BGLR: A Statistical Package for Whole Genome Regression and Prediction

Paulino Pérez
Colegio de Postgraduados, México

Gustavo de los Campos
University of Alabama at Birmingham

Abstract

Many modern genomic data analysis problems require implementing regressions where the number of unknowns (p, e.g., the number of marker effects) vastly exceeds sample size (n). Implementing these *large-p-with-small-n* regressions poses several statistical and computational challenges. Some of these challenges can be confronted using Bayesian methods, and the Bayesian approach allows integrating various parametric and non-parametric shrinkage and variable selection procedures in a unified and consistent manner. The **BGLR** R-package implements a large collection Bayesian regression models, including various parametric regressions where regression coefficients are allowed to have different types of prior densities (flat, normal, scaled-t, double-exponential and various finite mixtures of the spike-slab family) and semi-parametric methods (Bayesian reproducing kernel Hilbert spaces nregressions, RKHS). The software was originally developed as an extension of the **BLR** package and with a focus on genomic applications; however, the methods implemented are useful for many non-genomic 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.

Keywords: Bayesian Methods, Regression, Whole Genome Regression, Whole Genome Prediction, Genome Wide Regression, Variable Selection, Shrinkage, semi-parametric regression, RKHS, R.

1. Introduction

Many modern statistical learning problems involve the analysis of highly 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 posses 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 of interactions between alleles and with environmental factors. Recent developments in the areas of shrinkage estimation, both in the penalized and Bayesian regression frameworks, as well as in computational methods 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.

VanRaden *et al.* 2009; Hayes *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 last decade a large collection of parametric and non-parametric methods have been proposed and empirical evidence has demonstrated that there is no single approach that 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, 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) exists, most statistical packages implement a few types of methods and there is need of integrating these methods in a unified statistical and computational framework. Motivated by this need we have developed the R (R Core Team 2012) package **BGLR** (de los Campos and Pérez 2013). The package is available at CRAN and at the R-forge website `https://r-forge.r-project.org/projects/bglr/`.

Models. **BGLR** can be used with **continuous** (censored or not) and **categorical** traits (binary and ordinal). The user has control in choosing the prior assigned to effects and this can be used to control the extent and type of shrinkage of estimates. For **parametric linear regressions on covariates** (e.g., genetic markers, non-genetic co-variates) the user can choose a variety of prior densities, from flat priors (the so-called ***'fixed effects'***, a method that does not induce shrinkage of estimates) to priors that induce different types of shrinkage, including: Gaussian (**Bayesian Ridge Regression**, **BRR**), scaled-t (**BayesA** Meuwissen *et al.* 2001), Double-Exponential (**Bayesian LASSO**, **BL** Park and Casella 2008), and ***two component mixtures*** with a point of mass at zero and a with a slab that can be either Gaussian (**BayesC**, Habier *et al.* 2011) or scaled-t (**BayesB**, Habier *et al.* 2011). The **BGLR** package also implements ***Bayesian Reproducing Kernel Hilbert Spaces Regressions*** (**RKHS,** Wahba 1990) using Gaussian processes with arbitrarily user-defined co-variance structures. This class of models allows implementing semi-parametric regressions for various types of problems, including, scatter-plot smoothing (e.g., ***smoothing splines*** Wahba 1990), **spatial smoothing** (Cressie 1988), ***Genomic-BLUP*** (VanRaden 2008), non-parametric ***RKHS*** genomic regressions (Gianola *et al.* 2006; Gianola and van Kaam 2008; de los Campos *et al.* 2010) and ***pedigree-BLUP*** (Henderson 1975).

All the above-mentioned prior densities (e.g., Gaussian, Double Exponential, Scaled-t, finite mixtures) are index by **regularization parameters** that control the extent of shrinkage of estimates; rather than fixing them to some user-specified values we treat them as random. Consequently, in a deeper level of the hierarchal model these regularization parameters are assigned prior densities.

Algorithms. In **BGLR** samples from the posterior density are drawn using a Gibbs sampler (Geman and Geman 1984; Casella and George 1992); with scalar updating. This approach is very flexible but computationally demanding. To confront the computational challenges emerging in Markov Chain Monte Carlo (MCMC) implementations we have adopted a strategy that combines: (a) the use of built-in R functions for operations that can be vectorized with (b) customized compiled code (C and Fortran) developed to perform operations that cannot be vectorized. Thus, the kernel of our software is written in R, but the computationally demanding steps are carried out using customized routines written in C and Fortran code. The implementation makes use of BLAS routines `daxpy` and `ddot`. The computational performance of the algorithm can be greatly improved if R is linked against a tuned BLAS implementation with multithread support, for example OpenBLAS, ATLAS, Intel mkl, etc.

Ancillary functions and data sets. In addition to the main function (BGLR) the package comes with: (a) functions to read and write from the R-console `*.ped` and `*.bed` files (Purcell *et al.* 2007), (b) two publicly available data sets (see Section 4 for further details) and (c) various examples (type `demo(package='BGLR')` in the R-console).

In what remains of the article we discuss the **methods implemented** (**Section 2**), the **user interface** (**Section 3**), and the **data sets** included (**Section 4**) in the **BGLR** package in detail. **Application examples** are given in **Section 5**. A small benchmark is given in **Section 6**. Finally, the article is closed in **Section 7** with a few **concluding remarks**.

2. Statistical Models and Algorithms

The **BGLR** supports models for continuous (censored or not) and categorical (binary or ordinal multinomial) traits. We begin by considering the case of a continuous response without censoring; categorical and censored data are considered later on.

2.1. Conditional distribution of the data

For a continuous response (y_i; $i = 1, ..., n$) 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

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

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

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

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

where μ is an intercept, $\boldsymbol{X}_j$ are design matrices for predictors, $\boldsymbol{X}_j = \{x_{ijk}\}$, $\boldsymbol{\beta}_{jk}$ are vectors of effects associated to the columns of $\boldsymbol{X}_j$ and $\boldsymbol{u}_l = \{u_{l1}, ..., u_{ln}\}$ 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 likelihood:

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

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

2.2. Prior densities

The **residual variance** is assigned a Scaled-inverse Chi-square density $p(\sigma^2_\varepsilon) = \chi^{-2}(\sigma^2_\varepsilon|S_\varepsilon, df_\varepsilon)$ with degree of freedom df_ε (>0) and scale parameters S_ε (>0) and the intercept (μ) is assigned a flat prior. In the parameterization used in **BGLR**, the prior expectation of the Scaled-inverse Chi-square density $\chi^{-2}(\cdot|S., df.)$ is given by $\frac{S.}{df.-2}$.

Regression coefficients $\{\beta_{jk}\}$ can be assigned either un-informative (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 coefficient assigned informative priors, the choice of the prior will play 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*** induce 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, and they induce 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 usually refereed in the literature on GS as to **BayesC** (Habier *et al.* 2011) and a mixture of a point of mass at zero and a scaled-t slab, a model known as **BayesB** (Meuwissen *et al.* 2001). By assigning a non-null prior probability for the marker effect to be equal to zero, the priors used in BayesB and BayesC have potential for inducing variable selection.

Hyper-parameters. Each of the prior distributions above-described are indexed by one or more parameters that control the type and extent of shrinkage induced. We treat these regularization parameters as unknown; consequently a prior is assigned to these unknowns. Table 1 lists, for each of the prior densities implemented the set of hyper-parameters. Further details about how regularization parameters are inferred from the data are given in the Appendix.

Combining priors. Different priors can be specified for each of the elements of the linear predictor, $\{\boldsymbol{X}_1, \boldsymbol{X}_2, ..., \boldsymbol{X}_J, \boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_L\}$, 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 Box 3a of Section 5.

Gaussian Processes. The vectors of random effects $\boldsymbol{u}_l$ are assigned multivariate-normal priors with a mean equal to zero and co-variance matrix $Cov(\boldsymbol{u}_l, \boldsymbol{u}'_l) = \boldsymbol{K}_l\sigma^2_{ul}$ where $\boldsymbol{K}_l$ is an $n \times n$ symmetric positive semi-definite matrix and σ^2_{ul} is a variance parameter with prior density $\sigma^2_{ul} \sim \chi^{-2}(df_l, S_l)$. Special classes of models that can be implemented using these random effects include standard pedigree-regression models (Henderson 1975) in which case $\boldsymbol{K}_l$ is a pedigree-derived co-variance matrix, Genomic BLUP (VanRaden 2008), which case $\boldsymbol{K}_l$ may be a marker-derived relationship matrix, or models for spatial regressions (Cressie 1988) in which case $\boldsymbol{K}_l$ may be a co-variance matrix derived from spatial information. Illustration about the inclusion of these Gaussian processes into models for data analysis are given in examples of Section 5.

2.3. Algorithms

The R-package **BGLR** draws samples from the posterior density using a Gibbs sampler (Ge-

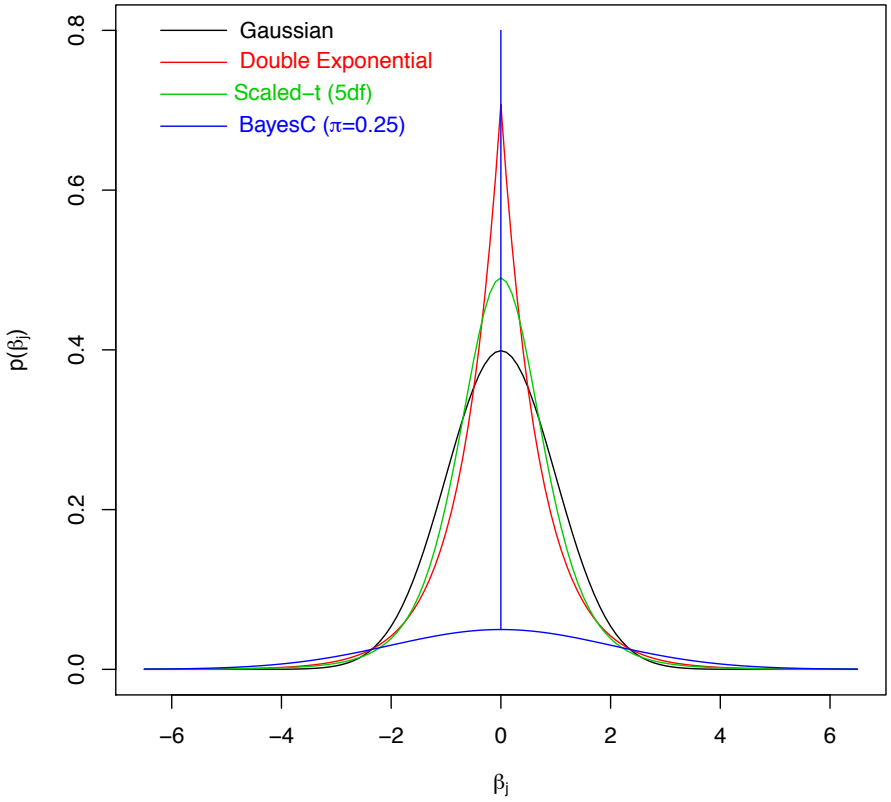

Figure 1: Prior Densities of Regression Coefficients Implemented in **BGLR**. All the densities displayed correspond to random variables with null mean and unit variance.

man 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 1974), and the finite-mixture priors are implemented using latent random Bernoulli variables linking effects to components of the mixtures.

Categorical traits. The argument `response_type` is used to indicate **BGLR** whether the response should be regarded as `'continuous'`, the default value, or `'ordinal'`. For continuos traits the response vector shoud be coercible to numeric; for ordinal traits the response can take onto K possible (ordered) values $y_i \in \{1, ..., K\}$ (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 above-described, 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 un-observed liability scores are sampled

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

Model (prior density)	**Hyper-parameters**	**Treatmend in BGLR**[1]
Flat	Mean (μ_β)	$\mu_\beta = 0$
(FIXED)	Variance (σ^2_β)	$\sigma^2_\beta = 1 \times 1^{10}$
Gaussian	Mean (μ_β)	$\mu_\beta = 0$
(BRR)	Variance (σ^2_β)	$\sigma^2_\beta \sim \chi^{-2}$
Scaled-t	Degrees of freedom (df_β)	User-specified (default value, 5)
(BayesA)	Scale (S_β)	$S_\beta \sim Gamma$
Double-Exponential		λ Fixed, user specified, or
(BL)	λ^2	$\lambda^2 \sim Gamma$, or
		$\frac{\lambda}{\max} \sim Beta^2$
Gaussian Mixture	π (prop. of non-null effects)	$\pi \sim Beta$
(BayesC)	df_β	User-specified (default value, 5)
	S_β	$S_\beta \sim Gamma$
Scaled-t Mixture	π (prop. of non-null effects)	$\pi \sim Beta$
(BayesB)	df_β	User-specified (default value, 5)
	S_β	$S_\beta \sim Gamma$

1: Further details are given in the Appendix. 2: This approach is further discussed in de los Campos *et al.* (2009b).

from truncate normal densities; once the un-observed liability has been sampled the Gibbs sampler proceed as if l_i were observed (see Albert and Chib 1993, for further details).

Missing data. 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 a triplet $\{a_i, y_i, b_i\}$; the elements of this triplet 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 un-censored 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 $(\boldsymbol{y}, \boldsymbol{a}, \boldsymbol{b})$. The vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ have `NULL` as default value; therefore, if only $\boldsymbol{y}$ is provided this is interpreted as a continuous trait without censoring. If $\boldsymbol{a}$ and $\boldsymbol{b}$ are provided together with $\boldsymbol{y}$ data is treated as censored. We treat censoring as a missing data problem; the missing values of y_i present due to censoring are sampled from truncated normal densities that satisfy $a_i < y_i < b_i$. Further details about models for censored data are given in examples of section 5.

3. Software interface

The R-package **BGLR** (de los Campos and Pérez 2013) inherits part of its user interface from **BLR** (de los Campos and Pérez 2010). A detailed description of this package can be

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

Type of point	a_i	y_i	b_i
Un-censored	`NULL`	y_i	`NULL`
Right censored	a_i	`NA`	`Inf`
Left censored	`-Inf`	`NA`	b_i
Interval censored	a_i	`NA`	b_i

found in (Pérez *et al.* 2010); however we have modified key elements of the user-interface, and the internal implementation, to provide the user more flexibility in building models for data analysis. 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 1a: 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 are run, and the 1st 500 samples are discarded. 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 estimated posterior means and several arguments used in the call.

Inputs

Box 1b 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 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.
- `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 defaults 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 degree of freedom and `S0` 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 the Appendix for further details).

Box 1b: 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. The parameters used to fit the model are also returned within the list. Box 1c shows the structure of the object returned after fitting the intercept model of Box 1a. The first element of the list `(y)` is the response vector used in the call to **BGLR**, `$whichNa` gives the index of the entries in `y` that were missing, these two elements are then followed by several entries describing the call (omitted in Box 1c), this is followed by estimated posterior means and estimated posterior standard deviations of the linear predictor (`$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 1c: Structure of the object returned by BGLR (after running the code in Box 1a)

```
str(fm)
List of 20
 $ 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 1d shows an example of the files generated after executing the commands given in Box 1a. 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) provide several useful functions for the analysis of samples used in Monte Carlo algorithms.

Box 1d: Files generated by BGLR (after running the code in Box 1a)

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

4. Datasets

The **BGLR** package comes with two genomic datasets 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 1,814 mice. Several phenotypes are available in the data frame `mice.phenos`. Each mouse was genotyped at 10,346 SNPs. We removed SNPs with minor allele frequency (MAF) smaller than 0.05, and missing marker genotypes imputed with the corresponding average genotype calculated 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 (average of two replicates) at four different environments; phenotypes (`wheat.Y`) were centered and standardized to a unit variance within environment. Each of the lines were genotyped for 1,279 Diversity Array Technology (DArT) markers. At each marker two possibly homocygous 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 2 illustrates how to load the wheat and mice data sets.

Box 2: Loading the mice data set included in BGLR

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

5. Application Examples

In this section we illustrate the use of **BGLR** with examples.

Fitting Models for Fixed and Random Effects for a Continuous Response

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. One possible linear model that we can fit to some of the continuous traits available in the mice data set is as follows:

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

where μ is an intercept, $\boldsymbol{X}_1$ is a design matrix for the effects of sex and litter size, and $\boldsymbol{\beta}_1$ is the corresponding vector of effects, which will be treated as 'fixed'; $\boldsymbol{X}_2$ is the design matrix for the effects of cage and $\boldsymbol{\beta}_2$ is the vector of cage effects which will treat as random (in the example fo Box 3a we assign a Gaussian prior to these effects); $\boldsymbol{X}_3$ is the matrix with marker genotypes and $\boldsymbol{\beta}_3$ the corresponding vector of marker effects to which, in the example below, we assign IID double-exponential priors.

Fitting the model. The code provided in Box 3a illustrates how to fit the model above-described using **BGLR**. 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 the element of the linear predictor. We can specify the predictors to be included in each of the inner lists either by providing the design matrix or by using a formula. When the formula is used, the design matrix is created internally using the `model.matrix()` function of R. Finally in the 3rd block of code we fit the model by calling the `BGLR()` function.

Box 3a: Fitting a model to markers and non-genetic effects in BGLR

```
#1# Loading and preparing the input data
 library(BGLR); data(mice);
 Y<-mice.pheno; X<-mice.X; A=mice.A;
 y<-Y$Obesity.BMI; y<-(y-mean(y))/sd(y)

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

#3# Fitting the model
  fm<-BGLR(y=y,ETA=ETA, nIter=12000, burnIn=2000)
  save(fm,file='fm.rda')
```

When BGLR begins to run, a message warns the user that hyper-parameters were not provided

and that consequently they were set using built-in rules; further details about these rules are given in the Appendix.

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 of model complexity. Also, as **BGLR** run, it saves samples of some of the parameters; these samples can be brought into the R-environment for posterior analysis. Box 3b illustrates how to extract from the returned object estimates of the posterior means and of the posterior deviations and how to create trace and density plots.

The first block of code (`#1#`) in Box 3b shows how to extract estimated posterior means and posterior standard deviations of effects. In this case we extract those corresponding to the third element of the linear predictor (`fm$ETA[[3]]`) which correspond to the markers, but the same could be done for any of the elements of the linear predictors. For models involving linear regressions `$b` and `$SD.b` give the estimated posterior means and posterior standard deviations of effects. The second block (`#2#`) of code of Box 3b shows how to extract the estimated posterior mean of the linear predictor, and also how to compute the estimated posterior mean of particular elements of the linear predictor, in this case we illustrate with genomic values (gHat). The third block of code (`#3#`) illustrates how to extract DIC (Spiegelhalter *et al.* 2002) and related statistics; finally, the fourth block of code (`#4#`) shows how to retrieve samples from the posterior distribution and produce trace plots. The plots produced by the code in Box 3b are given in Figure 2.

Box 3b: Extracting results from a model fitted using BGLR (continues from Box 3a)

```
#1# Estimated Marker Effects & posterior SDs
  bHat<- fm$ETA[[3]]$b
  SD.bHat<- fm$ETA[[3]]$SD.b
  plot(bHat^2, ylab='Estimated Squared-Marker Effect',
        type='o',cex=.5,col=4,main='Marker Effects')

#2# Predictions
  # Total 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)

  # Just the genomic part
    gHat<-X%*%fm$ETA[[3]]$b
    plot(gHat~y,xlab='Phenotype',
         ylab='Predicted Genomic Value',col=2,
         xlim=tmp,ylim=tmp); abline(a=0,b=1,col=4,lwd=2)

#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(var[e]));
   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_3_lambda.dat')
   plot(lambda,type='o',col=2,cex=.5,ylab=expression(lambda));
   abline(h=fm$ETA[[3]]$lambda,col=4,lwd=2);
   abline(v=fm$burnIn/fm$thin,col=4)
```

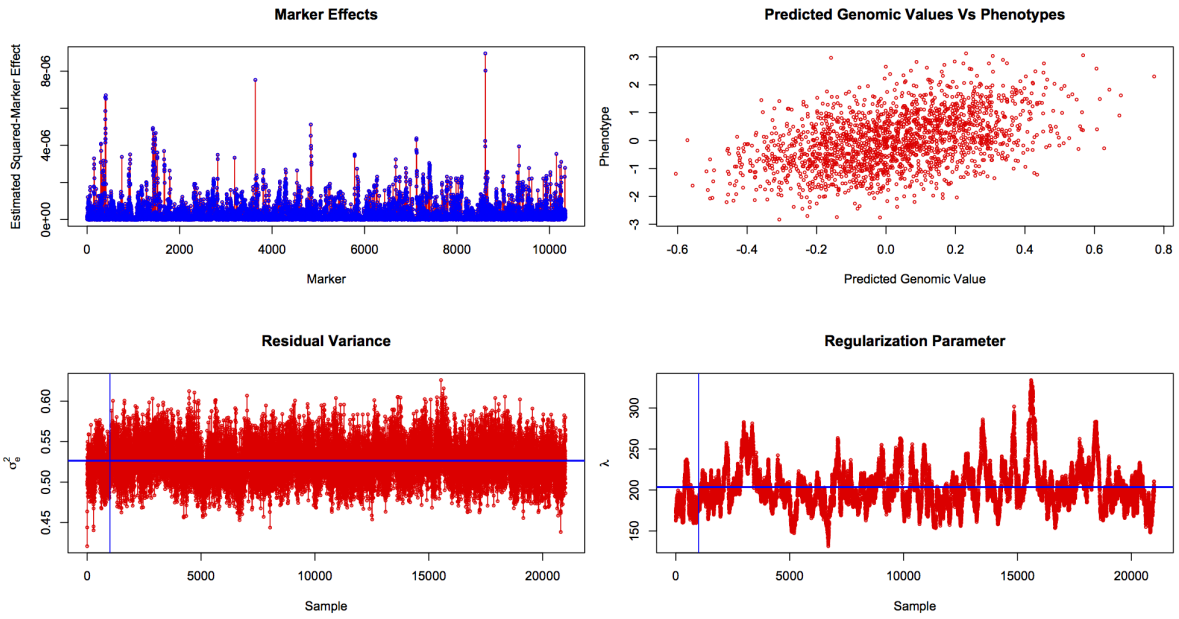

Figure 2: Squared-Estimated Marker Effects (top-left), phenotype versus predicted genomic values (top-right), trace plot of residual variance (lower-left) and trace plot of regularization parameter of the Bayesian Lasso (lower-right).

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 package. In the example of Box 4a we include two random effects, one representing a regression on pedigree, $\boldsymbol{a} \sim N(\mathbf{0}, \boldsymbol{A}\sigma^2_a)$, where $\boldsymbol{A}$ is a pedigree-derived numerator relationship matrix, and one representing a linear regression on markers, $\boldsymbol{g} \sim N(\mathbf{0}, \boldsymbol{G}\sigma^2_{gu})$ where, $\boldsymbol{G}$ is a marker-derived genomic relationship matrix. 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). Te user can implement a RKHS regression either by providing co-variance matrix ($\boldsymbol{K}$) or its eigen-value decomposition (see the example in Box 4a). When the co-variance matrix is provided, the eigen-value decomposition is computed internally.

Box 4a: Fitting a Pedigree + Markers regression using Gaussian Processes

```
#1# Loading and preparing the input data
 library(BGLR); data(wheat);
 Y<-wheat.Y; X<-wheat.X; A<-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(list(K=A, model='RKHS'),
           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')
```

Box 4b shows how to extract estimates, predictions, and samples from the fitted model. The first block of code (`#1`) shows how to obtain the predictions. The second block of code shows how to extract some goodness of fit related statistics. The third block of code shows how to extract the posterior mean of the variances components σ_a^2 and σ_{gu}^2. Note that in order to obtain the estimate it is necessary to specify the component number (1 or 2), this can be done by writing `fm$ETA[[1]]$varU` and `fm$ETA[[2]]$varU` respectively. Finally, the fourth block of code shows how to produce the trace plots for σ_a^2, σ_g^2 and σ_ε^2 (graphs not shown).

Box 4b: Extracting estimates, predictions, and samples from Reproducing Kernel Hilbert Spaces Regressions (continues from Box 4a first)

```
#1# Predictions
  # Total 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$ETA[[1]]$varU
  fm$ETA[[2]]$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_1_varU.dat')
plot(varA,type='o',col=2,cex=.5);

varU<-scan('PGBLUP_ETA_2_varU.dat')
plot(varU,type='o',col=2,cex=.5)
```

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 to use this approach for genomic prediction and since then several methodological and applied articles have been published (González-Recio *et al.* 2008; Gianola and de los Campos 2008; de los Campos *et al.* 2009a, 2010).

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 into the real line and must be positive semi-definite. For instance, if the information set is given by vectors of marker genotypes the RK, $K(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$ maps from pairs of vectors of genotypes, $\{\boldsymbol{x}_i, \boldsymbol{x}_{i'}\}$, onto the real line and must satisfy, $\sum_i \sum_{i'} \alpha_i \alpha_{i'} K(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) \geq 0$, for any non-null sequence of coefficients α_i. Following de los Campos *et al.* (2009a) the Bayesian RKHS regression can be represented as follows:

$$\left\{ \begin{array}{l} \boldsymbol{y} = \mathbf{1}\mu + \boldsymbol{u} + \boldsymbol{\varepsilon} \quad \text{with} \\ p(\mu, \boldsymbol{u}, \boldsymbol{\varepsilon}) \propto N(\boldsymbol{u}|\mathbf{0}, \boldsymbol{K}\sigma_u^2) N(\boldsymbol{\varepsilon}|\mathbf{0}, \boldsymbol{I}\sigma_\varepsilon^2) \end{array} \right. \tag{2}$$

where $\boldsymbol{K} = \{K(\boldsymbol{x}_i, \boldsymbol{x}_{i'})\}$ is an $(n \times n)$ matrix whose entries are the evaluations of the RK at pairs of points in input space. 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 ($\boldsymbol{A}$) replaced by the kernel matrix ($\boldsymbol{K}$). Box 5 features an example using a Gaussian Kernel evaluated in the (average) squared-Euclidean distance between genotypes, that is: $K(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = \exp\left\{-h \times \frac{\sum_{k=1}^{p}(x_{ik} - x_{i'k})^2}{p}\right\}$. In the example genotypes were centered and standardized, but this is not strictly needed. The bandwidth parameter controls how fast the co-variance function drops as the distance between pairs of vector genotypes increases. This parameter plays an important role. In this example we have chosen the bandwidth parameter to be equal to 0.5, further discussion about this parameter is given in next example.

Box 5: Fitting a Single Kernel Model in BGLR

```
#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#  Computing the distance matrix and then the krenel.
  X<-scale(X,center=TRUE,scale=TRUE)
  D<-(as.matrix(dist(X,method='euclidean'))^2)/p
  h<-0.5
  K<-exp(-h*D)

#3#  Single Kernel Regression using BGLR

 ETA<-list(list(K=K,model='RKHS'))
 fm<-BGLR(y=y,ETA=ETA,nIter=12000, burnIn=2000,saveAt='RKHS_h=0.5_')
```

Multi-Kernel Models. The bandwidth parameter of the Gaussian kernel can be chosen either using cross-validation (CV) or with Bayesian methods. The CV approach requires fitting models over a grid of values of h. The Bayesian approach estimates h, and all the model uknowns, form the data concurrently. The fully Bayesian treatment, which consist of treating h as unknown, is computationally demanding because, any time h is updated, the RK needs to be re-computed. To overcome this problem de los Campos *et al.* (2010) proposed to use a multi-kernel approach (named Kernel Averaging, KA) consisting on: (a) defining a sequence of kernels based on a set of reasonable values of h, and (b) fitting a multi-kernel model with as many random effects as kernels in the sequence. The model has the following form:

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

where $\boldsymbol{K}_l$ is the RK evaluated at the lth value of the bandwidth parameter in the sequence $\{h_1, ..., h_L\}$. 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 $N(\boldsymbol{u}|\mathbf{0}, \bar{\boldsymbol{K}}\sigma^2_u)$ where $\bar{\boldsymbol{K}}$ 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 or implementing a multi-kernel model requires defining a reasonable range for h. One possibility is to choose as a focal point for that range a value of h that gives a RK similar to the one given by the $\boldsymbol{G}$-matrix (this one represents the kernel for an additive model). The entries of the distance matrix $\boldsymbol{D} = \left\{D_{ii'} = \frac{\sum_{k=1}^{p}(x_{ik}-x_{i'k})^2}{p}\right\}$ can be calculated from the entries of the $\boldsymbol{G}$-matrix $\boldsymbol{G} = \left\{G_{ii'} = \frac{\sum_{k=1}^{p} x_{ik}x_{i'k}}{p}\right\}$; indeed, $D_{ii'} = G_{ii} + G_{i'i'} - 2G_{ii'}$. What value of h makes $\exp\{-h \times D_{ii'}\} \approx G_{ii'}$? Consider for instance a pair of full sibs in an outbreed population, in this case $E[G_{ii}] = E[G_{i'i'}] = 1$ and $E[G_{ii'}] = 0.5$ therefore, $E[D_{ii'}] = 1$, and using $h = 0.8$ we get $K(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = \exp\{-0.8\} \approx 0.4$. Similarly, for a pair of half-sibs se have: $E[G_{ii'}] = 0.5$, therefore, $E[D_{ii'}] = 1.5$ and $K(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = \exp\{-0.8 \times 1.5\} \approx 0.3$, which gives values for the RK for that type of relatives close to the ones given by the $\boldsymbol{G}$-matrix. In inbreed populations smaller values of h will be needed because $E[G_{ii'}] > 1$. Box 6 illustrates how to fit a multi-kernel model using $h = 0.5 \times \{1/5, 1, 5\}$. With this choice of values for the bandwidth parameter the model includes kernels that give correlations much smaller ($h = 2.5$) similar ($h = 0.5$), and much higher ($h = 0.1$) than the ones given by the $\boldsymbol{G}$ matrix. This is

illustrated in Figure 3 that displays the entries of the 1st row of the kernel matrix evaluated at each of the values of the bandwidth parameter in the grid.

Box 6: Fitting a RKHS Using a Multi-Kernel Methods (Kernel Averaging)

```
#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#  Computing D and then K
 X<-scale(X,center=TRUE,scale=TRUE)
 D<-(as.matrix(dist(X,method='euclidean'))^2)/p
 h<-0.5*c(1/5,1,5)

#3#  Kernel Averaging using BGLR
ETA<-list(list(K=exp(-h[1]*D),model='RKHS'),
          list(K=exp(-h[2]*D),model='RKHS'),
          list(K=exp(-h[3]*D),model='RKHS'))
fm<-BGLR(y=y,ETA=ETA,nIter=5000, burnIn=1000,saveAt='RKHS_KA_')

#1# Variance Components
  fm$ETA[[1]]$varU ; fm$ETA[[2]]$varU; fm$ETA[[3]]$varU
```

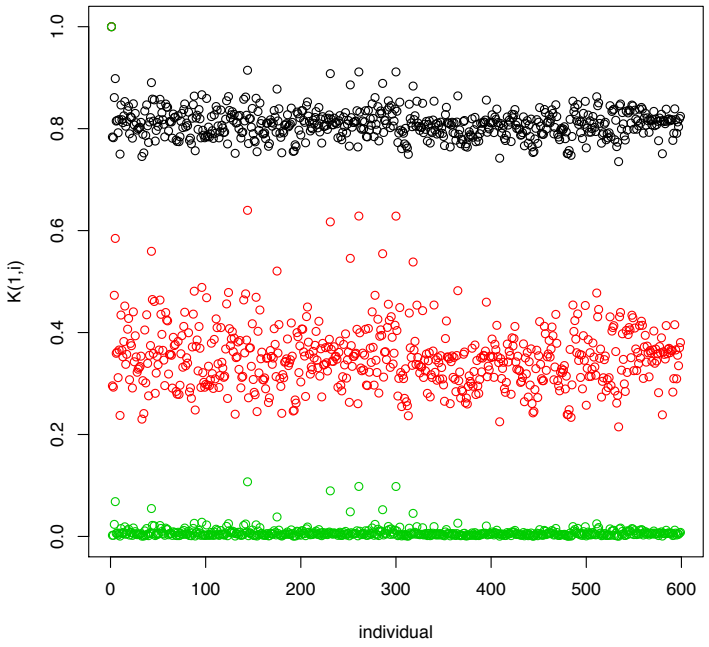

Figure 3: Entries of the 1st row of the (Gaussian) kernel matrix evaluated at three different values of the bandwidth parameter, $h = 0.5 \times \{1/5, 1, 5\}$.

Assessment of Prediction Accuracy

The 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 7 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 put missing values in the phenotypic vector (see Box 7). Once the model is fitted predictions for individuals in TST set can be obtained typing `fit$yHat[tst]` in the R command line. Figure 4 plots observed vs predicted phenotypes for individuals in training and TST sets.

Box 7: 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_')

  plot(fm$yHat,y,xlab="Phenotype",
     ylab="Pred. Gen. Value" ,cex=.8,bty="L")
  points(x=y[tst],y=fm$yHat[tst],col=2,cex=.8,pch=19)
  legend("topleft", legend=c("training","testing"),bty="n",
        pch=c(1,19), col=c("black","red"))

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

A cross-validation is simply a generalization of the TRN-TST evaluation presented in Box 7. 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.

5.1. 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'`.

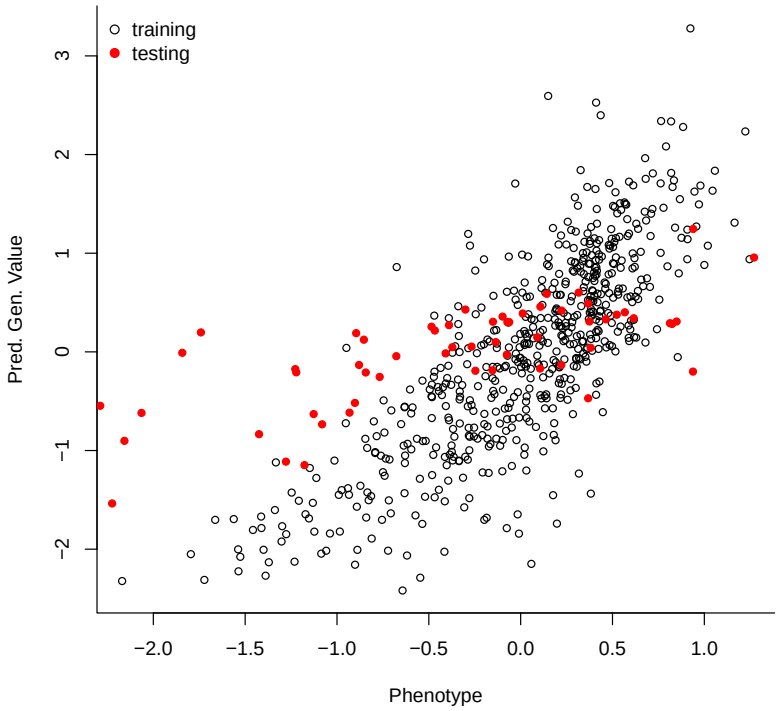

Figure 4: Estimated genetic values for training and testing sets. Predictions were derived using G-BLUP model (see Box7).

By default this argument is set equal to `'Gaussian'`. For binary and ordinal outcomes we should set `response_type='ordinal'`. Box 8 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 5 shows, for the binary outcome, a plot of predicted probability versus realized value in the TRN and TST datasets. The estimated posterior means and posterior standard deviations of marker effects and posterior means of the linear predictor are retrieved as described before (e.g., `fm$ETA[[1]]$b`, `fm$yHat`). For continuous outcomes the posterior mean of the linear predictor is also the conditional expectation function. For binary outcomes, the conditional expectation is simply the success probability; therefore, in this case **BGLR** also returns the estimated probabilities of each of the categories `fm$probs`).

Box 8: 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]
 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)
```

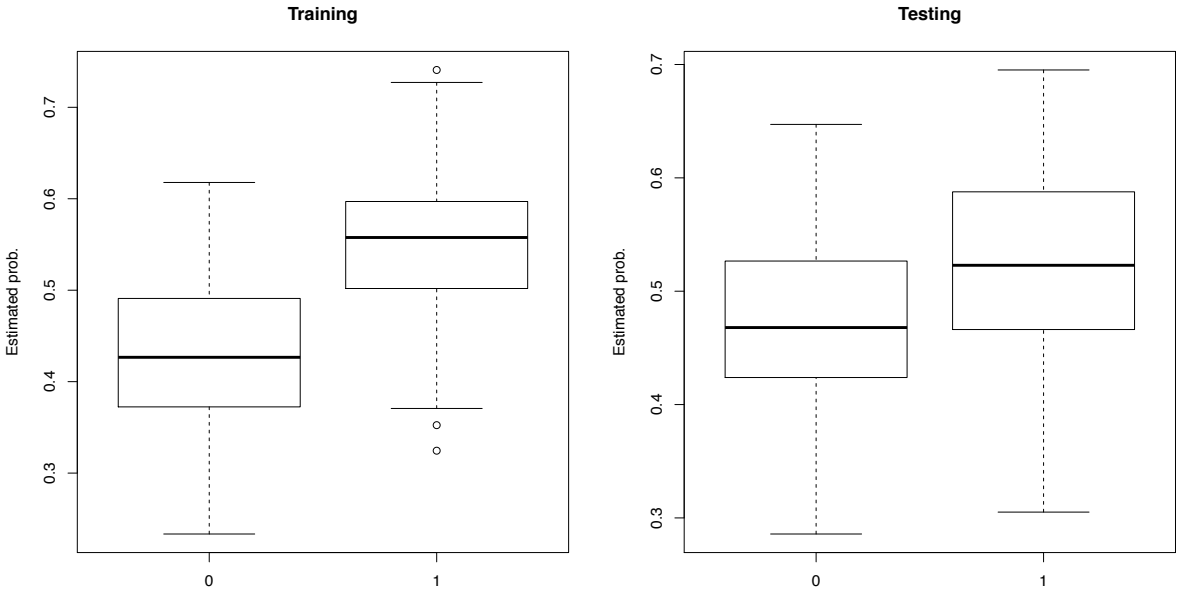

Figure 5: Estimated probability by category, versus observed category (binary response).

5.2. Regression with Censored Outcomes

Box 9 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 9), 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 9: 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]

#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])
```

6. 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=1,000) and four different marker densities (p=5K, 10K, 50K and 100K). The evaluation was carried out in an Intel(R) Xeon(R) processor @ 2 GHz. Computing time, expressed in seconds per thousand iterations of the Gibbs sampler are given in Figure 6. R was executed in a single thread and was linked against OpenBLAS. Computing 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 approximately 11 min to complete 1,000 iterations of the Gibbs sampler.

In general, the computational time of models BayesA and BL are slightly longer than that of BRR ($\sim$ 10% longer). The computational time of models using finite-mixture priors (e.g., models BayesB or BayesC) tend to be higher than those of BayesA, BL and BRR, unless the proportion of markers entering in the model is low.

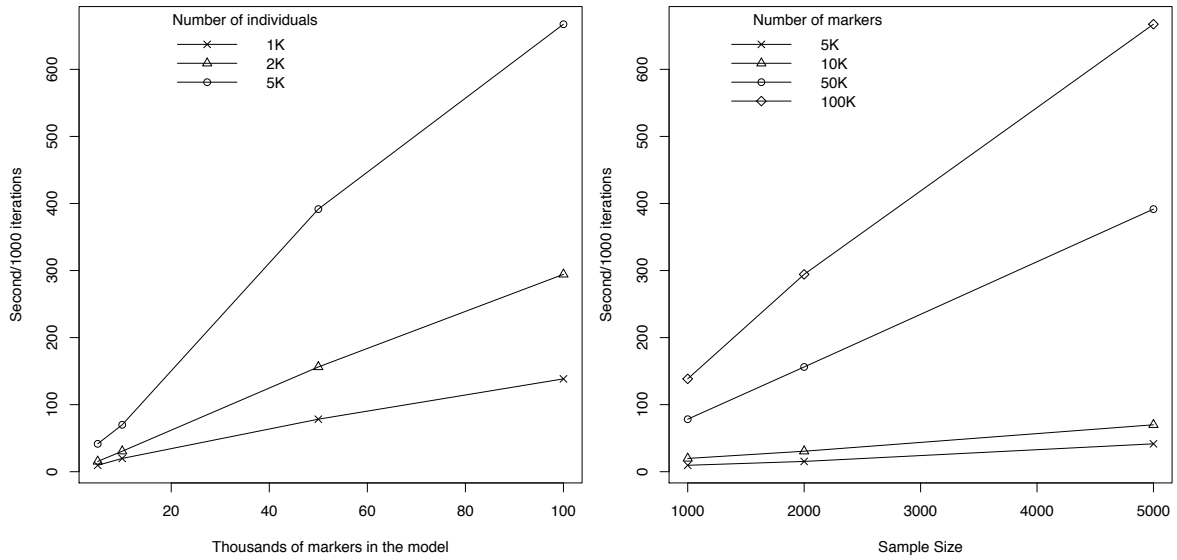

Figure 6: Seconds per 1000 iterations of the Gibbs sampler by number of markers and sample size. The Benchmark was carried out by fitting a Gaussian regression (BRR) using an Intel(R) Xeon(R) processor @ 2 GHz. Computations were carried out using a single thread.

7. Concluding Remarks

In **BGLR** we implemented, in a unified Bayesian framework, several methods commonly used in genome-enabled prediction, including various parametric models as well as Gaussian processes that can be used for parametric or semi-parametric regression/prediction. The package supports continuous (censored or not) as well as binary and ordinal traits. The user interface gives the user great latitude in combining different modeling approaches for data analysis. Operations that can be vectorized are performed using built-in R-functions, but 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 multi-thread 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 the R-environment genotype files written in ped and bed formats.

The Gibbs sampler implemented is computationally very intensive and our current implementation stores genotypes in memory; therefore, despite of the effort made in in the development of BGLR to make the algorithm 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, faster algorithms could be conceived, but these are in general not as flexible, in terms of the class of models that can be implemented, as the ones implemented in **BGLR**.

Future developments. Although some of the computationally intensive algorithms implemented in **BGLR** can benefit from multi-thread computing; there is large 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 so that genotypes do not need to be stored in memory. Future releases including these and other features 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.

Acknowledgements

In the development of **BGLR** Paulino Pérez and Gustavo de los Campos had financial support provided by NIH Grants: R01GM099992 and R01GM101219.

References

Albert JH, Chib S (1993). "Bayesian Analysis of Binary and Polychotomous Response Data." *Journal of the American Statistical Association*, **88**(422), 669–679.

Andrews DF, Mallows CL (1974). "Scale Mixtures of Normal Distributions." *Journal of the Royal Statistical Society. Series B (Methodological)*, **36**(1), 99–102. ISSN 00359246. doi:10.2307/2984774. URL http://dx.doi.org/10.2307/2984774.

Bates D, Vazquez AI (2009). *pedigreemm: Pedigree-based mixed-effects models*. R package version 0.2-4, URL http://CRAN.R-project.org/package=pedigreemm.

Bellman RE (1961). *Adaptive control processes - A Guided Tour*. Princeton University Press, Princeton, New Jersey, U.S.A.

Casella G, George EI (1992). "Explaining the Gibbs Sampler." *The American Statistician*, **46**(3), 167–174.

Cressie N (1988). "Spatial prediction and ordinary kriging." *Mathematical Geology*, **20**(4), 405–421. ISSN 0882-8121. doi:10.1007/BF00892986. URL http://dx.doi.org/10.1007/BF00892986.

Crossa J, de los Campos G, Pérez P, Gianola D, Burgueno J, Araus JL, Makumbi D, Singh RP, Dreisigacker S, Yan JB, Arief V, Banziger M, Braun HJ (2010). "Prediction of Genetic Values of Quantitative Traits in Plant Breeding Using Pedigree and Molecular Markers." *Genetics*, **186**(2), 713–U406.

de los Campos G, Gianola D, Rosa GJM (2009a). "Reproducing Kernel Hilbert Spaces Regression: A General Framework for Genetic Evaluation." *Journal of Animal Science*, **87**(6), 1883–1887.

de los Campos G, Gianola D, Rosa GJM, Weigel KA, Crossa J (2010). "Semi-parametric Genomic-enabled Prediction of Genetic Values Using Reproducing Kernel Hilbert Spaces Methods." *Genetics Research*, **92**, 295–308.

de los Campos G, Hickey JM, Pong-Wong R, Daetwyler HD, Calus MPL (2013a). "Whole Genome Regression and Prediction Methods Applied to Plant and Animal Breeding." *Genetics*, **193**, 327–345. doi:10.1534/genetics.112.143313.

de los Campos G, Naya H, Gianola D, Crossa J, Legarra A, Manfredi E, Weigel K, Cotes JM (2009b). "Predicting Quantitative Traits with Regression Models for Dense Molecular Markers and Pedigree." *Genetics*, **182**(1), 375–385.

de los Campos G, Pérez P (2010). "BLR: Bayesian Linear Regression R package, version 1.2." R package version 1.2.

de los Campos G, Pérez P (2013). "BGLR: Bayesian Generalized Regression R package, version 1.0." R package version 1.0, URL https://r-forge.r-project.org/projects/bglr/.

de los Campos G, Vazquez AI, Fernando RL, C KY, Daniel S (2013b). "Prediction of Complex Human Traits Using the Genomic Best Linear Unbiased Predictor." *PLoS Genetics*, **7**(7), e1003608. doi:10.1371/journal.pgen.1003608.

Endelman JB (2011). "Ridge regression and other kernels for genomic selection with R package rrBLUP." *Plant Genome*, **4**, 250–255.

Geman S, Geman D (1984). "Stochastic Relaxation, Gibbs Distributions and the Bayesian Restoration of Images." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **6**(6), 721–741.

Gianola D (2013). "Priors in Whole-Genome Regression: The Bayesian Alphabet Returns." *Genetics*, **90**, 525–540. ISSN 1469-5073.

Gianola D, de los Campos G (2008). "Inferring genetic values for quantitative traits non-parametrically." *Genetics Research*, **90**, 525–540. ISSN 1469-5073. doi:10.1017/S0016672308009890. URL http://journals.cambridge.org/article_S0016672308009890.

Gianola D, Fernando RL, Stella A (2006). "Genomic-assisted Prediction of Genetic Value with Semiparametric Procedures." *Genetics*, **173**(3), 1761–1776.

Gianola D, van Kaam JBCHM (2008). "Reproducing Kernel Hilbert Spaces Regression Methods for Genomic Assisted Prediction of Quantitative Traits." *Genetics*, **178**(4), 2289–2303.

González-Recio O, Gianola D, Long N, Weigel KA, Rosa GJM, Avendano S (2008). "Nonparametric Methods for Incorporating Genomic Information Into Genetic Evaluations: An Application to Mortality in Broilers." *Genetics*, **178**(4), 2305–2313. doi:10.1534/genetics.107.084293. http://www.genetics.org/content/178/4/2305.full.pdf+html, URL http://www.genetics.org/content/178/4/2305.abstract.

Habier D, Fernando R, Kizilkaya K, Garrick D (2011). "Extension of the Bayesian Alphabet for Genomic Selection." *BMC Bioinformatics*, **12**(1), 186.

Hayes B, Bowman P, Chamberlain A, Goddard M (2009). "Invited review: Genomic selection in dairy cattle: Progress and challenges." *Journal of Dairy Science*, **92**(2), 433 – 443. ISSN 0022-0302.

Henderson CR (1975). "Best linear Unbiased Estimation and Prediction Under a Selection Model." *Biometrics*, **31**(2), 423–447.

Hoerl AE, Kennard RW (1970). "Ridge Regression: Biased Estimation for Nonorthogonal Problems." *Technometrics*, **42**(1), 80–86.

Janss L, de los Campos G, Sheehan N, Sorensen D (2012). "Inferences from Genomic Models in Stratified Populations." *Genetics*, **192**(2), 693–704. doi:10.1534/genetics.112.141143. http://www.genetics.org/content/192/2/693.full.pdf+html, URL http://www.genetics.org/content/192/2/693.abstract.

Legarra A, Robert-Granié C, Manfredi E, Elsen JM (2008). "Performance of Genomic Selection in Mice." *Genetics*, **180**(1), 611–618.

Makowsky R, Pajewski NM, Klimentidis YC, Vazquez AI, Duarte CW, Allison DB, de los Campos G (2011). "Beyond Missing Heritability: Prediction of Complex Traits." *PLoS Genet*, **7**(4), e1002051.

Meuwissen THE, Hayes BJ, Goddard ME (2001). "Prediction of Total Genetic Value Using Genome-Wide Dense Marker Maps." *Genetics*, **157**(4), 1819–1829.

Okut H, Gianola D, Rosa GJM, Weigel KA (2011). "Prediction of Body Mass Index in Mice Using Dense Molecular Markers and a Regularized Neural Network." *Genetics Research*, **93**, 189–201.

Park T, Casella G (2008). "The Bayesian Lasso." *Journal of the American Statistical Association*, **103**(482), 681–686.

Pérez P, de los Campos G, Crossa J, Gianola D (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, Best N, Cowles K, Vines K (2006). "CODA: Convergence Diagnosis and Output Analysis for MCMC." *R News*, **6**(1), 7–11. URL http://CRAN.R-project.org/doc/Rnews/.

Purcell S, Neale B, Todd-Brown K, Thomas L, Ferreira MAR, Bender D, Maller J, Sklar P, de Bakker PIW, Daly MJ, Sham PC (2007). "PLINK: A Tool Set for Whole-Genome Association and Population-Based Linkage Analyses." *The American Journal of Human Genetics*, **81**, 559 – 575.

Quaas RL, Pollak EJ (1980). "Mixed Model Methodology for Farm and Ranch Beef Cattle Testing Programs." *Journal of Animal Science*, **51**(6), 1277–1287. http://www.journalofanimalscience.org/content/51/6/1277.full.pdf+html, URL http://www.journalofanimalscience.org/content/51/6/1277.short.

R Core Team (2012). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Spiegelhalter DJ, Best NG, Carlin BP, Van Der Linde A (2002). "Bayesian Measures of Model Complexity and Fit." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **64**(4), 583–639.

Tanner MA, Wong WH (1987). "The Calculation of Posterior Distributions by Data Augmentation." *Journal of the American Statistical Association*, **82**(398), 528–540.

Valdar W, Solberg LC, Gauguier D, Burnett S, Klenerman P, Cookson WO, Taylor MS, Rawlins JNP, Mott R, Flint J (2006a). "Genome-Wide Genetic Association of Complex Traits in Heterogeneous Stock Mice." *Nature Genetics*, **38**, 879–887.

Valdar W, Solberg LC, Gauguier D, Cookson WO, Rawlins JNP, Mott R, Flint J (2006b). "Genetic and Environmental Effects on Complex Traits in Mice." *Genetics*, **174**(2), 959–984.

VanRaden P, Tassell CV, Wiggans G, Sonstegard T, Schnabel R, Taylor J, Schenkel F (2009). "Invited Review: Reliability of genomic predictions for North American Holstein bulls." *Journal of Dairy Science*, **92**(1), 16 – 24. ISSN 0022-0302.

VanRaden PM (2008). "Efficient Methods to Compute Genomic Predictions." *Journal of Dairy Science*, **91**(11), 4414–23.

Vapnik V (1998). *Statistical learning theory*. 1 edition. Wiley. ISBN 0471030031.

Vazquez AI, Bates DM, Rosa GJM, Gianola D, Weigel KA (2010). "Technical Note: An R package for Fitting Generalized Linear Mixed Models in Animal Breeding." *Journal of Animal Science*, **88**(2), 497–504.

Vazquez AI, de los Campos G, Klimentidis YC, Rosa GJM, Gianola D, Yi N, Allison DB (2012). "A Comprehensive Genetic Approach for Improving Prediction of Skin Cancer Risk in Humans." *Genetics*, **192**(4), 1493–1502.

Wahba G (1990). *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611970128. http://epubs.siam.org/doi/pdf/10.1137/1.9781611970128, URL http://epubs.siam.org/doi/abs/10.1137/1.9781611970128.

Wimmer V, Albrecht T, Auinger HJ, Schoen CC (2012). "synbreed: a framework for the analysis of genomic prediction data using R." *Bioinformatics*, **28**(15), 2086–2087.

Yang J, Benyamin B, McEvoy BP, Gordon S, Henders AK, Nyholt DR, Madden PA, Heath AC, Martin NG, Montgomery GW, Goddard ME, Visscher PM (2010). "Common SNPs explain a large proportion of the heritability for human height." *Nature Genetics*, **42**(7), 565–569. doi:10.1038/ng.608. URL http://dx.doi.org/10.1038/ng.608.

Zhou X, Stephens M (2012). "Genome-wide efficient mixed-model analysis for association studies." *Nature Genetics*, **44**(7), 821–824. doi:10.1038/ng.2310. URL http://dx.doi.org/10.1038/ng.2310.

Appendices

1. Prior Densities Used in the BGLR R-Package

In this appendix 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 3a for an example). Table A1 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^2_{jk}, 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 2 of this appendix).

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"); 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.

2. 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 A1). 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.* (2013a) implemented using the prior mode instead of the prior mean.

Variance parameters. The residual variance (σ^2_ε, $\sigma^2_{u_l}$), of the RKHS model, and σ^2_β, 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

Table A1. 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`	$p(\boldsymbol{\beta}_j, \sigma^2_\beta) = \left\{\prod_k N(\beta_{jk}\|0, \sigma^2_\beta)\right\} \chi^{-2}(\sigma^2_\beta\|df_\beta, S_\beta)$	`list(X=, model="BRR",df0=,S0=,R2=)`
`BayesA`	$p(\boldsymbol{\beta}_j, \sigma^2_{\beta_j}, S_\beta) = \left\{\prod_k N(\beta_{jk}\|0, \sigma^2_{\beta_{jk}})\chi^{-2}(\sigma^2_{\beta_{jk}}\|df_\beta, S_\beta)\right\} G(S_\beta\|r, s)$	`list(X=, model="BayesA",df0=,rate0=, shape0=,R2=)`
`BL`	$p(\boldsymbol{\beta}_j, \boldsymbol{\tau}^2_j, \lambda^2\|\sigma^2_\varepsilon) = \left\{\prod_k N(\beta_{jk}\|0, \tau^2_{jk} \times \sigma^2_\varepsilon) Exp\left\{\tau^2_{jk}\|\frac{\lambda^2}{2}\right\}\right\} \times G(\lambda^2\|r, s)$, or	`list(X=,model="BL",lambda=,type="gamma", rate=,shape=,R2=)`[1]
	$p(\boldsymbol{\beta}_j, \boldsymbol{\tau}^2_j, \lambda\|\sigma^2_\varepsilon, \texttt{max}) = \left\{\prod_k N(\beta_{jk}\|0, \tau^2_{jk} \times \sigma^2_\varepsilon) Exp\left\{\tau^2_{jk}\|\frac{\lambda^2}{2}\right\}\right\} \times B(\lambda/\max\|p_0, \pi_0)$, or	`list(X=,model="BL",lambda=,type="beta", probIn=,counts=,max=,R2=)`[1]
	$p(\boldsymbol{\beta}_j, \boldsymbol{\tau}^2_j\|\sigma^2_\varepsilon, \lambda) = \left\{\prod_k N(\beta_{jk}\|0, \tau^2_{jk} \times \sigma^2_\varepsilon) Exp\left\{\tau^2_{jk}\|\frac{\lambda^2}{2}\right\}\right\}$	`list(X=,model="BL",lambda=,type="FIXED")`[1]
`BayesC`	$p(\boldsymbol{\beta}_j, \sigma^2_\beta, \pi) = \left\{\prod_k \left[\pi N(\beta_{jk}\|0, \sigma^2_\beta) + (1-\pi)1(\beta_{jk}=0)\right]\right\} \times\chi^{-2}(\sigma^2_\beta\|df_\beta, S_\beta) B(\pi\|p_0, \pi_0)$	`list(X=,model="BayesC",df0,S0, probIn=,counts=,R2=)`[2]
`BayesB`	$p(\boldsymbol{\beta}_j, \sigma^2_\beta, \pi) = \left\{\prod_k \left[\pi N(\beta_{jk}\|0, \sigma^2_\beta) + (1-\pi)1(\beta_{jk}=0)\right] \chi^{-2}(\sigma^2_{\beta_{jk}}\|df_\beta, S_\beta)\right\} B(\pi\|p_0, \pi_0) \times G(S_\beta\|r, s)$	`list(X=,model="BayesB",df0,rate0,shape0, probIn=,counts=,R2=)`[2]
`RKHS`	$p(\boldsymbol{u}_l, \sigma^2_{u_l}) = N(\boldsymbol{u}_l\|\boldsymbol{0}, \boldsymbol{K}_l \times \sigma^2_{u_l})\chi^{-2}(\sigma^2_{u_l}\|df_l, S_l)$	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.

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 to 1.1 and 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 $(s-1)/(2 \times (1-R2)/R2 \times MSx)$.

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 of models BayesA (in the case of BayesB) and BRR (in the case of BayesC), but the rules are modified by considering that 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 $S_\beta = var(y) \times R2 \times (df_\beta + 2)/MSx/\pi$.

Affiliation:

Paulino Pérez
Socio Economía Estadística e Informática
Colegio de Postgraduados, México
E-mail: perpdgo@colpos.mx

Gustavo de los Campos
Department of Biostatistics
Section on Statistical Genetics
University of Alabama at Birmingam
Telephone: +1/205/975-9248
Fax: +1 /205/975-2540
E-mail: gcampos@uab.edu
http://www.soph.uab.edu/ssg/people/campos