HW7

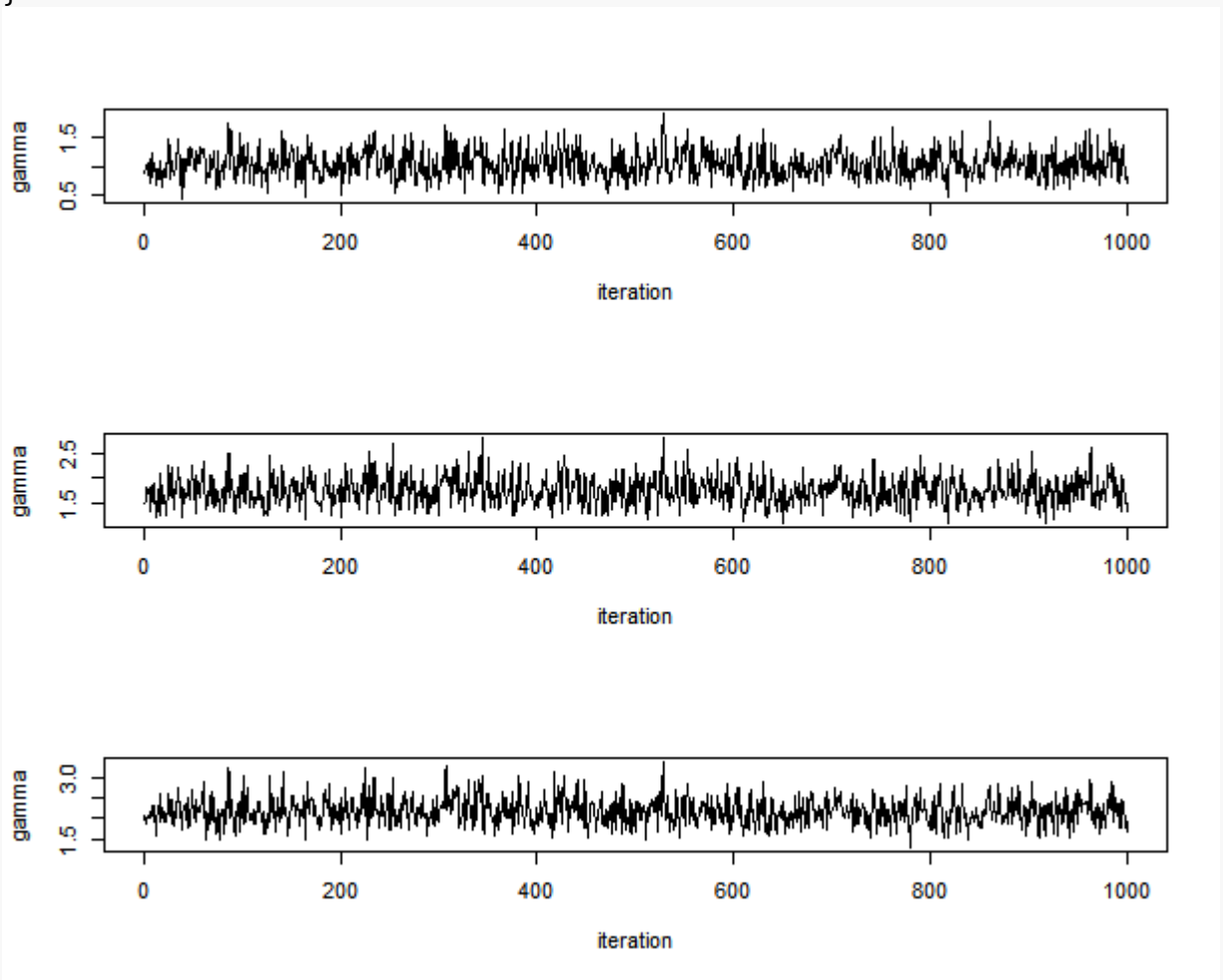
```r
# set
pacman::p_load(mcmc, rjags, runjags, dplyr, ggplot2, mvtnorm, truncnorm)
setwd('C:/Users/dnskd/Desktop/20Spring/Bayesian/week7')
rm(list=ls())

# 예제 7.1
data<-c(1,1,1,1,1,5,1,1,3,1,1,2,1,0,2,1,1,1,1,0,1,1,1,2,1,1,2,1,1,4,
        1,0,1,1,0,2,2,1,3,2,0,4,2,1,2,2,0,1,2,1,5,2,1,2,2,0,3,2,1,3,
        2,1,5,2,0,2,2,1,2,2,1,1,3,0,1,3,1,3,3,0,2,3,0,5,3,1,4,3,0,3,
        3,0,2,4,1,5,4,1,2,4,1,4,4,0,3,4,0,3,4,0,3,4,1,5,4,0,5,4,0,5)

n <- 40
data <- matrix(data, n, 3, byrow = T)
stress <- data[, 1]
SES <- data[, 2]
mental <- data[ ,3]
ftable(stress, SES, mental)

##            mental 1 2 3 4 5
## stress SES
## 1      0          2 2 0 0 0
##        1          2 3 1 1 1
## 2      0          1 1 1 1 0
##        1          1 3 2 0 2
## 3      0          1 2 1 0 1
##        1          0 0 1 1 0
## 4      0          0 0 3 0 2
##        1          0 1 0 1 2

X <- cbind(rep(1, n), stress, SES, stress*SES)
p <- ncol(X)
y <- mental
nLevels <- 5
nData <- length(y)
dimBeta <- ncol(X)

K <- 5
# Prior
gamma0 <- rep(0, K); sigma0 <- rep(10, K)

# initial
beta <- rep(0, p)
freq.y <- data.frame(table(y))$Freq
prob <- freq.y/sum(freq.y)
```

```r
gamma <- rep(0, K)
gamma[1] <- 0; gamma[K] <- Inf
for(i in 2:(K-1)){
  gamma[i] <- qnorm(sum(prob[1:i]), 0, 10)
  if(gamma[i] <= gamma[i-1]) gamma[i] <- gamma[i-1] + 0.1
}

z <- rep(0, n)
```

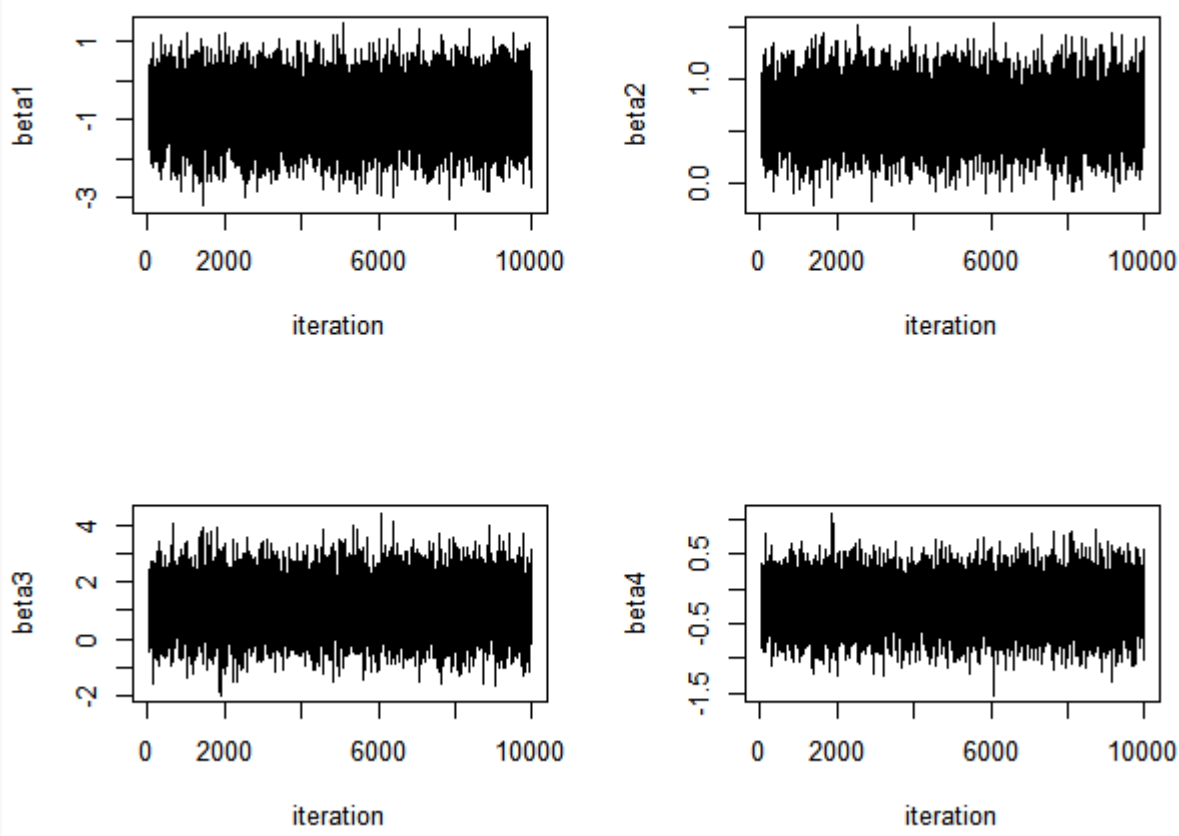
Start MCMC

```r
nwarm <- 5000; maxn <- 10000; nthin <- 100
beta.sim <- NULL
gamma.sim <- NULL

for(isim in 1:(nwarm + maxn*nthin)){
  #generate z
  for(i in 1:n){
    if(y[i]==1) z[i] <- rtruncnorm(1, -Inf, gamma[1], X[i,]%*%beta, 1)
  for(k in 2:K)
    if(y[i]==k) z[i] <- rtruncnorm(1, gamma[k-1], gamma[k], X[i,]%*%beta, 1)
  }


# generate beta
var.beta <- n/(n+1)*solve(t(X)%*%X)
mean.beta <- n/(n+1)*solve(t(X)%*%X) %*% (t(X)%*%z)
beta <- as.vector(rmvnorm(1, mean.beta, var.beta))


# generate gamma
gamma[1] <- 0; gamma[K] <- Inf
for(k in 2:(K-1)){
  ak <- max(max(z[y==k]), gamma[k-1])
  bk <- min(min(z[y==k+1]), gamma[k+1])
  gamma[k] <- rtruncnorm(1, ak, bk, gamma0[k], sigma0[k])
}


if(isim>nwarm & isim %%nthin==0){
  beta.sim <- rbind(beta.sim, beta)
  gamma.sim <- rbind(gamma.sim, gamma)
  }

}
#### End MCMC ####

#### Fig 7.2 연속된 1000 개의 표본에서 회귀계수 gamma 의 경로그림 ####
```

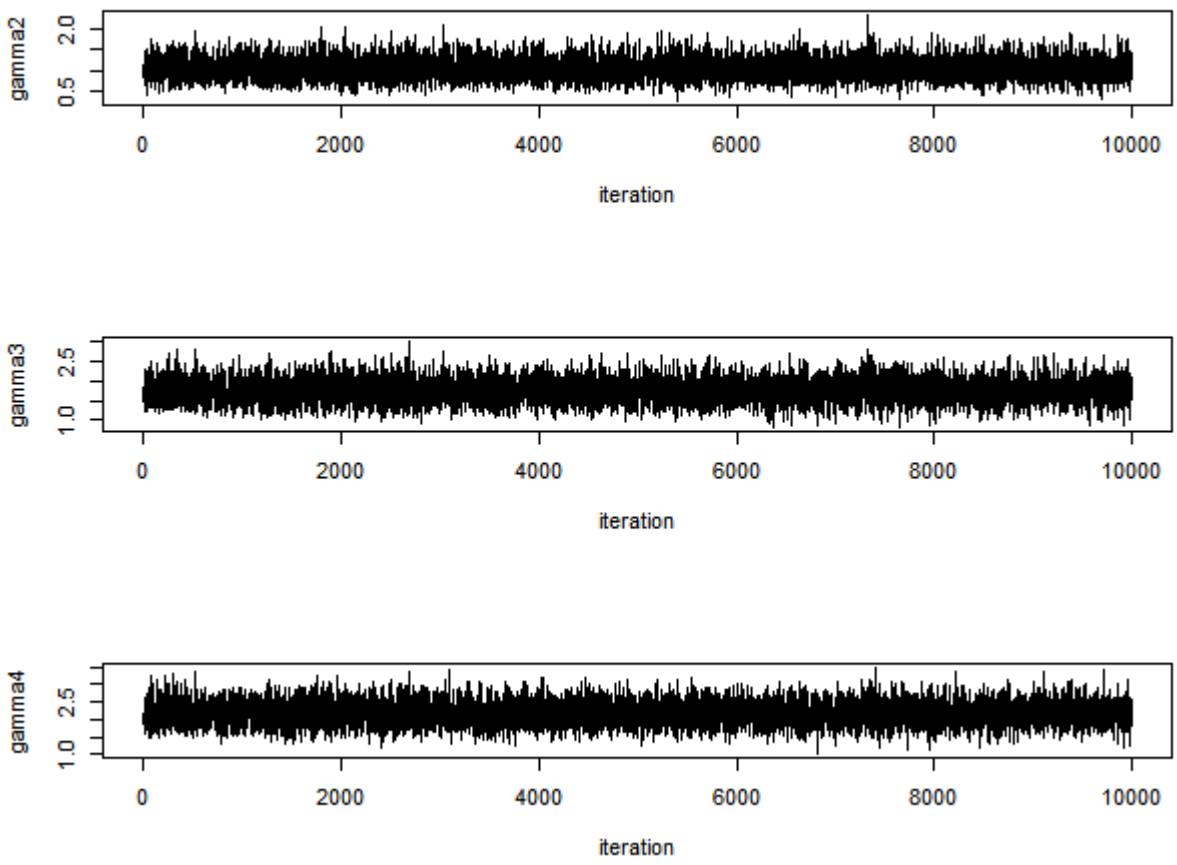
```r
par(mfrow=c(3,1))
for(i in 2:(K-1)){
  plot(gamma.sim[1:1000, i], type = 'l', xlab = 'iteration', ylab = "gamma")
}
```



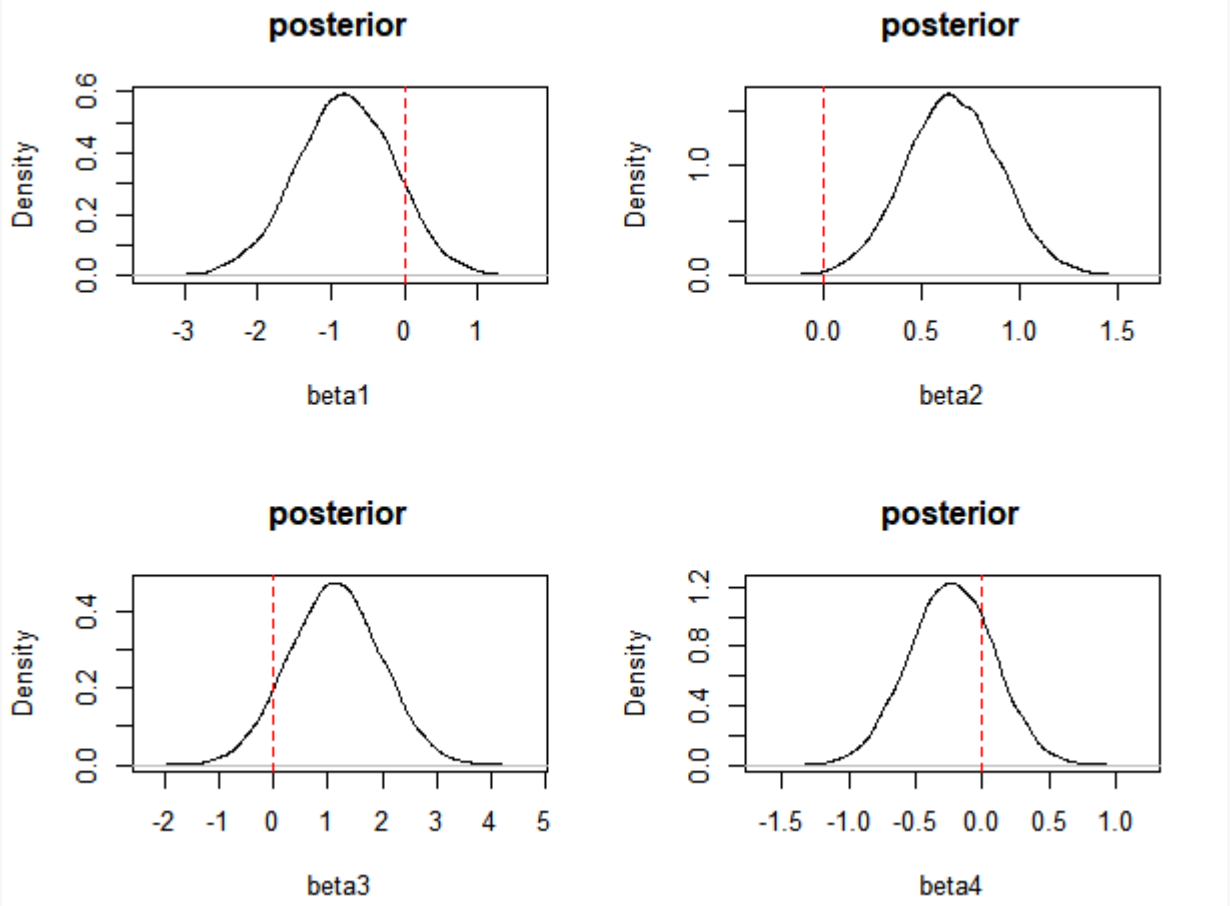

```r
#### Fig 7.3 매 100 번째의 표본을 취했을 때 회귀계수 Beta 의 경로그림 ####
par(mfrow = c(2,2))
for(i in 1:p){
  plot(beta.sim[,i], type = 'l', xlab ='iteration', ylab = paste0("beta", i))
}
```



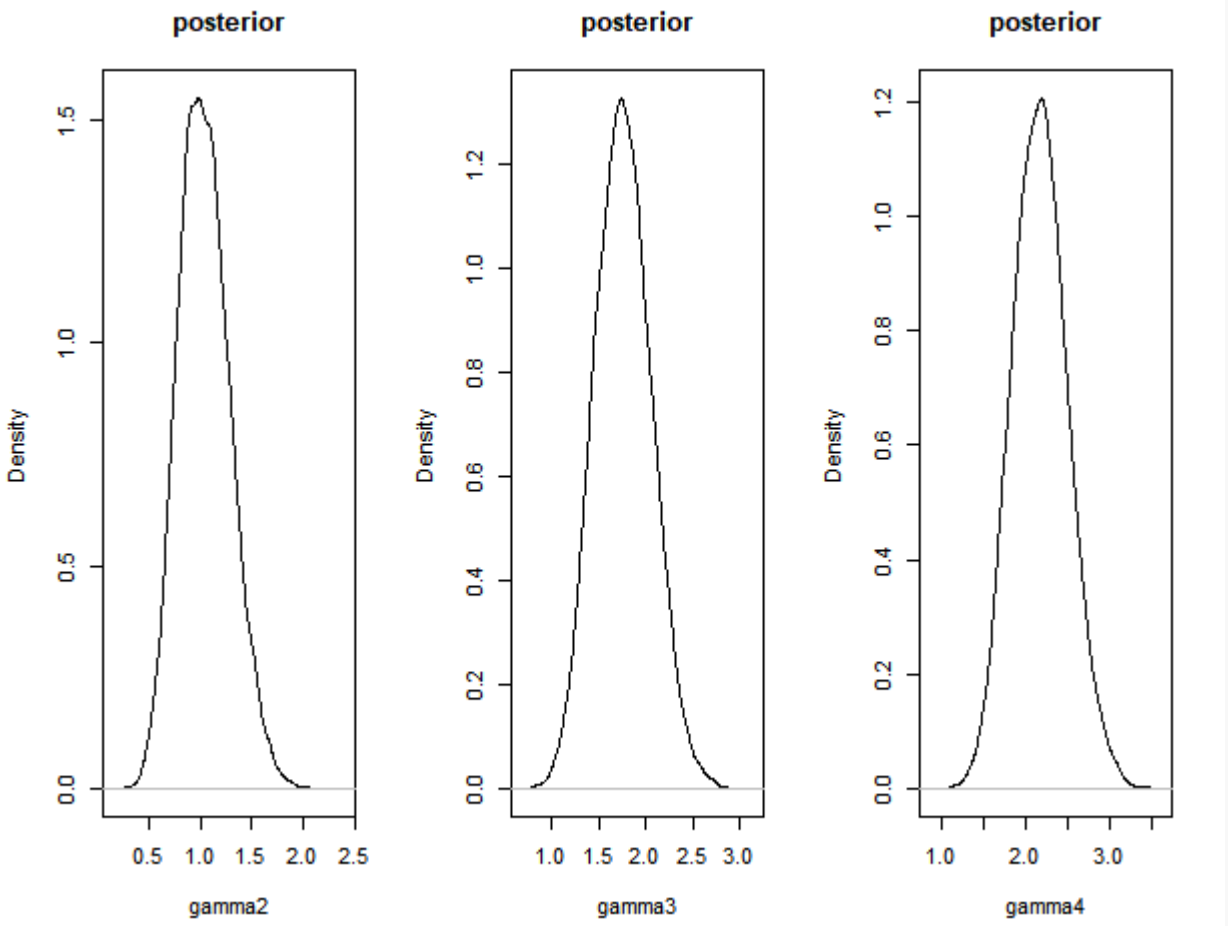

```
#### Fig 7.4 매 100 번째의 표본을 취했을 때 회귀계수 gamma 의 경로그림 ####
par(mfrow = c(3,1))
for(i in 2:(K-1)){
  plot(gamma.sim[,i], type = 'l', xlab = 'iteration', ylab = paste0('gamma',
i))
}
```



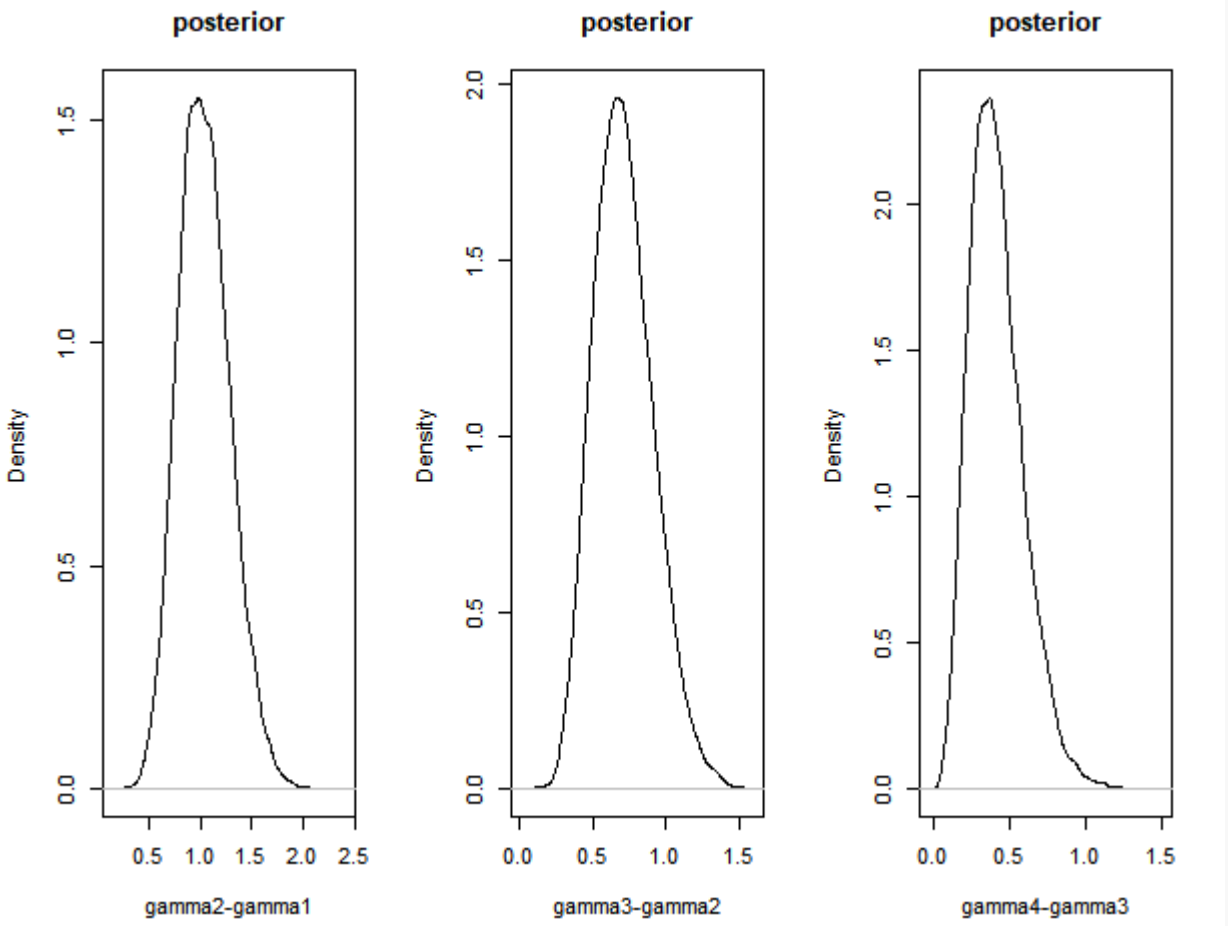

```
#### Fig 7.5 beta의 주변 사후분포 ####
par(mfrow=c(2,2))
for(i in 1:p){
  plot(density(beta.sim[,i]), type = 'l', xlab = paste0("beta", i), main = "p
osterior")
  abline(v = 0, lty = 2, col = 2)
}
```




```
#### Fig 7.6 gamma의 주변 사후분포 ####
par(mfrow = c(1,3))
for(i in 2:(K-1)){
  plot(density(gamma.sim[,i]), type = 'l', xlab = paste0("gamma",i), main = "
posterior")
}
```




```
#### Fig 7.7 gamma_k - gamma_k-1 의 주변 사후분포 ####
par(mfrow=c(1,3))
plot(density(gamma.sim[,2]-gamma.sim[,1]), type = 'l', xlab = 'gamma2-gamma1
', main = 'posterior')
plot(density(gamma.sim[,3]-gamma.sim[,2]), type = 'l', xlab = 'gamma3-gamma2
', main = 'posterior')
plot(density(gamma.sim[,4]-gamma.sim[,3]), type = 'l', xlab = 'gamma4-gamma3
', main = 'posterior')
```



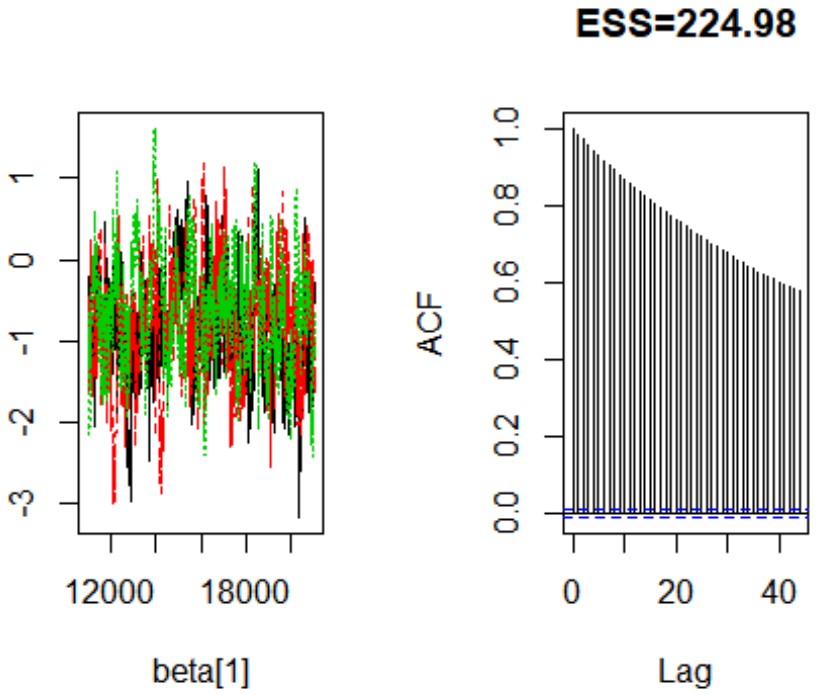

```
mean.gamma <- rep(0, K)
for(i in 1:K) mean.gamma[i] <- mean(gamma.sim[,i])

mean.beta <- rep(0, p)
for(i in 1:p)mean.beta[i] <- mean(beta.sim[,i])


#write.csv(gamma.sim, "gamma_sim.csv")
#write.csv(beta.sim, "beta_sim.csv")
```

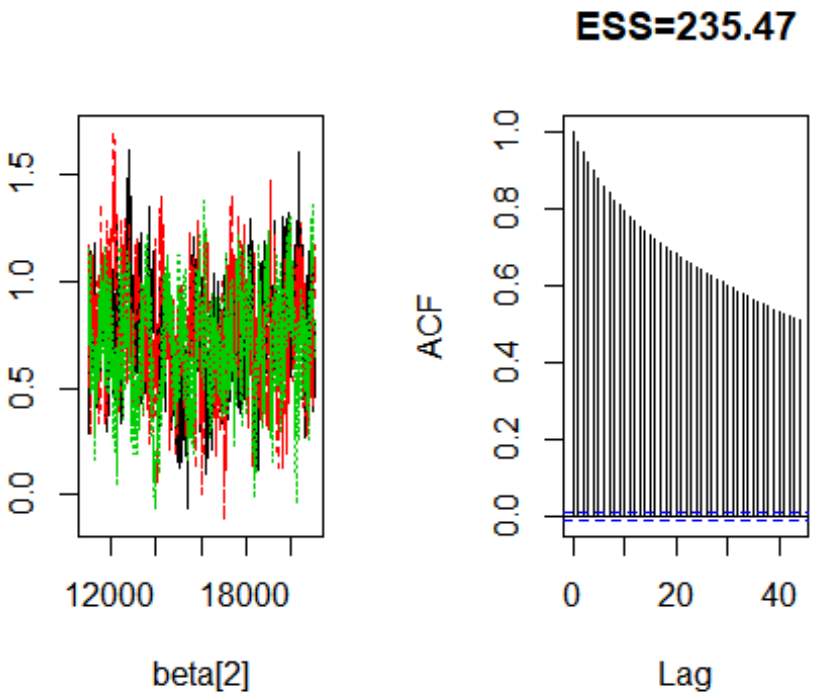
JAGS 활용

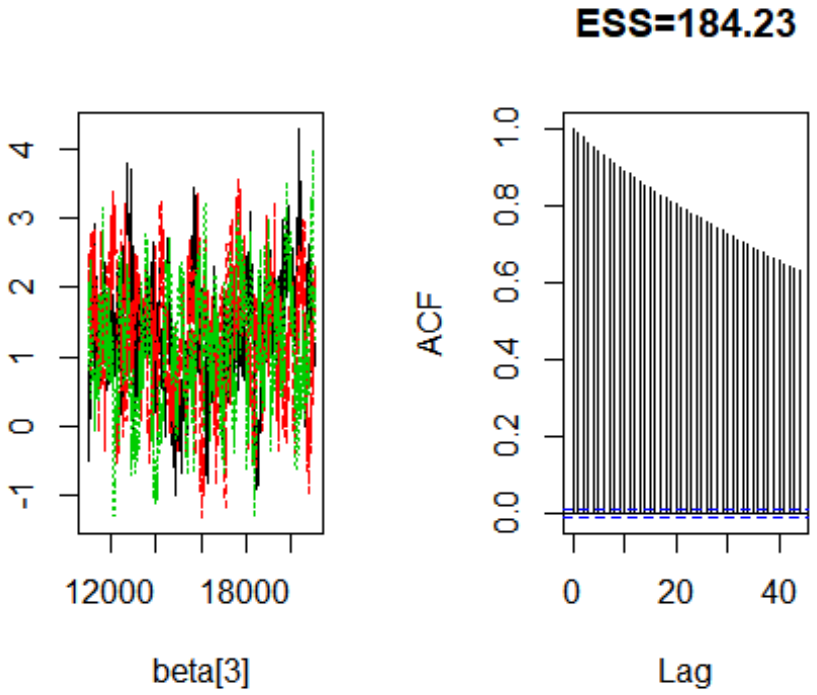
```
modelString = "
model{
for(i in 1:nData){
  y[i] ~ dcat(pr[i, 1:K])
  mu[i] <- inprod(X[i,], beta[])
  pr[i,1] <- phi((gamma1-mu[i]))
  pr[i, 2] <- phi(gamma[2] -mu[i]) - phi(gamma1-mu[i])
  for(k in 3:(K-1)){
```

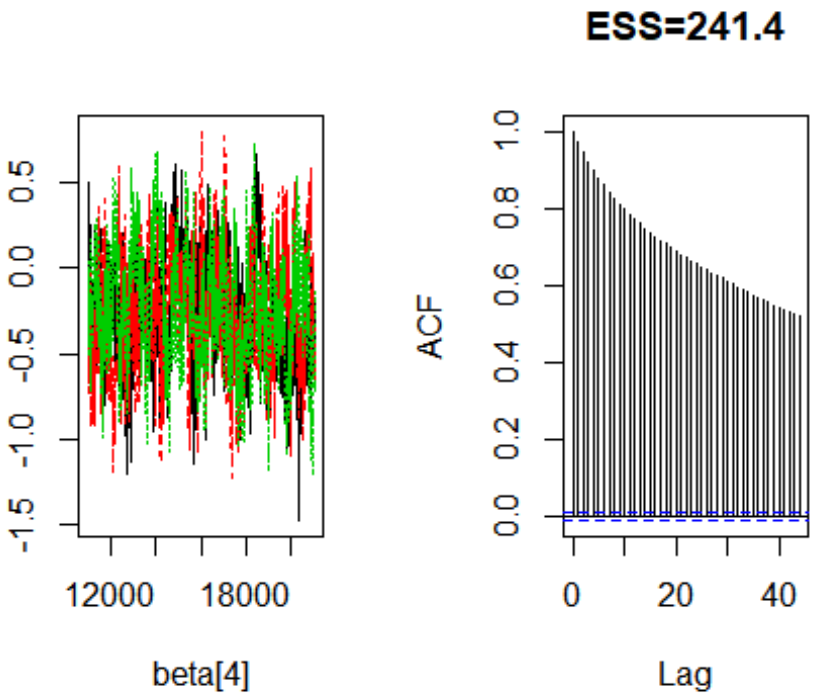
```
    pr[i,k] <- phi(gamma[k]-mu[i]) - phi(gamma[k-1] - mu[i])
  }
  pr[i,K] <- 1 - phi(gamma[K-1]-mu[i])
}

beta[1:dimBeta] ~ dmnorm(muBeta, TauBeta)

gamma1 <- 0

muGamma <- 1; tauGamma <- 0.01
for(k in 1:(K-1)){
  gamma0[k] ~ dnorm(muGamma, tauGamma)T(0,)
}
gamma[1:(K-1)] <- sort(gamma0)
}
"

writeLines(modelString, "model_ordProbit.txt")


X <- cbind(rep(1,n), stress, SES, stress*SES)
y <- mental
nLevels <- 5
nData <- length(y)
dimBeta <- ncol(X)
muBeta <- solve(t(X) %*% X) %*% t(X) %*% y
TauBeta <- 1/nData * t(X) %*% X
dataList <- list(nData = nData, K = nLevels, dimBeta = dimBeta, X = X, y = y,
 muBeta = muBeta, TauBeta = TauBeta)
betaInit <- rep(0, dimBeta)
gammaInit <- c(0:(nLevels-2)) + 0.5
initsList = list(beta = betaInit, gamma0 = gammaInit)

nChains = 3
jagsModel = jags.model(file = "model_ordProbit.txt", data = dataList, inits =
 initsList, n.chains = 3, n.adapt = 1000)

## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 40
##    Unobserved stochastic nodes: 5
##    Total graph size: 422
##
## Initializing model

update(jagsModel, n.iter = 10000)
codaSamples = coda.samples(jagsModel, variable.names = c("gamma[2:4]", "beta
```

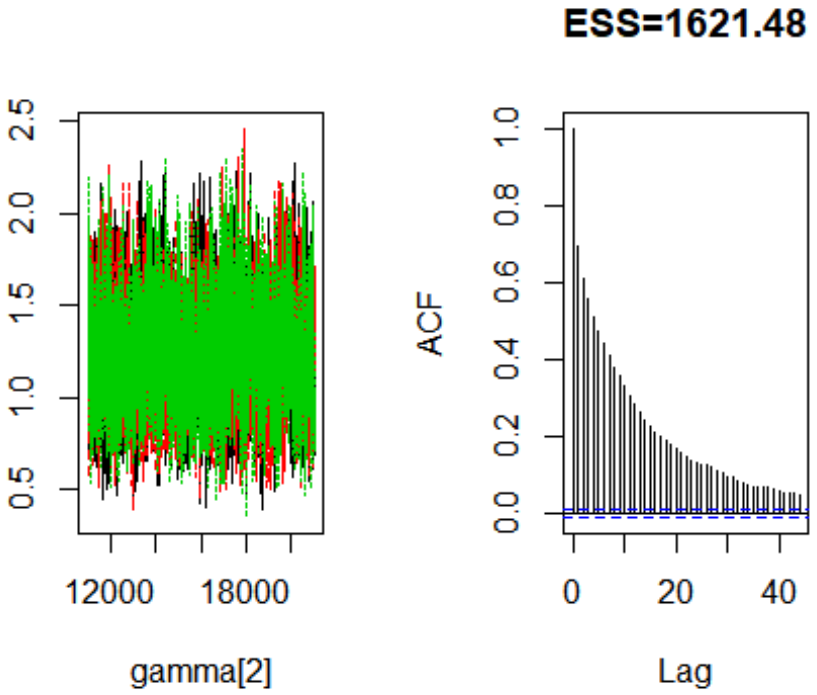
```
"), thin = 1, n.iter = 10000)

convDiag.plot = function(codaSamples, var){
  ESS = effectiveSize(codaSamples[, var])
  par(mfrow = c(1,2))
  traceplot(codaSamples[, var], xlab = var)
  acf(as.matrix(codaSamples[,var]), main = paste0("ESS=", round(ESS,2)))
}

convDiag = function(codaSamples){
  ESS = effectiveSize(codaSamples); cat("ESS = ", ESS, "\n")
  acceptRate = 1-rejectionRate(codaSamples); cat("Accept rate=", acceptRate,
"\n")
  gelman = gelman.diag(codaSamples); print(gelman)
  gelman.1 = as.matrix(gelman$psrf)
  if(max(gelman.1)>1.1) cat("Warning : Gelman Shrink Factor > 1.1", "\n")
  gelman.2 = gelman$mpsrf
  if(gelman.2>1.1) cat("Warning : Gelman Multivariate Shrink Factor > 1.1", "
\n")
}

var = colnames(codaSamples[[1]])
par(mfrow = c(length(var), 2))
for(i in 1:length(var))convDiag.plot(codaSamples, var[i])
```

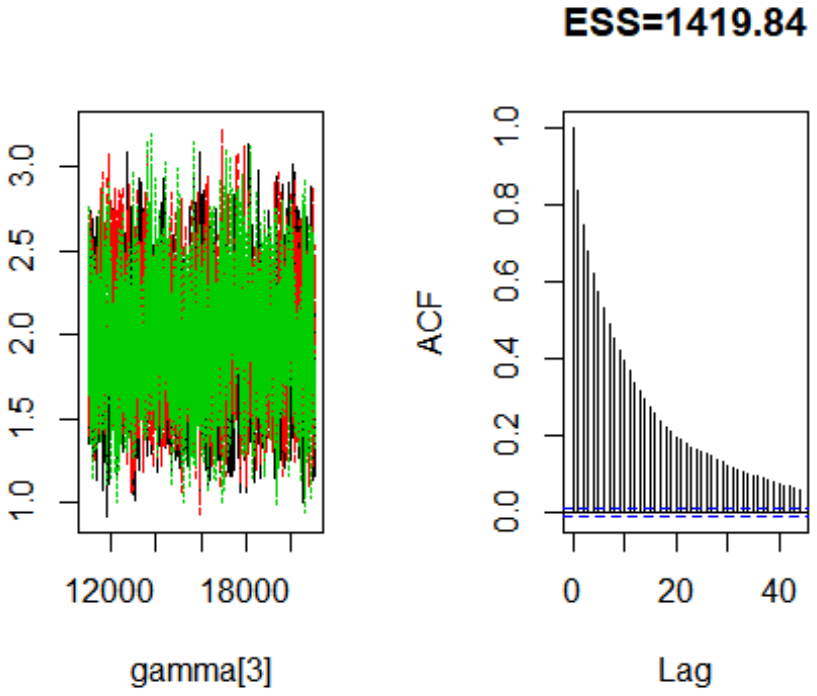

ESS=224.98
-3
-2
-1
0
1
12000
18000
beta[1]
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
Lag

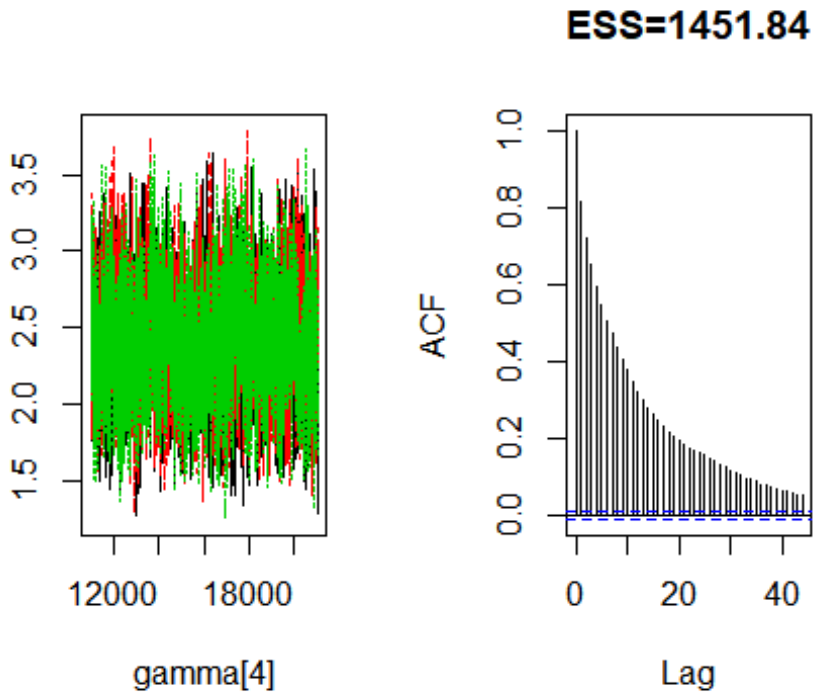

ESS=235.47
0.0
0.5
1.0
1.5
12000
18000
beta[2]
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
Lag


ESS=184.23
-1
0
1
2
3
4
12000
18000
beta[3]
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
Lag


ESS=241.4
-1.5
-1.0
-0.5
0.0
0.5
12000
18000
beta[4]
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
Lag


ESS=1621.48
0.5
1.0
1.5
2.0
2.5
12000
18000
gamma[2]
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
Lag


ESS=1419.84
1.0
1.5
2.0
2.5
3.0
12000
18000
gamma[3]
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
Lag

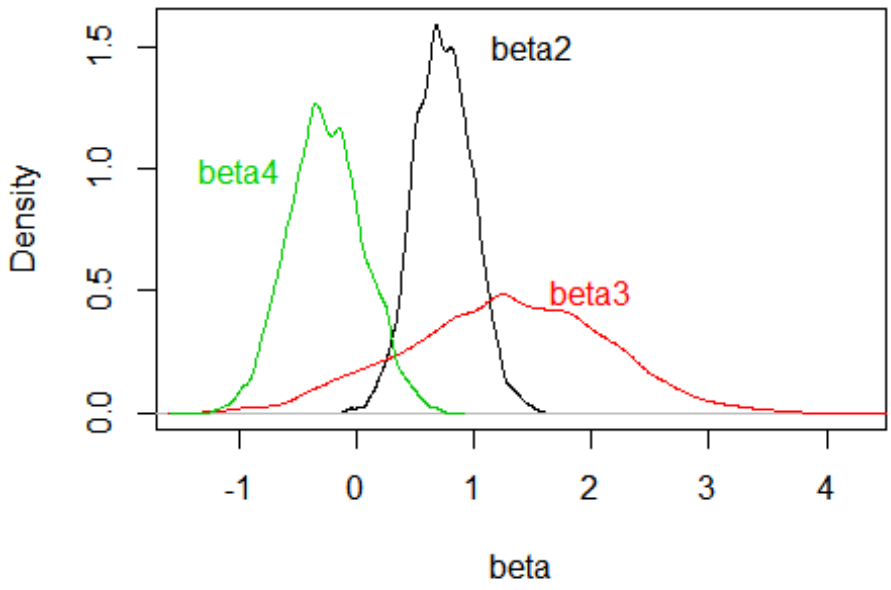



```
convDiag(codaSamples)
```

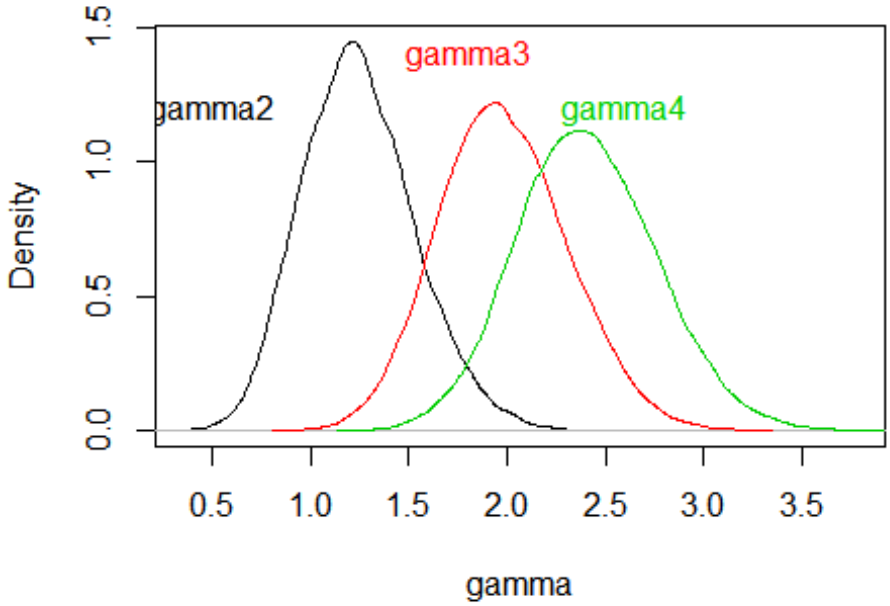
```
## ESS =  224.984 235.4713 184.2309 241.4034 1621.481 1419.839 1451.841
## Accept rate= 0.2693936 0.2693936 0.2693936 0.2693936 1 1 1
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## beta[1]        1.03       1.10
## beta[2]        1.02       1.07
## beta[3]        1.02       1.07
## beta[4]        1.01       1.04
## gamma[2]       1.00       1.00
## gamma[3]       1.00       1.00
## gamma[4]       1.00       1.00
##
## Multivariate psrf
##
## 1.02
```

```
#### Fig 7.10 JAGS 표본으로부터 추정된 모수의 사후분포 #####
mcmcSamples = as.matrix(codaSamples)
par(mfrow = c(1,1))
plot(density(mcmcSamples[,"beta[2]"]), xlim = c(min(mcmcSamples[, 2:4]), max
(mcmcSamples[, 2:4])), main = "", xlab = "beta")
text(1.5, 1.5, "beta2")
```

```
lines(density(mcmcSamples[, "beta[3]"]), col = 2)
text(2, 0.5, "beta3", col = 2)
lines(density(mcmcSamples[, "beta[4]"]), col = 3)
text(-1, 1, "beta4", col = 3)
```




```
par(mfrow = c(1,1))
plot(density(mcmcSamples[, "gamma[2]"]), xlim = c(min(mcmcSamples[,5:7]), max
(mcmcSamples[, 5:7])), main = "", xlab = "gamma")
text(0.5, 1.2, "gamma2")
lines(density(mcmcSamples[, "gamma[3]"]), col = 2)
text(1.8, 1.4, "gamma3", col = 2)
lines(density(mcmcSamples[,"gamma[4]"]), col = 3)
text(2.6, 1.2, "gamma4", col = 3)
```




```
para.hat <- apply(mcmcSamples, 2, mean)
para.hat
```

```
##    beta[1]    beta[2]    beta[3]    beta[4]   gamma[2]   gamma[3]
## -0.7785424  0.7374794  1.2580014 -0.2630291  1.2444578  1.9793060
##   gamma[4]
##  2.4048488
```

```
HPD <- apply(mcmcSamples, 2, function(x) quantile(x, c(0.025, 0.975)))
HPD
```

```
##          beta[1]   beta[2]    beta[3]    beta[4]  gamma[2] gamma[3]
## 2.5%  -2.1263010 0.2550008 -0.4135741 -0.8654589 0.7373883 1.377451
## 97.5%  0.5347398 1.2248201  2.8660253  0.3769558 1.8398144 2.640842
##       gamma[4]
## 2.5%  1.746419
## 97.5% 3.120179
```

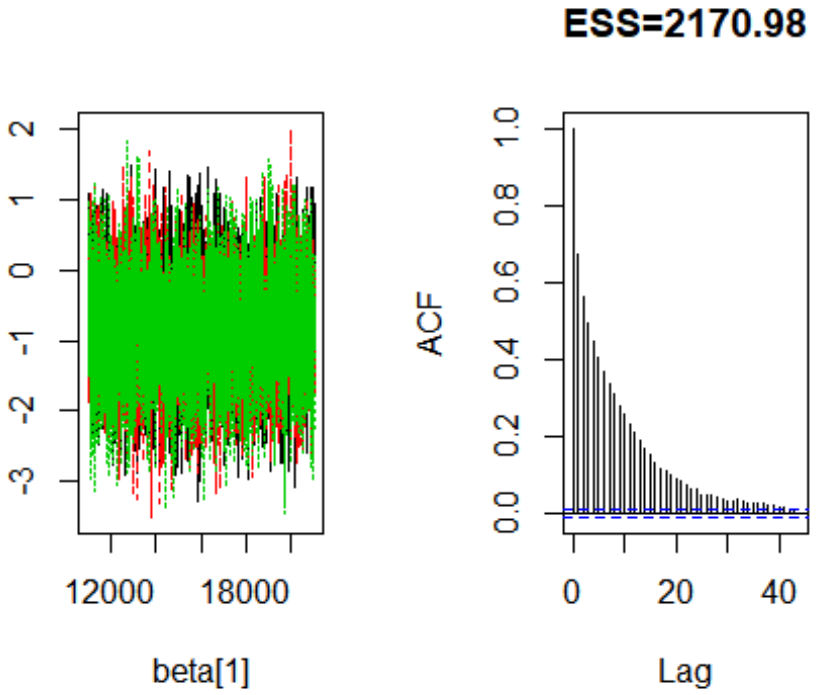
자료의 표준화

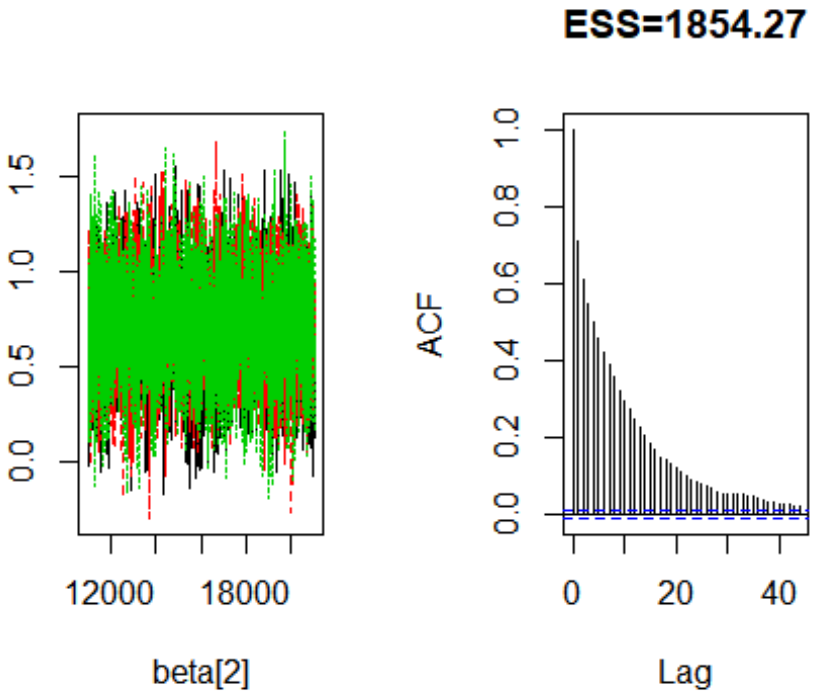
```
modelString="
data{
zX[1:nData, 1] = X[1:nData, 1]
for(j in 2:dimBeta){
Xm[j] <- mean(X[,j])
Xsd[j] <- sd(X[,j])
```

```
for(i in 1:nData){
zX[i,j] <- (X[i,j]-Xm[j])/Xsd[j]
}
}
}

model{
for(i in 1:nData){
y[i] ~ dcat(pr[i, 1:K])
pr[i, 1] <- phi((gamma1 - mu[i]))
pr[i, 2] <- max(0, phi(gamma[2] - mu[i]) - phi(gamma1 - mu[i]))

for(k in 3:(K-1)){
pr[i,k] <- max(0, phi(gamma[k]-mu[i]) - phi(gamma[k-1]-mu[i]))
}
pr[i,K] <- 1-phi(gamma[K-1]-mu[i])
mu[i] <- inprod(zX[i, ], zbeta[])
}

gamma1 <- 0

sigZbetaInv <- pow(K, -2)
for(j in 1:dimBeta){
zbeta[j] ~ dnorm(0, sigZbetaInv)
}

tauGamma <- 0.01
for(k in 1:(K-1)){
muGamma[k] <- k - 0.5
gamma[k] ~ dnorm(muGamma[k], tauGamma)T(0,)
}

for(j in 2:dimBeta){
beta[j] <- zbeta[j]/Xsd[j]
}
beta[1] <- zbeta[1] - inprod(zbeta[2:dimBeta], Xm[2:dimBeta]/Xsd[2:dimBeta])
}
"
writeLines(modelString, "model_ordProbit_std.txt")

zbetaInit <- rep(0, dimBeta)
gammaInit <- c(0:(nLevels-2)) + 0.5

dataList = list(nData=nData, K=nLevels, dimBeta = dimBeta, X=X, y = y)
initsList = list(zbeta = zbetaInit, gamma = gammaInit)
jagsModel.std = jags.model(file = "model_ordProbit_std.txt", data = dataList,
 inits = initsList, n.chains = 3, n.adapt = 1000)
```

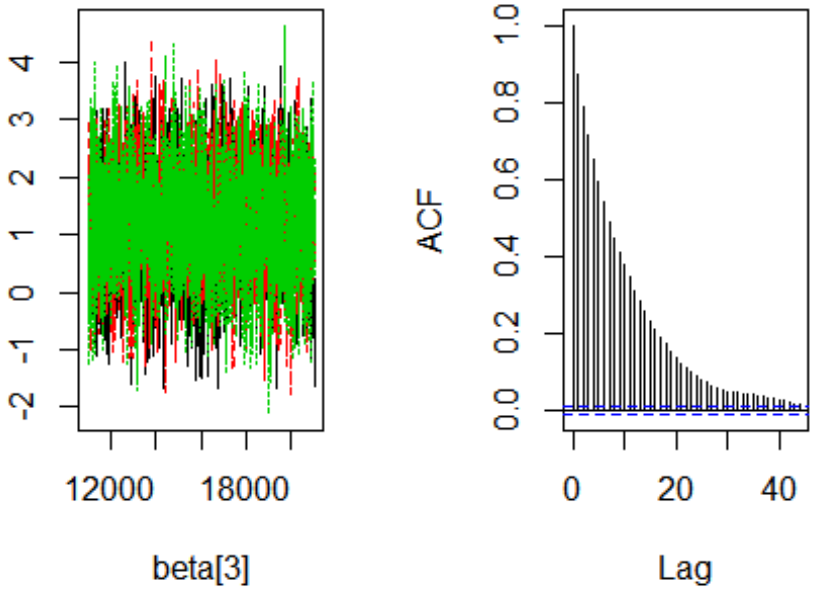
```
## Compiling data graph
##    Resolving undeclared variables
##    Allocating nodes
##    Initializing
##    Reading data back into data table
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 40
##    Unobserved stochastic nodes: 8
##    Total graph size: 608
##
## Initializing model
```

```
update(jagsModel.std, n.iter = 10000)
codaSamples = coda.samples(jagsModel.std, variable.names = c("gamma", "beta
"), n.iter = 10000)

var = colnames(codaSamples[[1]])

#### Fig 7.11 표준화된 자료 사용 시 모수의 경로그림과 자기상관 ####
for(i in 1:length(var))convDiag.plot(codaSamples, var[i])
```

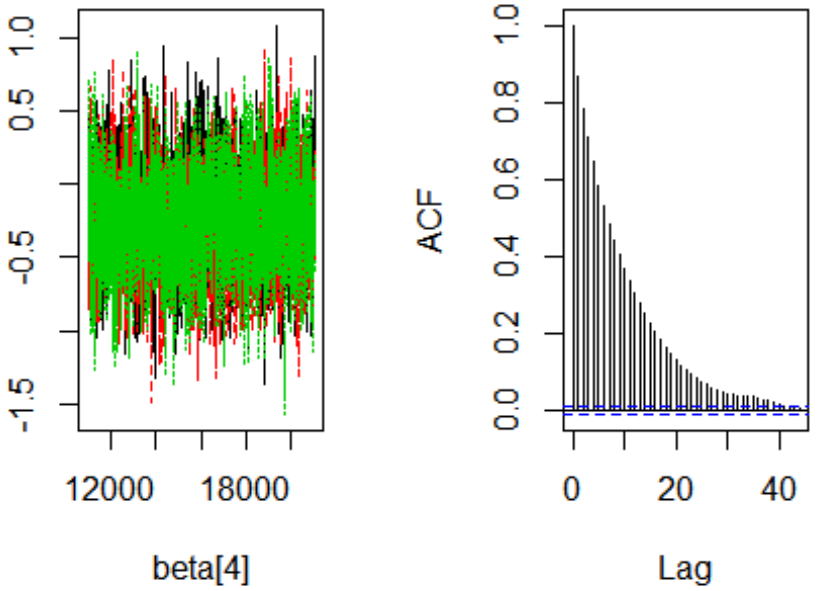

ESS=2170.98
2
1
0
-1
-2
-3
12000
18000
beta[1]
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
Lag


ESS=1854.27
1.5
1.0
0.5
0.0
12000
18000
beta[2]
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
Lag

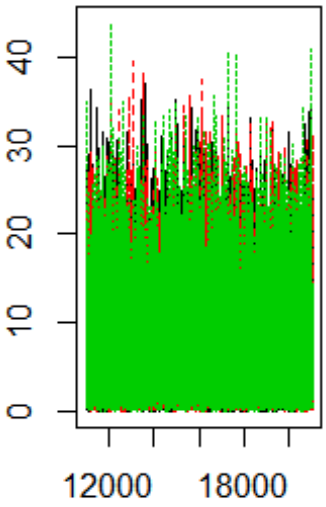
ESS=1517.43

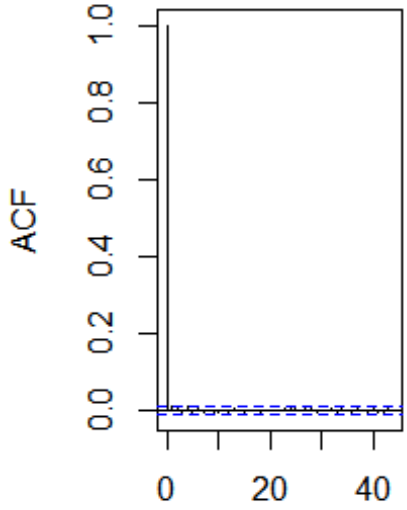


ESS=1502.36

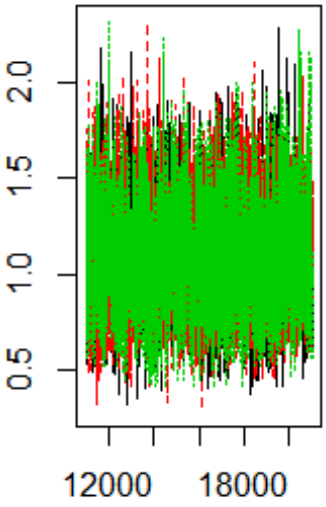


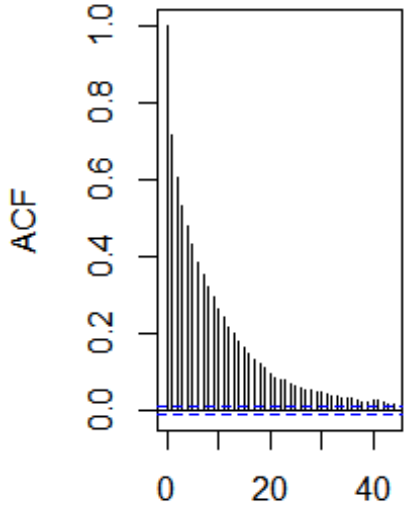
ESS=29232.49

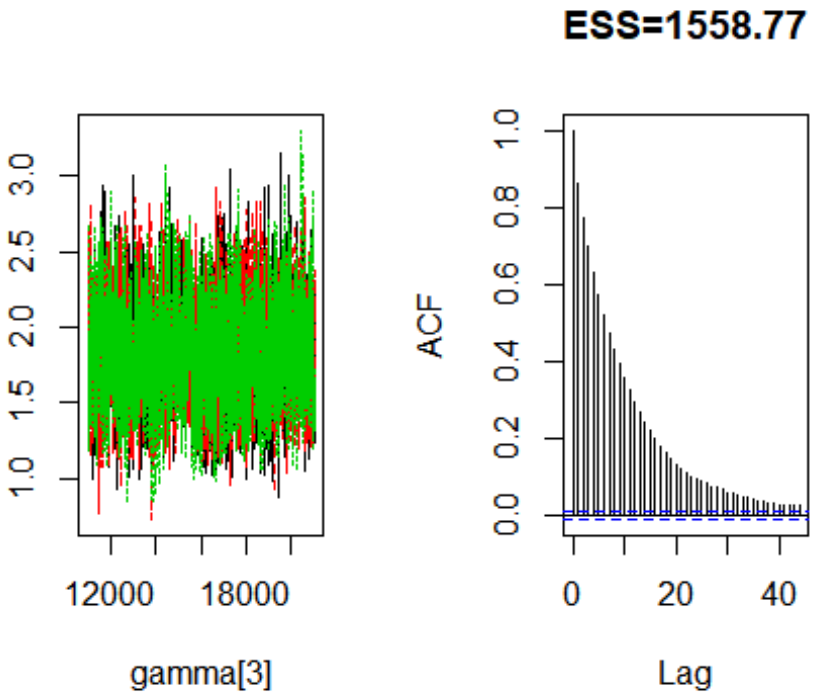


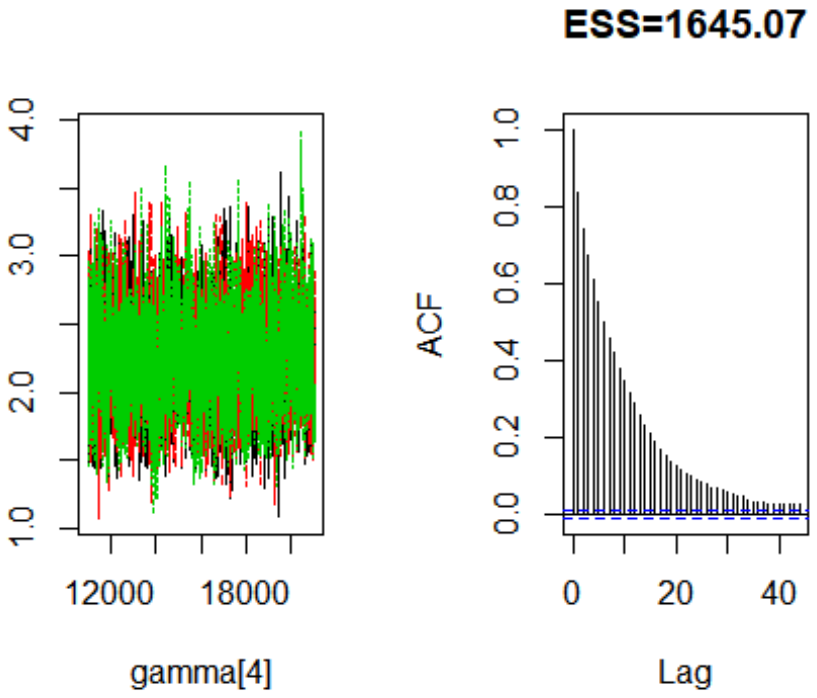


ESS=2092.39










```
convDiag(codaSamples)
```

```
## ESS =  2170.983 1854.27 1517.43 1502.359 29232.49 2092.391 1558.77 1645.07
2
## Accept rate= 1 1 1 1 1 1 1 1
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## beta[1]           1          1
## beta[2]           1          1
## beta[3]           1          1
## beta[4]           1          1
## gamma[1]          1          1
## gamma[2]          1          1
## gamma[3]          1          1
## gamma[4]          1          1
##
## Multivariate psrf
##
## 1
```