Ordinal probit regression

Leonardo Puricelli, Sangalli Chiara

Contents

Exploratory data analysis **1**

Methodology **4**
Latent variable . . . 4
Likelihood and parameters distributions . . . 4

Implementation **5**
Pseudo-code . . . 5
Code . . . 6

Analysis of the results **8**
Diagnostics . . . 8
Posterior estimates of the coefficients . . . 11

References **11**

Exploratory data analysis

The goal of this project is to implement a probit regression for an ordinal response variable with more than 2 levels.

The analysis will be performed on a dataset related to red variants of the Portuguese "*Vinho Verde*" wine, which contains 1599 observations of 11 predictors describing the amount of various chemicals present in the wine and the quality of the wine itself; 9 covariates are continuous, 2 are discrete and the response **quality** is an ordinal discrete variable that can assume values from 1 to 10. Table 1 shows the first 6 observations of our datasets and the values for each predictors, which have all been scaled.

Table 2 contains the frequency of each level of **quality**: as we can see the most extreme levels (1,2 and 9,10) never appear in our dataset, and the majority of the observations has quality 5 or 6. The same results is graphically displayed in Figure 1; hence our GLM will have as response a variable with 6 ordinal categories: $Y = \{3, 4, 5, 6, 7, 8\}$.

The boxplots in figure 2 and 3 show the relationship between some variables and **quality**. Via the general direction of the boxes, it can be estimated that there are positive correlations between the features **fixed.acidity**, **alcohol**, **sulphates** and the response (Figure 2), while **volatile.acidity**, **pH** and **density** seem to be negatively correlated to the target (Figure 3). The remaining predictors, **citric.acid**, **chlorides**, **residual.sugar, free.sulfur.dioxide** and **total.sulfur.dioxide** do not show a strong correlation with the response. In general, the significant number of outliers about **quality** levels 5 and 6 remarks once again that these are the most frequent values.

Table 1: Dataset

	1	2	3	4	5	6
fixed.acidity	-0.528	-0.298	-0.298	1.654	-0.528	-0.528
volatile.acidity	0.962	1.967	1.297	-1.384	0.962	0.738
citric.acid	-1.391	-1.391	-1.186	1.484	-1.391	-1.391
residual.sugar	-0.453	0.043	-0.169	-0.453	-0.453	-0.524
chlorides	-0.244	0.224	0.096	-0.265	-0.244	-0.265
free.sulfur.dioxide	-0.466	0.872	-0.084	0.108	-0.466	-0.275
total.sulfur.dioxide	-0.379	0.624	0.229	0.411	-0.379	-0.197
density	0.558	0.028	0.134	0.664	0.558	0.558
pH	1.288	-0.720	-0.331	-0.979	1.288	1.288
sulphates	-0.579	0.129	-0.048	-0.461	-0.579	-0.579
alcohol	-0.960	-0.585	-0.585	-0.585	-0.960	-0.960
quality	5.000	5.000	5.000	6.000	5.000	5.000

Table 2: Quality frquency table

3	4	5	6	7	8
0.0062539	0.0331457	0.4258912	0.3989994	0.1244528	0.011257

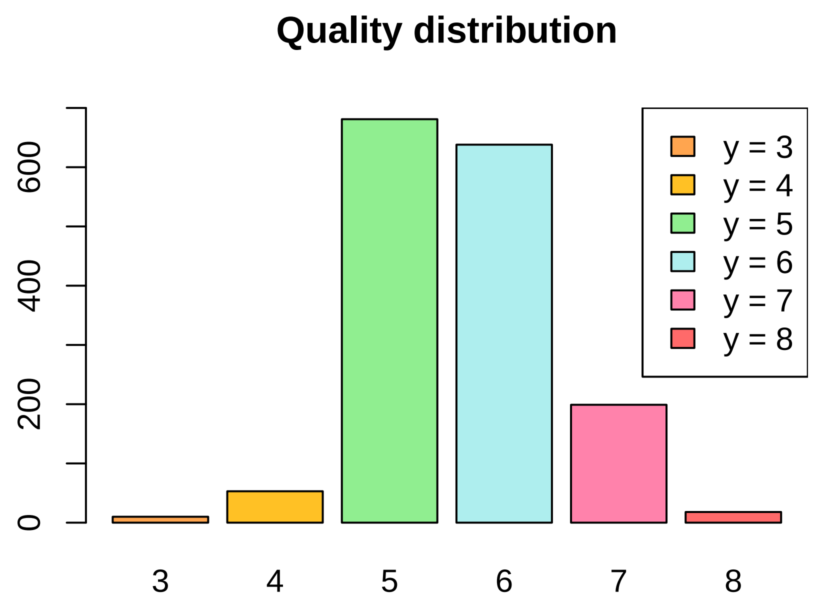

Figure 1: Distribution of the response

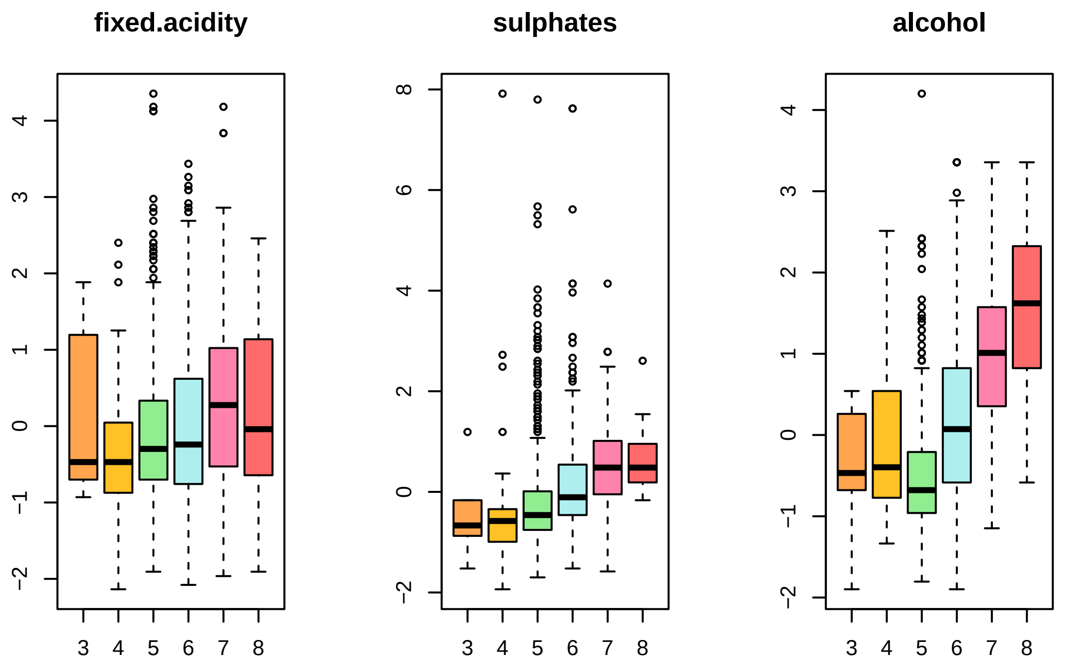

Figure 2: Boxplots showing positive correlation

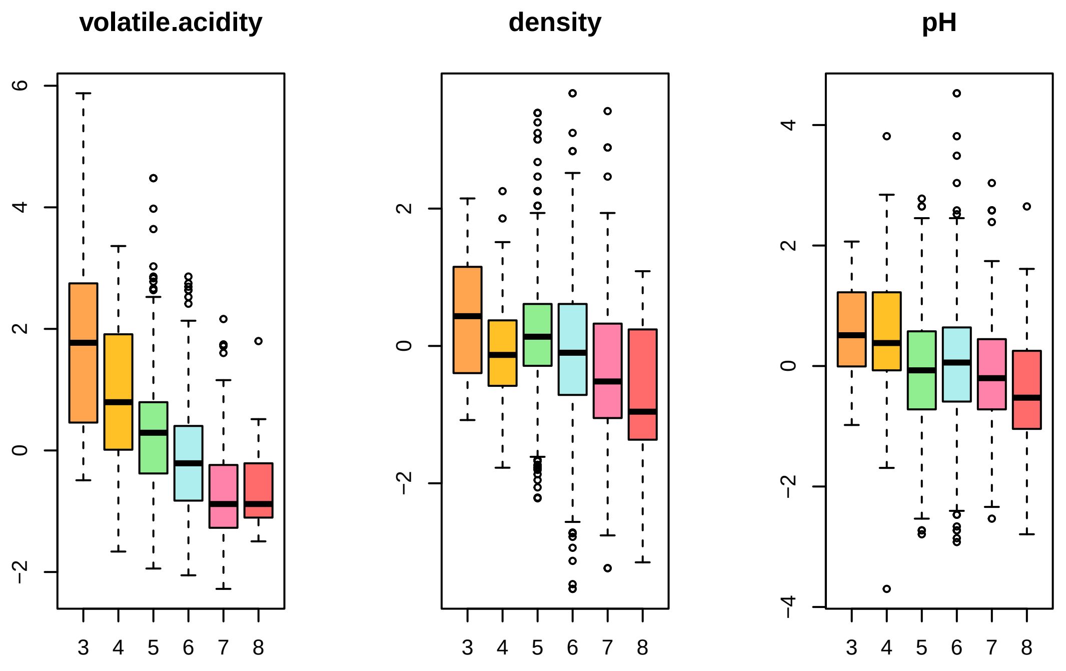

Figure 3: Boxplots showing negative correlations

Methodology

Latent variable

To study the relationship between our ordinal response and the predictors, we implement an ordinal probit regression, in which the link function is the inverse *c.d.f.* of a Standard Normal. In our case each value of the response Y_i takes one of K ordered categories, with $K = \{3, 4, 5, 6, 7, 8\}$.

Ordinal regression models are often specified in terms of cumulative probabilities: letting $p_{ik} = P[Y_i = k]$ being the probability of the i^{th} observation taking one of the possible values of K, we define the cumulative probability $\eta_{ik} = \sum_{k=1}^{K} p_{ik}$ to be the probability that the i^{th} observation is placed in category k or below. A well known probit regression model for p_{ik} is the one proposed by S. E. Fienberg, D. Lievesley and J. Rolph in the book "*Statistics for Social Science and Public Policy*" which is characterized by the following link function:

$$\eta_{ik} = \Phi(\theta_j - x_i^T \beta)$$

with $j = 0, \dots K$ and $i = 1, .., n$; $\underline{\theta} = \{\theta_0, ..., \theta_6\}$ is the vector of cut-offs.

This model is developed by assuming the existence of a latent continuous random variable $Z_i \sim N(x_i^T \beta, 1)$ that underlies the generation of the ordinal response and it is linked with **quality** as follows:

$$Y = \begin{cases} 3 & if \quad \theta_0 < z < \theta_1 \\ 4 & if \quad \theta_1 < z < \theta_2 \\ 5 & if \quad \theta_2 < z < \theta_3 \\ 6 & if \quad \theta_3 < z < \theta_4 \\ 7 & if \quad \theta_4 < z < \theta_5 \\ 8 & if \quad \theta_5 < z < \theta_6 \end{cases}$$

To match up the values taken by the response and the indexes of the cut-offs, for each y_i the corresponding lower and upper bounds are respectively $(y_i - 3)$ and $(y_i - 2)$. When Z_i falls between θ_{k-3} and θ_{k-2} the observation of the response is classified into category k. We fixed $\theta_0 = -\infty$ and $\theta_6 = \infty$, so our cut-offs vector can be rewritten as: $\underline{\theta} = (-\infty, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5, +\infty)$: the remaining 5 θ_j are unknown parameters of the regression.

To link this model to the probability that a wine receives a particular quality rate k, we can compute p_{ik} as follows:

$$p_{ik} = Pr[\theta_{k-3} < Z_i < \theta_{k-2}] = \Phi(\theta_{k-2} - x_i^T \beta) - \Phi(\theta_{k-3} - x_i^T \beta)$$

Likelihood and parameters distributions

Assuming that each response is independent from all the other $n - 1$ quality rates contained in the dataset, we can write the Likelihood as:

$$L(\underline{\beta}, \underline{\theta}) = \prod_{i=1}^{n} [\Phi(\theta_{k-2} - x_i^T \beta) - \Phi(\theta_{k-3} - x_i^T \beta)]$$

If we parameterize the model using the latent variable Z with β and θ we can derive the augmented likelihood as a function of all these 3 vectors of unknown parameters:

$$L(\underline{\beta}, \underline{\theta}, \underline{Z}) = \prod_{i=1}^{n} f(Z_i - x_i^T \beta) I(\theta_{k-3} < Z_i < \theta_{k-2})$$

The latent variable Z_i may be integrated out to obtain the first likelihood we wrote.

Before introducing the posterior distribution, we need to set up prior distributions for each parameters involved:

- $\underline{\beta} \sim N_p(\underline{\beta}_0, V_0^{-1})$: the vector of coefficients is distributed as multivariate normal with mean $\underline{\beta}_0$ and variance-covariance matrix V_0.
- $\underline{Z}|\underline{\beta} \sim N(X\underline{\beta}, 1)$: Z follows a normal distribution with mean equal to $X\underline{\beta}$ and variance 1.
- $p(\theta_k) \propto 1$: the cut-offs follow an improper constant prior.

The joint posterior density of the latent variable and model parameters is given by:

$$p(\beta, \theta, Z|Y) \propto L(\beta, \theta, Z)p(\beta)p(\theta)p(Z|\beta)$$

To make inference on this joint posterior we can exploit the full conditional distributions of the three parameters:

- the full conditional of β , $p(\beta|\theta, Z, Y)$ has the same distribution that we find in the binary probit regression studied in class: $\beta|\theta, Z, Y \sim N_p(\tilde{\beta}, \tilde{V}^{-1})$, where $\tilde{\beta} = (X^T X + V)^{-1}(X^T Z + V\beta_0)$ and $\tilde{V} = (X^T X + V)$;
- $p(z_i|\beta, \theta, y) \sim tN(\beta^T x_i, 1, \theta_{k-3}, \theta_{k-2})$: each z_i is distributed as a $Normal(\beta^T x_i, 1)$ truncated at the interval $[\theta_{k-3}; \theta_{k-2}]$;
- the full conditional for θ can be evaluated as $p(\theta|\beta, Z, y) \propto L(\beta, \theta, Z)p(\theta)$, but given that $p(\theta) \propto 1$, its conditional is proportional just to the likelihood: $p(\theta|\beta, Z, y) \propto \prod_{i=1}^{n}[\Phi(\theta_{k-2} - x_i^T\beta) - \Phi(\theta_{k-3} - x_i^T\beta)]$

Since only the full conditionals of β and Z are known distributions from which it's easy to draw, to sample from the posterior we can implement an hybrid Gibbs/Metropolis Hastings sampler in this way: a first MH step to retrieve a value of θ followed by a Gibbs step where we draw z and β from their full conditionals. Given that our goal is to approximate the distribution of the coefficients, at each iteration we store just the values of β.

Implementation

Pseudo-code

To approximate the posterior distribution of our 11 regression coefficients, we implement the following algorithm:

First we initialize vectors $\underline{\beta}^{(0)} = (\beta_1^{(0)}, \ldots, \beta_p^{(0)})$, $\underline{\theta}^{(0)} = (\theta_1^{(0)}, \ldots, \theta_{k-1}^{(0)})$ with p and k representing the number of predictors and number of categories; we also set the hyperparameters for the prior of β: β_0 is a vector of length p containing a sequence of 0 and V is a diagonal $(p * p)$ matrix with variances all equal to 0.01

For $s = 1, ..., S$:

1. MH step:
   - for $j = 1, \ldots, K-1$ propose g_j from a $Normal(\theta_j^{(s-1)}, 0.05)$ truncated at the interval $(g_{j-1}, \theta_{j+1}^{(s-1)})$; doing so we are sure that the elements of the updated vector of cut-offs maintain an increasing order. At the end of this step we will have a vector of 5 proposed values $\underline{g}$. Since we fixed the first and the last values of $\underline{\theta}$ to be respectively $\theta_0 = -\infty$ and $\theta_6 = \infty$, the MH algorithm will work to update the 5 remaining cut-offs.
   - Compute the acceptance ratio as follows:

$$r = \prod_{i=1}^{n} \frac{\Phi(g_{y_i-2} - x_i^T\beta^{(s-1)}) - \Phi(g_{y_i-3} - x_i^T\beta^{(s-1)})}{\Phi(\theta_{y_i-2}^{(s-1)} - x_i^T\beta^{(s-1)}) - \Phi(\theta_{y_i-3}^{(s-1)} - x_i^T\beta^{(s-1)})} \text{x} \prod_{j=1}^{K-1} \frac{\Phi((\theta_{j+1}^{(s-1)} - \theta_j^{(s-1)})/0.05) - \Phi((g_{j-1} - \theta_j^{(s-1)})/0.05)}{\Phi((g_{j+1} - g_j)/0.05) - \Phi((\theta_{j-1}^{(s-1)} - g_j)/0.05)}$$

Table 3: Initial cut-off values

	3|4	4|5	5|6	6|7	7|8	
-Inf	-3.018159	-2.18595	-0.1553253	1.468789	3.023089	Inf

The first term represents the ratio between the full conditional of θ evaluated at g and the same distribution evaluated at $\theta^{(s-1)}$, while the second is the correction term added because of the asymmetry of the proposal distribution

- Set $\underline{\theta}^{(s)} = \underline{g}$ with probability $min(r, 1)$, otherwise take $\underline{\theta}^{(s)} = \underline{\theta}^{(s-1)}$ with probability $(1 - min(r, 1)$

2. Gibbs for Z: draw a sample $(z_1, \ldots, z_n)$ from its full conditional $\sim tN(\beta^{(s-1)T} x_i, 1, \theta^{(s)}_{k-3}, \theta^{(s)}_{k-2})$
3. Gibbs for β: sample $(\beta_1^{(s)}, \ldots . \beta_p^{(s)})$ from $\underline{\beta}^{(s)} \sim N_p((X^T X + V)^{-1}(X^T Z^{(s)} + V \underline{\beta}_0), (V + X^T X))$ and store the sampled values of the coefficients in a matrix $(S * p)$

In the correction term of the acceptance ratio we divided each proposal by 0.05: this is the arbitrary variance value that we choose to implement our algorithm; this value should be changed whenever the resulting value of r_j falls out of the interval [0.25, 0.5].

Code

Firstly we define the following quantities from our dataset **wine.**

```r
yN = wine$quality                 # numeric response values
yF = as.factor(wine$quality)      # factor response values
X  = as.matrix(wine[,c(1:11)])    # data matrix
p  = ncol(X)                      # number of predictors
n  = nrow(X)                      # number of observations
K  = 6                            # number of categories
```

To initialize $\underline{\beta}^{(0)}$ we fit an ordinal probit regression model with the function ***polr()*** and use the frequentist estimates of the coefficients as starting values. This function requires the response to be a factor, so we consider **yF**.

```r
set.seed(123)
out_glm    = polr(yF ~  ., method="probit", data=X)        # frequentist model
beta.zero = out_glm$coefficients                  # frequentist estimates of beta
```

From the model fitted above we pull also out the starting values of $\underline{\theta}^{(0)}$ and we store them in **theta.zero.glm**, a vector of length 5 containing the cut-offs that we will update in our algorithm. Due to computational reasons, we add $\theta_0 = -\infty$ and $\theta_6 = +\infty$ as first and last cut-offs, so our vector of initial values is **theta.zero** represented in Table 3.

```r
theta.zero.glm = out_glm$zeta                # frequentist estimates of theta
theta.zero     = sort(c(theta.zero.glm,-Inf,Inf)) # ordered vector of thetas
```

Figure 4 shows the density of a $Normal(X\underline{\beta}^{(0)}, 1)$ truncated at the 5 cut-offs **theta.zero.glm**: for example we know that, if z_i falls into the green area, the level of **quality** of y_i will be equal to 5.

To effectively implement the algorithm we also initialized the number of iterations S, a vector g to store at each step the proposal values for $\underline{\theta}$, **beta.post**, a matrix of dimensions $(S * p)$ to store the sampled values for our 11 posterior coefficients β, and the prior hyper-parameters of the full conditional of β.

```r
S = 2000                        # number of iterations

g = c(-Inf,0,0,0,0,0,Inf)      # vector to store the proposal for theta
beta.post = matrix(NA, S, p) # matrix to store the posterior betas
```

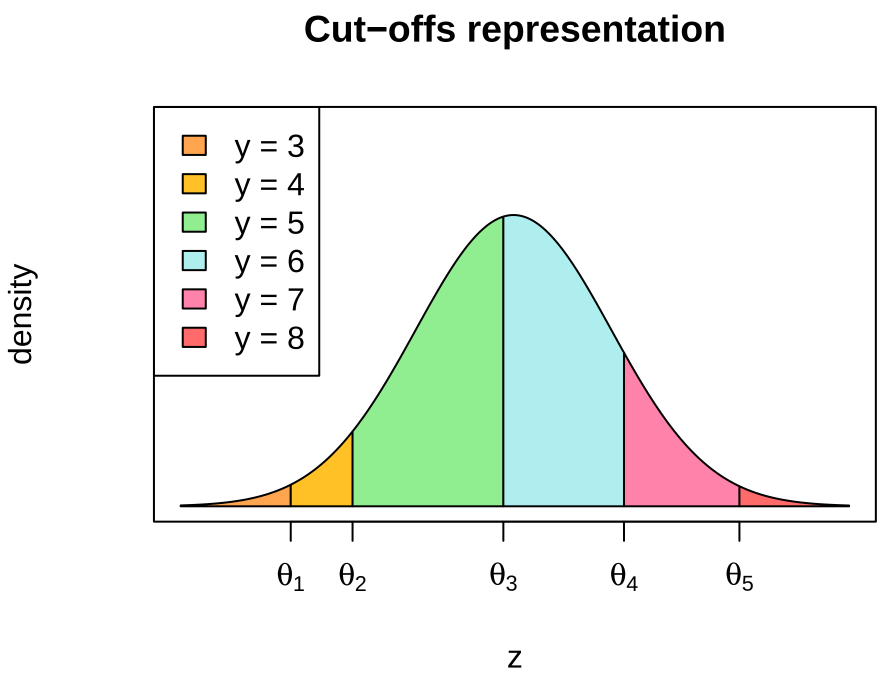

Figure 4: Representation of cut-offs

```
theta = theta.zero              # cut-offs vector updated at each iteration

beta  = beta.zero               # betas's vector updated at each iteration

beta.0 = rep(0, p)              # prior mean of betas
V = diag(0.01, p)               # prior variance-covariance matrix of betas
```

To implement the algorithm, we follow all the steps illustrated before, with some computational "tricks": to easily compute the acceptance ratio r we split it in four components (**num1**, **num2**, **den1**, **den2**) and to replace products and ratios with sums and differences, we take the logarithm of these quantities. Moreover, to avoid 0 in the components of the ratio (that would be NA in logarithmic scale) we replace all the null values with the middle value between the maximum and the minimum of each component.

To accept or reject the values proposed in $\underline{g}$ we compare r with the logarithm of a random draw u from $U \sim Unif(0,1)$.

We need also to specify that the indexes of $\underline{\theta}$ and $\underline{g}$ used in the pseudo-code are different from the ones implemented below: this mismatch is due to the fact that in our algorithm the indexes of the cut-offs goes from 0 to 6, while in the **R** language the positions of a vector of length 7 are selected with indexes from 1 to 7. For example θ_2 is accessed by $\theta[3]$.

After S iterations **beta.post** will be filled with approximated values from the posterior distribution of the coefficients.

```
set.seed(123)

########## GIBBS-MH ##########
```

```r
for (s in 1:S){

  ### MH for theta ###

  for (j in 2:K){
    g[j] = rtruncnorm(1,mean=theta[j],sd=0.5,a = g[j-1],b=theta[j+1])
  }

  num1 = pnorm(g[yN-1]-X%*%beta) - pnorm(g[yN-2]-X%*%beta)
  den1 = pnorm(theta[yN-1]-X%*%beta) - pnorm(theta[yN-2]-X%*%beta)
  num2 = pnorm((theta[3:7]-theta[2:6])/0.5) - pnorm((g[1:5]-theta[2:6])/0.5)
  den2 = pnorm((g[3:7]-g[2:6])/0.5) - pnorm((theta[1:5]-g[2:6])/0.5)

  for (i in 1:n){
    if(num1[i]==0){num1[i]=(max(num1)+min(num1)/2)}
    if(den1[i]==0){den1[i]=(max(den1)+min(den1)/2)}
    if(num2[min(i,5)]==0){num2[i]=(max(num2)+min(num2)/2)}
    if(den2[min(i,5)]==0){den2[i]=(max(den2)+min(den2)/2)}
  }

  r = sum(log(num1)) - sum(log(den1)) + sum(log(num2)) - sum(log(den2))
  u = runif(1,0,1)

  if(r>log(u)){theta <- g}

  ### Gibbs for zeta ###

  z = rtruncnorm(n, mean=X%*%beta, sd=1, a=theta[yN-2], b=theta[yN-1])

  ### Gibbs for beta ###

  V.n    = V + t(X)%*%X
  beta.n = solve(V + t(X)%*%X)%*%(t(X)%*%z + V%*%beta.0)
  beta   = c(rmvnorm(1,beta.n,solve(V.n)))

  beta.post[s,] = beta
}
```

To obtain the posterior estimates of the coefficients we just take the mean of each column of **beta.post**.

```r
beta.post.values <- colMeans(beta.post) # estimated posterior coefficients
```

Analysis of the results

Diagnostics

Before analyzing the posterior coefficients, we run a diagnostic on our chain to see whether it converges and has no autocorrelation problems.

Figure 5 shows traceplots for each posterior β distribution: it's immediate to see that all of them are approximating very well the estimated values of coefficients, represented with a red line.

Figure 6 shows autocorrelation plots for each predictor: in this case we can observe the presence of correlation for all covariates when $lag = 2$.

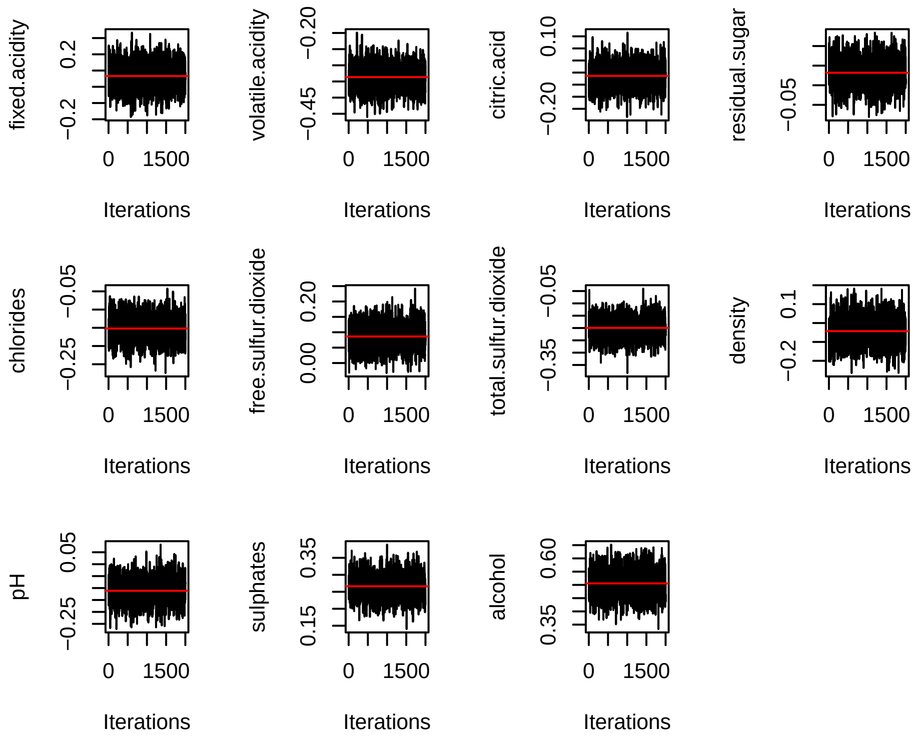

Figure 5: Traceplots

This problem can be solved by thinning the chain every two iterations: this means running again the algorithm and storing in the matrix **beta.post** only one iteration every two.

```
# we re-initialize all the inital values used in the algorithm
g           = c(-Inf,0,0,0,0,0,Inf)
beta.post.t = matrix(NA, S, p)
theta       = theta.zero
beta        = beta.zero
beta.0      = rep(0, p)
V           = diag(0.01, p)

# set the new lenght of the chain
S           = 2000
thin        = 2
n.iter      = S * thin
```

After re-setting the initial values we can run the same algorithm as before and we store the results in **beta.post.t**.

As we did before we take the column mean of **beta.post.t** to obtain the estimated posterior coefficients and we repeat the diagnostics.

```
beta.post.values.t = colMeans(beta.post.t)
```

Figures 7 and 8 show the diagnostics results after the thinning: all the traceplots show a path as good as before, and the autocorrelation in the acf are now lower.

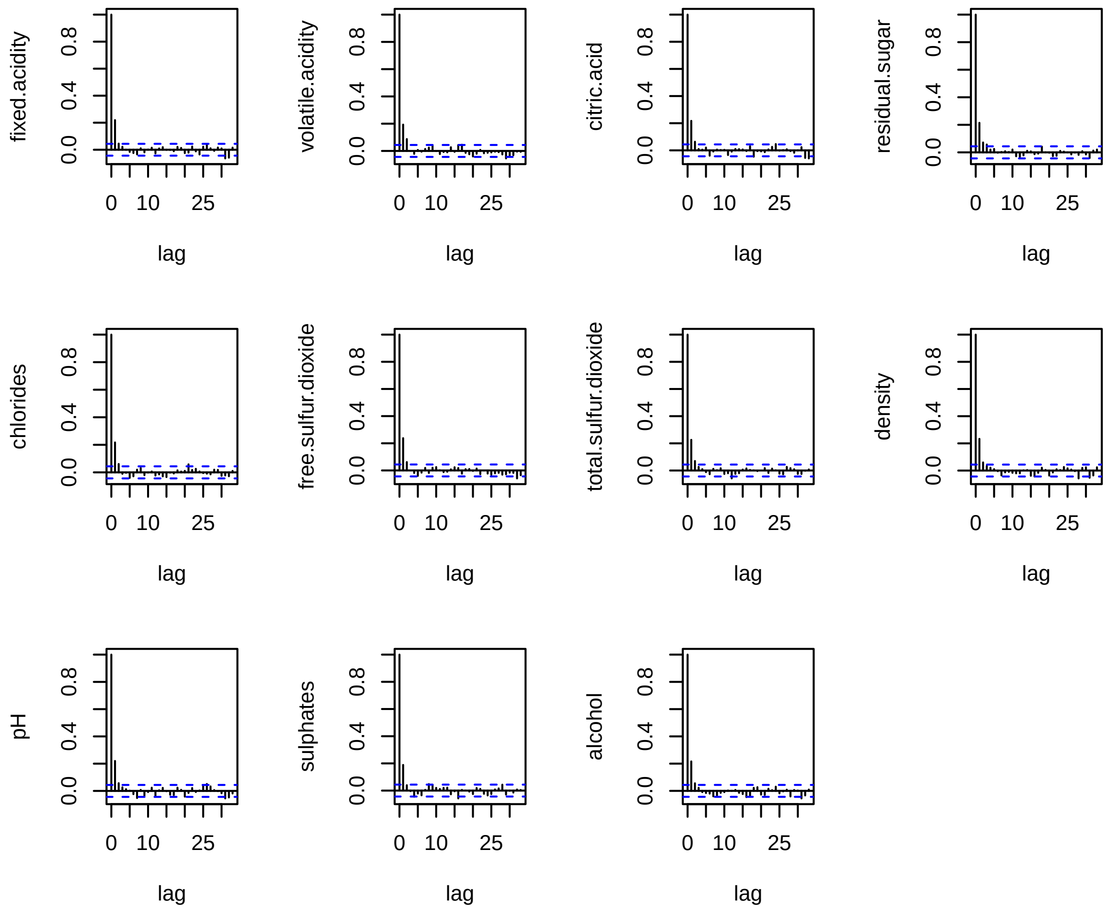

Figure 6: Acf plots

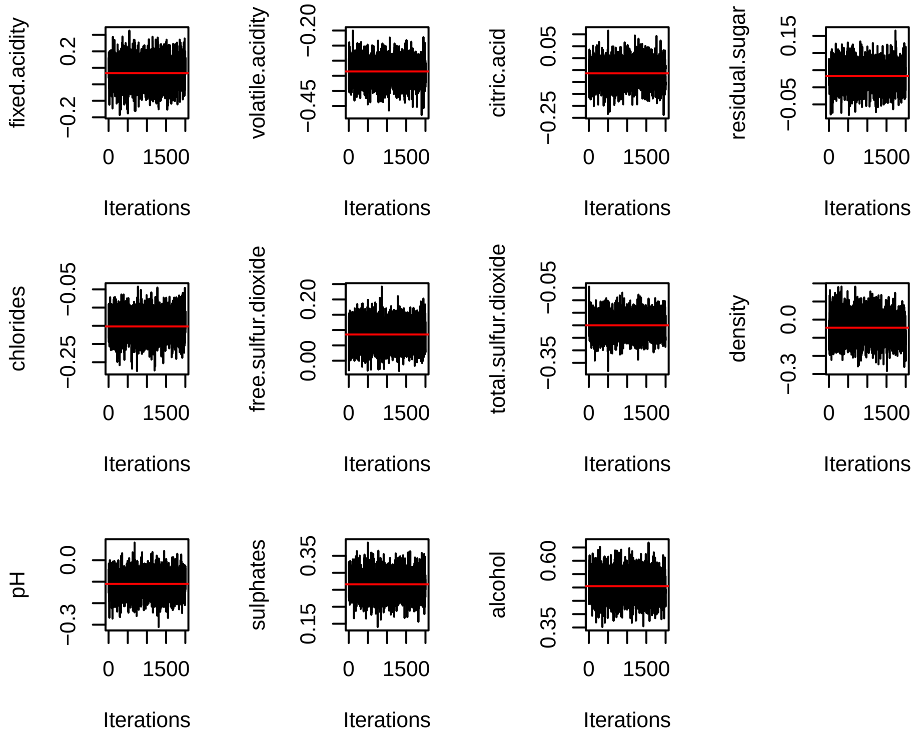

Figure 7: Traceplots after thinning

Posterior estimates of the coefficients

Table 4 shows the comparison between the posterior coefficients we obtained after the thinning and the their estimates obtained using the function **polr().**

From this table is clear that the posterior estimated values are almost identical to the frequentist coefficients. Moreover we can observe how the sign of the coefficients reflects the rrelationship between the covariates and the response: boxplots in Figure 2 highlighted a positive correlation between **quality** and **fixed.acidity**, **sulphates** and **alcohol** witch is confirmed by the positive sign of their posterior coefficients. The same can be observed for predictors like **volatile.acidity, density** and **pH**, whose boxplots showing negative correlation (Figure 3) match with the negative sign of their estimated coefficients.

Both Figure 9 and Figure 10 represents the distributions of the posterior coefficients β_j: from the first one we can observe that the median of each distribution is very close to their frequentist estimate (represented by the blue dot), while the histograms display a normal distribution.

References

- S. E. Fienberg, D. Lievesley, J. Rolph: *Statistics for Social Science and Public Policy.*
- S. Jackman: *Bayesian Analysis for the Social Sciences.*
- J. H Albert, S. Chib: *Bayesian Analysis of Binary and Polychotomous Response Data.*
- G. Consonni, F. Castelletti: *Bayesian Causal Inference in Probit Graphical Models.*
- https://rstudio-pubs-static.s3.amazonaws.com/552770_9d6bc497f80d4ab09f5076b398a62b09.html

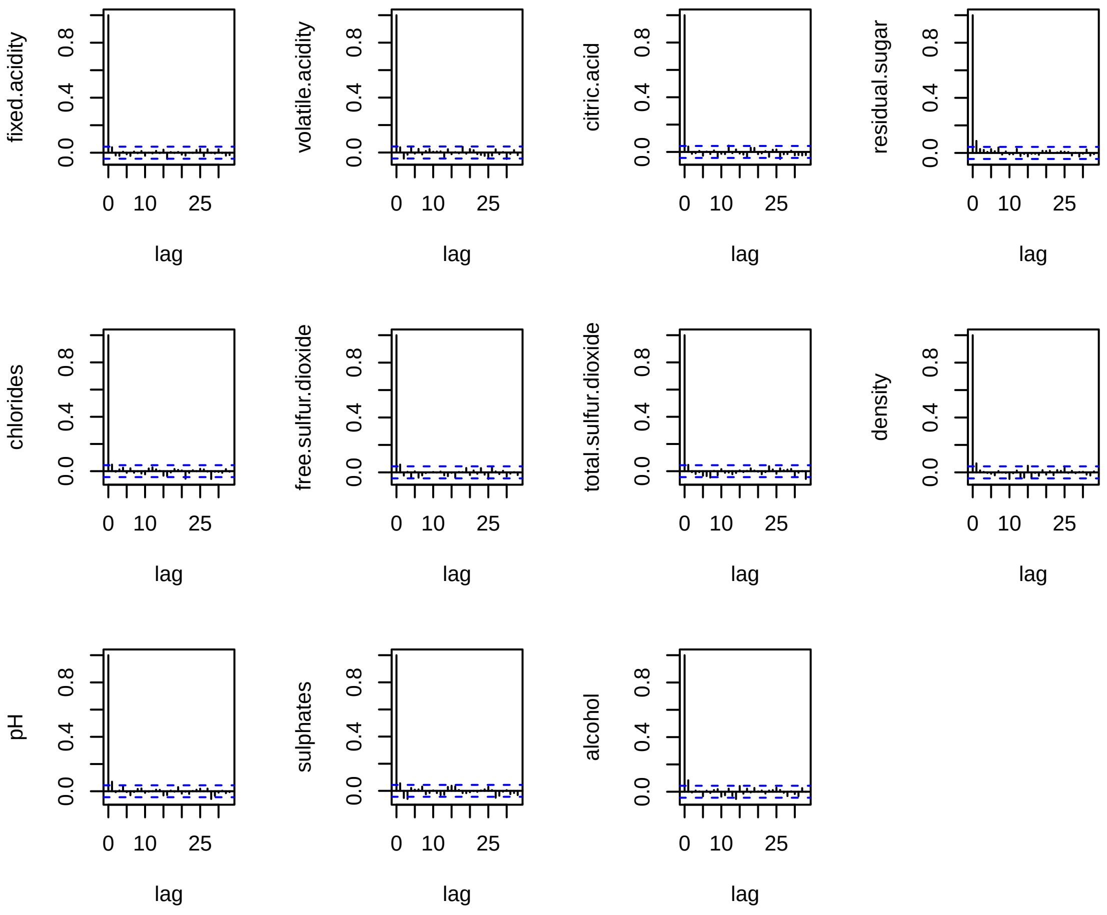

Figure 8: Acf after thinning

Table 4: Posterior estimates of the coefficients

	Posterior coefficients	Frequentist coefficients
fixed.acidity	0.06770	0.06573
volatile.acidity	-0.33546	-0.33649
citric.acid	-0.06333	-0.06436
residual.sugar	0.03271	0.03279
chlorides	-0.15169	-0.15152
free.sulfur.dioxide	0.08563	0.08623
total.sulfur.dioxide	-0.20002	-0.19980
density	-0.04460	-0.04221
pH	-0.11013	-0.11214
sulphates	0.26619	0.26603
alcohol	0.50484	0.50656

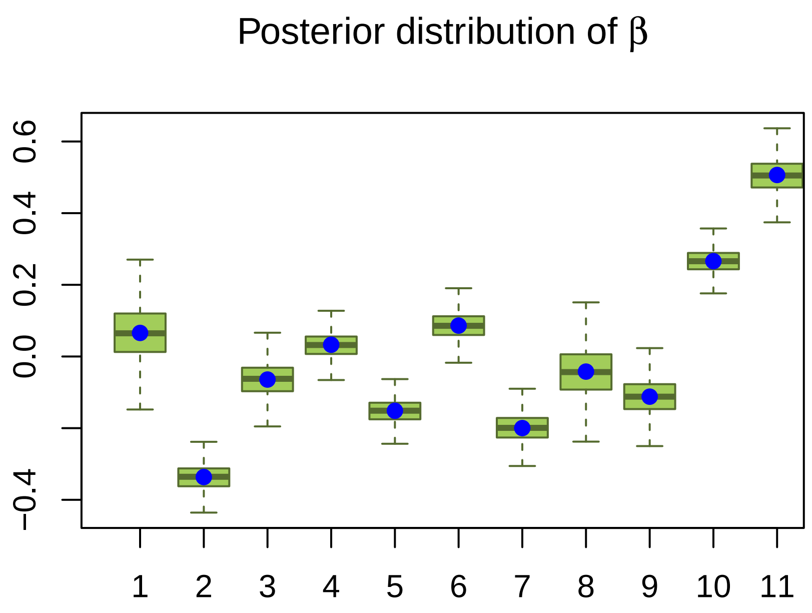

Figure 9: Boxplots of the posterior coefficients

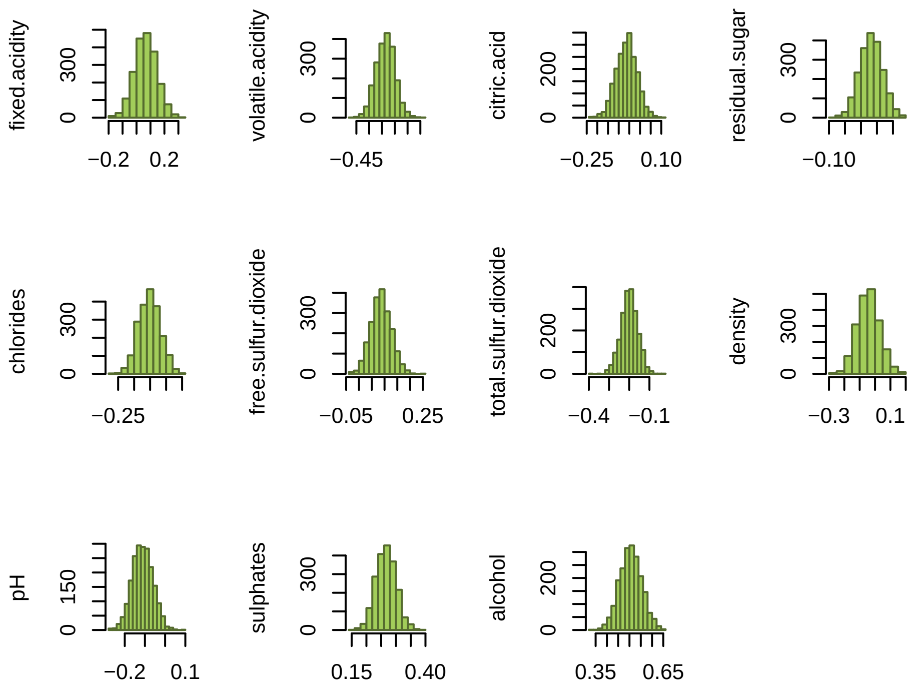

Figure 10: Histograms of the posterior coefficients