# IoT or Traditional agriculture: does it really matter?

SMDS-2023-2024

Sofia Noemi Crobeddu (mtr. 2130389)

M.Sc. in Data Science

# 1. Introduction and context

In the last years, the application of advanced technologies in agriculture has seen a rapid growth, driven by the need to enhance the productivity and the sustainability in farming practices. The integration of the Internet of Things (IoT) devices in agriculture (i.e. sensors and network devices) is groundbreaking, providing farmers with real-time data on various environmental and crop-related factors. This data-driven approach facilitates more informed decision-making and optimization of agricultural processes.

This project is based on the "Advanced IoT Agriculture 2024" dataset developed from a real IoT system deployed in agricultural environments. This system allows real-time monitoring of environmental conditions within a smart greenhouse, modifying the productivity of traditional settings. The observations of the dataset describe the growth of different varieties of plants within the two types of environment. The following study analyses the effectiveness of IoT environment compared to traditional one, through the application of Bayesian models (using MCMC simulation).

## 1.1. Illustration of the dataset

The dataset mentioned before was taken from the Kaggle official website ((Abdullah 2024)). It is composed by 30000 observations and 14 variables. Data were collected by the student Mohammed Ismail Lifta of Tikrit University (Iraq), in his MSc research thesis during the academic year 2023-2024, with the collaboration of the Agricultural Laboratory and the supervision of Professor (Assistant) Wisam Dawood Abdullah, administrator of Cisco Networking Academy.

Data describe the cultivation of three radish varieties, Dutch, Syrian, and Italian, in the two greenhouse (Smart and Traditional), outlined in the introductory paragraph. They were registered in two slots between November and December 2023 ((Lafta and Abdullah 2024)).

The variables are the following ones: - **Random**: categorical identifier for each record, with modalities R1, R2 and R3 representing different random sample. - **Average Chlorophyll in Plant (ACHP)**: average value of chlorophyll content in plants. It is an important indicator of photosynthetic capacity and describes the plant's health and the efficiency in converting light energy into chemical energy. - **Plant Height Growth Rate (PHR)**: a measure of the plant's vertical growth rate over time. It is an important indicator to comprehend the height development. - **Average wet weight of vegetative growth (AWWGV)**: average wet weight of the vegetative parts of the plant. It is useful for assessing the water content and overall biomass of vegetative growth. - **Average plant leaf area (ALAP)**: Measure of the average leaf area, important for estimating the area available for photosynthesis. - **Average number of leaves per plant (ANPL)**: average number of leaves per plant, which can influence the photosynthetic capacity and general health of the plant. - **Average root diameter (ARD)**: average root diameter. It is an indicator of the plant's ability to absorb water and nutrients from the soil. - **Average dry weight of roots (ADWR)**: average dry weight of the roots, indicative of root biomass after removing water content. It is useful for assessing structural and storage capacity. - **Percentage of dry matter in vegetative growth (PDMVG)**: percentage of dry matter present in the vegetative parts of the plant, reflecting the proportion of biomass that is not composed of water, that is essential for analyzing the structure and nutritional status of the plant. - **Average root length (ARL)**: average root length, which can influence the plant's ability to explore the soil and absorb nutrients and water. - **Average wet weight of roots (AWWR)**: average wet weight of the roots, including water content, an indicator of the overall biomass and water retention capacity of roots. - **Average dry weight of vegetative plants (ADWV)**: average dry weight of the vegetative parts of the plant. It represents the structural biomass of the plant excluding water content. - **Percentage of dry matter in root growth (PDMRG)**: percentage of dry matter in plant's roots. It is important to assess root health and functionality. - **Class**: categorical variable indicating the class to which each plant record belongs. There are 6 classes in total (SA, SB, SC, TA, TB, TC), and specifically they represents the three varieties of plants observed (A, B, C), in each type of greenhouse (Smart and Traditional).

## 1.2. Preliminary aspects

Before starting the exploratory data analysis, it is important to specify that for this project the variable Random is not taken into account since it is an identifier variable.

Moreover, for simplicity of investigations, for this work we consider only the first two varieties A (Dutch) and B (Syrian) of the plants. This decision was based, first of all, on the results reached by Mohamed Ismail Lafta and Wisam Dawood Abdullah who found strong

differences between these two varieties in particular ((Lafta and Abdullah 2024)), hence this comparison could be more interesting. Then the choice was also driven by the fact that the variable Class exposes 5000 observations in each of the 6 categories, and then, in the target variable, there is not a variety “more important” than another one.

# 1.3. Exploratory data analysis

Let’s start the analysis. First, we take a look at the general data before any change:

```
'data.frame':   30000 obs. of  14 variables:
 $ Random                                           : chr  "R1" "R1" "R2" "R1" ...
 $ Average of chlorophyll in the plant (ACHP)       : num  34.5 34.5 33.1 34.5 36.3 ...
 $ Plant height  rate (PHR)                         : num  54.6 54.6 67.1 54.6 45.6 ...
 $ Average wet weight of the growth vegetative (AWWGV)   : num  1.15 1.15 1.1 1.14 1.36 ...
 $ Average leaf area of the plant (ALAP)            : num  1284 1284 1009 1284 981 ...
 $ Average number of plant leaves (ANPL)            : num  5 5.02 5.01 4.99 4 ...
 $ Average root diameter (ARD)                      : num  16.3 16.3 16 16.3 17 ...
 $ Average dry weight of the root (ADWR)            : num  1.707 1.701 1.185 1.716 0.777 ...
 $ Percentage of dry matter for vegetative growth (PDMVG): num  18.4 18.4 19.4 18.4 31.4 ...
 $ Average root length (ARL)                        : num  19.7 19.8 20.8 19.7 17.3 ...
 $ Average wet weight of the root (AWWR)            : num  2.95 2.94 2.86 2.95 2.77 ...
 $ Average dry weight of vegetative plants (ADWV)   : num  0.209 0.216 0.2 0.223 0.424 ...
 $ Percentage of dry matter for root growth (PDMRG)      : num  57.6 57.6 41.3 57.6 27.9 ...
 $ Class                                            : chr  "SA" "SA" "SA" "SA" ...
```

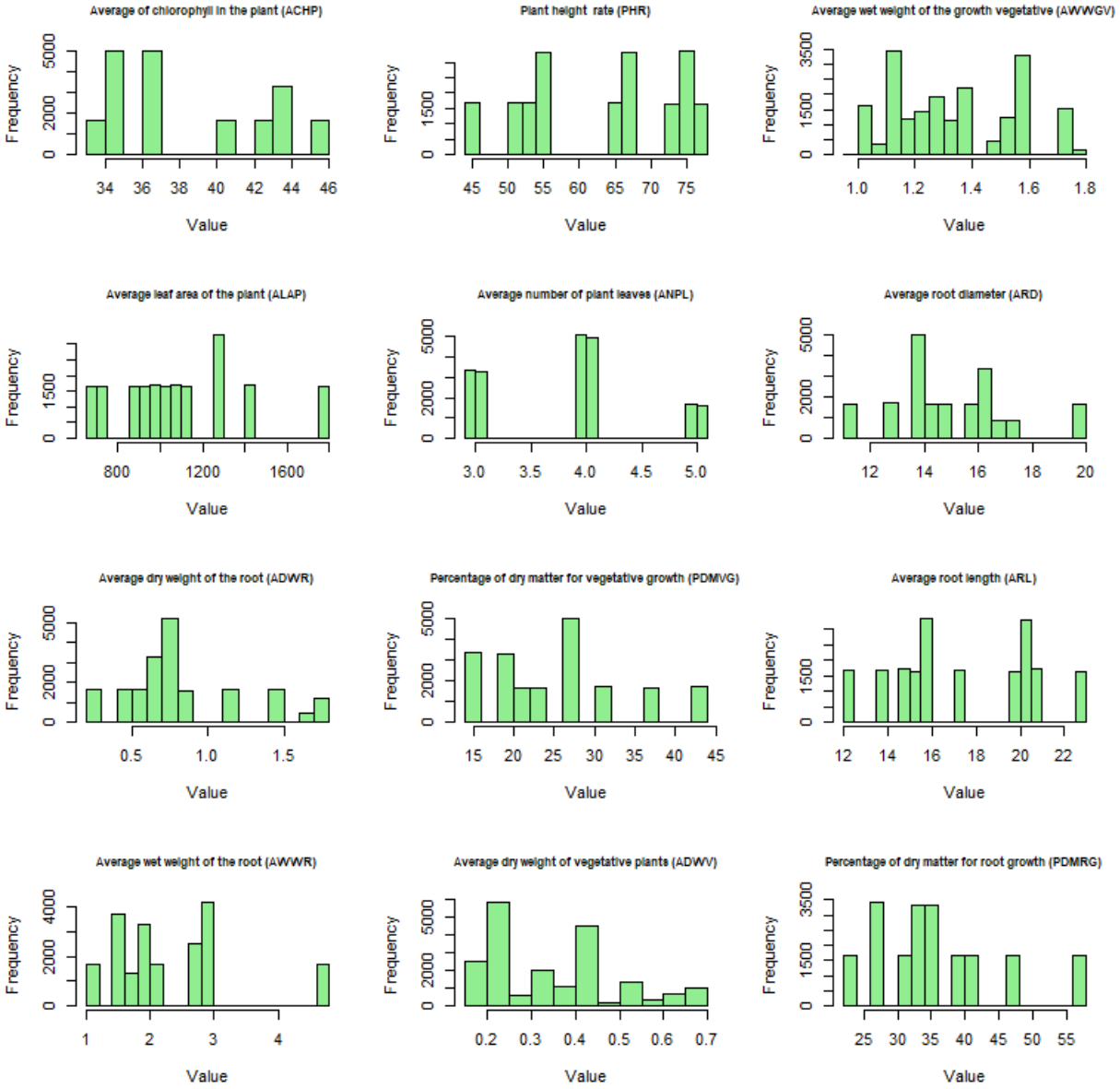
After this, we need to check if there are NA values:

```
[1] 0
```

As it is shown, there are not NA values in the dataset. Going on, we can now remove the variable Random and reduce the dataset with plants of variety A and B:

```
'data.frame':   20000 obs. of  13 variables:
 $ Average of chlorophyll in the plant (ACHP)       : num  34.5 34.5 33.1 34.5 36.3 ...
 $ Plant height  rate (PHR)                         : num  54.6 54.6 67.1 54.6 45.6 ...
 $ Average wet weight of the growth vegetative (AWWGV)   : num  1.15 1.15 1.1 1.14 1.36 ...
 $ Average leaf area of the plant (ALAP)            : num  1284 1284 1009 1284 981 ...
 $ Average number of plant leaves (ANPL)            : num  5 5.02 5.01 4.99 4 ...
 $ Average root diameter (ARD)                      : num  16.3 16.3 16 16.3 17 ...
 $ Average dry weight of the root (ADWR)            : num  1.707 1.701 1.185 1.716 0.777 ...
 $ Percentage of dry matter for vegetative growth (PDMVG): num  18.4 18.4 19.4 18.4 31.4 ...
 $ Average root length (ARL)                        : num  19.7 19.8 20.8 19.7 17.3 ...
 $ Average wet weight of the root (AWWR)            : num  2.95 2.94 2.86 2.95 2.77 ...
 $ Average dry weight of vegetative plants (ADWV)   : num  0.209 0.216 0.2 0.223 0.424 ...
 $ Percentage of dry matter for root growth (PDMRG)      : num  57.6 57.6 41.3 57.6 27.9 ...
 $ Class                                            : chr  "SA" "SA" "SA" "SA" ...
```

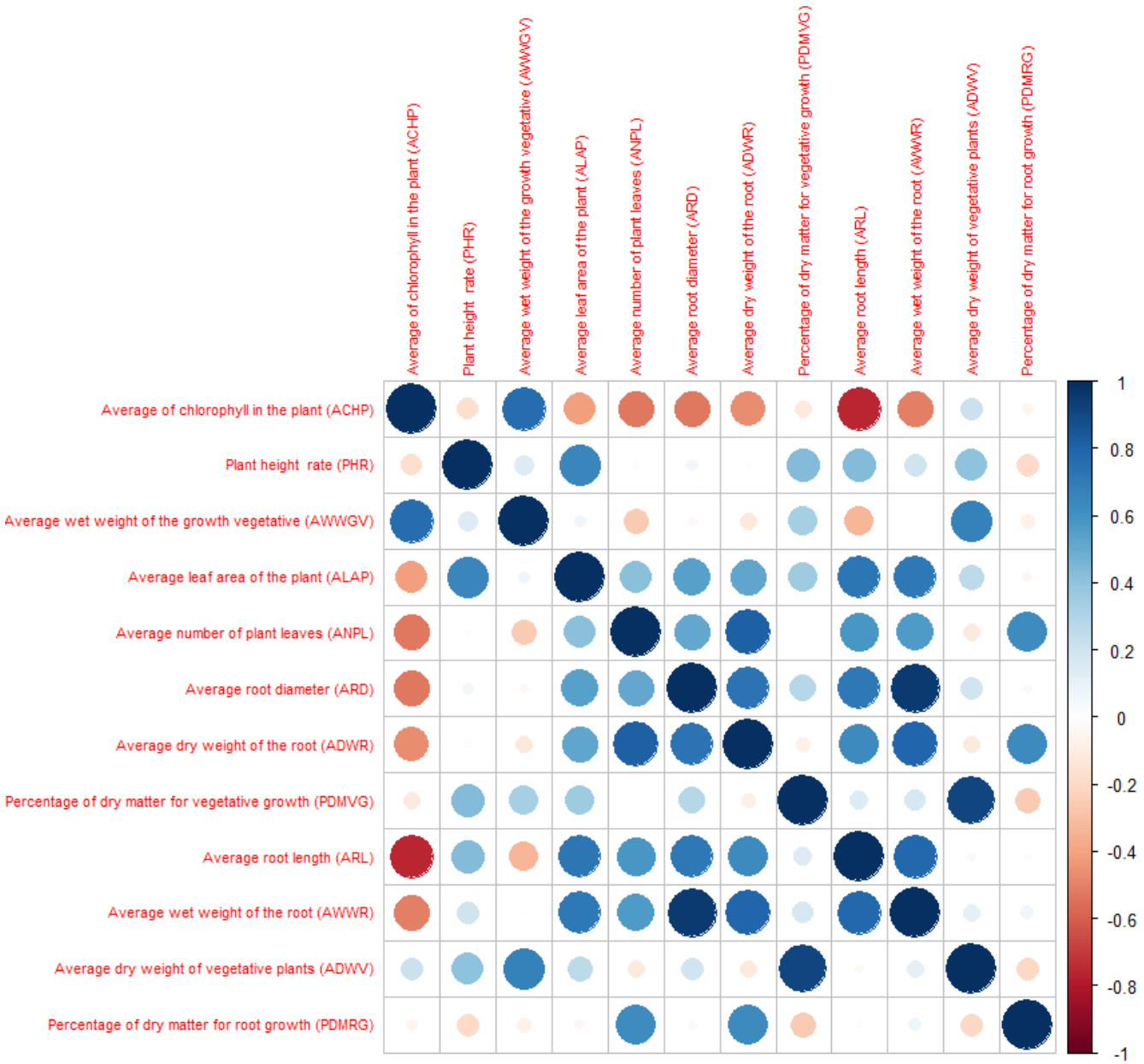
As expected, the dataset now has 20000 observation. We can now go on with the exploratory data analysis, taking a look at the distributions of the variables, that are our predictors in the models we will apply after. The histograms of frequency of the predictors are shown in Figure 1.




Distributions of predictors.

As we can see from the graphs, the 12 predictors do not seem to follow a Normal distribution. The majority of the variables seem to have skewness and multimodality. In particular, AWWGV, ADWR, AWWR and ADWV seem to be skewed. Instead, PHR, ALAP, and ARD presents more than one modality, suggesting that maybe there are different groups within the data, and this is confirm by the fact that plants can be grouped by the type of greenhouse or by the variety. Based on this evaluation, we need to transform data properly before applying statistical models.

After this analysis, we can have a look at the correlations between the variables, to better understand their relationships. From the correlation matrix (Figure 2) we can notice that, in general, we have more positive correlations (signed in blue tonalities). In particular we can see that there is a very strong positive relation of 0.95 between Average wet weight of the root (AWWR) and Average root diameter (ARD). This suggests that plants with thicker roots also tend to have heavier, water-rich roots. It is logical since thicker roots typically have more tissue capable of storing water and nutrients, contributing to a higher wet weight. Similar it is for the relation between Average dry weight of vegetative plants (ADWV) and Percentage of dry matter for vegetative growth (PDMVG) that have a correlation of 0.91. This makes sense since higher dry matter content means more structural biomass, leading to increased dry weight. Finally also a moderate, but still high, correlation of 0.81 between Average dry weight of the root (ADWR) and Average number of plant leaves (ANPL). Regarding the negative ones, we can say that there are less and weaker negative correlations (signed in red) for the majority, and from the graph we can notice that the variable Average of chlorophyll in the plant (ACHP) is the one that exposes this aspect the most. In particular, there is an high negative correlation of -0.75 between this variable and Average root length (ARL). These aspects regarding the variable ACHP suggests that plants probably tend to optimize for photosynthesis over root and leaf development. All the values of the correlation matrix are shown in the Appendix.




Corrplot.

Moreover, we have a look also at the variances of the predictors.

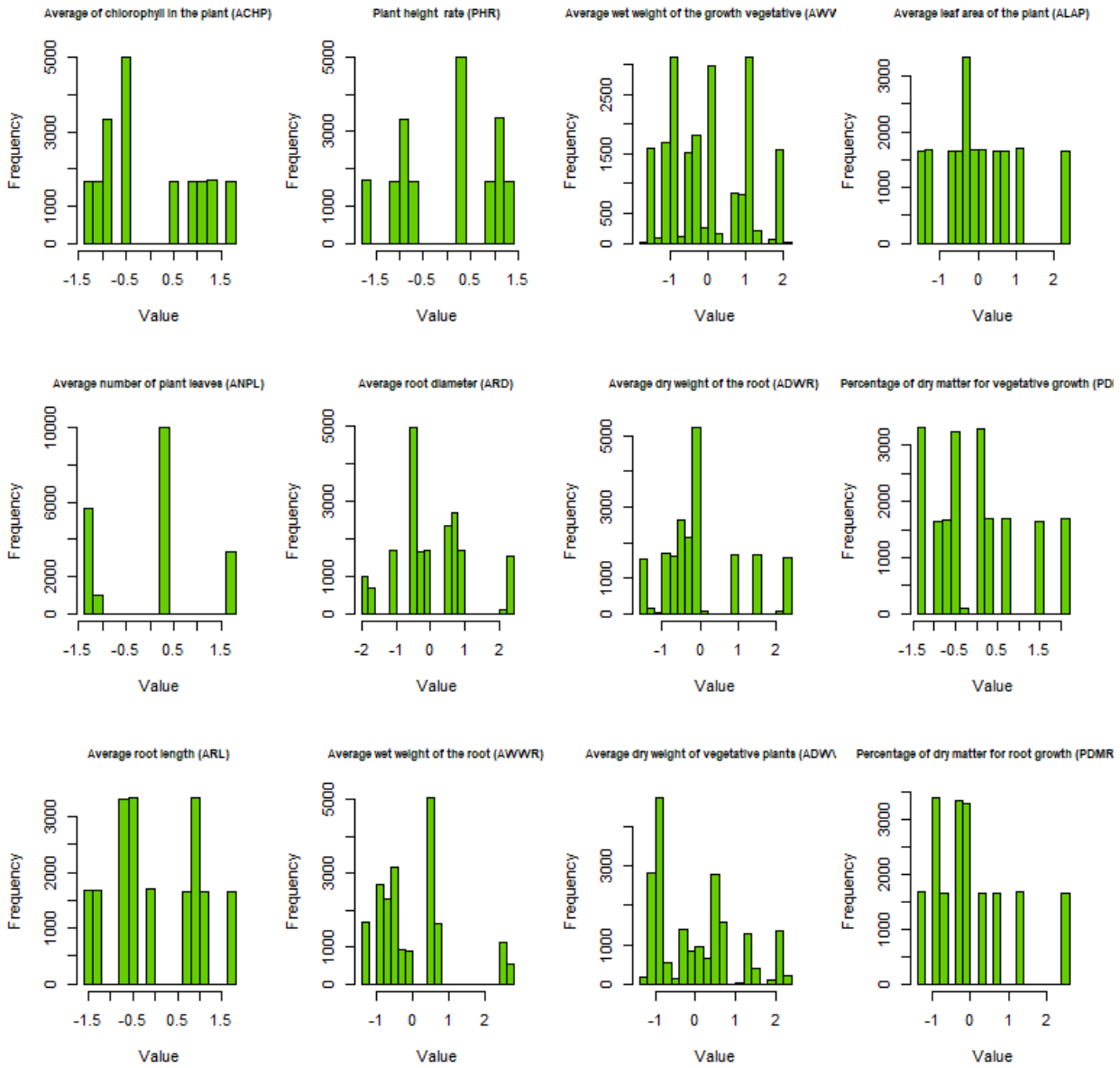
Variances of Predictors.

| Variable | Variance |
|---|---|
| Average of chlorophyll in the plant (ACHP) | 18.374 |
| Plant height rate (PHR) | 104.988 |
| Average wet weight of the growth vegetative (AWWGV) | 0.045 |
| Average leaf area of the plant (ALAP) | 84714.087 |
| Average number of plant leaves (ANPL) | 0.470 |
| Average root diameter (ARD) | 4.627 |
| Average dry weight of the root (ADWR) | 0.152 |
| Percentage of dry matter for vegetative growth (PDMVG) | 71.706 |
| Average root length (ARL) | 9.907 |
| Average wet weight of the root (AWWR) | 0.869 |

| Variable | Variance |
|---|---|
| Average dry weight of vegetative plants (ADWV) | 0.021 |
| Percentage of dry matter for root growth (PDMRG) | 79.284 |

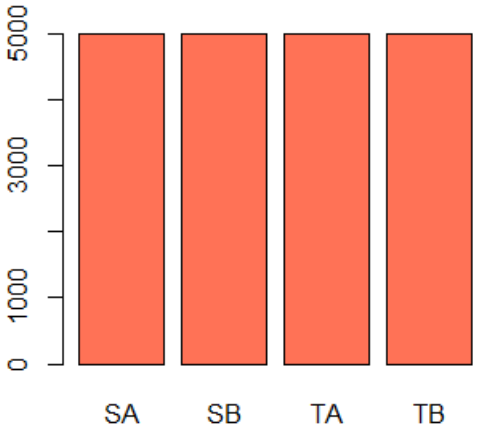
From the results shown in Table 1, we can see that the variable with the highest spread is Average leaf area of the plant (ALAP), instead the lowest one belongs to the variable Average dry weight of vegetative plants (ADWV). The extremely high variance of the predictor ALAP, highlights a wide range in leaf area among all the plants, suggesting significant variability in the plants' capacity for photosynthesis. This reflects also the slight variance of Average of chlorophyll in the plant (ACHP) that evokes the different efficiency at photosynthesis between plants. Probably these are consequences of the variety and the environment of each radish. In general variables with high variance are likely to play a significant role in distinguishing the classes, while predictors with low variance (in particular close to 0), influence less and suggest a relatively uniformity across plants for those specific aspects.

Before applying a statistical model, we need to operate the standardization to the data, to ensure that all variables contribute equally to the analysis. The standardized distributions are shown in Figure 3, and intuitively we can see that the standardization did not change the shapes of the distributions a lot.




Distributions of standardized predictors.

As the last step of the Exploratory data analysis, we need to take a look to the distribution of the variable Class that is the target variable in our models. The graph obtained is shown in Figure 4 and as it is evident, the categories are totally balanced: there are exactly 5000 observations for each of the 4 classes. This aspect is quite important for the selection of a sample to apply the models, and this will be done in the next sections.




Distribution of the variable Class.

# 2. Data adjustment and sample selection

## 2.1 . Preparation of data on the target variable

Before applying any model, the variable Class is transformed in factors. Moreover they are also created two new variables named Env and Variety. The first one is used to divide the plants in two groups based on the greenhouse: Smart (S) codified in 1 and Traditional (T) in 2. This variable will be useful for the application of the second Bayesian model in the next sections. The second variable instead, represents the varieties of the plants codified in 1 for A variety and 2 for B. Furthermore, not to lose the original category assigned, a new column called Class.chr where to save the class before the factorization, is added. The final dataset is shown below.

```
'data.frame':   20000 obs. of  16 variables:
 $ Average of chlorophyll in the plant (ACHP)            : num  34.5 34.5 33.1 34.5 36.3 ...
 $ Plant height  rate (PHR)                              : num  54.6 54.6 67.1 54.6 45.6 ...
 $ Average wet weight of the growth vegetative (AWWGV)   : num  1.15 1.15 1.1 1.14 1.36 ...
 $ Average leaf area of the plant (ALAP)                 : num  1284 1284 1009 1284 981 ...
 $ Average number of plant leaves (ANPL)                 : num  5 5.02 5.01 4.99 4 ...
 $ Average root diameter (ARD)                           : num  16.3 16.3 16 16.3 17 ...
 $ Average dry weight of the root (ADWR)                 : num  1.707 1.701 1.185 1.716 0.777 ...
 $ Percentage of dry matter for vegetative growth (PDMVG): num  18.4 18.4 19.4 18.4 31.4 ...
 $ Average root length (ARL)                             : num  19.7 19.8 20.8 19.7 17.3 ...
 $ Average wet weight of the root (AWWR)                 : num  2.95 2.94 2.86 2.95 2.77 ...
 $ Average dry weight of vegetative plants (ADWV)        : num  0.209 0.216 0.2 0.223 0.424 ...
 $ Percentage of dry matter for root growth (PDMRG)      : num  57.6 57.6 41.3 57.6 27.9 ...
 $ Class                                                 : num  1 1 1 1 1 1 1 1 1 1 ...
 $ Class.chr                                             : chr  "SA" "SA" "SA" "SA" ...
 $ Env                                                   : num  1 1 1 1 1 1 1 1 1 1 ...
 $ Variety                                               : num  1 1 1 1 1 1 1 1 1 1 ...
```

## 2.2. Multicollinearity analysis

In section 1.3. we discussed about the correlation matrix and we saw that there was a majority of positive correlations between variables which, in some cases, were also quite strong. This could cause multicollinearity and for this reason we need to verify it. To do this, we can use a quantitative method: the Variance Inflation Factor (VIF). It measures how much variance of a predictor is explained by the other ones of the model. If the VIF is high, usually more than 10, this means that the specific variable interested is largely collinear with the others, and this can cause problems in the model.

As shown in Table 2, the VIF is really high for many predictors. The common technique in this situation is to remove variables that cause collinearity and that usually have high values of VIF. In this specific case, after the elimination of the four variables Average wet weight of the growth vegetative (AWWGV), Average root diameter (ARD), Average dry weight of the root (ADWR) and Average dry weight of vegetative plants (ADWV), we reach acceptable levels of VIF, as shown in Table 3. It is interesting to notice that the waviable Average wet weight of the root (AWWR) that had the highest initial value of VIF, has now a very low value of 4.68, since the variables with high correlation with it were removed.

Variance Inflation Factor.

| Variable | VIF |
|---|---|
| Average of chlorophyll in the plant (ACHP) | 28.528827 |
| Plant height  rate (PHR) | 8.617544 |
| Average wet weight of the growth vegetative (AWWGV) | 74.076321 |
| Average leaf area of the plant (ALAP) | 15.776321 |
| Average number of plant leaves (ANPL) | 6.769150 |
| Average root diameter (ARD) | 129.235343 |
| Average dry weight of the root (ADWR) | 119.778746 |
| Percentage of dry matter for vegetative growth (PDMVG) | 99.602179 |
| Average root length (ARL) | 15.369452 |
| Average wet weight of the root (AWWR) | 187.243698 |
| Average dry weight of vegetative plants (ADWV) | 172.968515 |
| Percentage of dry matter for root growth (PDMRG) | 40.547827 |

Variance Inflation Factor after elimination of three variables.

| Variable | VIF |
|---|---|
| Average of chlorophyll in the plant (ACHP) | 3.038063 |
| Plant height  rate (PHR) | 3.342139 |
| Average leaf area of the plant (ALAP) | 4.707537 |
| Average number of plant leaves (ANPL) | 4.699092 |
| Percentage of dry matter for vegetative growth (PDMVG) | 1.566452 |
| Average root length (ARL) | 9.300580 |
| Average wet weight of the root (AWWR) | 4.675711 |
| Percentage of dry matter for root growth (PDMRG) | 2.956939 |

# 2.3. Selection of a sample from data

Applying Bayesian models to the entire dataset of 20000 observations can be computationally intensive and time expensive. For this reason, it is necessary to reduce the sample size to a manageable one. As specified in section 1.3., our target variable Class has 5000 observation for every category. To ensure the representativeness of the data, a suitable proposal is to take 150 observations for each class (600 observations in total). For this specific case, to avoid class imbalance and to reflect the original dataset, **stratified sampling** seems to be an appropriate technique, since it allows to maintain the proportionality of each class within the sample. To operate this sampling method, it was used the function *createDataPartition* from the library *caret*, to obtain an index to extract the sample from the initial dataset. It is important to specify that, as explained in the mentioned section, we need to use standardize predictors before applying models, even if we use a restricted sample.

```
#Selection of a sample through stratification
size <- 150 #Sample size for each class
#Index of stratification based on variable Class
idx <- createDataPartition(data$Class, p = size/5000,
                           #p = size/5000 specifies the ratio of
                           #observations to select from the total of
                           #each class (i.e. 3%)
                           list = FALSE, times = 1)
#Stratified sample
data.sampled <- data[idx, ]

#Check of sizes
check <- table(data.sampled$Class) #check of 150 for each one
str(data.sampled) #600x16
```

```
'data.frame':   600 obs. of  16 variables:
 $ Average of chlorophyll in the plant (ACHP)           : num  34.5 36.3 36.3 36.3 33.1 ...
 $ Plant height  rate (PHR)                             : num  54.6 45.6 45.6 45.6 67.1 ...
 $ Average wet weight of the growth vegetative (AWWGV)  : num  1.14 1.37 1.37 1.37 1.11 ...
 $ Average leaf area of the plant (ALAP)                : num  1284 981 981 981 1009 ...
 $ Average number of plant leaves (ANPL)                : num  4.99 4.01 4 4 5.01 ...
 $ Average root diameter (ARD)                          : num  16.3 17 17 17 16 ...
 $ Average dry weight of the root (ADWR)                : num  1.704 0.773 0.791 0.766 1.188 ...
 $ Percentage of dry matter for vegetative growth (PDMVG): num  18.4 31.4 31.4 31.4 19.4 ...
 $ Average root length (ARL)                            : num  19.7 17.3 17.3 17.3 20.8 ...
 $ Average wet weight of the root (AWWR)                : num  2.96 2.77 2.78 2.75 2.86 ...
 $ Average dry weight of vegetative plants (ADWV)       : num  0.226 0.417 0.401 0.434 0.201 ...
 $ Percentage of dry matter for root growth (PDMRG)     : num  57.6 27.9 27.9 27.9 41.3 ...
 $ Class                                                : num  1 1 1 1 1 1 1 1 1 1 ...
 $ Class.chr                                            : chr  "SA" "SA" "SA" "SA" ...
 $ Env                                                  : num  1 1 1 1 1 1 1 1 1 1 ...
 $ Variety                                              : num  1 1 1 1 1 1 1 1 1 1 ...
```

```
#Standardization of data sampled
sampled_standardized_predictors <- predict(preProcValues, data.sampled[, c(1:12)])
```

# 3. First Bayesian model: Multinomial logistic regression model

## 3.1. Formulation of the model

Having a problem of multiclass classification, the first and immediate model to apply is the Multinomial logistic regression one. This model is a natural extension of the Binomial logistic regression model and it is used when the target variable has more than 2 categories, which don't have a specific order. It can be applied if there is no multicollinearity between independent variables and the classes of the response variable are mutually exclusive and exhaustive. The target variable $Y_i = (Y_{i1}, \ldots, Y_{iK})$ with $Y_{ik}$ representing the frequency of the $k^{th}$ level, is assumed to have $K$ levels. The model can be expressed in the following way:

$$Y_i \sim multinomial(\pi_i, N_i)$$

and

$$\log(\frac{\pi_{ik}}{\pi_{i1}}) = \eta_{ik} = \beta_{0k} + \sum_{i=1}^{k} \beta_{jk} x_{ij}, \quad (3.1)$$

for $k = 2, \ldots, K$, where $pi_i = (\pi_{i1}, \pi_{i2}, \ldots, \pi_{iK})^T$ is the vector of the probabilities for every level of the variable $Y$ for individual $i$ (the probability that the plant $i$ belongs to category $k$), with $\pi_{i1} = 1 - \sum_{k=2}^{K} \pi_{ik}$.

In terms of response probabilities, solving the (3.1), we have that:

$$\pi_{i1} = \frac{1}{1 + \sum_{k=2}^{K} e^{\eta_{ik}}}$$

and also

$$\pi_{ik} = \frac{e^{\eta_{ik}}}{1 + \sum_{k=2}^{K} e^{\eta_{ik}}}$$

for $k = 2, \ldots, K$. The resulting summarizing term is:

$$\pi_{ik} = \frac{e^{\eta_{ik}}}{\sum_{k=1}^{K} e^{\eta_{ik}}}$$

with $\eta_{i1} = 0$ for $i = 1, 2, \ldots, n$.

# 3.2. Application of the model

In our case, since we removed four variables cause of multicollinearity, we can apply the **Multinomial logistic regression model**. This model can be implemented in JAGS in the following way:

```
\begin{verbatim}
model {
  #Likelihood
  for (i in 1:N) {
    y[i] ~ dcat(pi[i, 1:K])

    for (k in 1:K) {
      eta[i, k] <- beta0[k] +
                   beta[1, k] * ACHP[i] +
                   beta[2, k] * PHR[i] +
                   beta[3, k] * ALAP[i] +
                   beta[4, k] * ANPL[i] +
                   beta[5, k] * PDMVG[i] +
                   beta[6, k] * ARL[i] +
                   beta[7, k] * AWWR[i] +
                   beta[8, k] * PDMRG[i]

      expeta[i, k] <- exp(eta[i, k])
    }

    for (k in 1:K) {
      pi[i, k] <- expeta[i, k] / sum(expeta[i, ])
    }
  }

  #Priors
  for (k in 1:K){
    beta0[k] ~ dnorm(0, 0.1)

    #Predictors
    for (j in 1:8){
      beta[j, k] ~ dnorm(0, 0.1)
    }
  }
}
\end{verbatim}
```

and the commands in R are the following:

```
# First Bayesian model: Multinomial logistic
# model
y <- as.numeric(data.sampled$Class)
K <- length(unique(y))  #4, number of classes

# MCMC
S <- 10000
burn_in <- 2000

data.prepared <- list(y = y, ACHP = sampled_standardized_predictors$`Average of chlorophyll in the plant (ACHP)`,
    PHR = sampled_standardized_predictors$`Plant height  rate (PHR)`,
    ALAP = sampled_standardized_predictors$`Average leaf area of the plant (ALAP)`,
    ANPL = sampled_standardized_predictors$`Average number of plant leaves (ANPL)`,
    PDMVG = sampled_standardized_predictors$`Percentage of dry matter for vegetative growth (PDMVG)`,
    ARL = sampled_standardized_predictors$`Average root length (ARL)`,
    AWWR = sampled_standardized_predictors$`Average wet weight of the root (AWWR)`,
    PDMRG = sampled_standardized_predictors$`Percentage of dry matter for root growth (PDMRG)`,
    N = nrow(sampled_standardized_predictors), K = K)

params <- c("beta0", "beta")

inits <- list(inits1 = list(beta0 = rep(0, K), beta = matrix(0,
    8, K)), inits2 = list(beta0 = rep(1, K), beta = matrix(1,
    8, K)), inits3 = list(beta0 = rep(-0.5, K), beta = matrix(-0.5,
    8, K)))

start.time1 <- Sys.time()

jags.1 <- jags(data = data.prepared, inits = inits,
    parameters.to.save = params, model.file = "C:\\Users\\sofyc\\OneDrive\\Desktop\\SAPIENZA\\SDS II\\project\\mod
_agr.txt",
    n.chains = 3, n.iter = S, n.burnin = burn_in)
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 600
   Unobserved stochastic nodes: 36
   Total graph size: 33640

Initializing model
```

```
end.time1 <- Sys.time()
```

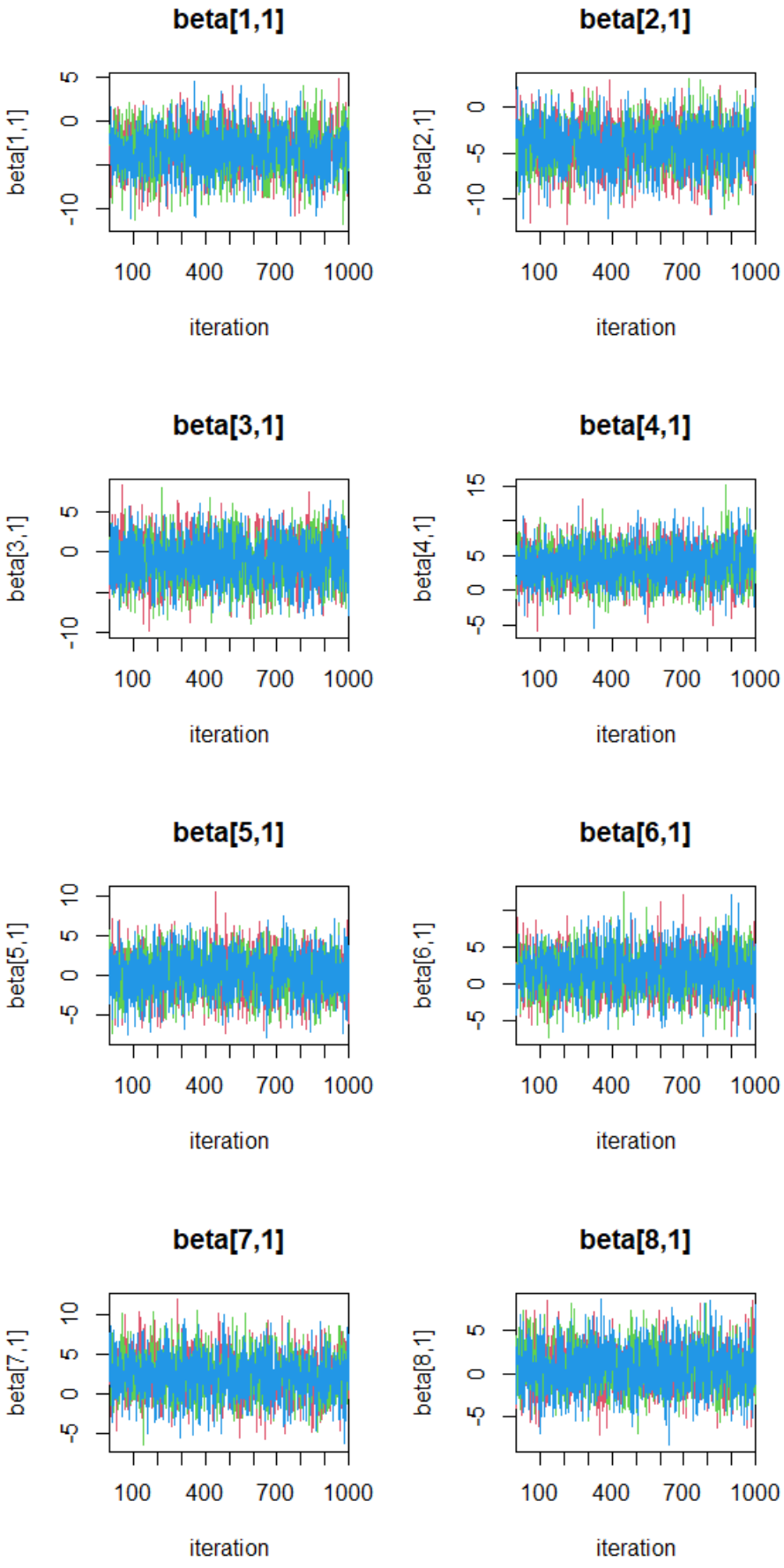
# 3.3. Convergence diagnostics and inferential findings

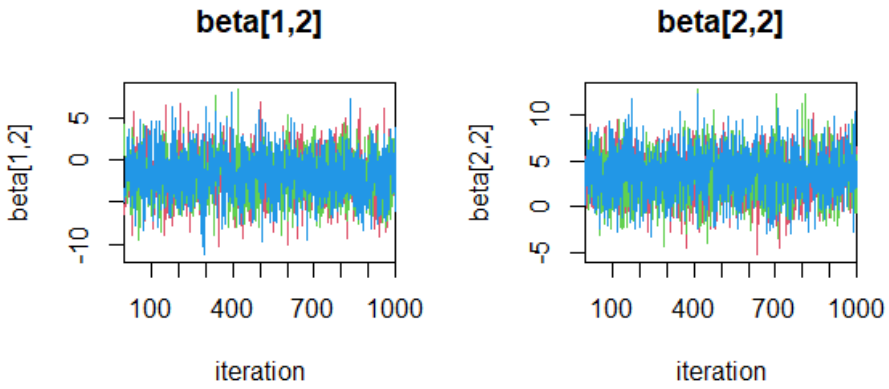
The time execution for this model is:

```
Time difference of 18.6473 mins
```

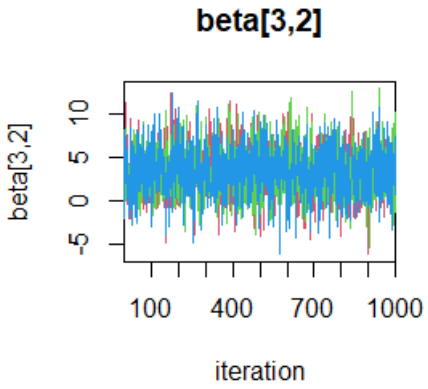
The first technique we use for diagnostic is represented by traceplots. They show the tendency of the samples for each parameter. In our case we have 36 traceplots since the model has 4 intercepts beta0 and 8 parameters beta for each of the 4 categories. Traceplots are displayed in Figure 5 and they show a good mix of the three chains of the model, exploring the entire parameter space. In these plots the firt chain is colored in red, the second in green and the third one in light blue.

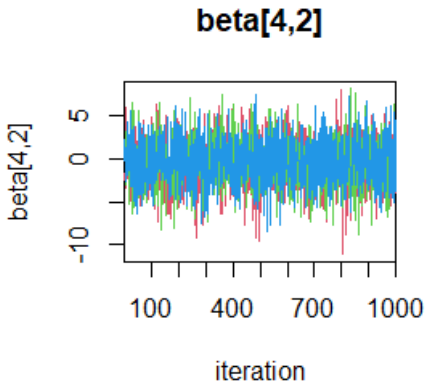
```
# Summary trace plots parameters (beta for each
# class and beta0 for each class) - of type
# beta[i, class] and beta0[class]
traceplot(jags.1, varname = c("beta"), ask = FALSE,
    col = c(2, 3, 4), match.head = FALSE)
# COLORS: 2=red, 3=green, 4=light blue
```

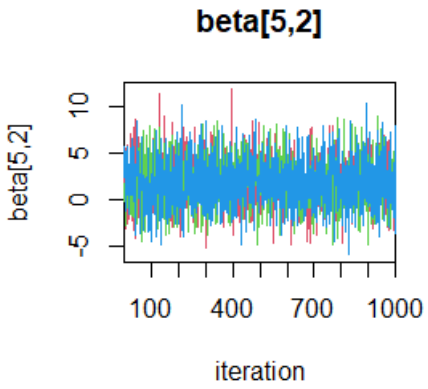

beta[1,1]
beta[1,1]
5
0
-10
100
400
700
1000
iteration
beta[2,1]
beta[2,1]
0
-5
-10
100
400
700
1000
iteration
beta[3,1]
beta[3,1]
5
0
-10
100
400
700
1000
iteration
beta[4,1]
beta[4,1]
15
5
0
-5
100
400
700
1000
iteration
beta[5,1]
beta[5,1]
10
5
0
-5
100
400
700
1000
iteration
beta[6,1]
beta[6,1]
5
0
-5
100
400
700
1000
iteration
beta[7,1]
beta[7,1]
10
5
0
-5
100
400
700
1000
iteration
beta[8,1]
beta[8,1]
5
0
-5
100
400
700
1000
iteration

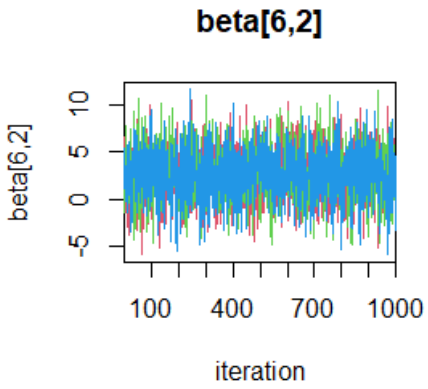

beta[1,2]
beta[1,2]
-10
0
5
100
400
700
1000
iteration

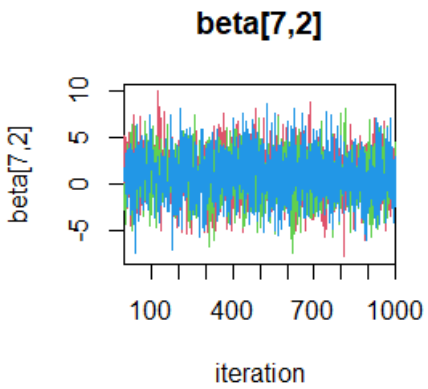

beta[2,2]
beta[2,2]
-5
0
5
10
100
400
700
1000
iteration

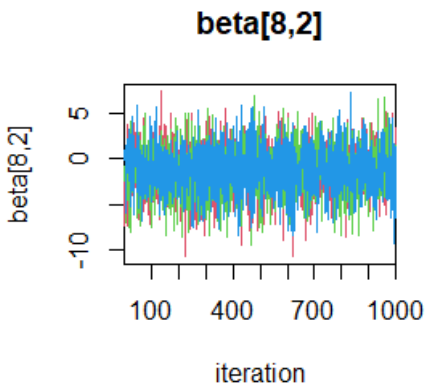

beta[3,2]
beta[3,2]
-5
0
5
10
100
400
700
1000
iteration

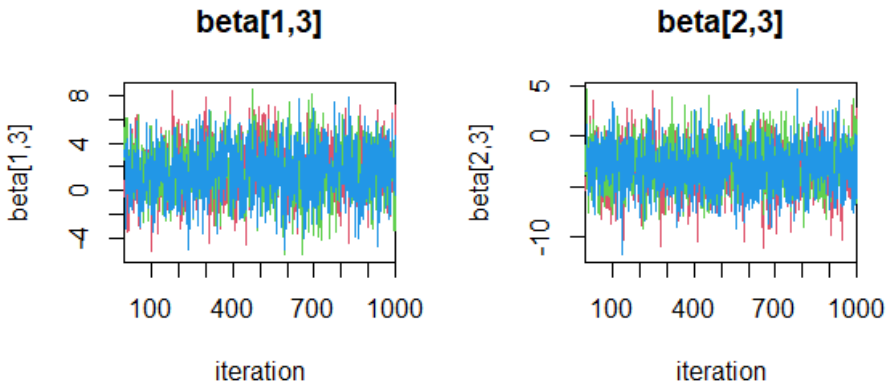

beta[4,2]
beta[4,2]
-10
0
5
100
400
700
1000
iteration


beta[5,2]
beta[5,2]
-5
0
5
10
100
400
700
1000
iteration

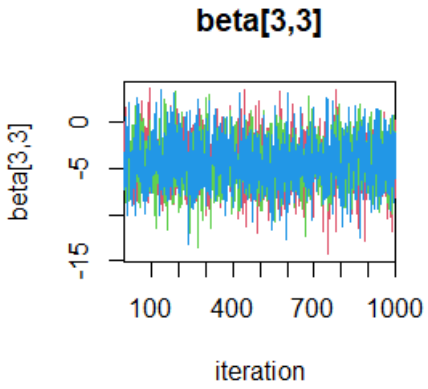

beta[6,2]
beta[6,2]
-5
0
5
10
100
400
700
1000
iteration

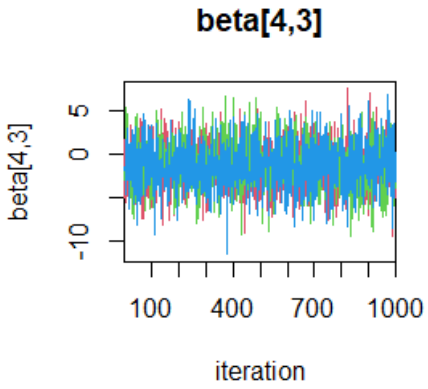

beta[7,2]
beta[7,2]
-5
0
5
10
100
400
700
1000
iteration

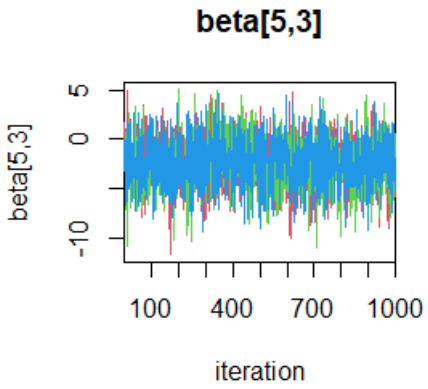

beta[8,2]
beta[8,2]
-10
0
5
100
400
700
1000
iteration

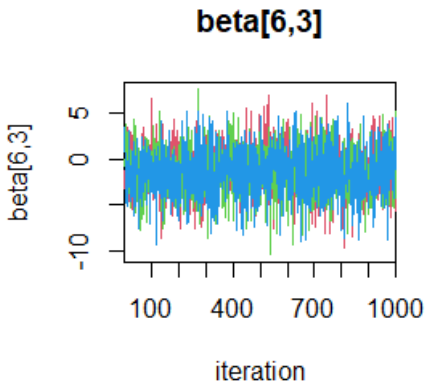

beta[1,3]
beta[1,3]
8
4
0
-4
100
400
700
1000
iteration

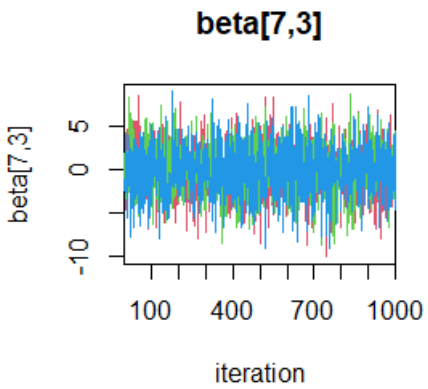

beta[2,3]
beta[2,3]
5
0
-10
100
400
700
1000
iteration

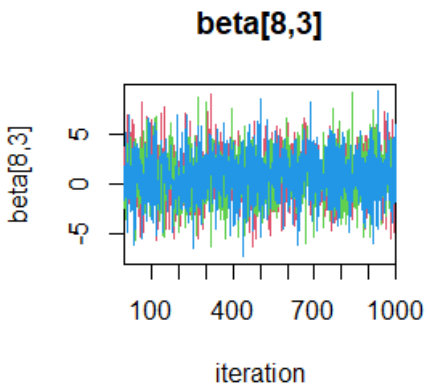

beta[3,3]
beta[3,3]
0
-5
-15
100
400
700
1000
iteration

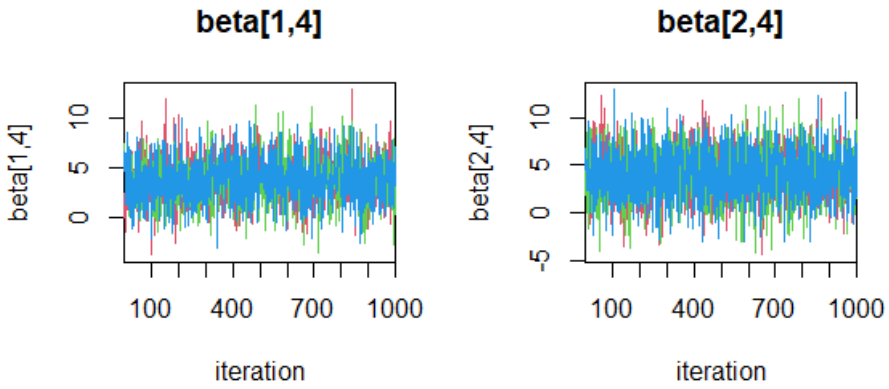

beta[4,3]
beta[4,3]
5
0
-10
100
400
700
1000
iteration


beta[5,3]
beta[5,3]
5
0
-10
100
400
700
1000
iteration

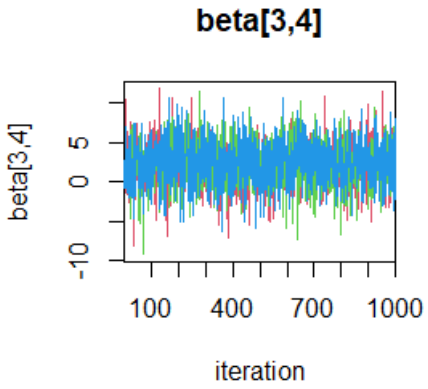

beta[6,3]
beta[6,3]
5
0
-10
100
400
700
1000
iteration

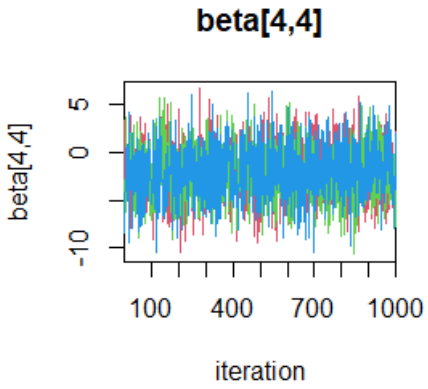

beta[7,3]
beta[7,3]
5
0
-10
100
400
700
1000
iteration

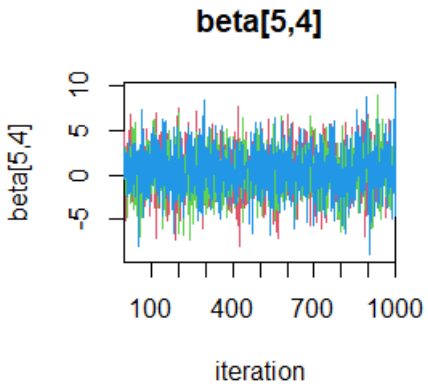

beta[8,3]
beta[8,3]
5
0
-5
100
400
700
1000
iteration

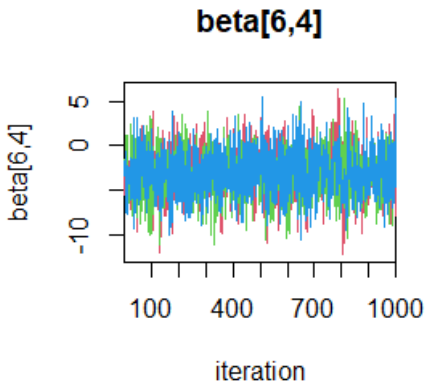

beta[1,4]
beta[1,4]
0
5
10
100
400
700
1000
iteration

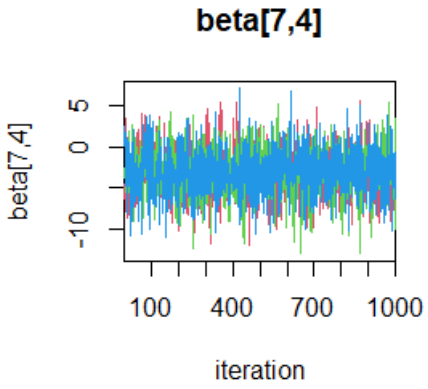

beta[2,4]
beta[2,4]
-5
0
5
10
100
400
700
1000
iteration

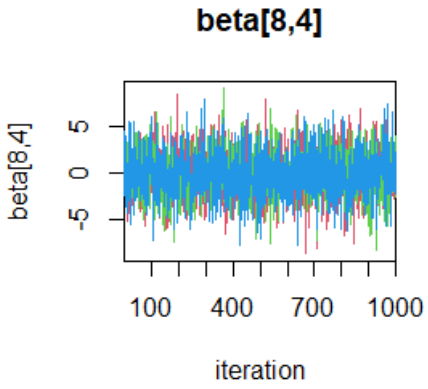

beta[3,4]
beta[3,4]
-10
0
5
100
400
700
1000
iteration

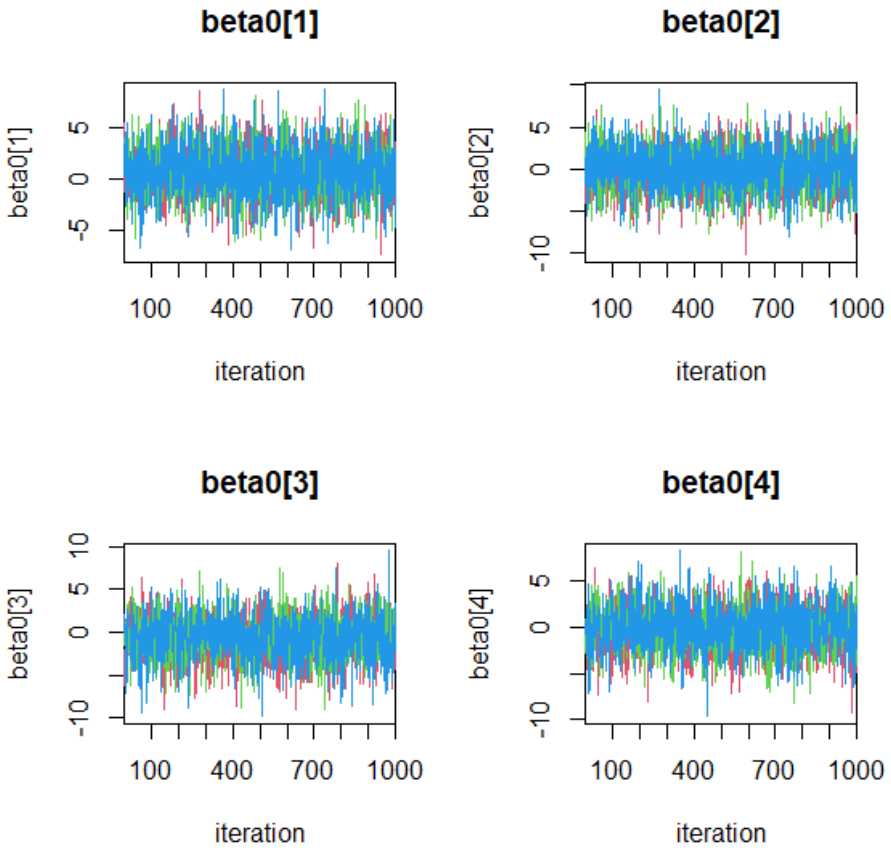

beta[4,4]
beta[4,4]
-10
0
5
100
400
700
1000
iteration


beta[5,4]
beta[5,4]
-5
0
5
10
100
400
700
1000
iteration


beta[6,4]
beta[6,4]
-10
0
5
100
400
700
1000
iteration


beta[7,4]
beta[7,4]
-10
0
5
100
400
700
1000
iteration


beta[8,4]
beta[8,4]
-5
0
5
100
400
700
1000
iteration

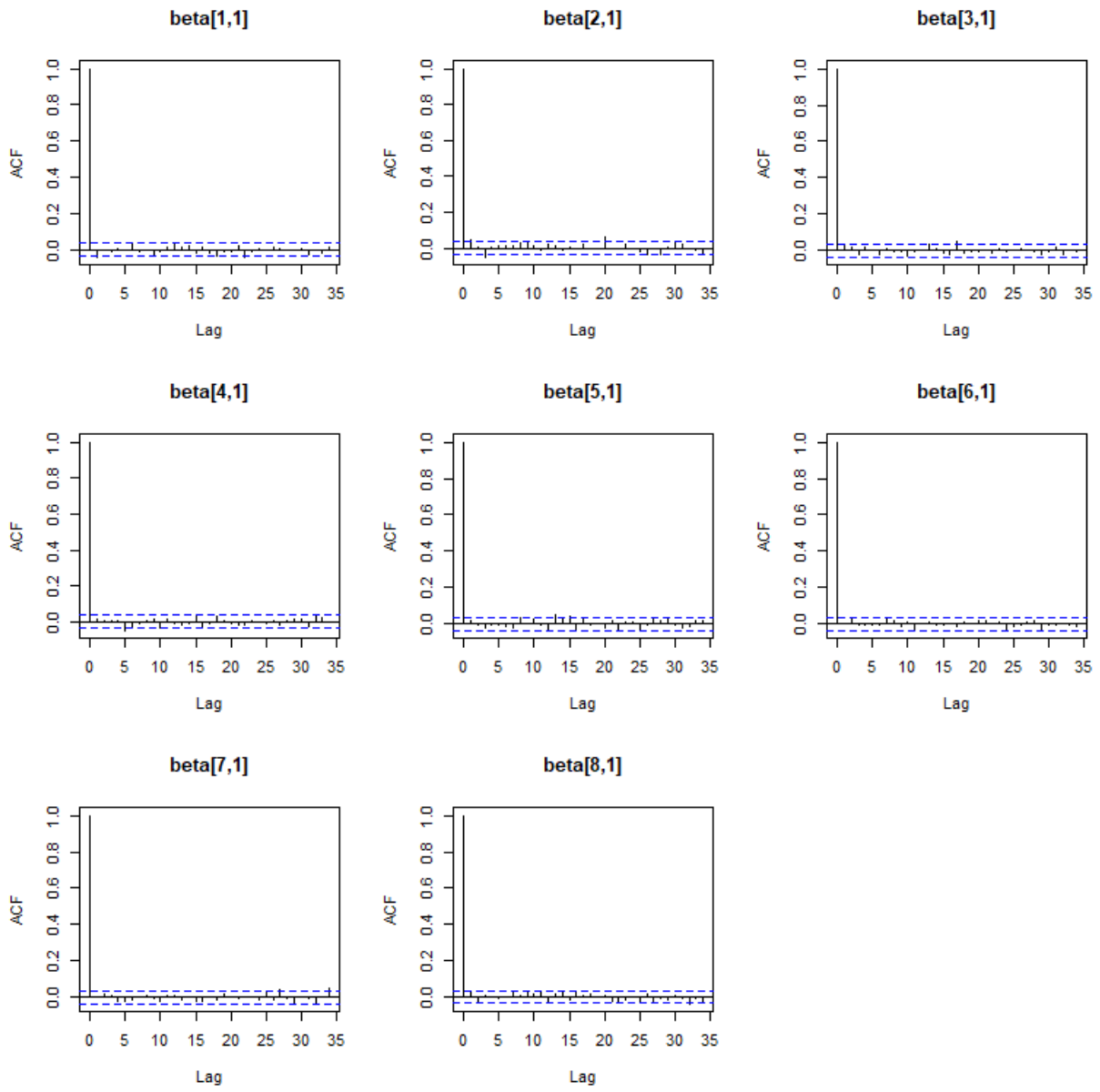



Traceplots, First model.

Another important instrument to control the convergence diagnostic of a Bayesian model, is represented also by **autocorrelation plots**. They are used to visualize the level of correlation between samples of the MCMC chains, i.e. they evaluate the dependence of a sample in a specific point of a chain with the previous and the following ones. This measure is made through *lags* that are like *delays*. Hence, for example, the lag 1 measures the correlation between a sample and the successive one, lag 2 compares every sample with the sample at distance 2, etc. This process implies that, if the autocorrelation is high, it means that samples are really dependent between each others. An autocorrelation at lag $t$ is indicated by $\rho_t$ and varies between 1 and -1. Another specification is that $\rho_0$ is always 1 since every sample is perfectly correlated with itself. A good model should have chains that mixed well such that the autocorrelation should go forward 0 quickly. Figures 6, 7, 8, 9 and 10 shown the autocorrelation plots for each parameter of the model: the 6, 7, 8 and 9 are about beta parameters for each class respectively, instead the last group of graphs regards the intercept beta0 for each class. As we can see, the autocorrelations are inside the confidence bands, beside some slightly exception for a sample like in beta[4,3] (fourth beta for class 3) or in beta0[1]. These confidence bands are calculated also considering the number of samples $n$. In our case we have 10000 iterations, 2000 burn_in and there is no thinning such that every sample of a chain is maintained. Then we have 8000 samples for each chain. Eventually, to plot the autocorrelations just for the first chain, we use this structure:

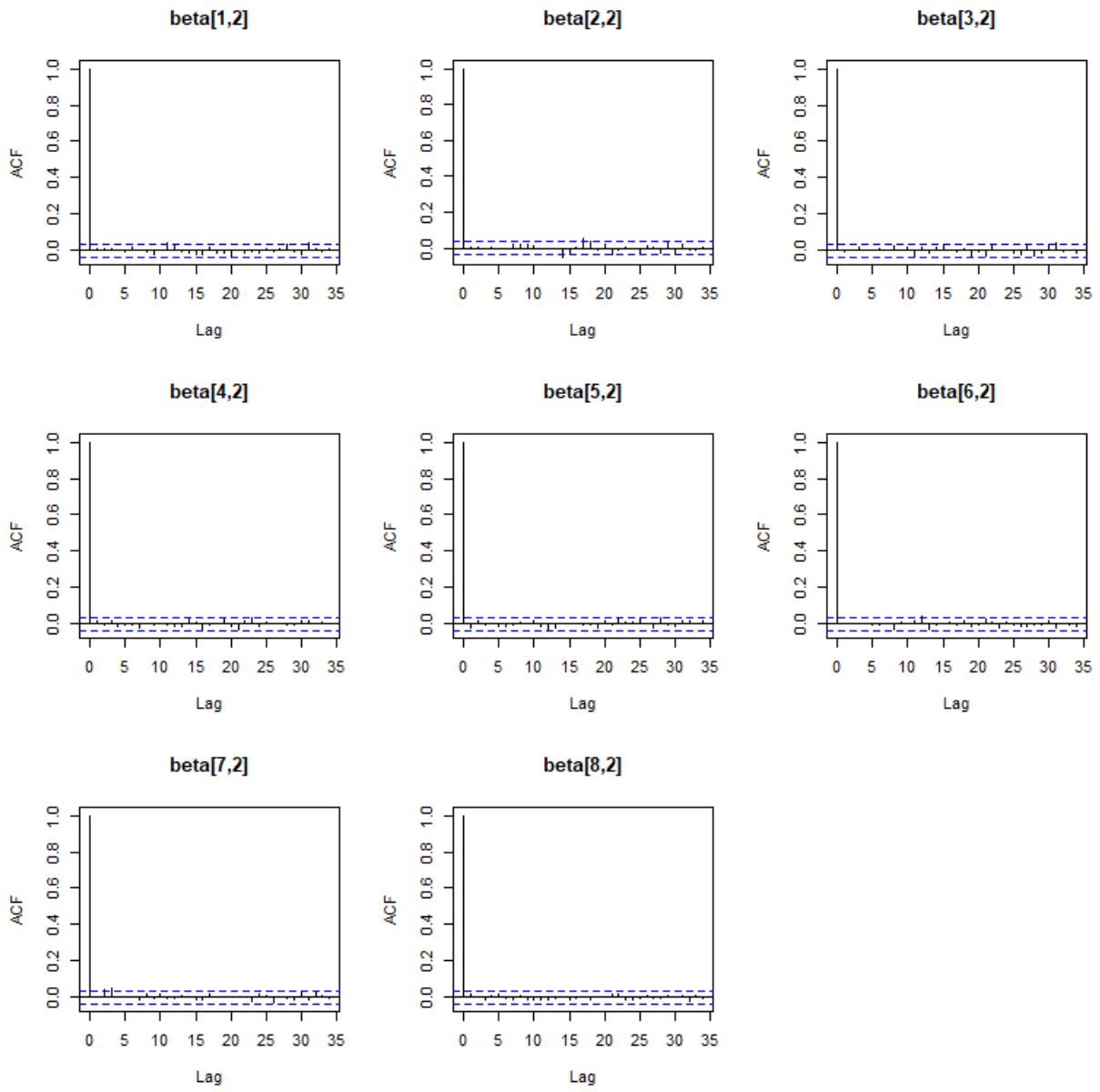
```
  acf(jags.1$BUGSoutput$sims.array[,1,parameter])
```

```
# Summary plot acf
par(mfrow = c(3, 3))
for (i in 1:8) {
    acf(jags.1$BUGSoutput$sims.matrix[, paste0("beta[",
        i, ",", 1, "]")], main = paste0("beta[", i,
        ",", 1, "]"))
}
par(mfrow = c(1, 1))
```



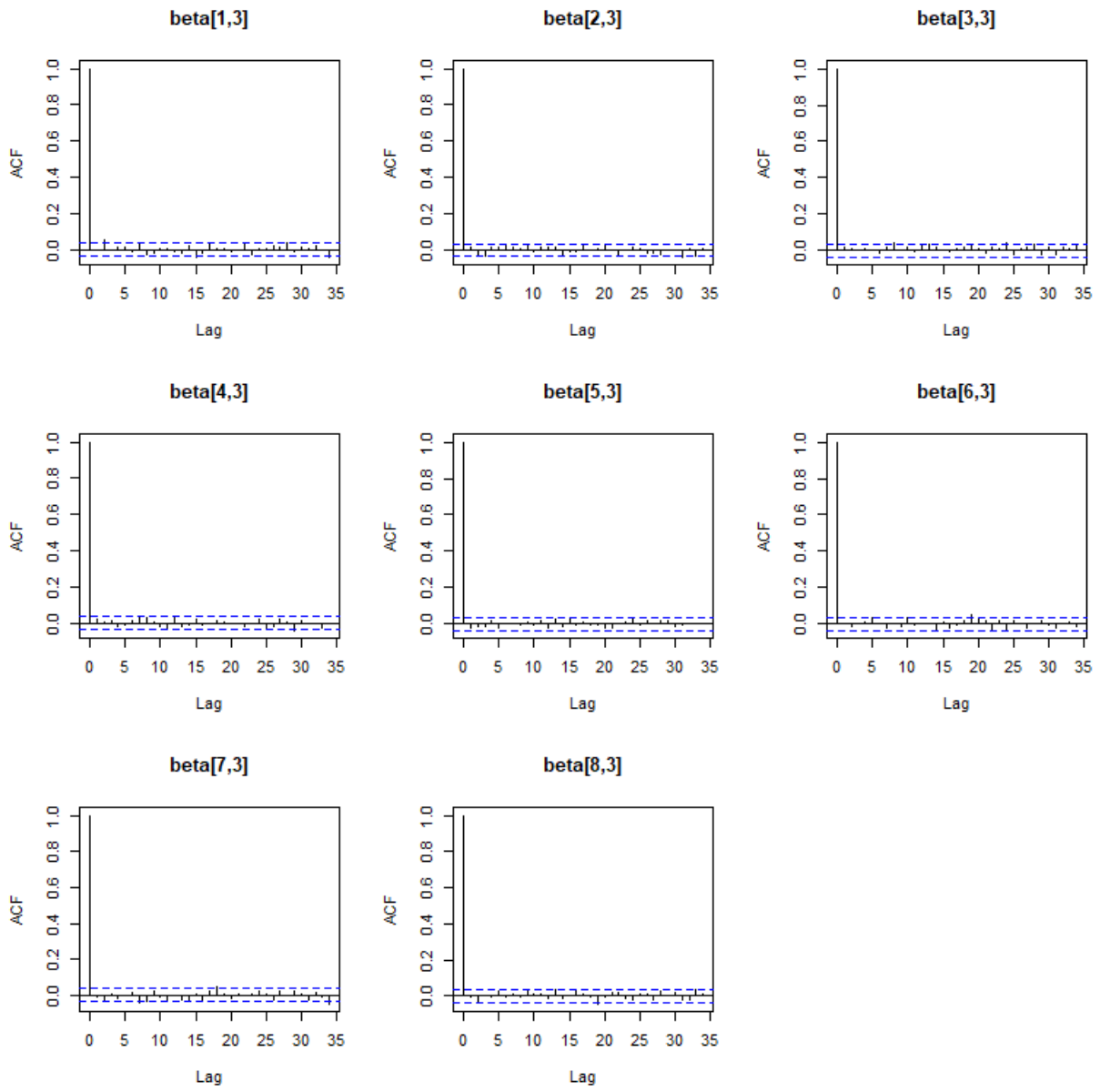

ACF plots beta - First class SA, First model.

```
# Summary plot acf
par(mfrow = c(3, 3))
for (i in 1:8) {
    acf(jags.1$BUGSoutput$sims.matrix[, paste0("beta[",
        i, ",", 2, "]")], main = paste0("beta[", i,
        ",", 2, "]"))
}
par(mfrow = c(1, 1))
```



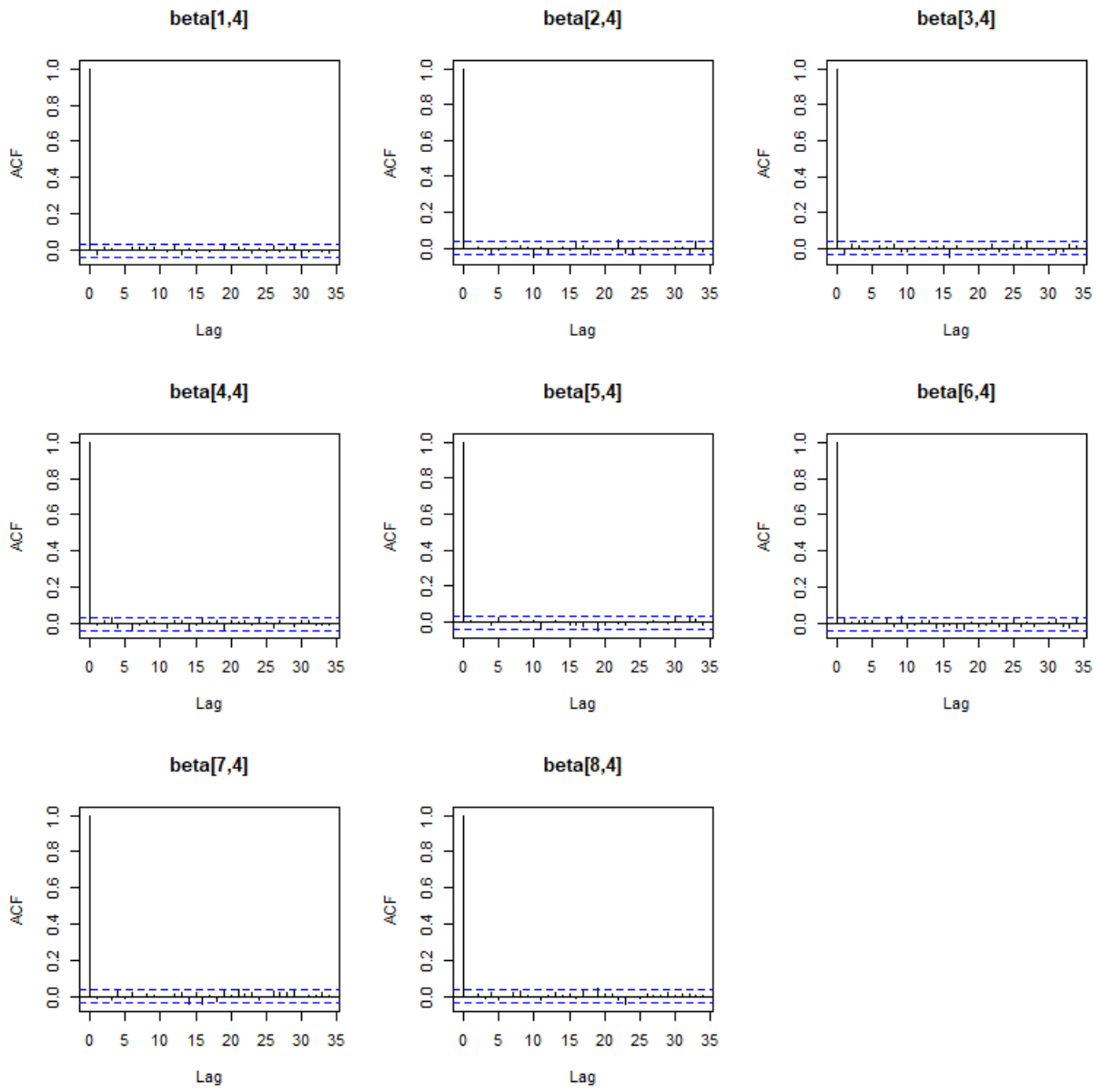

ACF plots beta - Second class SB, First model.

```
# Summary plot acf
par(mfrow = c(3, 3))
for (i in 1:8) {
    acf(jags.1$BUGSoutput$sims.matrix[, paste0("beta[",
        i, ",", 3, "]")], main = paste0("beta[", i,
        ",", 3, "]"))
}
par(mfrow = c(1, 1))
```



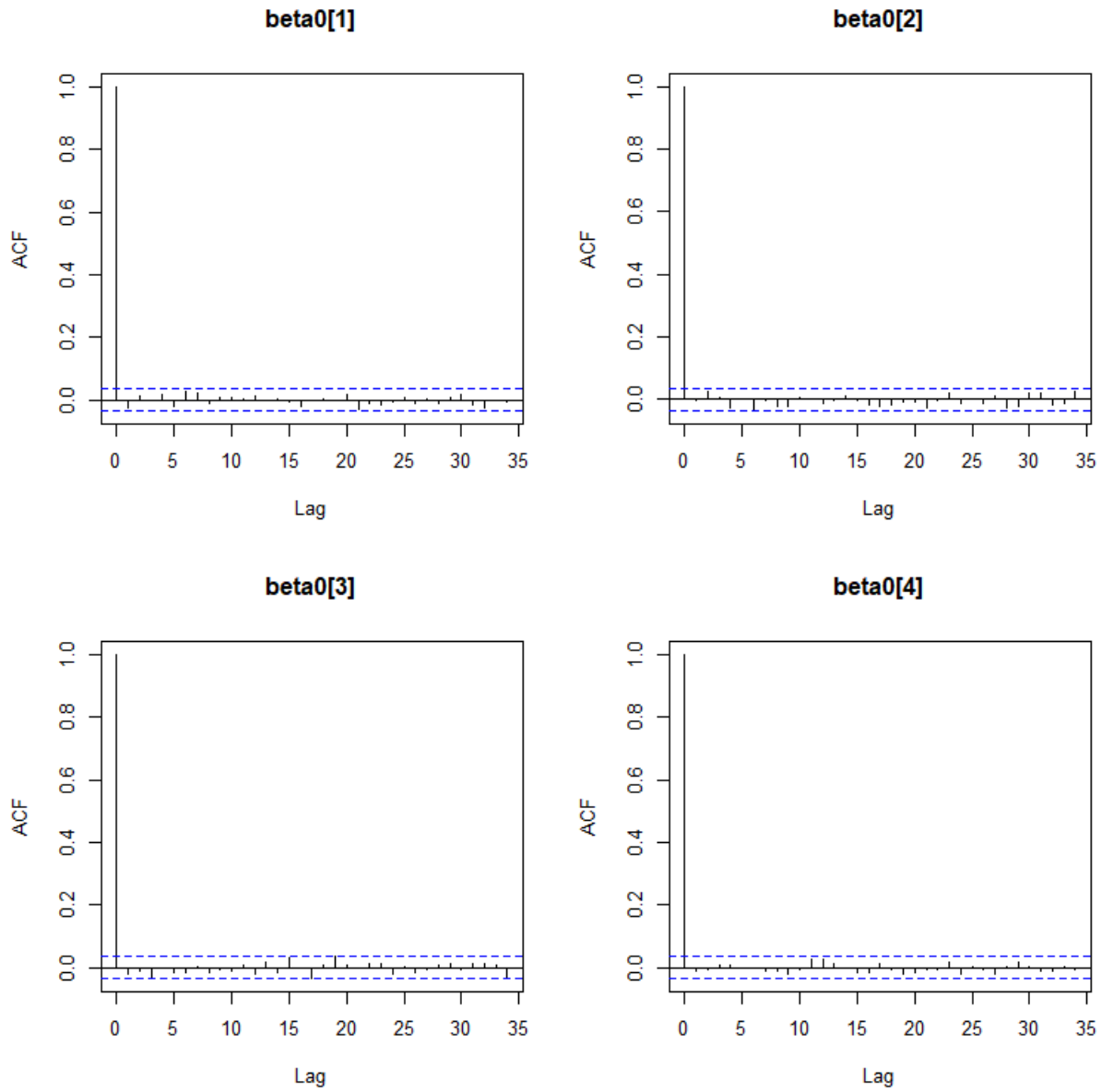

ACF plots beta - Third class TA, First model.

```
# Summary plot acf
par(mfrow = c(3, 3))
for (i in 1:8) {
    acf(jags.1$BUGSoutput$sims.matrix[, paste0("beta[",
        i, ",", 4, "]")], main = paste0("beta[", i,
        ",", 4, "]"))
}
par(mfrow = c(1, 1))
```




ACF plots beta - Fourth class TB, First model.

```
# Summary plot acf
par(mfrow = c(2, 2))
for (i in 1:4) {
    acf(jags.1$BUGSoutput$sims.matrix[, paste0("beta0[",
        i, "]")], main = paste0("beta0[", i, "]"))
}
```




ACF plots intercept, First model.

```
par(mfrow = c(1, 1))
```

Now we can look at the estimates of the parameters of the first model, and analyze its inference. The summaries of the First model are shown in Table 4 and 5. In Table 4 there are the summaries for the parameters *beta0* (for every class), and in Table 5 there are the summaries for the eight parameters *beta* (each for every class). The columns 2.5% and 97.5% represent the limits of the confidence interval at level $\alpha = 0.05$. These intervals show an high level of variability, for example for beta[8,2]. Also, some of them as for example beta[5,1], beta[7,3], beta[8,4], have a mean very close to 0, so for these parameters it is not possible to exclude the null effect. From the results of Table 5 of beta, we can also observe that plants with high value of Average of chlorophyll in the plant (ACHP), with parameter beta[1,k], are likely to belong to classes 3 (TA) and 4 (TB), i.e. to the Traditional greenhouse. Instead, plants with high Average root length (ARL), i.e. beta[6,k], and Average wet weight of the root (AWWR), i.e. beta[7,k], are likely to belong to the first two classes (Smart greenhouse). Regarding some possible interpretations for differences in varieties, we can see that plants of Syrian variety (B) probably have higher values of Plant height rate (PHR), i.e. beta[2,k], and Average leaf area of the plant (ALAP), i.e. beta[3,k], but also a likely lower value of Percentage of dry matter for root growth (PDMRG), i.e. beta[8,k], compared to the Dutch variety (A). From the results of Table 4 of beta0, we can see that the intercept for class 2 and 4 is really close to 0, and in general for each class beta0 has a high variability. Practically, since also the other intercept have an estimated mean of less than 1, this means that when all the predictors are 0, there is not so much difference between classes.

```
# Extracting estimates and quantiles of beta0 and
# beta
summary.jags1.beta0 <- as.data.frame(jags.1$BUGSoutput$summary[grep("beta0",
    rownames(jags.1$BUGSoutput$summary)), ][, c(1:3,
    5, 7:9)])
summary.jags1.beta0 <- round(summary.jags1.beta0, 2)
# Table 4 beta0
kable(summary.jags1.beta0, caption = "First Bayesian model analysis and inference of beta0.",
    align = "c")
```

First Bayesian model analysis and inference of beta0.

| | mean | sd | 2.5% | 50% | 97.5% | Rhat | n.eff |
|---|---|---|---|---|---|---|---|
| beta0[1] | 0.67 | 2.45 | -4.18 | 0.71 | 5.40 | 1.00 | 2000 |
| beta0[2] | 0.09 | 2.62 | -5.15 | 0.17 | 5.00 | 1.00 | 1900 |
| beta0[3] | -0.71 | 2.61 | -5.96 | -0.71 | 4.17 | 1.00 | 820 |
| beta0[4] | -0.18 | 2.49 | -5.15 | -0.15 | 4.64 | 1.01 | 390 |

```
# Extracting estimates and quantiles of beta
summary.jags1.beta <- as.data.frame(jags.1$BUGSoutput$summary[grep("^beta\\[",
    rownames(jags.1$BUGSoutput$summary)), ][, c(1:3,
    5, 7:9)])
summary.jags1.beta <- round(summary.jags1.beta, 2)
# Table 5 beta
kable(summary.jags1.beta, caption = "First Bayesian model analysis and inference of beta.",
    align = "c")
```

First Bayesian model analysis and inference of beta.

| | mean | sd | 2.5% | 50% | 97.5% | Rhat | n.eff |
|---|---|---|---|---|---|---|---|
| beta[1,1] | -3.33 | 2.46 | -8.35 | -3.24 | 1.34 | 1.00 | 3000 |
| beta[2,1] | -4.27 | 2.40 | -9.12 | -4.21 | 0.34 | 1.00 | 1500 |
| beta[3,1] | -1.02 | 2.76 | -6.47 | -1.04 | 4.32 | 1.00 | 1100 |
| beta[4,1] | 3.84 | 2.65 | -1.38 | 3.87 | 8.99 | 1.00 | 3000 |
| beta[5,1] | 0.21 | 2.60 | -5.06 | 0.24 | 5.24 | 1.00 | 3000 |
| beta[6,1] | 1.56 | 2.84 | -3.96 | 1.54 | 7.01 | 1.00 | 1400 |
| beta[7,1] | 2.20 | 2.66 | -3.06 | 2.16 | 7.50 | 1.00 | 3000 |
| beta[8,1] | 0.65 | 2.53 | -4.20 | 0.63 | 5.69 | 1.00 | 1800 |
| beta[1,2] | -1.81 | 2.74 | -7.18 | -1.77 | 3.50 | 1.00 | 1700 |
| beta[2,2] | 3.41 | 2.55 | -1.51 | 3.40 | 8.43 | 1.00 | 840 |
| beta[3,2] | 3.35 | 2.74 | -1.91 | 3.33 | 8.78 | 1.00 | 1800 |
| beta[4,2] | -0.47 | 2.69 | -5.72 | -0.43 | 4.85 | 1.00 | 2100 |
| beta[5,2] | 1.90 | 2.48 | -2.74 | 1.86 | 7.01 | 1.00 | 3000 |
| beta[6,2] | 2.68 | 2.77 | -2.81 | 2.70 | 7.99 | 1.00 | 3000 |
| beta[7,2] | 0.93 | 2.60 | -4.16 | 0.94 | 5.95 | 1.00 | 3000 |
| beta[8,2] | -1.31 | 2.67 | -6.64 | -1.23 | 3.83 | 1.00 | 3000 |
| beta[1,3] | 1.56 | 2.14 | -2.56 | 1.55 | 5.77 | 1.00 | 1800 |
| beta[2,3] | -3.05 | 2.31 | -7.55 | -3.07 | 1.37 | 1.01 | 300 |
| beta[3,3] | -4.53 | 2.64 | -9.54 | -4.57 | 0.84 | 1.00 | 1600 |
| beta[4,3] | -1.42 | 2.61 | -6.43 | -1.44 | 3.81 | 1.00 | 3000 |

| | mean | sd | 2.5% | 50% | 97.5% | Rhat | n.eff |
|---|---|---|---|---|---|---|---|
| beta[5,3] | -2.44 | 2.52 | -7.27 | -2.40 | 2.31 | 1.00 | 3000 |
| beta[6,3] | -1.40 | 2.59 | -6.45 | -1.47 | 3.75 | 1.00 | 550 |
| beta[7,3] | -0.14 | 2.87 | -5.96 | -0.15 | 5.38 | 1.00 | 3000 |
| beta[8,3] | 0.74 | 2.42 | -4.13 | 0.79 | 5.45 | 1.00 | 3000 |
| beta[1,4] | 3.73 | 2.21 | -0.46 | 3.75 | 8.09 | 1.00 | 2400 |
| beta[2,4] | 4.10 | 2.57 | -0.89 | 4.12 | 8.96 | 1.00 | 3000 |
| beta[3,4] | 2.25 | 2.75 | -3.22 | 2.27 | 7.54 | 1.00 | 1100 |
| beta[4,4] | -2.13 | 2.70 | -7.48 | -2.09 | 3.19 | 1.00 | 3000 |
| beta[5,4] | 0.37 | 2.49 | -4.49 | 0.40 | 5.30 | 1.00 | 1600 |
| beta[6,4] | -3.06 | 2.60 | -8.20 | -3.02 | 1.95 | 1.00 | 3000 |
| beta[7,4] | -3.12 | 2.81 | -8.81 | -3.03 | 2.37 | 1.00 | 2300 |
| beta[8,4] | -0.04 | 2.55 | -5.04 | -0.04 | 5.03 | 1.00 | 2100 |

As we can see from the tables, the sixth column is called **Rhat**. Introduced by Gelman and Rubin, it is also called $\hat{R}$ and it is another convergence diagnostic to evaluate if the chains converged to the stationary distribution. Practically, it compares the variance *between* $M$ simulated chains defined by

$$B_T = \frac{1}{M}\sum_{m=1}^{M}(\hat{I}^{(m)} - \hat{I}_T)^2$$

where $T$ is the number of iterations, with the variance *within* each chain defined by

$$W_T = \frac{1}{M}\sum_{m=1}^{M}[\frac{1}{T}\sum_{t=1}^{T}(h(\theta_i^{(m)}) - \hat{I}_T^{(m)})^2].$$

The formula of the exact Rhat is the following:

$$R_T = \frac{\frac{T-1}{T}W_T + \frac{M+1}{M}B_T}{W_T}.$$

When $T \to \infty$, $R_T$ is expected to decrease to 1. Hence the formula in this case is:

$$\hat{R} = \sqrt{\frac{\hat{W}}{\hat{B}}}$$

where $\hat{W}$ is the estimate of the variance within chains, and $\hat{B}$ is the estimate of the variance between chains. As we can see from the Tables 4 and 5, we have values of Rhat that suggest that the chains are well mixed (all are 1).

The last column is instead **n.eff**. This column represents the **Effective Sample Size (ESS)**, another method for the diagnostic of a Bayesian model. It is a measure that evaluates the quality of the samples generated by MCMC sampling and it gives the number of independent and identically distributed samples produced, despite they are correlated due to temporal dependency. It can be expressed by

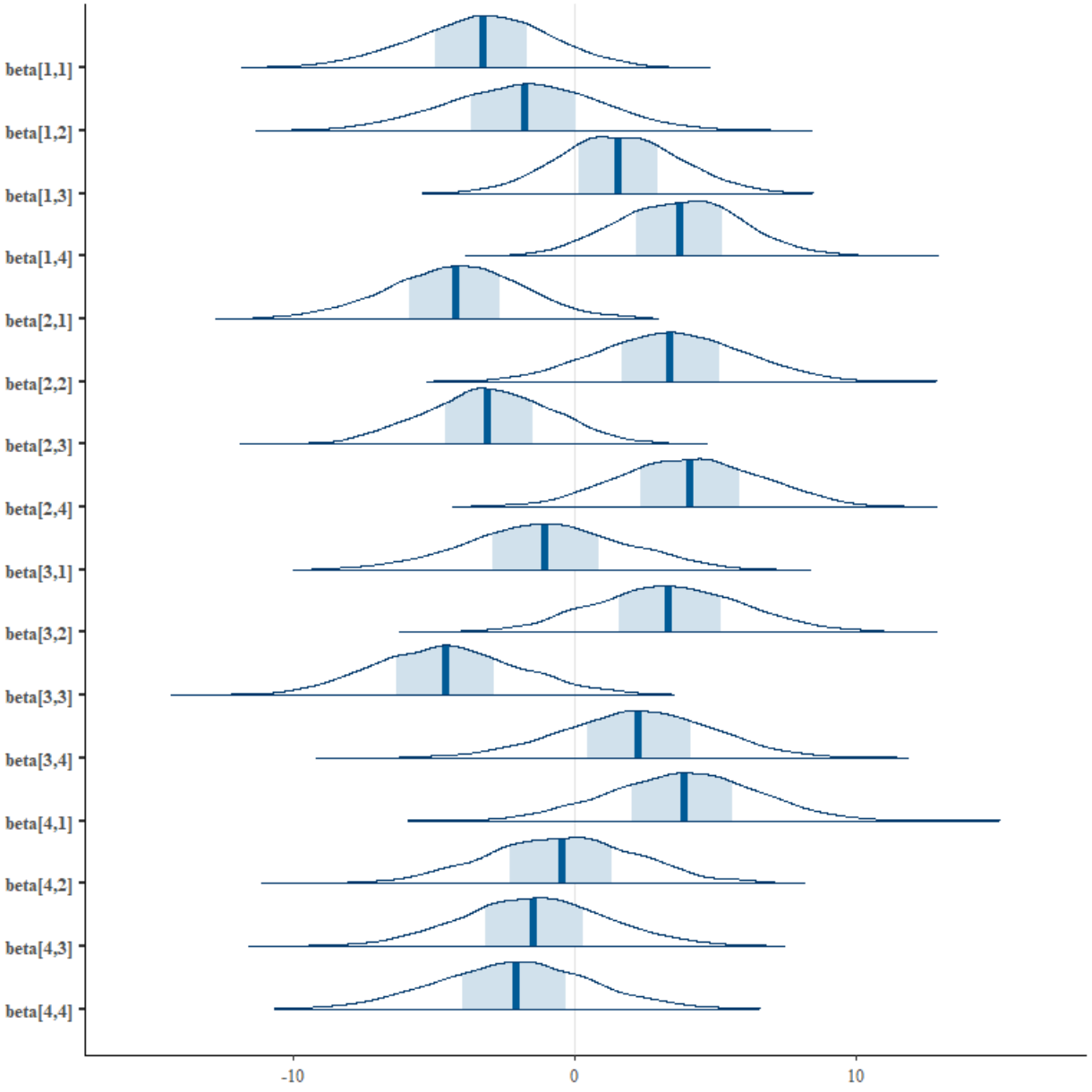
$$ESS = \frac{N}{1 + 2\sum_{k=1}^{\infty}\rho_k}$$

where $N$ is the number of total samples and $\rho_k$ is the autocorrelation at lag $k$. The more the autocorrelation between samples, the lower ESS will be compared to the real dimension of the samples $N$.

From Tables 4 and 5 we can see that the majority of the parameters have a ESS around 3000, that is elevate. This suggests that the MCMC explored well the parameter space, besides beta[6,4] that with a $n.eff$ of 8400 could indicate autocorrelation, but still acceptable (since it is close to 1000).

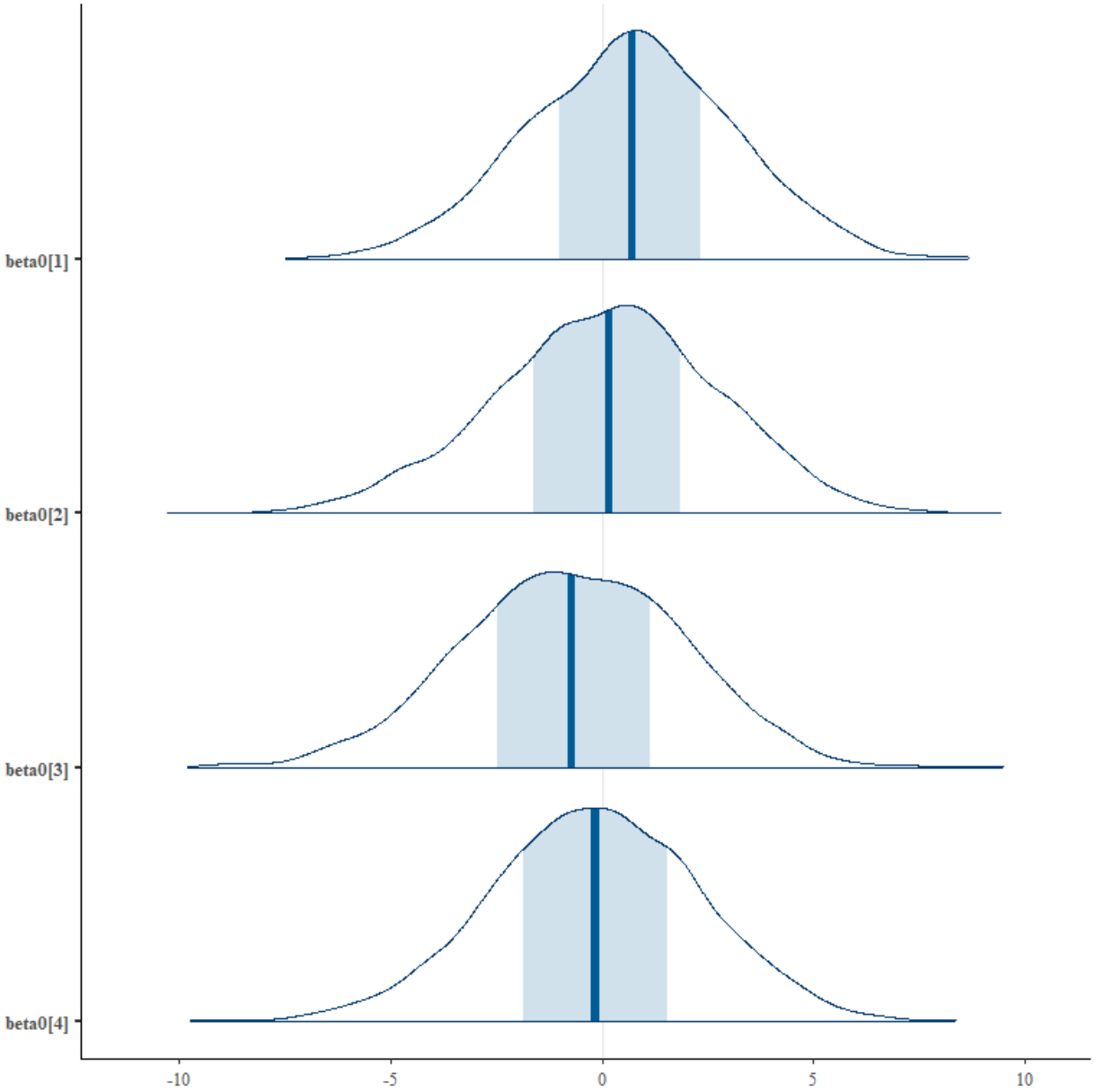
Now, let's look at the parameter's distribution. We can extract the information in this way:

```
# beta - Fig.11
mcmc.jags1 <- as.mcmc(jags.1)
mcmc_areas(mcmc.jags1, pars = c("beta[1,1]", "beta[1,2]",
    "beta[1,3]", "beta[1,4]", "beta[2,1]", "beta[2,2]",
    "beta[2,3]", "beta[2,4]", "beta[3,1]", "beta[3,2]",
    "beta[3,3]", "beta[3,4]", "beta[4,1]", "beta[4,2]",
    "beta[4,3]", "beta[4,4]"))
```




Posterior Distributions of beta, First model.

```
# beta0 - Fig. 12
mcmc_areas(mcmc.jags1, pars = c("beta0[1]", "beta0[2]",
    "beta0[3]", "beta0[4]"))
```




Posterior Distributions of beta0, First model.

As we can see the distributions seem symmetric and this is a good sign of convergence, even if the distributions have high variability. Moreover, the parameters beta[4,2] has a mean very close to zero, suggesting that it can be not significant. Same situation for the intercepts beta0[2] and beta0[4], as we have already noticed also by Tables 4 and 5 of parameters' estimates.

Continuing with the diagnostic, another important measure to look at is the **Monte Carlo Standard Error (MCSE)** that quantifies the uncertainty of the estimate of Monte Carlo simulation. It is the residual variability of the estimate due to the sampling, taking into account the autocorrelation. Basically, it measures how much an estimate obtained from MCMC simulation could vary from a simulation to another and, in this sense, it is an indicator of precision of the estimate of a quantity or the parameter of interest. If there was not autocorrelation between samples, MCSE would be equal to the Standard Error

$$SE = \frac{\sigma}{\sqrt{N}}$$

where $\sigma$ is the sampling standard deviation, and $N$ is the dimension of the sample. Due to autocorrelation, we have a reduction in the *Effective number of samples* as explained before, then the MCSE is defined as:

$$MCSE = \frac{\hat{\sigma}}{\sqrt{ESS}}.$$

We interpret a low MCSE as a more precise estimate for that parameter. Also, if ESS is too low, MCSE will be higher meaning that the estimates of the model are not accurate. This could suggest that the chain does not converge.

From the results shown below, we can see that for the intercept beta0 we have a standard error between 0.09 and 0.13, so this suggests a good precision in the estimates. Similar situation for beta parameters, in fact the s.e. varies between 0.08 and 0.13.

```
mcse.jags1 <- list()
# MCSE for beta0
mcse.jags1$beta0 <- lapply(1:3, function(mc) {
    sapply(1:4, function(class) {
        # Extract samples of beta0 for the chain
        # mc and the category class
        param.name <- paste0("beta0[", class, "]")
        MCSE(jags.1$BUGSoutput$sims.array[, mc, param.name])
    })
})

# MCSE for beta (8 parameters for 4 classes)
mcse.jags1$beta <- lapply(1:3, function(mc) {
    sapply(1:4, function(class) {
        sapply(1:8, function(coef) {
            # Extract samples of beta for the
            # chain mc and the category class
            param.name <- paste0("beta[", coef, ",",
                class, "]")
            MCSE(jags.1$BUGSoutput$sims.array[, mc,
                param.name])
        })
    })
})

# Results in 3 matrices (one per mc)
mcse.jags1$beta <- lapply(mcse.jags1$beta, function(beta.mc) {
    matrix(beta.mc, nrow = 8, ncol = 4)
})

# Show results
mcse.jags1
```

```
$beta0
$beta0[[1]]
[1] 0.1029925 0.1002628 0.1380075 0.1177316

$beta0[[2]]
[1] 0.09320065 0.09443793 0.13554549 0.10199904

$beta0[[3]]
[1] 0.08959609 0.08384003 0.13995976 0.11519911


$beta
$beta[[1]]
          [,1]       [,2]       [,3]       [,4]
[1,] 0.09535046 0.10889395 0.10545652 0.10528972
[2,] 0.10582112 0.11757580 0.11434082 0.12064407
[3,] 0.09221739 0.08957009 0.10131204 0.10572581
[4,] 0.09652237 0.09027923 0.10708929 0.11836464
[5,] 0.08564159 0.07804389 0.09240635 0.08900468
[6,] 0.09892830 0.09934398 0.12611501 0.12763821
[7,] 0.09535503 0.08647650 0.11017063 0.12612694
[8,] 0.08759592 0.08762816 0.09802955 0.10155594

$beta[[2]]
          [,1]       [,2]       [,3]       [,4]
[1,] 0.09342480 0.10382598 0.11303035 0.11069462
[2,] 0.10596248 0.08974524 0.09633362 0.10581470
[3,] 0.08971436 0.09643861 0.10077794 0.09993933
[4,] 0.09835679 0.08400938 0.12041149 0.11504458
[5,] 0.08218627 0.08465785 0.08878351 0.08088412
[6,] 0.10508709 0.11349216 0.10984520 0.12361884
[7,] 0.09171168 0.08280547 0.11595174 0.13636993
[8,] 0.07791461 0.08469640 0.09126735 0.09379496

$beta[[3]]
          [,1]       [,2]       [,3]       [,4]
[1,] 0.10756561 0.09884856 0.11233939 0.10473967
[2,] 0.11081333 0.09164913 0.09498779 0.09967614
[3,] 0.09323448 0.12077641 0.10843751 0.10406944
[4,] 0.10759431 0.08689978 0.11022894 0.11222361
[5,] 0.08536434 0.08677011 0.08501697 0.08266460
[6,] 0.10378273 0.10780878 0.11654672 0.13019343
[7,] 0.09581337 0.08780450 0.11872554 0.11192571
[8,] 0.09046812 0.09107016 0.09654955 0.09033340
```

**Geweke diagnostic** is instead a convergence diagnostic to evaluate if the chains have reached the stationarity. It is based on a test for the equality of the means of the first and last part of a Markov chain (by default the first 10% and the last 50%), to determine if the chain converged. In this sense it evaluates also if the sampling is representative, since if the samples are produced by the stationary distribution of the chain, the two means are equal. In this case the Geweke's statistic has an asymptotically standard normal distribution. If the statistic Z is close to 0, the means of the two sections are similar and the chain converges. If Z is far away from 0, this suggests that the means are different and the chain could not converge. The Geweke test is calculated for each parameter. The results ar shown below:

```
mcmc1.final <- mcmc.list(mcmc.jags1)
# Geweke diagnostic
(geweke.diag.jags1 <- geweke.diag(mcmc1.final))
```

```
[[1]]

Fraction in 1st window = 0.1
Fraction in 2nd window = 0.5

beta[1,1] beta[1,2] beta[1,3] beta[1,4] beta[2,1] beta[2,2]
 -0.73527  -1.11470  -1.96669  -0.96292   1.10708   0.92091
beta[2,3] beta[2,4] beta[3,1] beta[3,2] beta[3,3] beta[3,4]
 -0.41567   2.28789   1.29574   2.47290   2.99348  -0.85399
beta[4,1] beta[4,2] beta[4,3] beta[4,4] beta[5,1] beta[5,2]
  0.43900  -0.65912  -1.78431  -0.33722  -0.42407  -0.07358
beta[5,3] beta[5,4] beta[6,1] beta[6,2] beta[6,3] beta[6,4]
  1.59178   0.23246  -0.01546  -0.42844   0.69558   0.82304
beta[7,1] beta[7,2] beta[7,3] beta[7,4] beta[8,1] beta[8,2]
  0.76303   0.41007   0.99906  -0.38308  -0.93705  -1.02121
beta[8,3] beta[8,4]  beta0[1]  beta0[2]  beta0[3]  beta0[4]
 -0.60206  -1.71757   0.47794   1.03242   1.41992  -1.06825
 deviance
 -0.61133


[[2]]

Fraction in 1st window = 0.1
Fraction in 2nd window = 0.5

beta[1,1] beta[1,2] beta[1,3] beta[1,4] beta[2,1] beta[2,2]
  2.09935   0.86679   1.70411   0.20584  -1.24392  -1.00554
beta[2,3] beta[2,4] beta[3,1] beta[3,2] beta[3,3] beta[3,4]
 -1.22878   0.33719   0.25455  -1.00821   0.25592  -2.48728
beta[4,1] beta[4,2] beta[4,3] beta[4,4] beta[5,1] beta[5,2]
  1.37339   1.68749   1.28087   0.57622  -0.46762  -0.49075
beta[5,3] beta[5,4] beta[6,1] beta[6,2] beta[6,3] beta[6,4]
  0.54440  -1.10313  -0.03858  -0.23691   0.25424   0.28464
beta[7,1] beta[7,2] beta[7,3] beta[7,4] beta[8,1] beta[8,2]
  0.88962   0.23212  -0.33489   1.51429   0.50942   0.12780
beta[8,3] beta[8,4]  beta0[1]  beta0[2]  beta0[3]  beta0[4]
 -1.20700   0.45567   1.09915   1.79036  -0.21755   0.90116
 deviance
  0.34287


[[3]]

Fraction in 1st window = 0.1
Fraction in 2nd window = 0.5

beta[1,1] beta[1,2] beta[1,3] beta[1,4] beta[2,1] beta[2,2]
 -1.37290   1.96538  -1.51242  -1.66736   0.92685  -1.09409
beta[2,3] beta[2,4] beta[3,1] beta[3,2] beta[3,3] beta[3,4]
  1.66297   0.07797  -0.01935   0.44225  -0.26297  -0.69174
beta[4,1] beta[4,2] beta[4,3] beta[4,4] beta[5,1] beta[5,2]
 -2.64731   0.18963   1.56202  -1.83089  -1.37784   0.41149
beta[5,3] beta[5,4] beta[6,1] beta[6,2] beta[6,3] beta[6,4]
  1.57025   0.83020  -0.54226   0.56925  -0.66342   0.06663
beta[7,1] beta[7,2] beta[7,3] beta[7,4] beta[8,1] beta[8,2]
  1.32726   1.10627  -1.97767   0.15275   0.16701   1.49583
beta[8,3] beta[8,4]  beta0[1]  beta0[2]  beta0[3]  beta0[4]
 -0.95685   1.35857   0.14524   1.74142  -0.81314  -1.24719
 deviance
  0.81714
```

From these results in general we can see that many parameters have Z-score close to 0, and this suggests that there are not significant differences between sample means in the first and last part of the chain. Just in the second [[2]] chain there are a couple of them with $|Z > 2|$, for example beta[6,4] and beta0[3].

Going on, **Heidelberger-Welch Diagnostic** is a test with H0 which verifies that the sampled values comes from a stationary distributions. It works in this way: first it is applied to the whole chain, then after discarding the first 10%, 20%, of the chain until

either the null hypothesis is accepted, or 50% of the chain has been discarded. If there are a lot of “failures” of the test, then a longer MCMC run is needed. The other test shown is the **Halfwidth Mean Test** and it is a test to measure the average length of an interval of the parameters, to evaluate the convergence. However, we take in consideration only the Stationarity tests. The results are shown below:

```
# Heidelberger-Welch diagnostic
(heidel.diag.jags1 <- heidel.diag(mcmc1.final))
```

```
[[1]]

          Stationarity start     p-value
          test         iteration
beta[1,1] passed         1       0.3592
beta[1,2] passed         1       0.3972
beta[1,3] passed         1       0.0828
beta[1,4] passed         1       0.6928
beta[2,1] passed         1       0.3606
beta[2,2] passed         1       0.3503
beta[2,3] passed         1       0.5997
beta[2,4] passed         1       0.2215
beta[3,1] passed         1       0.7631
beta[3,2] passed       101       0.0655
beta[3,3] passed       101       0.1944
beta[3,4] passed         1       0.4024
beta[4,1] passed         1       0.9025
beta[4,2] passed         1       0.0988
beta[4,3] passed         1       0.7360
beta[4,4] passed         1       0.0654
beta[5,1] passed         1       0.2394
beta[5,2] passed         1       0.5696
beta[5,3] passed         1       0.6778
beta[5,4] passed         1       0.4062
beta[6,1] passed         1       0.2740
beta[6,2] passed         1       0.6877
beta[6,3] passed         1       0.5436
beta[6,4] passed         1       0.3403
beta[7,1] passed         1       0.1760
beta[7,2] passed         1       0.3068
beta[7,3] passed         1       0.6451
beta[7,4] passed         1       0.8744
beta[8,1] passed         1       0.8845
beta[8,2] passed         1       0.2273
beta[8,3] passed         1       0.0822
beta[8,4] passed         1       0.2523
beta0[1]  passed         1       0.9000
beta0[2]  passed         1       0.4776
beta0[3]  passed         1       0.5893
beta0[4]  passed         1       0.1045
deviance  passed         1       0.1728

          Halfwidth Mean    Halfwidth
          test
beta[1,1] passed    -3.2893 0.170
beta[1,2] failed    -1.8563 0.197
beta[1,3] failed     1.6555 0.173
beta[1,4] passed     3.7479 0.167
beta[2,1] passed    -4.3606 0.187
beta[2,2] passed     3.3071 0.182
beta[2,3] passed    -3.2516 0.170
beta[2,4] passed     4.0670 0.188
beta[3,1] failed    -0.8697 0.174
beta[3,2] passed     3.2048 0.175
beta[3,3] passed    -4.6624 0.200
beta[3,4] passed     2.0994 0.188
beta[4,1] passed     3.8788 0.178
beta[4,2] failed    -0.5786 0.169
beta[4,3] failed    -1.4708 0.187
beta[4,4] passed    -2.1020 0.191
beta[5,1] failed     0.1724 0.163
beta[5,2] passed     1.9339 0.149
beta[5,3] passed    -2.4348 0.170
beta[5,4] failed     0.3811 0.172
beta[6,1] failed     1.5085 0.187
beta[6,2] passed     2.6813 0.183
beta[6,3] failed    -1.1983 0.214
beta[6,4] passed    -2.9965 0.209
```

```
beta[7,1] passed     2.1387 0.166
beta[7,2] failed     0.9898 0.166
beta[7,3] failed    -0.1579 0.196
beta[7,4] passed    -3.2239 0.222
beta[8,1] failed     0.5305 0.156
beta[8,2] failed    -1.2822 0.164
beta[8,3] failed     0.8078 0.199
beta[8,4] failed    -0.0404 0.180
beta0[1]  failed     0.5586 0.178
beta0[2]  failed     0.0446 0.180
beta0[3]  failed    -0.8796 0.228
beta0[4]  failed    -0.3856 0.205
deviance  passed     3.6106 0.158

[[2]]

          Stationarity start     p-value
          test         iteration
beta[1,1] passed         1       0.37358
beta[1,2] passed         1       0.49539
beta[1,3] passed         1       0.18965
beta[1,4] passed         1       0.84905
beta[2,1] passed         1       0.45314
beta[2,2] passed         1       0.06250
beta[2,3] passed         1       0.45776
beta[2,4] passed         1       0.89438
beta[3,1] passed         1       0.82484
beta[3,2] passed         1       0.40191
beta[3,3] passed         1       0.56385
beta[3,4] passed         1       0.34661
beta[4,1] passed         1       0.23879
beta[4,2] passed         1       0.21679
beta[4,3] failed        NA       0.00767
beta[4,4] passed         1       0.35895
beta[5,1] passed         1       0.50540
beta[5,2] passed         1       0.58007
beta[5,3] passed         1       0.17846
beta[5,4] passed         1       0.40195
beta[6,1] passed         1       0.20637
beta[6,2] passed         1       0.91373
beta[6,3] passed         1       0.98100
beta[6,4] passed         1       0.98571
beta[7,1] passed         1       0.41261
beta[7,2] passed         1       0.57684
beta[7,3] passed         1       0.27984
beta[7,4] passed         1       0.42234
beta[8,1] passed         1       0.60824
beta[8,2] passed       301       0.16422
beta[8,3] passed         1       0.06467
beta[8,4] passed         1       0.55690
beta0[1]  passed         1       0.21075
beta0[2]  passed         1       0.20252
beta0[3]  passed         1       0.54705
beta0[4]  passed         1       0.08062
deviance  passed         1       0.60823

          Halfwidth Mean    Halfwidth
          test
beta[1,1] passed    -3.3773 0.172
beta[1,2] passed    -1.8959 0.167
beta[1,3] failed     1.4830 0.184
beta[1,4] passed     3.6494 0.178
beta[2,1] passed    -4.1511 0.175
beta[2,2] passed     3.3464 0.160
beta[2,3] passed    -2.7991 0.170
beta[2,4] passed     4.1511 0.184
beta[3,1] failed    -1.1535 0.164
beta[3,2] passed     3.4739 0.168
beta[3,3] passed    -4.5965 0.184
```

```
beta[3,4] passed     2.2549 0.183
beta[4,1] passed     3.7820 0.180
beta[4,2] failed    -0.4377 0.167
beta[4,3] <NA>           NA    NA
beta[4,4] passed    -2.2042 0.198
beta[5,1] failed     0.2173 0.158
beta[5,2] passed     1.9395 0.153
beta[5,3] passed    -2.4967 0.170
beta[5,4] failed     0.2516 0.151
beta[6,1] failed     1.4694 0.186
beta[6,2] passed     2.7463 0.206
beta[6,3] failed    -1.4173 0.194
beta[6,4] passed    -3.1248 0.194
beta[7,1] passed     2.2730 0.164
beta[7,2] failed     0.8619 0.162
beta[7,3] failed    -0.1084 0.189
beta[7,4] passed    -3.0217 0.212
beta[8,1] failed     0.7297 0.151
beta[8,2] failed    -1.2421 0.201
beta[8,3] failed     0.7157 0.165
beta[8,4] failed     0.0571 0.173
beta0[1]  failed     0.7388 0.169
beta0[2]  failed     0.0153 0.174
beta0[3]  failed    -0.5715 0.226
beta0[4]  failed    -0.1940 0.176
deviance  passed     3.5048 0.155

[[3]]

          Stationarity start     p-value
          test         iteration
beta[1,1] passed         1       0.6736
beta[1,2] passed       101       0.0563
beta[1,3] passed         1       0.1109
beta[1,4] passed         1       0.1519
beta[2,1] passed         1       0.2628
beta[2,2] passed         1       0.9089
beta[2,3] passed         1       0.5560
beta[2,4] passed         1       0.9762
beta[3,1] passed         1       0.9429
beta[3,2] passed         1       0.7825
beta[3,3] passed         1       0.8130
beta[3,4] passed         1       0.9223
beta[4,1] failed        NA       0.0254
beta[4,2] passed         1       0.7369
beta[4,3] passed         1       0.2975
beta[4,4] passed         1       0.0614
beta[5,1] passed         1       0.4121
beta[5,2] passed         1       0.8467
beta[5,3] passed         1       0.2514
beta[5,4] passed         1       0.4591
beta[6,1] passed         1       0.5306
beta[6,2] passed         1       0.6785
beta[6,3] passed         1       0.4261
beta[6,4] passed         1       0.8579
beta[7,1] passed         1       0.2008
beta[7,2] passed         1       0.5866
beta[7,3] passed         1       0.2443
beta[7,4] passed         1       0.1721
beta[8,1] passed         1       0.5663
beta[8,2] passed         1       0.8584
beta[8,3] passed         1       0.3247
beta[8,4] passed         1       0.7567
beta0[1]  passed         1       0.2258
beta0[2]  passed         1       0.1774
beta0[3]  passed         1       0.3236
beta0[4]  passed         1       0.6621
deviance  passed         1       0.9284
```

```
          Halfwidth Mean    Halfwidth
          test
beta[1,1] passed    -3.3300 0.181
beta[1,2] failed    -1.7183 0.197
beta[1,3] failed     1.5548 0.179
beta[1,4] passed     3.8049 0.179
beta[2,1] passed    -4.2846 0.187
beta[2,2] passed     3.5893 0.171
beta[2,3] passed    -3.0993 0.170
beta[2,4] passed     4.0932 0.181
beta[3,1] failed    -1.0464 0.169
beta[3,2] passed     3.3297 0.202
beta[3,3] passed    -4.3989 0.190
beta[3,4] passed     2.3875 0.182
beta[4,1] <NA>           NA    NA
beta[4,2] failed    -0.3817 0.171
beta[4,3] failed    -1.4127 0.183
beta[4,4] passed    -2.0760 0.199
beta[5,1] failed     0.2286 0.163
beta[5,2] passed     1.8293 0.158
beta[5,3] passed    -2.3955 0.162
beta[5,4] failed     0.4643 0.155
beta[6,1] failed     1.7157 0.194
beta[6,2] passed     2.6140 0.188
beta[6,3] failed    -1.5797 0.206
beta[6,4] passed    -3.0578 0.216
beta[7,1] passed     2.1737 0.164
beta[7,2] failed     0.9385 0.162
beta[7,3] failed    -0.1513 0.212
beta[7,4] passed    -3.0998 0.194
beta[8,1] failed     0.6772 0.162
beta[8,2] failed    -1.3198 0.163
beta[8,3] failed     0.6979 0.152
beta[8,4] failed    -0.1362 0.171
beta0[1]  failed     0.6988 0.167
beta0[2]  failed     0.2106 0.162
beta0[3]  failed    -0.6688 0.232
beta0[4]  failed     0.0511 0.198
deviance  passed     3.5229 0.160
```

As we can see, almost all passed the stationarity test, beside parameter beta[8,3] in the third chain.

The last diagnostic applied is **Raftery-Lewis** technique, used to calculate the necessary number of sampling MCMC to guarantee reliable estimates, and for the chain to be sufficiently convergent and representative of the posterior distribution. The idea is that if, for a specific parameter of interest $q$, we define an acceptable tolerance $r$ and a probability $s$ of being within that tolerance, this diagnostic will calculate the number of iterations $N$ and the number of burn-ins $M$ necessary to satisfy the specified conditions. This calculation is made with a 5% percentage of acceptable error, and

$$N \geq \frac{1}{(\hat{\sigma}^2 - \epsilon^2)}(\frac{1}{\epsilon}(\frac{z_{\alpha/2}}{\sigma}))^2$$

where $\hat{\sigma}$ is the estimate of the standard deviation of the statistic of interest, $\sigma$ is its exact standard deviation, $\epsilon$ is the acceptable error and $z_{\alpha/2}$ is the quantile of the normal distribution.

The results for the First Bayesian model are shown below and for an accuracy of 0.005 and a percentage of error of 5%, we need at least 3746 iterations for each chain.

```
# Raftery-Lewis Diagnostic diagnostic
(raftery.diag.jags1 <- raftery.diag(mcmc1.final))
```

```
[[1]]

Quantile (q) = 0.025
Accuracy (r) = +/- 0.005
Probability (s) = 0.95

You need a sample size of at least 3746 with these values of q, r and s

[[2]]

Quantile (q) = 0.025
Accuracy (r) = +/- 0.005
Probability (s) = 0.95

You need a sample size of at least 3746 with these values of q, r and s

[[3]]

Quantile (q) = 0.025
Accuracy (r) = +/- 0.005
Probability (s) = 0.95

You need a sample size of at least 3746 with these values of q, r and s
```

# 4. Second Bayesian model: Multilevel model with Hierarchical Random Terms

## 4.1. Formulation of the model

For the second model, we can formulate a two-levels model. The proposed one is a **Multilevel model with Hierarchical Random Terms**. It is basically a hierarchical multinomial regression model and, as before, it predicts a categorical response variable with $K$ categories, while also accounting for unobserved variability at different hierarchical levels through random effects. These two levels are the environment (also called greenhouse) and the varieties. The target variable, as in the previous Bayesian model, is $Y_i = (Y_{i1}, \ldots, Y_{iK})$ where $i$ is every plant observed. The model, again, can be expressed in the following way:

$$Y_i \sim multinomial(\pi_i, N_i)$$

with $\sum_{k=1}^{K} \pi_{ik} = 1$ and $\pi_{ik}$ the probability that plant $i$ belongs to category $k$. As in Multinomial logistic regression model, it is assumed that the probabilities of belonging to each category are determined by a linear combination of independent random variables $x_{ip}$ with $p = 1, \ldots, P$ and, for this case, also random effects at two levels (i.e. for environment and for variety). Hence, in this model we have that:

$$\log(\frac{\pi_{ik}}{\pi_{i1}}) = \eta_{ik} = \sum_{i=1}^{P} \beta_{pk} x_{ip} + u_{env[j],k} + u_{variety[j],k}, \quad (4.1)$$

where: - $\sum_{i=1}^{P} \beta_{pk} x_{ip}$ is the effect of the predictors weighted by their parameters, - $u_{env[j],k}$ is the random effect of the environment $j$ for the class $k$ and it captures the variability between greenhouses, - $u_{variety[j],k}$ is the random effect of the variety $j$ and it captures the variability between plant's varieties.

Through the softmax function we have that:

$$\pi_{ik} = \frac{\exp(\eta_{ik})}{\sum_{k=1}^{K} \exp(\eta_{ik})}.$$

Moreover, the random effect for environment and for variety are assumed to follow Normal distributions with mean zero and unknown variances such that $u_{\text{env}[j],k} \sim N(0, \tau_{\text{env}})$ and $u_{\text{variety}[j],k} \sim N(0, \tau_{\text{variety}})$.

## 4.2. Application of the model

The model explained in section 4.1. can be implemented in JAGS in the following way:

```
\begin{verbatim}
model {
  for (i in 1:N) {
    y[i] ~ dcat(pi[i, 1:K])

    #Likelihood
    for (k in 1:K) {
      eta[i, k] <- beta[1, k] * ACHP[i] +
                   beta[2, k] * PHR[i] +
                   beta[3, k] * ALAP[i] +
                   beta[4, k] * ANPL[i] +
                   beta[5, k] * PDMVG[i] +
                   beta[6, k] * ARL[i] +
                   beta[7, k] * AWWR[i] +
                   beta[8, k] * PDMRG[i] +
                   u_env[environment[i], k]  #Random effect for environment
                  + u_variety[variety[i], k]  #Random effect for variety
      expeta[i, k] <- exp(eta[i, k])
    }

    for (k in 1:K) {
      pi[i, k] <- expeta[i, k] / sum(expeta[i, ])
    }
  }

  #Priors (beta for each predictor p and class k)
  for (p in 1:8) {  #8 predictors
    for (k in 1:K) {
      beta[p, k] ~ dnorm(0, 0.1)
    }
  }

  #Random effect for environment
  for (j in 1:2) {  #2 environments: S=1, T=0
    for (k in 1:K) {
      u_env[j, k] ~ dnorm(0, tau_env)
    }
  }
  tau_env ~ dgamma(0.1, 0.1)

  #Random effect for variety
  for (j in 1:2) {  #2 varieties: A, B
    for (k in 1:K) {
      u_variety[j, k] ~ dnorm(0, tau_variety)
    }
  }
  tau_variety ~ dgamma(0.1, 0.1)
}
\end{verbatim}
```

and the commands in R are the following:

```
# Second Bayesian model: Multilevel model with
# Hierarchical Random Terms
y <- as.numeric(data.sampled$Class)
K <- length(unique(y))  #4, number of classes

# MCMC
S <- 10000
burn_in <- 2000

data.prepared <- list(y = y, ACHP = sampled_standardized_predictors$`Average of chlorophyll in the plant (ACHP)`,
    PHR = sampled_standardized_predictors$`Plant height  rate (PHR)`,
    ALAP = sampled_standardized_predictors$`Average leaf area of the plant (ALAP)`,
    ANPL = sampled_standardized_predictors$`Average number of plant leaves (ANPL)`,
    PDMVG = sampled_standardized_predictors$`Percentage of dry matter for vegetative growth (PDMVG)`,
    ARL = sampled_standardized_predictors$`Average root length (ARL)`,
    AWWR = sampled_standardized_predictors$`Average wet weight of the root (AWWR)`,
    PDMRG = sampled_standardized_predictors$`Percentage of dry matter for root growth (PDMRG)`,
    N = nrow(sampled_standardized_predictors), K = K,
    environment = as.numeric(data.sampled$Env), variety = as.numeric(data.sampled$Variety))

params <- c("beta", "u_env", "u_variety", "tau_env",
    "tau_variety")

inits <- list(inits1 = list(beta = matrix(0, 8, K),
    u_env = matrix(0, 2, K), u_variety = matrix(0,
        2, K), tau_env = 1, tau_variety = 1), inits2 = list(beta = matrix(1,
    8, K), u_env = matrix(1, 2, K), u_variety = matrix(1,
    2, K), tau_env = 1, tau_variety = 1), inits3 = list(beta = matrix(-0.5,
    8, K), u_env = matrix(-0.5, 2, K), u_variety = matrix(-0.5,
    2, K), tau_env = 1, tau_variety = 1))

start.time2 <- Sys.time()

jags.2 <- jags(data = data.prepared, inits = inits,
    parameters.to.save = params, model.file = "C:\\Users\\sofyc\\OneDrive\\Desktop\\SAPIENZA\\SDS II\\project\\mod
_agr_multilevel_random.txt",
    n.chains = 3, n.iter = S, n.burnin = burn_in)
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 600
   Unobserved stochastic nodes: 50
   Total graph size: 34854

Initializing model
```

```
end.time2 <- Sys.time()
```

# 4.3. Convergence diagnostics and inferential findings

The time execution of the second Bayesian model is the following:

```
Time difference of 23.41539 mins
```

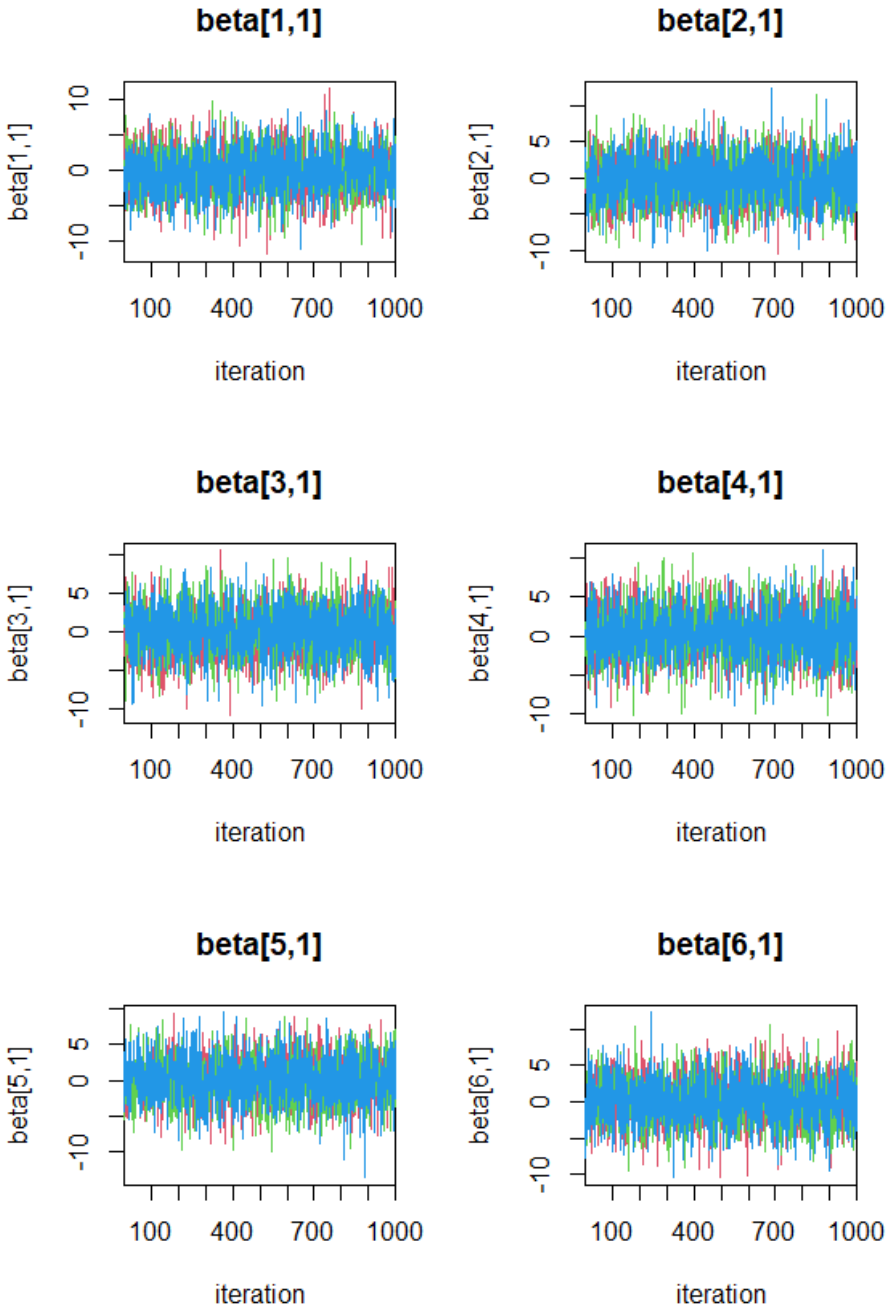
Hence, it is higher than the one of the first Bayesian model applied before, but it is necessary to precise that this timing is actually a bit variable, in fact in some cases the second model performs better, while in others is the first one, and the difference is just of 3 or 4 min-sec maximum.

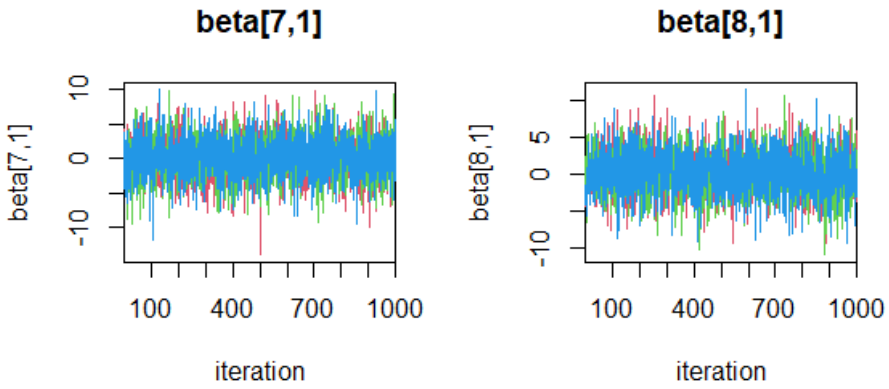
```
Time difference of 4.768093 mins
```

As before, we plot the traceplots and the colors of the chains are still the same: red for the first one, green for the second one and light blue for the third one. As we can see from Figure 13, beta parameters seem to mix very well. Regarding instead the parameters u_env and u_variety, we can see that they mix acceptably in some cases, and in others not so well as for example in u_env[2,1] or

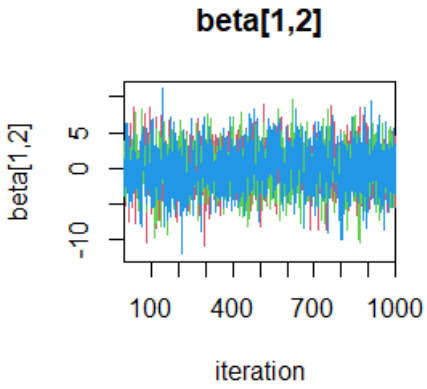
u_env[2,3], u_variety[2,2]. As we can see, tau_env seems almost always 0 apart from two groups of iterations. Regarding tau_variety instead, we can see that it captures more variability, compared to tau_env.

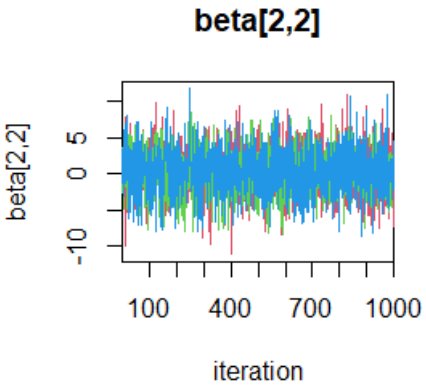
```
# Summary trace plots parameters (beta for each
# class)
traceplot(jags.2, varname = c("beta"), ask = FALSE,
    col = c(2, 3, 4), match.head = FALSE)
# COLORS: 2=red, 3=green, 4=light blue
```

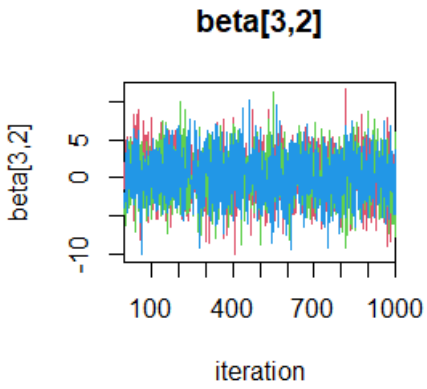


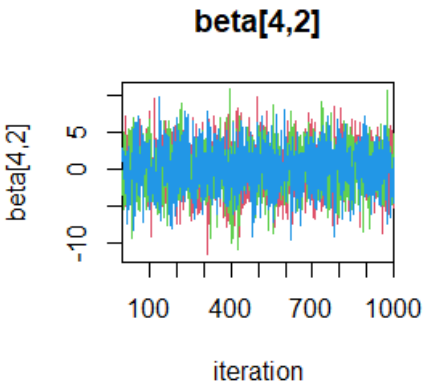

beta[7,1]
beta[7,1]
10
0
-10
100
400
700
1000
iteration

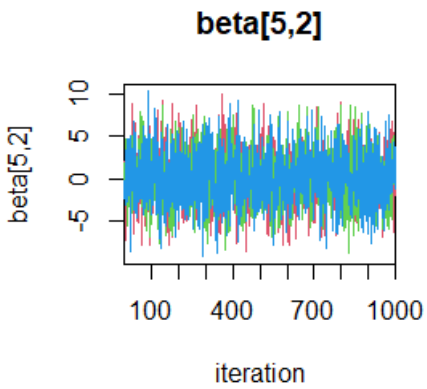

beta[8,1]
beta[8,1]
5
0
-10
100
400
700
1000
iteration

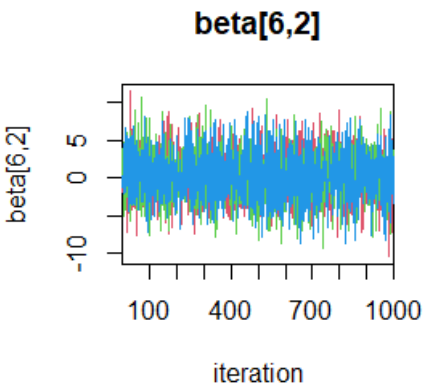

beta[1,2]
beta[1,2]
5
0
-10
100
400
700
1000
iteration

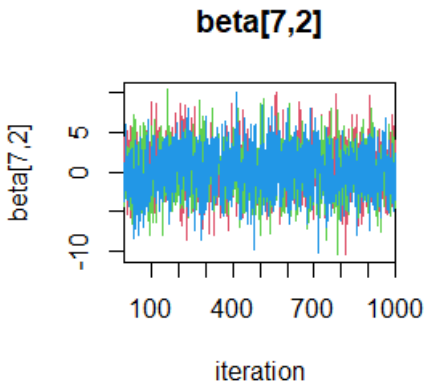

beta[2,2]
beta[2,2]
5
0
-10
100
400
700
1000
iteration

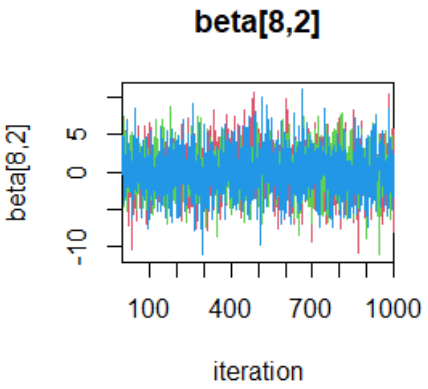

beta[3,2]
beta[3,2]
5
0
-10
100
400
700
1000
iteration

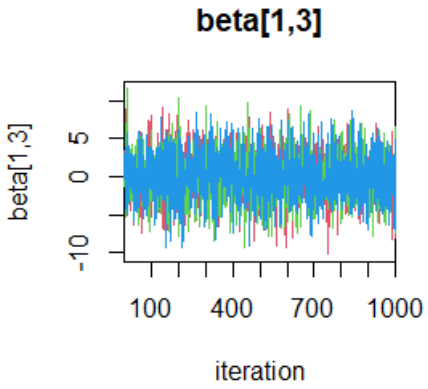

beta[4,2]
beta[4,2]
5
0
-10
100
400
700
1000
iteration

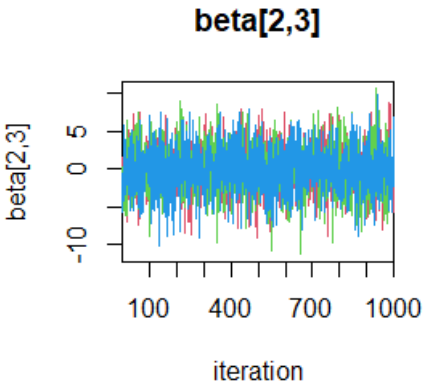

beta[5,2]
beta[5,2]
10
5
0
-5
100
400
700
1000
iteration

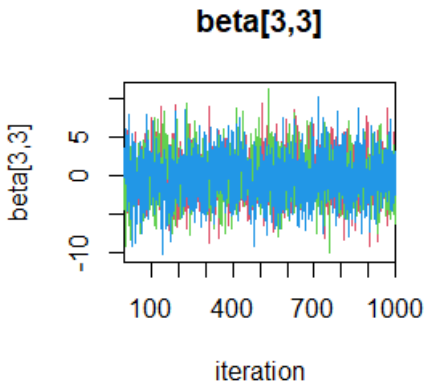

beta[6,2]
beta[6,2]
5
0
-10
100
400
700
1000
iteration

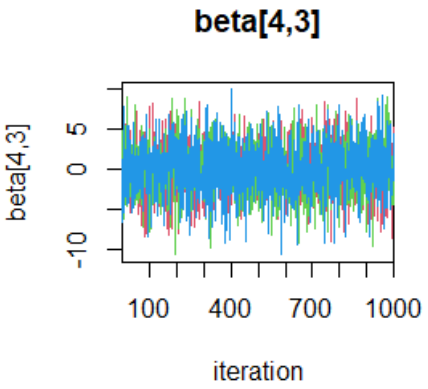

beta[7,2]
beta[7,2]
-10
0
5
100
400
700
1000
iteration

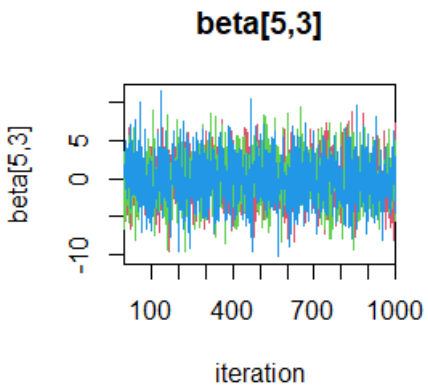

beta[8,2]
beta[8,2]
-10
0
5
100
400
700
1000
iteration

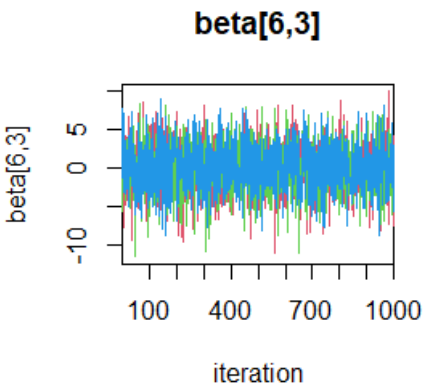

beta[1,3]
beta[1,3]
-10
0
5
100
400
700
1000
iteration

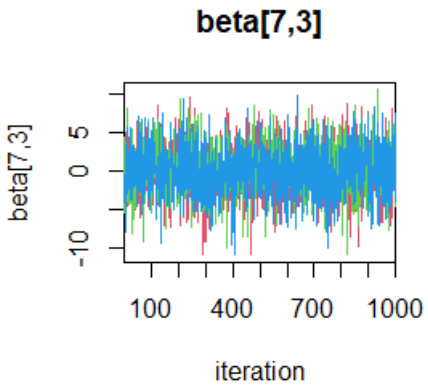

beta[2,3]
beta[2,3]
-10
0
5
100
400
700
1000
iteration

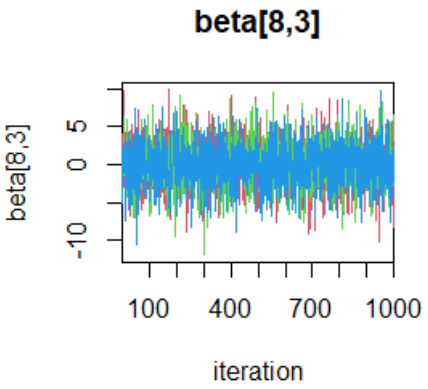

beta[3,3]
beta[3,3]
-10
0
5
100
400
700
1000
iteration

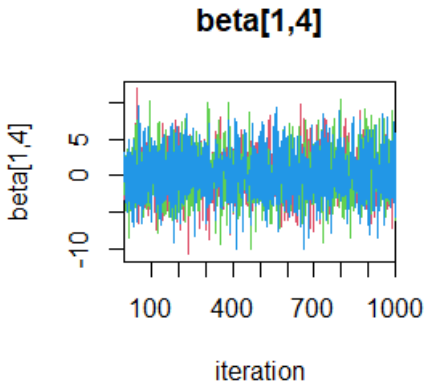

beta[4,3]
beta[4,3]
-10
0
5
100
400
700
1000
iteration

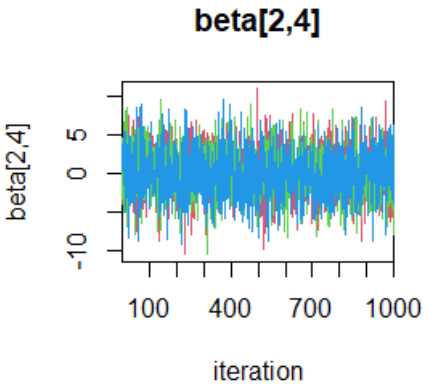

beta[5,3]
beta[5,3]
-10
0
5
100
400
700
1000
iteration

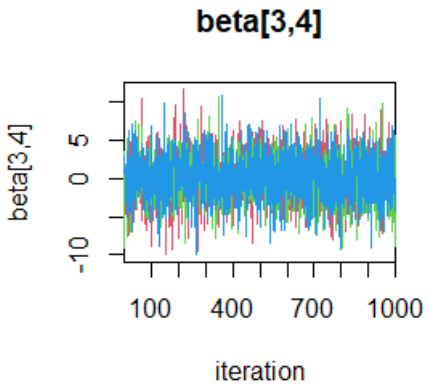

beta[6,3]
beta[6,3]
-10
0
5
100
400
700
1000
iteration

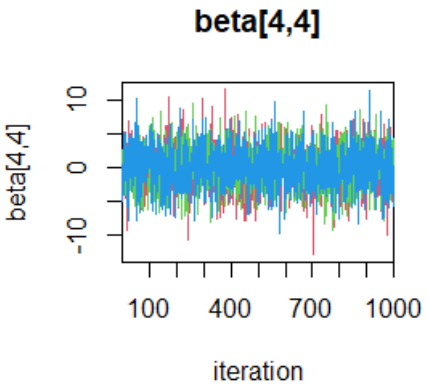

beta[7,3]
beta[7,3]
-10
0
5
100
400
700
1000
iteration

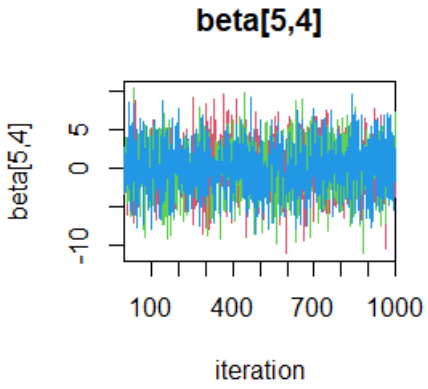

beta[8,3]
beta[8,3]
-10
0
5
100
400
700
1000
iteration

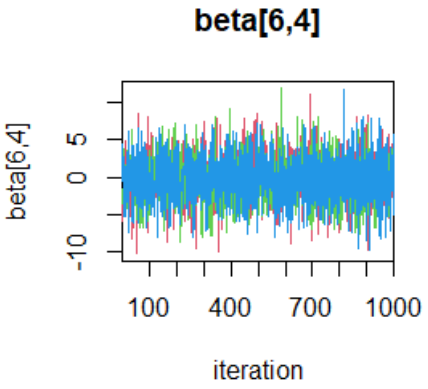

beta[1,4]
beta[1,4]
-10
0
5
100
400
700
1000
iteration

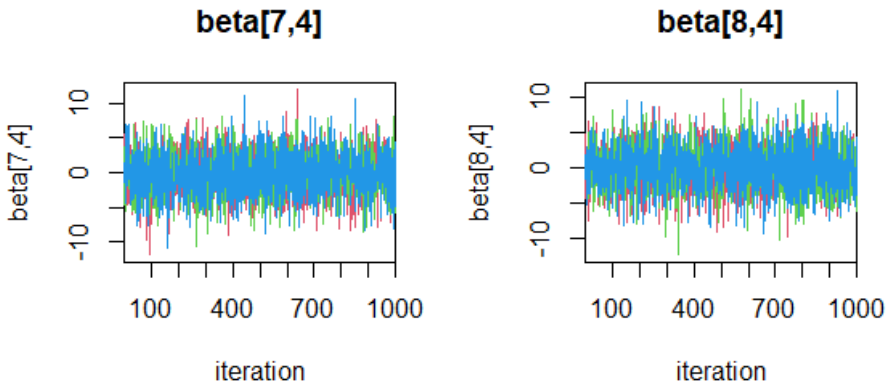

beta[2,4]
beta[2,4]
-10
0
5
100
400
700
1000
iteration


beta[3,4]
beta[3,4]
-10
0
5
100
400
700
1000
iteration


beta[4,4]
beta[4,4]
-10
0
10
100
400
700
1000
iteration

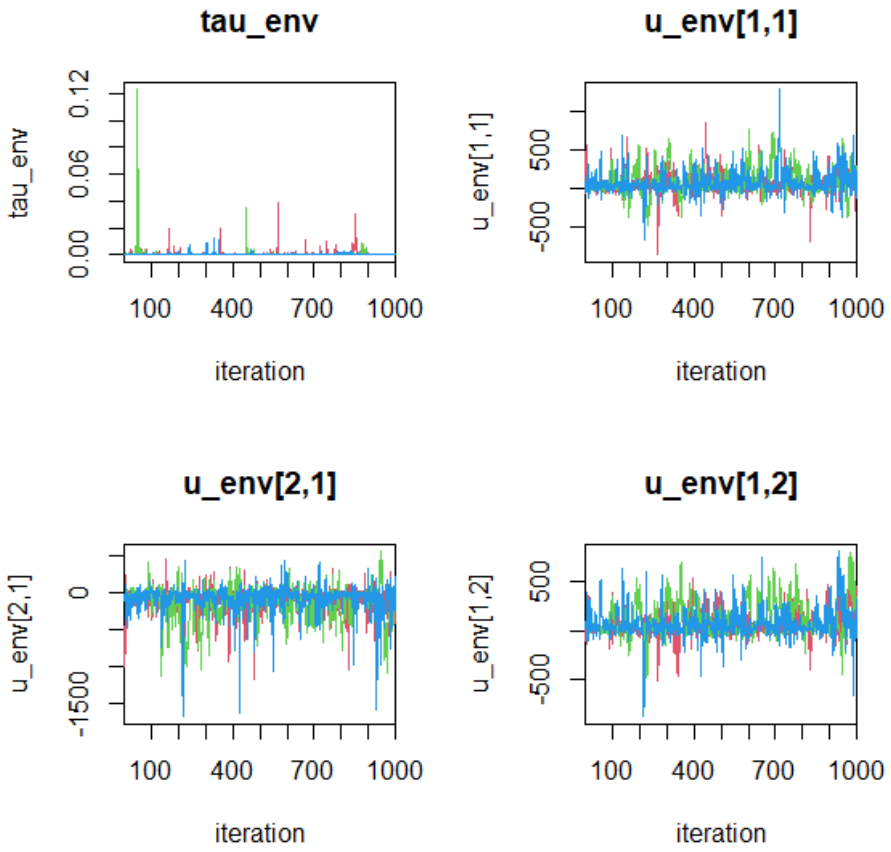

beta[5,4]
beta[5,4]
-10
0
5
100
400
700
1000
iteration

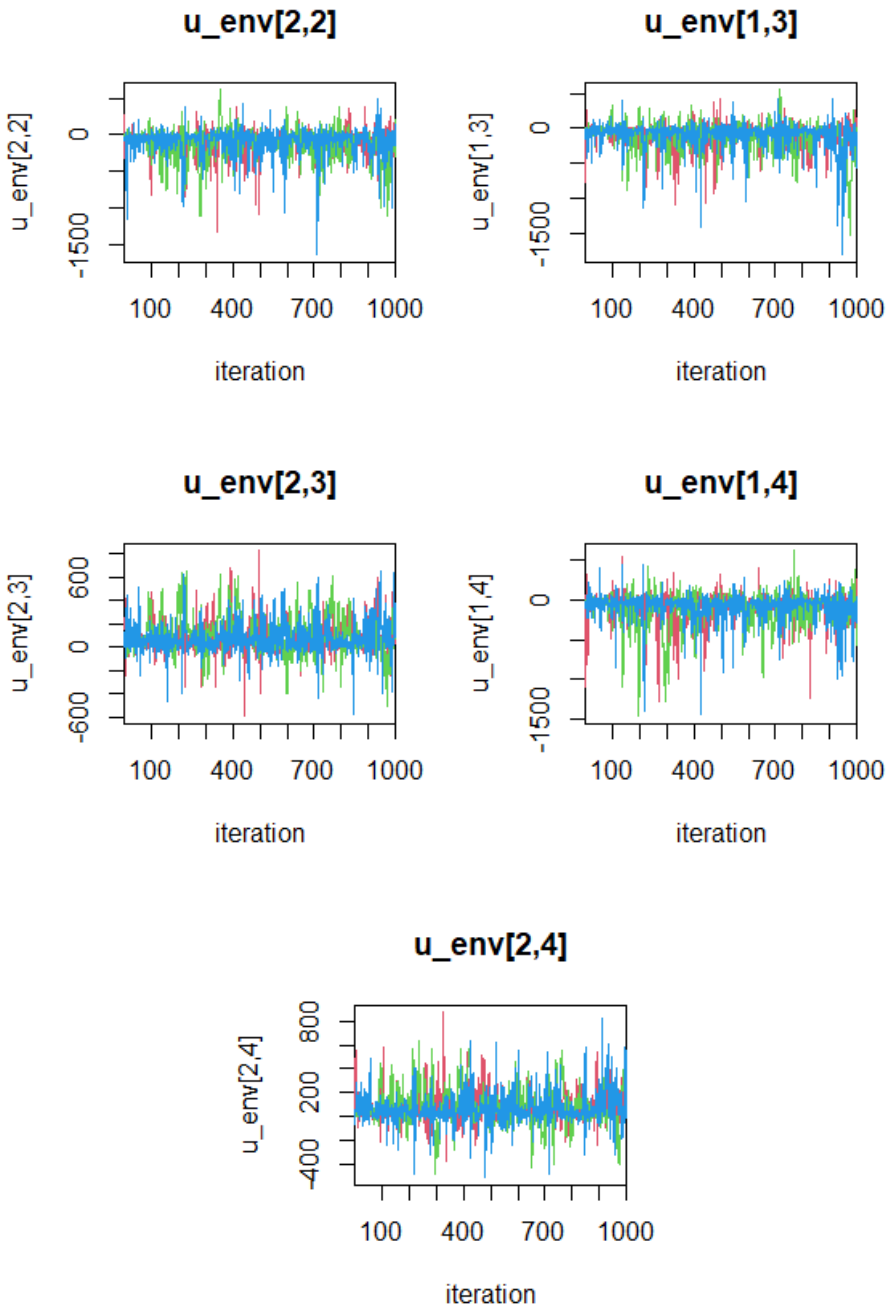

beta[6,4]
beta[6,4]
-10
0
5
100
400
700
1000
iteration




Traceplots, Second model.

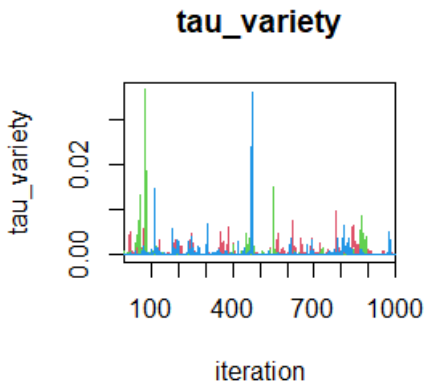
```
# Summary trace plots parameters (u_env for each
# class)
traceplot(jags.2, varname = c("u_env"), ask = FALSE,
    col = c(2, 3, 4), match.head = FALSE)
# COLORS: 2=red, 3=green, 4=light blue
```

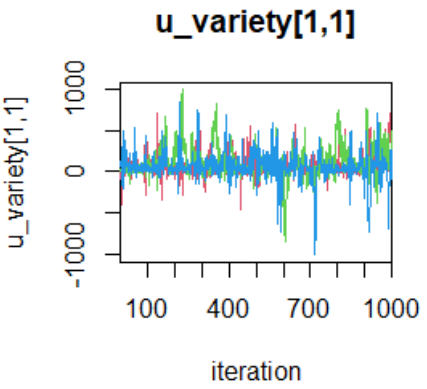


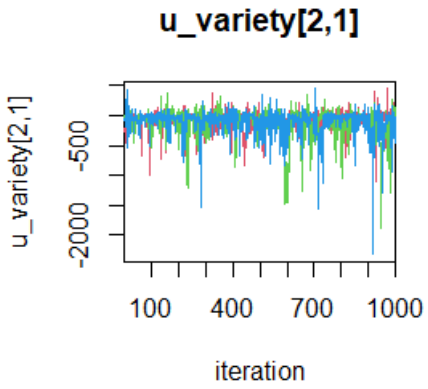


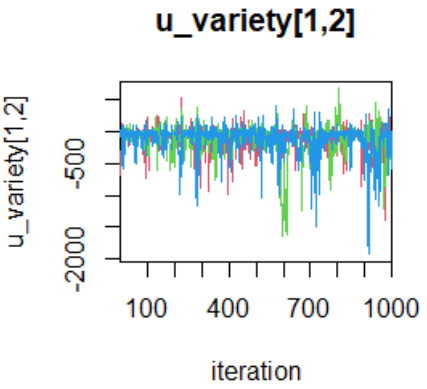

Traceplots u_env, Second model.

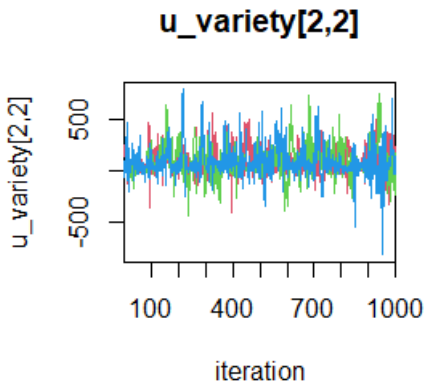
```
# Summary trace plots parameters (u_variety for
# each class)
traceplot(jags.2, varname = c("u_variety"), ask = FALSE,
    col = c(2, 3, 4), match.head = FALSE)
# COLORS: 2=red, 3=green, 4=light blue
```

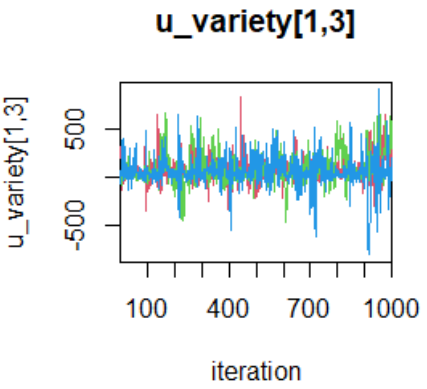

tau_variety
tau_variety
0.00
0.02
100
400
700
1000
iteration

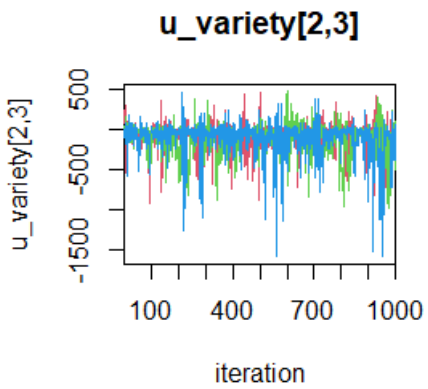

u_variety[1,1]
u_variety[1,1]
-1000
0
1000
100
400
700
1000
iteration

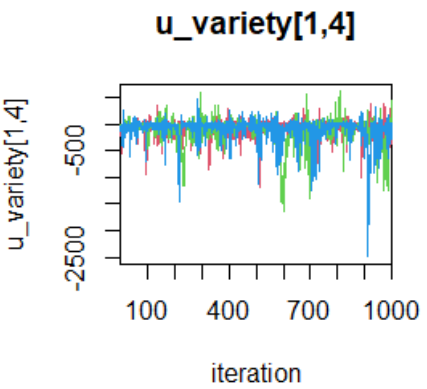

u_variety[2,1]
u_variety[2,1]
-2000
-500
100
400
700
1000
iteration

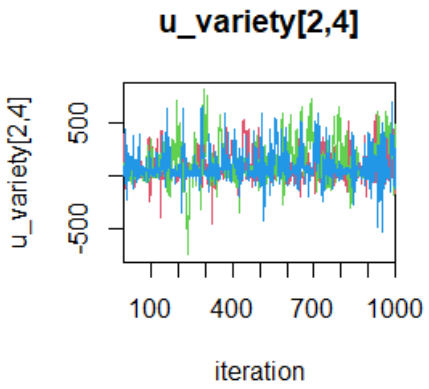

u_variety[1,2]
u_variety[1,2]
-2000
-500
100
400
700
1000
iteration


u_variety[2,2]
u_variety[2,2]
-500
500
100
400
700
1000
iteration


u_variety[1,3]
u_variety[1,3]
-500
500
100
400
700
1000
iteration


u_variety[2,3]
u_variety[2,3]
-1500
-500
500
100
400
700
1000
iteration

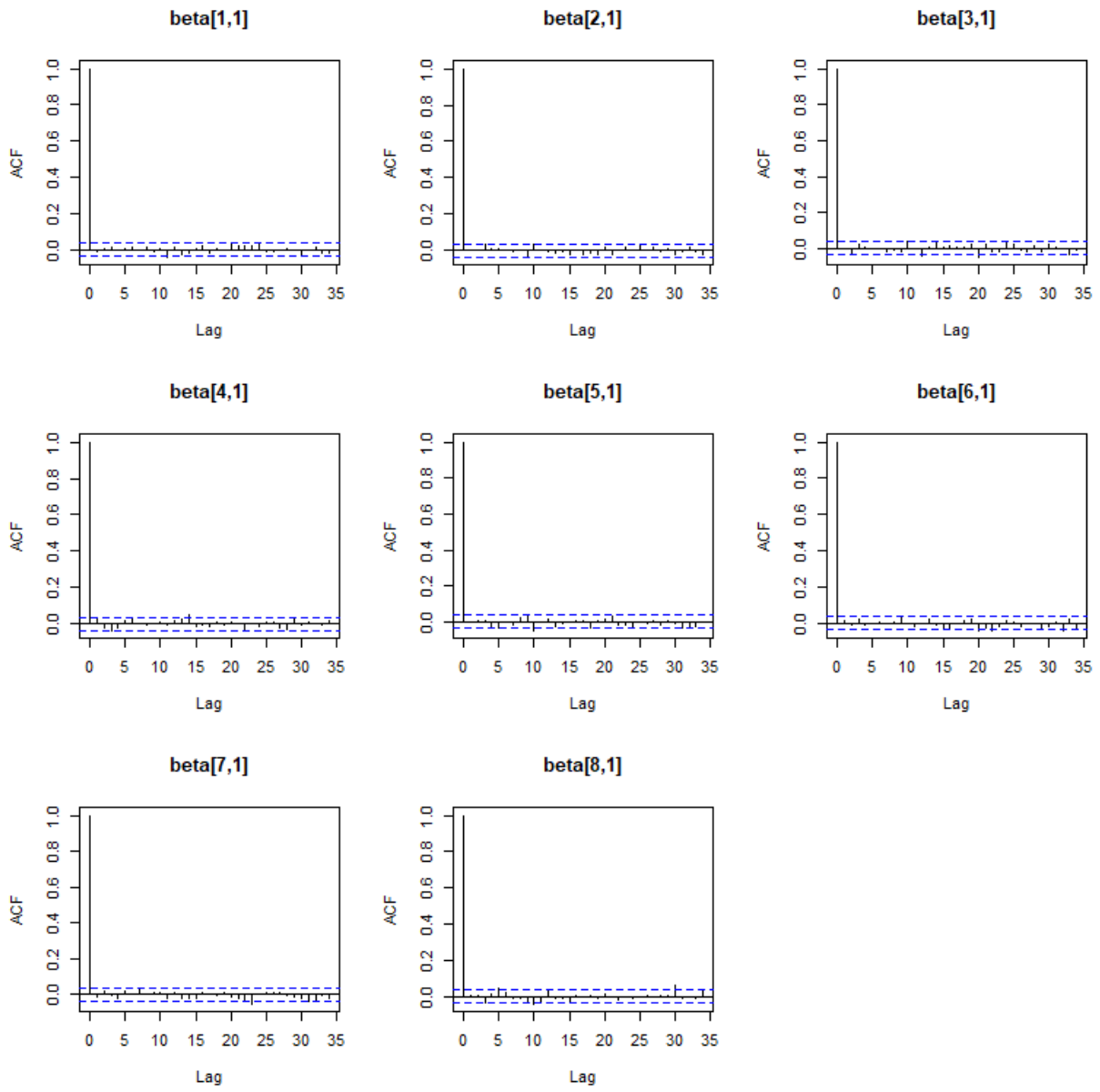

u_variety[1,4]
u_variety[1,4]
-2500
-500
100
400
700
1000
iteration




Traceplots u_variety, Second model.

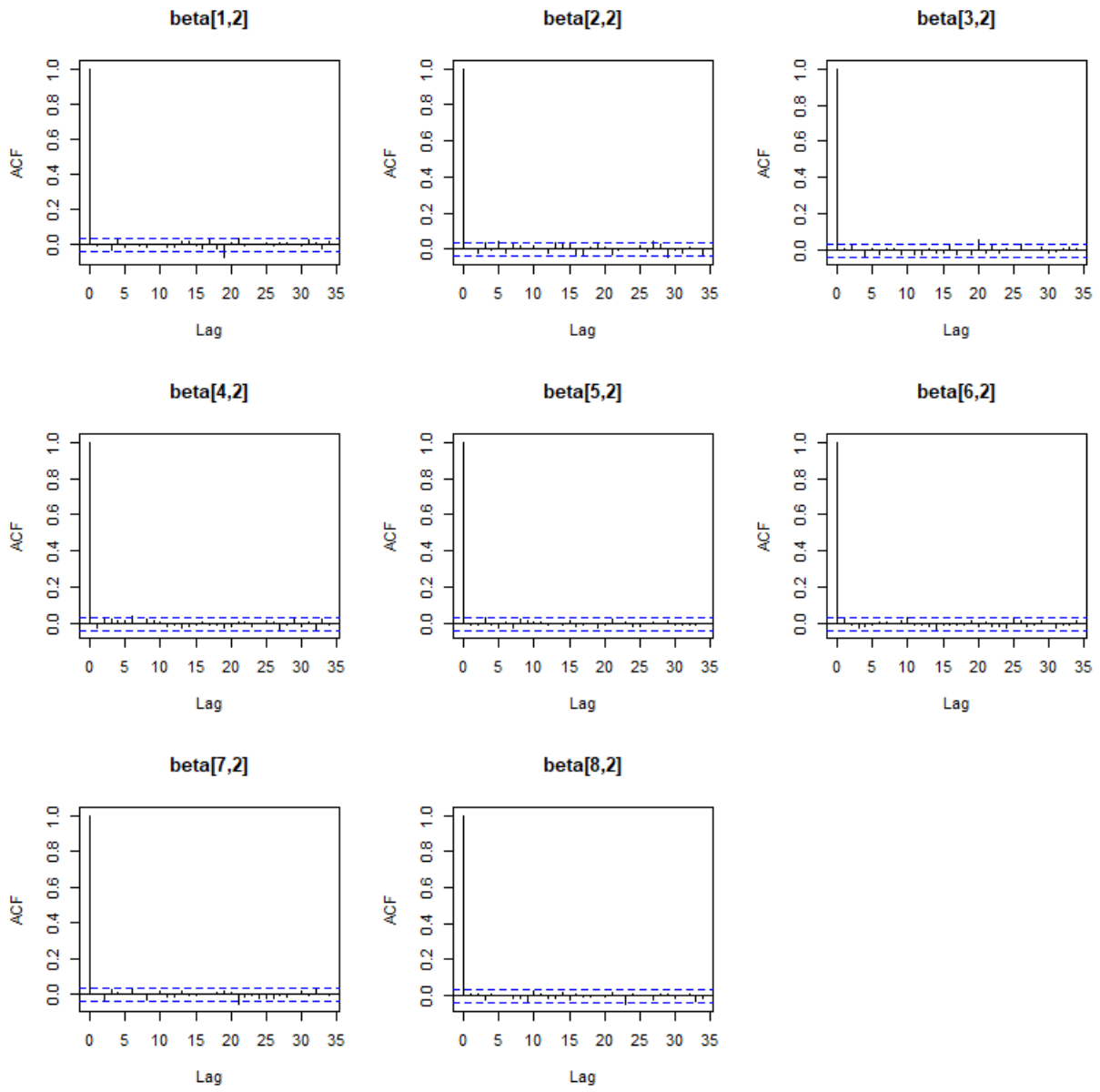
Let's look now at the autocorrelation plots. Figures from 16 to 23 show autocorrelation plots for each parameter of the model: the first four groups of graphs represent the acf for beta parameters for each class, instead the last 2 are for u_env and u_variety, for each class. As the first Bayesian model, also here the autocorrelations are inside the confidence bands, beside some very very slightly exception for a sample like in beta[5,1] (fifth beta for class 1) or u_env[1,3].

```
# Summary plot acf
par(mfrow = c(3, 3))
for (i in 1:8) {
    acf(jags.2$BUGSoutput$sims.matrix[, paste0("beta[",
        i, ",", 1, "]")], main = paste0("beta[", i,
        ",", 1, "]"))
}
par(mfrow = c(1, 1))
```



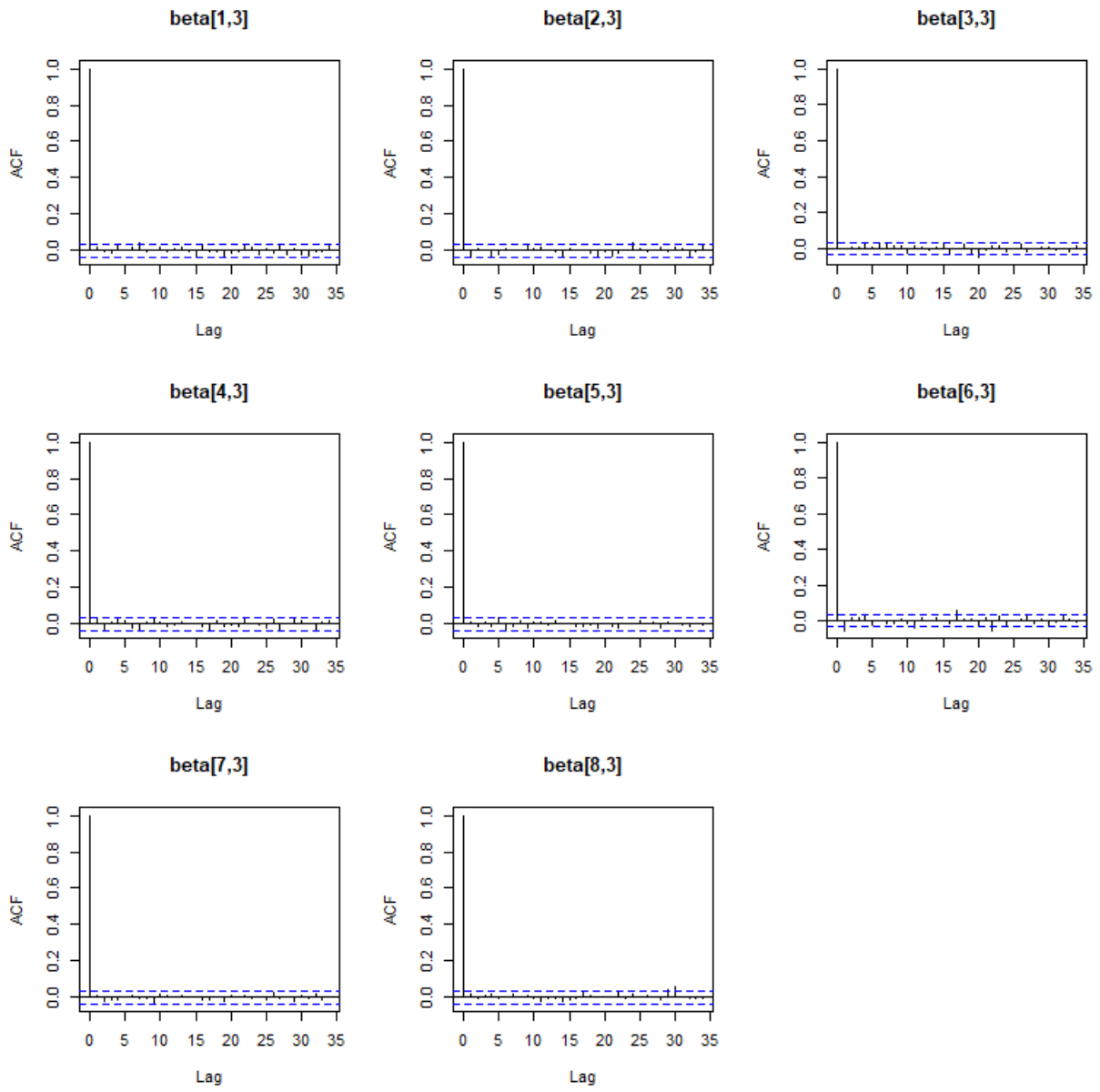

ACF plots beta - First class SA, Second model.

```
# Summary plot acf
par(mfrow = c(3, 3))
for (i in 1:8) {
    acf(jags.2$BUGSoutput$sims.matrix[, paste0("beta[",
        i, ",", 2, "]")], main = paste0("beta[", i,
        ",", 2, "]"))
}
par(mfrow = c(1, 1))
```



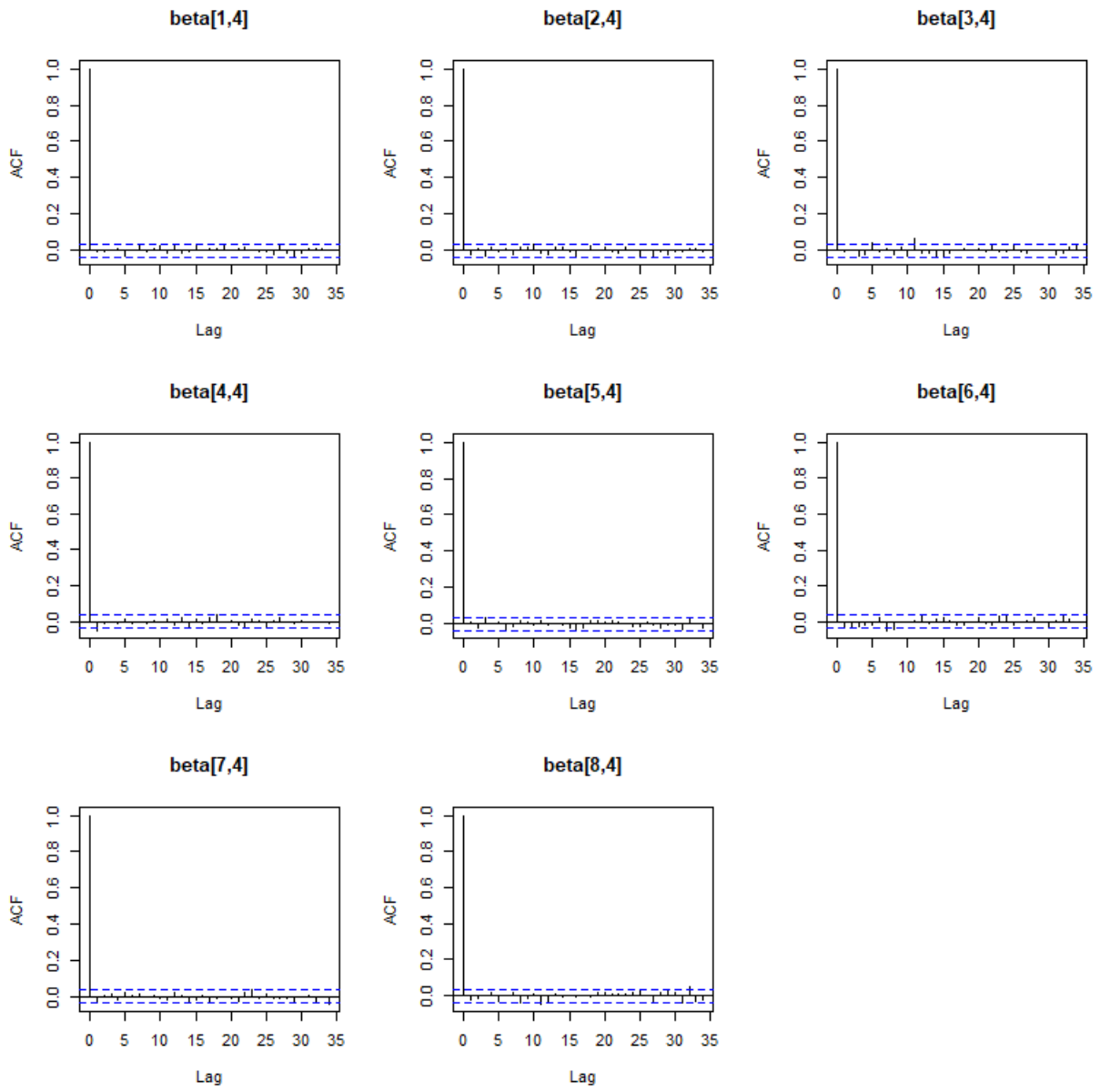

ACF plots beta - Second class SB, Second model.

```
# Summary plot acf
par(mfrow = c(3, 3))
for (i in 1:8) {
    acf(jags.2$BUGSoutput$sims.matrix[, paste0("beta[",
        i, ",", 3, "]")], main = paste0("beta[", i,
        ",", 3, "]"))
}
par(mfrow = c(1, 1))
```



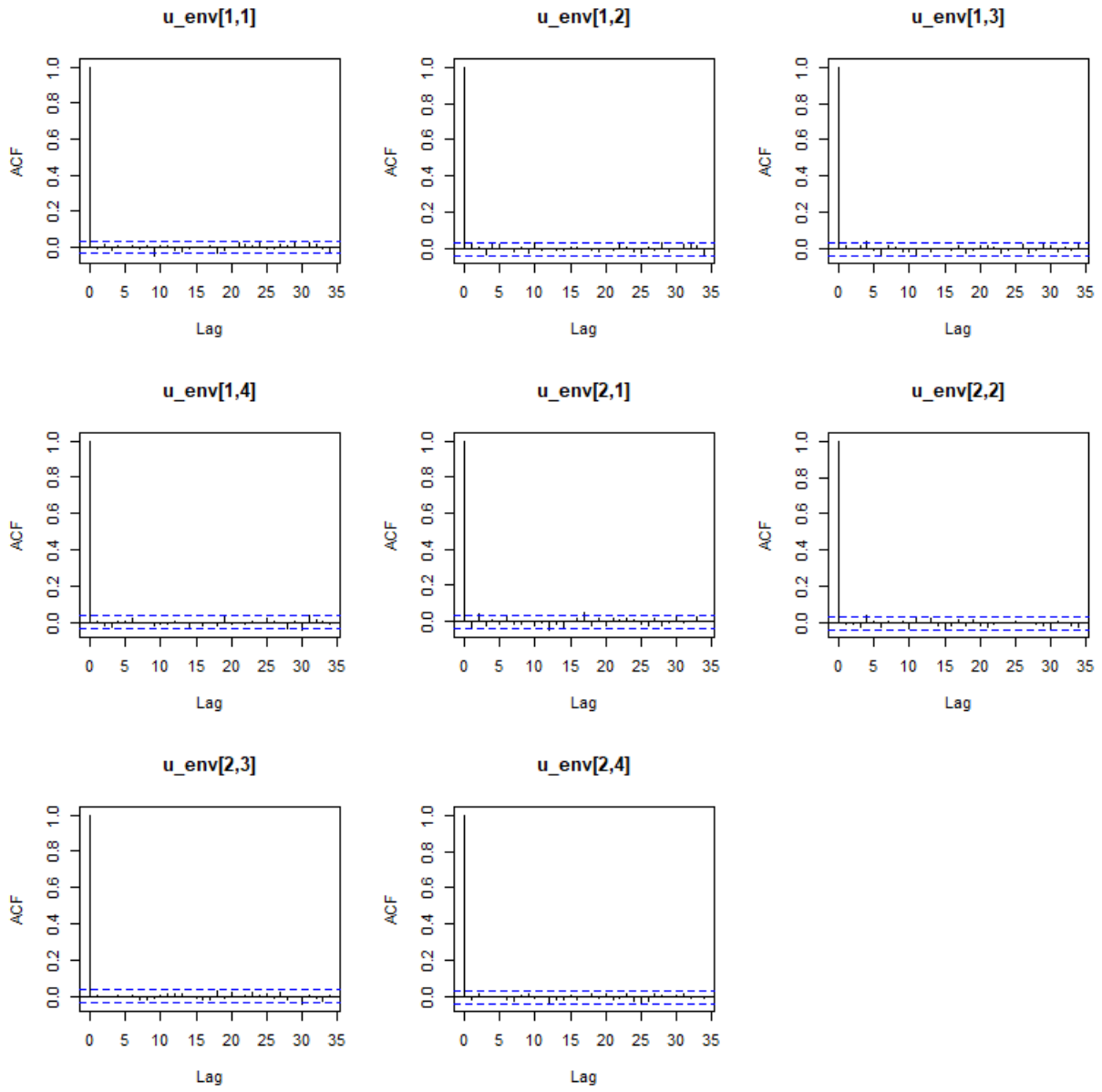

ACF plots beta- Third class TA, Second model.

```
# Summary plot acf
par(mfrow = c(3, 3))
for (i in 1:8) {
    acf(jags.2$BUGSoutput$sims.matrix[, paste0("beta[",
        i, ",", 4, "]")], main = paste0("beta[", i,
        ",", 4, "]"))
}
par(mfrow = c(1, 1))
```



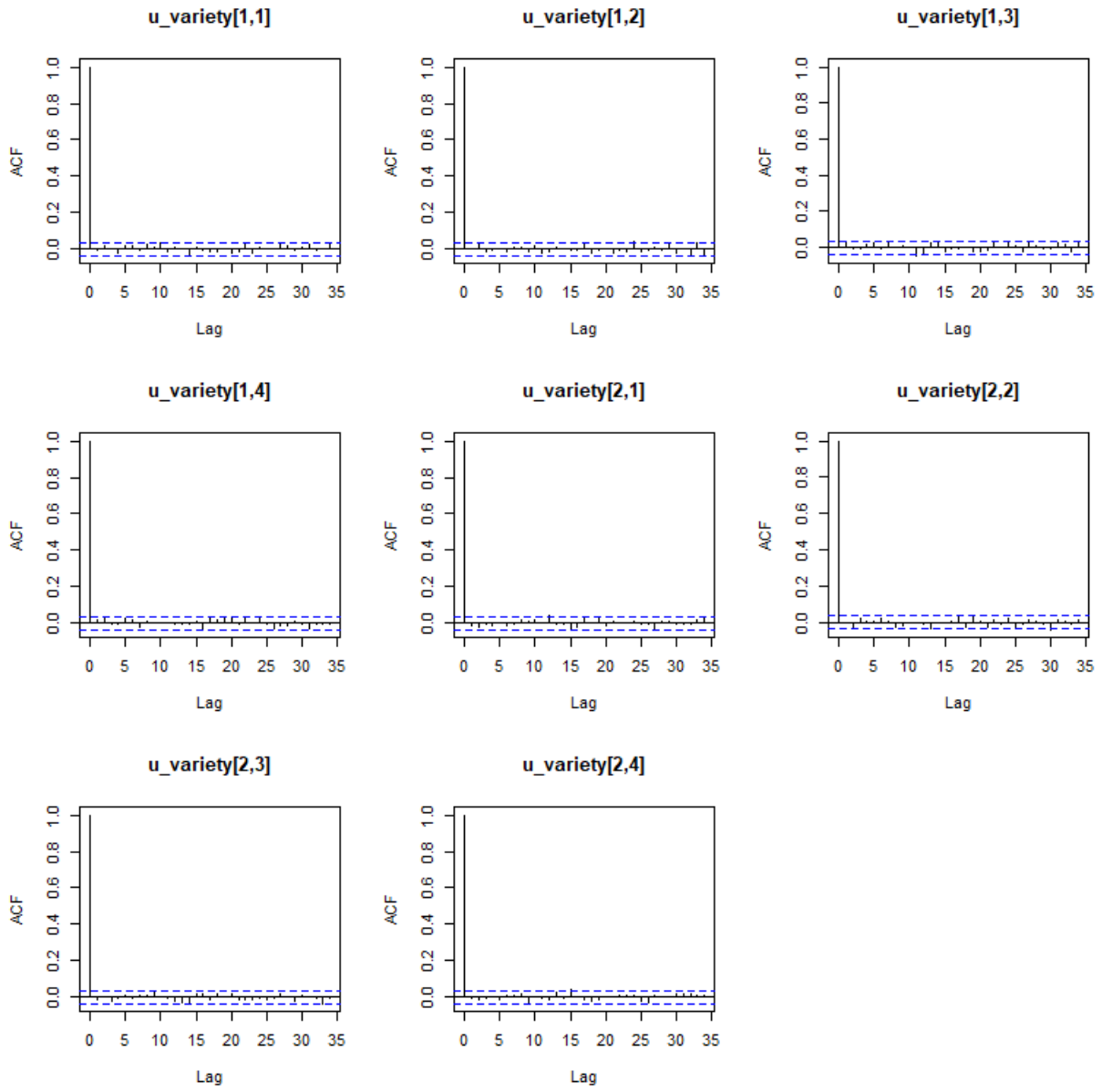

ACF plots beta - Fourth class TB, Second model.

```
# Summary plot acf
par(mfrow = c(3, 3))
for (i in 1:2) {
    for (k in 1:4) {
        acf(jags.2$BUGSoutput$sims.matrix[, paste0("u_env[",
            i, ",", k, "]")], main = paste0("u_env[",
            i, ",", k, "]"))
    }
}
par(mfrow = c(1, 1))
```




ACF plots u_env, Second model.

```
# Summary plot acf
par(mfrow = c(3, 3))
for (i in 1:2) {
    for (k in 1:4) {
        acf(jags.2$BUGSoutput$sims.matrix[, paste0("u_variety[",
            i, ",", k, "]")], main = paste0("u_variety[",
            i, ",", k, "]"))
    }
}
par(mfrow = c(1, 1))
```




ACF plots u_variety, Second model.

Now we can look at the estimates of the parameters of the first model, and analyze its inference. The summaries of the First model are shown in Table 6, 7 and 8. In Table 6 there are the summaries for the parameters *beta* (for every class), in Table 5 for *u_env* (each for every class) and in Table 8 for *u_variety* (each for every class). The intervals for beta show means very close to 0, and in general they have an high level of variability. In this model then, the parameters u_env and u_variety seem to influence the most. Based on the estimates of parameters beta, plants with Average number of plant leaves (ANPL), Percentage of dry matter for vegetative growth (PDMVG), Average root length (ARL) and Average wet weight of the root (AWWR) (respectively beta[4,k], beta[5,k], beta[6,k], beta[7,k]) higher, likely belong to classes 1 or 2 of Smart greenhouse. Instead, as in the first model, we can also observe that plants with high value of Average of chlorophyll in the plant (ACHP), with parameter beta[1,k], are likely to belong to classes 3 (TA) and 4 (TB), i.e. to the Traditional greenhouse.

```
# Extracting estimates and quantiles of beta
summary.jags2.beta <- as.data.frame(jags.2$BUGSoutput$summary[grep("^beta\\[",
    rownames(jags.2$BUGSoutput$summary)), ][, c(1:3,
    5, 7:9)])
summary.jags2.beta <- round(summary.jags2.beta, 2)
# Table beta
kable(summary.jags2.beta, caption = "Second Bayesian model analysis and inference of beta.",
    align = "c")
```

Second Bayesian model analysis and inference of beta.

| | mean | sd | 2.5% | 50% | 97.5% | Rhat | n.eff |
|---|---|---|---|---|---|---|---|
| beta[1,1] | -0.25 | 3.17 | -6.49 | -0.21 | 6.01 | 1 | 3000 |

| | mean | sd | 2.5% | 50% | 97.5% | Rhat | n.eff |
|---|---|---|---|---|---|---|---|
| beta[2,1] | -0.21 | 3.22 | -6.41 | -0.22 | 6.00 | 1 | 3000 |
| beta[3,1] | -0.01 | 3.07 | -6.12 | 0.02 | 6.03 | 1 | 3000 |
| beta[4,1] | 0.29 | 3.21 | -6.08 | 0.33 | 6.79 | 1 | 3000 |
| beta[5,1] | 0.00 | 3.16 | -6.26 | -0.01 | 6.30 | 1 | 3000 |
| beta[6,1] | 0.08 | 3.14 | -6.05 | 0.06 | 6.29 | 1 | 3000 |
| beta[7,1] | 0.04 | 3.10 | -5.94 | 0.07 | 6.19 | 1 | 3000 |
| beta[8,1] | 0.17 | 3.05 | -5.75 | 0.17 | 6.08 | 1 | 2200 |
| beta[1,2] | -0.05 | 3.15 | -6.26 | -0.08 | 6.17 | 1 | 3000 |
| beta[2,2] | 0.21 | 3.20 | -5.83 | 0.22 | 6.41 | 1 | 1200 |
| beta[3,2] | 0.25 | 3.09 | -5.74 | 0.34 | 6.07 | 1 | 3000 |
| beta[4,2] | 0.06 | 3.17 | -6.16 | 0.11 | 6.17 | 1 | 3000 |
| beta[5,2] | 0.15 | 3.08 | -5.93 | 0.10 | 6.25 | 1 | 3000 |
| beta[6,2] | 0.30 | 3.21 | -5.91 | 0.30 | 6.64 | 1 | 3000 |
| beta[7,2] | 0.15 | 3.11 | -5.69 | 0.18 | 6.18 | 1 | 570 |
| beta[8,2] | -0.02 | 3.08 | -6.01 | -0.06 | 6.11 | 1 | 3000 |
| beta[1,3] | 0.03 | 3.18 | -6.26 | 0.04 | 6.12 | 1 | 3000 |
| beta[2,3] | -0.01 | 3.17 | -6.20 | 0.03 | 6.10 | 1 | 3000 |
| beta[3,3] | -0.22 | 3.11 | -6.20 | -0.19 | 5.87 | 1 | 3000 |
| beta[4,3] | -0.21 | 3.17 | -6.45 | -0.23 | 6.09 | 1 | 3000 |
| beta[5,3] | -0.15 | 3.14 | -6.14 | -0.19 | 6.16 | 1 | 3000 |
| beta[6,3] | -0.07 | 3.08 | -6.07 | -0.14 | 6.01 | 1 | 770 |
| beta[7,3] | -0.03 | 3.22 | -6.15 | -0.08 | 6.44 | 1 | 800 |
| beta[8,3] | -0.01 | 3.08 | -6.14 | 0.02 | 5.99 | 1 | 790 |
| beta[1,4] | 0.19 | 3.15 | -5.91 | 0.24 | 6.58 | 1 | 3000 |
| beta[2,4] | 0.02 | 3.19 | -5.96 | -0.03 | 6.35 | 1 | 3000 |
| beta[3,4] | 0.04 | 3.10 | -6.05 | 0.02 | 6.07 | 1 | 3000 |
| beta[4,4] | -0.05 | 3.16 | -6.20 | -0.06 | 6.06 | 1 | 1000 |
| beta[5,4] | -0.03 | 3.18 | -6.22 | -0.10 | 6.25 | 1 | 3000 |
| beta[6,4] | -0.23 | 3.14 | -6.29 | -0.25 | 6.04 | 1 | 3000 |
| beta[7,4] | -0.27 | 3.11 | -6.23 | -0.27 | 5.77 | 1 | 3000 |
| beta[8,4] | -0.01 | 3.12 | -6.20 | -0.03 | 5.97 | 1 | 3000 |

```
# Extracting estimates and quantiles of u_env
summary.jags2.u_env <- as.data.frame(jags.2$BUGSoutput$summary[grep("^u_env\\[",
    rownames(jags.2$BUGSoutput$summary)), ][, c(1:3,
    5, 7:9)])
summary.jags2.u_env <- round(summary.jags2.u_env, 2)
# Table u_env
kable(summary.jags2.u_env, caption = "Second Bayesian model analysis and inference of u_env.",
    align = "c")
```

Second Bayesian model analysis and inference of u_env.

| | mean | sd | 2.5% | 50% | 97.5% | Rhat | n.eff |
|---|---|---|---|---|---|---|---|
| u_env[1,1] | 88.98 | 154.99 | -178.60 | 55.68 | 472.97 | 1.02 | 200 |

| | mean | sd | 2.5% | 50% | 97.5% | Rhat | n.eff |
|---|---|---|---|---|---|---|---|
| u_env[2,1] | -111.70 | 201.16 | -681.15 | -57.44 | 168.41 | 1.03 | 150 |
| u_env[1,2] | 89.49 | 152.68 | -155.55 | 56.11 | 461.18 | 1.04 | 80 |
| u_env[2,2] | -108.24 | 185.00 | -597.00 | -62.51 | 157.15 | 1.02 | 200 |
| u_env[1,3] | -106.08 | 195.09 | -612.47 | -57.23 | 163.90 | 1.02 | 470 |
| u_env[2,3] | 88.80 | 145.61 | -165.91 | 58.22 | 444.85 | 1.03 | 330 |
| u_env[1,4] | -109.93 | 198.81 | -692.35 | -58.25 | 157.33 | 1.01 | 390 |
| u_env[2,4] | 76.95 | 142.12 | -205.67 | 52.39 | 425.44 | 1.01 | 2000 |

```
# Extracting estimates and quantiles of u_variety
summary.jags2.u_variety <- as.data.frame(jags.2$BUGSoutput$summary[grep("^u_variety\\[",
    rownames(jags.2$BUGSoutput$summary)), ][, c(1:3,
    5, 7:9)])
summary.jags2.u_variety <- round(summary.jags2.u_variety,
    2)
# Table u_variety
kable(summary.jags2.u_variety, caption = "Second Bayesian model analysis and inference of u_variety.",
    align = "c")
```

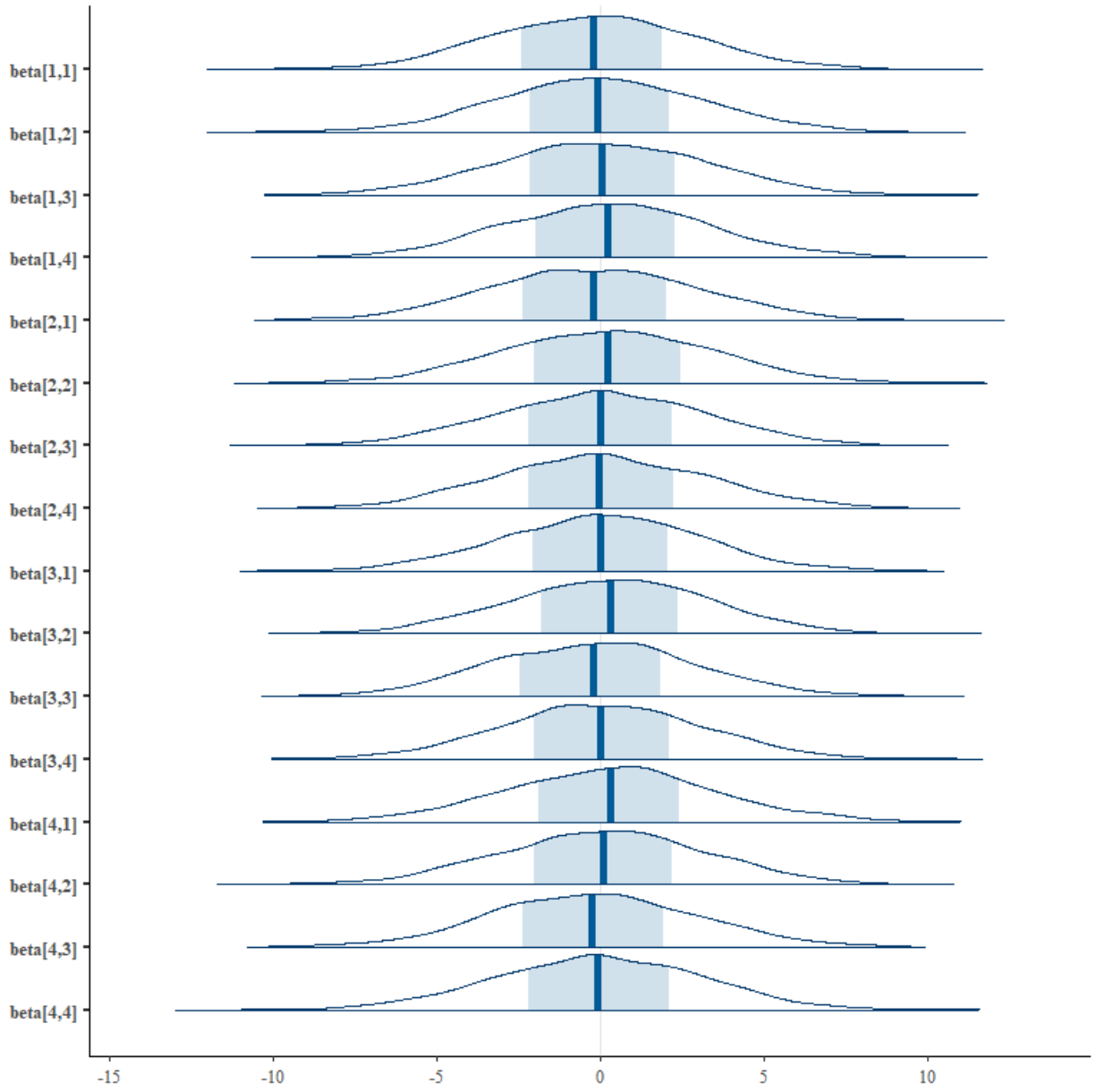
Second Bayesian model analysis and inference of u_variety.

| | mean | sd | 2.5% | 50% | 97.5% | Rhat | n.eff |
|---|---|---|---|---|---|---|---|
| u_variety[1,1] | 86.19 | 170.11 | -195.82 | 55.13 | 503.79 | 1.05 | 95 |
| u_variety[2,1] | -116.33 | 214.66 | -693.00 | -57.55 | 140.07 | 1.05 | 150 |
| u_variety[1,2] | -116.47 | 224.92 | -742.86 | -53.02 | 173.00 | 1.04 | 210 |
| u_variety[2,2] | 86.56 | 142.87 | -153.16 | 58.43 | 454.11 | 1.02 | 600 |
| u_variety[1,3] | 83.53 | 156.57 | -186.18 | 52.09 | 459.86 | 1.03 | 200 |
| u_variety[2,3] | -109.32 | 195.67 | -642.02 | -57.76 | 163.15 | 1.03 | 170 |
| u_variety[1,4] | -115.87 | 228.14 | -760.80 | -54.48 | 153.99 | 1.04 | 270 |
| u_variety[2,4] | 93.95 | 156.06 | -154.41 | 57.48 | 496.90 | 1.05 | 140 |

Regarding the Rhat we can see we have good results for beta and u_env, in fact they are all around 1 and less than 1.1. For u_variety instead we have values also that reach the 1.12 and this suggests some problems of convergence for this parameter. For the ESS represented by the column n.eff we have good values for parameters beta, instead for u_variety we see some low values (for u_env just in 2 cases), and this could indicate autocorrelation, even if they still have also good values registered.

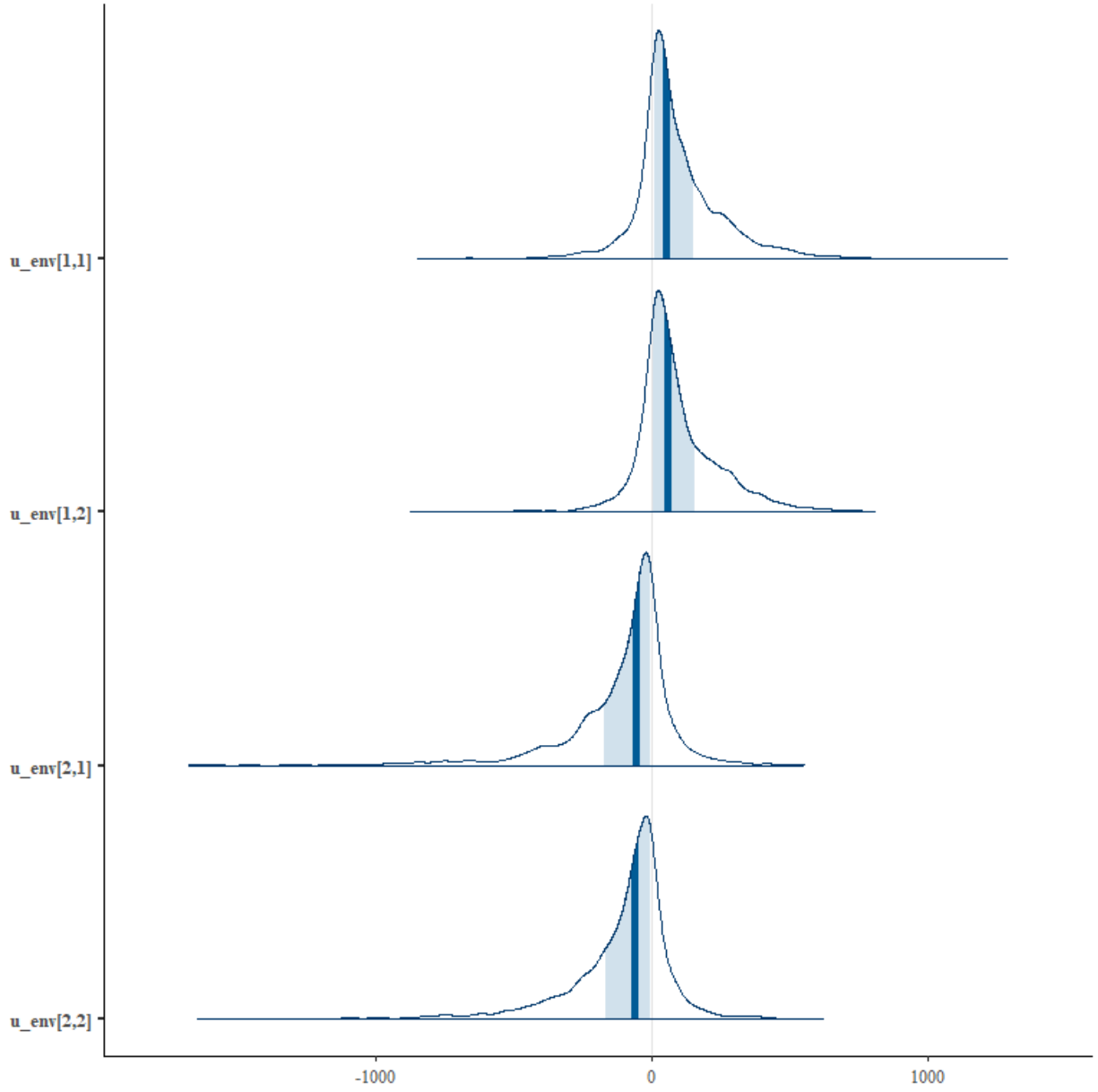
Now, let's look at the posterior distribution. They are shown in Figures 24 for beta, 25 for u_env 26 for u_variety.

```
# Posterior samples
mcmc.jags2 <- as.mcmc(jags.2)
mcmc_areas(mcmc.jags2, pars = c("beta[1,1]", "beta[1,2]",
    "beta[1,3]", "beta[1,4]", "beta[2,1]", "beta[2,2]",
    "beta[2,3]", "beta[2,4]", "beta[3,1]", "beta[3,2]",
    "beta[3,3]", "beta[3,4]", "beta[4,1]", "beta[4,2]",
    "beta[4,3]", "beta[4,4]"))
```



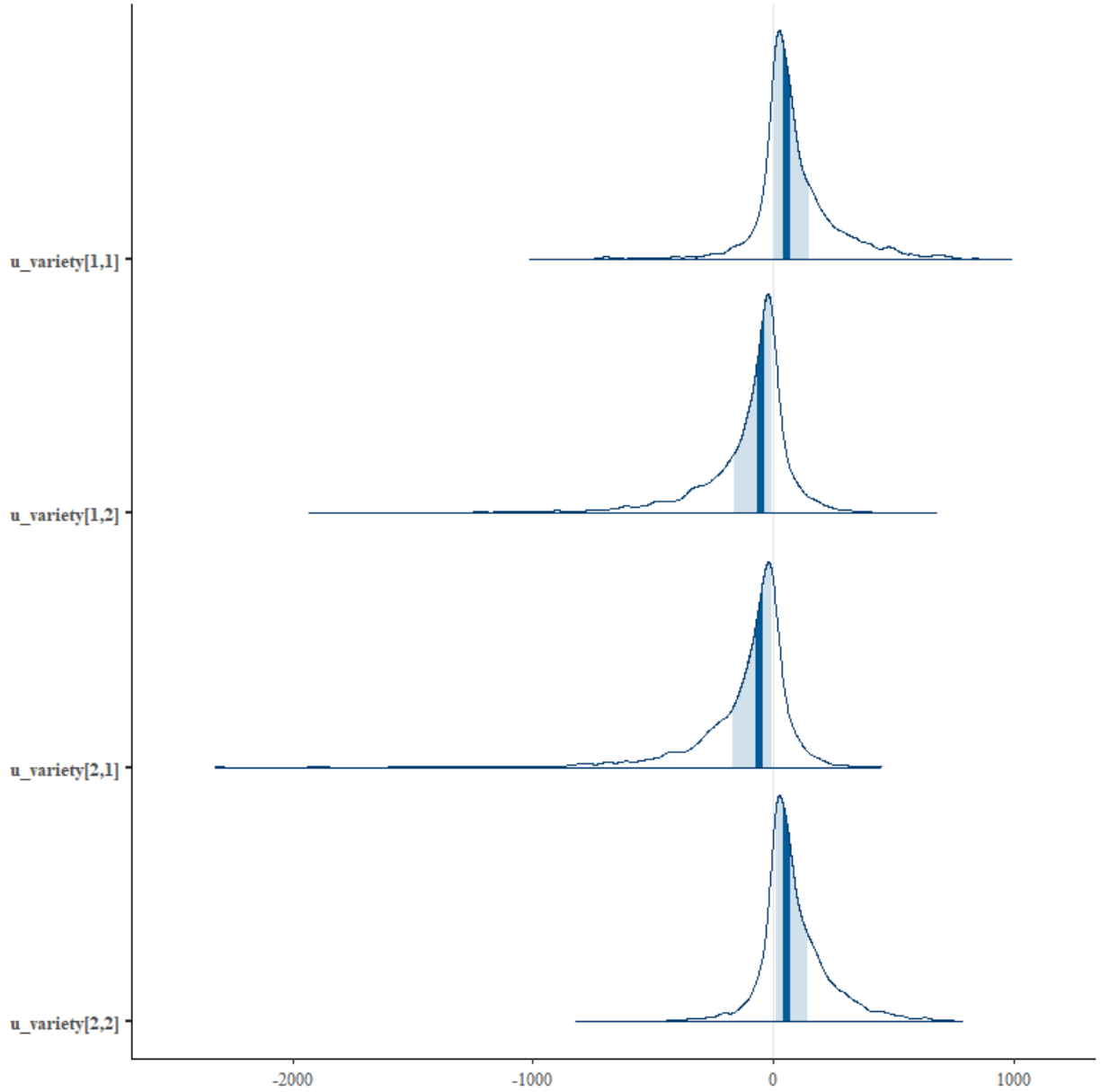

Posterior Distributions beta, Second model.

```
# Posterior samples
mcmc.jags2 <- as.mcmc(jags.2)
mcmc_areas(mcmc.jags2, pars = c("u_env[1,1]", "u_env[1,2]",
    "u_env[2,1]", "u_env[2,2]"))
```




Posterior Distributions u_env, Second model.

```
# Posterior samples
mcmc.jags2 <- as.mcmc(jags.2)
mcmc_areas(mcmc.jags2, pars = c("u_variety[1,1]", "u_variety[1,2]",
    "u_variety[2,1]", "u_variety[2,2]"))
```




Posterior distributions u_variety, Second model.

As we can see the distributions seem symmetric and this is a good sign of convergence, even if the distributions have high variability. Moreover, in general beta are close to 0, in particular the parameters beta[3,4], suggesting that it can be not significant. u_variety and u_env are less symmetric, but their means are far away from 0.

From the results shown below, we can see that for the parameters beta we have a MC standard error between 0.09 and 0.11, so this suggests a good precision in the estimates. For u_env instead, we have a higher MC standard error but still acceptable, and for u_variety it is similar but with very high values in the second and third chains. This reflects the ESS in particular for the last parameter mentioned, that was quite low.

```
mcse.jags2 <- list()
# MCSE for beta (8 parameters for 4 classes)
mcse.jags2$beta <- lapply(1:3, function(mc) {
    sapply(1:4, function(class) {
        sapply(1:8, function(coef) {
            # Extract samples of beta for the
            # chain mc and the category class
            param.name <- paste0("beta[", coef, ",",
                class, "]")
            MCSE(jags.2$BUGSoutput$sims.array[, mc,
                param.name])
        })
    })
})

# Results in 3 matrices (one per mc)
mcse.jags2$beta <- lapply(mcse.jags2$beta, function(beta.mc) {
    matrix(beta.mc, nrow = 8, ncol = 4)
})

# MCSE for u_env (2 parameters for 4 classes)
mcse.jags2$u_env <- lapply(1:3, function(mc) {
    sapply(1:4, function(class) {
        sapply(1:2, function(coef) {
            # Extract samples of u_env for the
            # chain mc and the category class
            param.name <- paste0("u_env[", coef, ",",
                class, "]")
            MCSE(jags.2$BUGSoutput$sims.array[, mc,
                param.name])
        })
    })
})

# Results in 3 matrices (one per mc)
mcse.jags2$u_env <- lapply(mcse.jags2$u_env, function(u_env.mc) {
    matrix(u_env.mc, nrow = 2, ncol = 4)
})

# MCSE for u_variety (2 parameters for 4 classes)
mcse.jags2$u_variety <- lapply(1:3, function(mc) {
    sapply(1:4, function(class) {
        sapply(1:2, function(coef) {
            # Extract samples of u_variety for
            # the chain mc and the category class
            param.name <- paste0("u_variety[", coef,
                ",", class, "]")
            MCSE(jags.2$BUGSoutput$sims.array[, mc,
                param.name])
        })
    })
})

# Results in 3 matrices (one per mc)
mcse.jags2$u_variety <- lapply(mcse.jags2$u_variety,
    function(u_variety.mc) {
        matrix(u_variety.mc, nrow = 2, ncol = 4)
    })

# Show results
mcse.jags2
```

```
$beta
$beta[[1]]
          [,1]       [,2]       [,3]       [,4]
[1,] 0.09948613 0.11372712 0.10148866 0.09292568
[2,] 0.09647093 0.10241533 0.09970764 0.11008788
[3,] 0.09933356 0.10545850 0.09379310 0.10022234
[4,] 0.09943088 0.10646396 0.09729774 0.10008307
[5,] 0.10369850 0.09185514 0.09538649 0.09333744
[6,] 0.09600540 0.09536706 0.09720774 0.10079061
[7,] 0.10899903 0.09160421 0.11219738 0.11116940
[8,] 0.09370318 0.09620175 0.10545511 0.09101927

$beta[[2]]
          [,1]       [,2]       [,3]       [,4]
[1,] 0.09672075 0.10783112 0.10611141 0.09603628
[2,] 0.10154913 0.09605738 0.09896075 0.09830543
[3,] 0.10286762 0.09870444 0.09671723 0.09275297
[4,] 0.10805213 0.09546023 0.10700244 0.09628556
[5,] 0.10259002 0.10185480 0.10334054 0.10015436
[6,] 0.10159797 0.10177745 0.09828748 0.11115708
[7,] 0.09561817 0.12602559 0.09729414 0.09791648
[8,] 0.10495500 0.09297310 0.09909149 0.11690660

$beta[[3]]
          [,1]       [,2]       [,3]       [,4]
[1,] 0.10403215 0.10537997 0.09700883 0.09426965
[2,] 0.10508559 0.10919949 0.09993500 0.10329402
[3,] 0.09176939 0.10718837 0.11170797 0.09954301
[4,] 0.09883817 0.09926414 0.09798900 0.10785051
[5,] 0.10630623 0.10554316 0.10483544 0.11419485
[6,] 0.09723668 0.09703970 0.09892128 0.10204698
[7,] 0.09559069 0.11307021 0.11998656 0.10205598
[8,] 0.09520800 0.09001078 0.09550049 0.09499044


$u_env
$u_env[[1]]
         [,1]     [,2]     [,3]     [,4]
[1,] 10.92315 10.61718 11.37301 15.99279
[2,] 11.35316  8.64947 10.98754 11.34479

$u_env[[2]]
         [,1]     [,2]     [,3]     [,4]
[1,] 27.03981 28.89807 28.41796 26.49339
[2,] 31.17636 23.53371 24.11290 22.00660

$u_env[[3]]
         [,1]     [,2]      [,3]      [,4]
[1,] 12.66059 14.93289 16.128997 14.468968
[2,] 14.61881 13.31488  8.917294  8.112188


$u_variety
$u_variety[[1]]
          [,1]      [,2]      [,3]     [,4]
[1,] 11.775842 10.636711 11.788622 11.26285
[2,]  8.865077  9.423258  8.718161 10.04243

$u_variety[[2]]
         [,1]     [,2]     [,3]     [,4]
[1,] 33.57500 43.66577 29.62802 34.19193
[2,] 35.71136 21.19349 22.27639 30.52477

$u_variety[[3]]
         [,1]     [,2]     [,3]     [,4]
[1,] 15.57873 24.35395 12.71765 22.67067
[2,] 20.09207 14.30468 17.23350 11.60197
```

From these results in general we can see that many parameters have Z-score close to 0, and this suggests that there are not significant differences between sample means in the first and last part of the chain. Just in the second [[2]] chain there are a couple of them with $|Z > 2|$, for example beta[3,3], beta[7,1] or u_env in the third chain that gave very bad outputs.

```
mcmc2.final <- mcmc.list(mcmc.jags2)
# Geweke diagnostic
(geweke.diag.jags2 <- geweke.diag(mcmc2.final))
```

```
[[1]]

Fraction in 1st window = 0.1
Fraction in 2nd window = 0.5

     beta[1,1]      beta[1,2]      beta[1,3]      beta[1,4]
       0.72197       -0.08680        0.39562        0.61540
     beta[2,1]      beta[2,2]      beta[2,3]      beta[2,4]
      -0.17520       -0.39713       -1.95459        1.55923
     beta[3,1]      beta[3,2]      beta[3,3]      beta[3,4]
      -0.65059        1.94145       -1.58069        0.46940
     beta[4,1]      beta[4,2]      beta[4,3]      beta[4,4]
       0.53935       -0.19975        1.00811       -0.56715
     beta[5,1]      beta[5,2]      beta[5,3]      beta[5,4]
       0.12279       -1.43339       -0.28162       -0.22161
     beta[6,1]      beta[6,2]      beta[6,3]      beta[6,4]
      -1.03118       -0.07620       -0.49685       -1.61046
     beta[7,1]      beta[7,2]      beta[7,3]      beta[7,4]
       0.19293        1.32723        1.35018       -1.48321
     beta[8,1]      beta[8,2]      beta[8,3]      beta[8,4]
      -0.75849       -0.08451        1.23575       -0.24791
      deviance        tau_env    tau_variety     u_env[1,1]
       0.46631        0.09264        1.58714        0.60821
    u_env[1,2]     u_env[1,3]     u_env[1,4]     u_env[2,1]
      -1.22793        1.70976       -0.15235        0.54315
    u_env[2,2]     u_env[2,3]     u_env[2,4] u_variety[1,1]
       0.49221       -0.05027       -0.01326       -2.21879
u_variety[1,2] u_variety[1,3] u_variety[1,4] u_variety[2,1]
       0.40034       -2.29755       -0.21851       -0.88078
u_variety[2,2] u_variety[2,3] u_variety[2,4]
      -0.56072       -0.09987       -2.28460


[[2]]

Fraction in 1st window = 0.1
Fraction in 2nd window = 0.5

     beta[1,1]      beta[1,2]      beta[1,3]      beta[1,4]
      -0.33312       -0.56146        1.19134        0.87277
     beta[2,1]      beta[2,2]      beta[2,3]      beta[2,4]
      -0.56478        0.49817        0.75323        1.40234
     beta[3,1]      beta[3,2]      beta[3,3]      beta[3,4]
      -0.02853        0.38206       -1.23727        0.36236
     beta[4,1]      beta[4,2]      beta[4,3]      beta[4,4]
       3.17427        0.28872       -0.52269       -3.21001
     beta[5,1]      beta[5,2]      beta[5,3]      beta[5,4]
       1.59047       -0.20701       -1.76488       -0.69266
     beta[6,1]      beta[6,2]      beta[6,3]      beta[6,4]
       1.43317        2.65315       -0.26631       -2.17633
     beta[7,1]      beta[7,2]      beta[7,3]      beta[7,4]
      -1.71201        0.22536       -1.49200        0.14932
     beta[8,1]      beta[8,2]      beta[8,3]      beta[8,4]
       0.27339       -1.53511       -0.90887       -0.93075
      deviance        tau_env    tau_variety     u_env[1,1]
       1.88925        1.55470        1.81988       -1.83702
    u_env[1,2]     u_env[1,3]     u_env[1,4]     u_env[2,1]
      -2.24067        3.13952        3.46506        3.94055
    u_env[2,2]     u_env[2,3]     u_env[2,4] u_variety[1,1]
       2.92857       -0.06566        0.20114       -2.18114
u_variety[1,2] u_variety[1,3] u_variety[1,4] u_variety[2,1]
       1.97987       -3.05947        2.48112        2.98973
u_variety[2,2] u_variety[2,3] u_variety[2,4]
      -2.61316        2.42528       -4.17799


[[3]]
```

```
Fraction in 1st window = 0.1
Fraction in 2nd window = 0.5

     beta[1,1]      beta[1,2]      beta[1,3]      beta[1,4]
    -1.2077330     -0.6383977      0.0005624     -0.2697882
     beta[2,1]      beta[2,2]      beta[2,3]      beta[2,4]
    -2.1582499      0.0784580     -0.3082415      1.2905872
     beta[3,1]      beta[3,2]      beta[3,3]      beta[3,4]
    -0.7126864     -0.2185439     -0.1344120      0.3910650
     beta[4,1]      beta[4,2]      beta[4,3]      beta[4,4]
    -0.3747670     -0.3354879     -0.0571950      1.8296630
     beta[5,1]      beta[5,2]      beta[5,3]      beta[5,4]
     1.2288701      0.1955381      1.4547383     -0.2126364
     beta[6,1]      beta[6,2]      beta[6,3]      beta[6,4]
     0.5752383      1.1362539     -0.0885482     -0.7501371
     beta[7,1]      beta[7,2]      beta[7,3]      beta[7,4]
    -0.9406292     -3.0232132      0.9684670     -0.7735781
     beta[8,1]      beta[8,2]      beta[8,3]      beta[8,4]
     0.2871906     -0.1432240      0.4762278      0.4118817
      deviance        tau_env    tau_variety     u_env[1,1]
    -1.1344662     -0.3864017     -0.4825336     -1.2194585
    u_env[1,2]     u_env[1,3]     u_env[1,4]     u_env[2,1]
    -0.3543162      1.9934827      3.1157471     -0.0447645
    u_env[2,2]     u_env[2,3]     u_env[2,4] u_variety[1,1]
     0.8299203     -0.6441971      0.1306346      2.5225614
u_variety[1,2] u_variety[1,3] u_variety[1,4] u_variety[2,1]
     2.6339693      1.1215360      1.5618398      2.9409563
u_variety[2,2] u_variety[2,3] u_variety[2,4]
    -0.0710695      2.4758028     -0.6620085
```

We take in consideration only the Stationarity tests. As we can see, all parameters passed the stationarity test.

```
# Heidelberger-Welch diagnostic
(heidel.diag.jags2 <- heidel.diag(mcmc2.final))
```

```
[[1]]

               Stationarity start     p-value
               test         iteration
beta[1,1]      passed         1       0.185558
beta[1,2]      passed         1       0.317390
beta[1,3]      passed         1       0.217108
beta[1,4]      passed         1       0.778943
beta[2,1]      passed         1       0.251220
beta[2,2]      passed         1       0.946034
beta[2,3]      passed         1       0.583425
beta[2,4]      passed         1       0.178409
beta[3,1]      passed         1       0.062484
beta[3,2]      passed         1       0.135879
beta[3,3]      passed         1       0.915485
beta[3,4]      passed         1       0.958821
beta[4,1]      passed         1       0.449777
beta[4,2]      passed         1       0.431691
beta[4,3]      passed         1       0.400969
beta[4,4]      passed         1       0.985463
beta[5,1]      passed         1       0.257242
beta[5,2]      passed         1       0.514519
beta[5,3]      passed         1       0.616122
beta[5,4]      passed         1       0.386273
beta[6,1]      passed         1       0.889959
beta[6,2]      passed         1       0.175669
beta[6,3]      passed         1       0.527464
beta[6,4]      passed         1       0.670087
beta[7,1]      passed         1       0.918000
beta[7,2]      passed         1       0.063752
beta[7,3]      passed         1       0.121070
beta[7,4]      passed         1       0.594285
beta[8,1]      passed         1       0.377020
beta[8,2]      passed         1       0.600861
beta[8,3]      passed         1       0.292258
beta[8,4]      passed         1       0.578420
deviance       passed         1       0.846843
tau_env        passed         1       0.976139
tau_variety    passed         1       0.362374
u_env[1,1]     failed        NA       0.000935
u_env[1,2]     passed         1       0.403459
u_env[1,3]     passed         1       0.287477
u_env[1,4]     passed         1       0.482106
u_env[2,1]     passed         1       0.269431
u_env[2,2]     failed        NA       0.003510
u_env[2,3]     passed         1       0.677637
u_env[2,4]     passed         1       0.642540
u_variety[1,1] passed         1       0.095049
u_variety[1,2] passed         1       0.632913
u_variety[1,3] passed         1       0.103650
u_variety[1,4] passed         1       0.359639
u_variety[2,1] passed       101       0.114696
u_variety[2,2] passed         1       0.719174
u_variety[2,3] passed         1       0.537081
u_variety[2,4] passed         1       0.194392

               Halfwidth Mean      Halfwidth
               test
beta[1,1]      failed    -1.88e-01 2.02e-01
beta[1,2]      failed    -4.32e-02 2.12e-01
beta[1,3]      failed     7.92e-02 1.89e-01
beta[1,4]      failed     2.01e-01 1.68e-01
beta[2,1]      failed    -1.42e-01 1.96e-01
beta[2,2]      failed     2.13e-01 2.03e-01
beta[2,3]      failed    -7.86e-03 1.95e-01
beta[2,4]      failed     7.52e-02 1.98e-01
beta[3,1]      failed    -8.87e-02 1.81e-01
beta[3,2]      failed     3.05e-01 2.00e-01
```

```
beta[3,3]      failed    -2.20e-01 1.84e-01
beta[3,4]      failed     7.89e-02 1.98e-01
beta[4,1]      failed     3.40e-01 1.96e-01
beta[4,2]      failed     6.32e-02 2.04e-01
beta[4,3]      failed    -2.12e-01 1.97e-01
beta[4,4]      failed    -2.48e-01 2.00e-01
beta[5,1]      failed    -1.03e-02 1.90e-01
beta[5,2]      failed     1.51e-01 1.81e-01
beta[5,3]      failed    -1.28e-01 1.89e-01
beta[5,4]      failed    -2.81e-02 1.84e-01
beta[6,1]      failed     1.84e-01 1.63e-01
beta[6,2]      failed     3.51e-01 1.88e-01
beta[6,3]      failed    -2.03e-01 1.94e-01
beta[6,4]      failed    -1.65e-01 1.75e-01
beta[7,1]      failed    -1.09e-02 2.03e-01
beta[7,2]      failed     3.91e-01 1.80e-01
beta[7,3]      failed    -1.82e-01 1.97e-01
beta[7,4]      failed    -3.39e-01 2.02e-01
beta[8,1]      failed     2.43e-01 1.84e-01
beta[8,2]      failed     2.24e-02 1.96e-01
beta[8,3]      failed     9.94e-03 1.94e-01
beta[8,4]      failed    -9.10e-02 1.79e-01
deviance       failed     1.46e-01 4.27e-02
tau_env        failed     6.55e-04 3.09e-04
tau_variety    failed     4.29e-04 1.24e-04
u_env[1,1]     <NA>             NA       NA
u_env[1,2]     failed     6.90e+01 2.02e+01
u_env[1,3]     failed    -9.23e+01 2.12e+01
u_env[1,4]     failed    -9.61e+01 2.30e+01
u_env[2,1]     failed    -9.25e+01 1.58e+01
u_env[2,2]     <NA>             NA       NA
u_env[2,3]     failed     7.71e+01 1.68e+01
u_env[2,4]     failed     7.85e+01 1.52e+01
u_variety[1,1] failed     7.39e+01 1.73e+01
u_variety[1,2] failed    -8.57e+01 1.50e+01
u_variety[1,3] failed     7.60e+01 1.61e+01
u_variety[1,4] failed    -8.99e+01 1.75e+01
u_variety[2,1] failed    -8.11e+01 1.40e+01
u_variety[2,2] failed     7.64e+01 1.52e+01
u_variety[2,3] failed    -7.96e+01 1.66e+01
u_variety[2,4] failed     7.41e+01 1.85e+01

[[2]]

               Stationarity start     p-value
               test         iteration
beta[1,1]      passed         1       0.8165
beta[1,2]      passed         1       0.1283
beta[1,3]      passed         1       0.2569
beta[1,4]      passed         1       0.3012
beta[2,1]      passed         1       0.7282
beta[2,2]      passed         1       0.3797
beta[2,3]      passed         1       0.2670
beta[2,4]      passed         1       0.0655
beta[3,1]      passed         1       0.8537
beta[3,2]      passed         1       0.0645
beta[3,3]      passed         1       0.5169
beta[3,4]      passed         1       0.6049
beta[4,1]      passed         1       0.2467
beta[4,2]      passed         1       0.7532
beta[4,3]      passed         1       0.1962
beta[4,4]      passed         1       0.2858
beta[5,1]      passed         1       0.3948
beta[5,2]      passed         1       0.9654
beta[5,3]      passed         1       0.1919
beta[5,4]      passed         1       0.4772
beta[6,1]      passed         1       0.0793
beta[6,2]      passed         1       0.1869
beta[6,3]      passed         1       0.7872
```

```
beta[6,4]      passed         1       0.5447
beta[7,1]      passed         1       0.2038
beta[7,2]      passed         1       0.3045
beta[7,3]      passed         1       0.5858
beta[7,4]      passed         1       0.6927
beta[8,1]      passed         1       0.5461
beta[8,2]      passed         1       0.0982
beta[8,3]      passed         1       0.2266
beta[8,4]      passed         1       0.4903
deviance       passed         1       0.4598
tau_env        passed       101       0.6652
tau_variety    passed         1       0.0730
u_env[1,1]     passed         1       0.1880
u_env[1,2]     passed         1       0.2800
u_env[1,3]     passed         1       0.2788
u_env[1,4]     passed       301       0.0952
u_env[2,1]     passed         1       0.4174
u_env[2,2]     passed         1       0.3474
u_env[2,3]     passed         1       0.2495
u_env[2,4]     passed         1       0.2523
u_variety[1,1] passed         1       0.4364
u_variety[1,2] passed         1       0.7529
u_variety[1,3] passed       101       0.0609
u_variety[1,4] passed         1       0.5828
u_variety[2,1] passed         1       0.1086
u_variety[2,2] passed         1       0.3802
u_variety[2,3] passed         1       0.6033
u_variety[2,4] passed         1       0.0582

               Halfwidth Mean      Halfwidth
               test
beta[1,1]      failed    -3.09e-01 1.94e-01
beta[1,2]      failed    -7.48e-02 1.92e-01
beta[1,3]      failed    -7.79e-02 1.99e-01
beta[1,4]      failed     1.62e-01 1.75e-01
beta[2,1]      failed    -2.03e-01 2.01e-01
beta[2,2]      failed     4.14e-02 1.92e-01
beta[2,3]      failed     2.84e-02 2.00e-01
beta[2,4]      failed     5.64e-02 1.98e-01
beta[3,1]      failed    -7.02e-03 1.92e-01
beta[3,2]      failed     1.83e-01 1.91e-01
beta[3,3]      failed    -1.16e-01 1.91e-01
beta[3,4]      failed    -2.30e-02 1.84e-01
beta[4,1]      failed     2.77e-01 2.05e-01
beta[4,2]      failed     6.83e-02 1.96e-01
beta[4,3]      failed    -2.88e-01 1.95e-01
beta[4,4]      failed     4.08e-02 1.92e-01
beta[5,1]      failed    -2.40e-02 1.97e-01
beta[5,2]      failed     9.94e-02 1.90e-01
beta[5,3]      failed    -1.32e-01 1.99e-01
beta[5,4]      failed    -1.25e-01 1.97e-01
beta[6,1]      failed     2.44e-02 1.95e-01
beta[6,2]      failed     3.02e-01 1.99e-01
beta[6,3]      failed    -1.49e-01 1.91e-01
beta[6,4]      failed    -2.23e-01 2.11e-01
beta[7,1]      failed     1.93e-02 1.95e-01
beta[7,2]      failed    -5.52e-02 2.10e-01
beta[7,3]      failed     1.97e-01 1.99e-01
beta[7,4]      failed    -2.63e-01 1.91e-01
beta[8,1]      failed     3.59e-02 1.90e-01
beta[8,2]      failed    -5.90e-02 1.85e-01
beta[8,3]      failed     1.71e-01 2.10e-01
beta[8,4]      failed     9.99e-02 2.14e-01
deviance       failed     9.99e-02 3.93e-02
tau_env        failed     2.62e-04 2.09e-04
tau_variety    failed     3.47e-04 2.61e-04
u_env[1,1]     failed     1.11e+02 3.79e+01
u_env[1,2]     failed     1.24e+02 3.76e+01
u_env[1,3]     failed    -1.23e+02 3.78e+01
```

```
u_env[1,4]     failed    -1.13e+02 2.92e+01
u_env[2,1]     failed    -1.44e+02 4.64e+01
u_env[2,2]     failed    -1.34e+02 3.42e+01
u_env[2,3]     failed     1.04e+02 3.68e+01
u_env[2,4]     failed     8.15e+01 3.45e+01
u_variety[1,1] failed     1.21e+02 4.53e+01
u_variety[1,2] failed    -1.35e+02 6.48e+01
u_variety[1,3] failed     1.13e+02 4.15e+01
u_variety[1,4] failed    -1.37e+02 6.54e+01
u_variety[2,1] failed    -1.45e+02 5.00e+01
u_variety[2,2] failed     9.66e+01 3.04e+01
u_variety[2,3] failed    -1.27e+02 3.18e+01
u_variety[2,4] failed     1.19e+02 3.99e+01

[[3]]

               Stationarity start     p-value
               test         iteration
beta[1,1]      passed       201       0.2834
beta[1,2]      passed         1       0.1678
beta[1,3]      passed         1       0.4040
beta[1,4]      passed         1       0.1730
beta[2,1]      passed         1       0.4620
beta[2,2]      passed         1       0.9327
beta[2,3]      passed         1       0.3674
beta[2,4]      passed         1       0.2237
beta[3,1]      passed         1       0.7186
beta[3,2]      passed         1       0.2054
beta[3,3]      passed         1       0.3990
beta[3,4]      passed         1       0.6279
beta[4,1]      passed         1       0.9588
beta[4,2]      passed         1       0.5457
beta[4,3]      passed         1       0.8404
beta[4,4]      passed         1       0.2606
beta[5,1]      passed         1       0.3383
beta[5,2]      passed         1       0.6050
beta[5,3]      passed         1       0.0987
beta[5,4]      passed         1       0.2172
beta[6,1]      passed         1       0.9639
beta[6,2]      passed         1       0.4649
beta[6,3]      passed         1       0.9794
beta[6,4]      passed         1       0.7100
beta[7,1]      passed         1       0.8408
beta[7,2]      passed         1       0.0839
beta[7,3]      passed         1       0.4563
beta[7,4]      passed         1       0.1214
beta[8,1]      passed         1       0.1623
beta[8,2]      passed         1       0.7919
beta[8,3]      passed         1       0.6247
beta[8,4]      passed         1       0.8441
deviance       passed         1       0.8321
tau_env        passed         1       0.3296
tau_variety    passed         1       0.7030
u_env[1,1]     passed         1       0.1984
u_env[1,2]     passed         1       0.2390
u_env[1,3]     passed         1       0.1223
u_env[1,4]     passed         1       0.1023
u_env[2,1]     passed         1       0.8627
u_env[2,2]     passed         1       0.7275
u_env[2,3]     passed         1       0.1200
u_env[2,4]     passed       301       0.0561
u_variety[1,1] passed         1       0.0907
u_variety[1,2] passed         1       0.0697
u_variety[1,3] passed         1       0.3861
u_variety[1,4] passed         1       0.1353
u_variety[2,1] passed         1       0.0983
u_variety[2,2] passed         1       0.6783
u_variety[2,3] passed       101       0.0613
u_variety[2,4] passed         1       0.7160
```

```
              Halfwidth Mean      Halfwidth
              test
beta[1,1]     failed    -1.42e-01 2.20e-01
beta[1,2]     failed    -3.58e-02 1.97e-01
beta[1,3]     failed     8.26e-02 1.91e-01
beta[1,4]     failed     2.01e-01 1.86e-01
beta[2,1]     failed    -2.80e-01 2.02e-01
beta[2,2]     failed     3.62e-01 1.99e-01
beta[2,3]     failed    -6.31e-02 1.94e-01
beta[2,4]     failed    -7.49e-02 1.98e-01
beta[3,1]     failed     5.22e-02 1.80e-01
beta[3,2]     failed     2.57e-01 2.02e-01
beta[3,3]     failed    -3.10e-01 2.05e-01
beta[3,4]     failed     5.53e-02 1.94e-01
beta[4,1]     failed     2.46e-01 1.95e-01
beta[4,2]     failed     6.32e-02 1.90e-01
beta[4,3]     failed    -1.33e-01 1.97e-01
beta[4,4]     failed     5.20e-02 1.98e-01
beta[5,1]     failed     4.39e-02 2.00e-01
beta[5,2]     failed     2.10e-01 1.73e-01
beta[5,3]     failed    -1.88e-01 1.96e-01
beta[5,4]     failed     4.96e-02 2.00e-01
beta[6,1]     failed     4.11e-02 1.93e-01
beta[6,2]     failed     2.56e-01 1.91e-01
beta[6,3]     failed     1.53e-01 1.87e-01
beta[6,4]     failed    -3.14e-01 1.99e-01
beta[7,1]     failed     1.09e-01 1.91e-01
beta[7,2]     failed     1.15e-01 1.98e-01
beta[7,3]     failed    -9.04e-02 2.11e-01
beta[7,4]     failed    -2.10e-01 1.80e-01
beta[8,1]     failed     2.16e-01 1.94e-01
beta[8,2]     failed    -3.23e-02 1.78e-01
beta[8,3]     failed    -2.07e-01 1.84e-01
beta[8,4]     failed    -3.96e-02 1.94e-01
deviance      failed     1.28e-01 2.98e-02
tau_env       failed     3.36e-04 1.71e-04
tau_variety   failed     3.73e-04 1.61e-04
u_env[1,1]    failed     8.15e+01 2.04e+01
u_env[1,2]    failed     7.59e+01 2.33e+01
u_env[1,3]    failed    -1.03e+02 2.55e+01
u_env[1,4]    failed    -1.04e+02 2.29e+01
u_env[2,1]    failed    -9.82e+01 2.21e+01
u_env[2,2]    failed    -9.94e+01 1.99e+01
u_env[2,3]    failed     8.53e+01 1.38e+01
u_env[2,4]    failed     7.74e+01 1.70e+01
u_variety[1,1] failed    6.39e+01 2.29e+01
u_variety[1,2] failed    -1.28e+02 3.89e+01
u_variety[1,3] failed    6.93e+01 2.01e+01
u_variety[1,4] failed    -1.21e+02 4.27e+01
u_variety[2,1] failed    -1.21e+02 3.16e+01
u_variety[2,2] failed    8.66e+01 2.06e+01
u_variety[2,3] failed    -1.26e+02 3.63e+01
u_variety[2,4] failed    8.83e+01 1.75e+01
```

# 5. Models comparison

## 5.1. DIC

To compare models it is usually used the **Deviance Information Criterion (DIC)** , that is a generalization of the frequentist criterion AIC. It is defined as:

$$DIC = D_{\hat{\theta}}(y) + 2p_D = \hat{D}_{avg}(y) + p_D = 2\hat{D}_{avg}(y) - D_{\hat{\theta}}(y)$$

where $p_D$ is the effective number of parameters in a Bayesian model, and the estimated posterior mean deviance $\hat{D}_{avg}(y)$ can be approximate as

$$\hat{D}_{avg}(y) \approx \frac{1}{M} \sum_{j} -2log(f(y|\theta^{(j)})).$$

The model with the lower value of DIC is preferred. Since this criteria does not have an absolute scale, it is used just to categorize the models. In our case we have the following results:

```
 DIC First model DIC Second model
       6.7839910        0.2444207
```

As we can see the DIC assigns a lower value to the second model. So, based on this criteria, the best model seems to be the second one.

# 6. Frequentist analysis and comparison

Let's see now two frequentist alternative models.

For the first Bayesian model, the direct alternative is the Multinomial model through the library nnet. From this model we can see that even if four predictors were removed and the variables were standardized, the standard errors are still high. Moreover, the log-odds represented by coefficients' estimates are calculated compared to the first category, i.e. the category of reference. In particular we can see that for the variables ACHP, PHR and ALAP, the log odds to belong to class 2, 3 and 4 corresponds in an increasing way, to the rise of them. Contrary is for ANPL where a decrease of it causes lower log odds of belonging to class 3, 4 and 2 (in this sequential order). For PDMVG similar situation like ANPL, but in this class order: class 2, 4 and 3. Beside this analysis, we can see that, from a statistical point of view, the residual deviance of the model is 0.00013 and this suggests a very good adaptation of the model to data.

```
data.freq1 <- data.frame(y = data.sampled$Class, ACHP = sampled_standardized_predictors$`Average of chlorophyll in
the plant (ACHP)`,
    PHR = sampled_standardized_predictors$`Plant height  rate (PHR)`,
    ALAP = sampled_standardized_predictors$`Average leaf area of the plant (ALAP)`,
    ANPL = sampled_standardized_predictors$`Average number of plant leaves (ANPL)`,
    PDMVG = sampled_standardized_predictors$`Percentage of dry matter for vegetative growth (PDMVG)`,
    ARL = sampled_standardized_predictors$`Average root length (ARL)`,
    AWWR = sampled_standardized_predictors$`Average wet weight of the root (AWWR)`,
    PDMRG = sampled_standardized_predictors$`Percentage of dry matter for root growth (PDMRG)`)

# Model
model.multinom <- multinom(y ~ ACHP + PHR + ALAP +
    ANPL + PDMVG + ARL + AWWR + PDMRG, data = data.freq1)
```

```
# weights:  40 (27 variable)
initial  value 831.776617
iter  10 value 4.616940
iter  20 value 0.305942
iter  30 value 0.134887
iter  40 value 0.026905
iter  50 value 0.009470
iter  60 value 0.000825
iter  70 value 0.000351
final  value 0.000091
converged
```

```
# Summary
summary(model.multinom)
```

```
Call:
multinom(formula = y ~ ACHP + PHR + ALAP + ANPL + PDMVG + ARL +
    AWWR + PDMRG, data = data.freq1)

Coefficients:
  (Intercept)      ACHP       PHR      ALAP      ANPL
2  -4.8726314  1.635771  7.820320  5.469397 -15.64892
3   0.4066638 10.331380  6.532813  6.086612 -16.00545
4  -1.2616257 17.626210 12.717746 31.252953 -16.08033
       PDMVG        ARL       AWWR      PDMRG
2  -6.964723  23.102573  -5.799045 -14.097773
3 -13.250701 -13.482511   5.123214   7.740780
4  -8.304708  -9.623753 -15.547222   9.952025

Std. Errors:
  (Intercept)     ACHP       PHR     ALAP     ANPL
2    73202.88 72594.42  73295.15 177617.6 11804.93
3    59927.55 17069.49 113664.17 133625.8 16929.33
4    69631.91 49074.48 138997.58 191839.9 20897.20
      PDMVG      ARL     AWWR     PDMRG
2  85455.48 55228.40 13729.47 111556.43
3 118690.59 50875.16 18436.06  96361.97
4 142947.93 66458.41 25643.36 105626.43

Residual Deviance: 0.0001828036
AIC: 54.00018
```

For the second Bayesian model we try to reproduce a the hierarchical effect using hot-encoding of the variables Env and Variety, i.e. transforming them in dummies. In this case of course, we ignore the variability within these two levels. The coefficients are always interpretable as in the first frequentist model, but we still have high standard errors that suggest an uncertainty of the estimates. Finally, the residual deviance is 0.00019 and it is really low as in the previous model.

```
data.freq2 <- data.frame(y = data.sampled$Class, ACHP = sampled_standardized_predictors$`Average of chlorophyll in
the plant (ACHP)`,
    PHR = sampled_standardized_predictors$`Plant height  rate (PHR)`,
    ALAP = sampled_standardized_predictors$`Average leaf area of the plant (ALAP)`,
    ANPL = sampled_standardized_predictors$`Average number of plant leaves (ANPL)`,
    PDMVG = sampled_standardized_predictors$`Percentage of dry matter for vegetative growth (PDMVG)`,
    ARL = sampled_standardized_predictors$`Average root length (ARL)`,
    AWWR = sampled_standardized_predictors$`Average wet weight of the root (AWWR)`,
    PDMRG = sampled_standardized_predictors$`Percentage of dry matter for root growth (PDMRG)`,
    Env = data.sampled$Env, Variety = data.sampled$Variety)

dummies <- model.matrix(~Env + Variety, data = data.freq2)[,
    -1]
data.freq2 <- cbind(data.freq2, dummies)
# Removing columns Env and Variety
data.freq2 <- data.freq2[, !(names(data.freq2) %in%
    c("Env", "Variety"))]

# Model
model.multilev <- multinom(y ~ ACHP + PHR + ALAP +
    ANPL + PDMVG + ARL + AWWR + PDMRG + dummies, data = data.freq2)
```

```
# weights:  48 (33 variable)
initial  value 831.776617
iter  10 value 2.783065
iter  20 value 0.208353
iter  30 value 0.043475
iter  40 value 0.001564
final  value 0.000057
converged
```

```
# Summary
summary(model.multilev)
```

```
Call:
multinom(formula = y ~ ACHP + PHR + ALAP + ANPL + PDMVG + ARL +
    AWWR + PDMRG + dummies, data = data.freq2)

Coefficients:
  (Intercept)       ACHP        PHR     ALAP      ANPL
2   -7.506184  0.8556575  -2.858031 17.29805 -63.08806
3   10.588195 16.4835474  42.538487 44.77358 -20.35041
4  -32.231825 58.6328611 -36.210621 62.67204 -48.92745
       PDMVG       ARL      AWWR     PDMRG dummiesEnv
2  -4.373406  13.26635  6.744853 12.763911  -34.78799
3 -12.161083 -17.78592 18.311588 -2.367961   97.86028
4  11.607163 -41.59849 14.564165 -7.219960  -57.19544
  dummiesVariety
2       30.13591
3     -114.58912
4       51.85948

Std. Errors:
  (Intercept)     ACHP       PHR      ALAP      ANPL
2    21112.49 77873.95 114085.97 110531.13  11448.17
3    12695.36 21800.21  43280.32  43777.36  11932.17
4    31139.98 29683.20  68309.75  28093.91 177025.49
      PDMVG      ARL      AWWR     PDMRG dummiesEnv
2  55812.38 39739.68  14742.59 117003.70   55293.87
3  11331.39 28732.07  68386.19  42225.30   17050.55
4 127578.17 23975.25 143818.52  40413.93   45160.38
  dummiesVariety
2       68182.09
3       30904.59
4       66278.84

Residual Deviance: 0.0001134539
AIC: 66.00011
```

In the frequentist analysis, as mentioned in section 5.1., the criterion used to compare models is the **Akaike Information Criterion (AIC)**. It is defined as:

$$AIC = D_{\hat{\theta}}(y) + 2p$$

where $p$ are the parameters of the model and $D_{\hat{\theta}}(y)$ is the deviance evaluated at a representative point $\hat{\theta}$ (usually the posterior mean). More deeply, the deviance is defined as $D(y, \theta) = -2logf(y|\theta)$, and $D_{\hat{\theta}}(y) = D(y, \hat{\theta}(y))$. In our case we have the following results:

```
 AIC First model AIC Second model
        54.00018         66.00011
```

The AIC is lower for the first frequentist model, in fact also the residual deviance was smaller. Hence, for the frequentist analysis is preferable the first one.

# Conclusions

For the Bayesian analysis we have seen that the best model seems to be the Multilevel model with Hierarchical Random Terms. The DIC is really good, but the standard deviation and the MCSE of u_env and u_variety don't perform well. In particular, through the traceplots, we could see that the variance of the environment t_env doesn't capture so much the differences between Smart and Traditional greenhouse. Instead, t_variety seems to capture more variability. This can suggest us that, to capture the differences between environments, we should train the model with more data, considering also the variety C for example. Another alternative is also to explore other models or modify the values of parameters' distribution in the one used in this present project. Hence, it highlights some problems, and it should be deepened.

For the frequentist analysis the preferable model resulted to be the first one, even if there was not so much difference between both. What we can see the model with dummies variable adds complexity and creates a potential overfitting even if it can represents a good adaptability to the data.

# Appendix

```
            ACHP          PHR         AWWGV         ALAP
ACHP   1.00000000 -0.17138866  0.761339702 -0.41268862
PHR   -0.17138866  1.00000000  0.149556144  0.65685745
AWWGV  0.76133970  0.14955614  1.000000000  0.06093296
ALAP  -0.41268862  0.65685745  0.060932964  1.00000000
ANPL  -0.52581196  0.01929580 -0.250601704  0.41487568
ARD   -0.52607885  0.05107555 -0.036101538  0.54831881
ADWR  -0.46323526  0.01573653 -0.124341200  0.52615903
PDMVG -0.11975660  0.43126293  0.338511727  0.35001739
ARL   -0.75655339  0.43060727 -0.337310551  0.72965776
AWWR  -0.50119613  0.20570906 -0.005767363  0.71123117
ADWV   0.21130205  0.40000081  0.671484808  0.26570270
PDMRG -0.05412461 -0.19786849 -0.079049919 -0.04591853
              ANPL          ARD         ADWR         PDMVG
ACHP  -0.525811959 -0.52607885 -0.46323526 -0.119756604
PHR    0.019295804  0.05107555  0.01573653  0.431262925
AWWGV -0.250601704 -0.03610154 -0.12434120  0.338511727
ALAP   0.414875678  0.54831881  0.52615903  0.350017394
ANPL   1.000000000  0.51748723  0.81270018 -0.002894181
ARD    0.517487228  1.00000000  0.73958117  0.282039172
ADWR   0.812700176  0.73958117  1.00000000 -0.084370113
PDMVG -0.002894181  0.28203917 -0.08437011  1.000000000
ARL    0.585271682  0.71579229  0.63788126  0.142559934
AWWR   0.566157854  0.95437575  0.79703852  0.179507383
ADWV  -0.117787047  0.19322177 -0.11329203  0.914772491
PDMRG  0.623815246  0.03813401  0.63137425 -0.258625706
               ARL         AWWR         ADWV        PDMRG
ACHP  -0.75655339 -0.501196132  0.21130205 -0.05412461
PHR    0.43060727  0.205709062  0.40000081 -0.19786849
AWWGV -0.33731055 -0.005767363  0.67148481 -0.07904992
ALAP   0.72965776  0.711231175  0.26570270 -0.04591853
ANPL   0.58527168  0.566157854 -0.11778705  0.62381525
ARD    0.71579229  0.954375748  0.19322177  0.03813401
ADWR   0.63788126  0.797038522 -0.11329203  0.63137425
PDMVG  0.14255993  0.179507383  0.91477249 -0.25862571
ARL    1.00000000  0.787192235 -0.03959971 -0.01847740
AWWR   0.78719224  1.000000000  0.11745892  0.06435410
ADWV  -0.03959971  0.117458923  1.00000000 -0.20566761
PDMRG -0.01847740  0.064354100 -0.20566761  1.00000000
```

# References

(National University Academic Success Center 2024) (Ntzoufras 2009) (Agresti 2015) (Tardella 2023) (JAGS Tutorial n.d.)


Abdullah, Wisam. 2024. “Advanced IoT Agriculture 2024.” Kaggle Dataset. https://www.kaggle.com/datasets/wisam1985/advanced-iot-agriculture-2024/data (https://www.kaggle.com/datasets/wisam1985/advanced-iot-agriculture-2024/data).

Agresti, Alan. 2015. *Categorical Data Analysis*. Hoboken, New Jersey: Department of Statistics, University of Florida; Wiley Interscience.

JAGS Tutorial. n.d. “JAGS Tutorial.” https://www.sas.rochester.edu/psc/thestarlab/help/JAGS.pdf (https://www.sas.rochester.edu/psc/thestarlab/help/JAGS.pdf).

Lafta, Mohamed Ismail, and Wisam Dawood Abdullah. 2024. “Data-Driven Farming: Implementing Internet of Things for Agricultural Efficiency.” *IAES International Journal of Artificial Intelligence (IJ-AI)* 13 (3): 3588–98. https://doi.org/10.11591/ijai.v13.i3.pp3588-3598 (https://doi.org/10.11591/ijai.v13.i3.pp3588-3598).

National University Academic Success Center. 2024. “Multinomial Logistic Regression.” 2024. https://resources.nu.edu/statsresources/Multinomiallogistic#:~:text=A%20multinomial%20logistic%20regression%20(or,that%20are%20categorical%20or%20continuous. (https://resources.nu.edu/statsresources/Multinomiallogistic#:~:text=A%20multinomial%20logistic%20regression%20(or,that%20are%20categorical%20or%20continuous.)

Ntzoufras, Ioannis. 2009. *Bayesian Modeling Using WinBUGS*. Edited by Paolo Giudici, Geof H. Givens, and Bani K. Mallick. Wiley Series in Computational Statistics. Athens, Greece: Department of Statistics, Athens University of Economics; Business; Wiley.

Tardella, Luca. 2023. “Statistical Methods for Data Science & Laboratory II.”