Genome-Wide Regression and Prediction with the BGLR Statistical Package

Paulino Pérez*,[1] and Gustavo de los Campos[†]
*Socio Economía Estadística e Informática, Colegio de Postgraduados 56230, México and [†]Department of Biostatistics, Section on Statistical Genetics, University of Alabama, Birmingham, Alabama 35294

ABSTRACT Many modern genomic data analyses require implementing regressions where the number of parameters (p, *e.g.*, the number of marker effects) exceeds sample size (n). Implementing these *large-p-with-small-n* regressions poses several statistical and computational challenges, some of which can be confronted using Bayesian methods. This approach allows integrating various parametric and nonparametric shrinkage and variable selection procedures in a unified and consistent manner. The BGLR R-package implements a large collection of Bayesian regression models, including parametric variable selection and shrinkage methods and semiparametric procedures (Bayesian reproducing kernel Hilbert spaces regressions, RKHS). The software was originally developed for genomic applications; however, the methods implemented are useful for many nongenomic applications as well. The response can be continuous (censored or not) or categorical (either binary or ordinal). The algorithm is based on a Gibbs sampler with scalar updates and the implementation takes advantage of efficient compiled C and Fortran routines. In this article we describe the methods implemented in BGLR, present examples of the use of the package, and discuss practical issues emerging in real-data analysis.

MANY modern statistical learning problems involve the analysis of high-dimensional data; this is particularly common in genetic studies where, for instance, phenotypes are regressed on large numbers of predictor variables (*e.g.*, SNPs) concurrently. Implementing these *large-p-with-small-n* regressions (where n denotes sample size and p represents the number of predictors) poses several statistical and computational challenges, including how to confront the so-called "curse of dimensionality" (Bellman 1961) as well as the complexity of a genetic mechanism that can involve various types and orders of interactions. Recent developments in shrinkage and variable selection estimation procedures have made the implementation of these *large-p-with-small-n* regressions feasible. Consequently, whole-genome-regression approaches (Meuwissen *et al.* 2001) are becoming increasingly popular for the analysis and prediction of complex traits in plants (*e.g.*, Crossa *et al.* 2010), animals (*e.g.*, Hayes *et al.* 2009, VanRaden *et al.* 2009), and humans (*e.g.*, Yang *et al.* 2010; Makowsky *et al.* 2011; Vazquez *et al.* 2012; de los Campos *et al.* 2013b).

In the past decade a large collection of parametric and nonparametric methods have been proposed and empirical evidence has demonstrated that no single approach performs best across data sets and traits. Indeed, the choice of the model depends on multiple factors such as the genetic architecture of the trait, marker density, sample size and the span of linkage disequilibrium (*e.g.*, de los Campos *et al.* 2013a). Although various software (BLR, Pérez *et al.* 2010; rrBLUP, Endelman 2011; synbreed, Wimmer *et al.* 2012; GEMMA, Zhou and Stephens 2012) exist, most statistical packages implement a few types of methods and the integration of these methods in a unified statistical and computational framework is needed. Motivated by this we have developed the R (R Core Team 2014) package BGLR. The package offers the user great latitude in combining different methods into models for data analysis and is available at CRAN (http://cran.at.r-project.org/web/packages/BGLR/index.html) and at the R-forge website (https://r-forge.r-project.org/projects/bglr/). In this article we discuss the statistical models implemented in BGLR (*Statistical Models, Algorithms, and Data*), present several examples based on real and simulated data (*Application Examples*), and provide benchmarks of computational time and memory usage

Copyright © 2014 by the Genetics Society of America
doi: 10.1534/genetics.114.164442
Manuscript received March 22, 2014; accepted for publication June 26, 2014; published Early Online July 9, 2014.
Available freely online through the author-supported open access option.
Supporting information is available online at http://www.genetics.org/lookup/suppl/doi:10.1534/genetics.114.164442/-/DC1.
[1]Corresponding author: Colegio de Postgraduados, Km. 36.5, Carretera Mexico, Montecillo Texcoco, Estado de México, México 56230. E-mail: perpdgo@colpos.mx

for a linear model (*Benchmark of Parametric Models*). All the examples and benchmarks presented in this article are based on BGLR version 1.0.3. In addition to the scripts presented in the boxes included this article we provide supplementary code in File S1, and text version of all the scripts used to produce the results presented in the article in File S2.

Statistical Models, Algorithms, and Data

The BGLR package supports models for continuous (censored or not) and categorical (binary or ordinal multinomial) traits. We first describe the models implemented in BGLR using a continuous, uncensored, response as example. Further details about censored and categorical outcomes are provided later on this article and in the supporting information, File S1.

For a continuous response (y_i; $i = 1, \ldots, n$) the data equation is represented as $y_i = \eta_i + \varepsilon_i$, where η_i is a linear predictor (the expected value of y_i given predictors) and ε_i are independent normal model residuals with mean zero and variance $w_i^2\sigma_\varepsilon^2$. Here, the $w_i^{'}s$ are user-defined weights (by default BGLR sets $w_i = 1$ for all data-points) and σ_ε^2 is a residual variance parameter. In matrix notation we have

$$\mathbf{y} = \boldsymbol{\eta} + \boldsymbol{\varepsilon},$$

where $\mathbf{y} = \{y_1, \ldots, y_n\}$, $\boldsymbol{\eta} = \{\eta_1, \ldots, \eta_n\}$, and $\boldsymbol{\varepsilon} = \{\varepsilon_1, \ldots, \varepsilon_n\}$.

The linear predictor represents the conditional expectation function, and it is structured as

$$\boldsymbol{\eta} = 1\mu + \sum_{j=1}^{J} \mathbf{X}_j \boldsymbol{\beta}_j + \sum_{l=1}^{L} \mathbf{u}_l, \tag{1}$$

where μ is an intercept, $\mathbf{X}_j$ are design matrices for predictors, $X_j = \{x_{ijk}\}$, $\boldsymbol{\beta}_j$ are vectors of effects associated to the columns of $\mathbf{X}_j$, and $\mathbf{u}_l = \{u_{l1}, \ldots, u_{ln}\}$ are vectors of random effects. The only element of the linear predictor included by default is the intercept. The other elements are user specified. Collecting the above assumptions, we have the following *conditional distribution of the data*:

$$p(\mathbf{y}|\boldsymbol{\theta}) = \prod_{i=1}^{n} N\left(y_i|\mu + \sum_{j=1}^{J}\sum_{k=1}^{K_j} x_{ijk}\beta_{jk} + \sum_{l=1}^{L} u_{li}, \sigma_\varepsilon^2 w_i^2\right),$$

where $\boldsymbol{\theta}$ represents the collection of unknowns, including the intercept, regression coefficients, other random effects, and the residual variance.

Prior density

The prior density is assumed to admit the following factorization:

$$p(\boldsymbol{\theta}) = p(\mu)p(\sigma_\varepsilon^2)\prod_{j=1}^{J} p(\boldsymbol{\beta}_j)\prod_{l=1}^{L} p(\mathbf{u}_l).$$

The *intercept* is assigned a flat prior and the *residual variance* is assigned a scaled-inverse χ^2 density $p(\sigma_\varepsilon^2) = \chi^{-2}(\sigma_\varepsilon^2|S_\varepsilon, \mathrm{df}_\varepsilon)$ with degrees of freedom $\mathrm{df}_\varepsilon(> 0)$ and scale parameter $S_\varepsilon(> 0)$. In the parameterization used in BGLR, the prior expectation of the scaled-inverse χ^2 density $\chi^{-2}(\cdot|S_., \mathrm{df})$ is given by $S_./(\mathrm{df} - 2)$.

Regression coefficients $\{\beta_{jk}\}$ can be assigned either uninformative (*i.e.*, flat) or informative priors. Those coefficients assigned flat priors, the so-called “fixed” effects, are estimated based on information contained in the likelihood solely. For the coefficients assigned informative priors, the choice of the prior plays an important role in determining the type of shrinkage of estimates of effects induced. Figure 1 provides a graphical representation of the prior densities available in BGLR. The *Gaussian prior* induces shrinkage of estimate similar to that of ridge regression (RR; Hoerl and Kennard 1970), where all effects are shrunk to a similar extent; we refer to this model as the Bayesian ridge regression (BRR). The *scaled-t* and *double exponential* (DE) densities have higher mass at zero and thicker tails than the normal density. These priors induces a type of shrinkage of estimates that is size-of-effect dependent (Gianola 2013). The scaled-t density is the prior used in model BayesA (Meuwissen *et al.* 2001), and the DE or Laplace prior is the one used in the BL (Park and Casella 2008). Finally, BGLR implements two *finite mixture priors*: a mixture of a point of mass at zero and a Gaussian slab, a model referred to in the literature on genomic selection (GS) as BayesC (Habier *et al.* 2011), and a mixture of a point of mass at zero and a scaled-t slab, a model known in the GS literature as BayesB (Meuwissen *et al.* 2001). By assigning a nonnull prior probability for the marker effect to be equal to zero, the priors used in BayesB and BayesC have potential for inducing variable selection.

Hyperparameters: Each of the prior densities described above are indexed by one or more parameters that control the type and extent of shrinkage/variable selection induced. We treat most of these regularization parameters as random; consequently a prior is assigned to these unknowns. Table 1 lists, for each of the prior densities implemented, the set of hyperparameters. Further details about how regularization parameters are inferred from the data are given in the supporting information, File S1.

Combining priors: Different priors can be specified for each of the set of coefficients of the linear predictor, $\{\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_J, \mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_L\}$, giving the user great flexibility in building models for data analysis; an example illustrating how to combine different priors in a model is given in Example 2 of *Application Examples*.

Gaussian processes: The vectors of random effects $\mathbf{u}_l$ are assigned multivariate-normal priors with a mean equal to zero and covariance matrix $\mathrm{Cov}(\mathbf{u}_l, \mathbf{u}_l^{'}) = \mathbf{K}_l\sigma_{ul}^2$, where $\mathbf{K}_l$ is a (user-defined) $n \times n$-symmetric positive semidefinite matrix and σ_{ul}^2 is a variance parameter with prior density $\sigma_{ul}^2 \sim \chi^{-2}(\mathrm{df}_l, S_l)$. These random effects can be used to deal with different types of problems, including but not limited to: (a) regressions on pedigree (Henderson 1975),

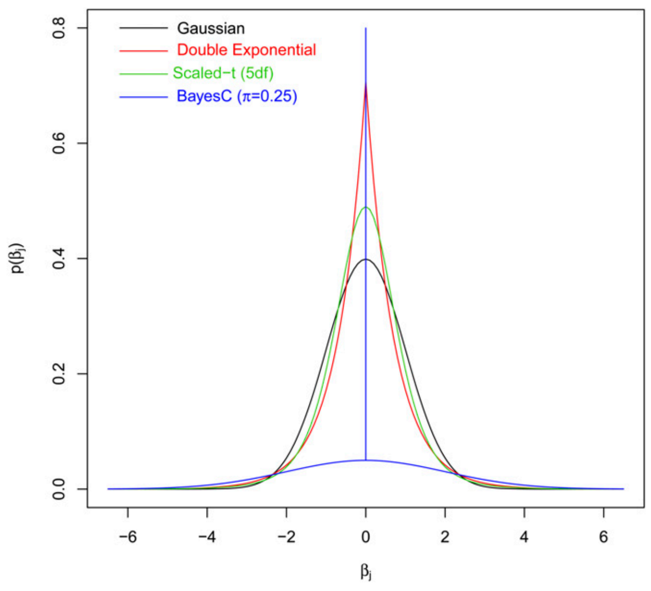

Figure 1 Prior densities of regression coefficients implemented in BGLR (all densities in the figure have null mean and unit variance).

Table 1 Prior densities available for regression coefficients in the BGLR package

Model (prior density)	Hyperparameters	Treatment in BGLR[a]
Flat (FIXED)	Mean (μ_β)	$\mu_\beta = 0$
	Variance (σ^2_β)	$\sigma^2_\beta = 1 \times 1^{10}$
Gaussian (BRR)	Mean (μ_β)	$\mu_\beta = 0$
	Variance (σ^2_β)	$\sigma^2_\beta \sim \chi^{-2}$
Scaled-t (BayesA)	Degrees of freedom (df$_\beta$)	User specified (default value, 5)
	Scale (S_β)	$S_\beta \sim$ *Gamma*
Double exponential (BL)	λ^2	λ fixed, user specified, or $\lambda^2 \sim$ *Gamma*, or $\lambda/\max \sim$ *Beta*[b]
Gaussian mixture (BayesC)	π (prop. of nonnull effects)	$\pi \sim$ *Beta*
	df$_\beta$	User specified (default value, 5)
	S_β	$S_\beta \sim$ *Gamma*
Scaled-t mixture (BayesB)	π (prop. of nonnull effects)	$\pi \sim$ *Beta*
	df$_\beta$	User specified (default value, 5)
	S_β	$S_\beta \sim$ *Gamma*

[a] Further details are given in the supporting information (Section A of File S1).
[b] This approach is further discussed in de los Campos *et al.* (2009b).

(b) genomic BLUP (VanRaden 2008), and (c) nonparametric genomic regressions based on reproducing kernel Hilbert spaces (RKHS) methods (*e.g.*, de los Campos *et al.* 2009a, 2010; Gianola *et al.* 2006). Examples about the inclusion of these Gaussian processes into models for data analysis are given in *Application Examples*.

Categorical Response

The argument `response_type` is used to indicate BGLR whether the response should be regarded as `gaussian`, the default value, or `ordinal`. For continuous traits the response vector should be coercible to numeric; for ordinal traits the response can take on K possible (ordered) values $y_i \in \{1, \ldots, K\}$ (the case where $K = 2$ corresponds to the binary outcome), and the response vector should be coercible to a factor. For categorical traits we use the probit link; here, the probability of each of the categories is linked to the linear predictor according to the following link function

$$P(y_i = k) = \Phi(\eta_i - \gamma_k) - \Phi(\eta_i - \gamma_{k-1}),$$

where $\Phi(\cdot)$ is the standard normal cumulative distribution function, η_i is the linear predictor, specified as described above, and γ_k are threshold parameters, with $\gamma_0 = -\infty$, $\gamma_k \geq \gamma_{k-1}$, $\gamma_K = \infty$. The probit link is implemented using data augmentation (Tanner and Wong 1987); this is done by introducing a latent variable (so-called liability) $l_i = \eta_i + \varepsilon_i$ and a measurement model $y_i = k$ if $\gamma_{k-1} \leq l_i \leq \gamma_k$. For identification purpouses, the residual variance is set equal to one. At each iteration of the Gibbs sampler the unobserved liability scores are sampled from truncate normal densities; once the unobserved liability has been sampled, the Gibbs sampler proceeds as if l_i were observed (see Albert and Chib 1993, for further details).

Missing Values

The response vector can contain missing values. Internally, at each iteration of the Gibbs sampler, missing values are sampled from the corresponding fully conditional density. Missing values in predictors are not allowed.

Censored Data

Censored data in BGLR is described using a triplet $\{a_i, y_i, b_i\}$, the elements of which must satisfy $a_i < y_i < b_i$. Here, y_i is the observed response (*e.g.*, a time-to-event variable, observable only in uncensored data points, otherwise missing, `NA`) and a_i and b_i define lower and upper bounds for the response, respectively. Table 2 gives the configuration of the triplet for the different types of data points. The triplets are provided to BGLR in the form of three vectors (**y**, **a**, **b**). The vectors **a** and **b** have `NULL` as default value; therefore, if only **y** is provided this is interpreted as a continuous trait without censoring. If **a** and **b** are provided together with **y**, data are treated as censored. Censoring is dealt with as a missing data problem; at each iteration of the MCMC algorithm the missing values of y_i, present due to censoring, are sampled from truncated normal densities that satisfy $a_i < y_i < b_i$. An example of how to fit models for censored data are given in the supporting information in File S1 (Section E).

Algorithms

The R-package BGLR draws samples from the posterior density using a Gibbs sampler (Geman and Geman 1984; Casella and George 1992) with scalar updating. For computational convenience the scaled-t and DE densities are represented as infinite mixtures of scaled normal densities (Andrews and Mallows

Table 2 Configuration of the triplet used to described censored data points in BGLR

Type of point	a_i	y_i	b_i
Uncensored	`NULL`	y_i	`NULL`
Right censored	a_i	`NA`	`Inf`
Left censored	`−Inf`	`NA`	b_i
Interval censored	a_i	`NA`	b_i

1974), and the finite-mixture priors are implemented using latent random Bernoulli variables linking effects to components of the mixtures. The computationally demanding steps are performed using compiled C and Fortran code.

User Interface

The R-package BGLR has a user interface similar to that of BLR (Pérez *et al.* 2010); however, we have modified key elements of the interface, and the internal implementation, to provide the user with greater flexibility for model building. All the arguments of the BGLR function have default values, except the vector of phenotypes. Therefore, the simplest call to the BGLR program is as follows.

Box 1: Fitting an intercept model

```
library(BGLR)
y<-50+rnorm(100)
fm<-BGLR(y=y)
```

When the call `fm<-BGLR(y = y)` is made, BGLR fits an intercept model, a total of 1500 cycles of a Gibbs sampler (the default value for the number of iterations; see Box 2) are run, and the first 500 samples are discarded, this is the default value for burn-in (see Box 2). As the Gibbs sampler collects samples, some are saved to the hard drive (only the most recent samples are retained in memory) in files with extension `*.dat`, and the running means required for computing estimates of the posterior means and of the posterior standard deviations are updated; by default, a thinning of 5 is used but this can be modified by the user using the `thin` argument of BGLR. Once the iteration process finishes, BGLR returns a list with estimates and the arguments used in the call.

Inputs

Box 2, displays a list of the main arguments of the BGLR function, a short description follows:

- `y, a, b` (y, coercible to either numeric or factor, `a` and `b` of type numeric) and `response_type` (character) are used to define the response.
- `ETA` (of type `list`) is used to specify the linear predictor. By default it is set to `NULL`, in which case only the intercept is included. Further details about the specification of this argument are given below.
- `nIter`, `burnIn`, and `thin` (all of type `integer`) control the number of iterations of the sampler, the number of samples discarded, and the thinning used to compute posterior means, respectively.
- `saveAt` (`character`) can be used to indicate BGLR where to store the samples and to provide a pre-fix to be appended to the names of the file where samples are stored. By default samples are saved in the current working directory and no pre-fix is added to the file names.
- `S0, df0, R2` (`numeric`) define the prior assigned to the residual variance, `df0` defines the degrees of freedom, and `S0` defines the scale. If the scale is `NULL`, its value is chosen so that the prior mode of the residual variance matches the variance of phenotypes times `1-R2` (see supporting information, Section A of File S1, for further details).

Box 2: Partial list of arguments of the BGLR function

```
BGLR( y, a = NULL, b = NULL, response_type = "gaussian",
      ETA = NULL,
      nIter = 1500, burnIn = 500, thin = 5,
      saveAt = "",
      S0 = NULL, df0 = 5, R2 = 0.5,...
     )
```

Return

The function BGLR returns a list with estimated posterior means and estimated posterior standard deviations and the arguments used to fit the model. Box 3, shows the structure of the object returned after fitting the intercept model of Box 1. The first element of the list (`y`) is the response vector used in the call to BGLR, `$whichNa` gives the position of the entries in `y` that were missing, these two elements are then followed by several entries describing the call (omitted in Box 3), and this is followed by estimated posterior means and estimated posterior standard deviations of the linear predictor (`$yHat` and `$SD.yHat`), the intercept (`$mu` and `$SD.mu`), and the residual variance (`$varE` and `$SD.varE`). Finally `$fit` gives a list with *DIC* and *DIC*-related statistics (Spiegelhalter *et al.* 2002).

Box 3: Structure of the object returned by BGLR (use after running the code in Box 1)

```
str(fm)
List of 18
 $ y            : num [1:100] 50.4 48.2 48.5 50.5 50.2 ...
 $ whichNa      : int(0)
  .
  .
  .
 $ yHat         : num [1:100] 49.7 49.7 49.7 49.7 49.7 ...
 $ SD.yHat      : num [1:100] 0.112 0.112 0.112 0.112 0.112 ...
 $ mu           : num 49.7
 $ SD.mu        : num 0.112
 $ varE         : num 1.11
 $ SD.varE      : num 0.152
 $ fit          :List of 4
  ..$ logLikAtPostMean: num -147
  ..$ postMeanLogLik  : num -148
  ..$ pD              : num 2.02
  ..$ DIC             : num 298
-attr(*, "class")= chr "BGLR"
```

Output files

Box 4, shows an example of the files generated after executing the commands given in Box 1. In this case samples of the intercept (`mu.dat`) and of the residual variance (`varE.dat`) were stored. These samples can be used to assess convergence and to estimate Monte Carlo error. The R-package coda (Plummer *et al.* 2006) provides several useful functions for the analysis of samples used in Monte Carlo algorithms.

Box 4: Files generated by BGLR (use after running the code in Box 1)

```
list.files()
[1] "mu.dat"   "varE.dat"
plot(scan("varE.dat"),type="o")
```

Data Sets

The BGLR package comes with two genomic data sets involving phenotypes, markers, pedigree, and other covariates.

Mice data set: This data set is from the Wellcome Trust (http://gscan.well.ox.ac.uk) and has been used for detection of quantitative trait loci (QTL) by Valdar *et al.* (2006a,b) and for whole-genome regression by Legarra *et al.* (2008), de los Campos *et al.* (2009b), and Okut *et al.* (2011). The data set consists of genotypes and phenotypes of 1814 mice. Several phenotypes are available in the data frame `mice.pheno`. Each mouse was genotyped at 10,346 SNPs. We removed SNPs with minor allele frequency (MAF) <0.05 and imputed missing genotypes with expected values computed with estimates of allele frequencies derived from the same data. In addition to this, an additive relationship matrix (`mice.A`) is provided; this was computed using the R-package `pedigreemm` (Bates and Vazquez 2009; Vazquez *et al.* 2010).

Wheat data set: This data set is from CIMMYT global wheat breeding program and comprises phenotypic, genotypic, and pedigree information of 599 wheat lines. The data set was made publicly available by Crossa *et al.* (2010). Lines were evaluated for grain yield (each entry corresponds to an average of two plot records) at four different environments; phenotypes (`wheat.Y`) were centered and standardized to a unit variance within environment. Each of the lines were genotyped for 1279 diversity array technology (DArT) markers. At each marker two homozygous genotypes were possible and these were coded as 0/1. Marker genotypes are given in the object `wheat.X`. Finally a matrix `wheat.A` provides the pedigree relationships between lines computed from the pedigree (see Crossa *et al.* 2010 for further details). Box 5, illustrates how to load the wheat and mice data sets.

Box 5: Loading the mice and wheat data sets included in BGLR

```
library(BGLR)
data(mice)
data(wheat)
ls()
```

Application Examples

In this section we illustrate the use of BGLR with various application examples, including comparison of shrinkage and variable selection methods (Example 1), how to fit models that account for genetic and nongenetic effects such as covariates or effects of the experimental design (Example 2), models for simultaneous regression on markers and pedigree (Example 3), reproducing kernel Hilbert spaces regression (Examples 4 and 5), and the assessment of prediction accuracy (Examples 6 and 7). The scripts used to fit the models discussed in each of these examples are presented in the text; additional scripts with code for post hoc analysis (*e.g.*, plots) are provided in the supporting information (File S1 and File S2).

Example 1: Comparison of shrinkage and variable selection methods

In this example we show how to fit models that induce variable selection and others that shrink estimates toward zero. In the example, we use simulated data generated using the marker genotypes for the mice data set. We assume a very simple simulation setting with only 10 QTL. Phenotypes were simulated under the standard additive model,

$$y_i = \sum_{j=1}^{p} x_{ij}\beta_j + \varepsilon_i,\ i = 1, \dots, n,$$

where $\varepsilon_i \sim N(0,\ 1 - h^2)$, $h^2 = 0.5$. Marker effects were sampled from the mixture model:

$$\beta_j = \begin{cases} N\left(0, h^2/10\right) & \text{if } j \in \{517, 1551, 2585, 3619, 4653, \\ & \qquad 5687, 6721, 7755, 8789, 9823\} \\ 0 & \text{otherwise.} \end{cases}$$

Box 6, shows the R code for simulating the phenotypes. The simulation settings can be changed using parameters that control sample size (`n`), the number of markers used (`p`), the number of QTL (`nQTL`), and trait heritability (`h2`).

Box 6: Shrinkage and variable selection (I: simulation)

```
rm(list=ls())
  library(BGLR);  set.seed(12345); data(mice);
  n<-nrow(mice.X); p<-ncol(mice.X);
  X<-scale(mice.X,scale=TRUE,center=TRUE)

## Toy simulation example
  nQTL<-10; p<-ncol(X); n<-nrow(X); h2<-0.5
  whichQTL<-seq(from=floor(p/nQTL/2),by=floor(p/nQTL),length=nQTL)
  b0<-rep(0,p)
  b0[whichQTL]<-rnorm(n=nQTL,sd=sqrt(h2/nQTL))
  signal<-as.vector(X%*%b0)
  error<-rnorm(n,sd=sqrt(1-h2))
  y<-signal+error
```

Box 7, shows code that can be used to fit a Bayesian ridge regression, BayesA, and BayesB. Once the models are fitted, estimates of marker effects, predictions, estimates of the residual variance, and measures of goodness of fit and model complexity can be extracted from the object returned by BGLR. Box S1 of File S1, provides the code used to extract the results presented next.

Box 7: Shrinkage and variable selection (II: fitting models)

```
  nIter<-4500; burnIn<-500
## Fitting models
## Bayesian Ridge Regression (Gaussian prior), equivalent to G-BLUP
 ETA<-list(MRK=list(X=X,model="BRR"))
 fmBRR<-BGLR(y=y,ETA=ETA, nIter=nIter, burnIn=burnIn,saveAt="BRR_")

## Bayes A(Scaled-t prior)
 ETA$MRK$model<-"BayesA"
 fmBA<-BGLR(y=y,ETA=ETA, nIter=nIter, burnIn=burnIn,saveAt="BA_")

## Bayes B (point of mass at zero + scaled-t slab)
 ETA$MRK$model<-"BayesB"
 fmBB<-BGLR(y=y,ETA=ETA, nIter=nIter, burnIn=burnIn,saveAt="BB_")
```

Table 3 provides the estimated residual variance, the deviance information criterion (*DIC*) and the effective number of parameters (*pD*) (Spiegelhalter *et al.* 2002). The estimated residual variances were all closed to the simulated value (0.5). According to *pD* (288.9, 200.2, 198.3 for the BRR, BayesA, and BayesB, respectively) the BRR was the most complex model, and *DIC* ("smaller is better") favored BayesA and BayesB over the BRR, clearly. This was expected given the simple trait architecture simulated. Model BayesA and BayesB gave very similar estimates and predictions; this happened because in BayesB the estimated probability for the markers to have nonnull effects was very high (>0.9); as this proportion approaches one, BayesB converges to BayesA. The correlation between the true and simulated signals were high in all cases (0.86 for the BRR, 0.947 for BayesA, and 0.955 for BayesB) but favored BayesA and BayesB over the BRR, clearly. We run the simulation using 30 QTL and removing the QTL genotypes in the data analysis and the ranking of the models, based on *DIC* and on the correlation between predicted and simulated signal was similar to the one reported above.

Figure 2 displays the absolute values of estimates of marker effects for models BayesA (red) and the BRR (black). The estimates of BayesB (not shown) are similar to those of BayesA. The vertical lines and blue dots give the position and absolute value of the simulated effects. The BRR gives a profile of estimated of effects where all markers had tiny effects; BayesA and BayesB give a very different profiles of effects: most of the simulated QTL were detected (except the first one), markers having no effects had very small estimated effects, and QTL had sizable estimated effects. This simulation illustrates how in ideal circumstances the choice of the prior density assigned to marker effects can make a big difference in terms of estimates of effects. However, the difference between models is expected to be much smaller under more complex genetic architectures and, perhaps more importantly, when the marker panel does not contain markers in tight LD with QTL, *e.g.*, Wimmer *et al.* (2013). The example also illustrates a very important concept: in high-dimensional regressions it is possible to have similar predictions with very different estimates of effects. Indeed, in the example presented above, although the correlation of effects estimated by BRR and BayesB was low (0.226), the correlation between predictions (`yHat`) derived from each of the models was relatively high (0.946).

Example 2: Fitting models for genetic and nongenetic factors

In the next example we illustrate how to fit models with various sets of predictors using the mice data set. Valdar *et al.* (2006b) pointed out that the cage where mice were housed had an important effect in the physiological covariates and Legarra *et al.* (2008) and de los Campos *et al.* (2009b) used models that accounted for sex, litter size, cage, familial relationships, and markers. Therefore, one possible linear model that we can fit to a continuous phenotype is

Table 3 Measures of Fit and of Model Complexity

Model	Residual variance	*DIC*	*pD*
Bayesian ridge regression	0.506 (0.020)	4200.0	288.9
BayesA	0.489 (0.019)	4047.4	200.2
BayesB	0.482 (0.019)	4017.6	198.3

$$\mathbf{y} = 1\mu + \mathbf{X}_1\boldsymbol{\beta}_1 + \mathbf{X}_2\boldsymbol{\beta}_2 + \mathbf{X}_3\boldsymbol{\beta}_3 + \boldsymbol{\varepsilon},$$

where $\mathbf{y}$ is the phenotype vector (body mass index, in the example), μ is an intercept, $\mathbf{X}_1$ is a design matrix for the effects of sex and litter size, $\boldsymbol{\beta}_1$ is the corresponding vector of effects, $\mathbf{X}_2$ is the design matrix for the effects of cage, $\boldsymbol{\beta}_2$ is a vector of cage effects, $\mathbf{X}_3$ is the matrix with marker genotypes, and $\boldsymbol{\beta}_3$ is the corresponding vector of marker effects. We treat $\boldsymbol{\beta}_1$ as "`FIXED`" and the other two vectors of effects as random; $\boldsymbol{\beta}_2$ is treated as Gaussian and marker effects, $\boldsymbol{\beta}_3$, are assigned IID double-exponential priors, which corresponds to the prior used in the Bayesian LASSO model.

Fitting the model: The script needed to fit the model above described is given in Box 8. The first block of code, `#1#`, loads the data. In the second block of code we set the linear predictor. This is specified using a two-level list. Each of the elements of the inner list is used to specify one element of the linear predictor; these elements are specified by providing a formula or a design matrix and a prior (model argument). When the formula is used, the design matrix is created internally using the `model.matrix()` function of R. Additional arguments in the specification of the linear predictors are optional (see Table S1, File S1 for the arguments used to specify hyperparameters). Finally in the third block of code we fit the model by calling the `BGLR()` function. When BGLR begins to run, a message warns the user that hyperparameters were not provided and that consequently they were set using built-in rules (see Table S1, File S1, for further details).

Box 8: Fitting a model to genetic and non-genetic effects using BGLR

```
#1# Loading and preparing the input data
 library(BGLR); data(mice);
 Y<-mice.pheno; X<-mice.X
 y<-Y$Obesity.BMI; y<-scale(y,center=TRUE,scale=TRUE)

#2# Setting the linear predictor
 ETA<-list(  FIXED=list(~factor(GENDER)+factor(Litter),
                   data=Y,model="FIXED"),
             CAGE=list(~factor(cage),data=Y, model="BRR"),
             MRK=list(X=X,  model="BL")
           )

#3# Fitting the model
  fm<-BGLR(y=y,ETA=ETA, nIter=105000, burnIn=5000)
  save(fm,file="fm.rda")
```

Extracting results: Once the model was fitted one can extract from the list returned by BGLR the estimated posterior means and the estimated posterior standard deviations as well as measures of model goodness of fit and model complexity. Also, as BGLR runs, it saves samples of some of the parameters; these samples can be brought into the R-environment for posterior analysis. Box S2 of File S1, illustrates how to extract estimates from the models fitted in Box 8. Some of the results are given in Figure 3. In the

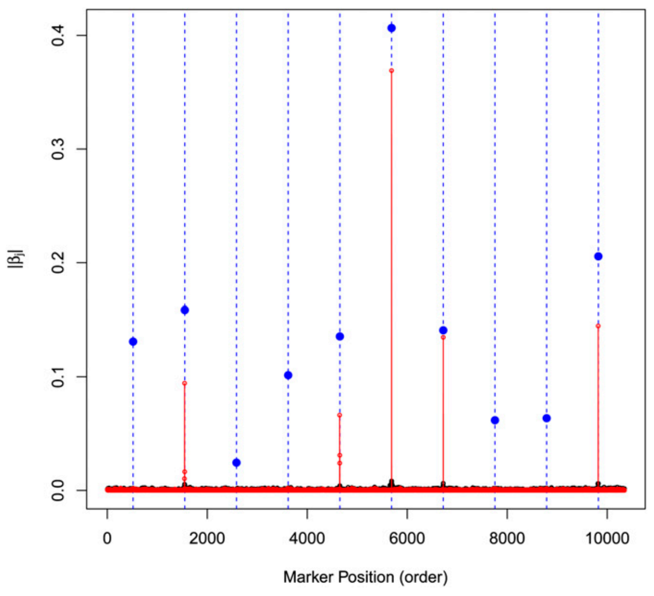

Figure 2 Absolute value of estimated marker effects (black, Bayesian ridge regression; red, BayesB; blue, simulated Value).

example, phenotypes were standardized to a unit sample variance and the estimated residual variance was 0.53, suggesting that the model explained ~47% of the phenotypic variance. Figure 3, top left, gives the absolute value of estimated effects and Figure 3, top right, gives a scatter plot of phenotypes *vs.* predicted genomic values; this prediction does not include differences due to sex, litter size, or cage. Figure 3, bottom, gives trace plots of the residual variance (left) and of the regularization parameter of the Bayesian LASSO (right). The residual variance had a very good mixing; however, the mixing of the regularization parameter was not as good. In general, with large numbers of markers long chains are needed to infer regularization parameters precisely.

Example 3: Fitting a pedigree+markers BLUP model using BGLR

In the following example we illustrate how to incorporate in the model Gaussian random effects with user-defined covariance structures. These types of random effects appear both in pedigree and genomic models. The example presented here uses the wheat data set included with the BGLR package. In the example of Box 9 we include two random effects, one representing a regression on pedigree, $\mathbf{a} \sim N(\mathbf{0},\ \mathbf{A}\sigma_a^2)$, where $\mathbf{A}$ is a pedigree-derived numerator relationship matrix, and one representing a linear regression on markers, $\mathbf{g} \sim N(\mathbf{0},\ \mathbf{G}\sigma_g^2)$ where $\mathbf{G}$ is a marker-derived genomic relationship matrix. For ease of interpretation of estimates of variance components we standardized both matrices to an average diagonal value of (approximately) one. The implementation of Gaussian processes in BGLR exploits the equivalence between these processes and random regressions on principal components (de los Campos *et al.* 2010; Janss *et al.* 2012). The user can implement a RKHS regression either by providing covariance matrix ($\mathbf{K}$) or its eigenvalue decomposition (see the example in Box 9). When the covariance matrix is provided, the eigenvalue decomposition is computed internally. Box S3 of File S1, shows how to extract estimates, predictions, and samples from the fitted model. The estimated residual variance (posterior standard deviation) was 0.43 (0.044), and the estimates of the variance components associated to the pedigree, σ_a^2, and markers, σ_g^2, were 0.24 (0.07) and 0.42 (0.09), respectively. The code in Box S3 of File S1, shows how to obtain samples of heritability from the samples collected before each of the variance components. The estimated posterior mean of the ratio of the genetic variance $(\sigma_a^2 + \sigma_g^2)$ relative to the total variance was 0.6; therefore, we conclude that ~60% of the phenotypic variance can be explained by genetic factors. In this example, the pedigree explained approximately one-third of the total genetic variance and markers explained the other two-thirds. The samples from the posterior distribution of σ_a^2 and σ_g^2 had a posterior correlation of -0.184; this happens because both $\mathbf{A}$ and $\mathbf{G}$ are, to some extent, redundant.

Box 9: Fitting a Pedigree + Markers regression using Gaussian Processes

```
#1# Loading and preparing the input data
 library(BGLR); data(wheat); set.seed(123);
 Y<-wheat.Y; X<-wheat.X; A<-wheat.A/mean(diag(wheat.A));
 y<-Y[,1]

#2# Computing the genomic relationship matrix
 X<-scale(X,center=TRUE,scale=TRUE)
 G<-tcrossprod(X)/ncol(X)

#3# Computing the eigen-value decomposition of G
 EVD<-eigen(G)

#3# Setting the linear predictor
 ETA<-list(PED=list(K=A, model="RKHS"),
           MRK=list(V=EVD$vectors,d=EVD$values, model="RKHS")
       )

#4# Fitting the model
  fm<-BGLR(y=y,ETA=ETA, nIter=12000, burnIn=2000,saveAt="PGBLUP_")
  save(fm,file="fmPG_BLUP.rda")
```

Reproducing kernel Hilbert spaces regressions: Reproducing Kernel Hilbert Spaces Regressions (RKHS) have been used for regression (*e.g.*, smoothing spline, Wahba 1990), spatial smoothing (*e.g.*, kriging, Cressie 1988), and classification problems (*e.g.*, support vector machine, Vapnik 1998). Gianola *et al.* (2006) proposed using this approach for genomic prediction and since then several methodological and applied articles have been published elsewhere (Gianola and de los Campos 2008; de los Campos *et al.* 2009a, 2010). In this section we illustrate how to implement RKHS using single- (Example 4) and multi- (Example 5) kernel methods.

Example 4: Single-kernel models

In RKHS the regression function is a linear combination of the basis function provided by the reproducing kernel (RK); therefore, the choice of the RK constitutes one of the central elements of model specification. The RK is a function that maps from pairs of points in input space (*i.e.*, pairs of individuals) into the real line and must be positive semidefinite. For instance, if the information set is given by vectors of marker genotypes the RK, $K(\mathbf{x}_i, \mathbf{x}_{i'})$ maps from pairs of vectors

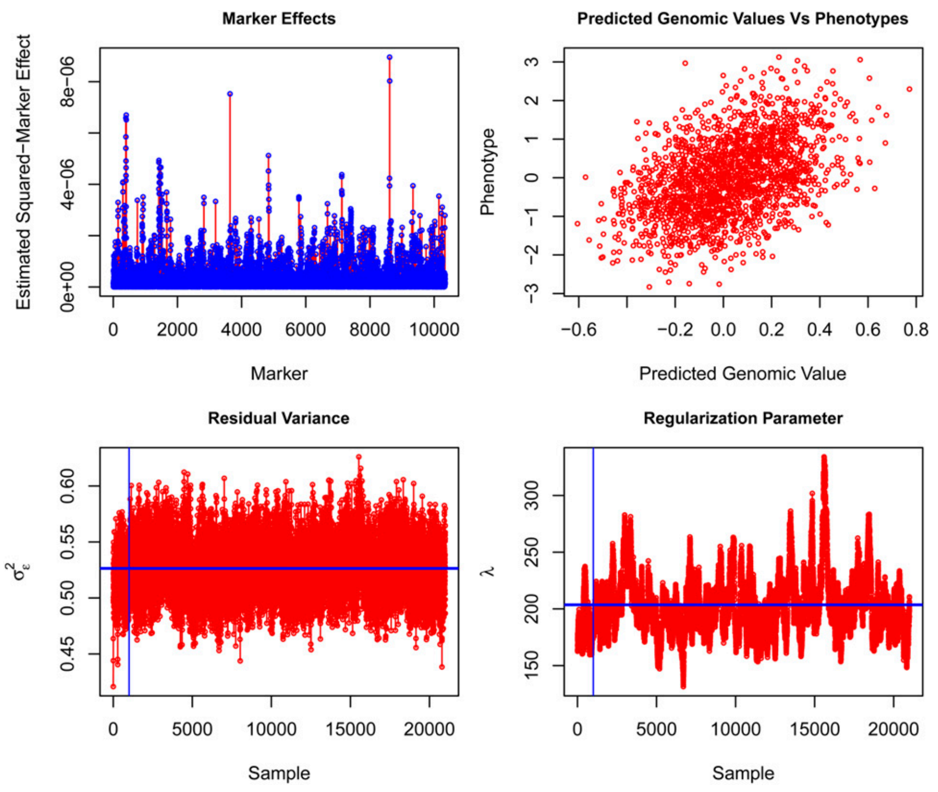

Figure 3 Squared-estimated marker effects (top left), phenotype *vs.* predicted genomic values (top right), trace plot of residual variance (bottom left), and trace plot of regularization parameter of the Bayesian LASSO (bottom right).

of genotypes, $\{\mathbf{x}_i, \mathbf{x}_{i'}\}$ onto the real line with a map that must satisfy $\sum_i \sum_{i'} \alpha_i \alpha_{i'} K(\mathbf{x}_i, \mathbf{x}_{i'}) \geq 0$, for any nonnull sequence of coefficients α_i. Following de los Campos *et al.* (2009a) the Bayesian RKHS regression can be represented as

$$\mathbf{y} = \mathbf{1}\mu + \mathbf{u} + \boldsymbol{\varepsilon} \quad \text{with } p(\mu, \mathbf{u}, \boldsymbol{\varepsilon}) \propto N\left(\mathbf{u}|\mathbf{0},\ \mathbf{K}\sigma^2_u\right) N\left(\boldsymbol{\varepsilon}|\mathbf{0},\ \mathbf{I}\sigma^2_\varepsilon\right), \tag{2}$$

where $\mathbf{K} = \{K(\mathbf{x}_i, \mathbf{x}_{i'})\}$ is an $(n \times n)$-matrix whose entries are the evaluations of the RK at pairs of points in input space. Note that the structure of the model described by (2) is that of the standard animal model (Quaas and Pollak 1980) with the pedigree-derived numerator relationship matrix ($\mathbf{A}$) replaced by the kernel matrix ($\mathbf{K}$). Box 10, features an example using a Gaussian kernel evaluated in the (average) squared-Euclidean distance between genotypes, that is:

$$K(\mathbf{x}_i, \mathbf{x}_{i'}) = \exp\left\{ -h \times \frac{\sum_{k=1}^{p} (x_{ik} - x_{i'k})^2}{p} \right\}.$$

In the example genotypes were centered and standardized, but this is not strictly needed. The bandwidth parameter, h, controls how fast the covariance function drops as the distance between pairs of vector genotypes increases. This parameter plays an important role in inferences and predictions. In this example we have arbitrarily chosen the bandwidth parameter to be equal to 0.25; further discussion about this parameter is given in the next example. With this choice of RK, the estimated residual variance was 0.41, which suggests that the RKHS model fitted in Box 10, fits the data slightly better than the pedigree + markers models of Box 9. Box S4 of File S1, provides supplementary code for the model fitted in Box 10.

Box 10: Fitting a Single Kernel Model in BGLR

```
#1# Loading and preparing the input data
 library(BGLR); data(wheat);  set.seed(123);
 Y<-wheat.Y; X<-wheat.X; n<-nrow(X); p<-ncol(X)
 y<-Y[,1]

#2#  Computing the distance matrix and then the kernel.
  X<-scale(X,center=TRUE,scale=TRUE)
  D<-(as.matrix(dist(X,method="euclidean"))^2)/p
  h<-0.25
  K<-exp(-h*D)

#3#  Single Kernel Regression using BGLR

 ETA<-list(K1=list(K=K,model="RKHS"))
 fm<-BGLR(y=y,ETA=ETA,nIter=12000, burnIn=2000,saveAt="RKHS_h=0.25_")
```

Example 5: Multikernel methods

The bandwidth parameter of the Gaussian kernel can be chosen using either cross-validation (CV) or Bayesian methods. From a Bayesian perspective, one possibility is to treat h as random; however, this is computationally demanding because the RK needs to be recomputed any time h is updated. To overcome this problem de los Campos *et al.* (2010) proposed using a multikernel approach (named kernel averaging, KA) consisting of: (a) defining a sequence of kernels based on a set of values of h, and (b) fitting a multikernel model with as many random effects as kernels in the sequence. The model has the form

$$\mathbf{y} = \mathbf{1}\mu + \sum_{l=1}^{L} \mathbf{u}_l + \boldsymbol{\varepsilon} \quad \text{with}$$
$$p(\mu, \mathbf{u}_1, \ldots, \mathbf{u}_L, \boldsymbol{\varepsilon}) \propto \prod_{l=1}^{L} N\left(\mathbf{u}_l|\mathbf{0},\ \mathbf{K}_l\sigma^2_{u_l}\right) N\left(\boldsymbol{\varepsilon}|\mathbf{0}, \mathbf{I}\sigma^2_\varepsilon\right), \tag{3}$$

where $\mathbf{K}_l$ is the RK evaluated at the lth value of the bandwidth parameter in the sequence $\{h_1, \ldots, h_L\}$. It can be shown (*e.g.*, de los Campos *et al.* 2010) that if variance components are known, the model of expression (3) is equivalent to a model with a single random effect whose distribution is $N(\mathbf{u}|\mathbf{0},\ \bar{\mathbf{K}}\sigma_u^2)$, where $\bar{\mathbf{K}}$ is a weighted average of all the RK used in (3) with weights proportional to the corresponding variance components (hence the name kernel averaging). Performing a grid search for h or implementing a multikernel model requires defining a reasonable range for h. If the value of h is too small, the entries of the resulting Gaussian kernel will approach a matrix full of ones; such a kernel will be redundant with the intercept, which is included as fixed effect. On the other hand, if h is too large, the off-diagonal values of the kernel matrix will approach zero, leading to a random effect that is confounded with the error term. Therefore, in choosing values for h both extremes should be avoided. Although there is no general rule to define the values of the bandwidth parameter, one possibility is to set h to values $h = 1/M \times \{1/5, 1, 5\}$, where M is the median squared Euclidean distance between lines (computed using off-diagonals only). With this choice, the median off-diagonals of $\mathbf{K}$ will be $\exp\left(-\frac{1}{5}\right)$, $\exp(-1)$ and $\exp(-5)$ for h equal to $h = \frac{1}{5} \times \frac{1}{M}$, $h = \frac{1}{M}$ and $h = \frac{5}{M}$, respectively. We use this approach in Box 11, to fit a multikernel model. The resulting entries of the RK are displayed in Figure 4. Box S5 of File S1, provides supplementary code that can be used to retrieve estimates from the model fitted in Box 11. The estimated residual variance was close to 0.3 and the estimated variance components for each of the kernels fitted were 0.62, 0.48, and 0.24 for h equal to 0.098, 0.490 and 2.450, respectively. The script provided in Box S5 of File S1, produces trace plots of variance components. The residual variance has a reasonably good mixing. The sum of the variances of the three kernels also has a reasonably good mixing; however, due to confounding between the kernels, individual variance components show a much poorer mixing.

Box 11: Fitting a RKHS Using a Multi-Kernel Methods (Kernel Averaging)

```
#1# Loading and preparing the input data
  library(BGLR); data(wheat);  set.seed(456);
  Y<-wheat.Y; X<-wheat.X; n<-nrow(X); p<-ncol(X)
  y<-Y[,1]

 #2#  Computing D and then K
  X<-scale(X,center=TRUE,scale=TRUE)
  D<-(as.matrix(dist(X,method="euclidean"))^2)/p
  h<-round(1/median(D[row(D)>col(D)]),2)
  h<-h*c(1/5,1,5)

  K1=exp(-h[1]*D)
  K2=exp(-h[2]*D)
  K3=exp(-h[3]*D)

 #3#  Kernel Averaging using BGLR
 ETA<-list(list(K=K1,model="RKHS"),
           list(K=K2,model="RKHS"),
           list(K=K3,model="RKHS"))
 fm<-BGLR(y=y,ETA=ETA,nIter=17000, burnIn=2000,saveAt="RKHS_KA_")
```

Assessment of prediction accuracy: In the previous examples we illustrated how to fit different types of models to training data; in the following section we consider two ways of assessing prediction accuracy: a single training–testing partition and multiple training–testing partitions.

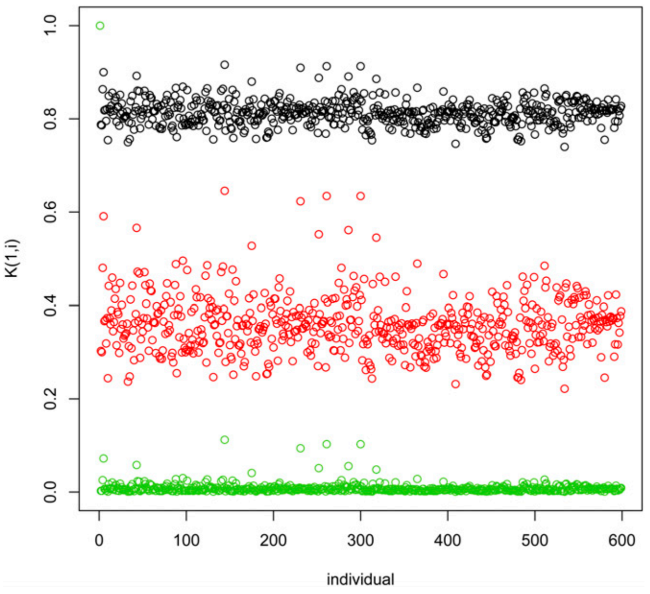

Figure 4 Entries of the first row of the (Gaussian) kernel matrix evaluated at three different values of the bandwidth parameter, h = 0.098, 0.490, 2.450.

Example 6: Assessment of prediction accuracy using a single training-testing partition

A simple way of assessing prediction accuracy consists of partitioning the data set into two disjoint sets: one used for model training (TRN) and one used for testing (TST). Box 12, shows code that fits a G-BLUP model in a TRN–TST setting using the wheat data set. The code randomly assigns 100 individuals to the TST set. The variable `tst` is a vector that indicates which data points belong to the TST data set; for these entries we introduce missing values in the phenotypic vector (see Box 12). Once the model is fitted, predictions for individuals in the TST set can be obtained typing `fm$yHat[ tst]` in the R command line; Box S6 of File S1 gives supplementary code to the example in Box 12, including the code used to produce Figure 5 that displays observed *vs.* predicted phenotypes for individuals in TRN (black dots) and TST (red dots) sets. The correlation between observed phenotypes and predictions was 0.83 in the TRN set and 0.60 in the TST set, and the regression of phenotypes on predictions was 1.49 and 1.24 for the TRN and TST set, respectively.

Box 12: Assessment of Prediction Accuracy (Continuous Response)

```
#1# Loading and preparing the input data
  library(BGLR); data(wheat);
  Y<-wheat.Y; X<-wheat.X; n<-nrow(X); p<-ncol(X)
  y<-Y[,1]
#2# Creating a Testing set
  yNA<-y
  set.seed(123)
  tst<-sample(1:n,size=100,replace=FALSE)
  yNA[tst]<-NA
#3#  Computing G
  X<-scale(X,center=TRUE,scale=TRUE)
  G<-tcrossprod(X)/p

#4#  Fits the G-BLUP model
  ETA<-list(list(K=G,model="RKHS"))
  fm<-BGLR(y=yNA,ETA=ETA,nIter=5000, burnIn=1000,saveAt="RKHS_")
```

Example 7: Model comparison based on multiple training–testing partitions

The example presented in the previous section is based on a single training–testing partition. A cross-validation is simply a generalization of the single TRN–TST evaluation. For a K-fold cross-validation there are K TRN–TST partitions; in each fold, the individuals assigned to that particular fold are used for TST and the remaining individuals are used for TRN. Another possibility is to generate multiple TRN–TST partitions with random assignment of subjects to either TRN or TST. Each partition yields a point estimate of prediction accuracy (*e.g.*, correlation between predictions and phenotypes). The variability of the point estimate across partitions (replicates) reflects uncertainty due to sampling of TRN and TST sets, and a precise estimate of prediction accuracy can be obtained by averaging the estimates of accuracy obtained in each partition.

Box 13, gives an example of an evaluation based on 100 TRN–TST partitions. In each partition two models (P, pedigree and PM, pedigree + markers) were fitted and used to predict yield in the TST data set. This yielded 100 estimates of the prediction correlation for each of the models fitted. These estimates should be regarded as paired samples because both share a common feature: the TRN–TST partition. Several statistics can be computed to compare the two models fitted, and a natural approach for testing the null hypotheses H_0: P and PM have the same prediction accuracy *vs.* H_A: the prediction accuracy of models P and PM are different is to conduct a paired-t-test based on the difference of the correlation coefficients. Figure 6 gives the estimated correlations for the pedigree + markers model (PM, vertical axis) *vs.* the pedigree-only model (P, horizontal axis) by environment. The great majority of the points lay above the 45° line indicating that in most partitions the PM model had higher prediction accuracy than the P-only model. The paired-t-test had P-values <0.001 in all environments indicating strong evidence against the null hypothesis (H_0: P and PM have the same prediction accuracy). The code used to generate the plot in Figure 6 and to carry out the t-test is given in Box S7, File S1.

Box 13: Replicated Training-Testing Partitions

```
#1# Loading and preparing the input data
 rm(list=ls()); trait<-1; library(BGLR); data(wheat); y<-wheat.Y[,trait]
 set.seed(123);
 nTST<-150; n<-length(y); nRep<-100; nIter<-12000; burnIn<-2000

#2# Computing the genomic relationship matrix and the EVD decomposition
X<-scale(wheat.X,center=TRUE,scale=TRUE)
G<-tcrossprod(X)/ncol(X)
EVDG<-eigen(G); EVDA<-eigen(wheat.A)

#3# Setting the linear predictor
 H0<-list(PED=list(V=EVDA$vectors,d=EVDA$values, model="RKHS"))
 HA<-H0
 HA$G<-list(V=EVDG$vectors,d=EVDG$values, model="RKHS")
 COR<-matrix(nrow=nRep,ncol=2,NA)

#4# Loop over TRN-TST partitions
   for(i in 1:nRep){
     tst<-sample(1:n,size=nTST,replace=FALSE)
     yNA<-y; yNA[tst]<-NA
     fm<-BGLR(y=yNA,ETA=H0, nIter=nIter, burnIn=burnIn)
     COR[i,1]<-cor(y[tst],fm$yHat[tst]); rm(fm)
     fm<-BGLR(y=yNA,ETA=HA, nIter=nIter, burnIn=burnIn)
     COR[i,2]<-cor(y[tst],fm$yHat[tst]); rm(fm)
}
```

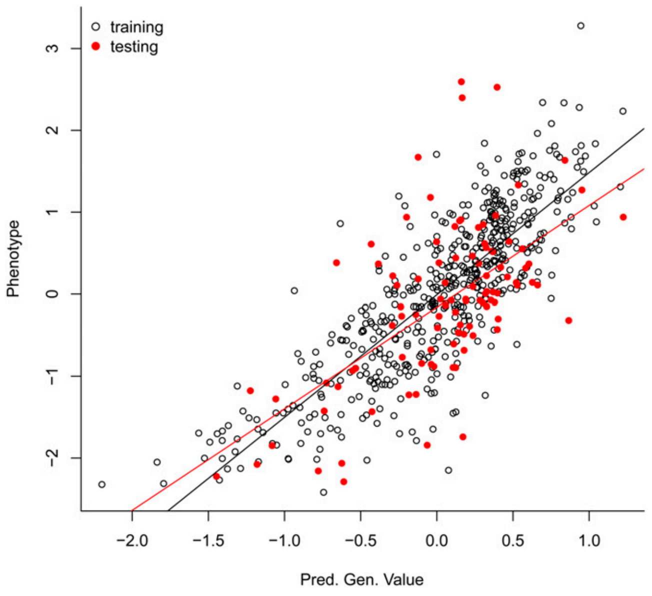

Figure 5 Estimated genetic values for training and testing sets. Predictions were derived using G-BLUP model (see Box 12).

Benchmark of Parametric Models

We carried out a benchmark evaluation by fitting a BRR to data sets involving three different sample size (n = 1K, 2K and 5K, K = 1000) and four different marker densities (P = 5K, 10K, 50K, and 100K). The evaluation was carried out in an Intel Xeon processor, 2 GHz, with R executed in a single thread and linked against OpenBLAS. Computing time (Figure 7) scales approximately proportional to the product of the number of records and the number of effects. For the most demanding scenario (n = 5K, P = 100K) it took ~11 min to complete 1000 iterations of the Gibbs sampler. Using the R functions `Rprof` and `summaryRprof` we performed an analysis of memory usage. As expected the amount of memory used scaled linearly with marker density with a maximum memory usage of ~6, 3, 0.6, and 0.3 Gb of RAM for n =5K and P = 100K, 50K, 10K, and 5K, respectively. Because R holds all the objects in virtual memory and the size of the objects depends on the underlying operating system, as a general rule, for an R session using more than 4 Gb of RAM, a 64-bit build of R would be needed.

Concluding Remarks

In BGLR we implemented in a unified Bayesian framework several methods commonly used in genome-enabled prediction, including various parametric models and Gaussian processes that can be used for parametric (*e.g.*, pedigree regressions or G-BLUP) or semiparametric regressions (*e.g.*, genomic regressions). The package supports continuous (censored or not) as well as binary and ordinal traits. The software interface gives the user great latitude in combining different modeling approaches for data analysis. In

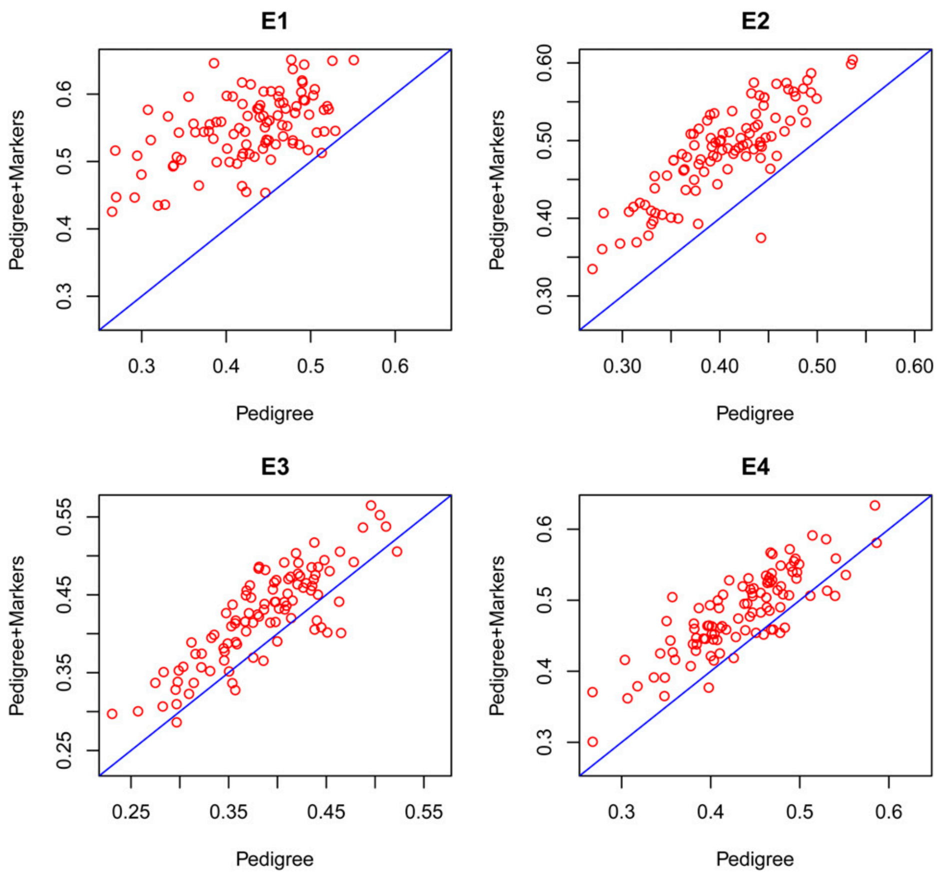

Figure 6 Estimated correlations between phenotypes and predictions in testing data sets (for a total of 100 training–testing partitions) by model (pedigree in the horizontal axis and pedigree + markers in the vertical axis) by environment (E1–E4).

the algorithm implemented in BGLR, operations that can be vectorized are performed using built-in R-functions; however, most of the computing intensive tasks are performed using compiled routines written in C and Fortran languages. The package is also able to take advantage of multithread BLAS implementations in both Windows and UNIX-like systems. Finally, together with the package we have included two data sets and ancillary functions that can be used to read into

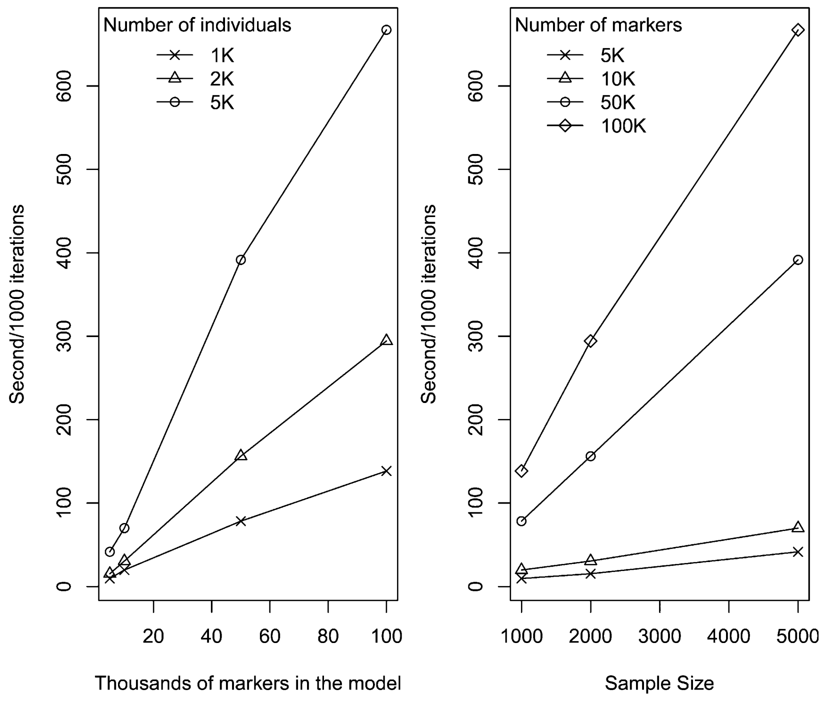

Figure 7 Seconds per 1000 iterations of the Gibbs sampler by number of markers and sample size. The benchmark was carried out by fitting a Gaussian regression (BRR) using an Intel Xeon processor, 2 GHz. Computations were carried out using a single thread.

the R-environment genotype files written in ped and bed formats (Purcell *et al.* 2007).

The Gibbs sampler implemented is computationally very intensive and our current implementation stores genotypes in memory; therefore, despite the effort made to make BGLR computationally efficient, performing regressions with hundreds of thousands of markers requires access to large amounts of RAM and the computational time can be considerable. Certainly, algorithms other than MCMC can be quite faster; however, the MCMC framework adopted offers great flexibility, as illustrated by the examples presented here.

Although some of the computationally intensive algorithms implemented in BGLR can benefit from multithread computing; there is room to further improve the computational performance of the software by making more intensive use of parallel computing. In future releases we plan to exploit parallel computing to a much greater extent. Also, we are currently working on modifying the software in ways that avoid loading genotypes in in memory. Future releases will be made at the R-Forge website (https://r-forge.r-project.org/R/?group_id=1525) first and, after considerable testing, at CRAN.

Acknowledgments

In the development of BGLR Paulino Pérez and Gustavo de los Campos received financial support provided by National Institutes of Health Grants R01GM099992 and R01GM101219. The authors thank Professor Daniel Gianola for comments made in a draft of the manuscript.

Literature Cited

Albert, J. H., and S. Chib, 1993 Bayesian analysis of binary and polychotomous response data. J. Am. Stat. Assoc. 88(422): 669–679.

Andrews, D. F., and C. L. Mallows, 1974 Scale mixtures of normal distributions. J. R. Stat. Soc., B 36(1): 99–102.

Bates, D., and A. I. Vazquez, 2009 pedigreemm: pedigree-based mixed-effects models, R package version 0.2–4.

Bellman, R. E., 1961 *Adaptive Control Processes: A Guided Tour.* Princeton University Press, Princeton, NJ.

Casella, G., and E. I. George, 1992 Explaining the gibbs sampler. Am. Stat. 46(3): 167–174.

Cressie, N., 1988 Spatial prediction and ordinary kriging. Math. Geol. 20(4): 405–421.

Crossa, J., G. de los Campos, P. Pérez, D. Gianola, J. Burgueno *et al.*, 2010 Prediction of genetic values of quantitative traits in plant breeding using pedigree and molecular markers. Genetics 186 : 713–724.

de los Campos, G., D. Gianola, and G. J. M. Rosa, 2009a Reproducing kernel Hilbert spaces regression: a general framework for genetic evaluation. J. Anim. Sci. 87(6): 1883–1887.

de los Campos, G., H. Naya, D. Gianola, J. Crossa, A. Legarra *et al.*, 2009b Predicting quantitative traits with regression models for dense molecular markers and pedigree. Genetics 182: 375–385.

de los Campos, G., D. Gianola, G. J. M. Rosa, K. A. Weigel, and J. Crossa, 2010 Semi-parametric genomic-enabled prediction of genetic values using reproducing kernel Hilbert spaces methods. Genet. Res. 92: 295–308.

de los Campos, G., J. M. Hickey, R. Pong-Wong, H. D. Daetwyler, and M. P. L. Calus, 2013a Whole genome regression and prediction methods applied to plant and animal breeding. Genetics 193: 327–345.

de los Campos, G., A. I. Vazquez, and R. L. Fernando, K. Y. C., and S. Daniel, 2013b Prediction of complex human traits using the genomic best linear unbiased predictor. PLoS Genet. 7(7): e1003608.

Endelman, J. B., 2011 Ridge regression and other kernels for genomic selection with r package rrblup. Plant Genome 4: 250–255.

Geman, S., and D. Geman, 1984 Stochastic relaxation, gibbs distributions and the Bayesian restoration of images. IEEE Trans. Pattern Anal. Mach. Intell. 6(6): 721–741.

Gianola, D., 2013 Priors in whole-genome regression: the Bayesian alphabet returns. Genetics 90: 525–540.

Gianola, D., and G. de los Campos, 2008 Inferring genetic values for quantitative traits non-parametrically. Genet. Res. 90: 525–540.

Gianola, D., R. L. Fernando, and A. Stella, 2006 Genomic-assisted prediction of genetic value with semiparametric procedures. Genetics 173: 1761–1776.

Habier, D., R. Fernando, K. Kizilkaya, and D. Garrick, 2011 Extension of the Bayesian alphabet for genomic selection. BMC Bioinformatics 12(1): 186.

Hayes, B., P. Bowman, A. Chamberlain, and M. Goddard, 2009 Invited review: genomic selection in dairy cattle: progress and challenges. J. Dairy Sci. 92(2): 433–443.

Henderson, C. R., 1975 Best linear unbiased estimation and prediction under a selection model. Biometrics 31(2): 423–447.

Hoerl, A. E., and R. W. Kennard, 1970 Ridge regression: biased estimation for nonorthogonal problems. Technometrics 42(1): 80–86.

Janss, L., G. de los Campos, N. Sheehan, and D. Sorensen, 2012 Inferences from genomic models in stratified populations. Genetics 192: 693–704.

Legarra, A., C. Robert-Granié, E. Manfredi, and J.-M. Elsen, 2008 Performance of genomic selection in mice. Genetics 180: 611–618.

Makowsky, R., N. M. Pajewski, Y. C. Klimentidis, A. I. Vazquez, C. W. Duarte *et al.*, 2011 Beyond missing heritability: prediction of complex traits. PLoS Genet. 7(4): e1002051.

Meuwissen, T. H. E., B. J. Hayes, and M. E. Goddard, 2001 Prediction of total genetic value using genome-wide dense marker maps. Genetics 157: 1819–1829.

Okut, H., D. Gianola, G. J. M. Rosa, and K. A. Weigel, 2011 Prediction of body mass index in mice using dense molecular markers and a regularized neural network. Genet. Res. 93: 189–201.

Park, T., and G. Casella, 2008 The Bayesian LASSO. J. Am. Stat. Assoc. 103(482): 681–686.

Pérez, P., G. de los Campos, J. Crossa, and D. Gianola, 2010 Genomic-enabled prediction based on molecular markers and pedigree using the Bayesian Linear Regression package in R. Plant Genome 3(2): 106–116.

Plummer, M., N. Best, K. Cowles, and K. Vines, 2006 Coda: convergence diagnosis and output analysis for mcmc. R News 6(1): 7–11.

Purcell, S., B. Neale, K. Todd-Brown, L. Thomas, M. A. R. Ferreira *et al.*, 2007 Plink: a tool set for whole-genome association and population-based linkage analyses. Am. J. Hum. Genet. 81: 559–575.

Quaas, R. L., and E. J. Pollak, 1980 Mixed model methodology for farm and ranch beef cattle testing programs. J. Anim. Sci. 51(6): 1277–1287.

R Core Team, 2014 *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Spiegelhalter, D. J., N. G. Best, B. P. Carlin, and A. Van Der Linde, 2002 Bayesian measures of model complexity and fit. J. R. Stat. Soc. Series B Stat. Methodol. 64(4): 583–639.

Tanner, M. A., and W. H. Wong, 1987 The calculation of posterior distributions by data augmentation. J. Am. Stat. Assoc. 82(398): 528–540.

Valdar, W., L. C. Solberg, D. Gauguier, S. Burnett, P. Klenerman *et al.*, 2006a Genome-wide genetic association of complex traits in heterogeneous stock mice. Nat. Genet. 38: 879–887.

Valdar, W., L. C. Solberg, D. Gauguier, W. O. Cookson, J. N. P. Rawlins *et al.*, 2006b Genetic and environmental effects on complex traits in mice. Genetics 174: 959–984.

VanRaden, P. M., 2008 Efficient methods to compute genomic predictions. J. Dairy Sci. 91(11): 4414–4423.

VanRaden, P., C. V. Tassell, G. Wiggans, T. Sonstegard, R. Schnabel *et al.*, 2009 Invited review: reliability of genomic predictions for north american holstein bulls. J. Dairy Sci. 92(1): 16–24.

Vapnik, V., 1998 *Statistical Learning Theory*, Ed. 1. Wiley, New York.

Vazquez, A. I., D. M. Bates, G. J. M. Rosa, D. Gianola, and K. A. Weigel, 2010 Technical note: an R package for fitting generalized linear mixed models in animal breeding. J. Anim. Sci. 88(2): 497–504.

Vazquez, A. I., G. de los Campos, Y. C. Klimentidis, G. J. M. Rosa, D. Gianola *et al.*, 2012 A comprehensive genetic approach for improving prediction of skin cancer risk in humans. Genetics 192: 1493–1502.

Wahba, G., 1990 *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics, Philadelphia.

Wimmer, V., T. Albrecht, H.-J. Auinger, and C.-C. Schoen, 2012 synbreed: a framework for the analysis of genomic prediction data using R. Bioinformatics 28(15): 2086–2087.

Wimmer, V., C. Lehermeier, T. Albrecht, H.-J. Auinger, Y. Wang *et al.*, 2013 Genome-wide prediction of traits with different genetic architecture through efficient variable selection. Genetics 195: 573–587.

Yang, J., B. Benyamin, B. P. McEvoy, S. Gordon, A. K. Henders *et al.*, 2010 Common snps explain a large proportion of the heritability for human height. Nat. Genet. 42(7): 565–569.

Zhou, X., and M. Stephens, 2012 Genome-wide efficient mixed-model analysis for association studies. Nat. Genet. 44(7): 821–824.

Communicating editor: S. Sen

GENETICS

Supporting Information
http://www.genetics.org/lookup/suppl/doi:10.1534/genetics.114.164442/-/DC1

Genome-Wide Regression and Prediction with the BGLR Statistical Package

Paulino Pérez and Gustavo de los Campos

DOI: 10.1534/genetics.114.164442

File S1

Supplementary Materials

A Prior Densities Used in the BGLR R-Package

In this section we describe the prior distributions assigned to the location parameters, $(\boldsymbol{\beta}_j, \boldsymbol{u}_l)$, entering in the linear predictor of eq. (1). For each of the unknown effects included in the linear predictor, $\{\boldsymbol{\beta}_1, .., \boldsymbol{\beta}_J, \boldsymbol{u}_1, ..., \boldsymbol{u}_L\}$, the prior density assigned is specified via the argument `model` in the corresponding entry of the list (see Box 8 for an example). Table S1 describes, for each of the options implemented, the prior density used. A brief description is given below.

FIXED. In this case regression coefficients are assigned flat priors, specifically we use a Gaussian prior with mean zero and variance equal to 1×10^{10}.

BRR. When this option is used regression coefficients are assigned normal IID normal distributions, with mean zero and variance σ^2_β. In a 2nd level of the hierarchy, the variance parameter is assigned a scaled-inverse Chi-squared density, with parameters df_β and S_β. This density is parameterized in a way that the prior expected value and mode are $E(\sigma^2_\beta) = \frac{S_\beta}{df_\beta - 2}$ and $Mode(\sigma^2_\beta) = \frac{S_\beta}{df_\beta + 2}$, respectively. By default, if df_β and S_β are not provided, BGLR sets $df_\beta = 5$ and solves for the scale parameter to match the R-squared of the model (see default rules to set hyper-parameters below). An analysis with fixed variance parameter can be obtained by choosing the degree of freedom parameter to a very large value (e.g., 1×10^{10}) and solving for the scale using $S_\beta = \sigma^2_\beta \times (df_\beta + 2)$; this gives a prior that collapses to a point of mass at σ^2_β.

BayesA. In this model the marginal distribution of marker effects is a scaled-t density, with parameters df_β and S_β. For computational convenience this density is implemented as an infinite mixture of scaled-normal densities. In a first level of the hierarchy marker effects are assigned normal densities with zero mean and marker-specific variance parameters, $\sigma^2_{\beta_{jk}}$. In a 2nd level of the hierarchy these variance parameters are assigned IID scaled-inverse Chi-squared densities with degree of freedom and scale parameters df_β and S_β, respectively. The degree of freedom parameter is regarded as known; if the user does not provide a value for this parameter BGLR sets $df_\beta = 5$. The scale parameter is treated as unknown, and BGLR assigns to this parameter a gamma density with rate and shape parameters r and s, respectively. The mode and coefficient of variation (CV) of the gamma density are $Mode(S_\beta) = (s-1)/r$ (for $s > 1$) and $CV(S_0) = 1/\sqrt{s}$. If the user does not provide shape and rate parameters BGLR sets $s = 1.1$, this gives a relatively un-informative prior with a CV of approximately 95%, and then solves for the rate so that the total contribution of the linear predictor matches the R-squared of the model (see default rules to set hyper-parameters, below). If one wants to run the analysis with fixed scale one can choose a very large value for the shape parameter (e.g., 1×10^{10}) and then solve for the rate so that the prior mode matches the desired value of the scale parameter using $r = (s-1)/S_\beta$.

Bayesian LASSO (BL). In this model the marginal distribution of marker effects is double-exponential. Following Park and Casella (2008) we implement the double-exponential density as a mixture of scaled normal densities. In the first level of the hierarchy, marker effects are assigned independent normal densities with null mean and maker-specific variance parameter $\tau^2_{jk} \times \sigma^2_\varepsilon$. The residual variance is assigned a scaled-inverse Chi-square density, and the marker-specific scale parameters, τ_{jk}^2, are assigned IID exponential densities with rate parameter $\lambda^2/2$. Finally, in the last level of the hierarchy λ^2 is either regarded as fixed (this is obtained by setting in the linear predictor the option `type="FIXED"`), or assigned either a Gamma ($\lambda^2 \sim Gamma(r, s)$ if `type="gamma"`) or a $\lambda/\max$ is assigned a Beta prior, if `type="beta"`, here max is a user-defined parameter representing the maximum value that λ can take). If nothing is specified, BGLR sets `type="gamma"` and $s = 1.1$, and solves for the scale parameter to match the expected R-squared of the model (see section B of this Supplementary Materials for further details).

BayesB-C. In these models marker effects are assigned IID priors that are mixtures of a point of mass at zero and a slab that is either normal (BayesC) or a scaled-t density (BayesB). The slab is structured as either in the BRR (this is the case of BayesC) or as in BayesA (this is the case of BayesB). Therefore, BayesB and BayesC extend BayesA and BRR, respectively, by introducing an additional parameter π which in the case of BGLR represents the prior proportion of non-zero effects. This parameter is treated as unknown and it is assigned a Beta prior $\pi \sim Beta(p_0, \pi_0)$, with $p_0 > 0$ and $\pi_0 \in [0, 1]$. The beta prior is parameterized in a way that the expected value by $E(\pi) = \pi_0$; on the other hand p_0 can be interpreted as the number of prior counts (priors "successes" plus prior "failures"); with this parametrization the variance of the Beta distribution is then given by $Var(\pi) = \frac{\pi_0(1-\pi_0)}{(p_0+1)}$, which is inversely proportional to p_0. Choosing $p_0 = 2$ and $\pi_0 = 0.5$ gives a uniform prior in the interval $[0, 1]$. Choosing a very large value for p_0 gives a prior that collapses to a point of mass at π_0.

Table S1. Prior densities implemented in BGLR.

`model=`	**Join distribution of effects and hyper-parameters**	**Specification of elements in the linear predictor**
`FIXED`	$p(\boldsymbol{\beta}_j) \propto 1$	`list(X=, model="FIXED")`
`BRR`	$p(\boldsymbol{\beta}_j,\sigma^2_\beta)=\left\{\prod_k N(\beta_{jk}\mid 0,\sigma^2_\beta)\right\}\chi^{-2}(\sigma^2_\beta\mid df_\beta,S_\beta)$	`list(X=, model="BRR",df0=,S0=,R2=)`
`BayesA`	$p(\boldsymbol{\beta}_j,\sigma^2_{\beta_j},S_\beta)=\left\{\prod_k N(\beta_{jk}\mid 0,\sigma^2_{\beta_{jk}})\chi^{-2}(\sigma^2_{\beta_{jk}}\mid df_\beta,S_\beta)\right\}G(S_\beta\mid r,s)$	`list(X=, model="BayesA",df0=,rate0=, shape0=,R2=)`
`BL`	$p(\boldsymbol{\beta}_j,\boldsymbol{\tau}^2_j,\lambda^2\mid\sigma^2_\varepsilon)=\left\{\prod_k N(\beta_{jk}\mid 0,\tau^2_{jk}\times\sigma^2_\varepsilon)Exp\left\{\tau^2_{jk}\mid\frac{\lambda^2}{2}\right\}\right\}\times G(\lambda^2\mid r,s)$, or	`list(X=,model="BL",lambda=,type="gamma", rate=,shape=,R2=)`[1]
	$p(\boldsymbol{\beta}_j,\boldsymbol{\tau}^2_j,\lambda\mid\sigma^2_\varepsilon,\texttt{max})=\left\{\prod_k N(\beta_{jk}\mid 0,\tau^2_{jk}\times\sigma^2_\varepsilon)Exp\left\{\tau^2_{jk}\mid\frac{\lambda^2}{2}\right\}\right\}\times B(\lambda/\max\mid p_0,\pi_0)$, or	`list(X=,model="BL",lambda=,type="beta", probIn=,counts=,max=,R2=)`[1]
	$p(\boldsymbol{\beta}_j,\boldsymbol{\tau}^2_j\mid\sigma^2_\varepsilon,\lambda)=\left\{\prod_k N(\beta_{jk}\mid 0,\tau^2_{jk}\times\sigma^2_\varepsilon)Exp\left\{\tau^2_{jk}\mid\frac{\lambda^2}{2}\right\}\right\}$	`list(X=,model="BL",lambda=,type="FIXED")`[1]
`BayesC`	$p(\boldsymbol{\beta}_j,\sigma^2_\beta,\pi) = \left\{\prod_k\left[\pi N(\beta_{jk}\mid 0,\sigma^2_\beta)+(1-\pi)1(\beta_{jk}=0)\right]\right\} \times\chi^{-2}(\sigma^2_\beta\mid df_\beta,S_\beta)B(\pi\mid p_0,\pi_0)$	`list(X=,model="BayesC",df0,S0, probIn=,counts=,R2=)`[2]
`BayesB`	$p(\boldsymbol{\beta}_j,\sigma^2_\beta,\pi) = \left\{\prod_k\left[\pi N(\beta_{jk}\mid 0,\sigma^2_\beta)+(1-\pi)1(\beta_{jk}=0)\right]\chi^{-2}(\sigma^2_{\beta_{jk}}\mid df_\beta,S_\beta)\right\} B(\pi\mid p_0,\pi_0)\times G(S_\beta\mid r,s)$	`list(X=,model="BayesB",df0,rate0,shape0, probIn=,counts=,R2=)`[2]
`RKHS`	$p(\boldsymbol{u}_l,\sigma^2_{u_l})=N(\boldsymbol{u}_l\mid\boldsymbol{0},\boldsymbol{K}_l\times\sigma^2_{u_l})\chi^{-2}(\sigma^2_{u_l}\mid df_l,S_l)$	Either `list(K=,model="RKHS",df0,S0,R2=)` or `list(V=,d=,model="RKHS",df0,S0,R2=)`[3]

$N(\cdot|\cdot,\cdot)$, $\chi^{-2}(\cdot|\cdot,\cdot)$, $G(\cdot|\cdot,\cdot)$, $Exp(\cdot|\cdot)$, $B(\cdot|\cdot,\cdot)$ denote normal, scaled inverse Chi-squared, gamma, exponential and beta densities, respectively. (1) `type` can take values `"FIXED"`, `"gamma"`, or `"beta"`; (2) `probIn` represents the prior probability of a marker having a non-null effect (π_0), counts (the number of 'prior counts') can be used to control how informative the prior is; (3) `V` and `d` represent the eigen-vectors and eigen-values of $\boldsymbol{K}$, respectively.

B Default rules for choosing hyper-parameters

BGLR has built-in rules to set values of hyper-parameters. The default rules assign proper, but weakly informative, priors with prior modes chosen in a way that, a priori, they obey a variance partition of the phenotype into components attributable to the error terms and to each of the elements of the linear predictor. The user can control this variance partition by setting the argument R2 (representing the **model R-squared**) of the BGLR function to the desired value. By default the model R2 is set equal to 0.5, in which case hyper-parameters are chosen to match a variance partition where 50% of the variance of the response is attributable to the linear predictor and 50% to model residuals. Each of the **elements of the linear predictor** has its own R2 parameter (see last column of Table S1). If these are not provided, the **R2** attributable to each element of the linear predictor equals the R-squared of the model divided the number of elements in the linear predictor. Once the R2 parameters are set, BGLR checks whether each of the hyper-parameters have been specified and if not, the built in-rules are used to set values for these hyper-parameters. Next we briefly describe the built-in rules implemented in BGLR; these are based on formulas similar to those described by de los Campos et al. (2013) implemented using the prior mode instead of the prior mean.

Variance parameters. The residual variance (σ_ε^2), $\sigma_{u_l}^2$ of the RKHS model, and σ_β^2, of the BRR, are assigned scaled-inverse Chi-square densities, which are indexed by a **scale** and a **degree of freedom** parameter. By default, if degree of freedom parameter is not specified, these are set equal to 5 (this gives a relatively un-informative scaled-inverse Chi-square and guarantees a finite prior variance) and the scale parameter is solved for to match the desired variance partition. For instance, in case of the residual variance the scale is calculated using $S_\varepsilon = var(y) \times (1 - R2) \times (df_\varepsilon + 2)$, this gives a prior mode for the residual variance equal to $var(y) \times (1 - R2)$. Similar rules are used in case of other variance parameters. For instance, if one element of the linear predictor involves a linear regression of the form $\boldsymbol{X\beta}$ with `model='BRR'` then $S_\beta = var(y) \times R2 \times (df_\beta + 2)/MSx$ where MSx is the sum of the sample variances of the columns of $\boldsymbol{X}$ and R2 is the proportion of phenotypic variance a-priori assigned to that particular element of the linear predictor. The selection of the scale parameter when the model is the RKHS regression is modified relative to the above rule to account for the fact that the average diagonal value of $\boldsymbol{K}$ may be different than 1, specifically we choose the scale parameter according to the following formula $S_l = var(y) \times R2 \times (df_l + 2)/mean(diag(\boldsymbol{K}))$.

In models **BayesA** and **BayesB** the scale-parameter indexing the t-prior assigned to marker effects is assigned a Gamma density with rate and shape parameters r and s, respectively. By default BGLR sets $s = 1.1$ and solves for the rate parameter using $r = (s-1)/S_\beta$ with $S_\beta = var(y) \times R2 \times (df_\beta + 2)/MSx$, here, as before, MSx represents the sum of the variances of the columns of X.

For the **BL**, the default is to set: `type='gamma'`, fix the shape parameter of the gamma density assigned λ^2 to 1.1 and then solve for the rate parameter to match the expected proportion of variance accounted for by the corresponding element of the linear predictor, as specified by the argument R2. Specifically, we set the rate to be $r = (s - 1)/(2 \times (1 - R2)/R2 \times MSx)$.

For models **BayesB** and **BayesC**, the default rule is to set $\pi_0 = 0.5$ and $p_0 = 10$. This gives a weakly informative beta prior for π with a prior mode at 0.5. The scale and degree-of freedom parameters entering in the priors of these two models are treated as in the case often models

BayesA (in the case of BayesB) and BRR (in the case of BayesC), but the rules are modified by considering that, a-priori, only a fraction of the markers (π) nave non-null effects; therefore, in BayesC we use $S_\beta = var(y) \times R2 \times (df_\beta + 2)/MSx/\pi$ and in BayesB we set $r = (s-1)/S_\beta$ with $S_\beta = var(y) \times R2 \times (df_\beta + 2)/MSx/\pi$.

C Supplementary R scripts

Box S1 illustrates how to extract estimates and predictions form the models fitted in Box 7.

Box S1: Supplementary code for the model fitted in Box 7

```
#Residual Variance
 fmBRR$varE; fmBRR$SD.varE
 fmBA$varE; fmBA$SD.varE
 fmBB$varE; fmBB$SD.varE
# DIC and pD
 fmBRR$fit
 fmBA$fit
 fmBB$fit
#Predictions
 fmBRR$yHat; fmBRR$SD.yHat
 fmBA$yHat;  fmBA$SD.yHat
 fmBB$yHat;  fmBB$SD.yHat
#Correlations between predicted and simulated signals
 cor(signal,fmBRR$yHat)
 cor(signal,fmBA$yHat)
 cor(signal,fmBB$yHat)

# Estimated effects
 tmp<-range(abs(b0))
 plot(numeric()~numeric(),ylim=tmp,xlim=c(1,p),
      ylab=expression(paste("|",beta[j],"|")),
      xlab="Marker Possition (order)")
 abline(v=whichQTL,lty=2,col=4)
 points(x=whichQTL,y=abs(b0[whichQTL]),pch=19,col=4)
 points(x=1:p,y=abs(fmBRR$ETA$MRK$b),col=1,cex=.5)
 lines(x=1:p,y=abs(fmBRR$ETA$MRK$b),col=1,cex=.5)
 points(x=1:p,y=abs(fmBB$ETA$MRK$b),col=2,cex=.5)
 lines(x=1:p,y=abs(fmBB$ETA$MRK$b),col=2,cex=.5)
```

Box S2 illustrates how to extract estimates and predictions form the models fitted in Box 8.

Box S2: Supplementary code for the model fitted in Box 8

```
#1# Estimated Marker Effects & posterior SDs
  bHat<- fm$ETA$MRK$b
  SD.bHat<- fm$ETA$MRK$SD.b
  plot(bHat^2, ylab="Estimated Squared-Marker Effect",
        type="o",cex=.5,col="red",main="Marker Effects",
        xlab="Marker")
   points(bHat^2,cex=0.5,col="blue")

#2# Predictions
  # Genomic Prediction
    gHat<-X%*%fm$ETA$MRK$b
    plot(fm$y~gHat,ylab="Phenotype",
         xlab="Predicted Genomic Value", col=2, cex=0.5,
         main="Predicted Genomic Values Vs Phenotypes",
         xlim=range(gHat),ylim=range(fm$y));

#3# Godness of fit and related statistics
   fm$fit
   fm$varE # compare to var(y)

#4# Trace plots
  list.files()

  # Residual variance
   varE<-scan("varE.dat")
   plot(varE,type="o",col=2,cex=.5,
          ylab=expression(sigma[epsilon]^2),
          xlab="Sample",main="Residual Variance");
   abline(h=fm$varE,col=4,lwd=2);
   abline(v=fm$burnIn/fm$thin,col=4)

  # lambda (regularization parameter of the Bayesian LASSO)
   lambda<-scan("ETA_MRK_lambda.dat")
   plot(lambda,type="o",col=2,cex=.5,
          xlab="Sample",ylab=expression(lambda),
          main="Regularization parameter");
   abline(h=fm$ETA$MRK$lambda,col=4,lwd=2);
   abline(v=fm$burnIn/fm$thin,col=4)
```

Box S3 shows how to extract estimates, predictions and variance components from the regression model fitted using the script provided in Box 9.

Box S3: Supplementary code for the model fitted in Box 9

```
#1# Predictions
  ## Phenotype prediction
    yHat<-fm$yHat
    tmp<-range(c(y,yHat))
    plot(yHat~y,xlab="Observed",ylab="Predicted",col=2,
            xlim=tmp,ylim=tmp); abline(a=0,b=1,col=4,lwd=2)

#2# Godness of fit and related statistics
   fm$fit
   fm$varE # compare to var(y)

#3# Variance components associated with the genomic and pedigree
  # matrices
  fm$ETA$PED$varU
  fm$ETA$PED$SD.varU

  fm$ETA$MRK$varU
  fm$ETA$MRK$SD.varU

#4# Trace plots
  list.files()
  # Residual variance
  varE<-scan("PGBLUP_varE.dat")
  plot(varE,type="o",col=2,cex=.5);

  #varA and varU
  varA<-scan("PGBLUP_ETA_PED_varU.dat")
  plot(varA,type="o",col=2,cex=.5);

  varU<-scan("PGBLUP_ETA_MRK_varU.dat")
  plot(varU,type="o",col=2,cex=.5)

  plot(varA~varU,col=2,cex=.5,main=paste("Cor= ",round(cor(varU,varA),3),sep=""))

  varG<-varU+varA
  h2<-varG/(varE+varG)
  mean(h2);sd(h2)

  mean(varU/varG)
  mean(varA/varG)
```

Box S4 provides supplementary code for the model fitted in Box 10.

Box S4: Supplementary code for the model fitted in Box 10

```
fm$varE
plot(y~fm$yHat)

plot(scan("RKHS_h=0.25_ETA_K1_varU.dat"),type="o",col=2,cex=0.5)
abline(h=fm$ETA$K1$varU,col=4)
plot(scan("RKHS_h=0.25_varE.dat"),type="o",col=2,cex=0.5)
abline(h=fm$varE,col=4)
```

Box S5 provides supplementary code for the model fitted in Box 11.

Box S5: Supplementary code for the model fitted in Box 11

```
# Posterior mean of the residual variance
fm$varE

# Posterior means of the variances of the kernels
VAR<-c(fm$ETA[[1]]$varU, fm$ETA[[2]]$varU, fm$ETA[[3]]$varU)
names(VAR)<-paste("Variance(h=",h,")",sep="")
barplot(VAR,ylab="Estimated Variance")

# Plots of variance components
varE<-scan("RKHS_KA_varE.dat")
varU1<-scan("RKHS_KA_ETA_1_varU.dat")
varU2<-scan("RKHS_KA_ETA_2_varU.dat")
varU3<-scan("RKHS_KA_ETA_3_varU.dat")
varU<-varU1+varU2+varU3

plot(varE,col=2,type="o",cex=.5,ylab="Residual Variance")
plot(varU,col=2,type="o",cex=.5,ylab="Variance",main="Genomic Variance")
plot(varU1,col=2,type="o",cex=.5,ylab="Variance",main=paste("Variance (h=",h[1],")"))
plot(varU2,col=2,type="o",cex=.5,ylab="Variance",main=paste("Variance (h=",h[2],")"))
plot(varU3,col=2,type="o",cex=.5,ylab="Variance",main=paste("Variance (h=",h[3],")"))
```

Box S6 provides supplementary code for the model fitted in Box 12.

Box S6: Supplementary code for the model fitted in Box 12

```
  # Assesment of correlation in TRN and TST data sets
   cor(fm$yHat[tst],y[tst])    #TST
   cor(fm$yHat[-tst],y[-tst])  #TRN

# Plot of phenotypes versus genomic prediction, by set (TRN/TST)
   plot(y~I(fm$yHat),ylab="Phenotype",
        xlab="Pred. Gen. Value" ,cex=.8,bty="L")
   points(y=y[tst],x=fm$yHat[tst],col=2,cex=.8,pch=19)
   legend("topleft", legend=c("training","testing"),bty="n",
          pch=c(1,19), col=c("black","red"))
   abline(lm(I(y[-tst])~I(fm$yHat[-tst]))$coef,col=1,lwd=2)
   abline(lm(I(y[tst])~I(fm$yHat[tst]))$coef,col=2,lwd=2)
```

Box S7 provides supplementary code for the model fitted in Box 13.

Box S7: Supplementary code for the model fitted in Box 13

```
# Comparing models using a paired t-test
 colMeans(COR)
 mean(COR[,2]-COR[,1])
 t.test(x=COR[,2],y=COR[,1],paired=TRUE,var.equal=FALSE)

# Plots of Correlations: Pedigree+Markers vs Pedigree Only
xy_limits<-range(as.vector(COR))
plot(COR[,2]~COR[,1],col="red",
     xlim=xy_limits,
     ylim=xy_limits,
     main="E1",
     xlab="Pedigree", ylab="Pedigree+Markers")
abline(0,1,col="blue")
```

D Regression with Ordinal and Binary Traits

For categorical traits BGLR uses the probit link and the phenotype vector should be coercible to a factor. The type of response is defined by setting the argument `response_type`. By default this argument is set equal to `"Gaussian"`. For binary and ordinal outcomes we should set `response_type="ordinal"`. Box S8 provides a simple example that uses the wheat data set with a discretized phenotype. The second block of code, `#2#`, presents the analysis of a binary outcome, and the third one, `#3#`, that of an ordinal trait. Figure S1 shows, for the binary outcome, a plot of predicted probability (`fmBin$probs`) versus realized value in the TRN and TST datasets.

Box S8: Fitting models with binary and ordinal responses

```
#1# Loading and preparing the input data
 library(BGLR); data(wheat);
 Y<-wheat.Y; X<-wheat.X; A<-wheat.A;
 y<-Y[,1]
 set.seed(123)
 tst<-sample(1:nrow(X),size=150)
 #2# Binary outcome
 yBin<-ifelse(y>0,1,0)
 yBinNA<-yBin; yBinNA[tst]<-NA
 ETA<-list(list(X=X,model="BL"))

 fmBin<-BGLR(y=yBinNA,response_type="ordinal", ETA=ETA,
          nIter=1200,burnIn=200)

 head(fmBin$probs)
 par(mfrow=c(1,2))
 boxplot(fmBin$probs[-tst,2]~yBin[-tst],main="Training",ylab="Estimated prob.")
 boxplot(fmBin$probs[tst,2]~yBin[tst],main="Testing", ylab="Estimated prob.")

#2# Ordinal outcome
 yOrd<-ifelse(y<quantile(y,1/4),1,ifelse(y<quantile(y,3/4),2,3))
 yOrdNA<-yOrd; yOrdNA[tst]<-NA

 ETA<-list(list(X=X,model="BL"))

 fmOrd<-BGLR(y=yOrdNA,response_type="ordinal", ETA=ETA,
          nIter=1200,burnIn=200)
 head(fmOrd$probs)
```

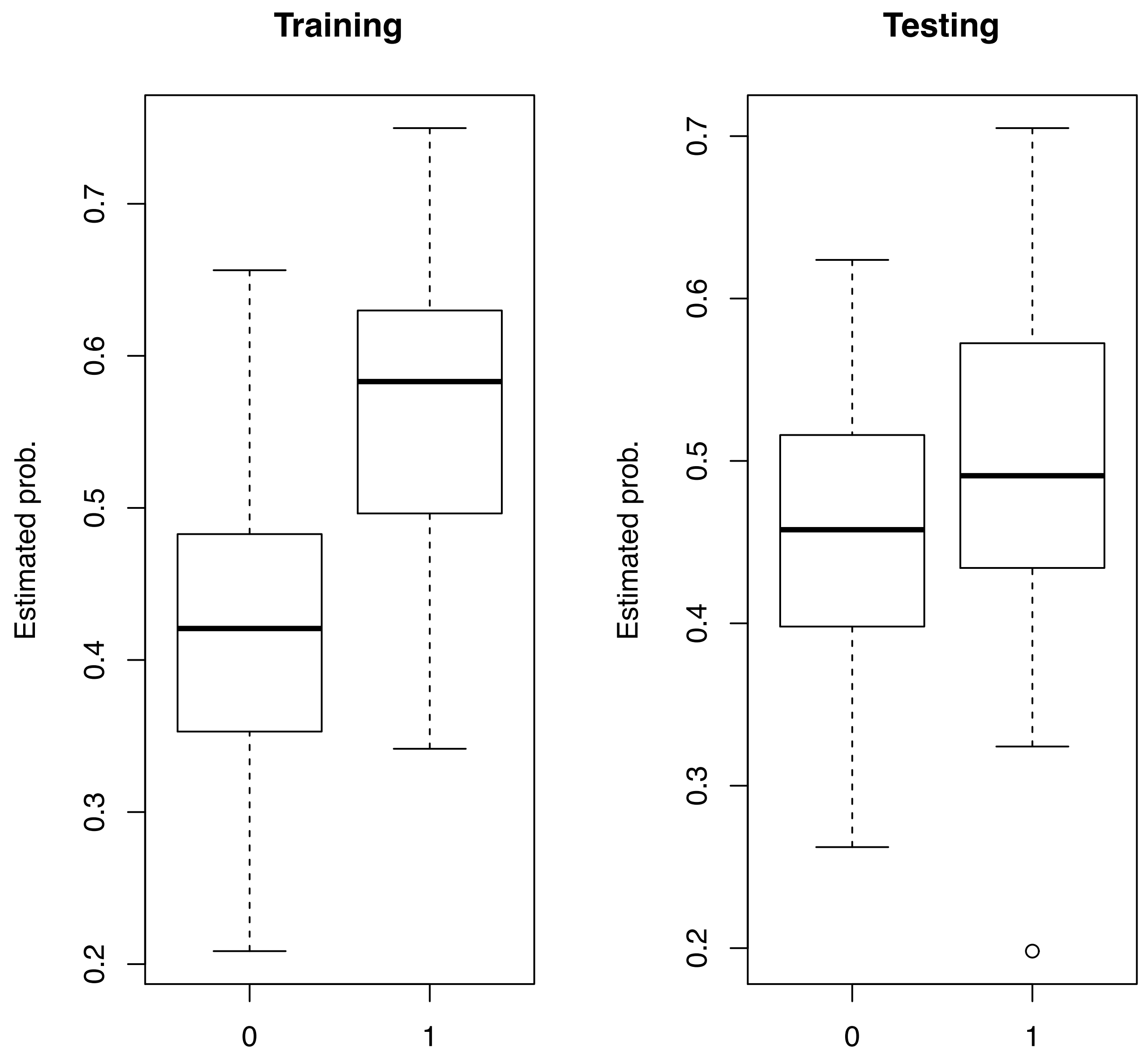

Figure S1: Estimated probability by category, versus observed category (binary response).

E Regression with Censored Outcomes

Box S9 illustrates how to fit a model to a censored trait. Note that in the case of censored trait the response is specified using a triplet (a_i, y_i, b_i) (see Table 2 for further details). For assessment of prediction accuracy (not done in Box S9), one can set $a_i = -\infty$, $y_i = NA$, $b_i = \infty$ for individuals in testing data sets, this way there is no information about the ith phenotype available for the model fit.

Box S9: Fitting censored traits

```
#1# Loading and preparing the input data
 library(BGLR); data(wheat);
 Y<-wheat.Y; X<-wheat.X; A<-wheat.A;
 y<-Y[,1]
 set.seed(123)

#censored
 n<-length(y)
 cen<-sample(1:n,size=200)
 yCen<-y
 yCen[cen]<-NA
 a<-rep(NA,n)
 b<-rep(NA,n)
 a[cen]<-y[cen]-runif(min=0,max=1,n=200)
 b[cen]<-Inf

#models
ETA<-list(list(X=X,model="BL"))

fm<-BGLR(y=yCen,a=a,b=b,ETA=ETA,nIter=12000,burnIn=2000)

cor(y[cen],fm$yHat[cen])
```

File S2

Boxes.R

Available for download at http://www.genetics.org/lookup/suppl/doi:10.1534/genetics.114.164442/-/DC1