Chapter 7

Bayesian and Classical Prediction Models for Categorical and Count Data

7.1 Introduction

Categorical and count response variables are common in many plant breeding programs. For this reason, in this chapter, the fundamental and practical issues for implementing genomic prediction models with these types of response variables are provided. Also, current open source software is used for its implementation. The prediction models proposed here are Bayesian and classical models. For each type of these models, we provide the fundamental principles and concepts, whereas their practical implementation is illustrated with examples in the genomic selection context. Taken into account are the most common terms in the predictor, such as the main effects of environments and genotypes, and interaction terms like genotype $\times$ environment interaction and marker $\times$ environment interaction. First, the models under the Bayesian framework are presented and then those under a classical framework. Next, the fundamentals and applications of the Bayesian ordinal regression model are provided.

7.2 Bayesian Ordinal Regression Model

In many applications, the response (Y) is not continuous, but rather multi-categorical in nature: ordinal (the categories of the response can be ordered, $1, \dots, C$) or nominal (the categories have no order). For example, in plant breeding, categorical scores for resistance and susceptibility are often collected [disease: 1 (no disease), 2 (low infection), 3 (moderate infection), 4 (high infection), and 5 (complete infection)]. Another example is in animal breeding, where calving ease is scored as 1, 2, or 3 to indicate normal birth, slight difficult birth, or extreme difficult birth, respectively. The categorical traits are assigned to one of a set of mutually exclusive and exhaustive response values.

It is well documented that linear regression models in this situation are difficult to justify (Gianola 1980, 1982; Gianola and Foulley 1983) and inadequate results can be obtained, unless there is a large number of categories and the data seem to show a distribution that is approximately normal (Montesinos-López et al. 2015b).

For the ordinal model, we appeal to the existence of a latent continuous random variable and the categories are conceived as contiguous intervals on the continuous scale, as presented in McCullagh (1980), where the points of division (thresholds) are denoted as $\gamma_0, \gamma_1, \dots, \gamma_C$. In this way, the ordinal model assumes that conditioned to $\mathbf{x}_i$ (covariates of dimension p), Y_i is a random variable that takes values $1, \dots, C$, with the following probabilities:

$$\begin{aligned} p_{ic} &= P(Y_i = c) = P(\gamma_{c-1} \leq L_i \leq \gamma_c) \\ &= F(\gamma_c + \mathbf{x}_i^T \boldsymbol{\beta}) - F(\gamma_{c-1} + \mathbf{x}_i^T \boldsymbol{\beta}), \quad c = 1, \dots, C, \end{aligned} \quad (7.1)$$

where $L_i = -\mathbf{x}_i^T \boldsymbol{\beta} + \epsilon_i$ is a latent variable, ϵ_i is a random error term with cumulative distribution function F , $\boldsymbol{\beta} = (\beta_1, \dots, \beta_p)^T$ denotes the parameter vector associated with the effects of the covariates, and $-\infty = \gamma_0 < \gamma_1 < \dots < \gamma_C = \infty$ are threshold parameters and also need to be estimated, that is, the parameter vector to be estimated in this model is $\boldsymbol{\theta} = (\boldsymbol{\beta}^T, \boldsymbol{\gamma}^T, \sigma_\beta^2)^T$, where $\boldsymbol{\gamma} = (\gamma_1, \dots, \gamma_{C-1})^T$.

All the ordinal models presented in this chapter share the property that the categories can be considered as contiguous intervals on some continuous scale, but they differ in their assumptions regarding the distributions of the latent variable. When F is the standard logistic distribution function, the resulting model is known as the **logistic** ordinal model:

$$p_{ic} = F(\gamma_c + \mathbf{x}_i^T \boldsymbol{\beta}) - F(\gamma_{c-1} + \mathbf{x}_i^T \boldsymbol{\beta}), \quad c = 1, \dots, C,$$

where $F(z) = \frac{1}{1 + \exp(-z)}$. When F is the standard normal distribution function, the resulting model is the ordinal **probit** model:

$$p_{ic} = F(\gamma_c + \mathbf{x}_i^T \boldsymbol{\beta}) - F(\gamma_{c-1} + \mathbf{x}_i^T \boldsymbol{\beta}), \quad c = 1, \dots, C,$$

where $F(z) = \int_{-\infty}^z \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right) du$. When the response value only takes two values, model (7.1) is reduced to the binary regression model, which in the first case is better known as logistic regression, while in the second case, it is known as the probit regression model. For this reason, logistic regression or probit regression is a particular case of ordinal regression even under a Bayesian framework; for this reason, the Bayesian frameworks that will be described for ordinal regression in this chapter are reduced to logistic or probit regression when only one threshold parameter needs to be estimated (γ_1), or equivalently this is set to 0 and an intercept parameter (β_0) is added to the vector coefficients that also need to be estimated,

$\boldsymbol{\beta} = (\beta_0, \beta_1, \dots, \beta_p)^T$. In this model, the following development will be concentrated in the probit ordinal regression model.

A Bayesian specification for the ordinal regression is very similar to the linear model described in Chap. 6, and assumes the following prior distribution for the parameter vector $\boldsymbol{\theta} = (\boldsymbol{\beta}^T, \boldsymbol{\gamma}^T, \sigma_\beta^2)^T$:

$$f(\boldsymbol{\theta}) \propto \frac{1}{(\sigma_\beta^2)^{\frac{p}{2}}} \exp \left(\left[-\frac{1}{2\sigma_\beta^2} \boldsymbol{\beta}^T \boldsymbol{\beta} \right] \right) \frac{\left(\frac{S_\beta}{2} \right)^{\frac{v_\beta}{2}} \exp \left(-\frac{S_\beta}{2\sigma_\beta^2} \right)}{\Gamma(\frac{v_\beta}{2}) (\sigma_\beta^2)^{1+\frac{v_\beta}{2}}}$$

That is, a flat prior is assumed for the threshold parameter $\boldsymbol{\gamma}$, a multivariate normal distribution for the vector of beta coefficients, $\boldsymbol{\beta} \mid \sigma_\beta^2 \sim N_p(\mathbf{0}, \sigma_\beta^2 \mathbf{I}_p)$, and a scaled inverse chi-squared for σ_β^2 . From now on, this model will be referred to as BRR in this chapter in analogy to the Bayesian linear regression and the way that this is implemented in the BGLR R package. Similar to the genomic linear regression model, the posterior distribution of the parameter vector does not have a known form and a Gibbs sampler is used to explore this; for this reason, in the coming lines, the Gibbs sampling method proposed by Albert and Chib (1993) is described. To make it possible to derive the full conditional distributions, the parameter vector is augmented with a latent variable in the representation of model (7.1).

The joint probability density function of the vector of observations, $\mathbf{Y} = (Y_1, \dots, Y_n)^T$, and the latent variables $\mathbf{L} = (L_1, \dots, L_n)^T$, evaluated in the vector values $\mathbf{y} = (y_1, \dots, y_n)^T$ and $\mathbf{l} = (l_1, \dots, l_n)^T$, is given by

$$\begin{aligned} f(\mathbf{y}, \mathbf{l} \mid \boldsymbol{\beta}, \boldsymbol{\gamma}) &= \prod_{i=1}^n f_{L_i}(l_i) I_{(\gamma_{y_i-1}, \gamma_{y_i})}(l_i) \\ &= \prod_{i=1}^n \exp \left[-\frac{1}{2} (l_i + \mathbf{x}_i^T \boldsymbol{\beta})^2 \right] I_{(\gamma_{y_i-1}, \gamma_{y_i})}(l_i) \\ &= \exp \left[-\frac{1}{2} \sum_{i=1}^n (l_i + \mathbf{x}_i^T \boldsymbol{\beta})^2 \right] \prod_{i=1}^n I_{(\gamma_{y_i-1}, \gamma_{y_i})}(l_i), \end{aligned}$$

where I_A is the indicator function of a set A, and $f_{L_i}(l_i)$ is the density function of the latent variable that in the ordinal probit regression model corresponds to the normal distribution with mean $-\mathbf{x}_i^T \boldsymbol{\beta}$ and unit variance. Then the full conditional of the j th component of $\boldsymbol{\beta}$, β_j , given the rest of the parameters and the latent variables, is given by

$$\begin{aligned}
f(\beta_j | -) &\propto f(\mathbf{y}, \mathbf{l} | \boldsymbol{\beta}, \boldsymbol{\gamma}) f(\boldsymbol{\beta} | \sigma_\beta^2) \\
&\propto \prod_{i=1}^n f_{L_i}(l_i) I_{(\gamma_{y_i-1}, \gamma_{y_i})}(l_i) f(\boldsymbol{\beta} | \sigma_\beta^2) \\
&\propto \exp \left[-\frac{1}{2} \sum_{i=1}^n (e_{ij} + x_{ij} \beta_j)^2 - \frac{1}{2\sigma_\beta^2} \beta_j^2 \right] \\
&\propto \exp \left\{ -\frac{1}{2\tilde{\sigma}_{\beta_j}^2} (\beta_j - \tilde{\beta}_j)^2 \right\},
\end{aligned}$$

where $e_{ij} = l_i + \sum_{k \neq j}^p x_{ik} \beta_k$, $\mathbf{l} = (l_1, \dots, l_n)^T$, $\mathbf{x}_j = [x_{1j}, \dots, x_{nj}]^T$, $\mathbf{e}_j = [e_{1j}, \dots, e_{nj}]^T$, $\tilde{\sigma}_{\beta_j}^2 = (\sigma_\beta^{-2} + \mathbf{x}_j^T \mathbf{x}_j)^{-1}$, and $\tilde{\beta}_j = -\tilde{\sigma}_{\beta_j}^2 (\mathbf{x}_j^T \mathbf{e}_j)$. So, the full conditional distribution of β_j is $N(\tilde{\beta}_j, \tilde{\sigma}_{\beta_j}^2)$.

Now, the full conditional distribution of the threshold parameter γ_c is

$$\begin{aligned}
f(\gamma_c | -) &\propto f(\mathbf{y}, \mathbf{l} | \boldsymbol{\beta}, \boldsymbol{\gamma}) \\
&\propto \prod_{i=1}^n I_{(\gamma_{y_i-1}, \gamma_{y_i})}(l_i) \\
&\propto \prod_{i \in \{i: y_i = c\}} I_{(\gamma_{c-1}, \gamma_c)}(l_i) \prod_{i \in \{i: y_i = c+1\}} I_{(\gamma_c, \gamma_{c+1})}(l_i) \\
&\propto I_{(a_c, b_c)}(\gamma_c),
\end{aligned}$$

where $a_c = \max \{l_i : y_i = c\}$ and $b_c = \min \{l_i : y_i = c+1\}$. So $\gamma_c | - \sim U(a_c, b_c)$, $c = 1, 2, \dots, C-1$. Now, the full conditional distribution of the latent variables is

$$f(\mathbf{l} | -) \propto f(\mathbf{y}, \mathbf{l} | \boldsymbol{\beta}, \boldsymbol{\gamma}) \propto \prod_{i=1}^n f_{L_i}(l_i) I_{(\gamma_{y_i-1}, \gamma_{y_i})}(l_i)$$

from which we have that conditional on the rest of parameters, the latent variables are independent truncated normal random variables, that is, $L_i | - \sim N(-\mathbf{x}_i^T \boldsymbol{\beta}, 1)$ truncated in the interval $(\gamma_{y_i-1}, \gamma_{y_i})$, $i = 1, \dots, n$.

The full conditional distribution of σ_β^2 is the same as in the linear regression model described in Chap. 6, so a Gibbs sampler for exploration of the posterior distribution of the parameters of the models can be implemented by iterating the following steps:

1. Initialize the parameters: $\beta_j = \beta_{j0}$, $j = 1, \dots, p$, $\gamma_c = \gamma_{c0}$, $c = 1, \dots, C-1$, $l_i = l_{i0}$, $i = 1, \dots, n$, and $\sigma_\beta^2 = \sigma_{\beta 0}^2$.
2. For each $i = 1, \dots, n$, simulate l_i from the normal distribution $N(-\mathbf{x}_i^T \boldsymbol{\beta}, 1)$ truncated in $(\gamma_{y_i-1}, \gamma_{y_i})$.

3. For each $j = 1, \dots, p$, simulate from the full conditional distribution of β_j , that is, from a $N(\tilde{\beta}_j, \tilde{\sigma}_{\beta_j}^2)$, where $\tilde{\sigma}_{\beta_j}^2 = (\sigma_{\beta}^{-2} + \mathbf{x}_j^T \mathbf{x}_j)^{-1}$ and $\tilde{\beta}_j = -\tilde{\sigma}_{\beta_j}^2 (\mathbf{x}_j^T \mathbf{e}_j)$.
4. For each $c = 1, 2, \dots, C - 1$, simulate from the full conditional distribution of γ_c , $\gamma_c \mid - \sim U(a_c, b_c)$, where $a_c = \max \{l_i : y_i = c\}$ and $b_c = \min \{l_i : y_i = c + 1\}$.
5. Simulate σ_{β}^2 from a scaled inverse chi-squared distribution with parameters $\tilde{v}_{\beta} = v_{\beta} + p$ and $\tilde{S}_{\beta} = S_{\beta} + \boldsymbol{\beta}^T \boldsymbol{\beta}$, that is, from a $\chi^{-2}(\tilde{v}_{\beta}, \tilde{S}_{\beta})$.
6. Repeat 1–5 until a burning period and the desired number of samples are reached.

A similar Gibbs sampler is implemented in the BGLR R package, and the hyperparameters are established in a similar way as the linear regression models described in Chap. 6. When the hyperparameters S_{β} and v_{β} are not specified, by default BGLR is assigned $v_{\beta} = 5$ and to S_{β} a value such that the prior mode of σ_{β}^2 ($\chi^{-2}(v_{\beta}, S_{\beta})$) is equal to half of the “total variance” of the latent variables (Pérez and de los Campos 2014). An implementation of this model can be done with a fixed value of the variance component parameter $\sigma_{0\beta}^2$ by choosing a very large value of the degree of freedom parameter ($v_{\beta} = 10^{10}$) and taking the scale value parameter $S_{\beta} = \sigma_{0\beta}^2(v_{\beta} + 2)$.

The basic R code with the BGLR implementation of this model is as follows:

```
ETA = list( list( X=X, model='BRR' ) )
A = BGLR(y=y, response_type='ordinal', ETA=ETA, nIter = 1e4, burnIn = 1e3)
Probs = A$Probs
```

where the predictor component is specified in **ETA**: argument **X** specifies the design matrix, whereas the argument **model** describes the priors of the model's parameters. The response variable vector is given in **y**, and the ordinal model is specified in the **response_type** argument. In the last line, the posterior means of the probabilities of each category are extracted, and can be used to give an estimation of a particular metric when comparing models or to evaluate the performance of this model, which will be explained later. An application of this ordinal model is given by Montesinos-López et al. (2015a).

Similar to the linear regression model in Chap. 6, when the number of markers to be used in genomic prediction is very large relative to the number of observations, $p \gg n$, the dimension of the posterior distribution (number of parameters in $\boldsymbol{\theta}$, $p + c - 1 + 1$) to be explored can be reduced by simulating the genetic vector effect $\mathbf{b} = (b_1, \dots, b_n)^T$, where $b_i = \mathbf{x}_i^T \boldsymbol{\beta}$, $i = 1, \dots, n$, instead of $\boldsymbol{\beta}$. Because $\boldsymbol{\beta} \mid \sigma_{\beta}^2 \sim N_p(\mathbf{0}, \sigma_{\beta}^2 \mathbf{I}_p)$, the induced distribution of $\mathbf{b} = (b_1, \dots, b_n)^T$ is $\mathbf{b} \mid \sigma_g^2 \sim N_n(\mathbf{0}, \sigma_g^2 \mathbf{G})$, where $\sigma_g^2 = p\sigma_{\beta}^2$ and $\mathbf{G} = \frac{1}{p} \mathbf{X}^T \mathbf{X}$. Then, with this and assuming a scaled inverse chi-squared distribution as prior for σ_g^2 , $\sigma_g^2 \sim \chi^{-2}(v_g, S_g)$, and a flat prior for the threshold parameters ($\boldsymbol{\gamma}$), an ordinal probit GBLUP Bayesian regression model specification of model (7.1) is given by

$$p_{ic} = P(Y_i = c) = \Phi(\gamma_c + b_i) - \Phi(\gamma_{c-1} + b_i), \quad c = 1, \dots, C, \quad (7.2)$$

where now $L_i = -b_i + \epsilon_i$ is the latent variable, and Φ is the cumulative normal standard distribution. In matrix form the model for the latent variable can be specified as $\mathbf{L} = \mathbf{b} + \boldsymbol{\epsilon}$, where $\mathbf{L} = (L_1, \dots, L_n)^T$ and $\boldsymbol{\epsilon} \sim N_n(\mathbf{0}, \mathbf{I}_n)$. A Gibbs sampler of the posterior of the parameters of this model can be obtained similarly as was done for model (7.1). Indeed, the full conditional density of $\mathbf{b}$ is given by

$$\begin{aligned} f(\mathbf{b}|-) &\propto f(\mathbf{y}, \mathbf{l}|\mathbf{b}, \boldsymbol{\gamma}) f(\mathbf{b}|\sigma_g^2) \\ &\propto \prod_{i=1}^n f_{L_i}(l_i) I_{(\gamma_{y_i-1}, \gamma_{y_i})}(l_i) f(\mathbf{b}|\sigma_g^2) \\ &\propto \exp \left[-\frac{1}{2}(\mathbf{l} + \mathbf{b})^T(\mathbf{l} + \mathbf{b}) - \frac{1}{2\sigma_g^2} \mathbf{b}^T \mathbf{G}^{-1} \mathbf{b} \right] \\ &\propto \exp \left[-\frac{1}{2}(\mathbf{b} - \tilde{\mathbf{b}})^T \tilde{\boldsymbol{\Sigma}}_b (\mathbf{b} - \tilde{\mathbf{b}}) \right], \end{aligned}$$

where $\tilde{\boldsymbol{\Sigma}}_b = (\sigma_g^{-2} \mathbf{G}^{-1} + \mathbf{I}_n)^{-1}$ and $\tilde{\mathbf{b}} = -\tilde{\boldsymbol{\Sigma}}_b \mathbf{l}$. So the full conditional distribution of $\mathbf{b}$ is $N_n(\tilde{\mathbf{b}}, \tilde{\boldsymbol{\Sigma}}_b)$. Then, a Gibbs sampler exploration of this model can be done by iterating the following steps:

1. Initialize the parameters: $b_i = b_{i0}$, $i = 1, \dots, n$, $\gamma_c = \gamma_{c0}$, $c = 1, \dots, C-1$, $l_i = l_{i0}$, $i = 1, \dots, n$, and $\sigma_\beta^2 = \sigma_{\beta 0}^2$.
2. For each $i = 1, \dots, n$, simulate l_i from the normal distribution $N(-\mathbf{x}_i^T \boldsymbol{\beta}, 1)$ truncated in $(\gamma_{y_i-1}, \gamma_{y_i})$.
3. Simulate $\mathbf{b}$ from $N_n(\tilde{\mathbf{b}}, \tilde{\boldsymbol{\Sigma}}_b)$, with $\tilde{\boldsymbol{\Sigma}}_b = (\sigma_g^{-2} \mathbf{G}^{-1} + \mathbf{I}_n)^{-1}$ and $\tilde{\mathbf{b}} = -\tilde{\boldsymbol{\Sigma}}_b \mathbf{l}$.
4. For each $c = 1, 2, \dots, C-1$, simulate from the full conditional distribution of γ_c $\gamma_c | - \sim U(a_c, b_c)$, where $a_c = \max \{l_i : y_i = c\}$ and $b_c = \min \{l_i : y_i = c+1\}$.
5. Simulate σ_g^2 from a scaled inverse chi-squared distribution with parameters $\tilde{v}_g = v_g + n$ and $\tilde{S}_g = S_g + \mathbf{b}^T \mathbf{b}$, that is, from a $\chi^{-2}(\tilde{v}_g, \tilde{S}_g)$.
6. Repeat 1–5 until a burning period and the desired number of samples are reached.

This model can also be implemented with the BGLR package:

```
ETA = list( list( K=G, model='RKHS' ) )
A = BGLR(Y=y, response_type='ordinal', ETA=ETA, nIter = 1e4, burnIn = 1e3)
Probs = A$Probs
```

where instead of specifying the design matrix and the prior model to the associated regression coefficients, now the genomic relationship matrix $\mathbf{G}$ is given, and the corresponding model in this case corresponds to RKHS, which can also be used

when another type of information between the lines is available, for example, the pedigree relationship matrix.

Like the Bayesian linear regression model, other variants of model BRR can be obtained by adopting different prior models for the beta coefficients (β): FIXED, BayesA, BayesB, BayesC, or BL, for example. See Chap. 6 for details of each of these prior models for the regression coefficients. Indeed, the full conditional distributions used to implement a Gibbs sampler in each of these ordinal models are the same as the corresponding Bayesian linear regression models, except that the full conditional distribution for the variance component of random errors is not needed and two full conditional distributions (for the threshold parameters and the latent variables) are added, where now the latent variable will play the role of the response value in the Bayesian linear regression. These models can also be implemented in the BGLR package.

For example, a Gibbs sampler implementation for ordinal model (7.1) with a BayesA prior (see Chap. 6) can be done by following steps:

1. Initialize the parameters: $\beta_j = \beta_{j0}, j = 1, \dots, p, \gamma_c = \gamma_{c0}, c = 1, \dots, C - 1, l_i = l_{i0}, i = 1, \dots, n$, and $\sigma_\beta^2 = \sigma_{\beta 0}^2$.
2. For each $i = 1, \dots, n$, simulate l_i from the normal distribution $N(-\mathbf{x}_i^T \beta, 1)$ truncated in $(\gamma_{y_i-1}, \gamma_{y_i})$.
3. For each $j = 1, \dots, p$, simulate from the full conditional distribution of β_j , that is, from a $N(\tilde{\beta}_j, \tilde{\sigma}_{\beta_j}^2)$, where $\tilde{\sigma}_{\beta_j}^2 = (\sigma_{\beta_j}^{-2} + \mathbf{x}_j^T \mathbf{x}_j)^{-1}$ and $\tilde{\beta}_j = \tilde{\sigma}_{\beta_j}^2 (\mathbf{x}_j^T \mathbf{e}_j)$.
4. For each $c = 1, 2, \dots, C - 1$, simulate from the full conditional distribution of γ_c , $\gamma_c | - \sim U(a_c, b_c)$, where $a_c = \max \{l_i : y_i = c\}$ and $b_c = \min \{l_i : y_i = c + 1\}$.
5. Simulate from the full conditional of each $\sigma_{\beta_j}^2$

$$\sigma_{\beta_j}^2 | \gamma, \mathbf{l}, \beta_0, \sigma_{-j}^2, S_\beta, \sim \chi_{\tilde{v}_{\beta_j}}^{-2},$$

where $\tilde{v}_{\beta_j} = v_\beta + 1$, scale parameter $\tilde{S}_{\beta_j} = S_\beta + \beta_j^2$, and σ_{-j}^2 is the vector $\sigma_\beta^2 = (\sigma_{\beta_1}^2, \dots, \sigma_{\beta_p}^2)$ but without the j th entry.

6. Simulate from the full conditional of S_β

$$\begin{aligned} f(S_\beta | -) &\propto \left[\prod_{j=1}^p f(\sigma_{\beta_j}^2 | S_\beta) \right] f(S_\beta) \\ &\propto \prod_{j=1}^p \left[\frac{\left(\frac{S_\beta}{2}\right)^{\frac{v_\beta}{2}}}{\Gamma\left(\frac{v_\beta}{2}\right) (\sigma_{\beta_j}^2)^{1+\frac{v_\beta}{2}}} \exp\left(-\frac{S_\beta}{2\sigma_{\beta_j}^2}\right) \right] S_\beta^{s-1} \exp(-rS_\beta) \\ &\propto S_\beta^{s+\frac{pv_\beta}{2}-1} \exp\left[-\left(r + \frac{1}{2} \sum_{j=1}^p \frac{1}{\sigma_{\beta_j}^2}\right) S_\beta\right] \end{aligned}$$

which corresponds to the kernel of the gamma distribution with rate parameter $\tilde{r} = r + \frac{1}{2} \sum_{j=1}^p \frac{1}{\sigma_{\beta j}^2}$ and shape parameter $\tilde{s} = s + \frac{p\nu_{\beta}}{2}$, and so, $S_{\beta} | - \sim \text{Gamma}(\tilde{r}, \tilde{s})$.

7. Repeat steps 2–6 depending on how many values you wish to simulate of the parameter vector $(\beta^T, \sigma_{\beta}^2, \gamma^T)$

The implementation of BayesA for an ordinal response variable in BGLR is as follows:

```
ETA = list( list( X=X, model='BayesA' ) )
A = BGLR(y=y, response_type='ordinal', ETA=ETA, nIter = 1e4, burnIn =
1e3)
Probs = A$Probs
```

The implementation of the other Bayesian ordinal regression models, BayesB, BayesC, and BL, in BGLR is done by only replacing in ETA the corresponding model, as was commented before, see Chap. 6 for details and the difference between all these prior models for the regression coefficients.

7.2.1 Illustrative Examples

Example 1 BRR and GBLUP To illustrate how these models work and how they can be implemented, here we used a toy data set with an ordinal response with five levels and consisting of 50 lines that were planted in three environments (a total of 150 observations) and the genotyped information of 1000 markers for each line. To evaluate the performance of the described models (BRR, BayesA, GBLUP, BayesB, BayesC, and BL), 10 random partitions were used in a cross-validation strategy, where 80% of the complete data set was used to train the model and the rest to evaluate the model in terms of the Brier score (BS) and the proportion of cases correctly classified (PCCC).

Six models were implemented with the BGLR R package, using a burn-in period equal to 1000 and 10,000 iterations, and the default hyperparameter for prior distribution. The results are shown in Table 7.1, where we can appreciate similar performance in terms of the Brier score metric in all models, and a similar but poor performance of all models when the proportion of cases correctly classified (PCCC) was used. With the BRR model only in 4 out of 10 partitions, the values of the PCCC were slightly greater or equal to what can be obtained by random assignation (1/5), while with the GBLUP model this happened in 3 out of 10 partitions. A similar behavior is obtained with the rest of the models. On average, the PCCC values were 0.1566, 0.1466, 0.15, 0.1466, 0.1533, and 0.1433 for the BRR, GBLUP, BayesA, BayesB, BayesC, and Bayes Lasso (BL) models, respectively.

Table 7.1 Brier score (BS) and proportion of cases correctly classified (PCCC) across 10 random partitions, with 80% of the total data set used for training and the rest for testing, under model (7.1) with different priors models to the marker coefficients: ordinal Bayesian Ridge regression model (BRR), BayesA, BayesB, BayesC, and BL (model (7.1)); and under the ordinal GBLUP regression model (7.2)

Model	BRR		GBLUP		BayesA	
PT	BS	PCCC	BS	PCCC	BS	PCCC
1	0.3957	0.1333	0.3937	0.1333	0.3923	0.1333
2	0.3891	0.2000	0.3884	0.2000	0.3896	0.2000
3	0.3977	0.2333	0.3944	0.2333	0.3935	0.2333
4	0.4168	0.1333	0.4154	0.1333	0.4131	0.2000
5	0.4079	0.0667	0.3991	0.0667	0.4013	0.0667
6	0.4068	0.1000	0.4038	0.1000	0.4074	0.1000
7	0.4342	0.1000	0.4258	0.1000	0.4298	0.0667
8	0.3867	0.2000	0.3845	0.1667	0.3858	0.1667
9	0.4145	0.1667	0.4171	0.1000	0.4183	0.1000
10	0.3875	0.2333	0.3862	0.2333	0.3866	0.2333
Average (SD)	0.4036 (0.01)	0.1566 (0.05)	0.4008 (0.01)	0.1466 (0.05)	0.4017 (0.01)	0.15 (0.06)
Model	BayesB		BayesC		BL	
PT	BS	PCCC	BS	PCCC	BS	PCCC
1	0.3924	0.1333	0.3943	0.1333	0.3918	0.1333
2	0.3881	0.2000	0.3885	0.2000	0.3878	0.2000
3	0.3941	0.2667	0.3951	0.2333	0.3935	0.2333
4	0.4141	0.1333	0.4167	0.1333	0.4142	0.1333
5	0.4037	0.0667	0.4044	0.0667	0.4047	0.0667
6	0.3978	0.1000	0.4022	0.1000	0.3984	0.1000
7	0.4268	0.0667	0.4286	0.1000	0.4281	0.0667
8	0.3849	0.1667	0.3862	0.2000	0.3852	0.1667
9	0.4184	0.1000	0.4155	0.1000	0.4169	0.1000
10	0.3869	0.2333	0.3862	0.2667	0.3861	0.2333
Average (SD)	0.4007 (0.01)	0.1466 (0.06)	0.4017 (0.01)	0.1533 (0.06)	0.4006 (0.01)	0.1433 (0.06)

The R code to reproduce the results in Table 7.1 is given in Appendix 1.

In some applications, additional information is available, such as the sites (locations) where the experiments were conducted, environmental covariates, etc., which can be taken into account to improve the prediction performance.

One extension of model (7.1) that takes into account environment effects and environment–marker interaction is given by

$$p_{ic} = P(Y_i = c) = F(\gamma_c - \eta_i) - F(\gamma_{c-1} - \eta_i), c = 1, \dots, C, \quad (7.3)$$

where now the predictor $\eta_i = \mathbf{x}_{Ei}^T \boldsymbol{\beta}_E + \mathbf{x}_i^T \boldsymbol{\beta} + \mathbf{x}_{EMi}^T \boldsymbol{\beta}_{EM}$, in addition to the marker effects ($\mathbf{x}_i^T \boldsymbol{\beta}$) in the second summand, is included the environment and the

environment–marker interaction effects, respectively. In the latent variable representation, this model is equivalently expressed as

$$P(Y_i = c) = P(\gamma_{c-1} \leq L_i \leq \gamma_c), c = 1, \dots, C,$$

where $\mathbf{L} = (L_1, \dots, L_n)^T = \mathbf{X}_E \boldsymbol{\beta}_E + \mathbf{X} \boldsymbol{\beta} + \mathbf{X}_{EM} \boldsymbol{\beta}_{EM} + \boldsymbol{\epsilon}$ is the vector with latent random variables of all observations, $\boldsymbol{\epsilon} \sim N_n(\mathbf{0}, \mathbf{I}_n)$ is a random error vector, $\mathbf{X}_E$, $\mathbf{X}$, and $\mathbf{X}_{EM}$ are the design matrices of the environments, markers, and environment–marker interactions, respectively, while $\boldsymbol{\beta}_E$ and $\boldsymbol{\beta}_{EM}$ are the vectors of the environment effects and the interaction effects, respectively, with a prior distribution that can be specified as was done for $\boldsymbol{\beta}$. In fact, with the BGLR function, it is also possible to implement all these extensions, since it allows using any of the several priors included here: FIXED, BRR, BayesA, BayesB, BayesC, and BL. For example, the basic BGLR code to implement model (7.3) with a flat prior (“FIXED”) for the environment effects, a BRR prior for marker effects and for the environment–marker interaction effects, is as follows:

```
X = scale(X)
#Environment matrix design
XE = model.matrix(~Env, data=dat_F)[, -1]
#Environment-marker interaction matrix design
XEM = model.matrix(~0+X:Env, data=dat_F)
ETA = list(list(X=XE, model='FIXED'), list(X=X, model='BRR'),
            list(X=XEM, model='BRR')) #Predictor
A = BGLR(y=y, response_type='ordinal', ETA=ETA, nIter = 1e4, burnIn =
1e3, verbose = FALSE)
Probs = A$probs
```

where **dat_F** is the data file that contains all the information of how the data was collected (GID: Lines or individuals; Env: Environment; y: response variable of the trait). Other desired prior models to beta coefficients of each predictor component are obtained only by replacing the “model” argument of each of the three components of the predictor. For example, for a BayesA prior model for the marker effects, in the second sub-list we must use model=**BayesA**.

The latent random vector of model (7.1) under the GBLUP specification, plus genotypic and environment×genotypic interaction effects, takes the form

$$\mathbf{L} = \mathbf{X}_E \boldsymbol{\beta}_E + \mathbf{Z}_L \mathbf{g} + \mathbf{Z}_{LE} \mathbf{g}_E + \boldsymbol{\epsilon} \quad (7.4)$$

which is like model (6.7) in Chap. 6, where $\mathbf{Z}_L$ and $\mathbf{Z}_{LE}$ are the incident matrices of the genotypes and environment×genotype interaction effects, respectively, and $\mathbf{g}$ and $\mathbf{g}_E$ are the corresponding random effects which have distributions $N_J(\mathbf{0}, \sigma_g^2 \mathbf{G})$ and $N_J(\mathbf{0}, \sigma_{g_E}^2 (\mathbf{I}_I \otimes \mathbf{G}))$, respectively, where J is the number of different lines in the data set. This model can be trained with the BGLR package as follows:

```

I = length(unique(dat_F$Env))
XE = model.matrix(~0+Env, data=dat_F)[, -1]
Z_L = model.matrix(~0+GID, data=dat_F, xlev = list(GID=unique
(dat_F$GID)))
K_L = Z_L %>% G %>% t(Z_L)
Z_LE = model.matrix(~0+GID:Env, data=dat_F,
xlev = list(GID=unique(dat_F$GID), Env = unique(dat_F$Env)))
K_LE = Z_LE %>% kronecker(diag(I), G) %>% t(Z_LE)
ETA = list(list(model='FIXED', X=XE),
list(model='RHKS', K=K_L)),
list(model='RKHS', K=K_LE))
A = BGLR(y, response_type = "ordinal", ETA=ETA, nIter = 1e4, burnIn =
1e3)
Probs = A$Probs

```

where `dat_F` is as before, that is, the data file that contains all the information of how the data were collected. Similar specification can be done when other kinds of covariates are available, for example, environmental covariates.

Example 2 Environments + Markers + Marker×Environment Interaction This example illustrates how models (7.3) and (7.4) can be fitted and used for prediction. Here we used the same data set as in Example 1, but now the environment and environment×marker effects are included in the predictor. For model (7.3), to the environment effect a flat prior is assigned [FIXED, a normal distribution with mean 0 and very large variance (10^{10})], and one of BRR, BayesA, BayesB, BayesC, or BL prior model is assigned to the marker and marker×environment interaction effects. For model (7.4), a flat prior is assigned for environment effects. The performance evaluation was done using the same 10 random partitions used in Example 1, where 80% of the complete data set was used for training the model and the rest for testing, in which the Brier score (BS) and the proportion of correctly classified (PCCC) metrics were computed.

The results are shown in Table 7.2. They indicate an improved prediction performance of these models compared to the models fitted in Example 1 which only takes into account the marker effects (see Table 7.1). However, this improvement is only slightly better under the Brier score, because the reduction in the average BS across all 10 partitions was 1.55, 3.74, 0.96, 0.83, 1.41, and 1.32% for the BRR, GBLUP, BayesA, BayesB, BayesC, and BL, respectively. A more notable improvement was obtained with the PCCC criteria, where now for all models in about 8 out of the 10 partitions, the value of this metric was greater than the one obtained by random chance only. Indeed, the average values across the 10 partitions were 0.27, 0.28, 0.28, 0.27, 0.28, and 0.27, for the BRR, GBLUP, BayesA, BayesB, BayesC, and BL, respectively. This indicates 74.47, 90.90, 86.67, 86.36, 80.43, and 86.05% improvement for these models with respect to their counterpart models when only marker effects or genomic effects were considered. So, greater improvement was observed with the GBLUP model with both metrics, but finally the performance of all models is almost undistinguishable but with an advantage in time execution of the GBLUP model with respect to the rest. These issues were

Table 7.2 Brier score (BS) and proportion of cases correctly classified (PCCC) across 10 random partitions, with 80% of the total data set used for training and the rest for testing, under model (7.3) with different prior models for the marker effects and the environment×marker interaction: BRR, BayesA, BayesB, BayesC, and BL; and under the ordinal GBLUP regression model (7.4)

Model	BRR		GBLUP		BayesA	
PT	BS	PCC	BS	PCC	BS	PCC
1	0.3942	0.3333	0.3721	0.3667	0.3925	0.2333
2	0.3872	0.3000	0.3881	0.2667	0.3871	0.3000
3	0.4081	0.2667	0.4019	0.2667	0.4054	0.2667
4	0.4428	0.1333	0.4375	0.0667	0.4448	0.1667
5	0.3853	0.2667	0.3602	0.3000	0.3837	0.2333
6	0.3905	0.3667	0.3681	0.3000	0.3899	0.3667
7	0.3995	0.1333	0.3849	0.2000	0.4007	0.1667
8	0.3737	0.3000	0.3690	0.3333	0.3816	0.3333
9	0.3999	0.3000	0.3893	0.3333	0.3989	0.3333
10	0.3937	0.3333	0.3925	0.3667	0.3953	0.4000
Average (SD)	0.3975 (0.01)	0.2733 (0.07)	0.3863 (0.02)	0.28 (0.09)	0.3979 (0.01)	0.28 (0.08)

Model	BayesB		BayesC		BL	
PT	BS	PCC	BS	PCC	BS	PCC
1	0.3913	0.3000	0.3894	0.2667	0.3890	0.2333
2	0.3868	0.3333	0.3868	0.3000	0.3851	0.3333
3	0.4077	0.2667	0.4081	0.2667	0.4063	0.2667
4	0.4456	0.1333	0.4420	0.1667	0.4403	0.1667
5	0.3809	0.2667	0.3805	0.2667	0.3814	0.2667
6	0.3883	0.3667	0.3862	0.3667	0.3846	0.3667
7	0.4026	0.1333	0.3992	0.1333	0.3964	0.1333
8	0.3735	0.3000	0.3735	0.3000	0.3739	0.3000
9	0.4026	0.3333	0.4015	0.3333	0.4037	0.3000
10	0.3948	0.3000	0.3948	0.3667	0.3939	0.3000
Average (SD)	0.3974 (0.01)	0.2733 (0.07)	0.3961 (0.01)	0.2766 (0.07)	0.3954 (0.01)	0.2666 (0.07)

pointed out in Chaps. 5 and 6. This difference is even greater when the number of markers is larger than the number of observations.

Example 3 Binary Traits For this example, we used the EYT Toy data set consisting of 40 lines, four environments (Bed5IR, EHT, Flat5IR, and LHT), and a response binary variable based on plant Height (0 = low, 1 = high). For this example, marker information is not available, only the genomic relationship matrix for the 40 lines. So, only the models (M3 and M4) in (7.3) and (7.4) are fitted, and the performance prediction for these models was evaluated using cross-validation. Also, in this comparison model, (M5) (7.5) is added but without the line×environment interaction effects, that is, only the environment effect and the genetic effects are taken into account in the linear predictor:

Table 7.3 Brier score (BS) and proportion of cases correctly classified (PCCC) across 10 random partitions, with 80% of the total data set used for training and the rest for testing, under models (7.3), (7.4), and (7.5) with a flat prior for environment effects

Model	M3		M4		M5	
PT	BS	PCCC	BS	PCCC	BS	PCCC
1	0.1841	0.8438	0.2250	0.7500	0.2166	0.7813
2	0.1580	0.8438	0.2307	0.5938	0.2221	0.7188
3	0.1960	0.7813	0.2357	0.6563	0.2308	0.6563
4	0.2261	0.7188	0.2119	0.6875	0.2102	0.6563
5	0.2108	0.6250	0.3275	0.4063	0.3174	0.4063
6	0.2070	0.7188	0.2280	0.6563	0.2242	0.6250
7	0.2094	0.6563	0.2261	0.6250	0.2248	0.5938
8	0.1864	0.7500	0.2322	0.7188	0.2293	0.6875
9	0.1813	0.7813	0.2643	0.5000	0.2561	0.5313
10	0.2310	0.6563	0.2865	0.5625	0.2878	0.5313
Average (SD)	0.199 (0.02)	0.7375 (0.07)	0.2468 (0.03)	0.6156 (0.1)	0.2419 (0.03)	0.6187 (0.1)

$$L = X_E\beta_E + Z_Lg + \epsilon \tag{7.5}$$

The results are presented in Table 7.3 with the BS and PCCC metrics obtained in each partition of the random CV strategy. From this we can appreciate that the best performance with both metrics was obtained with the model that considered only the genetic effects (M3; (7.3)). On average, models M4 and M5 gave a performance in terms of BS, that was 24.02 and 19.79% greater than the corresponding performance of model M3, while with model M3, on average across the 10 partitions, an improvement of 21.57 and 19.19% in terms of PCCC was obtained with regard to models M4 and M5. The difference between these last two models is only slight; the average BS value of the first model was 2.01% greater than that of the second, and the PCCC of the second model was 0.50% greater than that of the first. The R code to reproduce the results is in Appendix 3.

7.3 Ordinal Logistic Regression

As described at the beginning of this chapter, the ordinal logistic model is given in model (7.1) but with F the cumulative logistic distribution. Again, as in the ordinal probit regression model, the posterior distribution of the parameter is not easy to simulate and numerical methods are required. Here we describe the Gibbs sampler proposed by Montesinos-López et al. (2015b), which in addition to the latent variable L_i in the representation of model (7.1), the parameter vectors are also augmented with a Pólya-Gamma latent random variable.

By using the following identity proposed by Polson et al. (2013):

$$\frac{[\exp(\eta)]^a}{[1 + \exp(\eta)]^b} = 2^{-b} \exp(k\eta) \int_0^\infty \exp\left(-\frac{\eta^2}{2}\omega\right) f_\omega(\omega; b, 0) d\omega,$$

where $k = a - b/2$ and $f_\omega(\omega; b, 0)$ is the density of a Pólya-Gamma random variable ω with parameters b and $d = 0$ ($\omega \sim \text{PG}(b, d)$), (see Polson et al. (2013) for details), the joint distribution of the vector of observations, $\mathbf{Y} = (Y_1, \dots, Y_n)^\top$, and the latent variables $\mathbf{L} = (L_1, \dots, L_n)^\top$ can be expressed as

$$\begin{aligned} f(\mathbf{y}, \mathbf{l} | \boldsymbol{\beta}, \boldsymbol{\gamma}) &= \prod_{i=1}^n f_{L_i}(l_i) I_{(\gamma_{y_i-1}, \gamma_{y_i})}(l_i) \\ &= \prod_{i=1}^n \frac{\exp(-l_i - \eta_i)}{[1 + \exp(-l_i - \eta_i)]^2} I_{(\gamma_{y_i-1}, \gamma_{y_i})}(l_i) \\ &\propto \prod_{i=1}^n \left[\int_0^\infty \exp\left(-\frac{(l_i + \eta_i)^2}{2}\omega_i\right) f_{\omega_i}(\omega_i; 2, 0) d\omega_i \right] I_{(\gamma_{y_i-1}, \gamma_{y_i})}(l_i), \end{aligned}$$

where $\eta_i = \mathbf{x}_i^\top \boldsymbol{\beta}$ and $\omega_1, \dots, \omega_n$ are independent random variables with the same Pólya-Gamma distribution with parameters $b = 2$ and $d = 0$.

Now, the Bayesian specification developed in Montesinos-López et al. (2015b) assumes the same priors as for the ordinal probit model (BRR prior model), except that now for the threshold parameter vector $\boldsymbol{\gamma}$, the prior distribution proposed by Sorensen et al. (1995) is adopted, which is the distribution of the order statistics of a random sample of size $C - 1$ of the uniform distribution in (γ_L, γ_U) , that is,

$$f(\boldsymbol{\gamma}) = (C - 1)! \left(\frac{1}{\gamma_U - \gamma_L} \right)^{C-1} I_{\{\boldsymbol{\gamma} \in \mathbf{S}_\gamma\}},$$

where $\mathbf{S}_\gamma = \{(\gamma_1, \dots, \gamma_{C-1}) : \gamma_L < \gamma_1 < \gamma_2 < \dots < \gamma_{C-1} < \gamma_U\}$.

Then, by conditioning on the rest of parameters, including the latent Pólya-Gamma random variables $\boldsymbol{\omega} = (\omega_1, \dots, \omega_n)^\top$, the full conditional of γ_c is given by

$$\begin{aligned} f(\gamma_c | -) &\propto f(\mathbf{y}, \mathbf{l}, \boldsymbol{\omega} | \boldsymbol{\beta}, \boldsymbol{\gamma}) f(\boldsymbol{\gamma}) \\ &\propto \left[\prod_{i=1}^n I_{(\gamma_{y_i-1}, \gamma_{y_i})}(l_i) \right] (C - 1)! \left(\frac{1}{\gamma_U - \gamma_L} \right)^{C-1} I_{\mathbf{S}_\gamma}(\boldsymbol{\gamma}) \\ &\propto I_{(a_c, b_c)}(\gamma_c) \end{aligned}$$

where $f(\mathbf{y}, \mathbf{L}, \boldsymbol{\omega} | \boldsymbol{\beta}, \boldsymbol{\gamma})$ is the joint density of the observations and the vector of the latent random variables, $\mathbf{L}$ and $\boldsymbol{\omega}$, $a_c = \max\{\gamma_c - 1, \max\{l_i : y_i = c\}\}$, and $b_c = \min\{\gamma_c + 1, \min\{l_i : y_i = c + 1\}\}$. So the full conditional for the threshold parameters is also uniform, $\gamma_c | - \sim U(a_c, b_c)$, $c = 1, 2, \dots, C - 1$.

Now, the conditional distribution of β_j (j th element of $\boldsymbol{\beta}$) is given by

$$\begin{aligned} f(\beta_j | \sigma_\beta^2, \boldsymbol{\gamma}, \mathbf{L}, \boldsymbol{\omega}) &\propto f(\mathbf{y}, \mathbf{L}, \boldsymbol{\omega} | \boldsymbol{\beta}, \boldsymbol{\gamma}) f(\boldsymbol{\beta} | \sigma_\beta^2) \\ &\propto \exp \left[-\frac{1}{2} \sum_i^n \omega_i \frac{(l_i + \eta_i)^2}{2} - \frac{1}{2\sigma_\beta^2} \beta_j^2 \right] \\ &\propto \exp \left[-\frac{1}{2} \sum_i^n \omega_i \frac{(e_{ij} + x_{ij}\beta_j)^2}{2} - \frac{1}{2\sigma_\beta^2} \beta_j^2 \right] \\ &\propto \exp \left\{ -\frac{1}{2\tilde{\sigma}_{\beta_j}^2} (\beta_j - \tilde{\beta}_j)^2 \right\}, \end{aligned}$$

where $e_{ij} = l_i + \sum_{k \neq j}^p x_{ik}\beta_k$, $\mathbf{e}_j = [e_{1j}, \dots, e_{nj}]^T$, $\mathbf{x}_j = [x_{1j}, \dots, x_{nj}]^T$, $\tilde{\sigma}_{\beta_j}^2 = (\sigma_\beta^{-2} + \sum_{i=1}^n \omega_i x_{ij}^2)^{-1}$, and $\tilde{\beta}_j = -\tilde{\sigma}_{\beta_j}^2 (\sum_{i=1}^n \omega_i x_{ij} e_{ij})$. From this, the full conditional distribution of β_j is a normal distribution with mean $\tilde{\beta}_j$ and variance $\tilde{\sigma}_{\beta_j}^2$.

The full conditional distribution of σ_β^2 is the same as the one obtained for the ordinal probit model, $\sigma_\beta^2 | - \sim \chi^{-2}(\tilde{v}_\beta, \tilde{S}_\beta)$ with $\tilde{v}_\beta = v_\beta + p$ and $\tilde{S}_\beta = S_\beta + \boldsymbol{\beta}^T \boldsymbol{\beta}$. In a similar fashion, for the latent variable L_i , $i = 1, \dots, n$, it can be seen that its full conditional distributions are also truncated normal in $(\gamma_{y_i-1}, \gamma_{y_i})$ but with mean parameter $-\mathbf{x}_i^T \boldsymbol{\beta}$ and variance $1/\omega_i$, i.e., $L_i | - \sim N(-\mathbf{x}_i^T \boldsymbol{\beta}, \omega_i^{-1})$ truncated in $(\gamma_{y_i-1}, \gamma_{y_i})$, for each $i = 1, \dots, n$. Finally, by following Eq. (5) in Polson et al. (2013), note that the full joint conditional distribution of the Pólya random variables $\boldsymbol{\omega}$ can be written as

$$\begin{aligned} f(\boldsymbol{\omega} | -) &\propto f(\mathbf{y}, \mathbf{L}, \boldsymbol{\omega} | \boldsymbol{\beta}, \boldsymbol{\gamma}) \\ &\propto \prod_{i=1}^n \exp \left(-\frac{(l_i + \eta_i)^2}{2} \omega_i \right) f_{\omega_i}(\omega_i; 2, 0) \\ &\propto \prod_{i=1}^n f_{\omega_i}(\omega_i; 2, l_i + \eta_i). \end{aligned}$$

From here, conditionally to $\boldsymbol{\beta}, \sigma_\beta^2, \boldsymbol{\gamma}$, and $\mathbf{L}$, $\omega_1, \dots, \omega_n$ are independently Pólya-Gamma random variables with parameters $b = 2$ and $d = l_i + \eta_i$, $i = 1, \dots, n$, respectively, that is, $\omega_i | - \sim \text{PG}(2, l_i + \eta_i)$, $i = 1, \dots, n$.

With the above derived full conditionals, a Gibbs sampler exploration of this ordinal logistic regression model can be done with the following steps:

1. Initialize the parameters: $\beta_j = \beta_{j0}, j = 1, \dots, p, \gamma_c = \gamma_{c0}, c = 1, \dots, C - 1, l_i = l_{i0}, i = 1, \dots, n$, and $\sigma_\beta^2 = \sigma_{\beta 0}^2$.
2. Simulate $\omega_1, \dots, \omega_n$ independently Pólya-Gamma random variables with parameters $b = 2$ and $d = l_i + \eta_i, i = 1, \dots, n$.
3. For each $i = 1, \dots, n$, simulate l_i from the normal distribution $N(-\mathbf{x}_i^T \boldsymbol{\beta}, \omega_i^{-1})$ truncated in $(\gamma_{y_i-1}, \gamma_{y_i})$.
4. For each $j = 1, \dots, p$, simulate from the full conditional distribution of β_j , that is, from a $N(\tilde{\beta}_j, \tilde{\sigma}_{\beta_j}^2)$, where $\tilde{\sigma}_{\beta_j}^2 = (\sigma_\beta^{-2} + \sum_{i=1}^n \omega_i x_{ij}^2)^{-1}$ and $\tilde{\beta}_j = -\tilde{\sigma}_{\beta_j}^2 (\sum_{i=1}^n \omega_i x_{ij} e_{ij})$.
5. For each $c = 1, 2, \dots, C - 1$, simulate from the full conditional distribution of γ_c , $\gamma_c \mid - \sim U(a_c, b_c)$, where $a_c = \max \{\gamma_c - 1, \max \{l_i : y_i = c\}\}$ and $b_c = \min \{\gamma_c + 1, \min \{l_i : y_i = c + 1\}\}$.
6. Simulate σ_β^2 from a scaled inverse chi-squared distribution with parameters $\tilde{v}_\beta = v_\beta + p$ and $\tilde{S}_\beta = S_\beta + \boldsymbol{\beta}^T \boldsymbol{\beta}$, that is, from a $\chi^{-2}(\tilde{v}_\beta, \tilde{S}_\beta)$.
7. Repeat 1–6 until a burning period and a desired number of samples are reached.

Similar modifications can be done to obtain Gibbs samplers corresponding to other prior adopted models for the beta coefficients (FIXED, BayesA, BayesB, BayesC, or BL; see Chap. 6 for details of these priors). Also, for the ordinal logistic GBLUP Bayesian regression model specification as done in (7.2) for the ordinal probit model, a Gibbs sampler implementation can be done following the steps below, which can be obtained directly from the ones described above:

1. Initialize the parameters: $b_i = b_{i0}, j = 1, \dots, n, \gamma_c = \gamma_{c0}, c = 1, \dots, C - 1, l_i = l_{i0}, i = 1, \dots, n$, and $\sigma_g^2 = \sigma_{g0}^2$.
2. Simulate $\omega_1, \dots, \omega_n$ independently Pólya-Gamma random variables with parameters $b = 2$ and $d = l_i + b_i, i = 1, \dots, n$.
3. For each $i = 1, \dots, n$, simulate l_i from the normal distribution $N(-b_i, \omega_i^{-1})$ truncated in $(\gamma_{y_i-1}, \gamma_{y_i})$.
4. Simulate $\mathbf{b}$ from $N(\tilde{\mathbf{b}}, \tilde{\boldsymbol{\Sigma}}_b)$, with $\tilde{\boldsymbol{\Sigma}}_b = (\sigma_g^{-2} \mathbf{G}^{-1} + \mathbf{D}_\omega)^{-1}$ and $\tilde{\mathbf{b}} = -\tilde{\boldsymbol{\Sigma}}_b \mathbf{D}_\omega \mathbf{l}$, where $\mathbf{D}_\omega = \text{Diag}(\omega_1, \dots, \omega_n)$.
5. For each $c = 1, 2, \dots, C - 1$, simulate from the full conditional distribution of γ_c , $\gamma_c \mid - \sim U(a_c, b_c)$, where $a_c = \max \{\gamma_c - 1, \max \{l_i : y_i = c\}\}$ and $b_c = \min \{\gamma_c + 1, \min \{l_i : y_i = c + 1\}\}$.
6. Simulate σ_g^2 from a scaled inverse chi-squared distribution with parameters $\tilde{v}_g = v_g + n$ and $\tilde{S}_g = S_g + \mathbf{b}^T \mathbf{b}$, that is, from a $\chi^{-2}(\tilde{v}_g, \tilde{S}_g)$.
7. Repeat 1–6 until a burning period and the desired number of samples are reached.

There is a lot of empirical evidence that there are no large differences between the prediction performance of the probit or logistic regression models. For this reason, here we only explained the Gibbs sampler for ordinal data under a logistic framework and did not provide illustrated examples. Also, we did not provide illustrative examples because it is not possible to implement this logistic ordinal model in BGLR.

7.4 Penalized Multinomial Logistic Regression

An extension of the logistic regression model described in Chap. 3 is the multinomial regression model that is also used to explain or predict a categorical nominal response variable (that does not have any natural order) with more than two categories. For example, a study could investigate the association of markers with diabetes (diabetes and obesity, diabetes but no obesity, obesity but no diabetes, and no diabetes and no obesity); another example could be to study the effects of age group on the histological subtypes of woman cancer (adenocarcinoma, adenosquamous, others), predict the preference in an election among four candidates (C1, C2, C3, and C4) using socioeconomic and demographic variables, etc.

The multinomial logistic regression model assumes that, conditionally to a covariate $\mathbf{x}_i$, a multinomial response random variable Y_i takes one of the categories $1, 2, \dots, C$, with the following probabilities:

$$P(Y_i = c | \mathbf{x}_i) = \frac{\exp(\beta_{0c} + \mathbf{x}_i^T \boldsymbol{\beta}_c)}{\sum_{l=1}^C \exp(\beta_{0l} + \mathbf{x}_i^T \boldsymbol{\beta}_l)}, c = 1, \dots, C, \quad (7.6)$$

where $\boldsymbol{\beta}_c, c = 1, \dots, C$, is a vector of coefficients of the same dimension as $\mathbf{x}$.

Model (7.6) is not identifiable because by evaluating these probabilities with the parameter values $(\beta_{0c}^*, \boldsymbol{\beta}_c^{*T}) = (\beta_{0c} + \beta_{0c}^{**}, \boldsymbol{\beta}_{0c}^T + \boldsymbol{\beta}_c^{**T}), c = 1, \dots, C$, where β_{0c}^{**} and $\boldsymbol{\beta}_c^{**}$ are arbitrary constants and vectors, give equal probabilities than when computing this with the original parameter values $(\beta_{0c}, \boldsymbol{\beta}_c^T), c = 1, \dots, C$. A common constraint that avoids this lack of identifiability is to set $(\beta_{0C}, \boldsymbol{\beta}_C^T) = (0, \mathbf{0}^T)$, although any one of the other $C - 1$ of the vectors could be chosen.

With such a constraint $((\beta_{0C}, \boldsymbol{\beta}_C^T) = (0, \mathbf{0}^T))$, we can identify a model by assuming that the effects of the covariate vector $\mathbf{x}_i$ over the log “odds” of each of the categories $c = 1, \dots, C - 1$ with respect to category C (baseline category) are given (Agresti 2012) by

$$\log \left(\frac{P(Y_i = c | \mathbf{x})}{P(Y_i = C | \mathbf{x})} \right) = \beta_{0c} + \mathbf{x}_i^T \boldsymbol{\beta}_c, \quad c = 1, \dots, C - 1,$$

where the effects of $\mathbf{x}_i$ depend on the chosen response baseline category. Similar expressions can be obtained when using the unconstrained model in (7.6). From

these relationships, the log odds corresponding to the probabilities of any two categories, c and l , can be derived:

$$\log \left(\frac{P(Y_i = c|x)}{P(Y_i = l|x)} \right) = \beta_{0c} - \beta_{0l} + \mathbf{x}_i^T (\boldsymbol{\beta}_c - \boldsymbol{\beta}_l), \quad c = 1, \dots, C-1.$$

Sometimes the number of covariates is larger than the number of observations (for example, in expression arrays and genomic prediction where the number of markers is often larger than the number of phenotyped individuals), so the conventional constraint described above to force identifiability in the model is not enough and some identifiability problems remain. One way to avoid this problem is to use similar quadratic regularization maximum likelihood estimation methods (Zhu and Hastie 2004) as done for some models in Chap. 3, but here only on the slopes ($\boldsymbol{\beta}_c$) for the covariates, under which setting the constraints mentioned before are no longer necessary.

For a given value of the regularization parameter, $\lambda > 0$, the regularized maximum likelihood estimation of the beta coefficients, $\boldsymbol{\beta} = (\beta_{0c}, \boldsymbol{\beta}_c^T, \beta_{02}, \boldsymbol{\beta}_2^T, \dots, \beta_{0C}, \boldsymbol{\beta}_C^T)^T$, is the value of this that maximizes the penalized log-likelihood:

$$\ell_p(\boldsymbol{\beta}; \mathbf{y}) = \ell(\boldsymbol{\beta}; \mathbf{y}) - \lambda \sum_{c=1}^C \boldsymbol{\beta}_c^T \boldsymbol{\beta}_c, \quad (7.7)$$

where $\ell(\boldsymbol{\beta}; \mathbf{y}) = \log [L(\boldsymbol{\beta}; \mathbf{y})]$ is the logarithm of the likelihood and takes the form:

$$\begin{aligned} \ell(\boldsymbol{\beta}; \mathbf{y}) &= \sum_{i=1}^n \log [P(Y_i = y_i | \mathbf{x}_i)] = \sum_{i=1}^n \sum_{c=1}^C I_{\{y_i=c\}} \log \left[\frac{\exp(\beta_{0c} + \mathbf{x}_i^T \boldsymbol{\beta}_c)}{\sum_{l=1}^C \exp(\beta_{0l} + \mathbf{x}_i^T \boldsymbol{\beta}_l)} \right] \\ &= \sum_{i=1}^n \sum_{c=1}^C I_{\{y_i=c\}} (\beta_{0c} + \mathbf{x}_i^T \boldsymbol{\beta}_c) - \sum_{i=1}^n \log \left[\sum_{l=1}^C \exp(\beta_{0l} + \mathbf{x}_i^T \boldsymbol{\beta}_l) \right] \end{aligned} \quad (7.8)$$

When p is large ($p \gg n$), direct optimization of $\ell_p(\boldsymbol{\beta}; \mathbf{y})$ is almost impossible. An alternative is to use the sequential minimization optimization algorithm proposed by Zhu and Hastie (2004), which is applied after a transformation trick is used to make the involved computations feasible, because the number of parameters in the optimization is reduced to only $(n+1)C$ instead of $(p+1)C$.

Another alternative available in the glmnet package is the one proposed by Friedman et al. (2010) that is similar to that of the logistic Ridge regression in Chap. 3. This consists of maximizing (7.7) by using a block-coordinate descent strategy, where each block is formed by the beta coefficients corresponding to each class, $\boldsymbol{\beta}_c^{*T} = (\beta_{0c}, \boldsymbol{\beta}_c^T)$, but with $\ell(\boldsymbol{\beta}; \mathbf{y})$ replaced by a quadratic approximation with respect to beta coefficients of the chosen block, $(\beta_{0c}, \boldsymbol{\beta}_c^T)$, at the current values

of $\boldsymbol{\beta}$ ($\tilde{\boldsymbol{\beta}}$). That is, the update of block c is achieved by maximizing the following function with respect to β_{0c} and $\boldsymbol{\beta}_c$:

$$f_c(\beta_{0c}, \boldsymbol{\beta}_c) = \ell_c^*(\boldsymbol{\beta}; \mathbf{y}) - \lambda \boldsymbol{\beta}_c^T \boldsymbol{\beta}_c, \quad (7.9)$$

where $\ell_c^*(\boldsymbol{\beta}; \mathbf{y}) = -\frac{1}{2} \sum_{i=1}^n w_{ic} (y_{ic}^* - \beta_{0c} - \mathbf{x}_i^T \boldsymbol{\beta}_c)^2 + \tilde{c}$ is the second-order Taylor approximation of $\ell(\boldsymbol{\beta}; \mathbf{y})$ with respect to the beta coefficients that conform block c , $(\beta_{0c}, \boldsymbol{\beta}_c^T)$, about the current estimates $\tilde{\boldsymbol{\beta}} = (\tilde{\beta}_{0c}, \tilde{\boldsymbol{\beta}}_c^T, \tilde{\beta}_{02}, \tilde{\boldsymbol{\beta}}_2^T, \dots, \tilde{\beta}_{0C}, \tilde{\boldsymbol{\beta}}_C^T)^T$; $y_{ic}^* = \tilde{\beta}_{0c} + \mathbf{x}_i^T \tilde{\boldsymbol{\beta}}_c + w_{ic}^{-1} (I_{\{y_i=c\}} - \tilde{p}_c(\mathbf{x}_i))$ is the “working response,” $w_{ic} = \tilde{p}_c(\mathbf{x}_i) \times (1 - \tilde{p}_c(\mathbf{x}_i))$, and $\tilde{p}_c(\mathbf{x}_i)$ is $P(Y_i = c | \mathbf{x}_i)$ given in (7.6) but evaluated at the current parameter values.

When $\mathbf{X}$ is the design matrix with standardized independent variables, the updated parameters of block c , $\hat{\boldsymbol{\beta}}_c^{*T}$, can be obtained by the following formula:

$$\hat{\boldsymbol{\beta}}_c^{*T} = (\mathbf{X}^{*T} \mathbf{W}_c \mathbf{X}^* + \lambda \mathbf{D})^{-1} \mathbf{X}^{*T} \mathbf{W}_c \mathbf{y}^*,$$

where $\mathbf{X}^* = [\mathbf{1}_n \mathbf{X}]$, $\mathbf{W}_c = \text{Diag}(w_{1c}, \dots, w_{nc})$, $\mathbf{y}^* = (y_1^*, \dots, y_n^*)^T$, and $\mathbf{D}$ is an identity matrix of dimension $(p+1) \times (p+1)$ except that in the first entry we have the value of 0 instead of 1. However, in the context of $p \gg n$, a non-prohibited optimization of (7.9) is achieved by using coordinate descent methods as done in the glmnet package and commented in Chap. 3.6.2.

For other penalization terms, a very similar algorithm to the one described before can be used. For example, for Lasso penalty, the penalized likelihood (7.7) is modified as

$$\ell_p(\boldsymbol{\beta}; \mathbf{y}) = \ell(\boldsymbol{\beta}; \mathbf{y}) - \lambda \sum_{c=1}^C \sum_{j=1}^p |\beta_{cj}| \quad (7.10)$$

and block updating can be done as in (7.9), except that now $\boldsymbol{\beta}_c^T \boldsymbol{\beta}_c$ is replaced by $\sum_{j=1}^p |\beta_{cj}|$. Like the penalized logistic regression studied in Chap. 3, the more common approach for choosing the “optimal” regularization parameter λ in the penalized multinomial regression model in (7.7) is by using a k -fold cross-validation strategy with misclassification error as metrics. This will be used here. For more details, see Friedman et al. (2010). It is important to point out that the tuning parameter λ used in glmnet is equal to the one used in the penalized log-likelihood (7.7) and (7.10) but divided by the number of observations.

7.4.1 *Illustrative Examples for Multinomial Penalized Logistic Regression*

Example 4 To illustrate the penalized multinomial regression model, here we considered the ordinal data of Example 1, but considering the data as nominal, which under the prediction paradigm may not be so important. To evaluate the performance of this model with Ridge (7.9) and Lasso penalization (7.10), the same 10 random partitions as used in Example 1 were used in the cross-validation strategy. By combining these two penalizations and including different covariates information, the performance of six resulting models were evaluated: penalized Ridge multinomial logistic regression model (PRMLRM) with markers as predictors (PRMLRM-1); PRMLRM with environment and markers as predictors (PRMLRM-2); PRMLRM with environment, markers, and environment $\times$ marker interaction as predictors (PRMLRM-3); and the penalized Lasso multinomial logistic regression models, PLMLRM-1, PLMLRM-2, and PLMLRM-3, which have the same predictors as PRMLRM-1, PRMLRM-2, and PRMLRM-3, respectively.

The results are shown in Table 7.4, where again metrics BS and PCCC were computed for each partition. We can observe a similar performance of all models with both metrics. The greater difference in the average BS values across the 10 partitions was found between PRMLRM-3 (worse) model and the PLMLRM-1-PLMRM-2 (best) models, given a 1.34% greater average BS value of the first one compared to the other two. With respect to the other metric, the best average PCCC value (0.29) was obtained with model PRMLRM-2, which gave a 2.36% better performance than the average value of the worse PCCC performance obtained with models PRMLRM-1-2 (0.28).

All these models gave a better performance than all models fitted in Example 1, but only a slightly better performance than the models considered in Example 2, which contain additional information besides the marker information [the only inputs included in the models in Example 1 and PRMLRM-1 and PLMLRG-1], also included environment and environment $\times$ marker interaction as predictors. Specifically, even the simpler models in Table 7.4 that use only marker information in the predictor (PRMLRM-1 and PLMLRM-1) gave results comparable to the more complex and competitive models fitted in Example 2. The relative difference in the average BS value between the better models in each of these two sets of compared models (models in Example 2 vs. models in Table 7.4) was 1.25%, while the relative difference in the average PCCC was 3.57%. The R code to reproduce the results of models under Ridge penalization is in Appendix 4. The codes corresponding to the Lasso penalization models are the same as those for Ridge penalization but with the alpha value set to 1 instead of 0, as needs to be done in the basic code of the glmnet to fit the penalized Ridge multinomial logistic regression models:

```
A = cv.glmnet(X, y, family='multinomial', type.measure = "class",
nfolds = 10, alpha = 0)
```

Table 7.4 Brier score (BS) and proportion of cases correctly classified (PCCC) across 10 random partitions, with 80% of the total data set used for training and the rest for testing, under model (7.6), with Ridge and Lasso penalization and different covariable information: the penalized Ridge multinomial logistic regression (PRMLR) model with markers as covariates (PRMLRM-1); the PRMLR model with environment and markers as covariates (PRMLRM-2); the PRMLR model with environment, markers, and environment×marker interaction as covariates (PRMLRM-3); PLMLRM-1, PLRMLRM-2, and PLMLRM-3 denote the penalized Lasso multinomial logistic regression model with the same covariates as in PRMLRM-1, PRMLRM-2, and PRMLRM-3, respectively

Model	PRMLRM-1		PRMLRM-2		PRMLRM-3	
PT	BS	PCCC	BS	PCCC	BS	PCCC
1	0.3755	0.3000	0.3755	0.3000	0.4632	0.3000
2	0.3831	0.3000	0.3831	0.3000	0.3957	0.2667
3	0.3953	0.2667	0.3954	0.2667	0.4052	0.4000
4	0.4093	0.2333	0.4099	0.2333	0.4271	0.0667
5	0.3578	0.3333	0.3578	0.3333	0.3578	0.3333
6	0.3639	0.3333	0.3639	0.3333	0.3639	0.3333
7	0.3885	0.2000	0.3885	0.2000	0.3885	0.2000
8	0.3696	0.3333	0.3700	0.3667	0.3746	0.3333
9	0.3926	0.2333	0.3926	0.2333	0.3926	0.2333
10	0.3955	0.3333	0.3956	0.3333	0.3913	0.2667
Average (SD)	0.3831 (0.01)	0.2866 (0.05)	0.3832 (0.01)	0.29 (0.05)	0.3959 (0.03)	0.2733 (0.09)

Model	PLMLRM-1		PLMLRM-2		PLMLRM-3	
PT	BS	PCCC	BS	PCCC	BS	PCCC
1	0.3644	0.3333	0.3644	0.3333	0.3516	0.4333
2	0.3831	0.3000	0.3831	0.3000	0.3813	0.3000
3	0.3968	0.3000	0.3968	0.3000	0.3996	0.3000
4	0.4147	0.2333	0.4147	0.2333	0.4305	0.1667
5	0.3586	0.3000	0.3586	0.3000	0.3586	0.3000
6	0.3623	0.3000	0.3623	0.3000	0.3667	0.3000
7	0.3885	0.2000	0.3885	0.2000	0.3885	0.2000
8	0.3636	0.3667	0.3636	0.3667	0.3710	0.4000
9	0.3926	0.2333	0.3926	0.2333	0.3926	0.2333
10	0.3913	0.2667	0.3913	0.2667	0.3919	0.2667
Average (SD)	0.3815 (0.01)	0.2833 (0.05)	0.3815 (0.01)	0.2833 (0.05)	0.3832 (0.02)	0.29 (0.08)

where **X** and **y** are the design matrix and vector of response variable for training the multinomial model, and the multinomial model is specified as family=**multinomial**'; with "**class**" we specified the metric that will be used in the inner cross-validation (CV) strategy to internally select the "optimal" value of the regularization parameter λ (over a grid of 100 values of λ , by default). The misclassification error is equal to $1 - \text{PCCC}$; nfolds is the number of folds that are used internally with the inner CV strategy and when this value is not specified, by default it is set to 10; alpha is a mixing parameter for a more general penalty function

(Elastic Net penalty) that when it is set to 0, the Ridge penalty is obtained, while the Lasso penalty is used when alpha is set to 1; however, when alpha takes values between 0 and 1, the Elastic Net penalization is implemented.

A kind of GBLUP implementation of the simpler models (PRMLRM-1 and PLMLRM-1) considered in Example 4 can be done by considering the genomic relationship matrix derived from the marker information and using as predictor the lower triangular factor of the Cholesky decomposition or its root squared matrix. Something similar can be done to include the Env $\times$ Line interaction term. The kinds of “GBLUP” implementation of PRMLRM-1 and PLMLRM-1 are referred to as GMLRM-R1 and GMLRM-L1, respectively. In both cases, the input matrix is $X = Z_L L_g$, where Z_L is the incident matrix of the lines and L_g is the lower triangular part of the Cholesky decomposition of G , $G = L_g L_g^T$. For the rest of the models, this kind of “GBLUP” implementation will be referred to as GMLRM-R2 and GMLRM-R3 for Ridge penalty, and as GMLRM-L2 and GMLRM-L3 for Lasso penalty, and in both cases the input matrix X will be $X = [X_E, Z_L L_g]$ when an environment and genetic effect is present in the predictor (GMLRM-R2, GMLRM-L3), or $X = [X_E, Z_L L_g, Z_{EL}(I_I \otimes L_g)]$ when the genotype $\times$ environment interaction effect is also taken into account (GMLRM-R2, GMLRM-L3), and where I is the number of environments.

The basic code for glmnet implementation of the GMLRM-L3 model is given below, from which the other five kinds of “GBLUP” implementations described above can be performed by only removing the corresponding predictors and changing to 0 the alpha value in the Ridge penalty:

```
Lg = t(chol(G))
#Environment matrix design
XE = model.matrix(~Env, data=dat_F)
#Matrix design of the genetic effect
ZL = model.matrix(~0+GID, data=dat_F)
ZLa = ZL%*%Lg
#Environment-genetic interaction
ZEL = model.matrix(~0+GID:Env, data=dat_F)
ZELa = ZEL%*%kronecker(diag(dim(XE)[2]), Lg)
X = cbind(XE[, -1], ZLa, ZELa) #Input X matrix
A = cv.glmnet(X, y, family='multinomial', type.measure = "class",
              alpha = 1) #alpha=0 for Ridge penalty
```

Example 5 Here we considered the data in Example 4 to illustrate the implementation of the following six kinds of “GBLUP” models: GMLRM-R1, GMLRM-R2, GMLRM-R3, GMLRM-L1, GMLRM-L2, and GMLRM-L3. To evaluate the performance of these models, the same CV strategy was used, where for each of the 10 random partitions, 80% of the full data set was taken to train the models and the rest to evaluate their performance.

Table 7.5 Brier score (BS) and proportion of cases correctly classified (PCCC) across 10 random partitions, with 80% of the data used for training and the rest for testing, for multinomial model (7.6) under the six types of “GBLUP” implementation of the Ridge and Lasso penalization in Example 4 that were obtained by varying the input information: GMLRM-R1 ($X = Z_I L_g$), GMLRM-R2 ($X = [X_E, Z_I L_g]$), GMLRM-R3 ($X = [X_E, Z_I L_g, Z_{EL}(I_I \otimes L_g)]$), GMLRM-L1 ($X = Z_I L_g$), GMLRM-L2 ($X = [X_E, Z_I L_g]$), and GMLRM-L3 ($X = [X_E, Z_I L_g, Z_{EL}(I_I \otimes L_g)]$), where the first three use Ridge penalization and the last three Lasso penalty

Model	GMLRM-R1		GMLRM-R2		GMLRM-R3	
PT	BS	PCCC	BS	PCCC	BS	PCCC
1	0.3639	0.3000	0.3625	0.3333	0.3641	0.3333
2	0.3831	0.3000	0.3831	0.3000	0.4187	0.3000
3	0.3705	0.3333	0.3773	0.3000	0.4078	0.2333
4	0.4083	0.2667	0.4083	0.2333	0.4172	0.2667
5	0.3578	0.3333	0.3578	0.3333	0.3578	0.3333
6	0.3639	0.3333	0.3639	0.3333	0.3634	0.3667
7	0.3885	0.2000	0.3885	0.2000	0.3889	0.2000
8	0.3697	0.3667	0.3697	0.3667	0.3875	0.3333
9	0.3926	0.2333	0.3926	0.2333	0.3970	0.2667
10	0.3913	0.2667	0.3913	0.2667	0.3913	0.2667
Average (SD)	0.3789 (0.01)	0.2933 (0.05)	0.3795 (0.01)	0.29 (0.05)	0.3893 (0.02)	0.29 (0.05)
Model	GMLRM-L1		GMLRM-L2		GMLRM-L3	
PT	BS	PCCC	BS	PCCC	BS	PCCC
1	0.3865	0.3667	0.3505	0.4000	0.3598	0.3667
2	0.3831	0.3000	0.3831	0.3000	0.3806	0.3000
3	0.3946	0.2667	0.3946	0.2667	0.3948	0.2667
4	0.4147	0.2333	0.4147	0.2333	0.4147	0.2333
5	0.3564	0.3333	0.3564	0.3333	0.3823	0.2667
6	0.3578	0.3667	0.3578	0.3667	0.3639	0.3333
7	0.4078	0.1333	0.4078	0.1333	0.4291	0.1333
8	0.3670	0.3667	0.3670	0.3667	0.3684	0.3667
9	0.3926	0.2333	0.3926	0.2333	0.3926	0.2333
10	0.3854	0.3333	0.3854	0.3333	0.3913	0.2667
Average (SD)	0.3845 (0.01)	0.2933 (0.07)	0.3809 (0.02)	0.2966 (0.08)	0.3877 (0.02)	0.2766 (0.07)

The results presented in Table 7.5 are comparable to those given in Table 7.4 for Example 4. The relative difference between the models in Table 7.4 with better average BS value (PLMLRM-1) and the model in Table 7.5 with better average BS value (GMLRM-R1) is 0.6990%, in favor of the latter model. Regarding the average PCCC, the “kind” of GBLUP model with better value, GMLRM-L2, was 2.2989% greater than the average PCCC value of the better models in Table 7.4, PMLR-R2 and PMLR-L3. Furthermore, for all models in Table 7.5, the greatest difference between the average BS values was observed in models GMLRM-1 (better) and GMLRM-R3 (worse), a relative difference of 2.748%. The best average PCCC performance was observed with model GMLRM-L2, while the worst one was with model GMLRM-L3, a relative difference of 7.23% between the best and the worst.

The R code to reproduce the results in Table 7.5 for the first three models, see Appendix 5. By only changing the value of alpha to 1, the corresponding R code allows implementing the last three Lasso penalty models.

7.5 Penalized Poisson Regression

In contrast to ordinal data, sometimes the response value is not known to be upper bound, rather it is often a count or frequency data, for example, the number of customers per minute in a specific supermarket, number of infected spikelets per plant, number of defects per unit, number of seeds per plant, etc. The most often used count model to relate a set of explanatory variables with a count response variable is Poisson regression.

Given vector covariates $\mathbf{x}_i = (x_{i1}, \dots, x_{ip})^T$, the Poisson log-linear regression modeled the number of events Y_i , as a Poisson random variable with mass density

$$P(Y_i = y | \mathbf{x}_i) = \frac{\lambda_i^y \exp(-\lambda_i)}{y!}, y = 0, 1, 2, \dots, \quad (7.11)$$

where the expected value of Y_i is given by $\lambda_i = \exp(\beta_0 + \mathbf{x}_i^T \boldsymbol{\beta}_0)$ and $\boldsymbol{\beta}_0 = (\beta_1, \dots, \beta_p)^T$ is the vector of beta coefficients effect of the covariates. For example, Y_i may be the number of defects of a product under specific production conditions (temperature, raw material, schedule, operator, etc.) or the number of spikelets per plant which can be predicted from environment and marker information, etc.

For the parameter estimation of this model, the more usual method is maximum likelihood estimation, but this is not suitable in the context of a large number of covariates, so here, as in the logistic and multinomial models, we describe the penalized likelihood method. For a Ridge penalty, the penalized log-likelihood function is given by

$$\begin{aligned} \ell_p(\beta_0, \boldsymbol{\beta}_0; \mathbf{y}) &= \sum_{i=1}^n \log [f_{Y_i}(y_i; \beta_0, \boldsymbol{\beta}_0)] - \lambda \sum_{j=1}^p \beta_j^2 \\ &= \sum_{i=1}^n y_i (\beta_0 + \mathbf{x}_i^T \boldsymbol{\beta}_0) - \sum_{i=1}^n \exp(\beta_0 + \mathbf{x}_i^T \boldsymbol{\beta}_0) - \sum_{i=1}^n \log(y_i!) - \frac{\lambda}{2} \sum_{j=1}^p \beta_j^2. \end{aligned}$$

This can be optimized directly by using traditional numerical methods, but in a general genomic prediction context where $p \gg n$, other alternatives could be more useful. One is the same procedure described before for multinomial and logistic regressions, in which the log-likelihood is replaced by a second-order approximation at the current beta coefficients and then the resulting weighted least-squares problem is solved by coordinate descent methods. This method for Poisson regression is also

implemented in the `glmnet` package and the basic code for implementing this is as follows:

```
A = cv.glmnet(X, y, family='poisson', type.measure = "mse", nfolds =
10,
              alpha = 0)
```

where again $\mathbf{X}$ and $\mathbf{y}$ are the design matrix and vector of response variable for training the Poisson model which is specified by `family='poisson'`; the metric that will be used internally in the inner cross-validation (CV) strategy to choose the “optimal” regularization parameter λ (over a grid of 100 values of λ , by default) is specified in “`type.measure`”. Additionally, to the mean square error (**mse**), the deviance (**deviance**) and the mean absolute error (**mae**) can also be adopted. The other parameters are the same as described for logistic and multinomial regression models. For implementing the Lasso Poisson regression (Poisson regression with Lasso penalty), this same code can be used by only making the argument `alpha = 1`, instead of 0. Further, in the genomic context, as accomplished before for other kinds of responses (continuous, binary, nominal), with this function more complex models involving marker (genetic), environment, or/and marker $\times$ environment interaction effects can also be implemented.

Like the predictors described for the multinomial model, all these predictors are shown in Table 7.6, where the first three (PRPRM-1–3) correspond to Poisson Ridge

Table 7.6 Penalized Poisson regression models with Ridge or Lasso penalties and different covariate information: penalized Ridge Poisson regression (PRPRM) model with markers as covariates (PRPRM-1); PRPRM model with environment and markers as covariates (PRPRM-2); PRPRM model with environment, markers, and environment $\times$ marker interaction as covariates (PRPRM-3); PLPRM-1, PLPRM-2, and PLPRM-3 denote the corresponding penalized Lasso Poisson regression models. GMLRM-R1–R3 are the corresponding GBLUP implementation of the PRPRM models, while GMLRM-L1–L3 are the respective GBLUP of the PLPRM models. $\mathbf{X}_E$, $\mathbf{X}_M$, and $\mathbf{X}_{EM}$ are the design matrices of environment, markers and environment $\times$ marker interaction, $\mathbf{Z}_L$ and $\mathbf{Z}_{EL}$ are the incident matrices of lines and environment $\times$ line interaction effects, and $\mathbf{L}_g$ is the lower triangular part of the Cholesky decomposition of $\mathbf{G}$, $\mathbf{G} = \mathbf{L}_g \mathbf{L}_g^T$

Penalization	Model	Predictor effects	Matrix design
Ridge	PRPRM-1	Markers	$\mathbf{X} = \mathbf{X}_M$
	PRPRM-2	Env + Markers	$\mathbf{X} = [\mathbf{X}_E, \mathbf{X}_M]$
	PRPRM-3	Env + Markers + Env $\times$ Marker	$\mathbf{X} = [\mathbf{X}_E, \mathbf{X}_M, \mathbf{X}_{EM}]$
Lasso	PLPRM-1	Markers	$\mathbf{X} = \mathbf{X}_M$
	PLPRM-2	Env + Markers	$\mathbf{X} = [\mathbf{X}_E, \mathbf{X}_M]$
	PLPRM-3	Env + Markers + Env $\times$ Marker	$\mathbf{X} = [\mathbf{X}_E, \mathbf{X}_M, \mathbf{X}_{EM}]$
Ridge	GPRM-R1	Genetic	$\mathbf{X} = \mathbf{Z}_L \mathbf{L}_g$
	GPRM-R2	Env + Genetic	$\mathbf{X} = [\mathbf{X}_E, \mathbf{Z}_L \mathbf{L}_g]$
	GPRM-R3	Env + Genetic + Env $\times$ Genetic	$\mathbf{X} = [\mathbf{X}_E, \mathbf{Z}_L \mathbf{L}_g, \mathbf{Z}_{EL}(\mathbf{I}_I \otimes \mathbf{L}_g)]$
	GPRM-L1	Genetic	$\mathbf{X} = \mathbf{Z}_L \mathbf{L}_g$
Lasso	GPRM-L2	Env + Genetic	$\mathbf{X} = [\mathbf{X}_E, \mathbf{Z}_L \mathbf{L}_g]$
	GPRM-L3	Env + Genetic + Env $\times$ Genetic	$\mathbf{X} = [\mathbf{X}_E, \mathbf{Z}_L \mathbf{L}_g, \mathbf{Z}_{EL}(\mathbf{I}_I \otimes \mathbf{L}_g)]$

regression with marker, environment plus marker, and environment plus markers plus environment $\times$ marker interaction effects, respectively. The second group of three models (PLPRM-1–3) are, respectively, the same as before but with the Lasso penalization. The third group—three models (GRPRM-R1–R3)—are the corresponding GBLUP implementation of the first three predictors, while the last group—three models (GLPRM-L1–L3)—corresponds to the GBLUP implementation of the second group—three Lasso penalized models.

Example 6 To illustrate how to fit the Poisson model with the predictors given in Table 7.6 and compare these in terms of the mean squared error of prediction, we considered a small data set that is part of the data set used in Montesinos-López et al. (2016). It consists of 50 lines in three environments and a total of 1635 markers. The performance evaluation was done using 10 random partitions where 80% of the data was used to train the model and the rest to test the model. The glmnet R package was used to train all these regression models.

The results are shown in Table 7.7, where the first six models are the first six penalized Poisson regression models given in Table 7.6. With respect to the Ridge penalized models (PRPRM-1–3), we can observe a slightly better performance of the more complex models that take into account environment, marker, and environment $\times$ marker interaction effects (PRPRM-3). As for the penalized Lasso Poisson regression models (PLPRM-1–3), but with a more notable difference, the more complex models also resulted in better performance, and this was also better than the penalized Ridge model (PRPRM-3).

Furthermore, in its GBLUP implementation of the Poisson regression models with Ridge penalty, better MSE performance was obtained with the model that includes environment and genetic effects (GPRM-R2) and this was the best among all Ridge penalized models (PRPRM-1–3 and GPRM-R1–R3); the average MSE of the best PRPRM (PRPRM-3) is 7.5668 greater than the average MSE of the best GPRM with Ridge penalty (GPRM-R2). Under the GBLUP implementations of the Poisson regression models with Lasso penalty (GPRM-L1–L3), the best performance prediction was also obtained when using the environment and genetic effects (GPRM-R2), but its average MSE was slightly different (by only 2.71%) than that obtained with the average MSE of the best Lasso penalized Poisson regression model (PLPRM-3). Additionally, this GBLUP implementation (GPRM-L2) also showed the best average MSE performance among all 12 implemented models, and a notable difference with the second best (GPRM-R2) that gave a relative MSE that was 12.01% greater than that for GPRM-L2 model.

The worse average MSE performance of model **GPRM-L3** is because of the high value of the MSE obtained in partition 2. However, this high variability of the MSE observed across partitions can be unexpected for other larger data sets.

The R code to reproduce these results is shown in Appendix 6.

Table 7.7 Mean square error of prediction (MSE) of six penalized Poisson regression models with Ridge or Lasso penalties and different covariate information (first six models) and MSE of the GBLUP implementation of the corresponding models (second six models), across 10 random partitions, with 80% of the total data set used for training and the rest for testing. See Table 7.6 for details of each model

Model	PRPRM-1	PRPRM-2	PRPRM-3	PLPRM-1	PLPRM-2	PLPRM-3
PT	MSE	MSE	MSE	MSE	MSE	MSE
1	3.7346	3.7380	3.5678	3.7573	3.0309	3.0019
2	2.0492	2.1012	2.1012	2.1194	2.0408	2.0511
3	2.6610	2.6757	2.7178	2.6642	2.1885	2.1930
4	4.6873	4.7079	4.4640	4.5714	4.3558	3.8514
5	3.4277	3.4252	3.3573	3.3317	2.5760	2.6570
6	2.6719	2.6589	2.6592	2.9291	2.1381	2.1104
7	2.7093	2.6601	2.6601	2.6601	3.2845	2.4267
8	3.8690	3.8791	3.7156	3.7156	3.1182	3.1605
9	1.9520	1.9699	1.8081	1.8081	1.5642	1.7106
10	4.6182	4.6086	4.5979	4.6416	3.6876	3.8178
Average (SD)	3.238 (0.98)	3.2424 (0.97)	3.1648 (0.93)	3.2198 (0.96)	2.7984 (0.85)	2.698 (0.74)

Model	GPRM-R1	GPRM-R2	GPRM-R3	GPRM-L1	GPRM-L2	GPRM-L3
PT	MSE	MSE	MSE	MSE	MSE	MSE
1	3.7110	3.4279	3.8221	3.8243	2.8399	2.9275
2	2.1012	1.7678	2.1050	2.1012	1.8690	69.3985
3	2.6606	2.2621	2.7178	2.7178	2.0578	1.9326
4	4.7780	4.8127	4.2797	4.2820	3.9353	3.3327
5	3.5584	3.0863	3.5038	3.3910	2.6122	2.6139
6	2.6117	2.0641	2.8482	2.7042	2.0844	1.9349
7	2.7123	2.5304	2.9432	2.6601	2.7140	2.3736
8	3.7156	3.5213	3.7107	3.6467	2.9335	3.0768
9	1.8100	1.6769	1.8005	1.8081	1.4778	1.6548
10	4.5861	4.2733	4.5821	4.8303	3.7439	4.1025
Average (SD)	3.2244 (1.0)	2.9422 (1.06)	3.2312 (0.9)	3.1965 (0.96)	2.6267 (0.79)	9.3347 (21.11)

7.6 Final Comments

In this chapter, we give the fundamentals of some popular Bayesian and classical regression models for categorical and count data. We also provide many practical examples for implementing these models for genomic prediction. The examples used take into account in the predictor not only the effect of markers or genotypes but also illustrate how to take into account the effects of environment, genotype $\times$ environment interaction, and marker $\times$ environment interaction. Also, for each type of response variables, we calculated the prediction performance using appropriate

metrics, which is very important since the metrics for evaluating the prediction performance are dependent upon the type of response variable. It is important to point out that the components to include in the predictor are not restricted to those illustrated here, since the user can include other components as main effects or interaction effects in similar fashion as illustrated in the examples provided.

Appendix 1

```
rm(list=ls(all=TRUE))
library(BGLR)
load('dat_ls.RData', verbose=TRUE)
#Phenotypic data
dat_F = dat_ls$dat_F
#Marker information
dat_M = dat_ls$dat_M
#Trait response values
y = dat_F$y
summary(dat_F)
#Sorting data phenotypic data set first by Environment, and then by GID
dat_F = dat_F[order(dat_F$Env, dat_F$GID), ]
#10 random partitions
K = 10; n = length(y)
set.seed(1)
PT = replicate(K, sample(n, 0.20*n))

#Brier score function
BS_f<-function(y,p)
{
  n = length(y)
  CM = matrix(0, nr=n, nc=dim(p) [2])
  CM[cbind(1:n,y)] <- 1
  mean(rowSums((p-CM)^2))/2
}
#Matrix design of markers for all observations
Pos = match(dat_F$GID, row.names(dat_M))
X = dat_M[Pos,]
X = scale(X)
#Ordinal BRR model
ETA_BRR = list(list(X=X, model='BRR'))
Tab = data.frame(PT = 1:K, BS=NA, PCC=NA)
set.seed(1)
for(k in 1:K)
{
  Pos_tst = PT[,k]
  y_NA = y
  y_NA[Pos_tst] = NA
  A = BGLR(y=y_NA, response_type='ordinal', ETA=ETA_BRR,
    nIter = 1e4, burnIn = 1e3, verbose = FALSE)
  Tab$BS[k] = BS_f(y[Pos_tst], A$probs[Pos_tst,])
}
```

```

  Tab$PCC[k] = mean(y[Pos_tst]==apply(A$probs,1,which.max)[Pos_tst])
}
Tab
#Ordinal Bayesian GBLUP model
#Genomic relationship matrix
X_M = scale(dat_M)
G = (X_M%*%t(X_M))/dim(X_M)[2]
dat_F$GID = factor(dat_F$GID,levels=colnames(G))
Z_L = model.matrix(~0+GID,data=dat_F)
Ga = Z_L%*%G%*%t(Z_L)
ETA_GBLUP = list(list(K=Ga,model='RKHS'))
Tab = data.frame(PT = 1:K,BS=NA,PCC=NA)
set.seed(1)
for(k in 1:K)
{
  Pos_tst = PT[,k]
  y_NA = y
  y_NA[Pos_tst] = NA
  A = BGLR(y=y_NA,response_type='ordinal',ETA=ETA_GBLUP,
    nIter = 1e4,burnIn = 1e3,verbose = FALSE)
  Tab$BS[k] = BS_f(y[Pos_tst],A$probs[Pos_tst,])
  Tab$PCC[k] = mean(y[Pos_tst]==apply(A$probs,1,which.max)[Pos_tst])
}
Tab

#Ordinal BayesA model
ETA_BA = list(list(X=X,model='BayesA'))
Tab = data.frame(PT = 1:K,BS=NA,PCC=NA)
set.seed(1)
for(k in 1:K)
{
  Pos_tst = PT[,k]
  y_NA = y
  y_NA[Pos_tst] = NA
  A = BGLR(y=y_NA,response_type='ordinal',ETA=ETA_BA,
    nIter = 1e4,burnIn = 1e3,verbose = FALSE)
  Tab$BS[k] = BS_f(y[Pos_tst],A$probs[Pos_tst,])
  Tab$PCC[k] = mean(y[Pos_tst]==apply(A$probs,1,which.max)[Pos_tst])
}
Tab

#Ordinal BayesB model
ETA_BB = list(list(X=X,model='BayesB'))
Tab = data.frame(PT = 1:K,BS=NA,PCC=NA)
set.seed(1)
for(k in 1:K)
{
  Pos_tst = PT[,k]
  y_NA = y
  y_NA[Pos_tst] = NA
  A = BGLR(y=y_NA,response_type='ordinal',ETA=ETA_BB,
    nIter = 1e4,burnIn = 1e3,verbose = FALSE)
  Tab$BS[k] = BS_f(y[Pos_tst],A$probs[Pos_tst,])
  Tab$PCC[k] = mean(y[Pos_tst]==apply(A$probs,1,which.max)[Pos_tst])
}

```

```

}
Tab
#Ordinal BayesC model
ETA_BC = list(list(X=X,model='BayesC'))
Tab = data.frame(PT = 1:K, BS=NA, PCC=NA)
set.seed(1)
for(k in 1:K)
{
  Pos_tst = PT[,k]
  y_NA = y
  y_NA[Pos_tst] = NA
  A = BGLR(y=y_NA,response_type='ordinal',ETA=ETA_BC,
    nIter = 1e4,burnIn = 1e3,verbose = FALSE)
  Tab$BS[k] = BS_f(y[Pos_tst],A$probs[Pos_tst,])
  Tab$PCC[k] = mean(y[Pos_tst]==apply(A$probs,1,which.max)[Pos_tst])
}
Tab

#Ordinal BL model
ETA_BL = list(list(X=X,model='BL'))
Tab = data.frame(PT = 1:K, BS=NA, PCC=NA)
set.seed(1)
for(k in 1:K)
{
  Pos_tst = PT[,k]
  y_NA = y
  y_NA[Pos_tst] = NA
  A = BGLR(y=y_NA,response_type='ordinal',ETA=ETA_BL,
    nIter = 1e4,burnIn = 1e3,verbose = FALSE)
  Tab$BS[k] = BS_f(y[Pos_tst],A$probs[Pos_tst,])
  Tab$PCC[k] = mean(y[Pos_tst]==apply(A$probs,1,which.max)[Pos_tst])
}
Tab

```

Appendix 2

```

rm(list=ls(all=TRUE))
library(BGLR)
load('dat_ls.RData',verbose=TRUE)
#Phenotypic data
dat_F = dat_ls$dat_F
#Marker information
dat_M = dat_ls$dat_M
y = dat_F$y
dat_F = dat_F[order(dat_F$Env,dat_F$GID),]
#PT
#10 random partitions
K = 10
n = length(y)
set.seed(1)

```

```

PT = replicate(K, sample(n, 0.20*n))
#Brier score
BS_f<-function(y,p)
{
  n = length(y)
  CM = matrix(0,nr=n,nc=dim(p)[2])
  CM[cbind(1:n,y)]<-1
  mean(rowSums((p-CM)^2))/2
}
#Matrix design of markers
Pos = match(dat_F$GID,row.names(dat_M))
X = dat_M[Pos,]
X = scale(X)

#Environment matrix design
XE = model.matrix(~Env,data=dat_F)[,-1]
#Environment-marker interaction
Envs = unique(dat_F$Env)
Markers = colnames(X)
XEM = model.matrix(~0+X:Env,data=dat_F)
Pos = !(colnames(XEM)%in%paste0('X',Markers,' :Env',Envs[length
(Envs)]))
XEM = XEM[,Pos]

#Models to evaluate
Models = c('BRR','BayesA','BayesB','BayesC','BL')
for(m in 1:5)
{
  ETA = list(list(X=XE,model='FIXED'),list(X=X,model=Models[m]),
    list(X=XEM,model=Models[m]))
  Tab = data.frame(PT = 1:K,BS=NA)
  set.seed(1)
  for(k in 1:K)
  {
    Pos_tst = PT[,k]
    y_NA = y
    y_NA[Pos_tst] = NA
    A = BGLR(y=y_NA,response_type='ordinal',ETA=ETA,
      nIter = 1e4,burnIn = 1e3,verbose = FALSE)
    Tab$BS[k] = BS_f(y[Pos_tst],A$probs[Pos_tst,])
    Tab$PCC[k] = mean(y[Pos_tst]==apply(A$probs,1,which.max)[Pos_tst])
    if(dim(A$probs)[2]<5) stop
  }
}
#Saving the output (Tab) for each model
write.csv(Tab,file=paste0('Tab_M3',Models[m],'.csv'),row.names =
FALSE)
}

#GBLUP regression model (4)
#Genomic relationship matrix
G = (dat_M%*%t(dat_M))/dim(dat_M)[2]
dat_F$GID = factor(dat_F$GID,levels=colnames(G))
Z_L = model.matrix(~0+GID,data=dat_F)

```

```

Ga = Z_L%*%G%*%t(Z_L)
#"Genomic" relationship matrix for interaction
Ga2 = kronecker(diag(3),G)
ETA_GBLUP = list(list(K=Ga,model='RKHS'),
                  list(X=XE,model='FIXED'),
                  list(K=Ga2,model='RKHS'))
Tab = data.frame(PT = 1:K,BS=NA)
set.seed(1)
for(k in 1:K)
{
  Pos_tst = PT[,k]
  y_NA = y
  y_NA[Pos_tst] = NA
  A = BGLR(y=y_NA,response_type='ordinal',ETA=ETA_GBLUP,
           nIter = 1e4,burnIn = 1e3,verbose = FALSE)
  Tab$BS[k] = BS_f(y[Pos_tst],A$probs[Pos_tst,])
  Tab$PCC[k] = mean(y[Pos_tst]==apply(A$probs,1,which.max)[Pos_tst])
}
Tab

```

Appendix 3

```

rm(list=ls(all=TRUE))
library(BGLR)
load('Data_Toy_EYT.RData',verbose=TRUE)
#Phenotypic data
dat_F = Pheno_Toy_EYT
head(dat_F)
dat_F = dat_F[order(dat_F$Env,dat_F$GID),]
y = dat_F$Height
#Genomic relationship matrix
G = G_Toy_EYT
#PT
#10 random partitions
K = 10
n = length(y)
set.seed(1)
PT = replicate(K,sample(n,0.20*n))
#Brier score
#The min value of y must be 1
BS_f<-function(y,p)
{
  n = length(y)
  CM = matrix(0,nr=n,nc=dim(p)[2])
  CM[cbind(1:n,y)]<-1
  mean(rowSums((p-CM)^2))/2
}
#M3
dat_F$GID = factor(dat_F$GID,levels=colnames(G))
Z_L = model.matrix(~0+GID,data=dat_F)

```



```

Ga = Z_L%%G%%t(Z_L)
ETA_GBLUP = list(list(K=Ga,model='RKHS'))
#GBLUP
Tab = data.frame(PT = 1:K, BS=NA)
set.seed(1)
for(k in 1:K)
{
  Pos_tst = PT[,k]
  y_NA = y
  y_NA[Pos_tst] = NA
  A = BGLR(y=y_NA,response_type='ordinal',ETA=ETA_GBLUP,
    nIter = 1e4,burnIn = 1e3,verbose = FALSE)
  Tab$BS[k] = BS_f(y[Pos_tst]+1,A$probs[Pos_tst,])
  lbs = as.numeric(colnames(A$probs))
  yp = apply(A$probs,1,function(x) lbs[which.max(x)])
  Tab$PCC[k] = mean(y[Pos_tst]==yp[Pos_tst])
}
Tab

```

#M4

```

#Environment matrix design
XE = model.matrix(~Env,data=dat_F)[,-1]
#"Genomic" relationship matrix for interaction
Ga2 = kronecker(diag(4),G)
ETA_GBLUP = list(list(X=XE,model='FIXED'),
  list(K=Ga,model='RKHS'),
  list(K=Ga2,model='RKHS'))
#GBLUP
Tab = data.frame(PT = 1:K, BS=NA)
set.seed(1)
for(k in 1:K)
{
  Pos_tst = PT[,k]
  y_NA = y
  y_NA[Pos_tst] = NA
  A = BGLR(y=y_NA,response_type='ordinal',ETA=ETA_GBLUP,
    nIter = 1e4,burnIn = 1e3,verbose = FALSE)
  Tab$BS[k] = BS_f(y[Pos_tst]+1,A$probs[Pos_tst,])
  lbs = as.numeric(colnames(A$probs))
  yp = apply(A$probs,1,function(x) lbs[which.max(x)])
  Tab$PCC[k] = mean(y[Pos_tst]==yp[Pos_tst])
}
Tab

```

#M5

```

#Environment matrix design
XE = model.matrix(~Env,data=dat_F)[,-1]
#mean(G!=Ga2)
ETA_GBLUP = list(list(K=Ga,model='RKHS'),
  list(X=XE,model='FIXED'))
Tab = data.frame(PT = 1:K, BS=NA)
set.seed(1)
for(k in 1:K)

```

```

{
  Pos_tst = PT[,k]
  y_NA = y
  y_NA[Pos_tst] = NA
  A = BGLR(y=y_NA, response_type='ordinal', ETA=ETA_GBLUP,
           nIter = 1e4, burnIn = 1e3, verbose = FALSE)
  Tab$BS[k] = BS_f(y[Pos_tst]+1, A$probs[Pos_tst,])
  lbs = as.numeric(colnames(A$probs))
  yp = apply(A$probs, 1, function(x) lbs[which.max(x)])
  Tab$PCC[k] = mean(y[Pos_tst]==yp[Pos_tst])
}
Tab

```

Appendix 4 (Example 4)

```

rm(list=ls(all=TRUE))
library(glmnet)
load('dat_ls.RData', verbose=TRUE)
#Phenotypic data
dat_F = dat_ls$dat_F
#Marker information
dat_M = dat_ls$dat_M
#Response variable
y = dat_F$y
dat_F = dat_F[order(dat_F$Env, dat_F$GID), ]
head(dat_F)
#PT
#10 random partitions
K = 10
n = length(y)
set.seed(1)
PT = replicate(K, sample(n, 0.20*n))

#Brier score function
BS_f<-function(y,p)
{
  n = length(y)
  CM = matrix(0, nr=n, nc=dim(p)[2])
  CM[cbind(1:n,y)] <- 1
  mean(rowSums((p-CM)^2))/2
}
#PRMLRM-1
#Matrix design of markers
Pos = match(dat_F$GID, row.names(dat_M))
X = dat_M[Pos,]
Tab = data.frame(PT = 1:K, BS=NA, PCC=NA)
set.seed(1)
for(k in 1:K)
{
  Pos_tst = PT[,k]

```

```

y_tr = as.character(y[-Pos_tst])
X_tr = X[-Pos_tst,]
A = cv.glmnet(X_tr, y_tr, family='multinomial',
              type.measure = "class", # Misclassification error
              nfolds = 10, parallel = FALSE, thresh=1e-10,
              alpha = 0) # alpha=1 for lasso penalization
p_tst = predict(A, newx = X[Pos_tst,], type='response', s=A$lambda.
min)[, , 1]
y_tst = y[Pos_tst]
Tab$BS[k] = BS_f(y_tst, p_tst)
yp_tst = predict(A, newx = X[Pos_tst,], type='class', s=A$lambda.min)
yp_tst = as.numeric(yp_tst)
Tab$PCC[k] = mean(y_tst==yp_tst)
}
Tab
#PRMLRM-2
# Marker + Env effect
#Environment_marker interaction
XE = model.matrix(~Env, data=dat_F)[, -1]
#Environment matrix design
XEM = model.matrix(~0+X:Env, data=dat_F)
dim(XEM)
head(XEM)[, 1:5]
#XEMa = kronecker(diag(3), unique(X))
#sum((XEM-XEMa)^2)
#Matrix desing for PMR
X = cbind(XE, X)
Tab = data.frame(PT = 1:K, BS=NA, PCC=NA)
set.seed(1)
for(k in 1:K)
{
  Pos_tst = PT[, k]
  y_tr = as.character(y[-Pos_tst])
  X_tr = X[-Pos_tst,]
  A = cv.glmnet(X_tr, y_tr, family='multinomial',
                type.measure = "class",
                nfolds = 10, parallel = FALSE, thresh=1e-10,
                alpha = 0) # alpha=1 for lasso penalization
  p_tst = predict(A, newx = X[Pos_tst,], type='response', s=A$lambda.
min)[, , 1]
  y_tst = y[Pos_tst]
  Tab$BS[k] = BS_f(y_tst, p_tst)
  yp_tst = predict(A, newx = X[Pos_tst,], type='class', s=A$lambda.min)
  yp_tst = as.numeric(yp_tst)
  Tab$PCC[k] = mean(y_tst==yp_tst)
}
Tab
#PRMLRM-3
#Marker + Env + Env:Marker effect
#Matrix design of markers
Pos = match(dat_F$GID, row.names(dat_M))
X = dat_M[Pos,]
#Environment_marker interaction

```

```

XEM = model.matrix(~0+X:Env,data=dat_F)
X = cbind(XE,X,XEM)
Tab = data.frame(PT = 1:K,BS=NA,PCC=NA)
set.seed(1)
for(k in 1:K)
{
  Pos_tst = PT[,k]
  y_tr = as.character(y[-Pos_tst])
  X_tr = X[-Pos_tst,]
  A = cv.glmnet(X_tr, y_tr, family='multinomial',
               type.measure = "class",
               nfolds = 10, parallel = FALSE, thresh=1e-10,
               alpha = 0) # alpha=1 for lasso penalization
  #plot(A)
  p_tst = predict(A,newx = X[Pos_tst,],type='response',s=A$lambda.
min)[,1]
  y_tst = y[Pos_tst]
  Tab$BS[k] = BS_f(y_tst,p_tst)
  yp_tst = predict(A,newx = X[Pos_tst,],type='class',s=A$lambda.min)
  yp_tst = as.numeric(yp_tst)
  Tab$PCC[k] = mean(y_tst==yp_tst)
}
Tab

```

Appendix 5

```

rm(list=ls(all=TRUE))
library(glmnet)
load('dat_ls.RData',verbose=TRUE)
#Phenotypic data
dat_F = dat_ls$dat_F
#Marker information
dat_M = dat_ls$dat_M
y = dat_F$y
dat_F = dat_F[order(dat_F$Env,dat_F$GID),]
#PT
#10 random partitions
K = 10
n = length(y)
set.seed(1)
PT = replicate(K,sample(n,0.20*n))
#Brier score function
BS_f<-function(y,p)
{
  n = length(y)
  CM = matrix(0,nr=n,nc=dim(p)[2])
  CM[cbind(1:n,y)]<-1
  mean(rowSums((p-CM)^2))/2
}
#GMLRM-R1: Genomic effects

```

```

#Matrix design of markers
X = dat_M
#GIDs = sort(row.names(X))
#X = X[match(GIDs,row.names(X)),]
G = tcrossprod(X)/dim(X)[2]
dat_F$GID = factor(dat_F$GID,levels=row.names(G))
ZL = model.matrix(~0+GID,data=dat_F)
Lg = t(chol(G)) #Lower triangular part of the Cholesky descomposition of
G
#Design matrix to use with glmnet
ZLa = ZL%*%Lg
X = ZLa
Tab = data.frame(PT = 1:K,BS=NA,PCC=NA)
set.seed(1)
for(k in 1:K)
{
  Pos_tst = PT[,k]
  y_tr = as.character(y[-Pos_tst])
  X_tr = X[-Pos_tst,]
  A = cv.glmnet(X_tr, y_tr, family='multinomial',
    type.measure = "class", # Misclassification error
    nfolds = 10, parallel = FALSE, thresh=1e-10,
    alpha = 0) #take alpha=1 for lasso penalty, GMLRM-L1
  #Prediction of testing set
  p_tst = predict(A,newx = X[Pos_tst,],type='response',s=A$lambda.
min)[, ,1]
  y_tst = y[Pos_tst]
  Tab$BS[k] = BS_f(y_tst,p_tst)
  yp_tst = predict(A,newx = X[Pos_tst,],type='class',s=A$lambda.min)
  yp_tst = as.numeric(yp_tst)
  Tab$PCC[k] = mean(y_tst==yp_tst)
}
Tab
#GMLRM-R2: Genomic + Env effect
#Matrix design of markers
ZLa = ZL%*%Lg
#Environment_marker interaction
XE = model.matrix(~Env,data=dat_F)
#Matrix design to use in glmnet
X = cbind(XE[, -1], ZLa)
Tab = data.frame(PT = 1:K,BS=NA,PCC=NA)
set.seed(1)
for(k in 1:K)
{
  Pos_tst = PT[,k]
  y_tr = as.character(y[-Pos_tst])
  X_tr = X[-Pos_tst,]
  A = cv.glmnet(X_tr, y_tr, family='multinomial',
    type.measure = "class",
    nfolds = 10, parallel = FALSE, thresh=1e-10,
    alpha = 0) # take alpha=1 for lasso penalty, GMLRM-L2
  p_tst = predict(A,newx = X[Pos_tst,],type='response',s=A$lambda.
min)[, ,1]

```

```

y_tst = y[Pos_tst]
Tab$BS[k] = BS_f(y_tst,p_tst)
#Prediction of testing set
yp_tst = predict(A,newx = X[Pos_tst,],type='class',s=A$lambda.min)
yp_tst = as.numeric(yp_tst)
Tab$PCC[k] = mean(y_tst==yp_tst)
}
Tab

#GMLRM-R3: Genomic + Env effect + Env:Marker effect
#Matrix design of markers
ZLa = ZL%%Lg
#Environment-genetic interaction
ZEL = model.matrix(~0+GID:Env,data=dat_F)
ZELa = ZEL%%kronecker(diag(dim(XE)[2]),Lg)
#Input matrix to use in glmnet
X = cbind(XE,ZLa,ZELa)
Tab = data.frame(PT = 1:K,BS=NA,PCC=NA)
set.seed(1)
for(k in 1:K)
{
  Pos_tst = PT[,k]
  y_tr = as.character(y[-Pos_tst])
  X_tr = X[-Pos_tst,]
  A = cv.glmnet(X_tr,y_tr,family='multinomial',
               type.measure = "class",
               nfolds = 10, parallel = FALSE, thresh=1e-10,
               alpha = 0) # alpha=1 for lasso penalty, GMLRM-L3
  p_tst = predict(A,newx = X[Pos_tst,],type='response',s=A$lambda.
min)[,1]
  y_tst = y[Pos_tst]
  Tab$BS[k] = BS_f(y_tst,p_tst)
  #Prediction of testing set
  yp_tst = predict(A,newx = X[Pos_tst,],type='class',s=A$lambda.min)
  yp_tst = as.numeric(yp_tst)
  Tab$PCC[k] = mean(y_tst==yp_tst)
}
Tab

```

Appendix 6

```

rm(list=ls(all=TRUE))
library(glmnet)
load('dat_ls.RData',verbose=TRUE)
#Phenotypic data
dat_F = dat_ls$dat_F
#Marker information
dat_M = dat_ls$dat_M
y = dat_F$y
dat_F$Env = as.character(dat_F$Loc)

```

```

dat_F = dat_F[order(dat_F$Env, dat_F$GID),]
head(dat_F)
#PT
#10 random partitions
K = 10
n = length(y)
set.seed(1)
PT = replicate(K, sample(n, 0.20*n))
#Penalized Ridge (lasso) Poisson regression models
#PRPRM-1-3 (PLPRM-1-3)
#Matrix design of markers
Pos = match(dat_F$GID, row.names(dat_M))
X = dat_M[Pos,]
#Environment matrix design
XE = model.matrix(~Env, data=dat_F)[-1]
#Environment-marker interaction
Envs = unique(dat_F$Env)
Markers = colnames(X)
XEM = model.matrix(~0+X:Env, data=dat_F)
Pos = !(colnames(XEM) %in% paste0('X', Markers, ':Env', Envs[length(Envs)]))
XEM = XEM[, Pos]
X = X# Design matrix for PRPRM-1 and PLPRM-1
#X = cbind(XE, X) # Uncomment this line to implement PRPRM-2 or PLPRM-2
#X = cbind(XE, X, XEM) # Uncomment this line to implement PRPRM-3 or PLPRM-3
3
Tab = data.frame(PT = 1:K, MSEP=NA)
set.seed(1)
for(k in 1:K)
{
  Pos_tst = PT[,k]
  y_tr = y[-Pos_tst]
  X_tr = X[-Pos_tst,]
  A = cv.glmnet(X_tr, y_tr, family='poisson',
    type.measure = "mse",
    nfolds = 10, parallel = FALSE, thresh=1e-10,
    alpha = 0)# Take alpha=1 for lasso penalty
  yp_tst = predict(A, newx = X[Pos_tst,], type='response', s=A$lambda.min)
  y_tst = y[Pos_tst]
  Tab$MSEP[k] = mean((y_tst-yp_tst)^2)
}
Tab

#GBLUP implementation of penalized Ridge (lasso) Poisson regression models
#GPRM-R1-R3 (GPRM-L1-L3)
#Matrix design of Environments
XE = model.matrix(~Env, data=dat_F)
#Matrix design of markers
X = dat_M
G = tcrossprod(X)/dim(X)[2] # Genomic relationship matrix

```

```

dat_F$GID = factor(dat_F$GID, levels=row.names(G))
dat_F = dat_F[order(dat_F$Env, dat_F$GID), ]
ZL = model.matrix(~0+GID, data=dat_F)
mean(substring(colnames(ZL), 4) != row.names(G))
Lg = t(chol(G))
#Matrix design of genetic effects
ZLa = ZL%*%Lg
#Environment-genetic interaction matrix design
ZEL = model.matrix(~0+GID:Env, data=dat_F)
ZELa = ZEL%*%kronecker(diag(dim(XE)[2]), Lg)
X = ZLa # Matrix design for GPRM-R1 and GPRM-L1
#X = cbind(XE[, -1], ZLa) # Uncomment this line to implement GPRM-R2 or
GPRM-L2
#X = cbind(XE[, -1], ZLa, ZELa) ) # Uncomment this line to implement GPRM-
R3 or GPRM-L3
Tab = data.frame(PT = 1:K, MSEP=NA)
set.seed(1)
for(k in 1:K)
{
  Pos_tst = PT[, k]
  y_tr = y[-Pos_tst]
  X_tr = X[-Pos_tst, ]
  A = cv.glmnet(X_tr, y_tr, family='poisson',
               type.measure = "mse",
               nfolds = 10, parallel = FALSE, thresh=1e-10,
               alpha = 0) # Take alpha=1 for lasso penalty
  yp_tst = predict(A, newx = X[Pos_tst, ], type='response', s=A$lambda.
min)
  y_tst = y[Pos_tst]
  Tab$MSEP[k] = mean((y_tst-yp_tst)^2)
}
Tab

```

References

- Agresti A (2012) Categorical data analysis, 3rd edn. Wiley, Hoboken, NJ
- Albert JH, Chib S (1993) Bayesian analysis of binary and polychotomous response data. *J Am Stat Assoc* 88(422):669–679
- Friedman J, Hastie T, Tibshirani R (2010) Regularization paths for generalized linear models via coordinate descent. *J Stat Softw* 33(1):1–22. <http://www.jstatsoft.org/v33/i01/>
- Gianola D (1980) A method of sire evaluation for dichotomies. *J Anim Sci* 51(6):1266–1271
- Gianola D (1982) Theory and analysis of threshold characters. *J Anim Sci* 54(5):1079–1096
- Gianola D, Foulley JL (1983) Sire evaluation for ordered categorical data with a threshold model. *Genet Select Evol* 15(2):201–224
- McCullagh P (1980) Regression models for ordinal data. *J R Stat Soc B Methodol* 42(2):109–142
- Montesinos-López OA, Montesinos-López A, Pérez-Rodríguez P, de los Campos G, Eskridge K, Crossa J (2015a) Threshold models for genome-enabled prediction of ordinal categorical traits in plant breeding. *G3* 5(2):291–300
- Montesinos-López OA, Montesinos-López A, Crossa J, Burgueño J, Eskridge K (2015b) Genomic-enabled prediction of ordinal data with Bayesian logistic ordinal regression. *G3* 5(10):2113–2126

- Montesinos-López A, Montesinos-López OA, Crossa J, Burgueño J, Eskridge KM, Falconi-Castillo E et al (2016) Genomic Bayesian prediction model for count data with genotype $\times$ environment interaction. *G3* 6(5):1165–1177
- Pérez P, de los Campos G. (2014) BGLR: a statistical package for whole genome regression and prediction. *Genetics* 198(2):483–495
- Polson NG, Scott JG, Windle J (2013) Bayesian inference for logistic models using Pólya–Gamma latent variables. *J Am Stat Assoc* 108(504):1339–1349
- Sorensen DA, Andersen S, Gianola D, Korsgaard I (1995) Bayesian inference in threshold models using Gibbs sampling. *Genet Sel Evol* 27(3):1–21
- Zhu J, Hastie T (2004) Classification of gene microarrays by penalized logistic regression. *Biostatistics* 5(3):427–443

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.