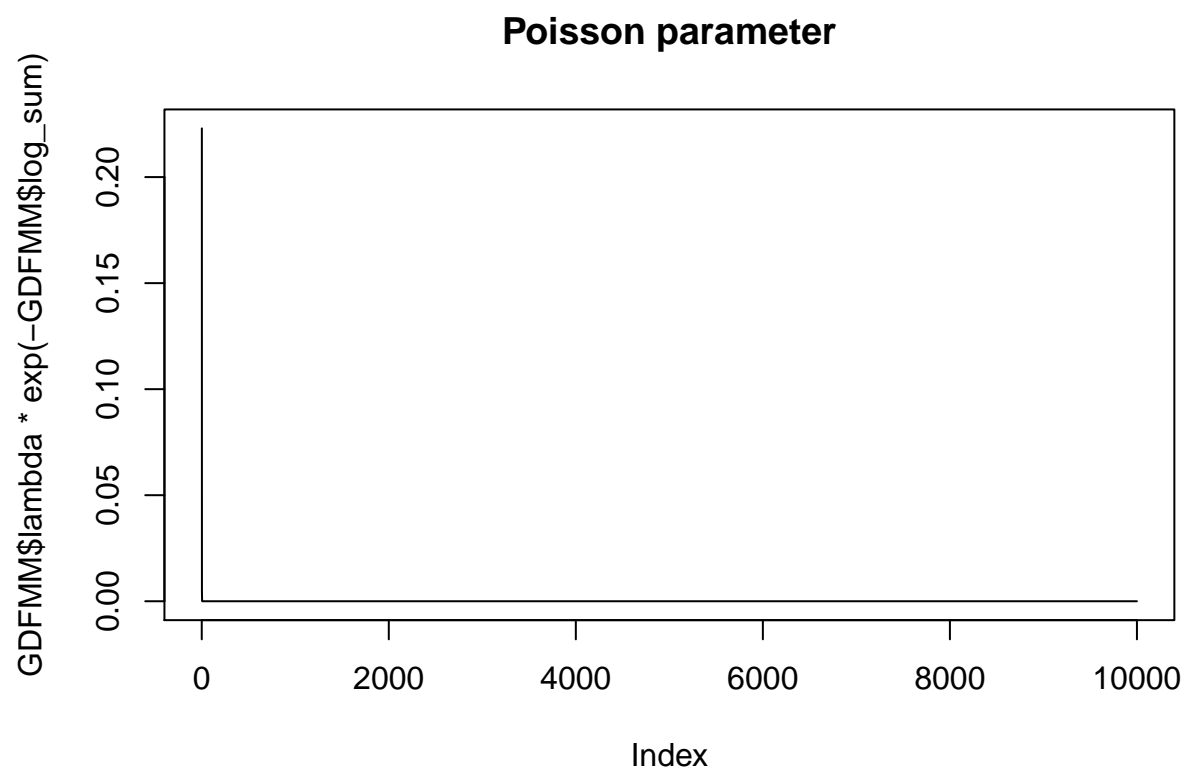
```
plot(GDFMM$Mstar, type = 'l', main = "Mstar")
```



```
#Parametro Poisson: lambda * exp(-log_sum)
summary(GDFMM$lambda*exp(-GDFMM$log_sum))
```

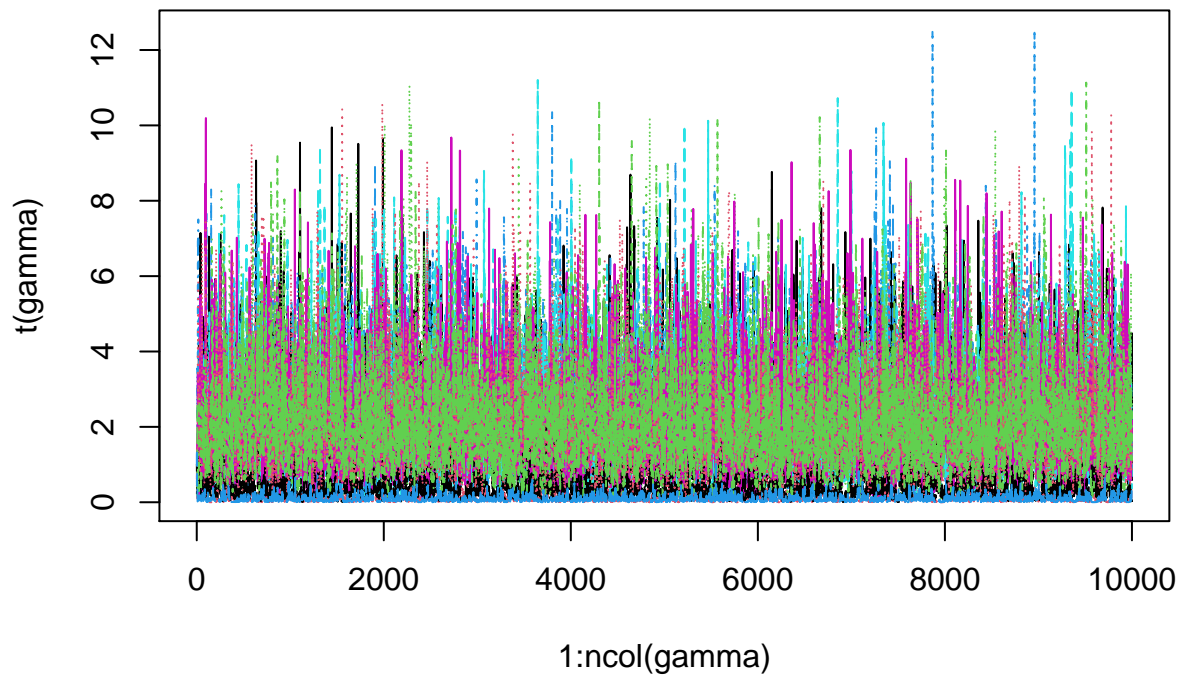
```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.000e+00 0.000e+00 0.000e+00 2.295e-05 0.000e+00 2.230e-01
```

```
plot(GDFMM$lambda*exp(-GDFMM$log_sum), type = 'l', main = "Poisson parameter")
```



```
#gamma.s
gamma = GDFMM$gamma
post_mean_gamma = rowMeans(gamma)
post_mean_gamma

## [1] 2.07539249 0.09212319 2.06284642 2.07099712 2.16693369 2.14944435
## [7] 0.34978824 2.09361469 2.13996121 0.09097763
matplot(x = 1:ncol(gamma), y = t(gamma), type = 'l')
```



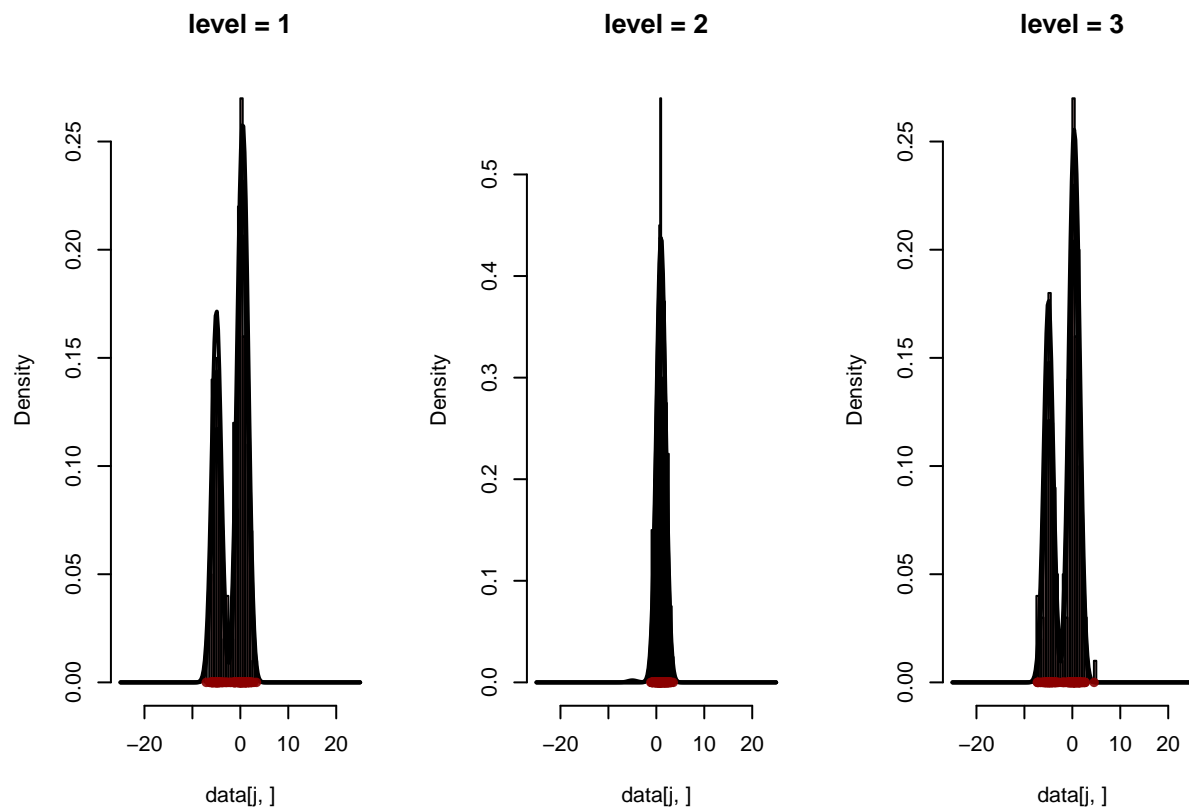
```

# Predictive -----

l_grid = 200
grid = seq(-25,25,length.out = l_grid)

# Predictive in all groups
Pred_all = predictive_all_groups(grid = grid, fit = GDFMM)
par(mfrow = c(1,3))
for(j in 1:3){
  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))
}

```

Simile a prima ma con un burnin molto più lungo.

$$\nu = 1/10$$

```
niter <- 10000
burnin <- 1
thin <- 1

option<-set_options("nu" = 1/10, "Mstar0" = 3, "Lambda0" = 3, "mu0" = 0, "sigma0" = 1, "gamma0" = 0.01,
  "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" = T, "UpdateGamma" = T, "UpdateS" = T,
  "UpdateTau" = T, "UpdateLambda" = T, "partition" = real_partition
)

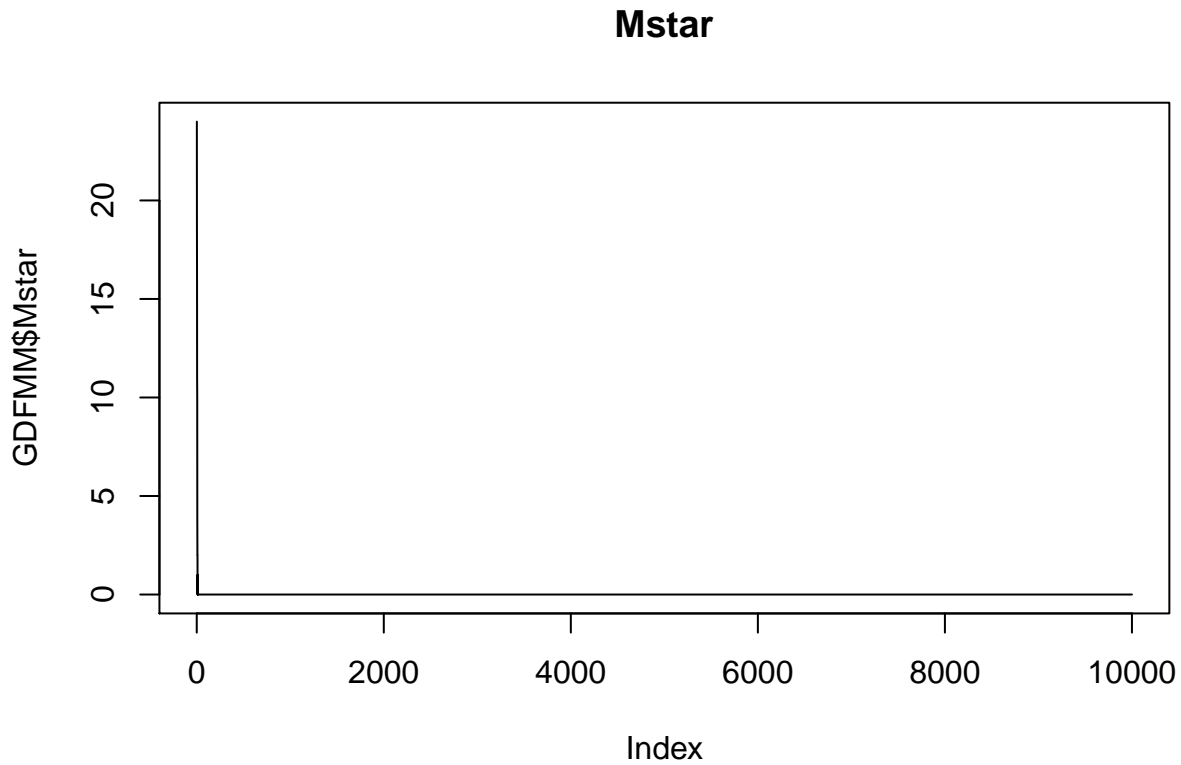
GDFMM = GDFMM_sampler(data, niter, burnin, thin, seed = 123, FixPartition = T, option = option)

##
## Check that provided partition is well formed. It must start from 0 and all values must be contiguous
## GibbsSampler.cpp - Initialization
## initialize_Partition with non empty partition_vec
## Watch out modification: Mstar is not set to zero but to Mstar0
## (K, Mstar, M) = (3,3,6)

#Mstar
summary(GDFMM$Mstar)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.000  0.000   0.000   0.009  0.000   24.000
```

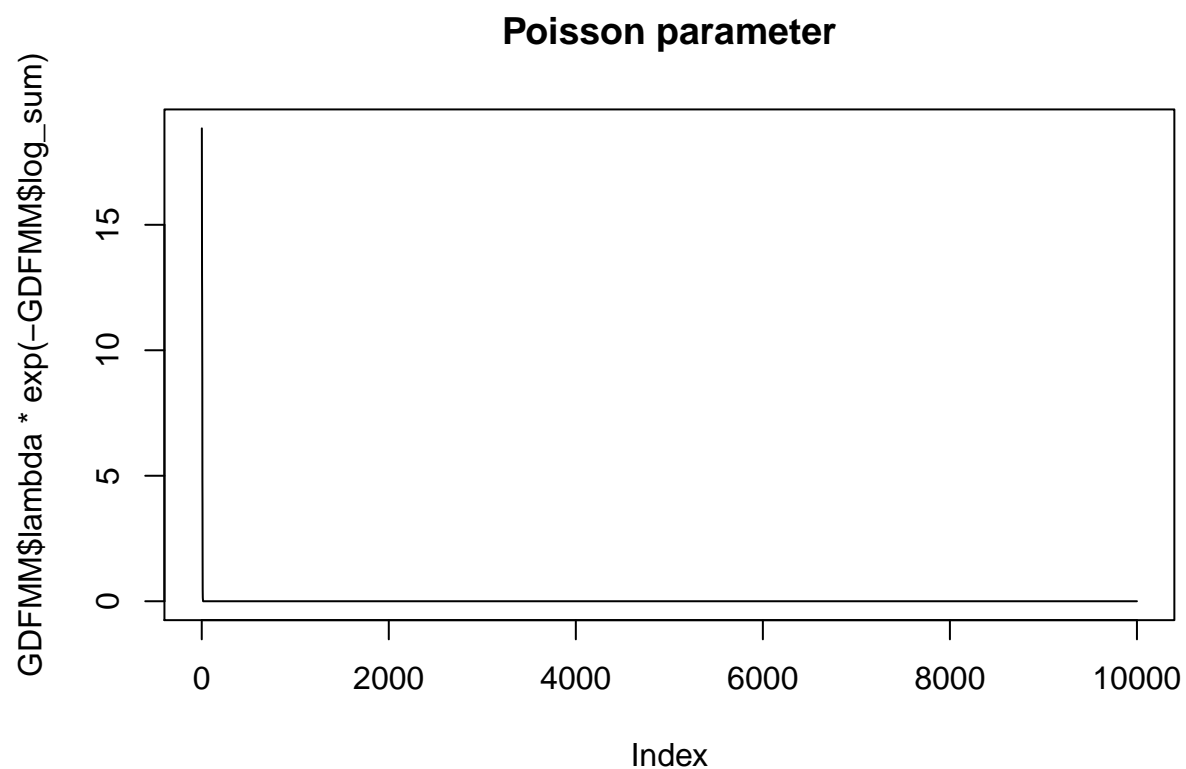
```
plot(GDFMM$Mstar, type = 'l', main = "Mstar")
```



```
#Parametro Poisson: lambda * exp(-log_sum)
summary(GDFMM$lambda*exp(-GDFMM$log_sum))
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.000000 0.000000 0.000000 0.009092 0.000000 18.842386
```

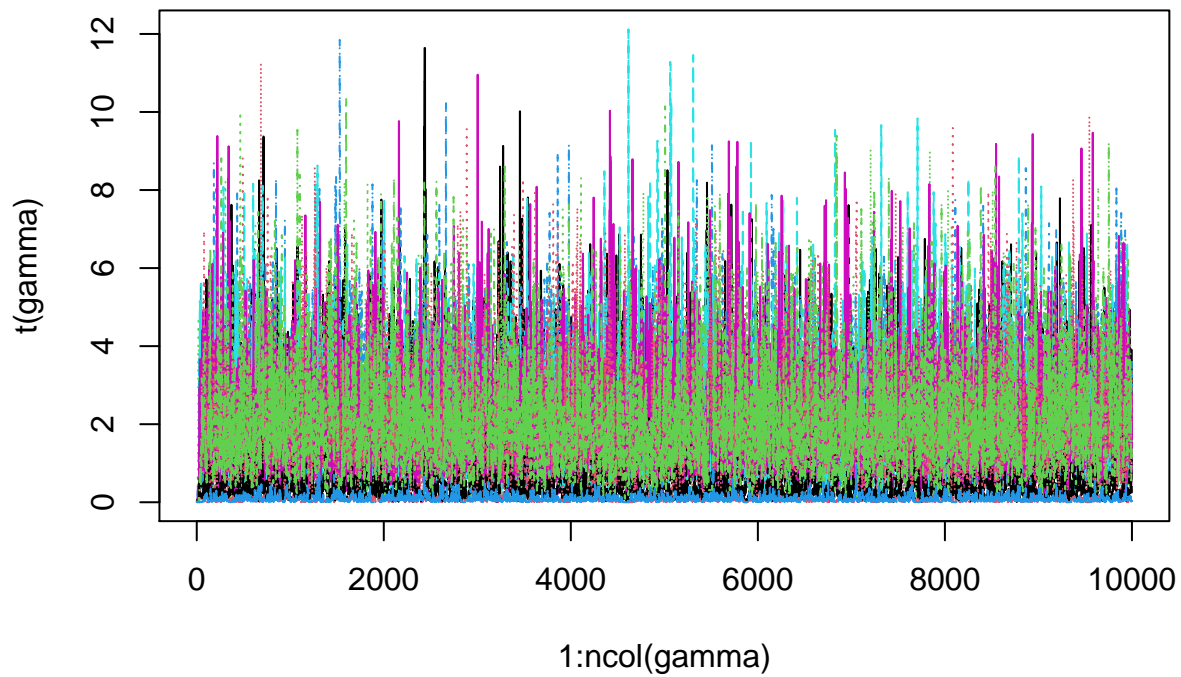
```
plot(GDFMM$lambda*exp(-GDFMM$log_sum), type = 'l', main = "Poisson parameter")
```



```
#gamma.s
gamma = GDFMM$gamma
post_mean_gamma = rowMeans(gamma)
post_mean_gamma

## [1] 2.05035215 0.08274475 2.00323627 2.04485304 2.14551010 2.16624475
## [7] 0.34753815 2.04681731 2.07042859 0.08689060

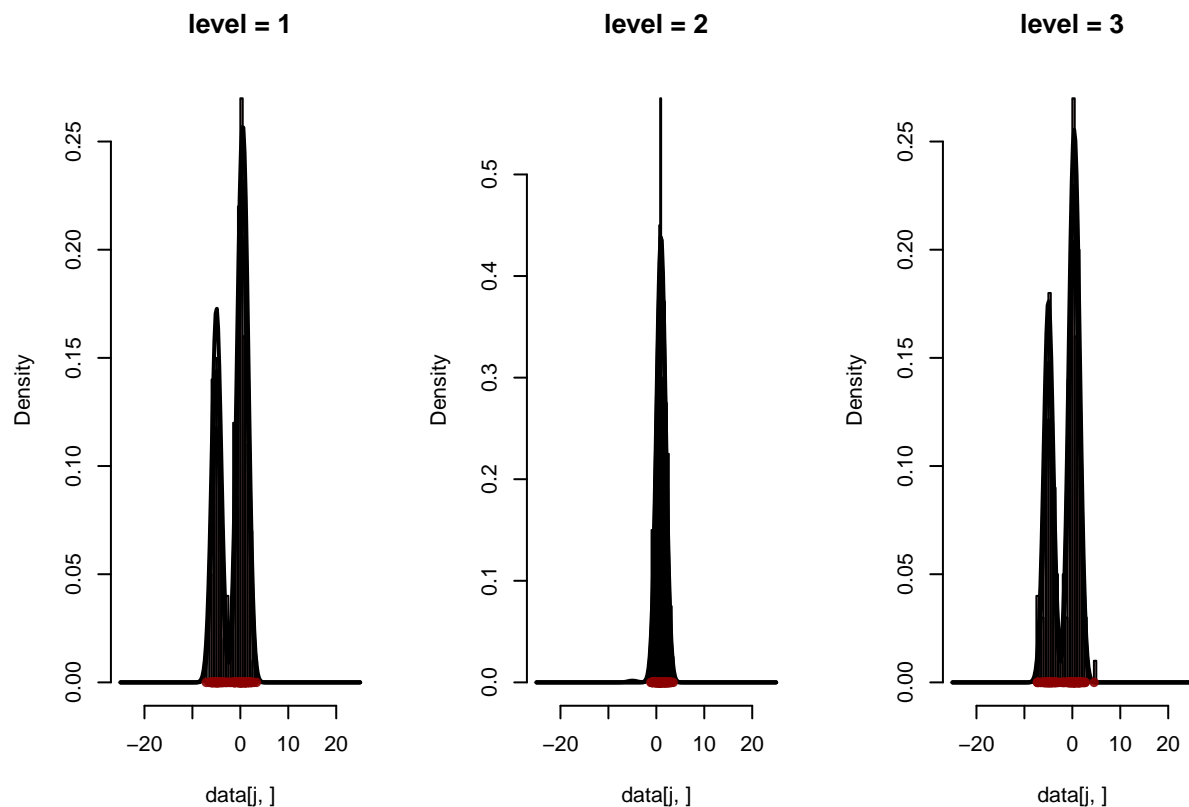
matplot(x = 1:ncol(gamma), y = t(gamma), type = 'l')
```



```
# Predictive -----

l_grid = 200
grid = seq(-25,25,length.out = l_grid)

# Predictive in all groups
Pred_all = predictive_all_groups(grid = grid, fit = GDFMM)
par(mfrow = c(1,3))
for(j in 1:3){
  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))
}
```



Così è molto più stabile, non dipende più dal valore di γ_0 (a meno che non si mette il valore iniziale molto grande, tipo 1000).

$\nu = 1/100$

```
niter <- 5000
burnin <- 5000
thin <- 1

option<-set_options("nu" = 1/100, "Mstar0" = 3, "Lambda0" = 3, "mu0" = 0, "sigma0" = 1, "gamma0" = 100,
  "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" = T, "UpdateGamma" = T, "UpdateS" = T,
  "UpdateTau" = T, "UpdateLambda" = T, "partition" = real_partition
)

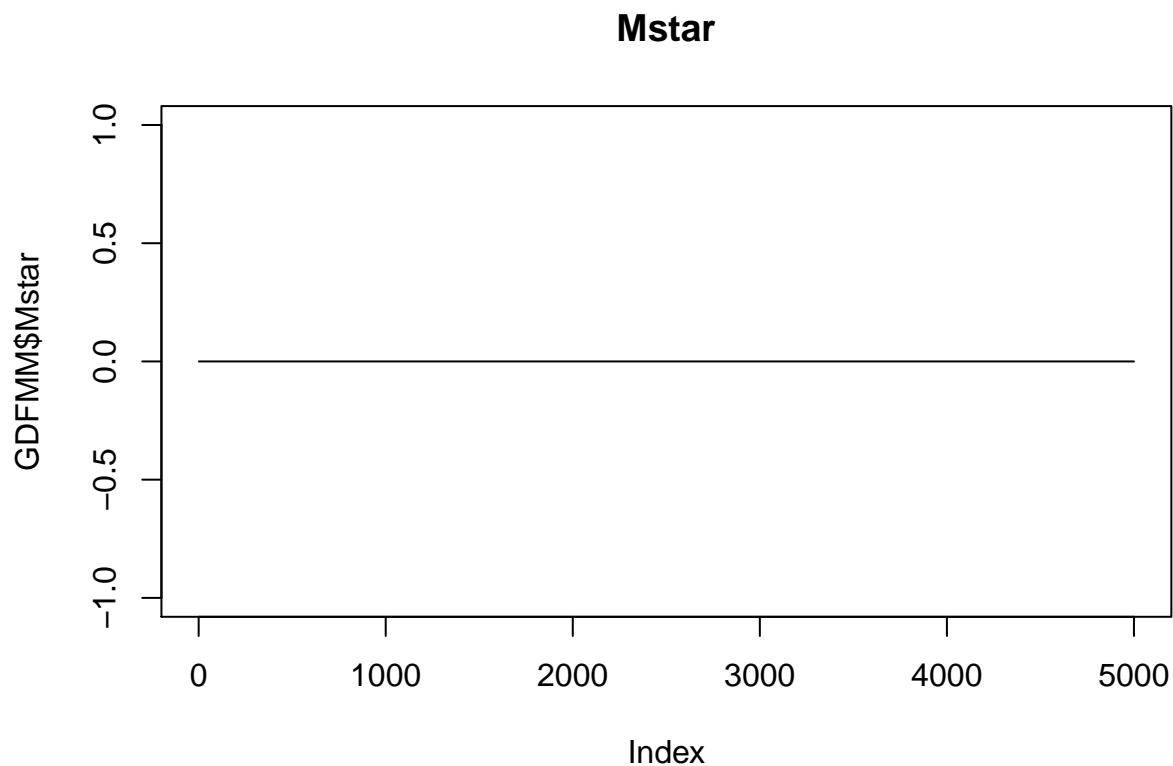
GDFMM = GDFMM_sampler(data, niter, burnin, thin, seed = 123, FixPartition = T, option = option)

##
## Check that provided partition is well formed. It must start from 0 and all values must be contiguous
## GibbsSampler.cpp - Initialization
## initialize_Partition with non empty partition_vec
## Watch out modification: Mstar is not set to zero but to Mstar0
## (K, Mstar, M) = (3,3,6)
```

```
#Mstar
summary(GDFMM$Mstar)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##         0         0         0         0         0         0
```

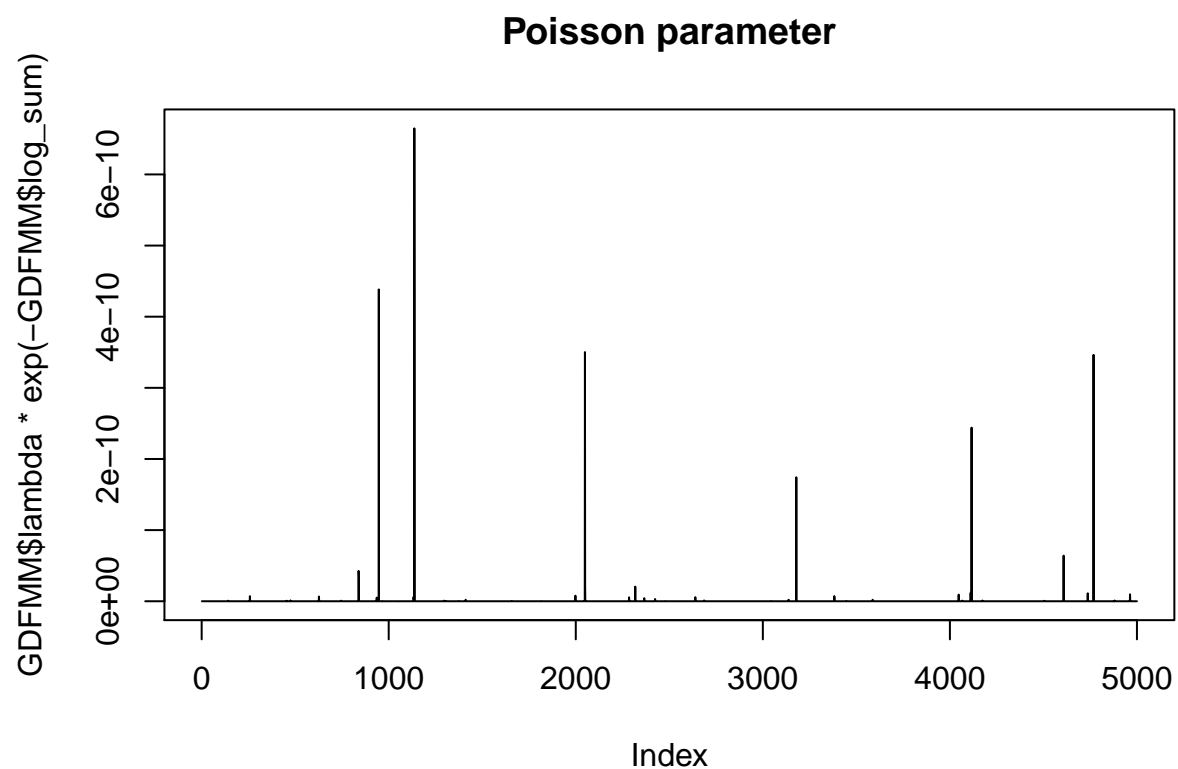
```
plot(GDFMM$Mstar, type = 'l', main = "Mstar")
```



```
#Parametro Poisson: lambda * exp(-log_sum)
summary(GDFMM$lambda*exp(-GDFMM$log_sum))
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.000e+00 0.000e+00 0.000e+00 5.930e-13 0.000e+00 6.648e-10
```

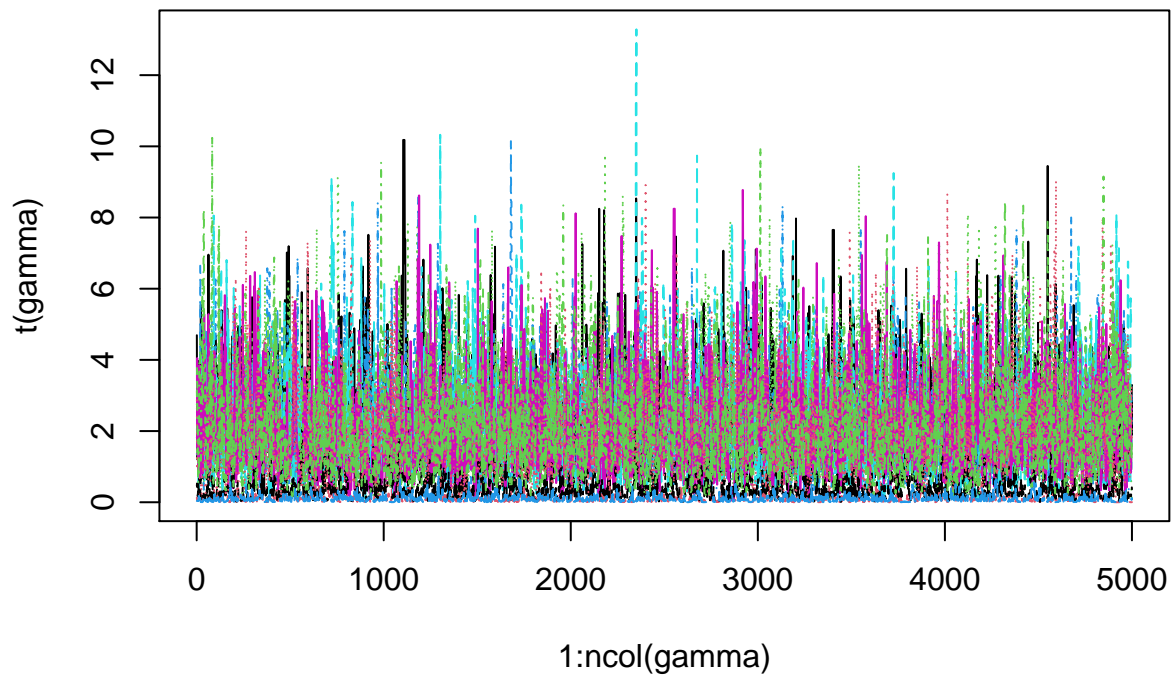
```
plot(GDFMM$lambda*exp(-GDFMM$log_sum), type = 'l', main = "Poisson parameter")
```



```
#gamma.s
gamma = GDFMM$gamma
post_mean_gamma = rowMeans(gamma)
post_mean_gamma

## [1] 2.11387439 0.09477742 2.08148741 1.98697735 2.13929128 2.11934193
## [7] 0.35210858 1.96929892 2.04167690 0.09649607

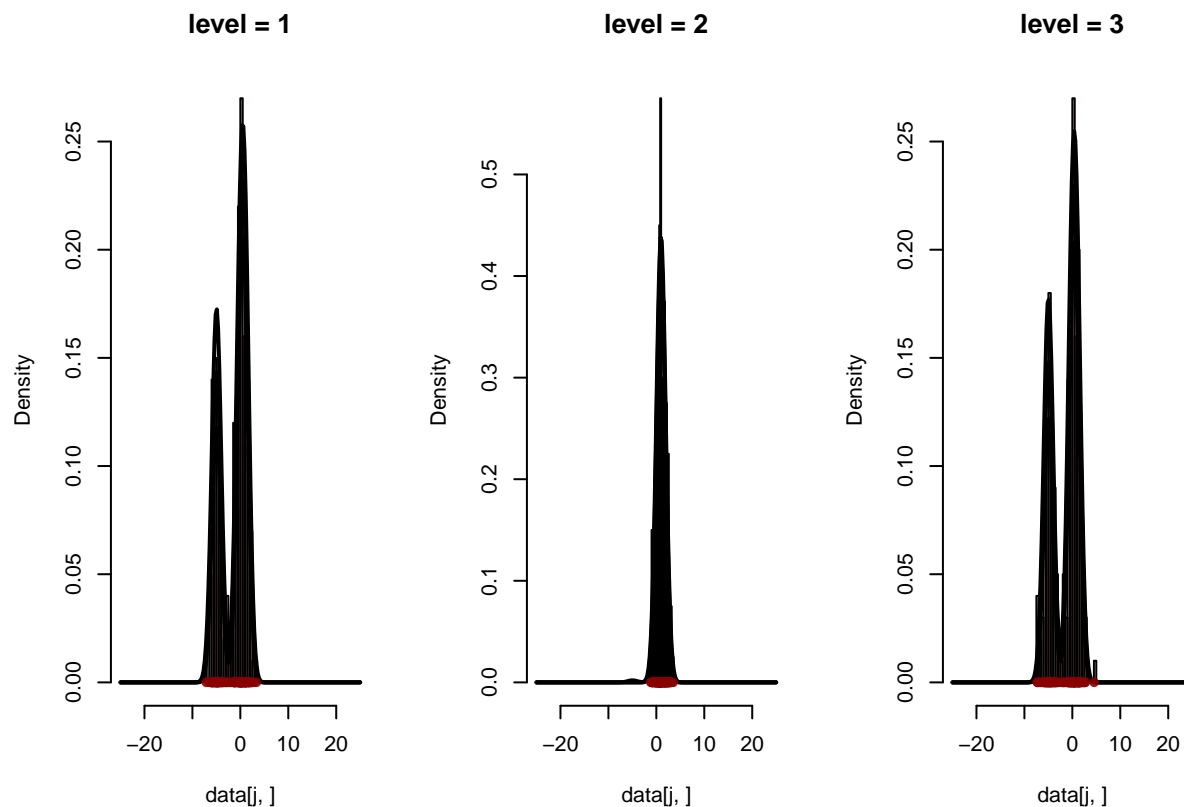
matplot(x = 1:ncol(gamma), y = t(gamma), type = 'l')
```



```
# Predictive -----

l_grid = 200
grid = seq(-25,25,length.out = l_grid)

# Predictive in all groups
Pred_all = predictive_all_groups(grid = grid, fit = GDFMM)
par(mfrow = c(1,3))
for(j in 1:3){
  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))
}
```

Qua l'effetto del valore iniziale è decisamente mitigato, inoltre la fase per arrivare alla convergenza della catenza è un po' più smooth, non è più a scalino.

Plot funzione

Voglio capire come sono fatte le funzioni $f(u, \gamma) = \frac{1}{(u+1)^\gamma}$. Inizio con un plot delle linee di livello. Si vede che se la U è maggiore di 1, allora la gamma deve essere davvero piccolissima per avere dei valori vicini ad 1 (NB, ricordarsi che questo è solo uno dei d termini che sono coinvolti, il coefficiente che vediamo noi è il prodotto di d termini fatti così). Stessa cosa per quando $\gamma > 1$. Diciamo che ci si aspetta che entrambi i valori siano minori di 1.

```
f = function(l){
  u = l[[1]]
  gamma = l[[2]]
  return( 1/(1+u)^gamma )
}

Umin = 0
Umax = 4
gamma_min = 0.001
gamma_max = 4

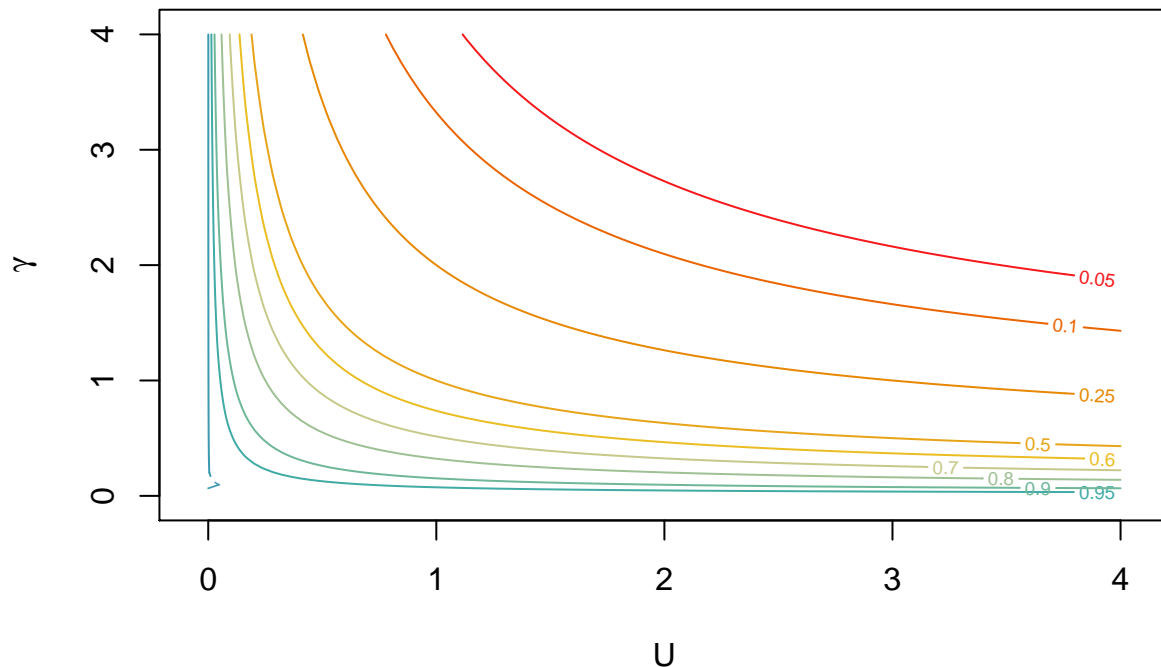
U <- seq(Umin,Umax,length=100)
gamma <- seq(gamma_min,gamma_max,length=100)
```

```

grid <- expand.grid(U, gamma)
f_values <- apply(grid, 1, f)
f_values <- matrix(f_values, nrow = length(U),
                  ncol = length(gamma), byrow=F)

# plot with confidence levels
#levels <- seq(0,1,length.out = 15)
levels <- c(1,0.95,0.9,0.8,0.7,0.6,0.5,0.25,0.1,0.05)
my_col = hcl.colors (n= length( levels ), palette = " Zissou1 ")
contour(U, gamma, f_values, col = my_col,
        levels = levels, labels = stringr::str_trunc(as.character(levels), width = 4, ellipsis = "..."),
        xlab="U", ylab=expression(gamma),
        xlim = c(Umin-0.05,Umax+0.05), ylim = c(gamma_min-0.05,gamma_max+0.05)
        )

```



Provo ad indagare valori più vicini a zero. Qui è per $U \in (0, 0.5)$.

```

Umin = 0
Umax = 0.5
gamma_min = 0.001
gamma_max = 4

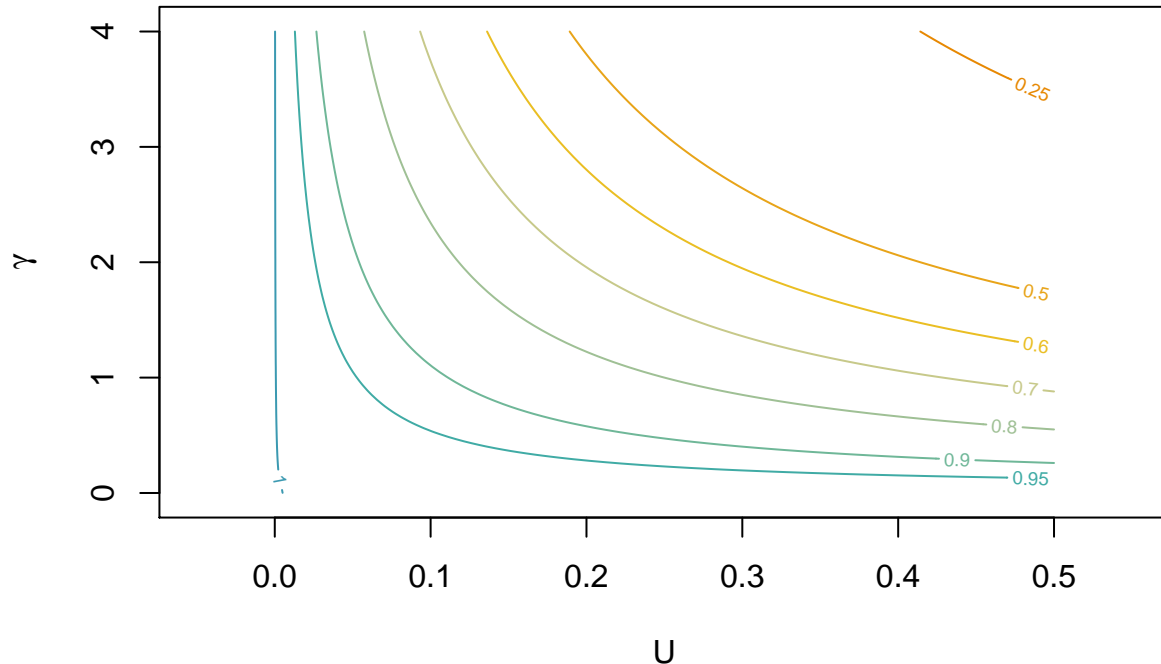
U <- seq(Umin,Umax,length=100)
gamma <- seq(gamma_min,gamma_max,length=100)

grid <- expand.grid(U, gamma)

```

```
f_values <- apply(grid, 1, f)
f_values <- matrix(f_values, nrow = length(U),
                  ncol = length(gamma), byrow=F)

# plot with confidence levels
#levels <- seq(0,1,length.out = 15)
levels <- c(1,0.95,0.9,0.8,0.7,0.6,0.5,0.25,0.1,0.05)
my_col = hcl.colors (n= length( levels ), palette = " Zissou1 ")
contour(U, gamma, f_values, col = my_col,
        levels = levels, labels = stringr::str_trunc(as.character(levels), width = 4, ellipsis = "..."),
        xlab="U", ylab=expression(gamma),
        xlim = c(Umin-0.05,Umax+0.05), ylim = c(gamma_min-0.05,gamma_max+0.05)
        )
```



Provo ad indagare valori più vicini a zero. Qui è per $\gamma \in (0.001, 0.5)$.

```
Umin = 0
Umax = 4
gamma_min = 0.001
gamma_max = 0.5

U <- seq(Umin,Umax,length=100)
gamma <- seq(gamma_min,gamma_max,length=100)

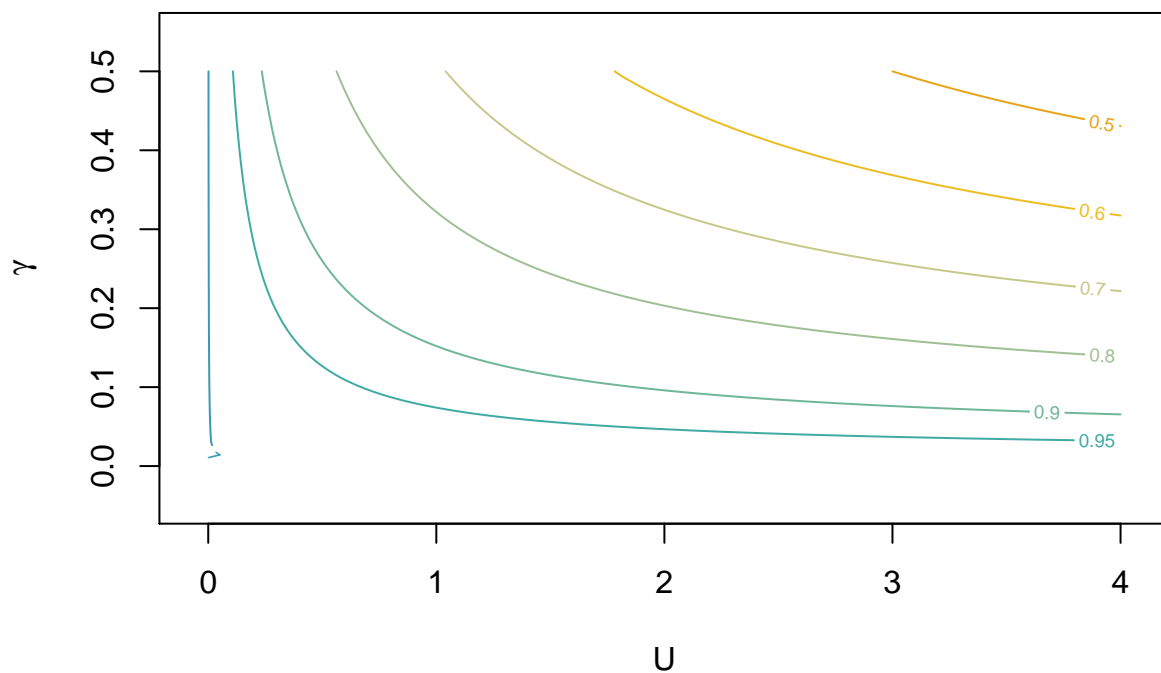
grid <- expand.grid(U, gamma)
f_values <- apply(grid, 1, f)
```

```

f_values <- matrix(f_values, nrow = length(U),
                  ncol = length(gamma), byrow=F)

# plot with confidence levels
#levels <- seq(0,1,length.out = 15)
levels <- c(1,0.95,0.9,0.8,0.7,0.6,0.5,0.25,0.1,0.05)
my_col = hcl.colors (n= length( levels ), palette = " Zissou1 ")
contour(U, gamma, f_values, col = my_col,
        levels = levels, labels = stringr::str_trunc(as.character(levels), width = 4, ellipsis = ""),
        xlab="U", ylab=expression(gamma),
        xlim = c(Umin-0.05,Umax+0.05), ylim = c(gamma_min-0.05,gamma_max+0.05)
        )

```



Faccio i plot uni-dimensional - gamma

```

f_gamma = function(x, u){
  return( 1/(1+u)^x )
}

U = c(0.1,0.25,0.5,1,2,4)
gamma_min = 0.001
gamma_max = 4

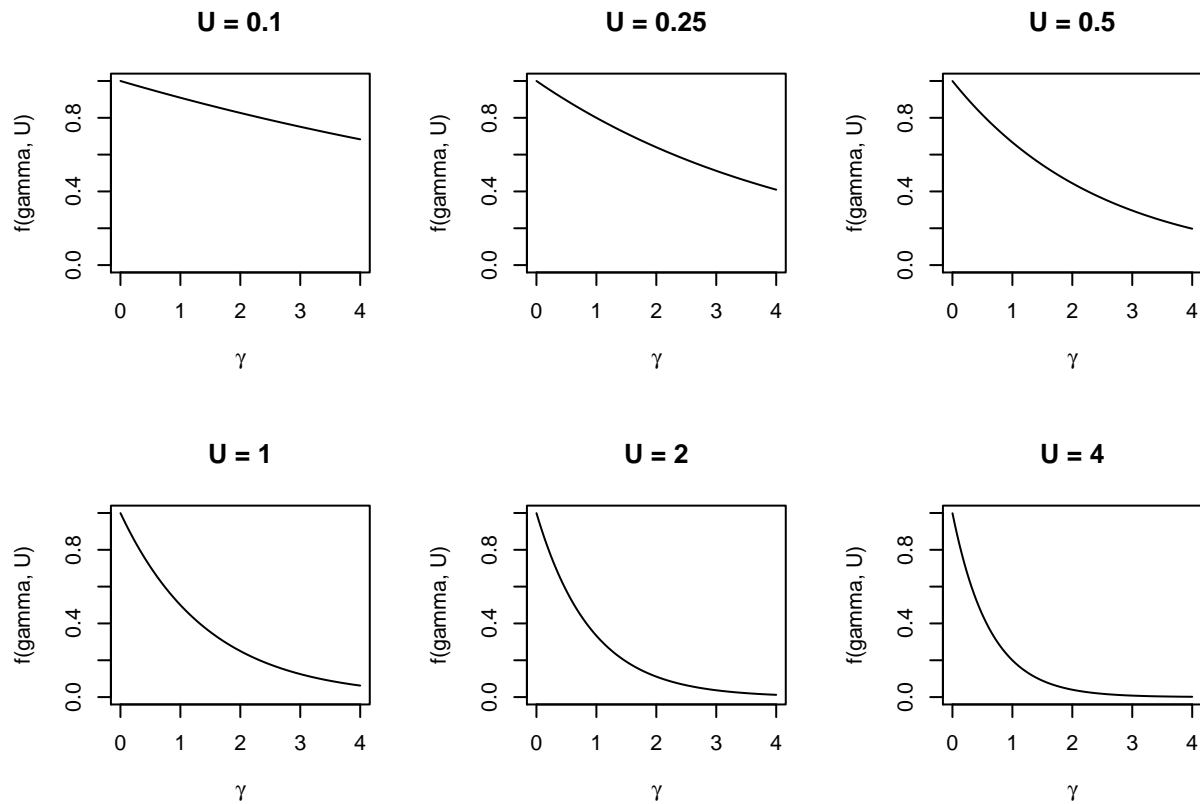
par(mfrow = c(2,3))
for(i in 1:length(U)){
  curve(expr = f_gamma(x = x, u = U[i]), from = gamma_min, to = gamma_max,
        ylab = "f(gamma, U)", xlab = expression(gamma),

```

```

    main = paste0("U = ", U[i]), ylim = c(0,1))
}

```



Faccio i plot uni-dimensionali - U

```

f_u = function(x, gamma){
  return( 1/(1+x)^gamma )
}

gamma = c(0.01,0.25,0.5,1,2,4)
Umin = 0
Umax = 4

par(mfrow = c(2,3))
for(i in 1:length(U)){
  curve(expr = f_u(x = x, gamma = gamma[i]), from = Umin, to = Umax,
        ylab = "f(gamma, U)", xlab = expression(U),
        main = paste(expression(gamma), " = ", gamma[i]), ylim = c(0,1))
}

```

