ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Bayesian Analysis of Olympic Performance (13)

Bayesian Model

Student Giorgia Treglia

Identification Number 6042123

Academic Year 2024/2025

Contents

Introduction

The 2012 Summer Olympics (the Games of the XXX Olympiad) was an international sports event that had place in London, which brought together athletes from over 200 countries, showing not only their athletic talents but also reflecting broader social and economic factors influencing national performances. While, during the Olympics, all the public was focusing on the performance of the athletes or the count of the total number of medals won, this study aims to explore the factors that may explain national performance from a statistical modelling perspective. The focus is to investigate and model the total number of medals (the response variable) won by each country, identifying which socioeconomic or demographic characteristics most influence Olympic success. Bayesian modelling techniques can be applied cause they help to better understand how income levels, population size, and related macroeconomic indicators affect the response variable. Bayesian inference is useful because it allows the incorporation of prior beliefs and a systematic model comparison using the Deviance Information Criterion (DIC). As written before, the aim of the study is to build and evaluate Bayesian models that can explain the distribution of Olympic medal counts across countries using socioeconomic indicators, more in the specific, the object is to identify the most influential variables through model comparison and variable selection techniques, in order to assess the goodness of fit and predictive accuracy of different modelling choices. By using R and Win BUGS and applying tools such as the Deviance Information Criterion and Bayesian variable selection, the goal is to offer an interpretable analysis of the factors contributing to national success in the Olympic Games.

Chapter 1

Mathematical Specification of the Model

To analyse the factors that influence Olympic success, as indicated by the total number of medals won by each nation, is useful to considerate the Bayesian generalized linear model. In this specific, let is y_i represent the total number of medals won by each country i, for $i = 1, 2, \dots, n$, where $n = 203$ countries. The response variable is a function of specific covariates related to income, population size, gross domestic product, as well as their polynomial and logarithmic transformations, which are included for potential nonlinearities in the relationships. First, the response y_i was assumed to follow a Poisson distribution with mean μ_i , where the logarithm of the mean is modelled as a linear combination of the covariates. The formula is the following:

$$y_i \sim Poisson(\mu_i) \qquad log(\mu_i) = \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}$$

where x_{ij} represents the $j-th$ covariate for country i, and β_j are the corresponding regression coefficients. The Poisson distribution allows the model to capture the effects of the predictors on the expected number of medals, but the previous analysis revealed evidence of overdispersion, which violates the assumption of the Poisson model. In order to address this issue, an alternative was considering the Negative Binomial distribution, which introduces an additional dispersion parameter r. Under the Negative Binomial formulation, the response is modelled as:

$$y_i \sim NegBin(\mu_i,\ r) \qquad log(\mu_i) = \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}$$

where r represents the degree of overdispersion, and the variance of y_i is given by $\mu_i + \mu_i^2/r$. This formulation has the same log-linear structure for the mean but relaxes the variance assumption, allowing for more flexibility and improved fit since the data have more variability than the Poisson model can handle. For both models, weakly informative priors were assigned to the regression coefficients $\beta_i \sim N(0,\ 0.001)$, reflecting initial uncertainty while ensuring a larger range of possible values. In the Negative Binomial model, the overdispersion parameter r was assigned a Gamma prior with parameters chosen to reflect the prior information. Model estimation was performed using Markov Chain Monte Carlo (MCMC) methods implemented via Win BUGS, through R using the R2WinBUGS package. Posterior inference is based on three parallel chains of 10,000 iterations each, with a burn-in of 2,000 and a thinning interval of 5. Convergence was assessed through standard

diagnostics including trace plots, the potential scale reduction factor, and the effective sample size. So, model comparison between the Poisson and Negative Binomial specifications was done using the Deviance Information Criterion (DIC), which balances model fit and complexity.

1.2 Posterior Analysis

Following the specification of the Poisson and Negative Binomial models, both were estimated using Markov Chain Monte Carlo (MCMC) methods. Convergence diagnostics were applied to all parameters in each model, including the Gelman-Rubin statistic, which approached 1.0 for most parameters in both models, indicating acceptable convergence. However, some coefficients in both structures showed higher values and low effective sample sizes, suggesting that some areas where poorly mixed, or convergence was incomplete. These issues were more pronounced in the Poisson model, particularly for the higher order terms and the deviance. The Deviance Information Criterion (DIC), used to assess and compare model fit, is showed in the table below.

MODEL	DIC	PD
Poisson	787.3	15.6
Negative Binomial	625.5	15.8

For the Poisson model, the DIC is 787.3 with an effective number of parameters of 15.6. In contrast, the Negative Binomial model resulted in a lower DIC of 625.5, with a similar complexity level (number of parameters of 15.8). The reduction of DIC provides strong evidence in favour of the Negative Binomial model, suggesting that it better captures the structure and variability.

Chapter 2

Variable Selection Using Win BUGS

To build the Bayesian model, a variable selection was performed using Bayesian Adaptive Sampling (BAS), where variables such as GoldMedals, Silver, IncomeCubed, Income.SQ, PopnCubed, and PopnSQ had already been excluded based on low posterior inclusion probabilities. Model selection was performed by evaluating the impact of removing individual or groups of covariates and to determine the most appropriate structure for the negative binomial regression, models were compared using the Deviance Information Criterion (DIC) and the effective number of parameters. The table below summarizes the performance of these models:

VARIABLES REMOVED	DIC	PD	CONVERGENCE NOTES
Full Model (After BAS)	618.6	11.8	(Rhat > 1.1)
Ln.GDP	619.2	8.4	Acceptable
Ln.PopnSize, Ln.GDP	662.9	8.4	Acceptable
PopnSize, Ln.GDP	617.1	8.9	(Rhat > 2)
Income, Ln.GDP	621.9	9.3	Acceptable
Ln.Income, Ln.GDP	651.4	8.9	Acceptable
Ln.GDP, PopnSize, Income	619.7	8.6	(Rhat > 1.2)
Ln.GDP, PopnSize, GDPCubed	**615.1**	**8.8**	**Good**
Ln.GDP, GDPCubed, Income, PopnSize	619.3	8.5	(Rhat > 3)
GDP, Income, PopnSize, Ln.GDP	644.6	6.9	(Rhat > 1.2)
GDPSQ, Income, PopnSize, Ln.GDP	625.3	7.2	(Rhat > 1.2)

Compared to the model obtained after BAS, which have a DIC of 618.6 but suffered from serious convergence issues where several parameters (beta6–7, beta10–12) showed Rhat values far above 1.1, with some reaching beyond 3.5 and extremely low effective sample sizes (3–6), the model excluding Ln.GDP, PopnSize, and GDPCubed variables showed better performace. Based on DIC values alone, this model has a DIC of 615.1, the lowest among all considered models, and exhibited good convergence diagnostics, with Rhat values close to 1.0 and acceptable effective sample sizes across all parameters. Given the data structure, no hierarchical clustering or repeated measures were

present. Therefore, the introduction of random effects was not deep necessary. Moreover, the Negative Binomial model already accounts for over-dispersion relative to the Poisson distribution, and the DIC values, along with convergence diagnostics, confirmed that no additional over-dispersion component was required. In conclusion, based on the DIC comparisons, stepwise variable selection, and convergence analysis, the final model was chosen to be the Negative Binomial regression excluding the variables Ln.GDP, PopnSize, and GDPCubed.

2.2 Inference Based on the Final Model

The final Negative Binomial regression model was fitted using three chains, each with 10,000 iterations, discarding the first 2,000 iterations as burn-in and thinning every five samples. The posterior summaries for each model parameter are presented below:

PARAMETER	MEAN	SD	RHAT	N.EFF
B0 (Intercept)	3.1	0.4	1.0	87
B1 (Bronze)	0.4	0.1	1.0	2800
B2 (BordaPoints)	0.0	0.0	1.0	1200
B3 (Income)	-0.1	0.1	1.0	300
B4 (GDP)	-12.0	2.7	1.1	760
B5 (GDPSQ)	3.6	1.4	1.1	320
B6 (Ln.Income)	0.8	0.1	1.0	150
B7 (Ln.PopnSize)	0.5	0.1	1.0	160
r	1.2	0.3	1.0	710

Instead, the final model predicting Total Medals is:

$$TotalMedal_i = B_0 + B_1(Bronze_i) + B_2(BordaPoints_i) + B_3(Income_i) + B_4(GDP_i) + B_5(GDPSQ_i) + B_6(Ln.Income_i) + B_7(Ln.PopnSize_i)$$

The 95% confidence interval suggests that Bronze (0.2, 0.5), GDP (-16.5, -6.2), GDPSQ (0.3, 5.8), Ln.Income (0.5, 1.0), and Ln.PopnSize (0.4, 0.7) are important predictors, as their intervals do not contain zero, while the coefficient for Income (-0.3, 0.0) has a credible interval that includes zero, indicating weaker evidence for its influence. Meanwhile, BordaPoints (0.0, 0.0) shows a coefficient very close to zero with no real variability, suggesting that it does not play a significant role in predicting the number of medals. The interpretation of the model suggests that winning more bronze

medals is associated with an increase in total medals won; while GDP is associated with fewer medals (as shown by the negative coefficient), but the positive coefficient for GDP squared shows that there is no-linear relationship between GDP and Medal Count. Additionally, higher income and larger population sizes are positively associated with winning more medals, as indicated by the positive effects of log-transformed Income and Population Size. The intercept instead indicates the expected number of medals when all predictors are at their baseline values.

The convergence and mixing behaviour of the MCMC chains for the final model were studied through trace plots and autocorrelation functions (ACF), which are presented in the appendix. The trace plots show that the chains for most parameters mix reasonably well, although some parameters (like beta4) display slower convergence patterns. Also the ACF plots indicate that for several parameters, particularly beta4, there is high autocorrelation across many lags, suggesting that the samples are not fully independent, but overall, the diagnostics support that the MCMC algorithm has sufficiently explored the posterior distribution for reliable inference.

2.3 Variable Selection using BAS

Variable selection was performed using the Bayesian Adaptive Sampling (BAS), where six different prior specifications were used, combining three choices for the prior on the coefficients (BIC approximation, g-prior, and hyper-g prior) with two options for the prior on the model space (uniform and beta-binomial). For each combination, the most probable models were identified, and posterior inclusion probabilities (PIP) were calculated for all variables, as showed in the table:

PARAMETER	BIC.U	BIC.BB	GPRIOR.U	GPRIOR.BB	HYPERG.U	HYPERG.BB
Intercept	1	1	1	1	1	1
GoldMedals	0	0	0	0	0	0
Silver	0	0	0	0	0	0
Bronze	0.99	0.99	0.99	0.99	0.99	0.99
BordaPoints	0.99	0.99	0.99	0.99	0.99	0.99
Income	0.06	0.04	0.07	0.01	0	0
PopnSize	0.33	0.31	0.06	0.01	0.14	0.08
GDP	0.99	0.99	0.1	0.019	0.99	0.99
IncomeSQ	0.068	0.04	0.06	0.01	0.003	0
PopnSQ	0.33	0.29	0.06	0.01	0.14	0.07
IncomeCubed	0.06	0.039	0.06	0.01	0.003	0

PopnCubed	0.34	0.32	0.06	0.01	0.14	0.08
GDPSQ	0.98	0.98	0.12	0.02	0.98	0.99
GDPCubed	0.12	0.08	0.13	0.03	0.02	0.01
Ln.Income	0.07	0.04	0.07	0.01	0.003	0.001
Ln.PopnSize	0.08	0.04	0.07	0.01	0.004	0.001
Ln.GDP	0.09	0.05	0.08	0.02	0.004	0.001

Across all prior, Bronze, Borda Points, GDP, and GDPSQ consistently have posterior inclusion probabilities close to 1, confirming their essential role in explaining the variability of the medal counts. In contrast, GoldMedals, Silver, and other variables such as IncomeCubed and GDPCubed showed very low inclusion probabilities under every prior specification, suggesting that their contribution to the predictive performance of the model is low. Regarding model identification, under the BIC prior with a uniform prior (with similar results when using Beta-Binomial prior) the MAP model included Bronze, Borda Points, GDP, and GDPSQ, with PopnCubed also appearing in some of the highest probability models. The g-prior selects mainly Bronze, Borda Points and GDP, while the hyper-g prior slightly increased the inclusion probability for variables like PopnSize and PopnCubed. Summarizing across priors, the median probability model (including variables with PIP > 0.5) consistently comprised Bronze, Borda Points, GDP, and GDPSQ, suggesting that these variables are strongly associated with the medal count and should be retained in the final model.

Conclusion

This study aimed to investigate the socioeconomic determinants of Olympic success using Bayesian statistical modelling. Starting from a comprehensive set of covariates, both Poisson and Negative Binomial regression models were considered to account for the count nature of the medal data and potential overdispersion. Through model comparison based on the Deviance Information Criterion (DIC), the Negative Binomial model was clearly favoured, providing a better fit for the observed data variability. The final model demonstrated acceptable convergence diagnostics across parameters and produced interpretable results, reinforcing that economic strength, particularly GDP related measures, and a broader athletic success (like bronze medal counts and overall performance scores like Borda Points) are major drivers behind a nation's overall medal count. Interestingly, GDP showed a non-linear effect, with the squared term compensating for the initial negative association, suggesting that economic prosperity benefits Olympic performance. Overall, the Bayesian framework provided a flexible and rigorous approach for model selection and inference, allowing to systematically identify the most influential predictors while accounting for model uncertainty.

Code

```
library(R2OpenBUGS)
library(coda)
library(BAS)
setwd("C:/Users/Armando/Downloads/Bayesian Model/")
data <- read.csv("13_Olympics.csv", sep = ",", stringsAsFactors = FALSE)
data <- data[2:18]
data <- na.omit(data)
colnames(data)

# Full Model Poisson

data_list <- list(TotalMedals = data$TotalMedals, GoldMedals = data$GoldMedals,
    Silver = data$Silver, Bronze = data$Bronze, BordaPoints = data$BordaPoints,
    Income = data$Income, PopnSize = data$PopnSize, GDP = data$GDP,
    IncomeSQ = data$Income.SQ, PopnSQ = data$PopnSQ, IncomeCubed = data$IncomeCubed,
    PopnCubed = data$PopnCubed, GDPSQ = data$GDPSQ, GDPCubed = data$GDPCubed,
    LnIncome = data$Ln.Income., LnPopnSize = data$Ln.PopnSize.,
    LnGDP = data$Ln.GDP., N = nrow(data))

inits <- function() {
  list( beta0 = 0, beta1 = 0, beta2 = 0, beta3 = 0, beta4 = 0, beta5 = 0,
        beta6 = 0, beta7 = 0, beta8 = 0, beta9 = 0, beta10 = 0, beta11 = 0,
        beta12 = 0, beta13 = 0, beta14 = 0, beta15 = 0, beta16 = 0)
}

params <- c( "beta0", "beta1", "beta2", "beta3", "beta4", "beta5", "beta6",
             "beta7", "beta8", "beta9", "beta10", "beta11", "beta12",
             "beta13", "beta14", "beta15", "beta16")

bugs <- bugs(data_list, inits, parameters = params, model.file = "Poisson.txt",
             n.chains = 3, n.iter = 10000, n.burnin = 2000, n.thin = 5)
print(bugs)

# DIC: 787
# Full Model neg-bin

data_list <- list(TotalMedals = data$TotalMedals, GoldMedals = data$GoldMedals,
    Silver = data$Silver, Bronze = data$Bronze, BordaPoints = data$BordaPoints,
    Income = data$Income, PopnSize = data$PopnSize, GDP = data$GDP,
    IncomeSQ = data$Income.SQ, PopnSQ = data$PopnSQ, IncomeCubed = data$IncomeCubed,
    PopnCubed = data$PopnCubed, GDPSQ = data$GDPSQ, GDPCubed = data$GDPCubed,
    LnIncome = data$Ln.Income., LnPopnSize = data$Ln.PopnSize.,
    LnGDP = data$Ln.GDP., N = nrow(data))

inits <- function() {
  list( beta0 = 0, beta1 = 0, beta2 = 0, beta3 = 0, beta4 = 0, beta5 = 0,
```

```
        beta6 = 0, beta7 = 0, beta8 = 0, beta9 = 0, beta10 = 0, beta11 = 0,
        beta12 = 0, beta13 = 0, beta14 = 0, beta15 = 0, beta16 = 0, r = 1)
}

params <- c("beta0", "beta1", "beta2", "beta3", "beta4", "beta5", "beta6",
            "beta7", "beta8", "beta9", "beta10", "beta11", "beta12",
            "beta13", "beta14", "beta15", "beta16", "r")

bugs_1 <- bugs(data_list, inits, parameters = params, model.file = "BinNeg.txt",
             n.chains = 3, n.iter = 10000, n.burnin = 2000, n.thin = 5)
print(bugs_1)

# DIC: 625
# Variable Selection

bas_bic <- bas.lm(TotalMedals ~ ., data, prior = "BIC",
modelprior = uniform(), method = "MCMC")
bas_bic_bb <- bas.lm(TotalMedals ~ ., data, prior = "BIC",
modelprior = beta.binomial(),
method = "MCMC")
bas_gprior <- bas.lm(TotalMedals ~ ., data, prior = "g-prior",
modelprior = uniform(), method = "MCMC")
bas_gprior_bb <- bas.lm(TotalMedals ~ ., data, prior = "g-prior",
modelprior = beta.binomial(), method = "MCMC")
bas_hyperg <- bas.lm(TotalMedals ~ ., data, prior = "hyper-g",
modelprior = uniform(), method = "MCMC")
bas_hyperg_bb <- bas.lm(TotalMedals ~ ., data, prior = "hyper-g",
modelprior = beta.binomial(), method = "MCMC")

summary(bas_bic)
summary(bas_bic_bb)
summary(bas_gprior)
summary(bas_gprior_bb)
summary(bas_hyperg)
summary(bas_hyperg_bb)

# Remove Selected Variables
data <- data[, names(data) != "GoldMedals"]
data <- data[, names(data) != "Silver"]
data <- data[, names(data) != "IncomeCubed"]
data <- data[, names(data) != "Income.SQ"]

# selected model neg-bin

data_list <- list(TotalMedals = data$TotalMedals, Bronze = data$Bronze,
BordaPoints = data$BordaPoints, Income = data$Income, PopnSize = data$PopnSize,
GDP = data$GDP, PopnSQ = data$PopnSQ, PopnCubed = data$PopnCubed,
GDPSQ = data$GDPSQ, GDPCubed = data$GDPCubed, LnIncome = data$Ln.Income.,
LnPopnSize = data$Ln.PopnSize., LnGDP = data$Ln.GDP., N = nrow(data))

inits <- function() {
  list( beta0 = 0, beta1 = 0, beta2 = 0, beta3 = 0, beta4 = 0, beta5 = 0,
        beta6 = 0, beta7 = 0, beta8 = 0, beta9 = 0, beta10 = 0, beta11 = 0,
        beta12 = 0, r = 1)
```

```
}

params <- c("beta0", "beta1", "beta2", "beta3", "beta4", "beta5", "beta6",
            "beta7", "beta8", "beta9", "beta10", "beta11","beta12", "r")

bugs_2 <- bugs(data_list, inits, parameters = params, model.file = "BinNeg_2.txt",
             n.chains = 3, n.iter = 10000, n.burnin = 2000, n.thin = 5)
print(bugs_2)

# DIC: 618

gelman.diag(bugs_2)
samples <- as.mcmc.list(bugs_2)
samples_matrix <- as.matrix(samples)

par(mfrow = c(2,2))
traceplot(samples[, "beta1"], main = "Traceplot for beta1")
traceplot(samples[, "beta2"], main = "Traceplot for beta2")
traceplot(samples[, "beta3"], main = "Traceplot for beta3")
traceplot(samples[, "beta4"], main = "Traceplot for beta4")

par(mfrow=c(2,2))
traceplot(samples[, "beta5"], main = "Traceplot for beta5")
traceplot(samples[, "beta6"], main = "Traceplot for beta6")
traceplot(samples[, "beta7"], main = "Traceplot for beta7")
traceplot(samples[, "beta8"], main = "Traceplot for beta8")

par(mfrow=c(2,2))
traceplot(samples[, "beta9"], main = "Traceplot for beta9")
traceplot(samples[, "beta10"], main = "Traceplot for beta10")
traceplot(samples[, "beta11"], main = "Traceplot for beta11")
traceplot(samples[, "beta12"], main = "Traceplot for beta12")

# Variable Selection

data <- data[, names(data) != "PopnCubed"]
data <- data[, names(data) != "PopnSQ"]

# selected model neg-bin

data_list <- list(TotalMedals = data$TotalMedals, Bronze = data$Bronze,
BordaPoints = data$BordaPoints, Income = data$Income, PopnSize = data$PopnSize,
GDP = data$GDP, GDPSQ = data$GDPSQ, GDPCubed = data$GDPCubed,
LnIncome = data$Ln.Income., LnPopnSize = data$Ln.PopnSize.,
LnGDP = data$Ln.GDP., N = nrow(data))

inits <- function() {
  list( beta0 = 0, beta1 = 0, beta2 = 0, beta3 = 0, beta4 = 0, beta5 = 0,
        beta6 = 0, beta7 = 0, beta8 = 0, beta9 = 0, beta10 = 0, r = 1)
}

params <- c("beta0", "beta1", "beta2", "beta3", "beta4", "beta5", "beta6",
            "beta7", "beta8", "beta9", "beta10", "r")
```

```
bugs_3 <- bugs(data_list, inits, parameters = params, model.file = "BinNeg_3.txt",
             n.chains = 3, n.iter = 10000, n.burnin = 2000, n.thin = 5)
print(bugs_3)

# DIC: 618

samples <- as.mcmc.list(bugs_3)
samples_matrix <- as.matrix(samples)
gelman.diag(bugs_3)

# Remove variables
data <- data[, names(data) != "Ln.GDP."]
data <- data[, names(data) != "PopnSize"]
data <- data[, names(data) != "GDPCubed"]

# selected model neg-bin

data_list <- list(TotalMedals = data$TotalMedals, Bronze = data$Bronze,
                  BordaPoints = data$BordaPoints, Income = data$Income,
                  GDP = data$GDP, GDPSQ = data$GDPSQ, LnIncome = data$Ln.Income.,
                  LnPopnSize = data$Ln.PopnSize., N = nrow(data))

inits <- function() {
  list( beta0 = 0, beta1 = 0, beta2 = 0, beta3 = 0, beta4 = 0, beta5 = 0,
        beta6 = 0, beta7 = 0, r = 1)
}

params <- c("beta0", "beta1", "beta2", "beta3", "beta4", "beta5", "beta6",
"beta7", "r")

bugs_4 <- bugs(data_list, inits, parameters = params, model.file = "BinNeg_4.txt",
               n.chains = 3, n.iter = 10000, n.burnin = 2000, n.thin = 5)
print(bugs_4)

# ln gdp : DIC 619.2 pd 8.4
# ln popsize and ln gdp: DIC 662.9 pd = 8.4
# popsize e ln gdp: DIC 617.1 pd = 8.9
# income e ln gdp: DIC 621.9  pd = 9.3
# ln income e ln gdp: DIC 651.4 pd = 8.9

# income, popnsize, ln gdp: DIC 619.7 pd = 8.6
# PopnSize, Ln GDP, GDPCubed: DIC 615.1 pd = 8.8 good
# Popnsize,LNGDP, income, GDPCubed: DIC 619.3 pd 8.5
# Popnsize, lngdp, income, gdpcubed, gdp : DIC 644.6 pd = 6.9
# Popnsize, lngdp, income, gdpcubed, gdpsq: DIC 625.3 pd 7.2

samples <- as.mcmc.list(bugs_4)
samples_matrix <- as.matrix(samples)

par(mfrow = c(2,2))
traceplot(samples[, "beta1"], main = "Traceplot for beta1")
traceplot(samples[, "beta2"], main = "Traceplot for beta2")
traceplot(samples[, "beta3"], main = "Traceplot for beta3")
traceplot(samples[, "beta4"], main = "Traceplot for beta4")
```

```
par(mfrow=c(2,2))
traceplot(samples[, "beta5"], main = "Traceplot for beta5")
traceplot(samples[, "beta6"], main = "Traceplot for beta6")
traceplot(samples[, "beta7"], main = "Traceplot for beta7")

par(mfrow = c(2,2))
densplot(samples[, "beta1"], main = "Density for beta1")
densplot(samples[, "beta2"], main = "Density for beta2")
densplot(samples[, "beta3"], main = "Density for beta3")
densplot(samples[, "beta4"], main = "Density for beta4")

par(mfrow = c(2,2))
densplot(samples[, "beta5"], main = "Density for beta5")
densplot(samples[, "beta6"], main = "Density for beta6")
densplot(samples[, "beta7"], main = "Density for beta7")

par(mfrow = c(2,2))
acf(samples_matrix[, "beta1"], main = "ACF for beta1")
acf(samples_matrix[, "beta2"], main = "ACF for beta2")
acf(samples_matrix[, "beta3"], main = "ACF for beta3")
acf(samples_matrix[, "beta4"], main = "ACF for beta4")

par(mfrow = c(2,2))
acf(samples_matrix[, "beta5"], main = "ACF for beta5")
acf(samples_matrix[, "beta6"], main = "ACF for beta6")
acf(samples_matrix[, "beta7"], main = "ACF for beta7")
```

Models

Poisson Model

```
model {
  for (i in 1:N) {
    TotalMedals[i] ~ dpois(lambda[i])
    log(lambda[i]) <- beta0 + beta1 * GoldMedals[i] + beta2 * Silver[i] +
    beta3 * Bronze[i] + beta4 * BordaPoints[i] + beta5 * Income[i] +
    beta6 * PopnSize[i] + beta7 * GDP[i] + beta8 * IncomeSQ[i] +
    beta9 * PopnSQ[i] + beta10 * IncomeCubed[i] + beta11 * PopnCubed[i] +
    beta12 * GDPSQ[i] + beta13 * GDPCubed[i] + beta14 * LnIncome[i] +
    beta15 * LnPopnSize[i] + beta16 * LnGDP[i]
  }

  beta0 ~ dnorm(0, 0.001)
  beta1 ~ dnorm(0, 0.001)
  beta2 ~ dnorm(0, 0.001)
  beta3 ~ dnorm(0, 0.001)
  beta4 ~ dnorm(0, 0.001)
  beta5 ~ dnorm(0, 0.001)
  beta6 ~ dnorm(0, 0.001)
  beta7 ~ dnorm(0, 0.001)
  beta8 ~ dnorm(0, 0.001)
  beta9 ~ dnorm(0, 0.001)
  beta10 ~ dnorm(0, 0.001)
  beta11 ~ dnorm(0, 0.001)
  beta12 ~ dnorm(0, 0.001)
  beta13 ~ dnorm(0, 0.001)
  beta14 ~ dnorm(0, 0.001)
  beta15 ~ dnorm(0, 0.001)
  beta16 ~ dnorm(0, 0.001)

}
```

Negative Binomial Model

```
model {
  for (i in 1:N) {
    TotalMedals[i] ~ dnegbin(p[i], r)
    p[i] <- r / (r + lambda[i])
    log(lambda[i])<- beta0 + beta1 * GoldMedals[i] + beta2 * Silver[i] +
    beta3 * Bronze[i] + beta4 * BordaPoints[i] + beta5 * Income[i] +
    beta6 * PopnSize[i] + beta7 * GDP[i] + beta8 * IncomeSQ[i] +
    beta9 * PopnSQ[i] + beta10 * IncomeCubed[i] + beta11 * PopnCubed[i] +
    beta12 * GDPSQ[i] + beta13 * GDPCubed[i] + beta14 * LnIncome[i] +
    beta15 * LnPopnSize[i] + beta16 * LnGDP[i]
```

```
  }

  beta0 ~ dnorm(0, 0.001)
  beta1 ~ dnorm(0, 0.001)
  beta2 ~ dnorm(0, 0.001)
  beta3 ~ dnorm(0, 0.001)
  beta4 ~ dnorm(0, 0.001)
  beta5 ~ dnorm(0, 0.001)
  beta6 ~ dnorm(0, 0.001)
  beta7 ~ dnorm(0, 0.001)
  beta8 ~ dnorm(0, 0.001)
  beta9 ~ dnorm(0, 0.001)
  beta10 ~ dnorm(0, 0.001)
  beta11 ~ dnorm(0, 0.001)
  beta12 ~ dnorm(0, 0.001)
  beta13 ~ dnorm(0, 0.001)
  beta14 ~ dnorm(0, 0.001)
  beta15 ~ dnorm(0, 0.001)
  beta16 ~ dnorm(0, 0.001)

  r ~ dgamma(0.01, 0.01)
}
```

Negative Binomial Model (2)

```
model {
  for (i in 1:N) {
    TotalMedals[i] ~ dnegbin(p[i], r)
    p[i] <- r / (r + lambda[i])
    log(lambda[i])<- beta0 + beta1 * Bronze[i] + beta2 * BordaPoints[i] +
    beta3 * Income[i] + beta4 * PopnSize[i] + beta5 * GDP[i] +
    beta6* PopnSQ[i] + beta7 * PopnCubed[i] + beta8 * GDPSQ[i] +
    beta9 * GDPCubed[i] + beta10 * LnIncome[i] + beta11 * LnPopnSize[i] +
    beta12 * LnGDP[i]
  }

  beta0 ~ dnorm(0, 0.001)
  beta1 ~ dnorm(0, 0.001)
  beta2 ~ dnorm(0, 0.001)
  beta3 ~ dnorm(0, 0.001)
  beta4 ~ dnorm(0, 0.001)
  beta5 ~ dnorm(0, 0.001)
  beta6 ~ dnorm(0, 0.001)
  beta7 ~ dnorm(0, 0.001)
  beta8 ~ dnorm(0, 0.001)
  beta9 ~ dnorm(0, 0.001)
  beta10 ~ dnorm(0, 0.001)
  beta11 ~ dnorm(0, 0.001)
  beta12 ~ dnorm(0, 0.001)

  r ~ dgamma(0.01, 0.01)
}‘
```

Negative Binomial Model (3)

```
model {
  for (i in 1:N) {
    TotalMedals[i] ~ dnegbin(p[i], r)
    p[i] <- r / (r + lambda[i])
    log(lambda[i])<- beta0 + beta1 * Bronze[i] + beta2 * BordaPoints[i] +
    beta3 * Income[i] + beta4 * PopnSize[i] + beta5 * GDP[i] +
    beta6 * GDPSQ[i] + beta7 * GDPCubed[i] + beta8 * LnIncome[i] +
    beta9 * LnPopnSize[i] + beta10 * LnGDP[i]
  }

  beta0 ~ dnorm(0, 0.001)
  beta1 ~ dnorm(0, 0.001)
  beta2 ~ dnorm(0, 0.001)
  beta3 ~ dnorm(0, 0.001)
  beta4 ~ dnorm(0, 0.001)
  beta5 ~ dnorm(0, 0.001)
  beta6 ~ dnorm(0, 0.001)
  beta7 ~ dnorm(0, 0.001)
  beta8 ~ dnorm(0, 0.001)
  beta9 ~ dnorm(0, 0.001)
  beta10 ~ dnorm(0, 0.001)

  r ~ dgamma(0.01, 0.01)
}
```

Negative Binomial Model (4)

```
model {
  for (i in 1:N) {
    TotalMedals[i] ~ dnegbin(p[i], r)
    p[i] <- r / (r + lambda[i])
    log(lambda[i])<- beta0 + beta1 * Bronze[i] + beta2 * BordaPoints[i] +
    beta3 * Income[i] + beta4 * GDP[i] + beta5 * GDPSQ[i] +
    beta6 * LnIncome[i] + beta7 * LnPopnSize[i]

  }

  beta0 ~ dnorm(0, 0.001)
  beta1 ~ dnorm(0, 0.001)
  beta2 ~ dnorm(0, 0.001)
  beta3 ~ dnorm(0, 0.001)
  beta4 ~ dnorm(0, 0.001)
  beta5 ~ dnorm(0, 0.001)
  beta6 ~ dnorm(0, 0.001)
  beta7 ~ dnorm(0, 0.001)

  r ~ dgamma(0.01, 0.01)
}
```

Appendix

```
library(R2OpenBUGS)
library(coda)
library(BAS)
setwd("C:/Users/Armando/Downloads/Bayesian Model/")
data <- read.csv("13_Olympics.csv", sep = ",", stringsAsFactors = FALSE)
data <- data[2:18]
```

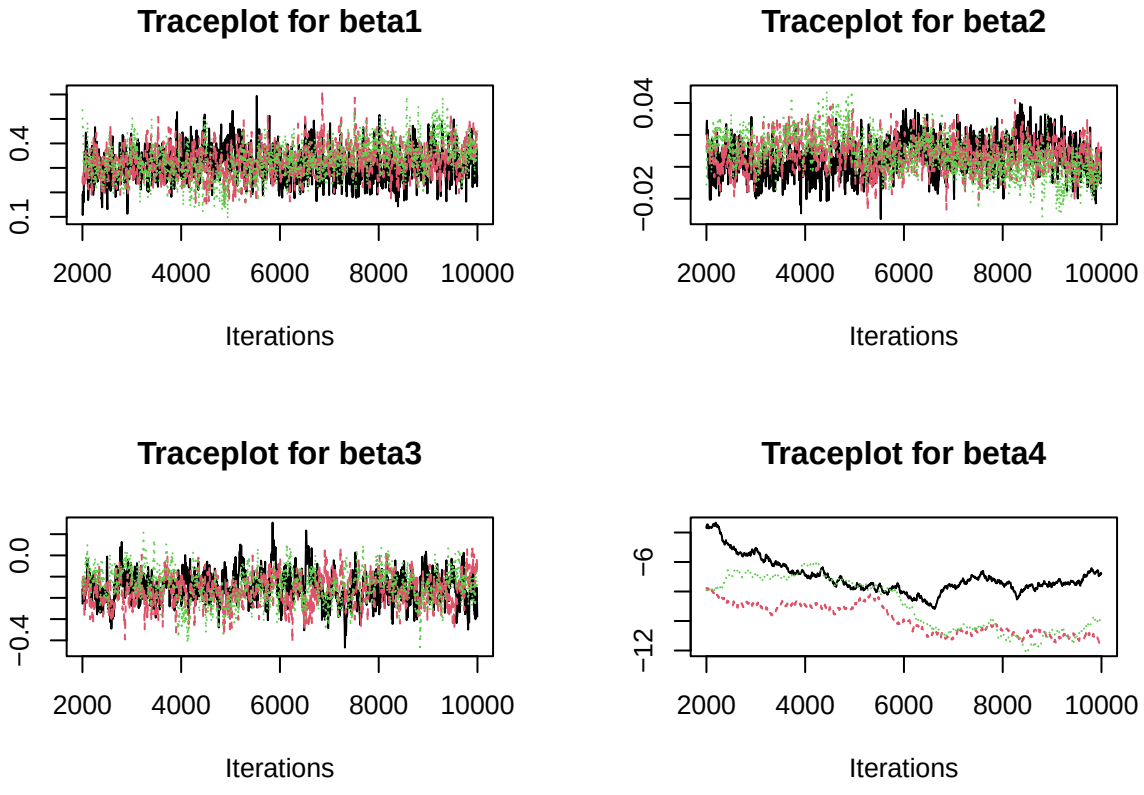
```
data <- data[, names(data) != "GoldMedals"]
data <- data[, names(data) != "Silver"]
data <- data[, names(data) != "IncomeCubed"]
data <- data[, names(data) != "Income.SQ"]
data <- data[, names(data) != "PopnCubed"]
data <- data[, names(data) != "PopnSQ"]
data <- data[, names(data) != "Ln.GDP."]
data <- data[, names(data) != "PopnSize"]
data <- data[, names(data) != "GDPCubed"]

data_list <- list(TotalMedals = data$TotalMedals, Bronze = data$Bronze,
                  BordaPoints = data$BordaPoints, Income = data$Income,
                  GDP = data$GDP, GDPSQ = data$GDPSQ, LnIncome = data$Ln.Income.,
                  LnPopnSize = data$Ln.PopnSize., N = nrow(data))

inits <- function() {
  list( beta0 = 0, beta1 = 0, beta2 = 0, beta3 = 0, beta4 = 0, beta5 = 0,
        beta6 = 0, beta7 = 0, r = 1)
}

params <- c("beta0", "beta1", "beta2", "beta3", "beta4", "beta5", "beta6",
            "beta7", "r")

bugs_4 <- bugs(data_list, inits, parameters = params, model.file = "BinNeg_4.txt",
               n.chains = 3, n.iter = 10000, n.burnin = 2000, n.thin = 5)
print(bugs_4)
```

```
## Inference for Bugs model at "BinNeg_4.txt",
## Current: 3 chains, each with 10000 iterations (first 2000 discarded), n.thin = 5
## Cumulative: n.sims = 24000 iterations saved
##           mean  sd  2.5%   25%   50%   75% 97.5% Rhat n.eff
## beta0      3.1 0.4   2.2   2.8   3.1   3.4   3.9  1.0    87
## beta1      0.4 0.1   0.2   0.3   0.4   0.4   0.5  1.0  2800
## beta2      0.0 0.0   0.0   0.0   0.0   0.0   0.0  1.0  1200
## beta3     -0.1 0.1  -0.3  -0.2  -0.1  -0.1   0.0  1.0   300
## beta4    -12.0 2.7 -16.5 -14.0 -12.5 -10.4  -6.2  1.1   760
## beta5      3.6 1.4   0.3   2.7   3.9   4.7   5.8  1.1   320
## beta6      0.8 0.1   0.5   0.7   0.8   0.9   1.0  1.0   150
```

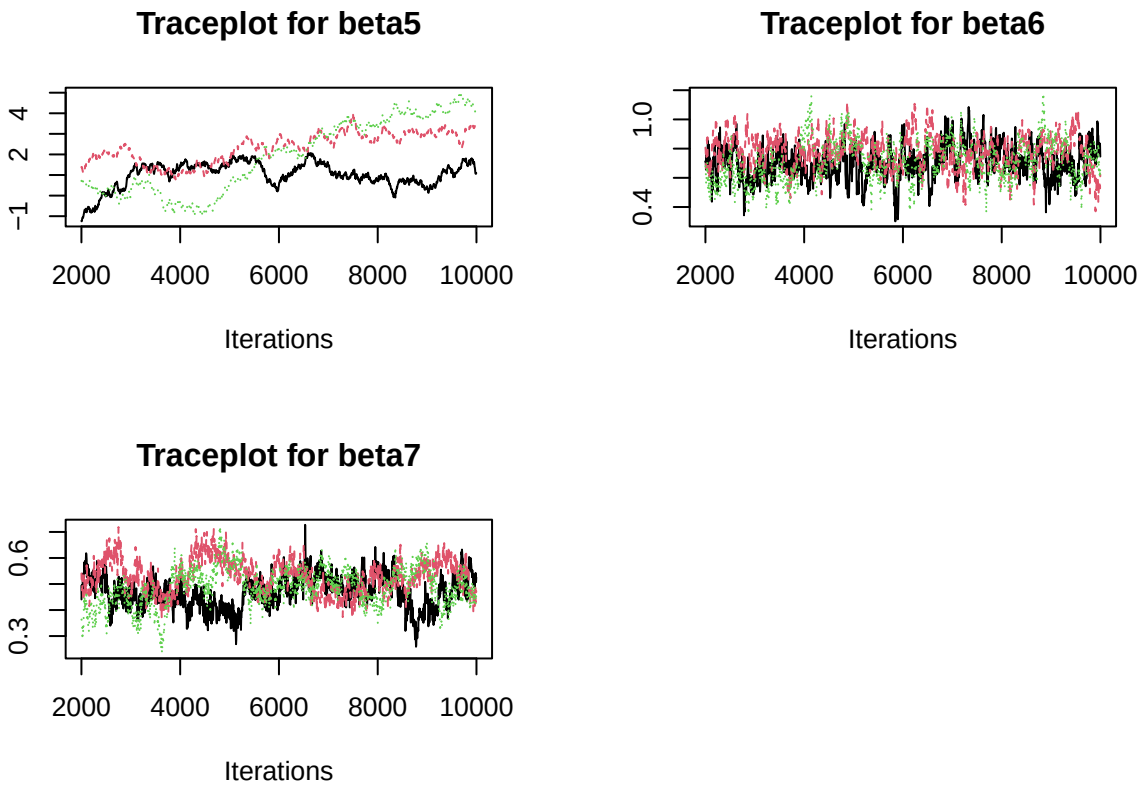
```
## beta7      0.5 0.1   0.4   0.5   0.5   0.6   0.7  1.0   160
## r          1.2 0.3   0.8   1.0   1.2   1.4   1.8  1.0   710
## deviance 606.3 4.9 599.2 602.7 605.4 608.9 618.3  1.0    90
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = Dbar-Dhat)
## pD = 8.8 and DIC = 615.1
## DIC is an estimate of expected predictive error (lower deviance is better).
```

Diagnostic Plots

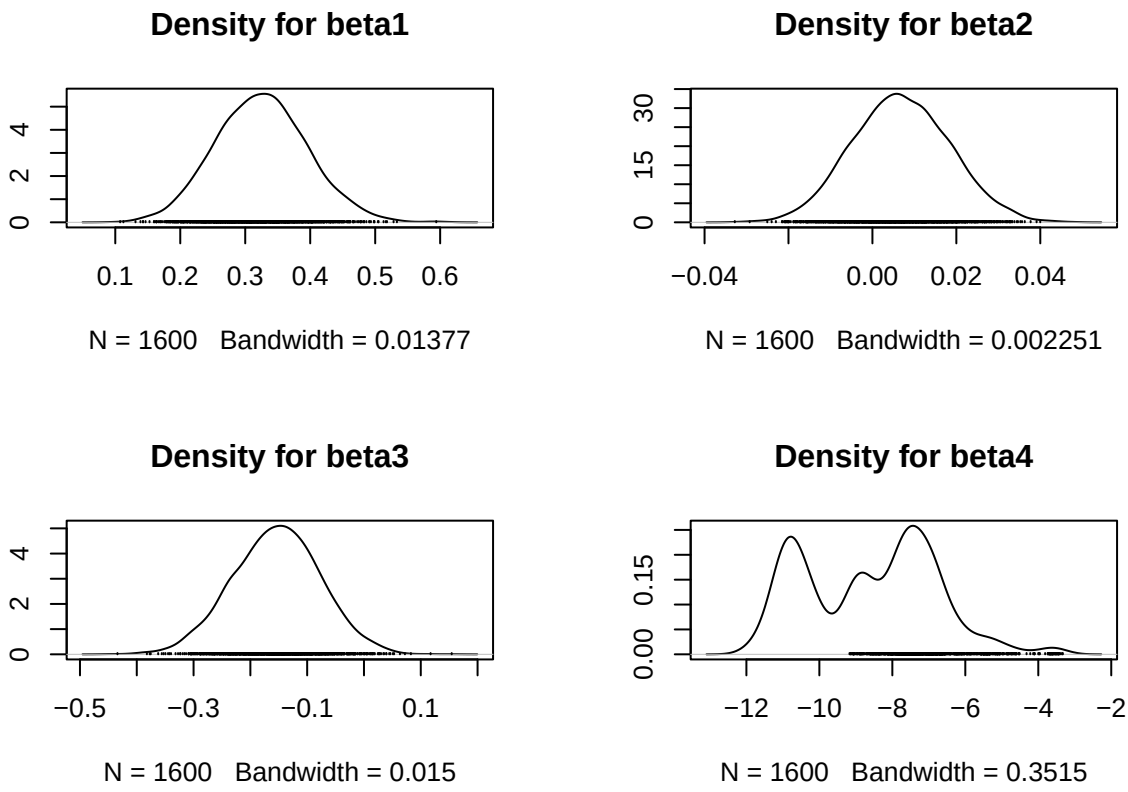
```
samples <- as.mcmc.list(bugs_4)
samples_matrix <- as.matrix(samples)

par(mfrow = c(2,2))
traceplot(samples[, "beta1"], main = "Traceplot for beta1")
traceplot(samples[, "beta2"], main = "Traceplot for beta2")
traceplot(samples[, "beta3"], main = "Traceplot for beta3")
traceplot(samples[, "beta4"], main = "Traceplot for beta4")
```



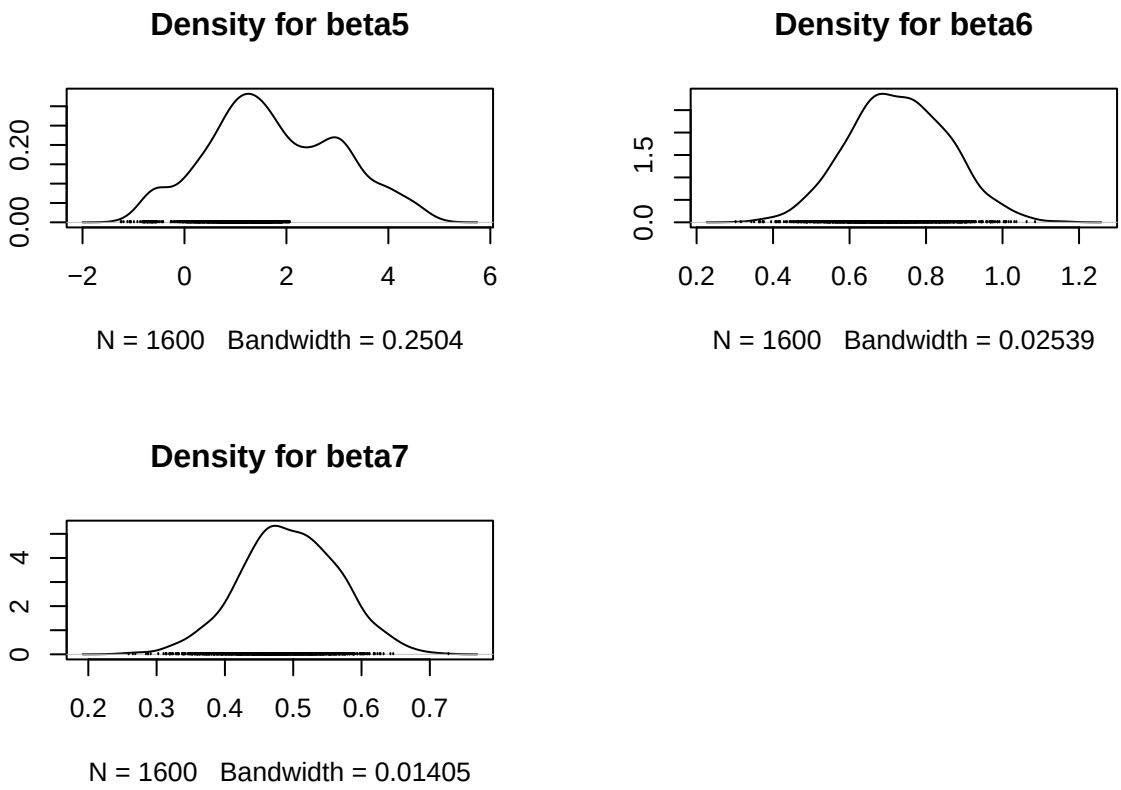
```
par(mfrow=c(2,2))
traceplot(samples[, "beta5"], main = "Traceplot for beta5")
traceplot(samples[, "beta6"], main = "Traceplot for beta6")
traceplot(samples[, "beta7"], main = "Traceplot for beta7")

par(mfrow = c(2,2))
```



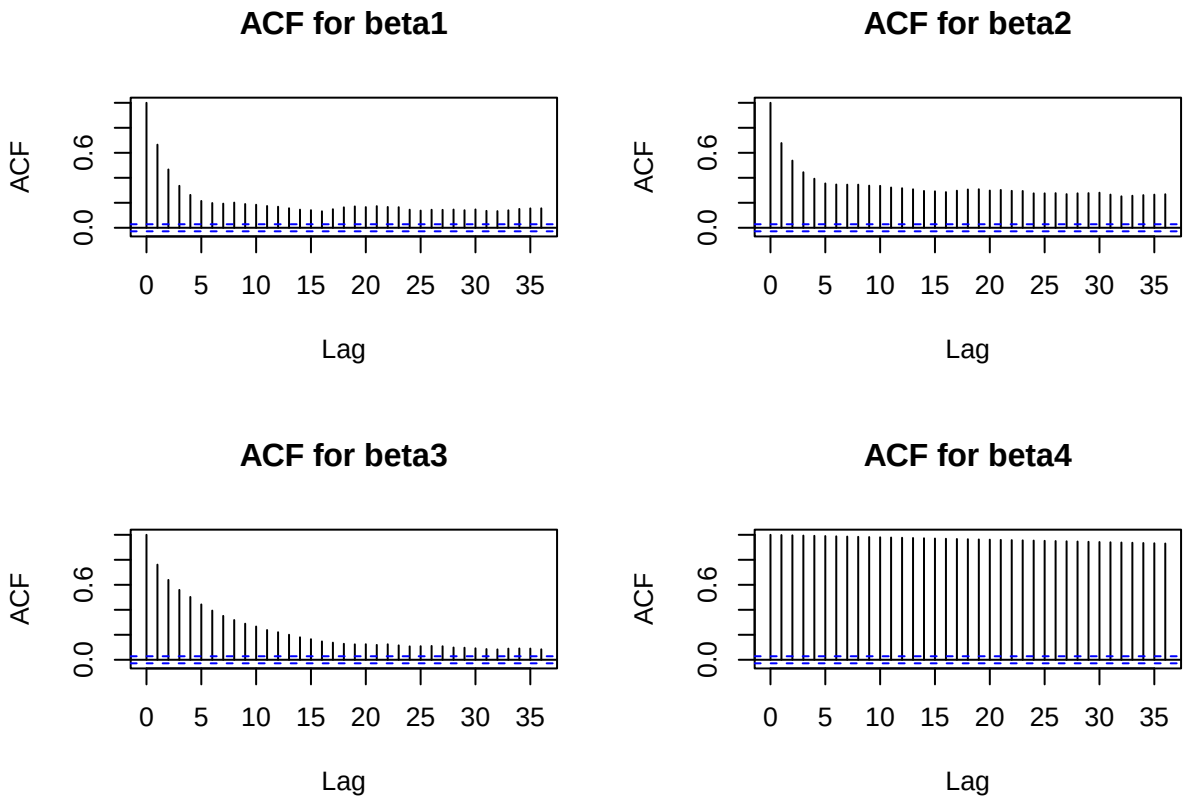

```
densplot(samples[, "beta1"], main = "Density for beta1")
densplot(samples[, "beta2"], main = "Density for beta2")
densplot(samples[, "beta3"], main = "Density for beta3")
densplot(samples[, "beta4"], main = "Density for beta4")
```



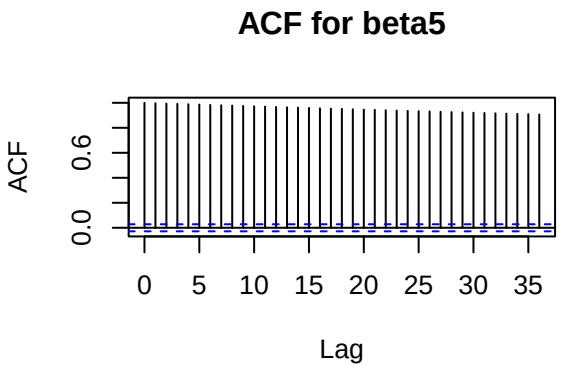

```
par(mfrow = c(2,2))
densplot(samples[, "beta5"], main = "Density for beta5")
densplot(samples[, "beta6"], main = "Density for beta6")
densplot(samples[, "beta7"], main = "Density for beta7")

par(mfrow = c(2,2))
```

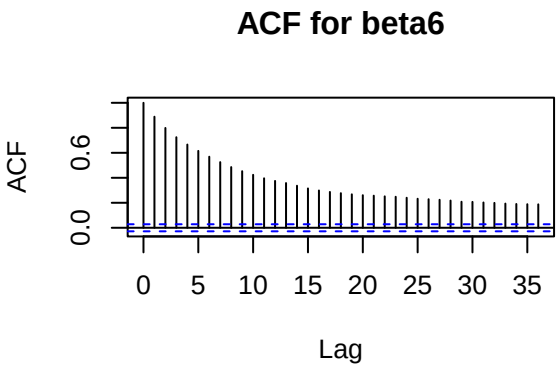


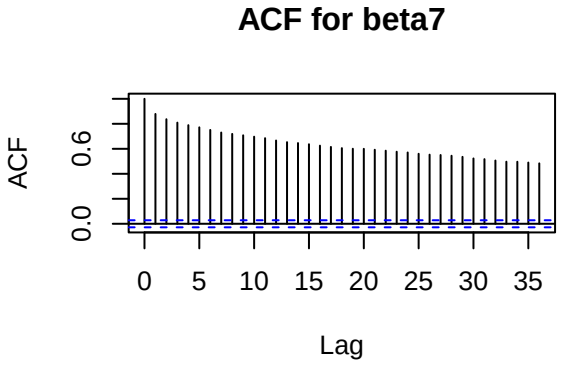

```r
acf(samples_matrix[, "beta1"], main = "ACF for beta1")
acf(samples_matrix[, "beta2"], main = "ACF for beta2")
acf(samples_matrix[, "beta3"], main = "ACF for beta3")
acf(samples_matrix[, "beta4"], main = "ACF for beta4")
```




```
par(mfrow = c(2,2))
acf(samples_matrix[, "beta5"], main = "ACF for beta5")
acf(samples_matrix[, "beta6"], main = "ACF for beta6")
acf(samples_matrix[, "beta7"], main = "ACF for beta7")
```


ACF for beta5
ACF
0.0
0.6
0
5
10
15
20
25
30
35
Lag


ACF for beta6
ACF
0.0
0.6
0
5
10
15
20
25
30
35
Lag


ACF for beta7
ACF
0.0
0.6
0
5
10
15
20
25
30
35
Lag