Rmd_Introducing_nu

Partizione fissa

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

Let us consider $S_{jm} \sim Gamma(\gamma_j, 1)$, $S'_{jm} \sim Gamma(\gamma_j, \nu)$. 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 transorm, 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)

```r
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 me
                          # values are collected level by level, so first all the values in level 1, th
                          # 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]] ],
```

```r
                            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

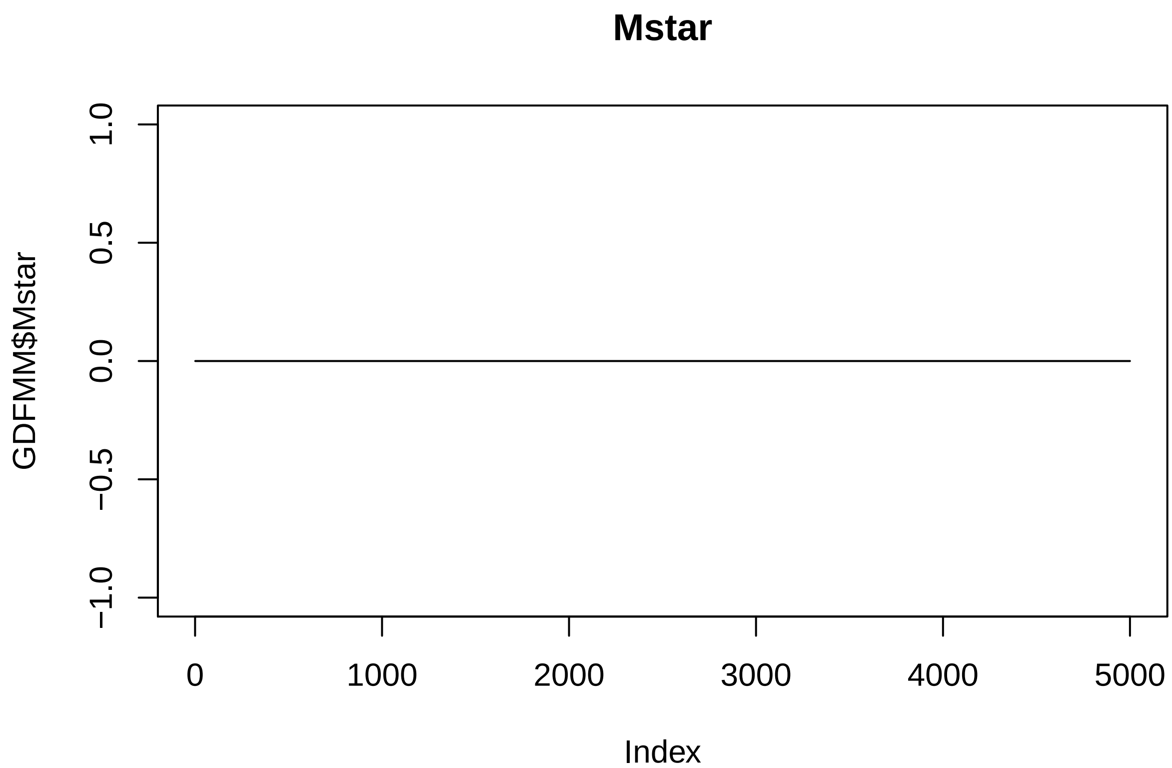
```r
niter  <- 5000
burnin <- 1
thin   <- 1

option<-list("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
## initialize_Partition with non empty partition_vec
## Watch out modification: Mstar is not set to zero but to Mstar0
##  (K, Mstar, M) = (3,3,3)
## Chiamato initialize_S con gs_engine, mette casuale!
```

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

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

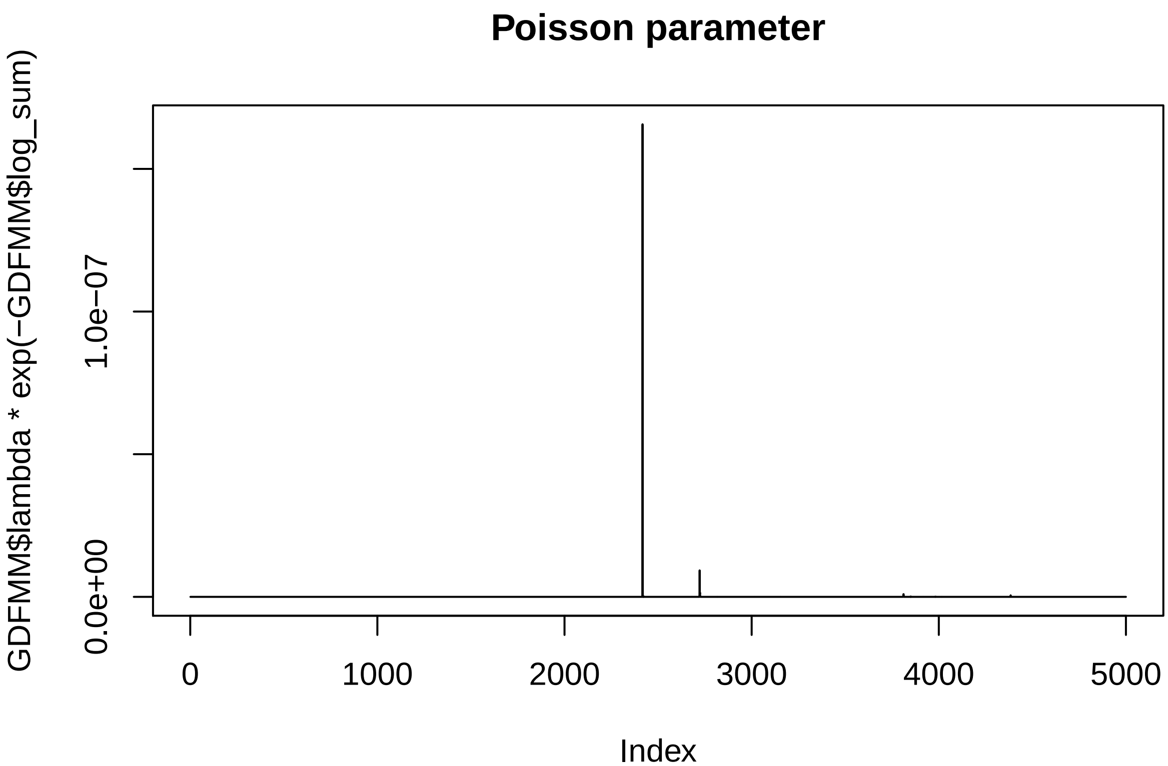
```r
plot(GDFMM$Mstar, type = 'l', main = "Mstar")
```




```r
#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 3.610e-11 0.000e+00 1.656e-07
```

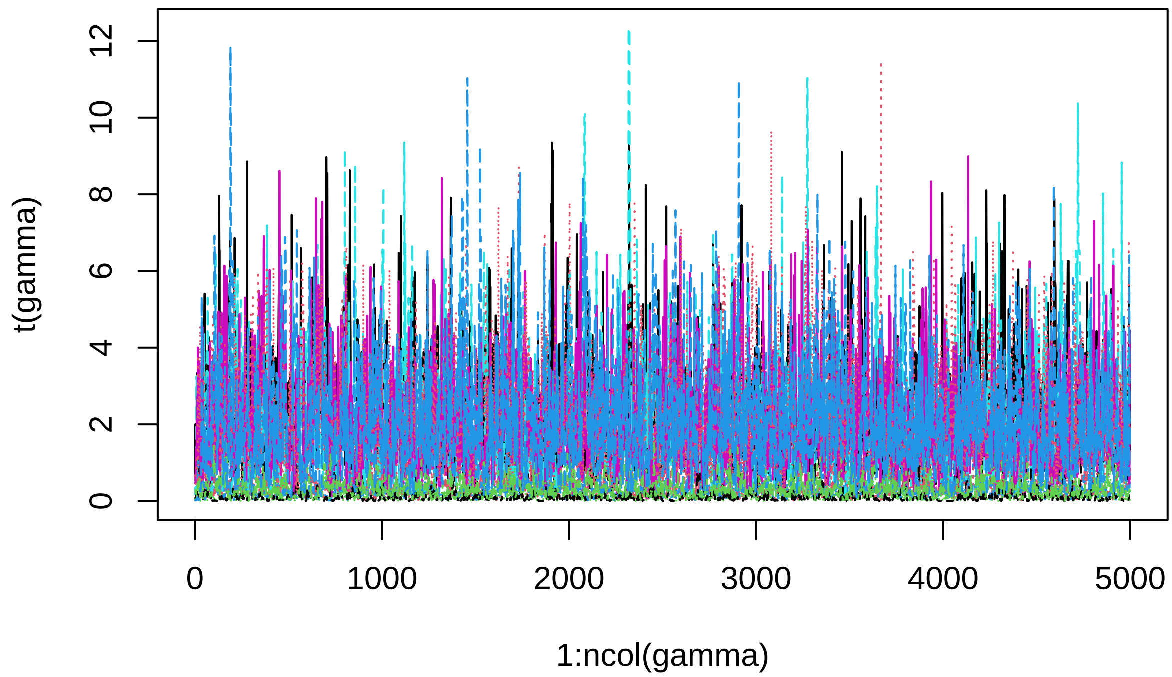
```r
plot(GDFMM$lambda*exp(-GDFMM$log_sum), type = 'l', main = "Poisson parameter")
```



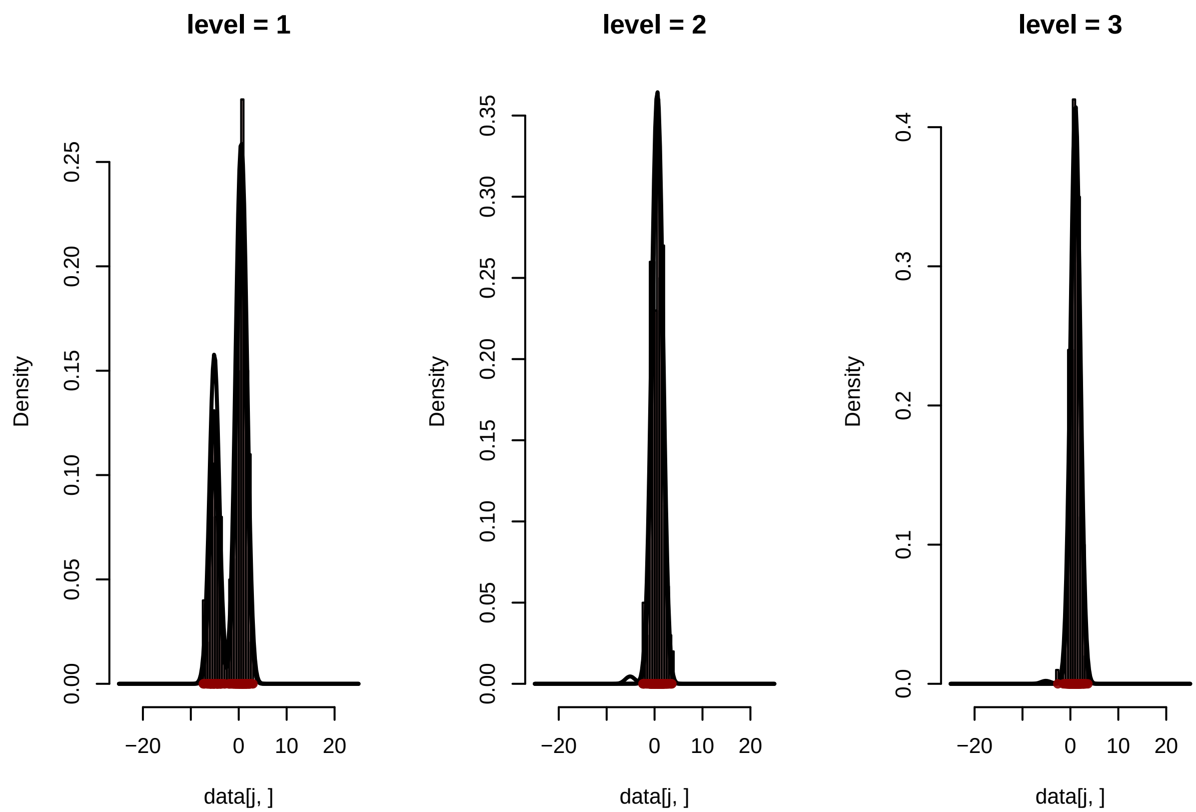

```r
#gammas
gamma = GDFMM$gamma
post_mean_gamma = rowMeans(gamma)
post_mean_gamma
```

```
##  [1] 2.11642664 0.34818014 0.10119906 0.35012935 2.10226729 2.07496146
##  [7] 0.09152449 2.04074233 0.33918290 2.11280495
```

```r
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)

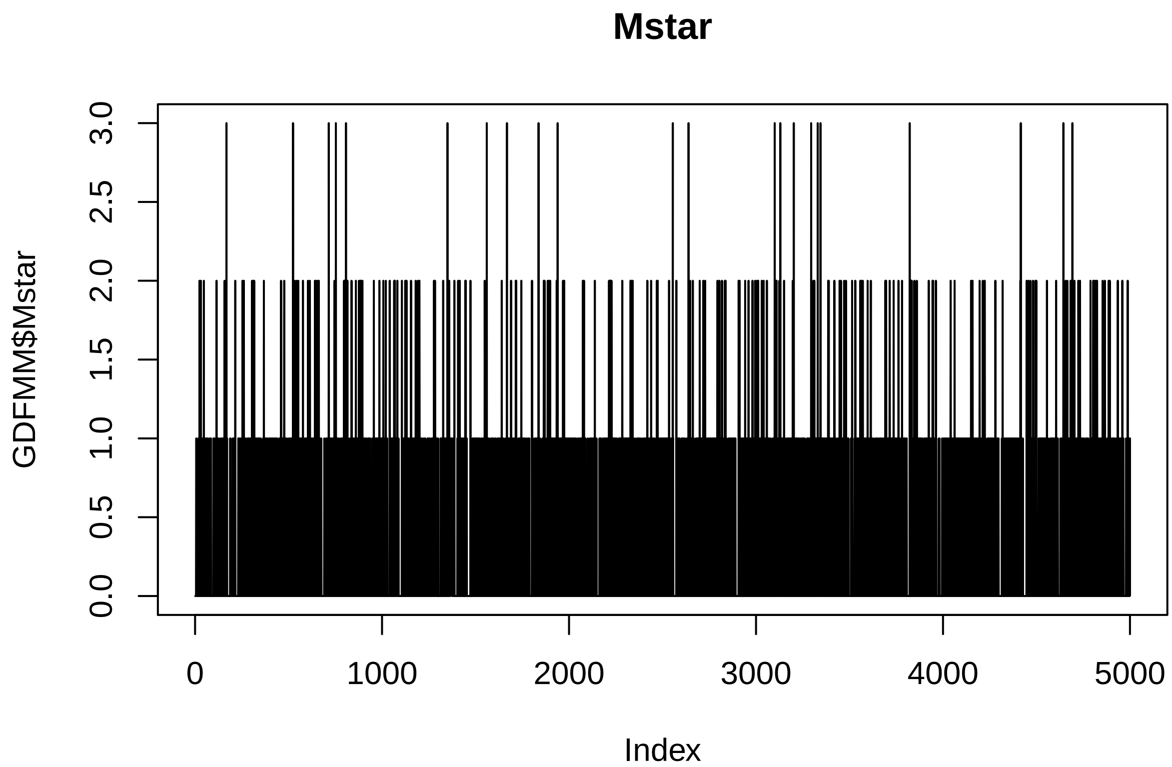
```
niter  <- 5000
burnin <- 1
thin   <- 1

option<-list("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 contiguou
## initialize_Partition with non empty partition_vec
## Watch out modification: Mstar is not set to zero but to Mstar0
##  (K, Mstar, M) = (3,3,3)
## Chiamato initialize_S con gs_engine, mette casuale!
```

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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.3302  1.0000  3.0000
```

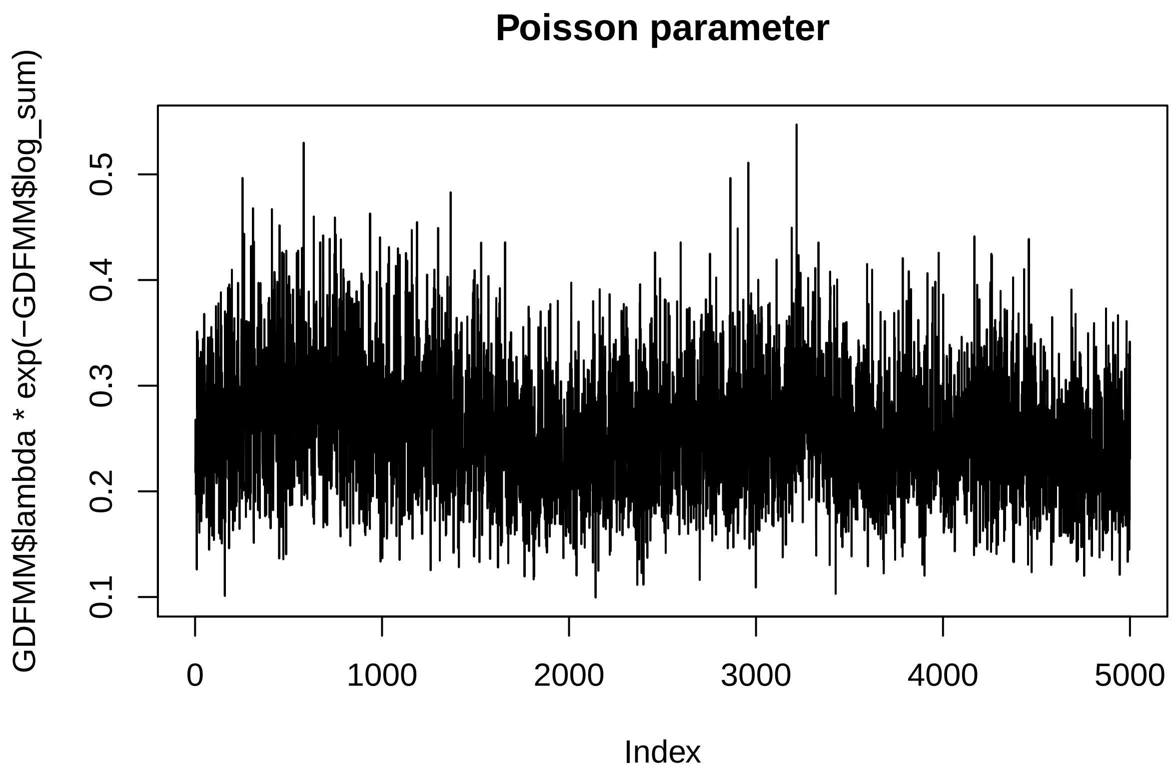
```r
plot(GDFMM$Mstar, type = 'l', main = "Mstar")
```



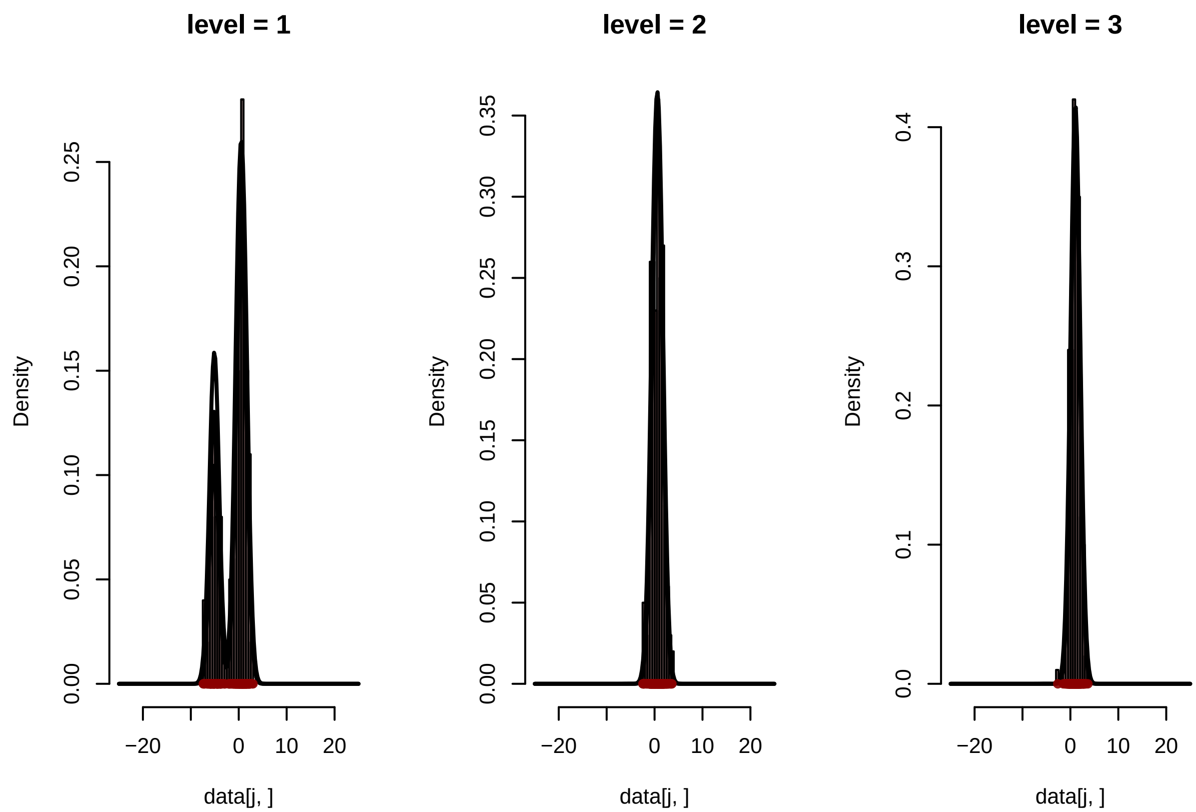

```r
#Parametro Poisson: lambda * exp(-log_sum)
summary(GDFMM$lambda*exp(-GDFMM$log_sum))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.09942 0.21359 0.25113 0.25619 0.29342 0.54724
```

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




```r
#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.

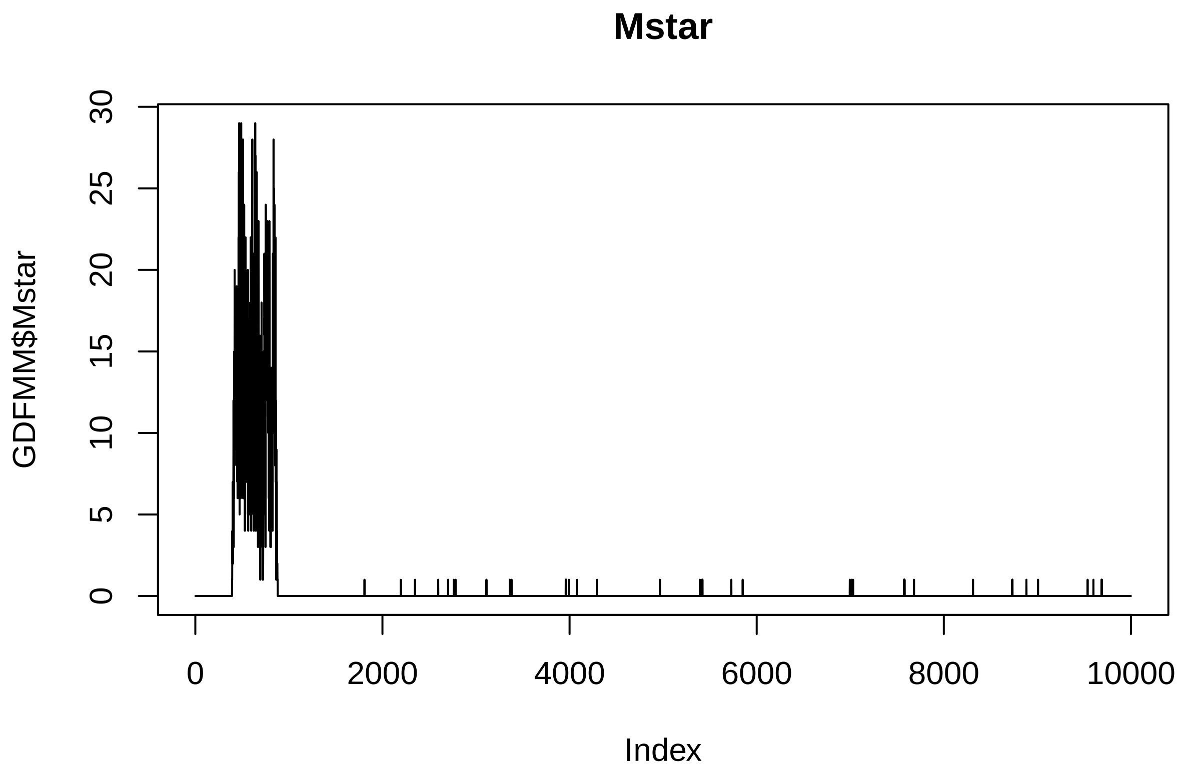
```
niter  <- 10000
burnin <- 1
thin   <- 1

option<-list("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
## initialize_Partition with non empty partition_vec
## Watch out modification: Mstar is not set to zero but to Mstar0
##  (K, Mstar, M) = (3,3,3)
## Chiamato initialize_S con gs_engine, mette casuale!
```

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

```
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.5962  0.0000 29.0000
```

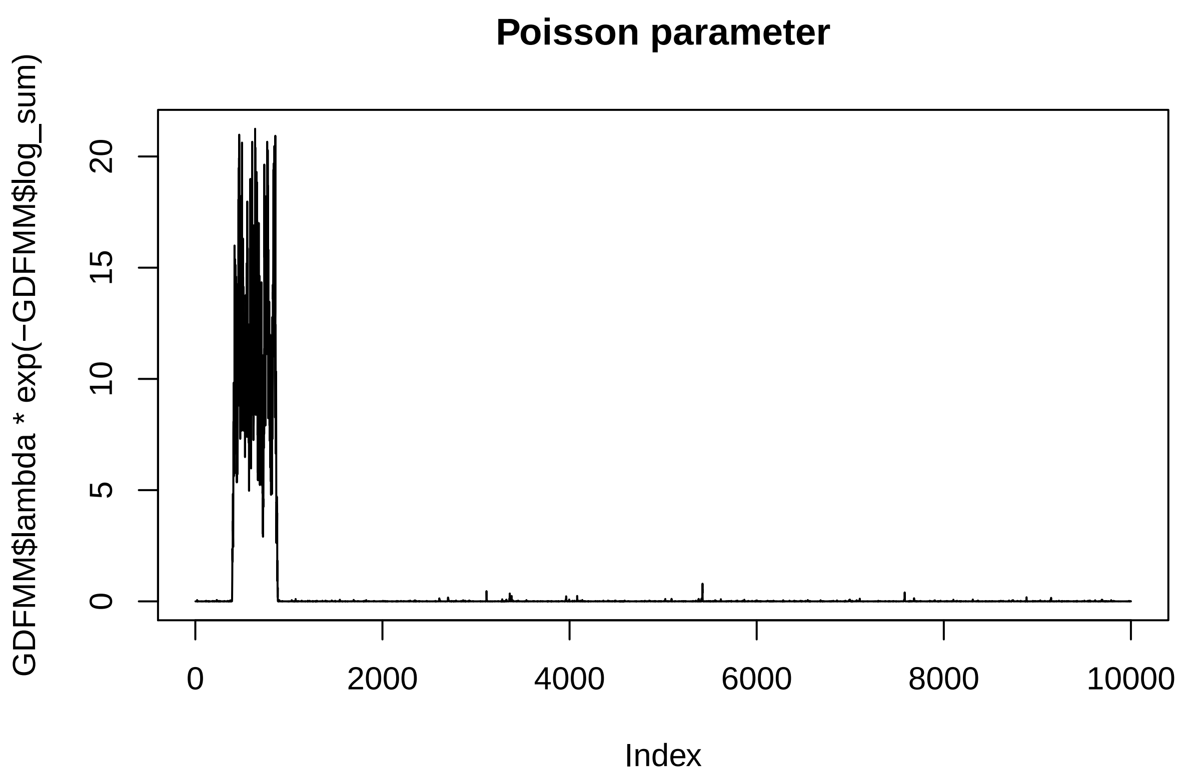
```r
plot(GDFMM$Mstar, type = 'l', main = "Mstar")
```




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

```
##      Min.    1st Qu.    Median      Mean    3rd Qu.      Max.
##  0.000000  0.000025  0.000202  0.543893  0.001621 21.245085
```

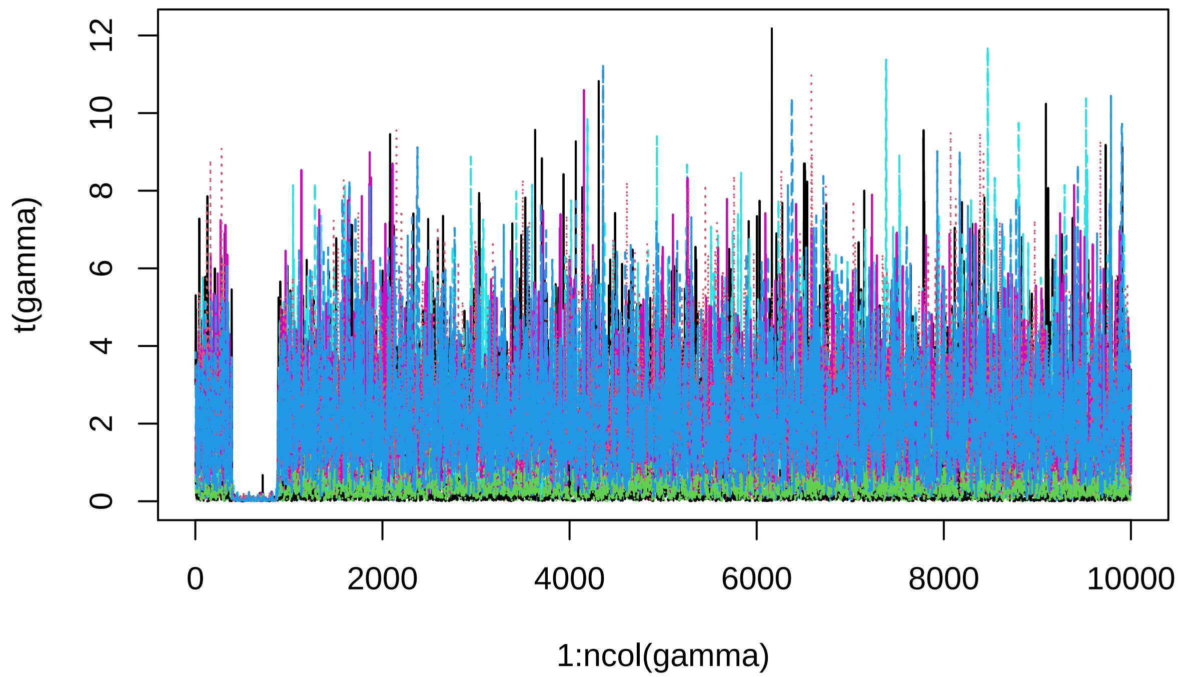
```r
plot(GDFMM$lambda*exp(-GDFMM$log_sum), type = 'l', main = "Poisson parameter")
```



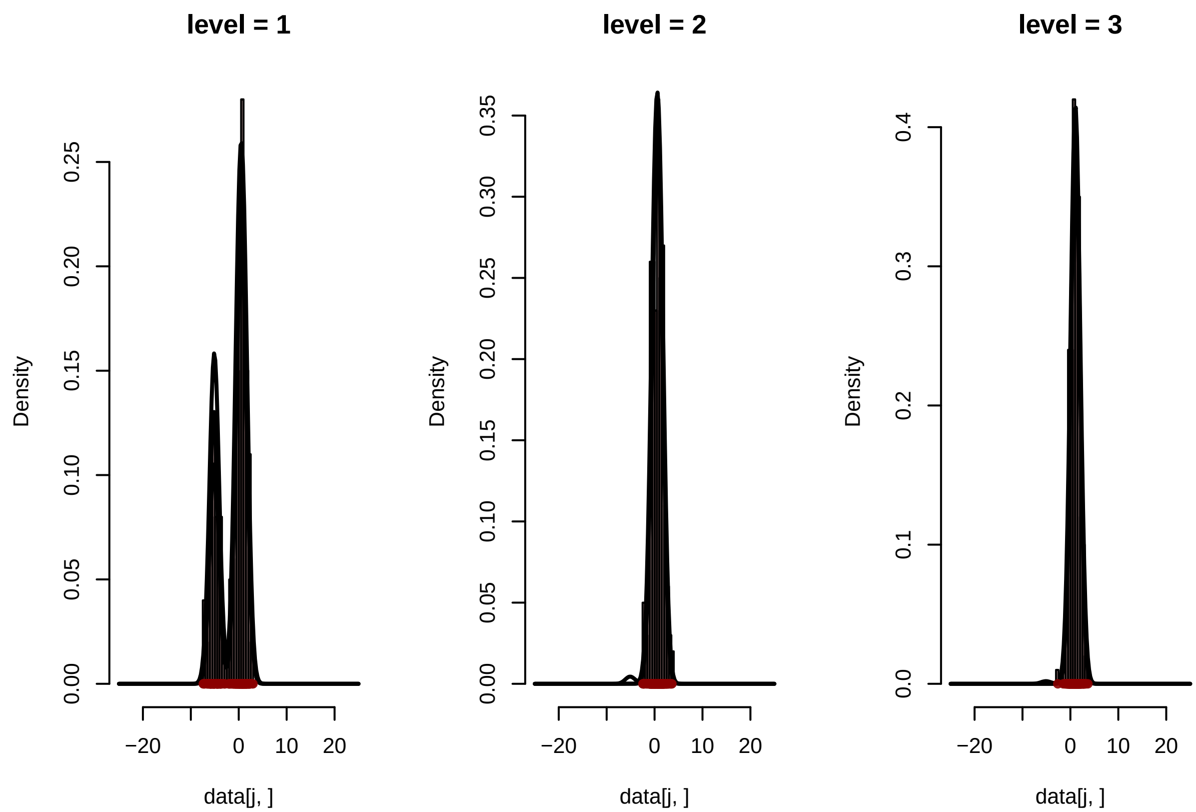

```r
#gammas
gamma = GDFMM$gamma
post_mean_gamma = rowMeans(gamma)
post_mean_gamma
```

```
##  [1] 2.04331792 0.34848033 0.08673198 0.33094559 2.00022865 1.98886880
##  [7] 0.09120647 1.96353807 0.32226123 2.05461754
```

```r
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.

$\nu = \mathbf{0.75}$, $\gamma_0 = 0.1$

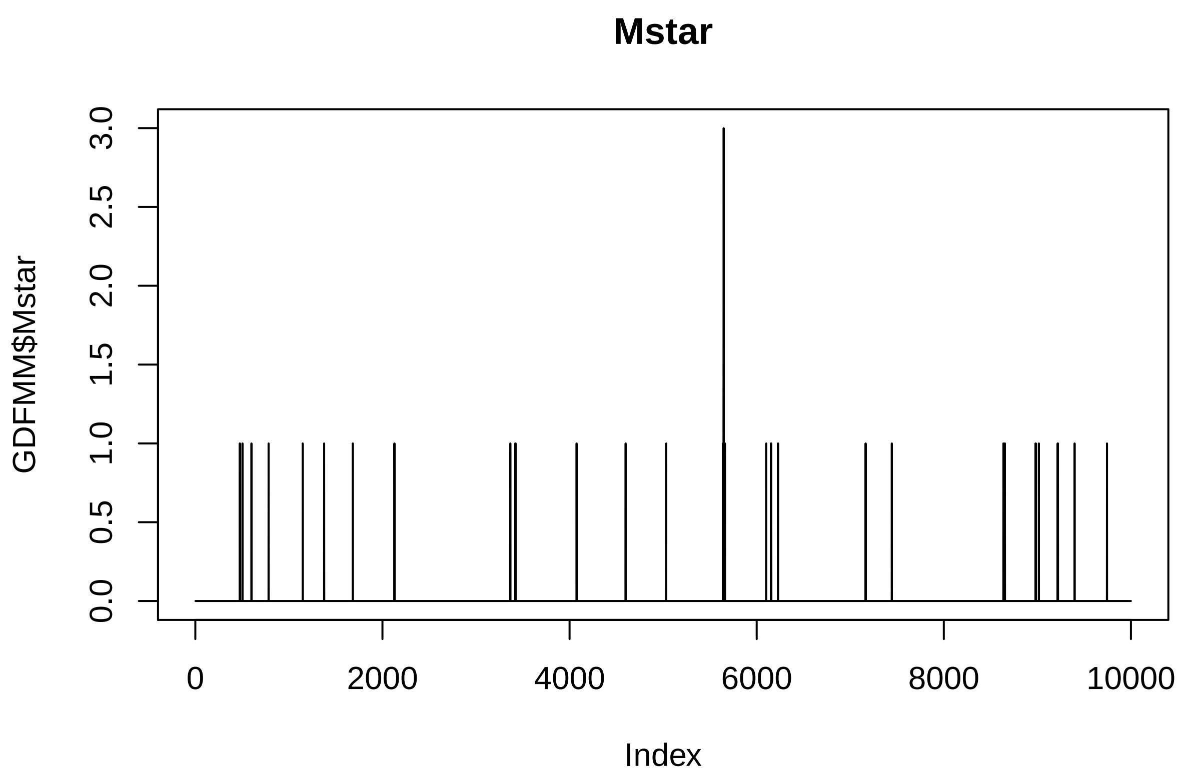
```r
niter  <- 10000
burnin <- 1
thin   <- 1

option<-list("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 contiguou
## initialize_Partition with non empty partition_vec
## Watch out modification: Mstar is not set to zero but to Mstar0
##  (K, Mstar, M) = (3,3,3)
## Chiamato initialize_S con gs_engine, mette casuale!
```

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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.0043  0.0000  3.0000
```

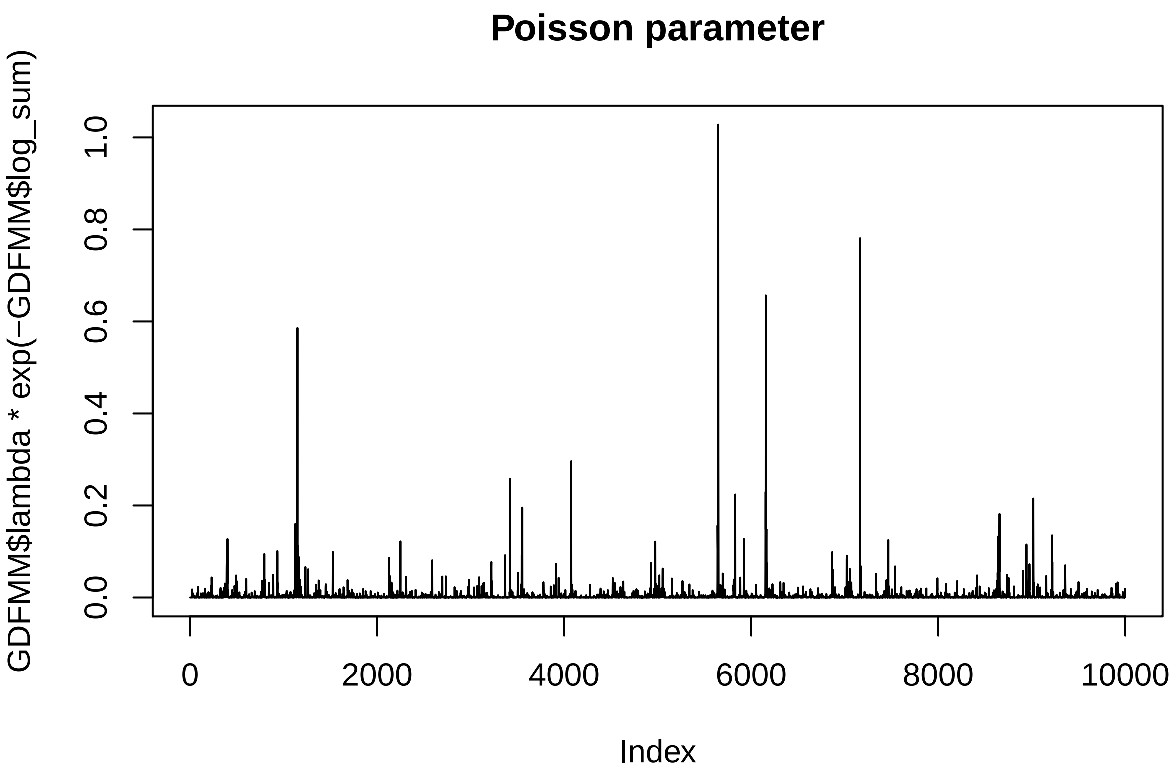
```r
plot(GDFMM$Mstar, type = 'l', main = "Mstar")
```




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

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000000 0.0000284 0.0002082 0.0034738 0.0013124 1.0278877
```

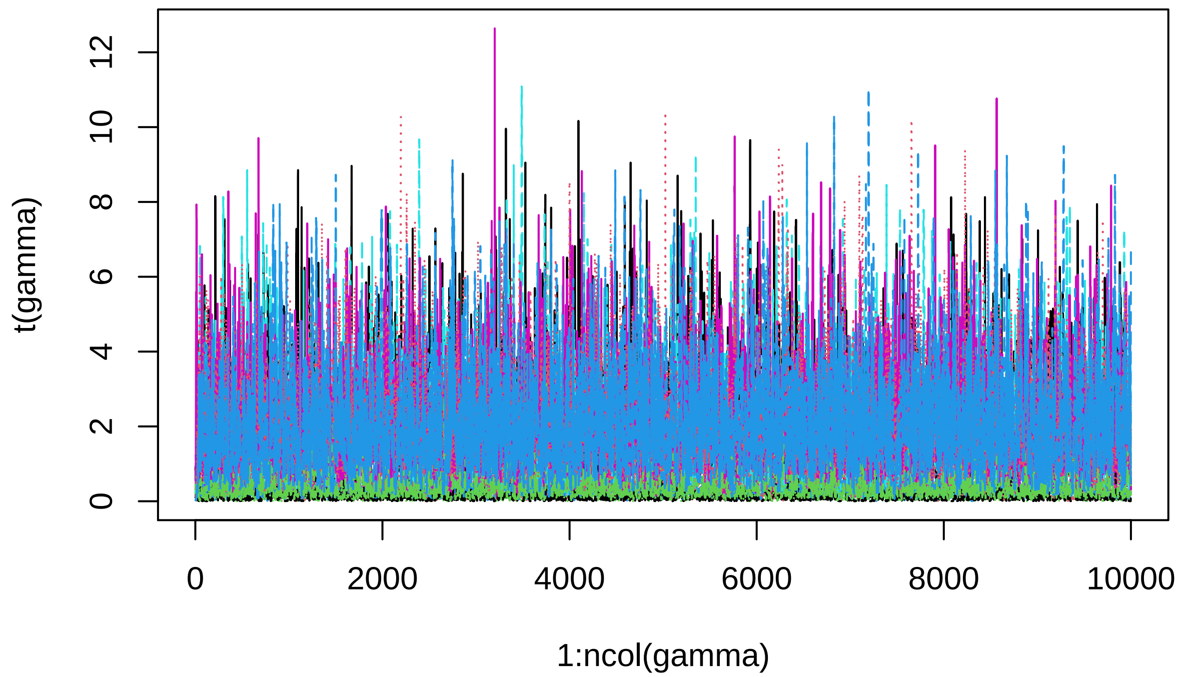
```r
plot(GDFMM$lambda*exp(-GDFMM$log_sum), type = 'l', main = "Poisson parameter")
```



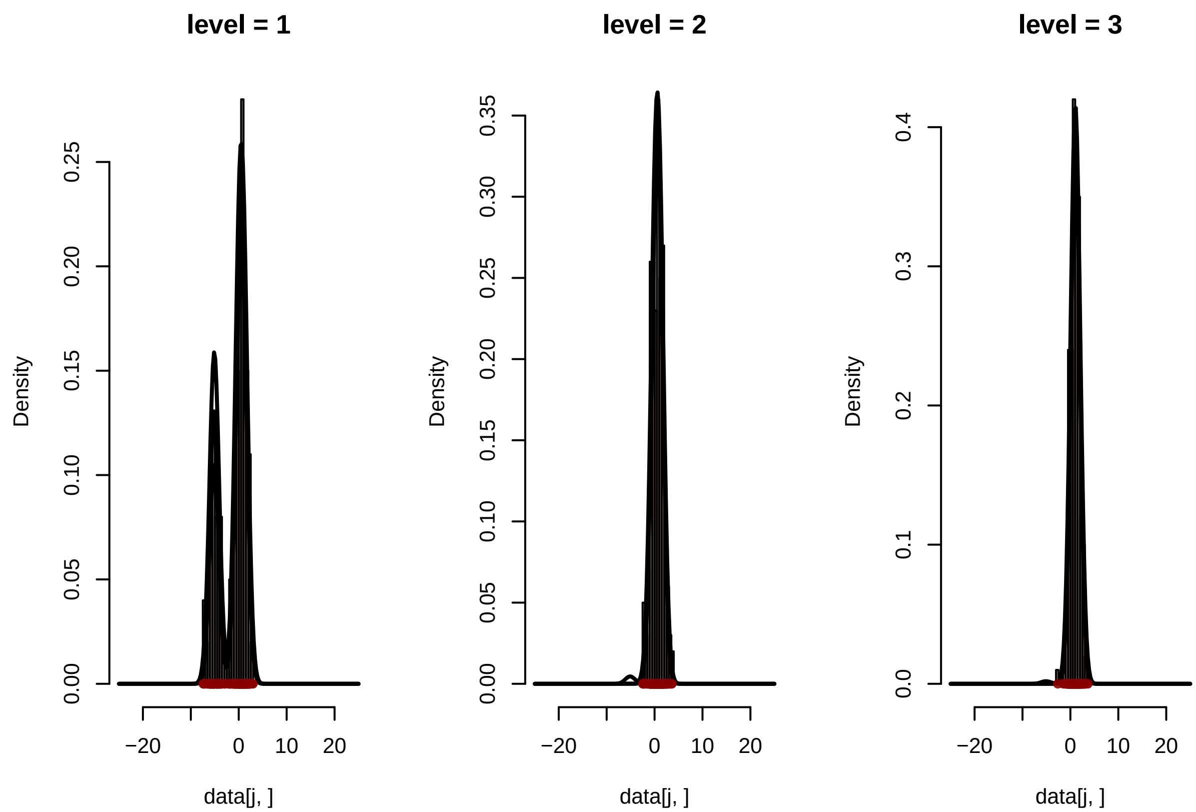

```
#gammas
gamma = GDFMM$gamma
post_mean_gamma = rowMeans(gamma)
post_mean_gamma
```

```
##  [1] 2.10187368 0.34859593 0.08901578 0.34701087 2.07630628 2.08659957
##  [7] 0.08648302 2.02942832 0.34480589 2.09876421
```

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




```r
# 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

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

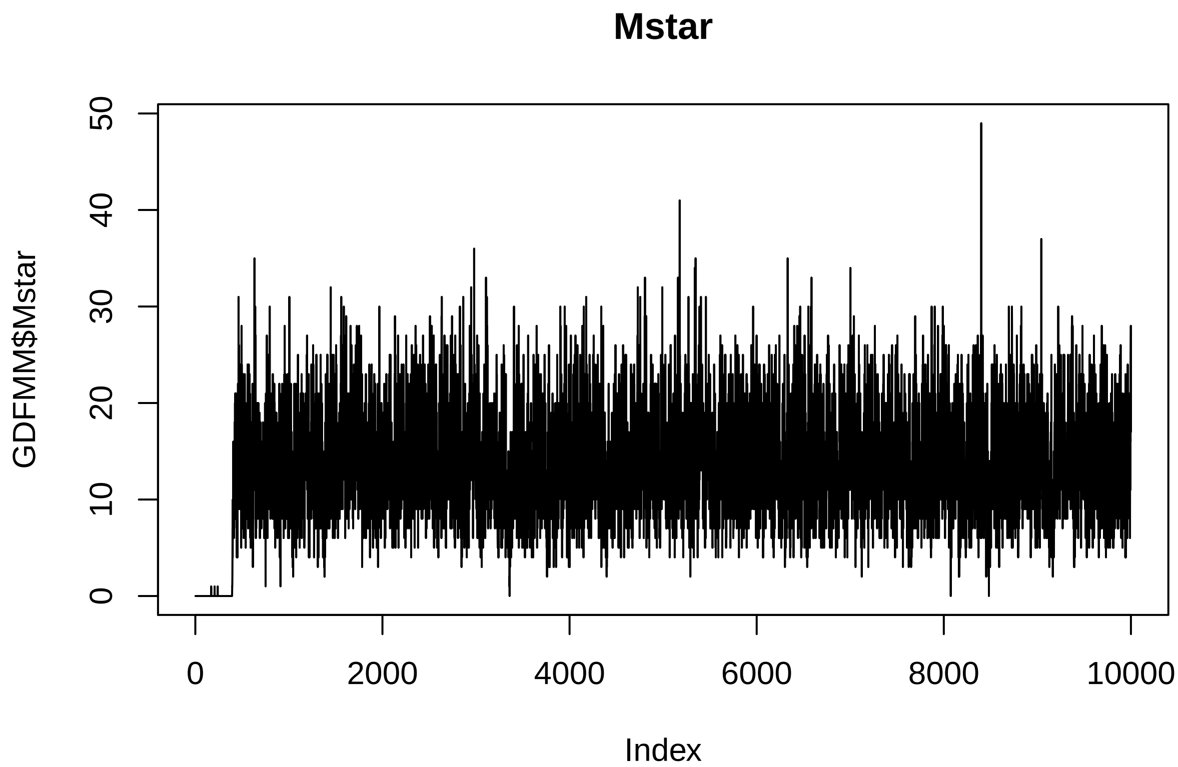
```r
niter  <- 10000
burnin <- 1
thin   <- 1

option<-list("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
## initialize_Partition with non empty partition_vec
## Watch out modification: Mstar is not set to zero but to Mstar0
##  (K, Mstar, M) = (3,3,3)
## Chiamato initialize_S con gs_engine, mette casuale!
```

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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   10.00   13.00   13.32   17.00   49.00
```

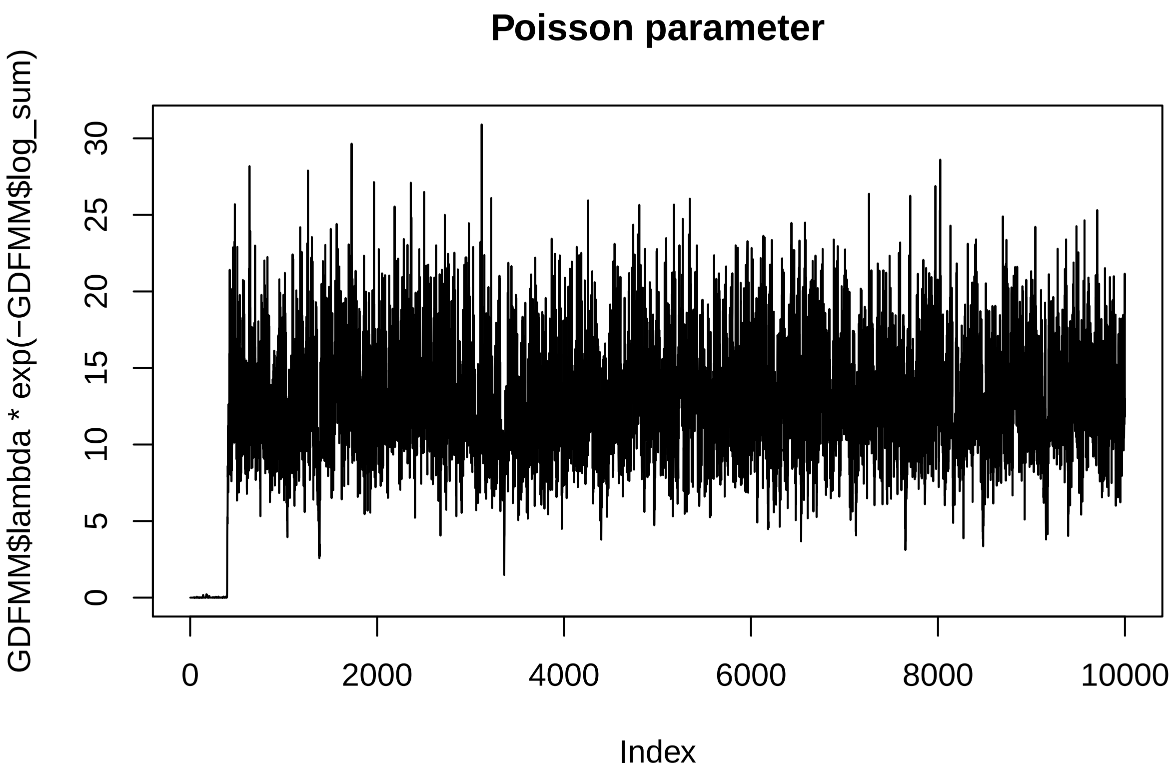
```r
plot(GDFMM$Mstar, type = 'l', main = "Mstar")
```




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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   10.12   12.57   12.50   15.17   30.91
```

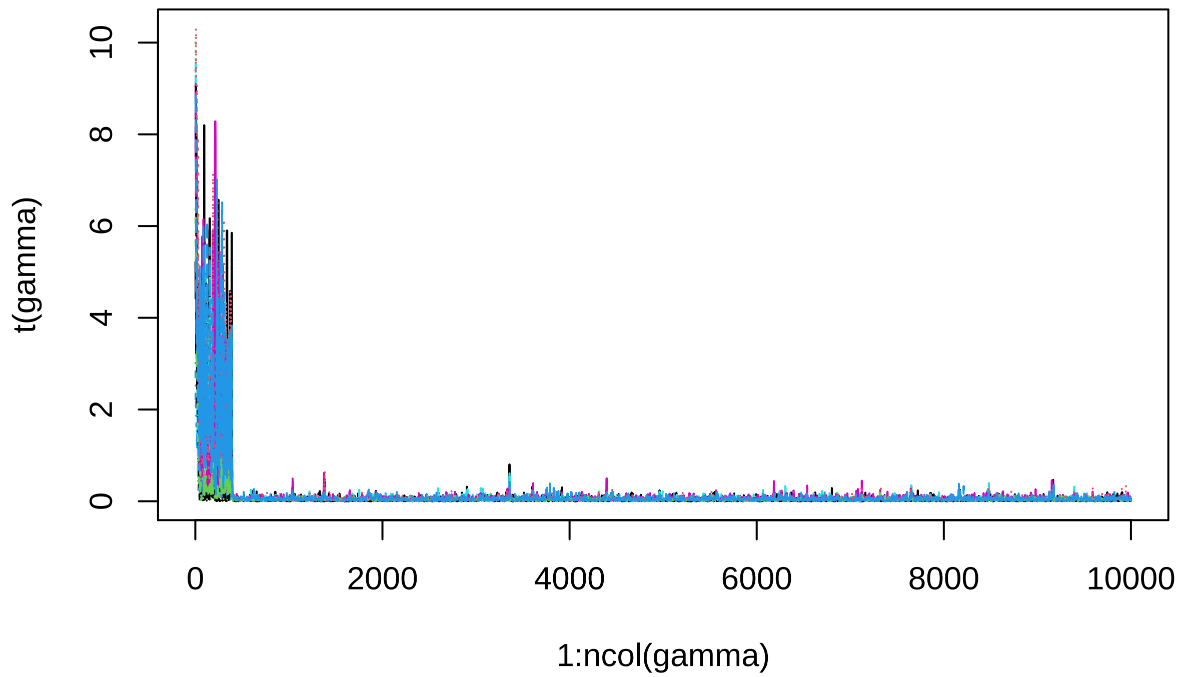
```r
plot(GDFMM$lambda*exp(-GDFMM$log_sum), type = 'l', main = "Poisson parameter")
```



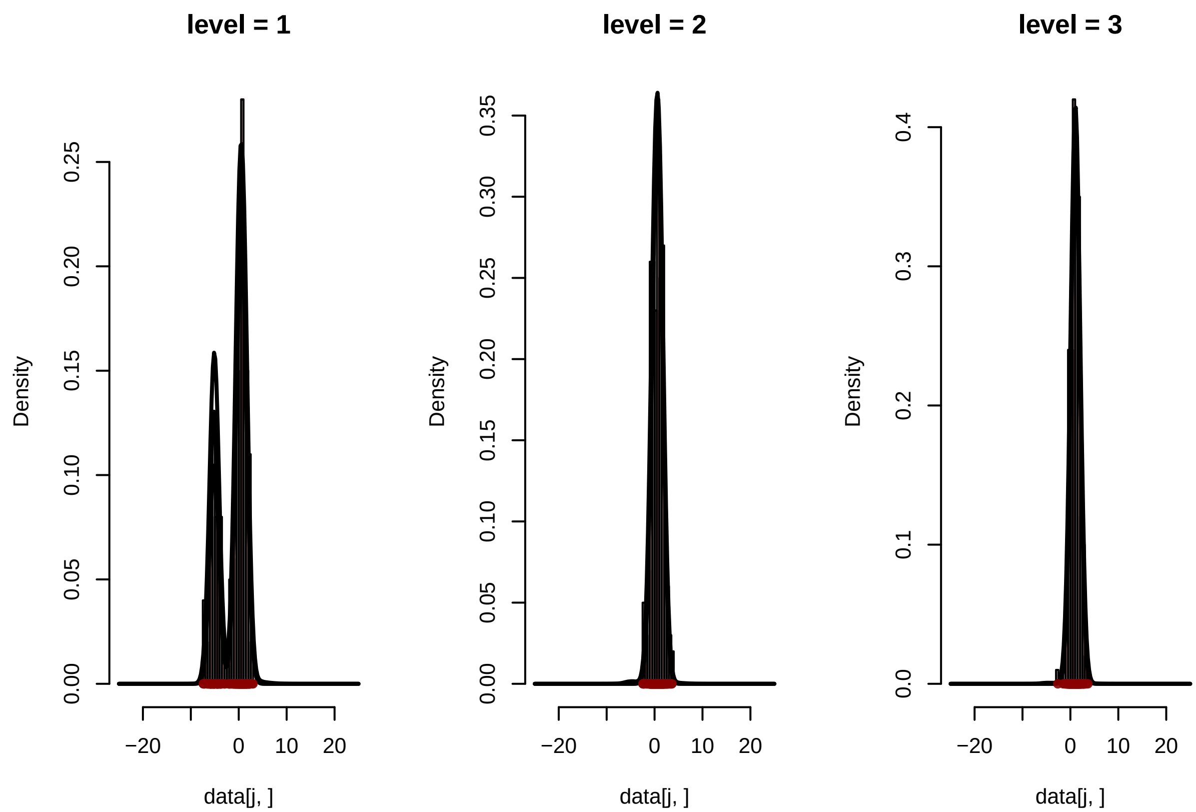

```r
#gammas
gamma = GDFMM$gamma
post_mean_gamma = rowMeans(gamma)
post_mean_gamma
```

```
##  [1] 0.13805645 0.06346115 0.03472355 0.04375452 0.11679295 0.13346602
##  [7] 0.03042872 0.14201634 0.05106840 0.14268830
```

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




```r
# 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 iterzioni, si assesta attorno a 15 o 20.

ν = **0.70**, $\gamma_0 = 1$

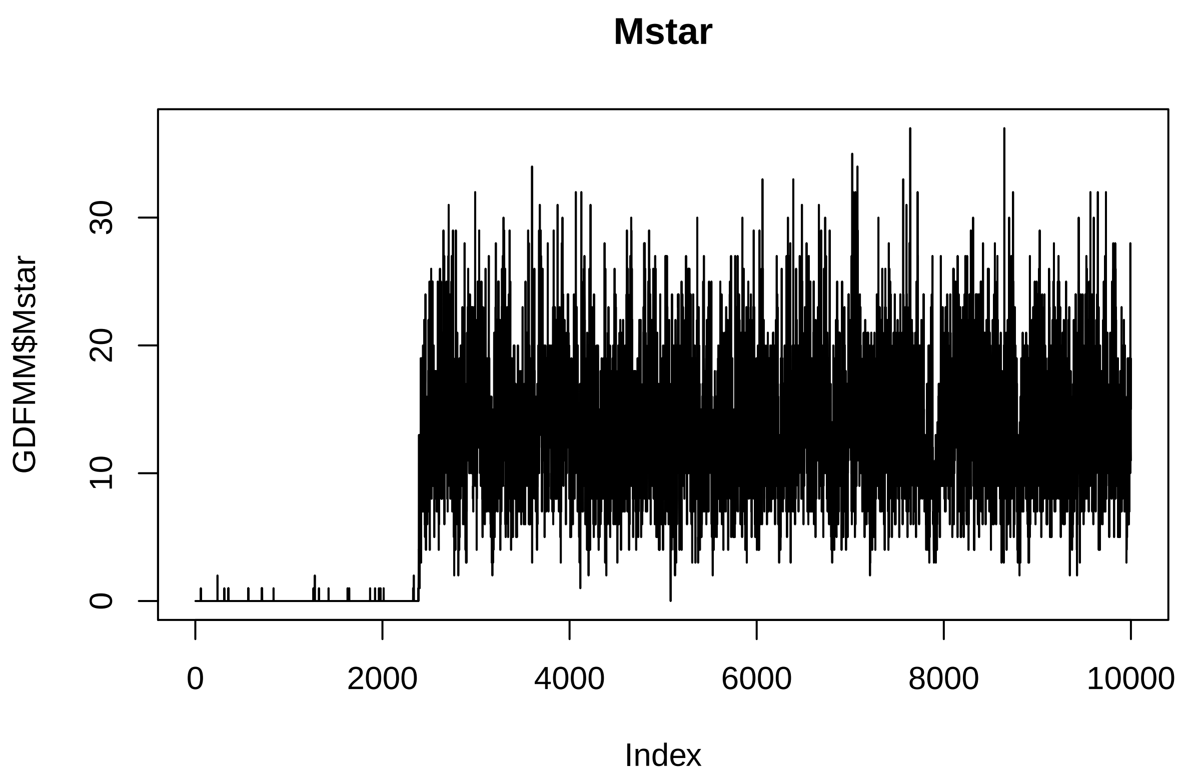
```r
niter  <- 10000
burnin <- 1
thin   <- 1

option<-list("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
## initialize_Partition with non empty partition_vec
## Watch out modification: Mstar is not set to zero but to Mstar0
##  (K, Mstar, M) = (3,3,3)
## Chiamato initialize_S con gs_engine, mette casuale!
```

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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    4.00   11.00   10.45   16.00   37.00
```

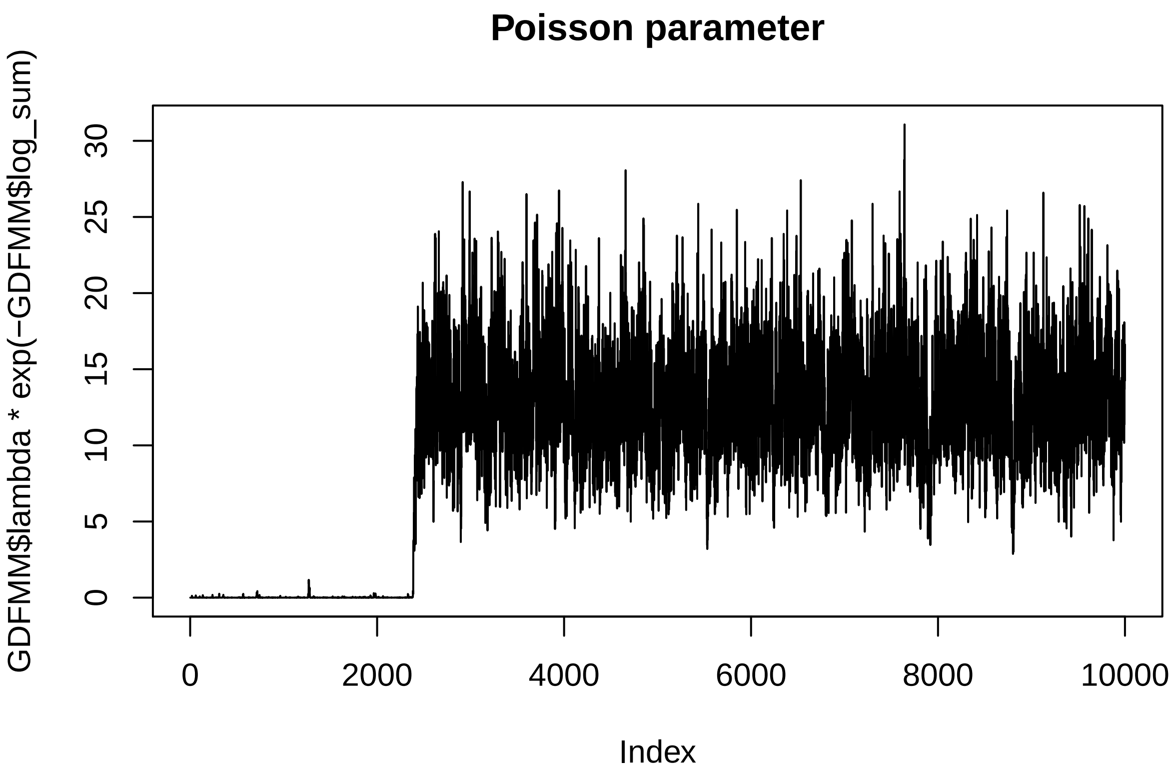
```r
plot(GDFMM$Mstar, type = 'l', main = "Mstar")
```




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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   5.786  11.281   9.803  14.235  31.078
```

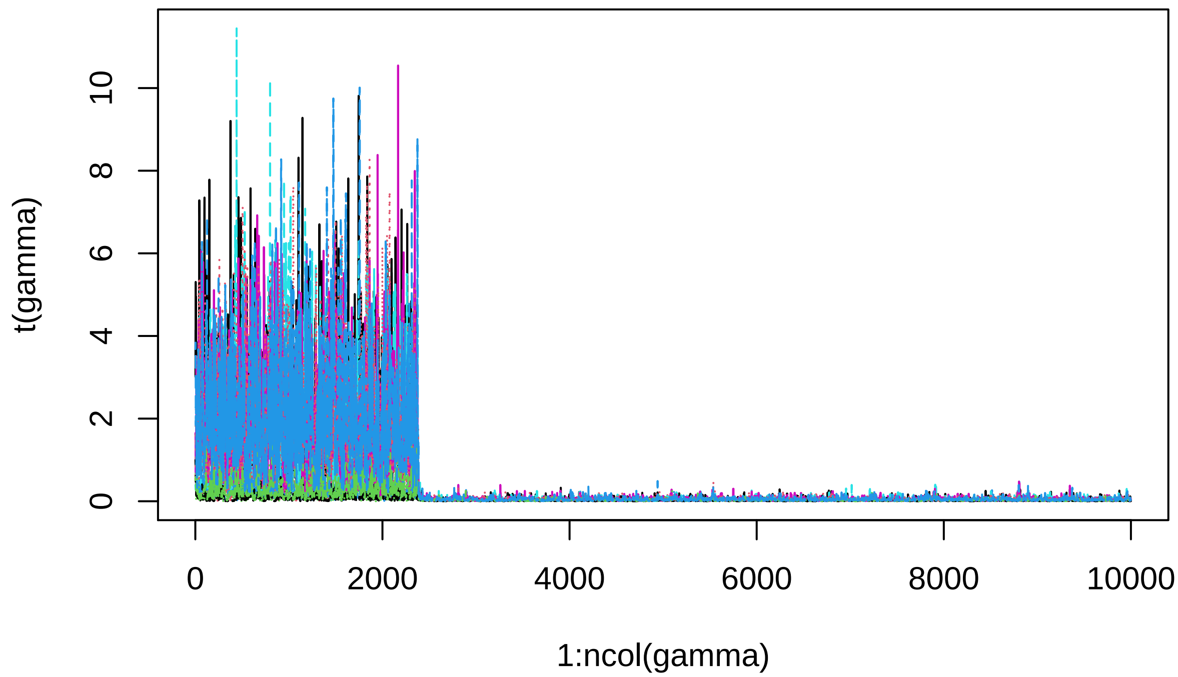
```r
plot(GDFMM$lambda*exp(-GDFMM$log_sum), type = 'l', main = "Poisson parameter")
```



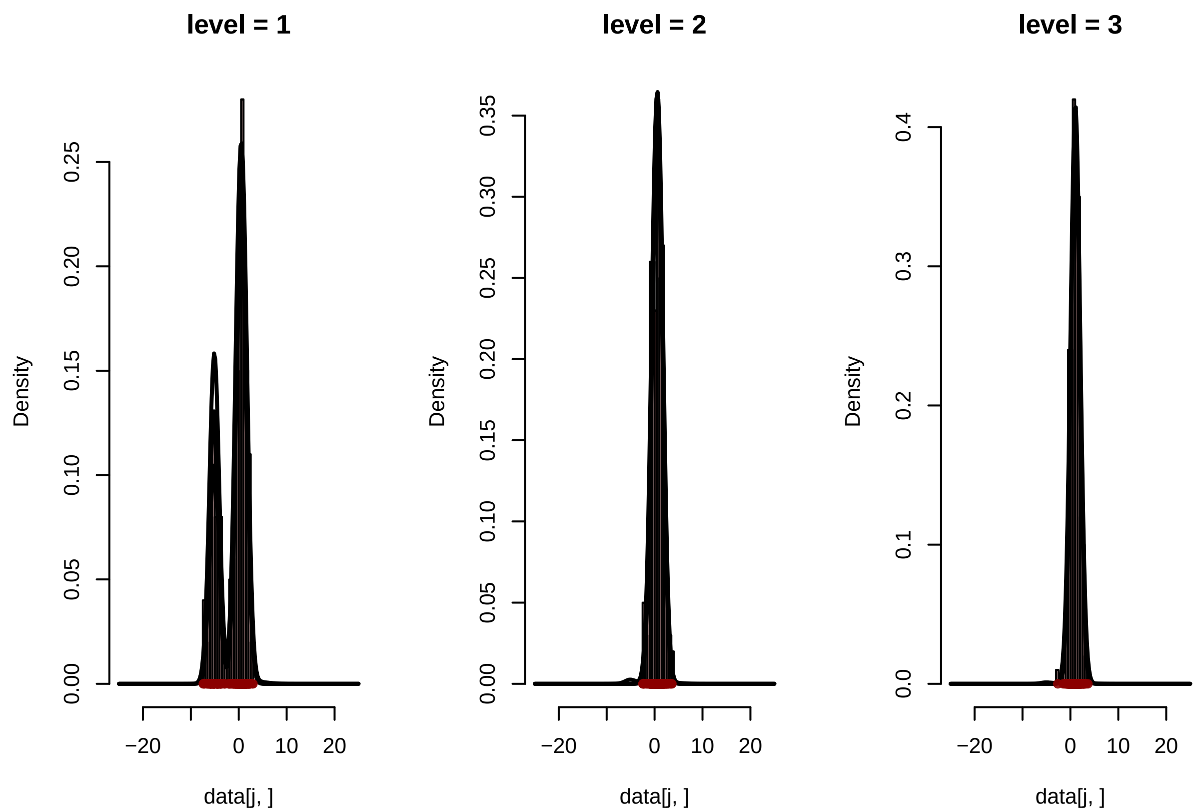

```
#gammas
gamma = GDFMM$gamma
post_mean_gamma = rowMeans(gamma)
post_mean_gamma
```

```
##  [1] 0.55659685 0.10617708 0.03678898 0.10305602 0.55339485 0.51277949
##  [7] 0.03157496 0.52572553 0.09978354 0.54568236
```

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




```r
# 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.

$\nu = 1/10$

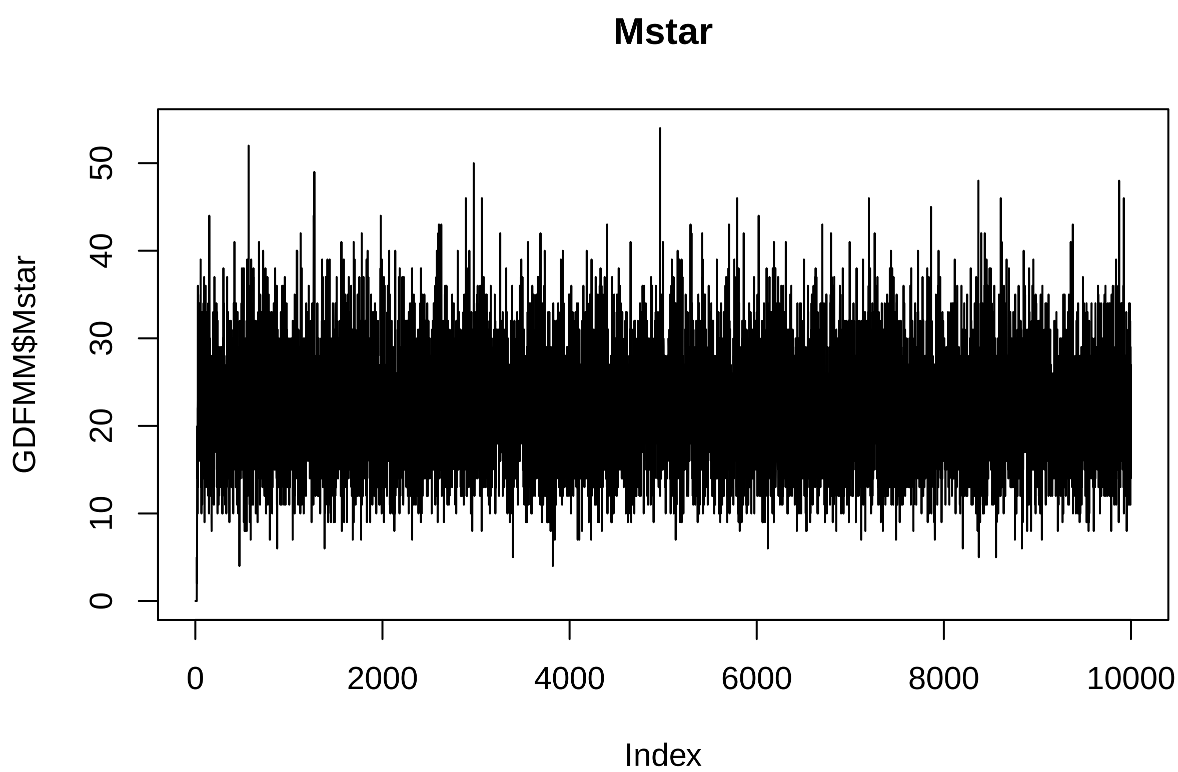
```
niter  <- 10000
burnin <- 1
thin   <- 1

option<-list("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
## initialize_Partition with non empty partition_vec
## Watch out modification: Mstar is not set to zero but to Mstar0
##  (K, Mstar, M) = (3,3,3)
## Chiamato initialize_S con gs_engine, mette casuale!
```

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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   17.00   21.00   21.74   26.00   54.00
```

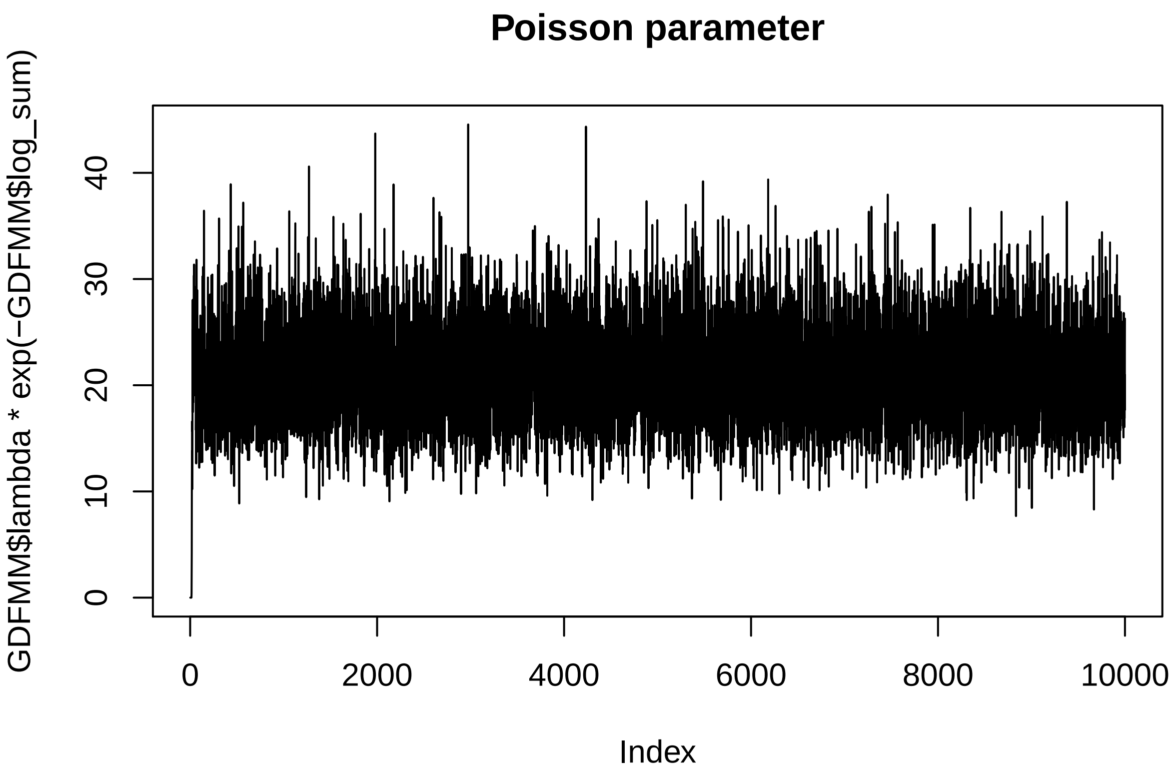
```r
plot(GDFMM$Mstar, type = 'l', main = "Mstar")
```




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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   17.78   20.60   20.92   23.78   44.56
```

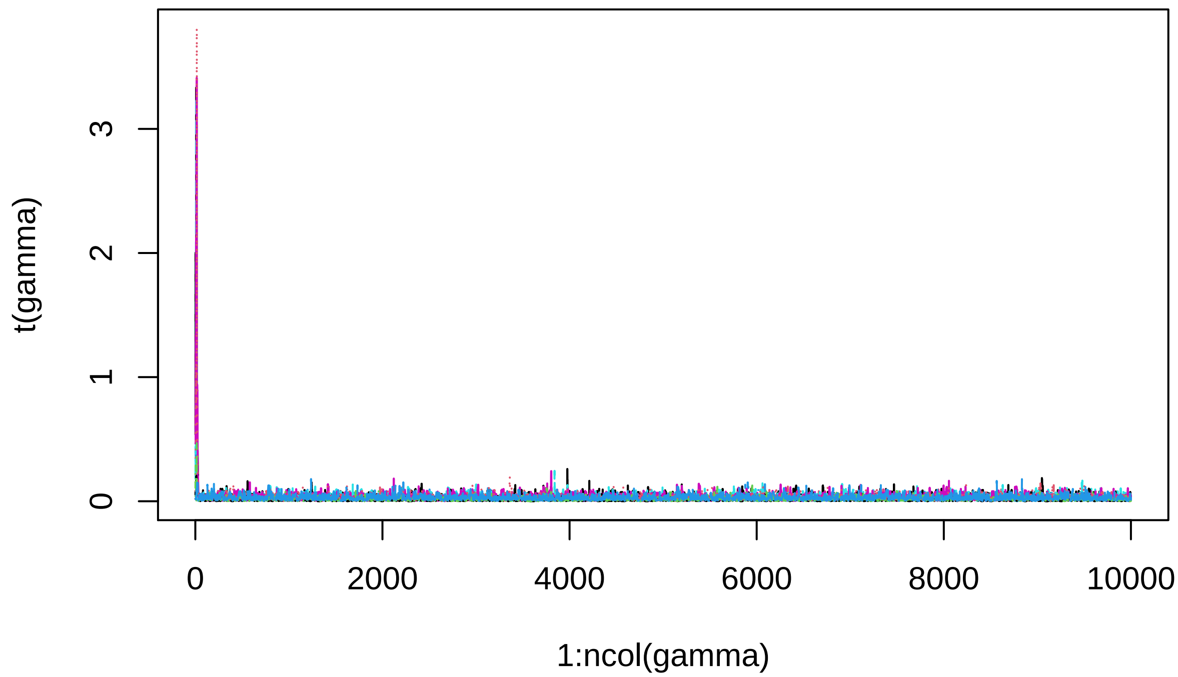
```r
plot(GDFMM$lambda*exp(-GDFMM$log_sum), type = 'l', main = "Poisson parameter")
```



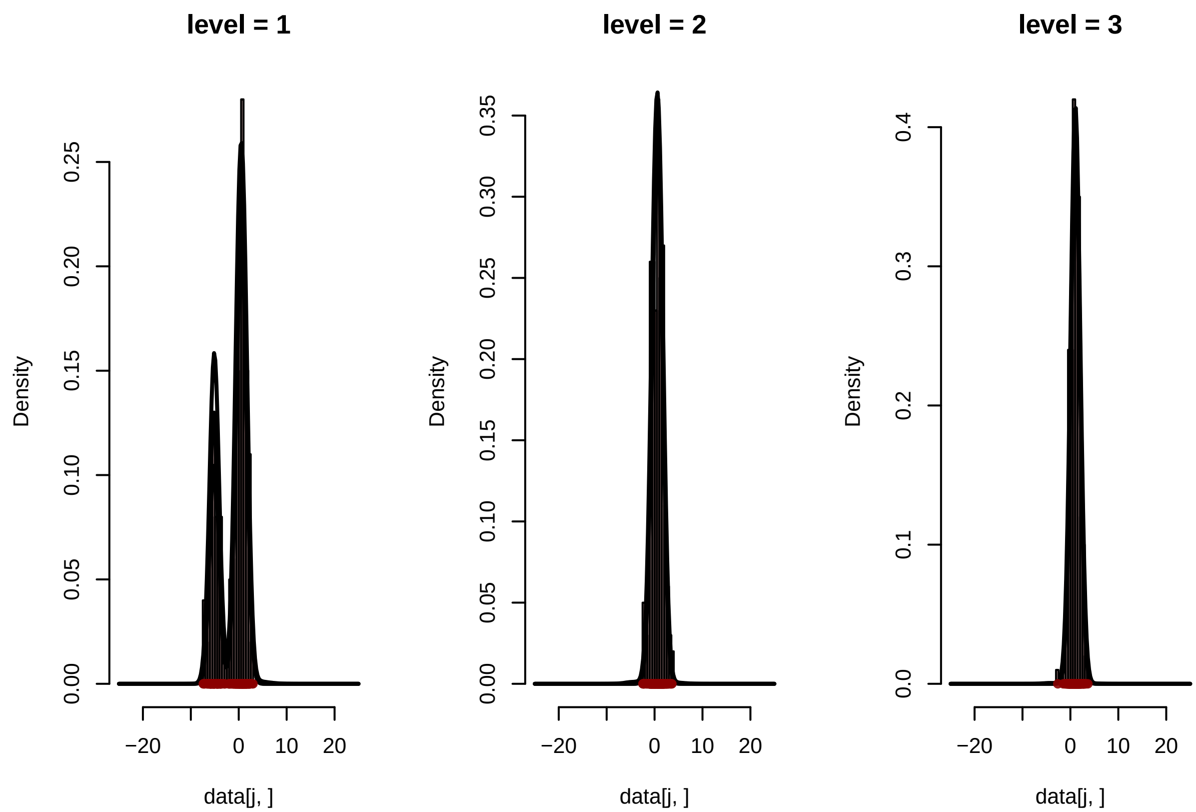

```r
#gammas
gamma = GDFMM$gamma
post_mean_gamma = rowMeans(gamma)
post_mean_gamma
```

```
##  [1] 0.032834845 0.019432552 0.008484405 0.018814696 0.032143058 0.032900091
##  [7] 0.007954722 0.031719156 0.018070062 0.028817427
```

```r
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).

$\nu = 1/100$

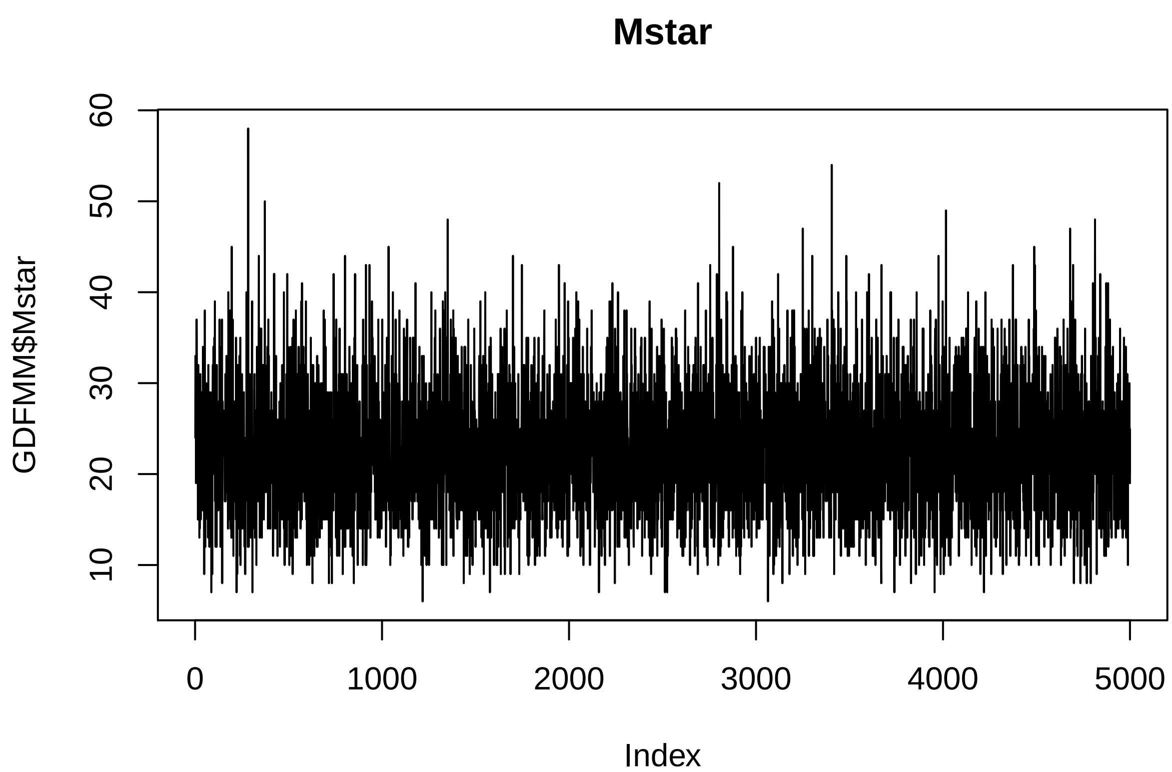
```
niter  <- 5000
burnin <- 5000
thin   <- 1

option<-list("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
## initialize_Partition with non empty partition_vec
## Watch out modification: Mstar is not set to zero but to Mstar0
##  (K, Mstar, M) = (3,3,3)
## Chiamato initialize_S con gs_engine, mette casuale!
```

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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     6.0    18.0    22.0    22.6    27.0    58.0
```

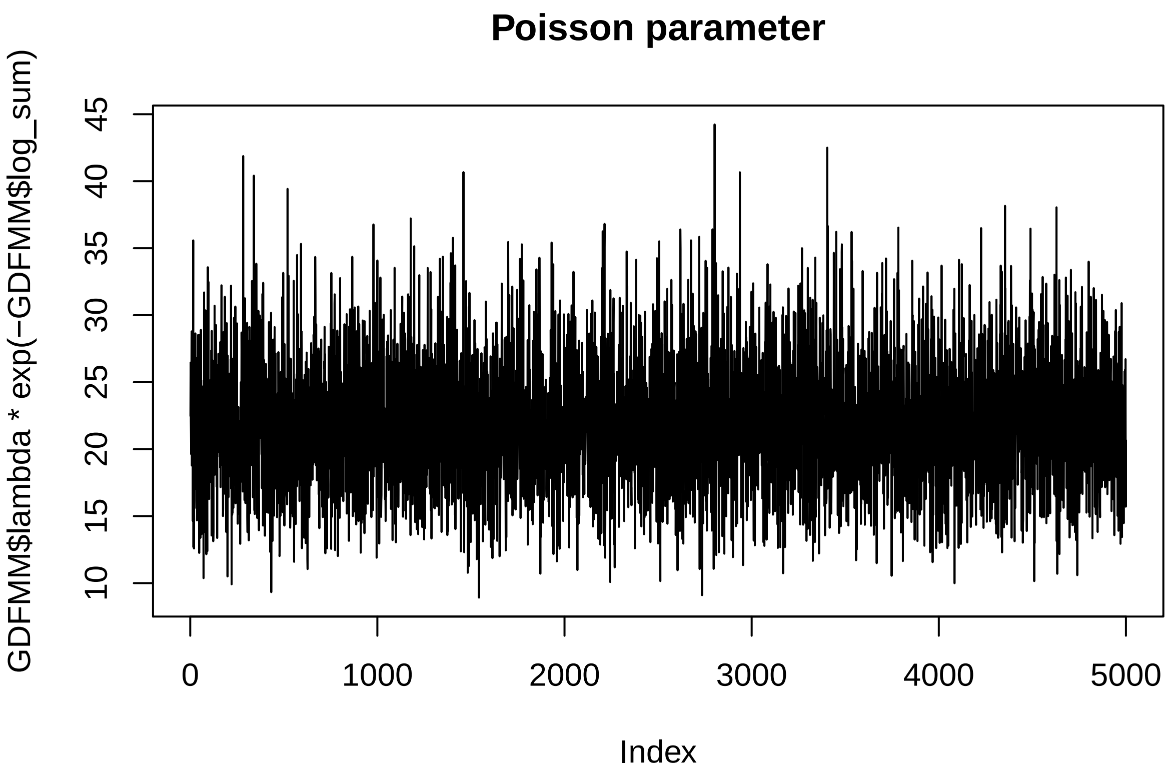
```r
plot(GDFMM$Mstar, type = 'l', main = "Mstar")
```




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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   8.918  18.518  21.430  21.785  24.718  44.238
```

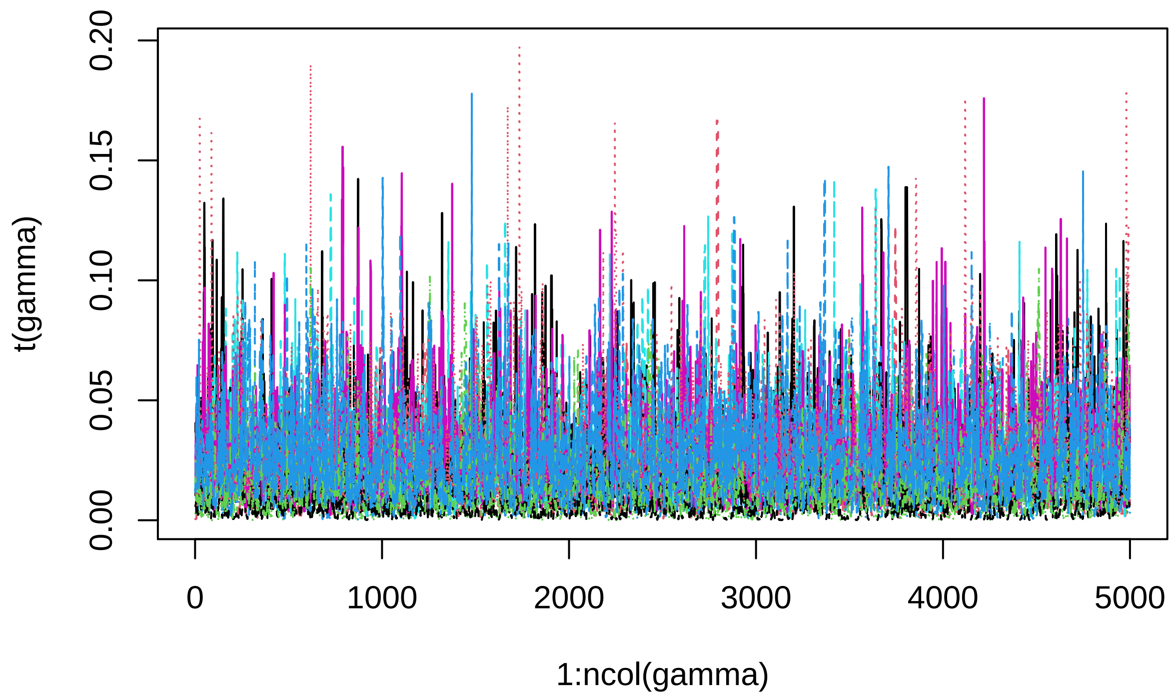
```r
plot(GDFMM$lambda*exp(-GDFMM$log_sum), type = 'l', main = "Poisson parameter")
```



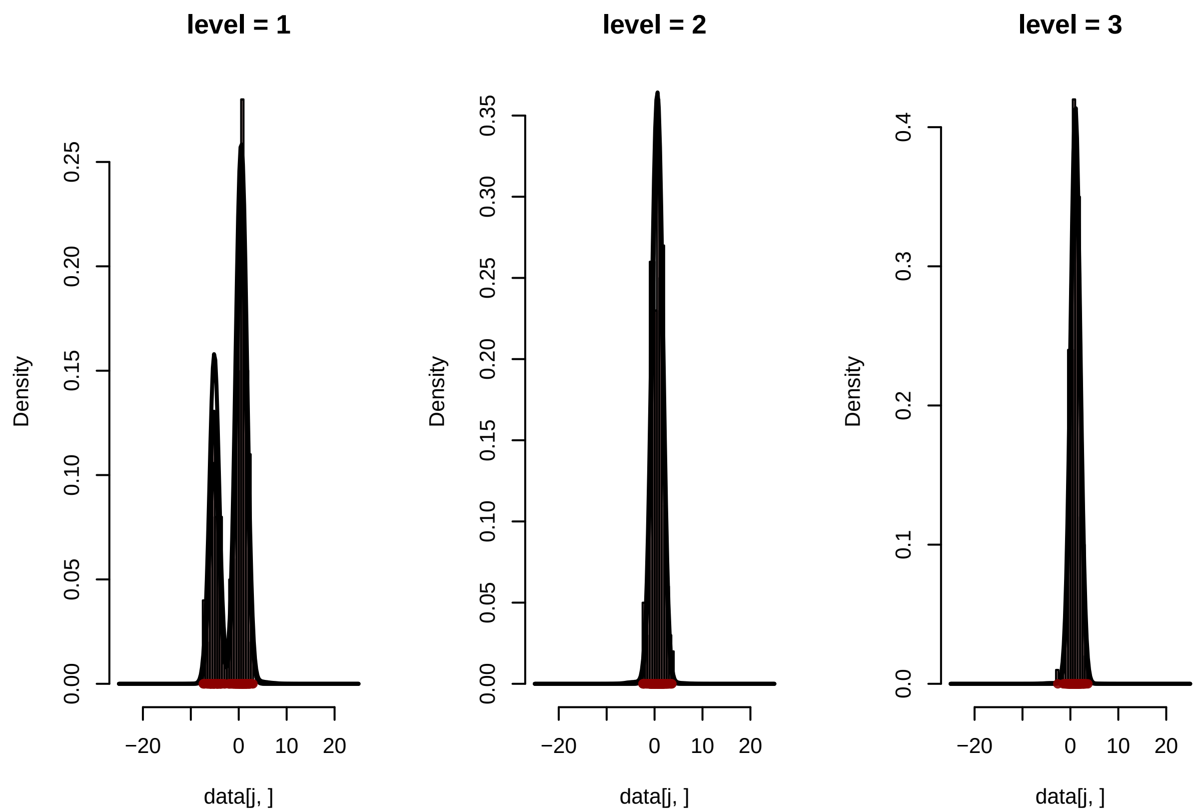

```
#gammas
gamma = GDFMM$gamma
post_mean_gamma = rowMeans(gamma)
post_mean_gamma
```

```
##  [1] 0.029974416 0.017558429 0.007260093 0.016880978 0.027063380 0.028790121
##  [7] 0.007034412 0.028765647 0.016808761 0.027832859
```

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




```r
# 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))

}
```



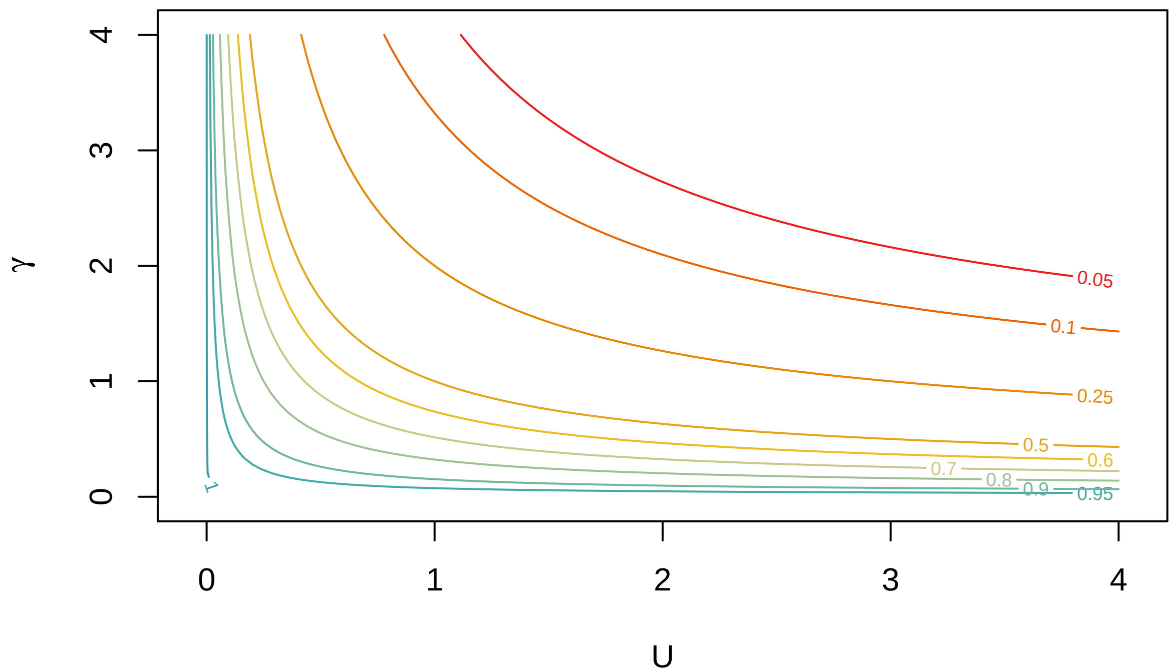
Qua l'effetto del valore iniziale è decisamente mitigato, inoltre la fase per arrivare alla convergenza della catenza è un po' più smooth, non è più a scalino.

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)
```

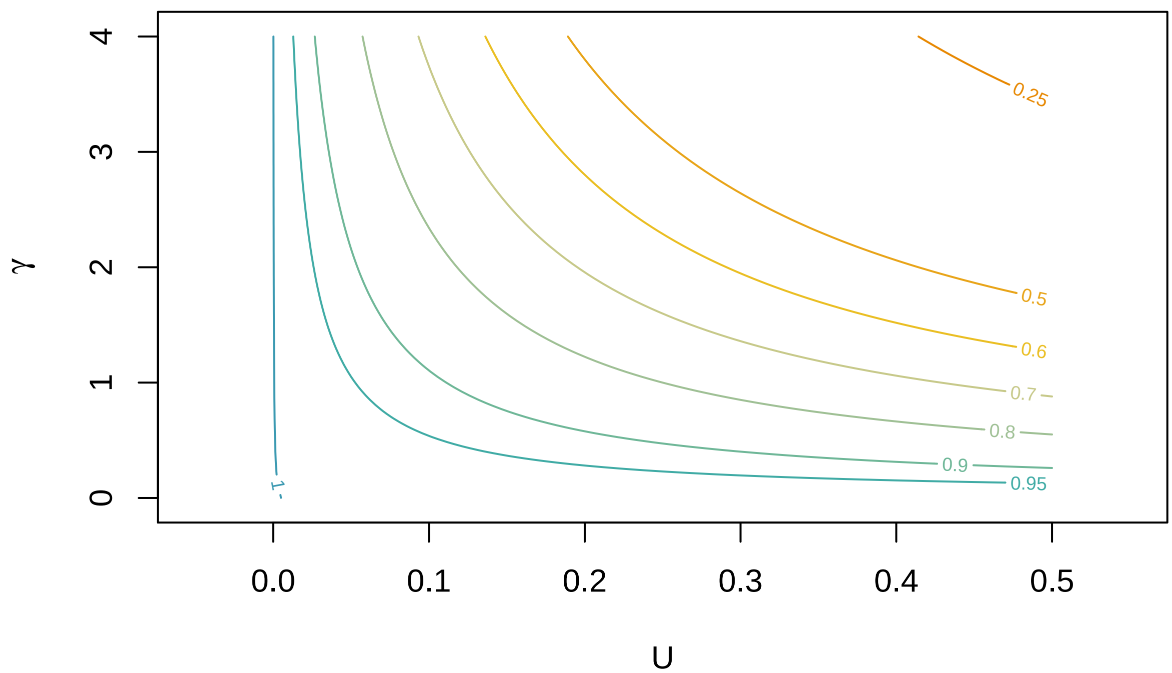
```r
grid <- expand.grid(U, gamma)
f_values <- apply(grid, 1, f)
f_values <- matrix(f_values, nrow = length(U),
                   ncol = length(gamma), byrow=F)

# plot with confindence 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 $U \in (0, 0.5)$.

```r
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)
```

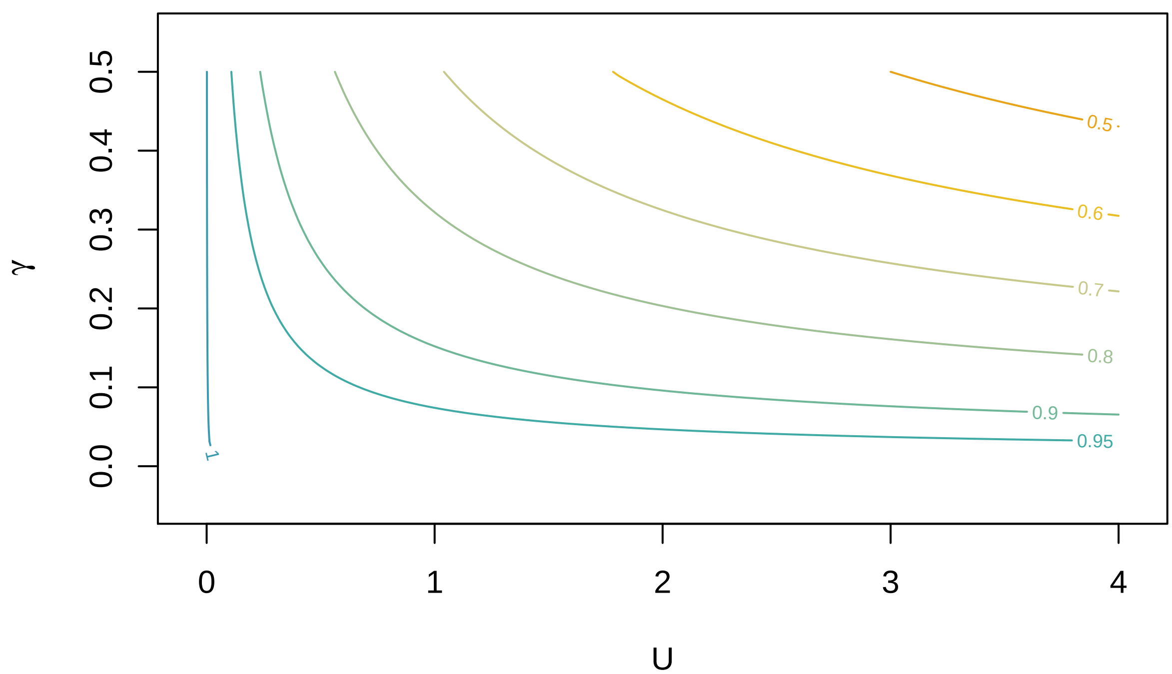
```r
f_values <- apply(grid, 1, f)
f_values <- matrix(f_values, nrow = length(U),
                   ncol = length(gamma), byrow=F)

# plot with confindence 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 $\gamma \in (0.001, 0.5)$.

```r
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)
```

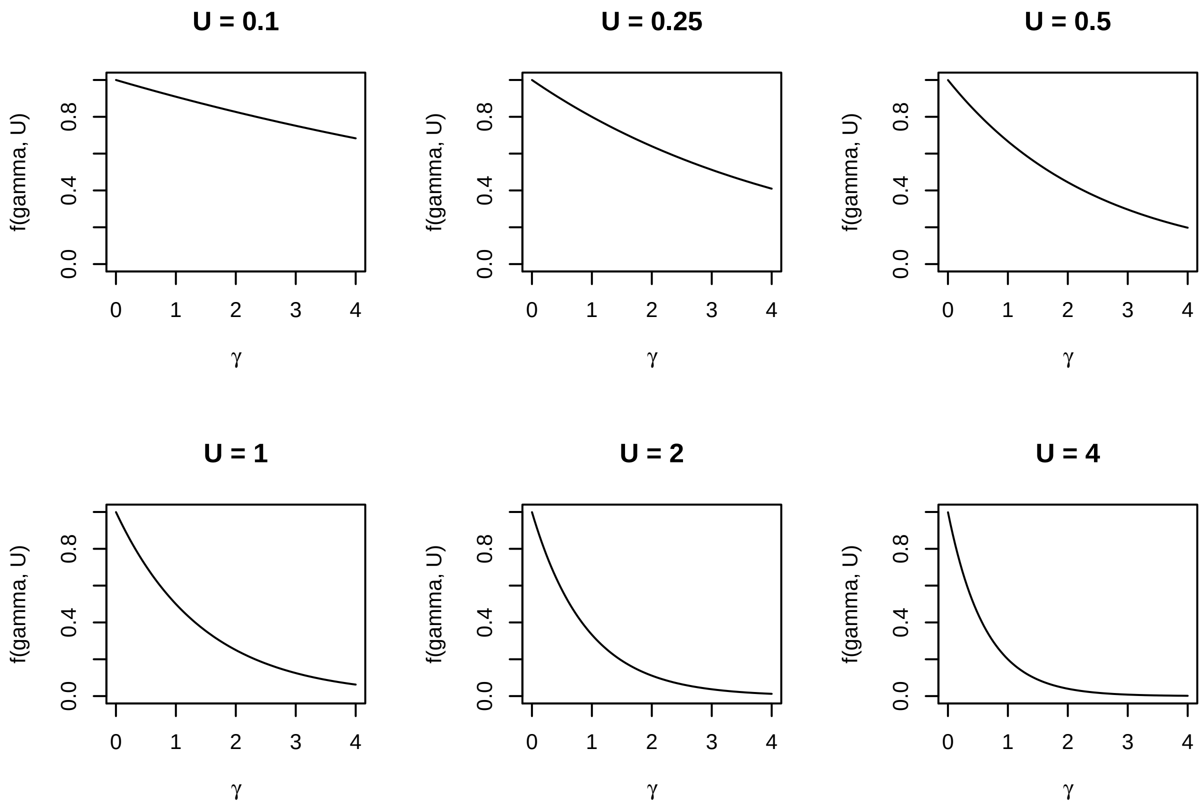
```
f_values <- matrix(f_values, nrow = length(U),
                  ncol = length(gamma), byrow=F)

# plot with confindence 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-dimensionali - gamma

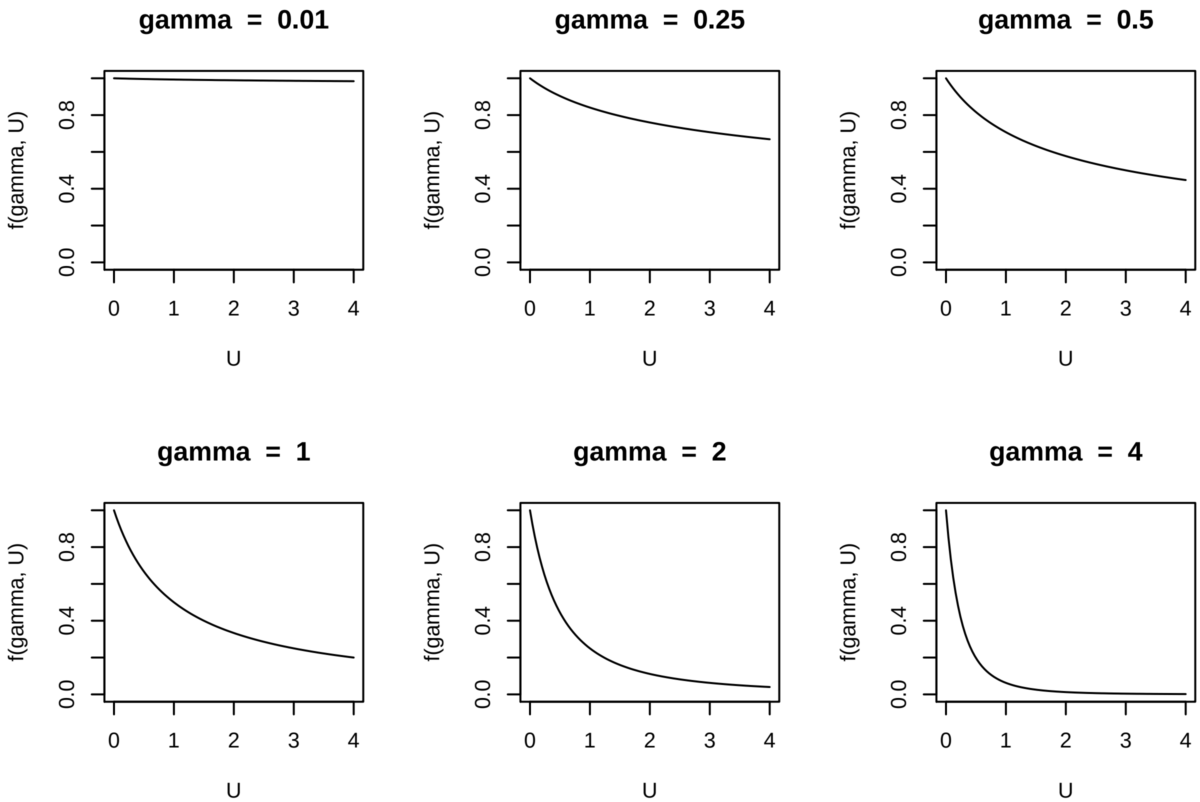
```
f_gamma = function(x, u){
  return( 1/(1+u)^x )
}

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))
}
```


gamma = 0.01
gamma = 0.25
gamma = 0.5
gamma = 1
gamma = 2
gamma = 4
f(gamma, U)
0.0
0.4
0.8
0
1
2
3
4
U