A Bayesian Model for Multinomial Ordered Climbing Data

In sport climbing, the difficulty of a route is defined by a scale of grades. In general the maximum grade climbed depends on lots of factors that are difficult to measure, such as mental and physical strength,external conditions and personal preferencies. Neglecting these, the aim of the analysis is to see how quantities like weight, gender or years of experiences affect the maximum climbing grade.

A classification model is performed by implementing a Gibbs sample for multinomial ordered categories.

Climbing Data

The original source of the data is 8a.nu, a blog where climbers can register their ascents.

Original dataset:

```
RangeIndex: 49598 entries, 0 to 49597
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id_user         49598 non-null  int64
 1   is_female       49598 non-null  int64
 2   height          49598 non-null  int64
 3   weight          49598 non-null  int64
 4   is_bouldering   49598 non-null  int64
 5   index_grade     49598 non-null  int64
 6   age             31342 non-null  float64
 7   years_climbing  36110 non-null  float64
dtypes: float64(2), int64(6)
```

Variable description and preparation

- 'index_grade' is the target variable and is a number between 0 and 79, corresponding to the maximum grade; these where grouped in four ordered categories: 'beginner','intermediate','advanced','pro'. It was renamed 'max_climbing_grade'.

The others are the auxiliary variables:

- 'id_user' only provides the registration number in 8a.nu; it was discarded
- 'is_bouldering' = 1 if the grade is referred to the bouldering activity, otherwise the grade is referred to sport climbing
- 'height' expressed in [cm]; only height in the interval [140cm,230cm] were selected
- 'weight' expressed in [kg]; only weight in the interval [40kg,100kg] were selected

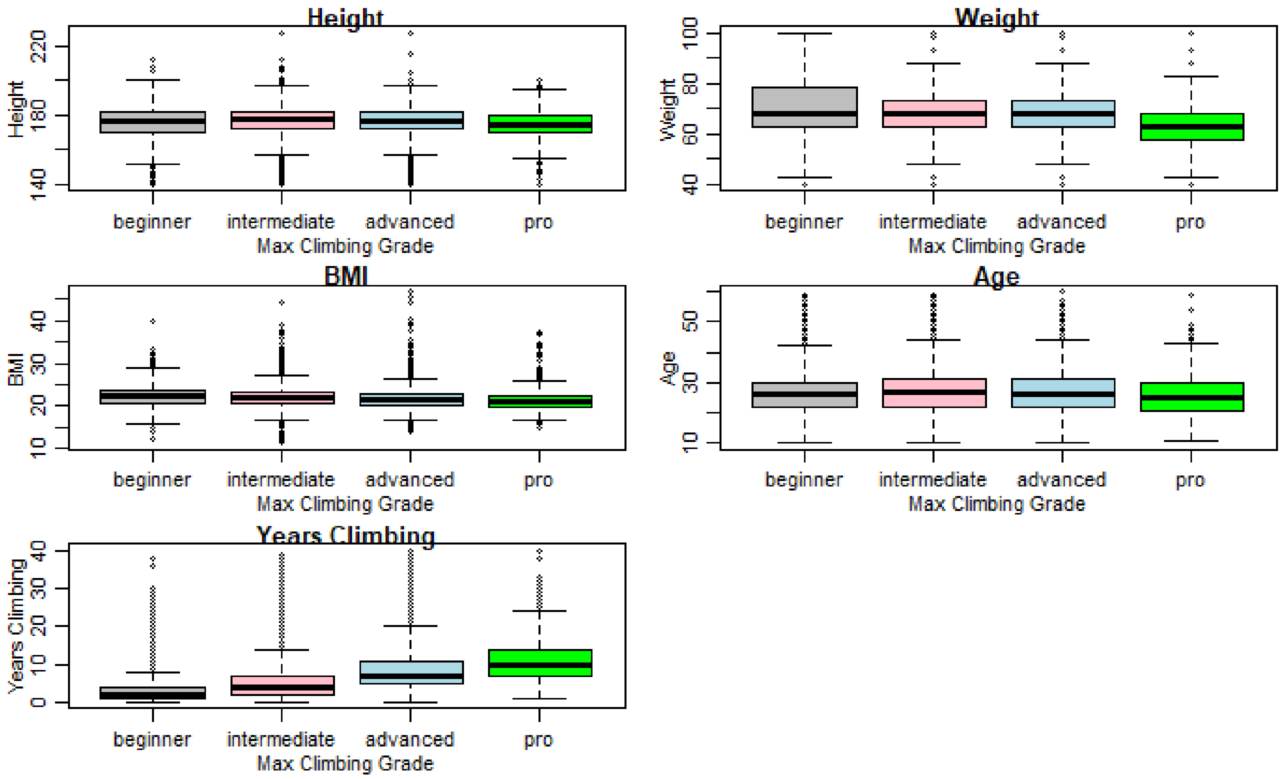
- ‘age’ and ‘years_climbing’ expressed in years. For this variables there were the most missing values. For simplicity only the climbers for which these variables were not null were selected.
- The variable ‘BMI’: weight/(height/100)^2 [kg/m^2] was added.

Except for the two dichotomous variables, the other variables were standardized.

After dropping the missing value rows and removing the outliers the dataset has 21290 rows.

Data Exploration

Boxplots were used for the visualization of the continous variables splitted in the four grade categories:

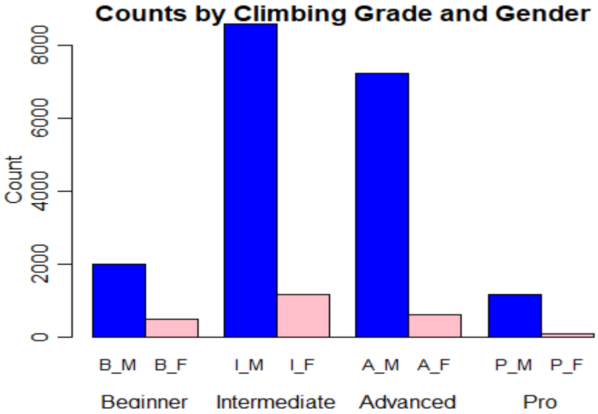


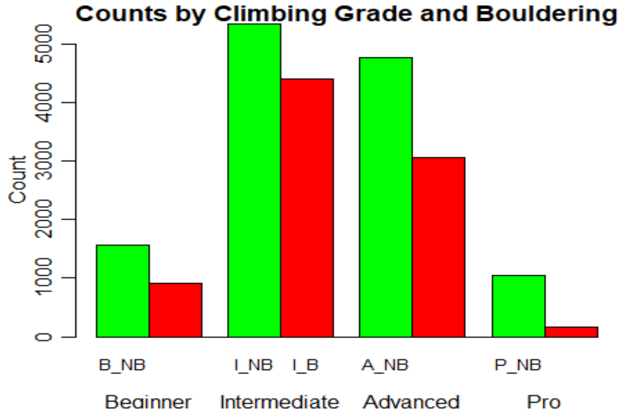
From a first sight it seems that there are no big differences between the four categories of climbers in terms of height, BMI and age, while pro-climber seems to be slightly lighter than the others. On the other end there is a positive correlation between years of climbing and maximum grade; this highlights the importance of experience in climbing.

For the dichotomous variables two histograms were produced:

First of all there are more male than female climber. The ratio between the gender is particularly big in the pro-climber group, suggesting that gender should influence the max grade climbed.

From the second graphs there are always more sport climbers than boulderers in all the groups and this is particularly true for the pros; this suggest that boulder grades are thougher and for the pro-group there is a higher level of specialization in one of the two disciplines.





The Gibbs Sampler

The implemented sampler is based on the model proposed by Albert and Siddharta (1993). Each observed variable $Y_1,...,Y_N$ ('max_climbing_grade'), where N = 1000, belongs to one of the J = 4 following ordered categories: 'beginner','intermediate','advanced','pro'.

Let pij = P[Y_i = j]; the cumulative probability is defined: $\eta_{ij} = \sum_{k=1}^{j} pik$

Then one popular regression model for {pij} is given by $\eta_{ij} = \Phi(\gamma_j - X_i^T \beta)$ with i = 1,...,N and j = 1,...,J-1.

The model is motivated by assuming the existence of N independent continuous random variable Z_i distributed $N(X_i^T \beta,1)$ and Y_i is observed when $\gamma_{j-1} < Z_i \leq \gamma_j$. In this model the regression vector β and the bin boundaries $\gamma_1,.., \gamma_{j-1}$ are unknown and is costumary to assign a flat noninformative prior to β. To ensure that the parameters are identifiable it is necessary to impose a restriction on the bin boundaries; without losing generality $\gamma_1=0$ is taken.

The implementation of the Gibbs sampler is made by simulate from the following full conditional distributions in this order:

- Γj given Z,γ, β and {γ_k with k ≠ j} is a uniform on the interval [max {max {Z_i:Y_i =j}, γ_{j-1}}, min {min {Z_i:Y_i =j + 1}, γ_{j-1}}].
- Z_i given β, γ and Y_i = j is $N(X_i^T \beta,1)$ truncated at the left by γ_{j-1} and at the right by γ_j
- B given γ,Z is distributed N_k(beta_hat_z, $(X^TX)^{-1}$) where beta_hat_z = $((X^TX)^{-1} (X^TZ)^{-1})$

Several other auxiliary functions were used in the implementation (see the complete code in the appendix)

The initial values of β and z are fixed small, while a good choice for the initial value of γ seems the following:

```
z_0 <- numeric(n)
```

```
gamma_0 = c(0,10,20)
```

```
beta_0 = rep(0.01,8)
```

G = 100000 is the number of iterations, burnin = 10000 and thinning = 100.

The results of the Gibbs sampling are saved in the matrix mat_gamma_thinned and mat_beta_thinned and were evaluated using mcmcplots library. Also for such high value of iterations the MC associated to β_1, γ_2 and γ_3 is not completely stationary and their acf are converging but do not reach 0 (the complete diagnostic of the MC is in the appendix)

Posteriors of the Predictors

In the next section are visualized the posterior density of all the beta and gamma coefficients, their median values and 95% posterior credible intervals.

```
The posterior median of beta[1] is 2.037351
The 95% posterior credible interval for beta[1] is (1.881077,2.229924)
The posterior median of beta[2] is -0.5554109
The 95% posterior credible interval for beta[2] is (-0.7005482,-0.4103863)
The posterior median of beta[3] is 0.5730131
The 95% posterior credible interval for beta[3] is (-0.007623444,1.182607)
The posterior median of beta[4] is -1.165576
The 95% posterior credible interval for beta[4] is (-2.027123,-0.2935609)
The posterior median of beta[5] is 0.7426267
The 95% posterior credible interval for beta[5] is (0.1213928,1.361621)
The posterior median of beta[6] is -0.4261984
The 95% posterior credible interval for beta[6] is (-0.5280433,-0.3292225)
The posterior median of beta[7] is 0.7772508
The 95% posterior credible interval for beta[7] is (0.6798635,0.8755803)
The posterior median of beta[8] is -0.4218506
The 95% posterior credible interval for beta[8] is (-0.7842498,-0.06301909)
```

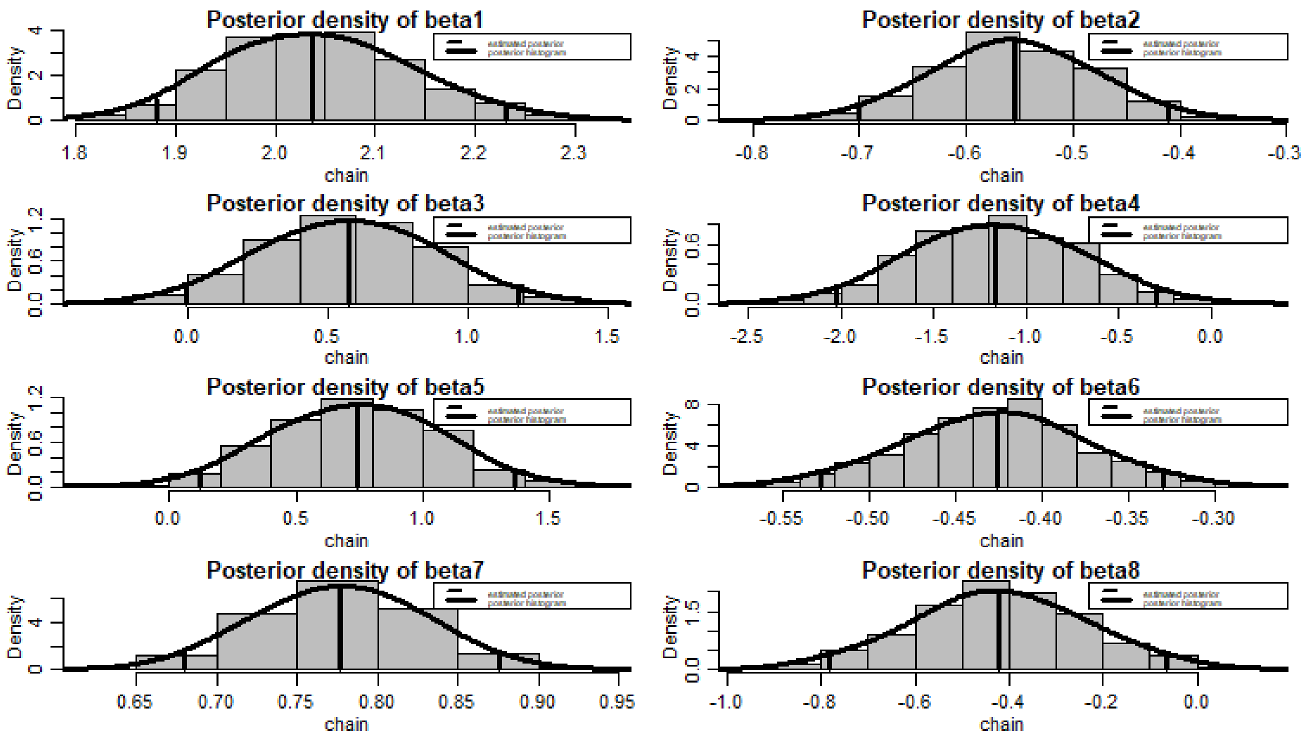
The posterior means for the beta and gamma are:

```
hat_beta <- apply(mat_beta_thinned,2,mean)
hat_gamma <- apply(mat_gamma_thinned,2,mean)
```

```
hat_beta
```

```
2.0419578 -0.5552113  0.5735952 -1.1600116  0.7411577 -0.4268442  0.7772476 -
0.4193687
```

```
hat_gamma
 0.000000 1.733900 3.457668
```



Model Evaluation

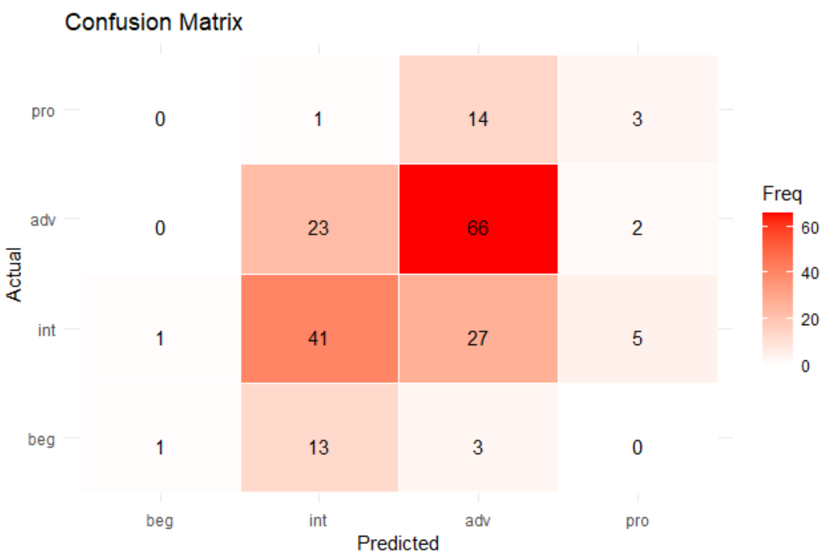
To evaluate the model a test set of 200 elements is extracted from the original dataset. A prevision function was implemented; given the posterior vector of predictors hat_gamma and hat_beta and the matrix X of observation,it returns the category for which each observation is more probable to belong, based on the model $\eta_{ij} = \Phi(\gamma_j - X_i^T \beta)$.

The accuracy of the model is given by:

```
accuracy = sum(y_pred == y_real)/nrow(X)*100
```

```
"accuracy: 55.5".
```

This is the confusion matrix: `conf_matrix = table(y_real,y_pred)`



For the ‘intermediate’ and ‘advanced’ category most observations are on the main diagonal, so they tend to be classified correctly, while almost all observation belonging to the ‘beginner’ and ‘pro’ categories are misclassified.

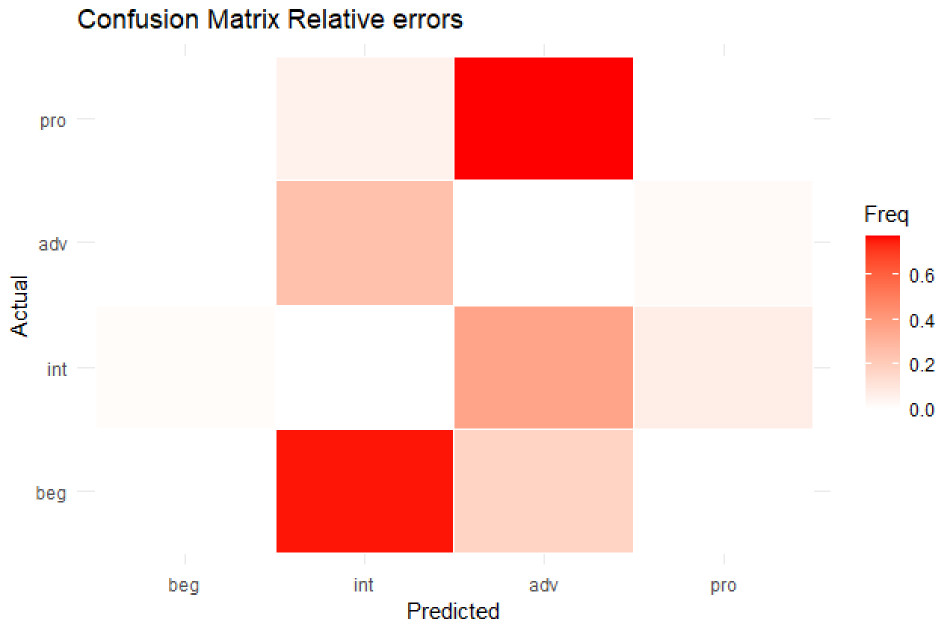
Let’s now focus on the error rate, so each value in the confusion matrix is divided by the number of observations in the corresponding class and the diagonal is filled with zeros.

```
row_sums <- rowSums(conf_matrix)

conf_matrix_normalized <- conf_matrix / row_sums[row(conf_matrix)]

diag(conf_matrix_normalized) = 0
```

```
      y_pred
y_real          1          2          3          4
     1 0.00000000 0.76470588 0.17647059 0.00000000
     2 0.01351351 0.00000000 0.36486486 0.06756757
     3 0.00000000 0.25274725 0.00000000 0.02197802
     4 0.00000000 0.05555556 0.77777778 0.00000000
```



The classifier tends to misclassify most beginner climbers as intermediate and most of pros as advanced. Also, some intermediates are classified as advanced and vice versa.

So, in general, the model tends to classify intermediate and advanced climbers better than beginners and pros. This can be caused by the fact that these two last categories are only a minority in the dataset and there are few data available. Another reason can be that in the dataset aren’t present important features that can be used to distinguish beginners and pros effectively, like level of training of the climber and frequency of outdoor climbing.

The effect of gender and weight

Height is fixed at 175cm, age at 26y and years of experience at 6y for non boulderer, while weight can vary from a grid of values from 43kg to 98kg and is_female is 0 or 1.

The linear predictor and the probability to belong to the advaced or pro categoris are:

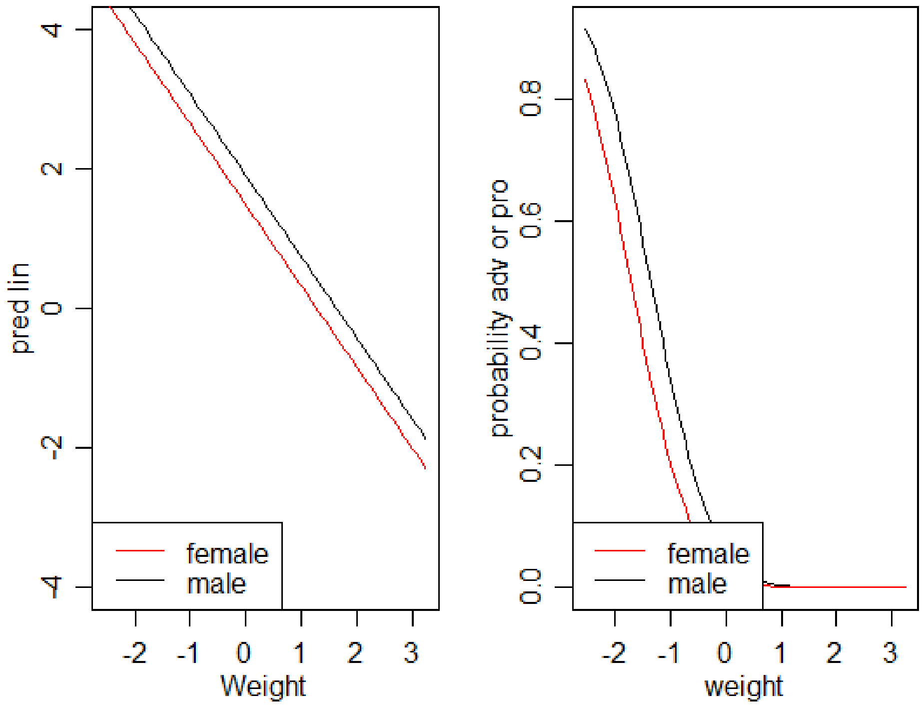
```
pred_lin_post_male = hat_beta[1] + hat_beta[2]*bould + hat_beta[3]*height +
hat_beta[4]*grid_weight + hat_beta[6]*age + hat_beta[7]*years_exp
```

```
pred_lin_post_female = hat_beta[1] + hat_beta[2]*bould + hat_beta[3]*height
+hat_beta[4]*grid_weight + hat_beta[6]*age + hat_beta[7]*years_exp + hat_beta[8]
```

```
p3_post_male = 1-pnorm(hat_gamma[2]-pred_lin_post_male)
```

```
p3_post_female = 1-pnorm(hat_gamma[2]-pred_lin_post_female)
```

Please note that the values on the x-axis are standardized:



The linear predictors are parallel lines and from the probability curves we can see that females tend to have a maximum climbed grade slightly lower than the male one and if the weight increase the probability of being an advanced or pro climber decrease.

The effect of bouldering and years of experience

Height is fixed at 175cm, weight at 73kg, BMI at 23.74 age at 26y for a male climber, while years of experience can vary from a grid of values from 0 to 30y and is_bouldering is 0 or 1.

The linear predictor and the probability to belong to the advanced or pro categories are:

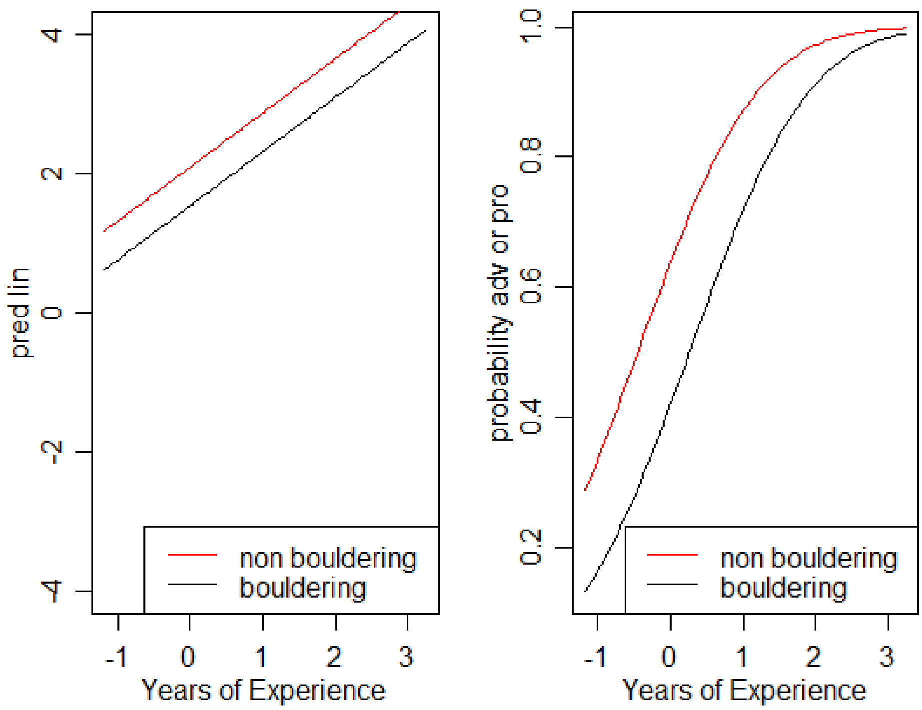
```
pred_lin_post_boul = hat_beta[1] + hat_beta[2] + hat_beta[3]*height +
hat_beta[4]*weight + hat_beta[5]*BMI + hat_beta[6]*age + hat_beta[7]*grid_exp
```

```
pred_lin_post_non_boul = hat_beta[1] + hat_beta[3]*height + hat_beta[4]*weight +
hat_beta[5]*BMI + hat_beta[6]*age + hat_beta[7]*grid_exp
```

```
p3_post_boul = 1-pnorm(hat_gamma[2]-pred_lin_post_boul)
```

```
p3_post_non_boul = 1-pnorm(hat_gamma[2]-pred_lin_post_non_boul)
```

Please note that the values on the x-axis are standardized:



The linear predictors are parallel lines and from the probability curves we can see that boulder grades are harder than sport climbing ones and the years of experience are a crucial factor for the maximum level reached.

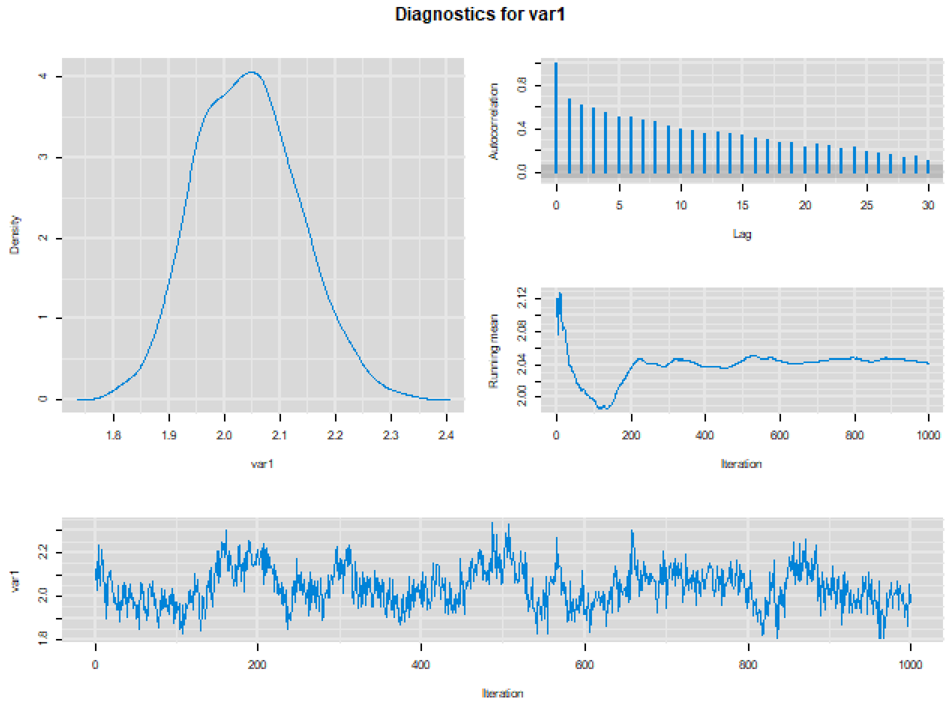
SOURCES

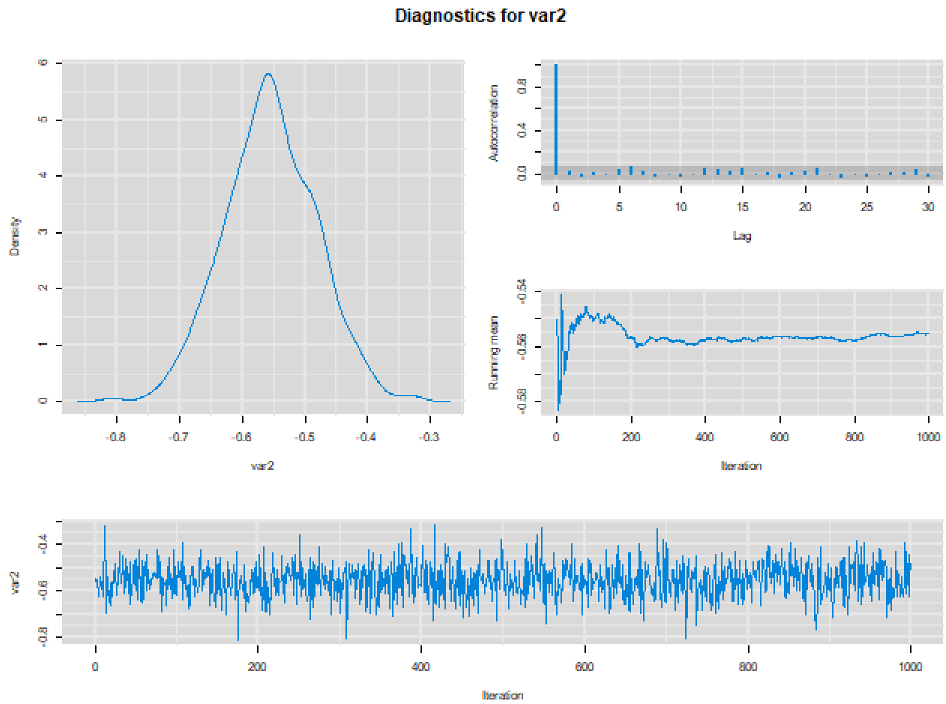
- Albert and Siddharta: 'Bayesian Analysis of Binary and Polychotomous Response Data'
- Prof. R.Argiento: notes of 'Applied Statistical Modelling' course
- A.Geron: Hands on Machine Learning
- Source of data: https://www.kaggle.com/datasets/jordizar/climb-datase

APPENDIX

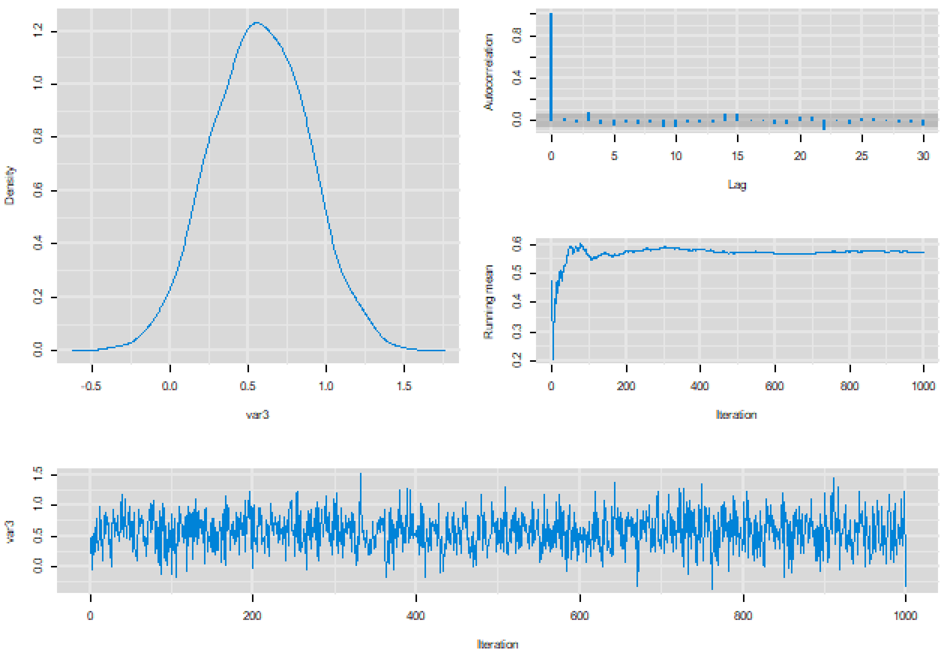
Diagnostic of the MC

Beta MC

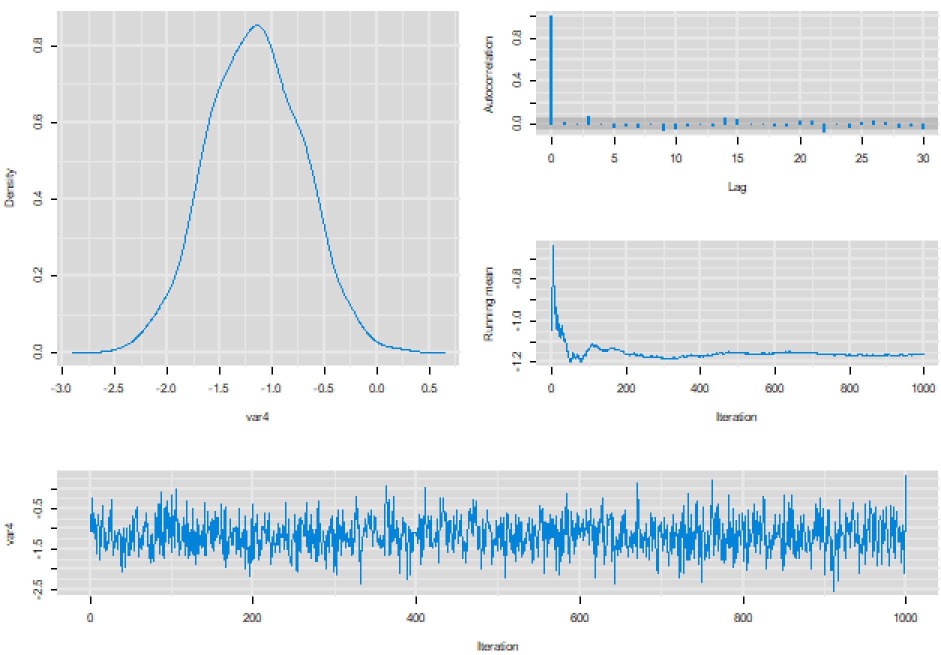




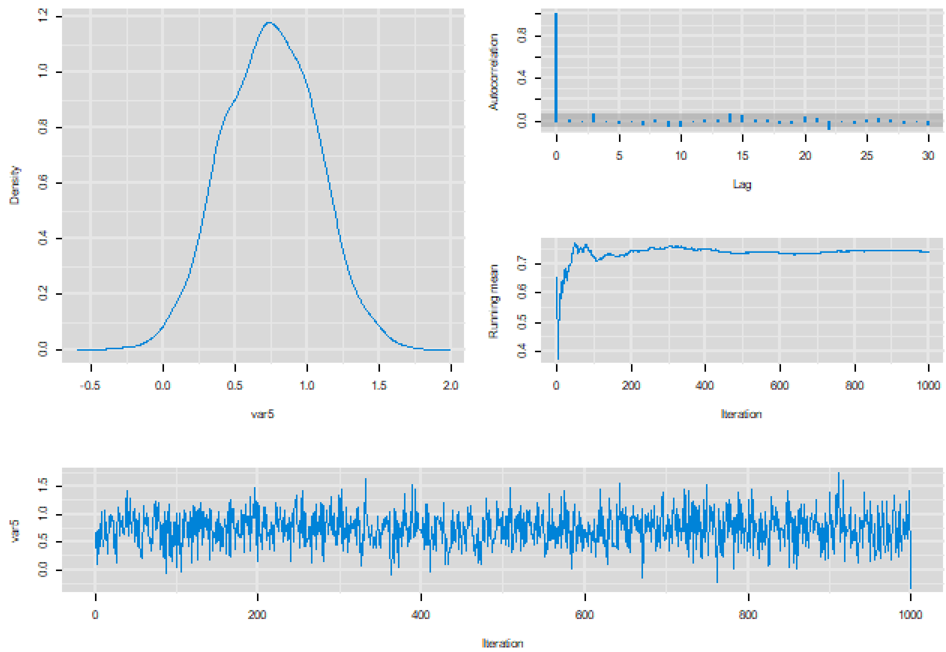
Diagnostics for var3



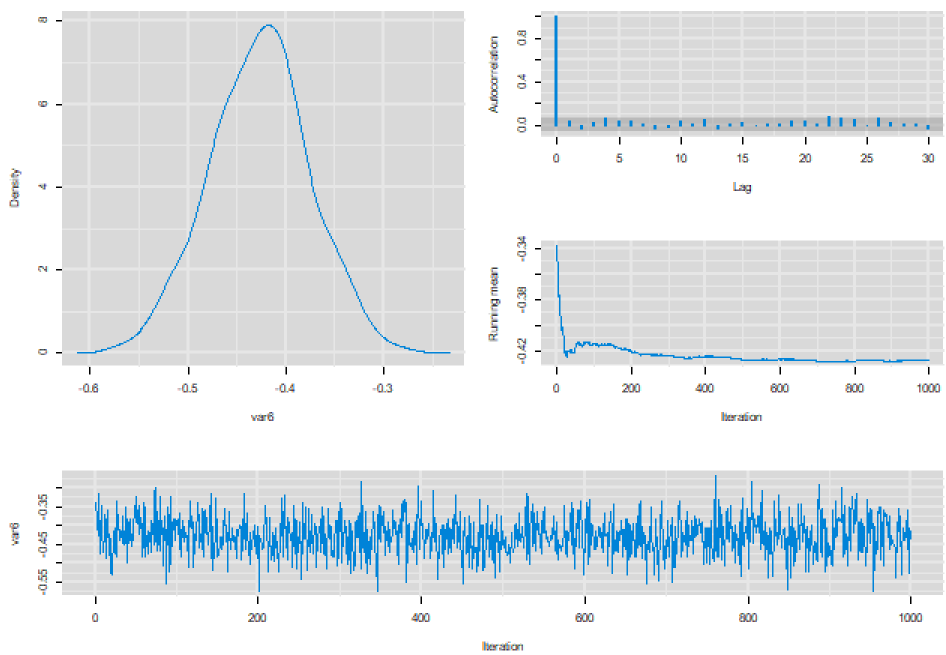
Diagnostics for var4



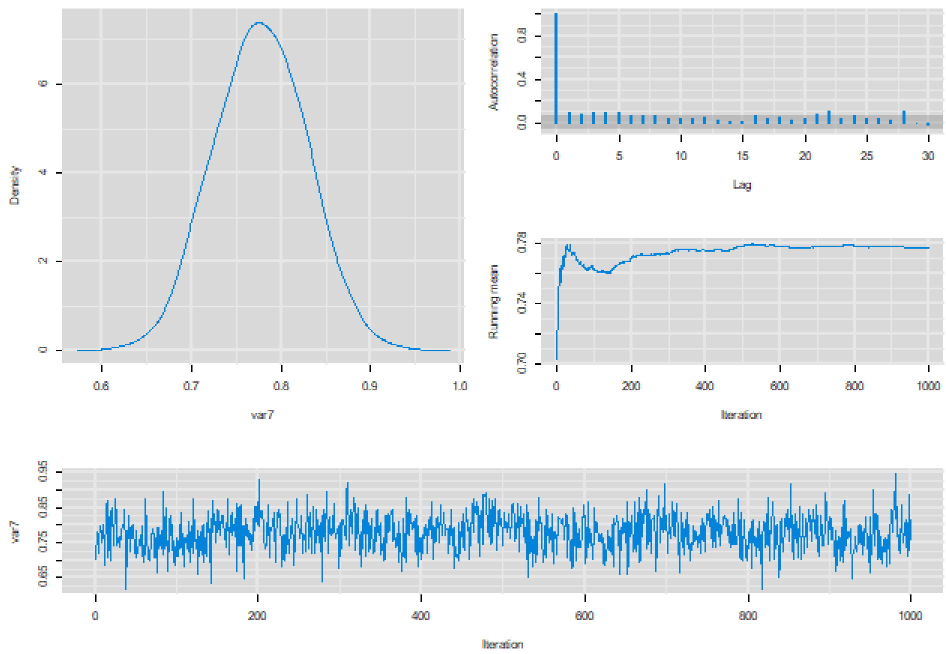
Diagnostics for var5



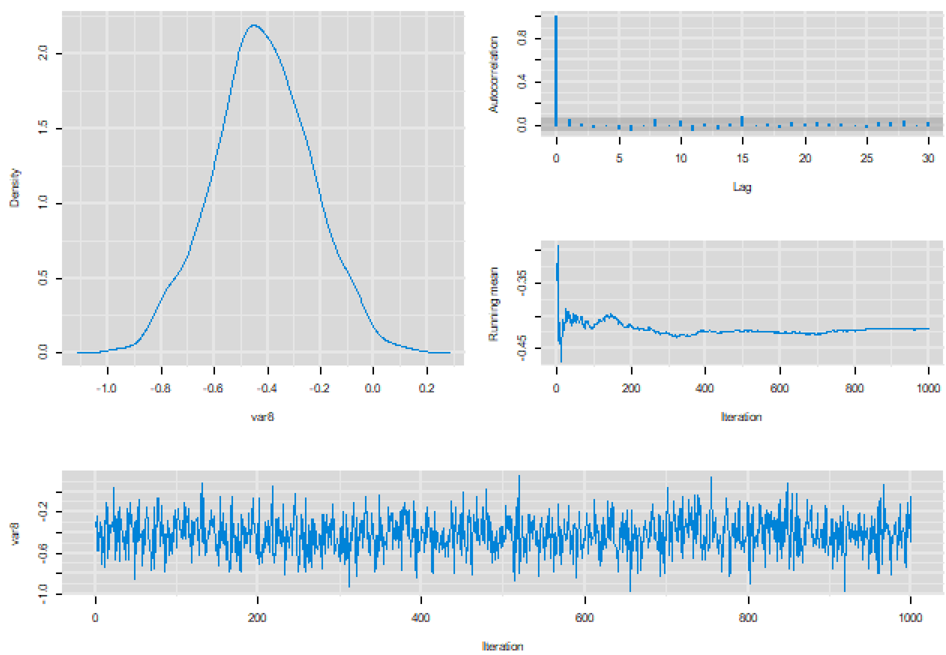
Diagnostics for var6



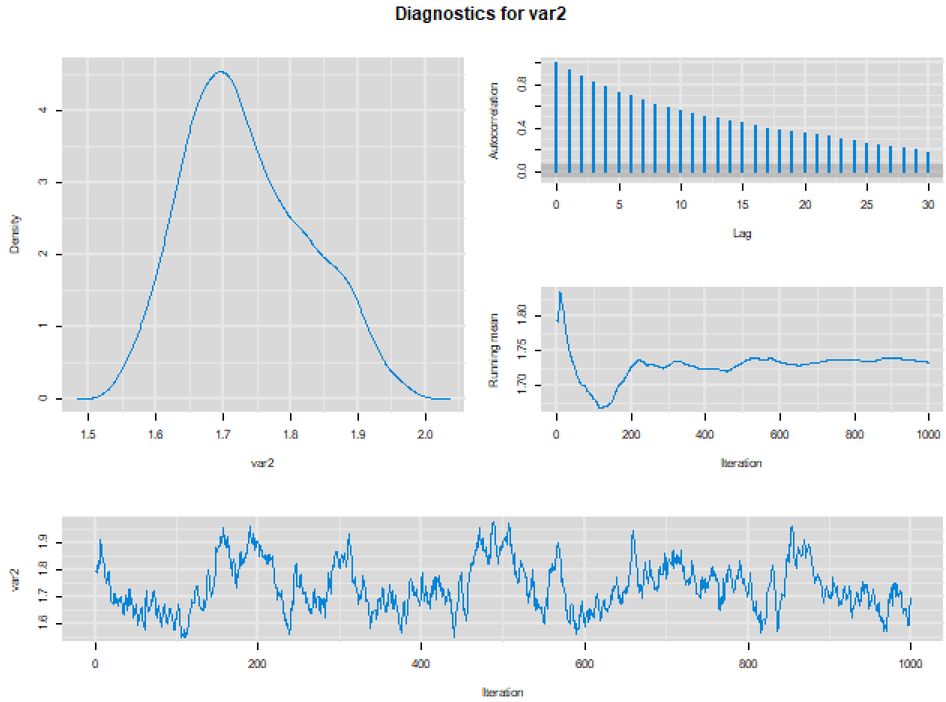
Diagnostics for var7



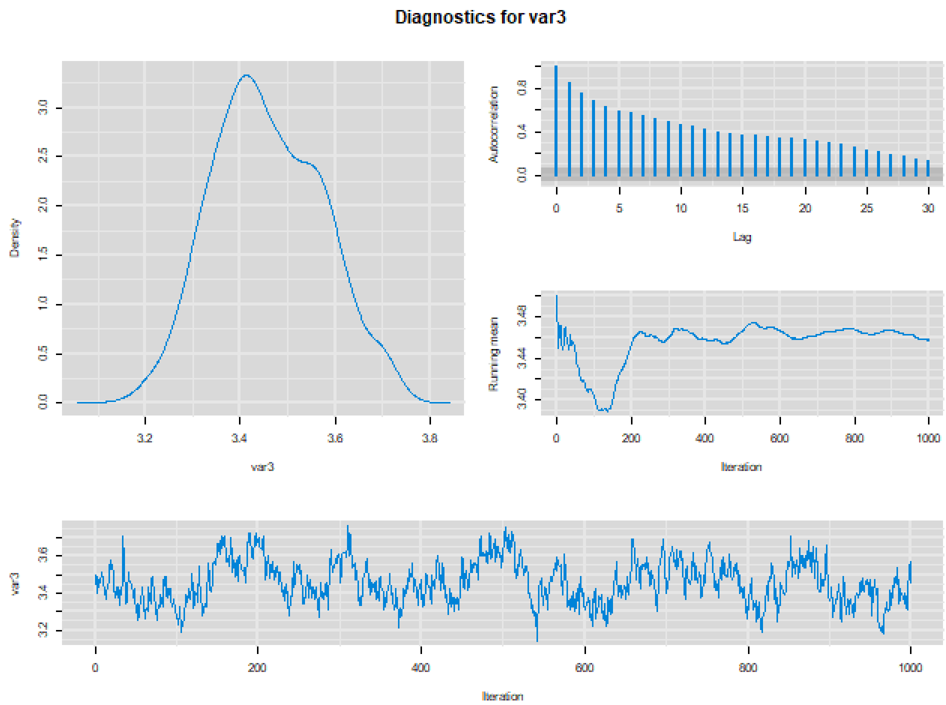
Diagnostics for var8



Gamma MC



Diagnostics for var2
Density
var2
Autocorrelation
Lag
Running mean
Iteration



Diagnostics for var3
Density
var3
Autocorrelation
Lag
Running mean
Iteration

CODE

```
library(mvtnorm)

library(truncnorm)

library(mcmcplots)
```

#DATASET

```
df_ready <- read.csv("C:/Users/Utente/Desktop/Climbing_pj/df_ready_std.csv")

df = df_ready[1:1000,]

x2 <- df$is_bouldering

x3 <- df$height

x4 <- df$weight

x5 <- df$BMI

x6 <- df$age

x7 <- df$years_climbing

x8 <- df$is_female

y <- df$max_climbing_grade

n <- length(y)

tilde_X <- model.matrix(y~x2+x3+x4+x5+x6+x7+x8)

inv_xt_x = solve(t(tilde_X)%*%tilde_X)

# Hyperparameters

z_0 <- numeric(n)

ind_cat <- numeric(n)

gamma_0 = c(0,10,20)

beta_0 = rep(0.01,8)

for (i in 1:n){

  if(y[i] == 'beginner'){

    z_0[i] = -5

  }

  else if(y[i] == 'intermediate'){

    z_0[i] = 5

  }

  else if(y[i] == 'advanced'){

    z_0[i] = 15

  }

  else{

    z_0[i] = 25

  }

}
```

#AUXILIARY FUNCTIONS

```
#Dato vettore z e vettore gamma (estremi categorie)

#ritorna vettore di categorie per ogni elemento di z

find_cat = function(){

  vett_out = numeric(n)

  for(i in 1:n){

    if(z_curr[i] < gamma_curr[1]){

      vett_out[i] = 1

    }

    else if(z_curr[i] < gamma_curr[2]){

      vett_out[i] = 2

    }

    else if(z_curr[i] < gamma_curr[3]){

      vett_out[i] = 3

    }

    else{

      vett_out[i] = 4

    }

  }

  return(vett_out)

}


#In: vettore di z e vettore gamma

#ritorna il massimo elemento di z appartenente a categoria cat

max_value = function(cat){

  max_val = -Inf

  for(i in 1:n){

    if(ind_curr[i] == cat){

      if(z_curr[i]>max_val){

        max_val = z_curr[i]

      }

    }

  }

  if(max_val == -Inf){

    if(cat == 2){

      max_val = gamma_curr[1]

    }

    else{
```

```
      max_val = gamma_curr[2]
    }
  }
  return(max_val)
}

min_value = function(cat){
  min_val = Inf #check
  for(i in 1:n){
    if(ind_curr[i] == cat){
      if(z_curr[i]<min_val){
        min_val = z_curr[i]
      }
    }
  }
  if(min_val == Inf){
    min_val = gamma_curr[3]
  }
  return(min_val)
}

estrai_gamma = function(){
  gamma_curr[1] = 0
  gamma_curr[2] = runif(1,min = max(max_value(2),gamma_curr[1]),max = min(min_value(3),gamma_curr[3]))
  gamma_curr[3] = runif(1,min = max(max_value(3),gamma_curr[2]),max = min_value(4))
  return(gamma_curr)
}

estrai_z = function(){
  vett_out = numeric(n)
  for(i in 1:n){
    m = tilde_X[i,]%*%t(beta_curr)
    if(ind_curr[i] == 1){
      vett_out[i] = rtruncnorm(1,a = -Inf,b = gamma_curr[1],mean = m,1)
    }
    else if(ind_curr[i] == 4){
      vett_out[i] =rtruncnorm(1,a = gamma_curr[3],b=Inf,mean = m,1)
    }
    else{
```

```
      vett_out[i] =rtruncnorm(1,a = gamma_curr[ind_curr[i]-1],b=gamma_curr[ind_curr[i]],mean = m,1)
      }
  }
  return(vett_out)
}

estrai_beta = function(){
  beta_hat = inv_xt_x %*% (t(tilde_X)%*%z_curr)
  vett_out = rmvnorm(1,beta_hat,inv_xt_x)
  return(vett_out)
}
```

#GIBBS SAMPLER

```
G = 100000
burnin = 10000
thinning = 100
n_iter = G + burnin
gamma_curr = gamma_0
z_curr = z_0
ind_cat = find_cat()
beta_curr = t(beta_0)
mat_gamma = matrix(nrow = n_iter+1,ncol = 3)
mat_z = matrix(nrow = n_iter+1,ncol = n)
mat_ind = matrix(nrow = n_iter+1,ncol = n)
mat_beta = matrix(nrow = n_iter+1,ncol = 8)
mat_gamma[1,] = gamma_0
mat_beta[1,] = beta_0
mat_z[1,] = z_0
mat_ind[1,] = ind_cat

for(i in 1:n_iter){
  ind_curr = find_cat()
  gamma_curr = estrai_gamma()
  z_curr = estrai_z()
  beta_curr = estrai_beta()
  mat_gamma[i+1,] = gamma_curr
  mat_z[i+1,] = z_curr
  mat_beta[i+1,] = beta_curr
```

```
  mat_ind[i+1,] = ind_curr}

# Scarto burn-in samples
mat_beta <- mat_beta[(burnin + 1):n_iter,]
mat_gamma <- mat_gamma[(burnin + 1):n_iter,]
# Thinning
mat_beta_thinned <- mat_beta[seq(from = 1, to = nrow(mat_beta), by = thinning),]
mat_gamma_thinned <- mat_gamma[seq(from = 1, to = nrow(mat_gamma), by = thinning),]

save(mat_beta_thinned, file = "mat_beta_thinned_file.RData")
save(mat_gamma_thinned, file = "mat_gamma_thinned_file.RData")

mcmcplot(as.mcmc(mat_beta_thinned))
mcmcplot(as.mcmc(mat_gamma_thinned))

 #PREVISION AND EVALUATION
hat_beta <- apply(mat_beta_thinned,2,mean)
hat_gamma <- apply(mat_gamma_thinned,2,mean)

prevision = function(){
  n = nrow(X)
  prev = numeric(n)
  for(i in 1:n){
    prob = numeric(4)
    for(j in 1:4){
      if(j == 2 | j == 3){
        prob[j] = pnorm(hat_gamma[j]-t(X[i,])%*%hat_beta) - pnorm(hat_gamma[j-1]-t(X[i,])%*%hat_beta)
      }
      else if(j == 1){
        prob[j] = pnorm(hat_gamma[j]-t(X[i,])%*%hat_beta)
      }
      else{
        prob[j] = 1 - pnorm(hat_gamma[3]-t(X[i,])%*%hat_beta)
      }
      prev[i] = which.max(prob)
    }
  }
  return(prev)
}
```

```
df_test = df_ready[1001:1200,]
x2 <- df_test$is_bouldering
x3 <- df_test$height
x4 <- df_test$weight
x5 <- df_test$BMI
x6 <- df_test$age
x7 <- df_test$years_climbing
x8 <- df_test$is_female
y_real <- df_test$max_climbing_grade

for(i in 1:length(y_real)){
  if(y_real[i] == 'beginner'){
    y_real[i] = 1
  }
  if(y_real[i] == 'intermediate'){
    y_real[i] = 2
  }
  if(y_real[i] == 'advanced'){
    y_real[i] = 3
  }
  if(y_real[i] == 'pro'){
    y_real[i] = 4
  }

}

X <- model.matrix(y_real~x2+x3+x4+x5+x6+x7+x8)
y_pred = prevision()

accuracy = sum(y_pred == y_real)/nrow(X)*100
print(paste('accuracy:',accuracy))

library(ggplot2)

conf_matrix = table(y_real,y_pred)
row_sums <- rowSums(conf_matrix)
conf_matrix_normalized <- conf_matrix / row_sums[row(conf_matrix)]
diag(conf_matrix_normalized) = 0
```

```
colnames(conf_matrix_normalized) <- c("beg", "int",'adv','pro')
rownames(conf_matrix_normalized) <- c("beg", "int",'adv','pro')
conf_df <- as.data.frame(as.table(conf_matrix_normalized))
 ggplot(data = conf_df, aes(x = y_pred, y = y_real, fill = Freq)) +
  geom_tile(color = "white")  +
  scale_fill_gradient(low = "white", high = "red") +
  theme_minimal() +
  labs(x = "Predicted", y = "Actual", title = "Confusion Matrix Relative errors")
```

#EFFECT EVALUATION

```
#WEIGHT AND GENDER
height = -0.163980
age = -0.143855
years_exp = -0.151480
bould = 0
range(x4)
grid_weight = seq(range(x4)[1],range(x4)[2],length.out = 100)

pred_lin_post_male = hat_beta[1] + hat_beta[2]*bould + hat_beta[3]*height +
  hat_beta[4]*grid_weight + hat_beta[6]*age + hat_beta[7]*years_exp
pred_lin_post_female = hat_beta[1] + hat_beta[2]*bould + hat_beta[3]*height +
  hat_beta[4]*grid_weight + hat_beta[6]*age + hat_beta[7]*years_exp + hat_beta[8]
p3_post_male = 1-pnorm(hat_gamma[3]-pred_lin_post_male)
p3_post_female = 1-pnorm(hat_gamma[3]-pred_lin_post_female)

par(mfrow=c(1,2),mar=c(3,3,1,1),mgp=c(1.75,.75,0))
plot(grid_weight,pred_lin_post_male,type="l",ylim=c(-4,4),xlab ="Weight",ylab="pred lin")
lines(grid_weight,pred_lin_post_female,type="l",col="red",xlab="Weight",ylab="pred lin")
legend("bottomleft",c("female","male"),
       col=c("red","black"),lty=c(1,1))
plot(grid_weight,p3_post_male,type="l", xlab ="weight", ylab="probability adv or pro")
lines(grid_weight,p3_post_female,type="l", xlab ="weight", ylab="probability adv or pro",col="red")
legend("bottomleft",c("female","male"),
       col=c("red",'black'),lty=c(1,1))

#Bouldering and years of experience
height = -0.163980
weight = 0.502524
```

```
BMI =0.888000

age = -0.143855

fem = 0

range(x7)

grid_exp = seq(range(x7)[1],range(x4)[2],length.out = 100)


pred_lin_post_boul = hat_beta[1] + hat_beta[2] + hat_beta[3]*height +

  hat_beta[4]*weight + hat_beta[5]*BMI + hat_beta[6]*age + hat_beta[7]*grid_exp

pred_lin_post_non_boul = hat_beta[1] + hat_beta[3]*height +

  hat_beta[4]*weight + hat_beta[5]*BMI + hat_beta[6]*age + hat_beta[7]*grid_exp

p3_post_boul = 1-pnorm(hat_gamma[2]-pred_lin_post_boul)

p3_post_non_boul = 1-pnorm(hat_gamma[2]-pred_lin_post_non_boul)


par(mfrow=c(1,2),mar=c(3,3,1,1),mgp=c(1.75,.75,0))

plot(grid_exp,pred_lin_post_boul,type="l",ylim=c(-4,4),xlab ="Years of Experience",ylab="pred lin")

lines(grid_exp,pred_lin_post_non_boul,type="l",col="red",xlab="Years of Experience",ylab="pred lin")

legend("bottomright",c("non bouldering","bouldering"),

       col=c("red","black"),lty=c(1,1))

plot(grid_exp,p3_post_boul,type="l", xlab ="Years of Experience", ylab="probability adv or pro")

lines(grid_exp,p3_post_non_boul,type="l", xlab ="Years of Experience", ylab="probability adv
pro",col="red")

legend("bottomright",c("non bouldering","bouldering"),

       col=c("red",'black'),lty=c(1,1))
```