ABSTRACT. The BGLR R-package implements various types of single-trait shrinkage/variable selection Bayesian regressions. The package was first released in 2014, since then it has become a software very often used in genomic studies. We recently develop functionality for multi-trait models. The implementation allows users to include an arbitrary number of random effects terms. For each set of predictors, users can choose diffuse, Gaussian, and Gaussian-Spike-Slab multivariate priors. Unlike other software packages for multi-trait genomic regressions, BGLR offers many specifications for (co)variance parameters (unstructured, diagonal, Factor Analytic, and recursive). Samples from the posterior distribution of the models implemented in the Multitrait function are generated using a Gibbs sampler, which is implemented by combining code written in the R and C programming languages. In this manuscript we provide an overview of the models and methods implemented BGLR’s Multitrait function, present examples that illustrate the use of the package, and benchmark the performance of the software.
Keywords: Bayesian, high-dimensional regression, multivariate models, multi-trait models, Gibbs sampling, genomic regressions, pedigree regressions.

Introduction

High-dimensional regression problems are ubiquitous in many research areas. To confront the ‘curse’ of dimensionality, a common approach is to use regularized regression procedures such as penalized, reduced rank, or Bayesian methods. In genetics, the continued development of genotyping and sequencing technologies has led to the development of large volumes of genomic data. Genomic prediction (GP) models have been adopted in plant and animal breeding for Genomic Selection (Meuwissen, Hayes, and Goddard 2001) and are used for complex-trait prediction in human as well (de los Campos, Gianola, and Allison 2010; Yang et al. 2010). Several software packages offer implementations to fit Bayesian (de los Campos et al. 2013; Pérez and de los Campos 2014) and non-Bayesian (Friedman, Hastie, and Tibshirani 2010; Endelman 2011) models for single-trait genomic models.

In most genomic data set, genotypes are evaluated for multiple traits; likewise, in experimental settings genotypes are often evaluated in multiple environments. Thus, soon after genomic selection (GS) received attention, it become clear that extensions to multi-trait/multi-environment settings were needed (Burgueño et al. 2012). Several software packages implement multivariate mixed-effects Gaussian models using likelihood/REML methods (Gilmour, Thompson, and Cullis 1995; Meyer 2007; Lippert et al. 2014; Covarrubias-Pazaran 2016) or within a Bayesian setting (Hadfield 2010; Cheng, Fernando, and Garrick 2018; Montesinos-López et al. 2019). However, the extension of GP to multi-trait settings within the context of Bayesian variable selection models is more challenging; thus, the availability of multi-trait Bayesian variable selection models is much more limited. Within the open-source domain, we are only aware of one software package that implement multi-trait Bayesian variable selection and shrinkage methods for the Julia language (Cheng, Fernando, and Garrick 2018).

The BGLR-R package (Pérez and de los Campos 2014) was originally developed for single-trait models; recently, we developed the Multitrait function which implements shrinkage and variable
selection models for multi-trait analysis. The software combines features of the single-trait ‘BGLR’ function (a similar user-interface, and similar programming strategies, and similar default rules for setting hyper-parameters) with some of the features of the MTM R-package, a software that we developed and maintained in GitHub (https://github.com/QuantGen/MTM) since 2013 (de los Campos and Grueneberg 2013). MTM implements Bayesian Gaussian multivariate models with structured and un-structured (co)variance matrix using the methods described in de los Campos and Gianola (2007). However, MTM was fully developed using the R-language and thus it does not scale for problems involving large numbers of samples. Moreover, MTM does not implement variable selection methods.

In the Multitrait function of the BGLR-R package, users can specify models for an arbitrary number of random effects. For each random effect users can choose between diffuse, Gaussian, and spike-slab priors. For (co)variance parameters the software offers the possibility of using either structured (e.g., diagonal, factor analytic, or recursive structures) or completely unstructured specifications. Inputs for each of the random effects could be relationship matrices (aka kernels, e.g., pedigree- or SNP-derived genomic relationship matrices), eigenvectors and eigenvalue values from those matrices, or numeric matrices, which would be used to provide SNP genotypes, factorizations of relationship matrices (e.g., scaled eigenvectors or Cholesky factors), environmental covariates, and other quantitative predictors. Importantly, the trait matrix can have arbitrary patterns of missing values.

In what remains of this manuscript we describe the statistical models and the algorithms implemented in the Multitrait function, present examples that illustrate the use of the software, and provide an overview of the computational time required for analyses involving various sample sizes, number of predictors (e.g., number of SNPs) and number of traits.

**Materials and Methods**

The data-equation of the Multitrait function can include various types of effects, in matrix form, the data-equation can be represented as follows:

\[ Y = \mathbf{1} \mu' + \mathbf{X}_1 \mathbf{B}_1 + \mathbf{X}_2 \mathbf{B}_2 + \cdots + \mathbf{U}_1 + \mathbf{U}_2 + \cdots + \mathbf{E}, \]  

[1]
where, $\mathbf{Y}$ is an $n \times t$ matrix (individuals in rows, traits in columns) of phenotypes (missing values are allowed), $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_t)'$ is a vector of trait-specific intercepts, $\mathbf{X}_*$ is an incidence matrix for effects of a set of predictors (e.g., SNPs in columns) and $\mathbf{B}_*$ is a matrix of effects (predictors in rows, effects on each of the traits in columns), $\mathbf{U}_*$ is a matrix of random effects (individuals in rows, traits in columns), and $\mathbf{E}$ is a matrix with error terms (individuals in rows, traits in columns). The error terms are assumed to be Independent and Identically Distributed (IID), each row following a Multivariate Normal (MVN) distribution with zero mean and covariance matrix $\mathbf{R}_0$; therefore, the conditional distribution of the data given the regression parameters and the error covariance matrix is

$$p(\mathbf{Y}|\boldsymbol{\theta}) = \prod_{i=1}^n MVN(\mathbf{y}_i|\eta_i, \mathbf{R}_0),$$

where $\eta_i = \mathbf{x}_i' \beta_1 + \mathbf{x}_i' \beta_2 + \cdots + \mathbf{u}_i + \mathbf{e}_i$, where $\eta_i$ is a $t$-dimensional vector containing the conditional mean of the $i^{th}$ observation for each of the $t$-traits, $\mathbf{x}_i$ and $\mathbf{u}_i$ are the $i^{th}$ rows of the corresponding $\mathbf{X}_*$ and $\mathbf{U}_*$ matrices, and $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{R}_0\}$.

**Prior distribution:** The software assumes independent priors for each set of predictors; therefore,

$$p(\boldsymbol{\theta}) = p(\boldsymbol{\mu}) \times p(\mathbf{B}_1) \times p(\mathbf{B}_2) \times \cdots \times p(\mathbf{U}_1) \times p(\mathbf{U}_2) \times \cdots \times p(\mathbf{R}_0).$$

**Intercepts** are assigned diffuse priors $p(\boldsymbol{\mu}) = MVN(0, \mathbf{I}_m)$, with $m$ equal to 1,000 times the maximum of the variance of each of the phenotypes.

The random-effects matrices ($\mathbf{U}_*$) are each assigned MVN distributions of the form

$$p(\mathbf{u}_*) = MVN(\mathbf{u}_*|\mathbf{0}, \mathbf{\Omega}_*)$$

where $\mathbf{u}_* = \text{vec}(\mathbf{U}_*)$ is a vector with the columns of $\mathbf{U}_*$ stacked, $\mathbf{\Omega}_*$ is a $t \times t$ (unknown) covariance matrix, and $\mathbf{K}_*$ is a $n \times n$ covariance structure (e.g., a genomic relationship matrix), and $\otimes$ is the Kronecker product operator (more information about the prior on $\mathbf{\Omega}_*$ is provided below).

**Regression coefficients** $\mathbf{B}_*$ can be either assigned MVN priors or priors from the spike-slab family. In the MVN prior,

$$p(\mathbf{B}_*) = \prod_{j=1}^{p_*} MVN(\mathbf{b}_j|\mathbf{0}, \mathbf{\Omega}_*)$$

[5a]
where $b_{sj}$ is the $j^{th}$ row of $B_s$ and $\Omega_s$ is a t×t (unknown) covariance matrix describing covariance among the effects of the $j^{th}$ predictor on each of the traits.

For variable selection models, users can choose a multivariate spike-slab priors with a point of mass at zero and a MVN slab (this is one possible multivariate-extension of the model known as Bayes-C in the animal breeding literature, e.g. Habier et al. 2011). For the spike-slab prior, we represent each of the entries of $B_s = A_s \# D_s = \{b_{sjk} = \alpha_{sjk} \times d_{sjk}\}$ as the product of a normal random variable ($\alpha_{sjk}$) and a dummy variable ($d_{sjk}$) which controls whether the $j^{th}$ predictor enters in the equation for the $k^{th}$ trait; thus,

$$p(A_s, D_s) = \left\{ \prod_{j=1}^{P_s} \text{MVN}(\alpha_{sj} | 0, \Omega_s) \right\} \times \left\{ \prod_{j=1}^{P_s} \prod_{k=1}^{t} \pi_{sjk}^{d_{sjk}} (1 - \pi_{sjk})^{1-d_{sjk}} \right\}$$  \[5b\]

Above, $\{\pi_{s1}, \pi_{s2}, ..., \pi_{st}\}$ represent the proportion of non-zero effects in each of the traits. By default, each of these parameters are treated as unknown and they are assigned independent Beta priors, that is: $p(\pi_{s1}, \pi_{s2}, ..., \pi_{st}) \propto \prod_{k=1}^{t} \pi_{sk}^{\alpha_1} (1 - \pi_{sk})^{\alpha_2}$, where $\alpha_1$ and $\alpha_2$ are the prior shape1 and shape2 parameters for the $k^{th}$ trait.

Table 1 describes for each type of random effects ($X, B_s$ and $U_s$) the priors implemented and how to specify it.

Finally, for each of the covariance matrices involved in the model ($R_0$ and each of the $\Omega_s$) the software offers the option to modeled it as un-structured (in which case and Inverse-Wishart (IW) prior is used), or structured (with three options in this case, see Table 2). Covariance structures are specified using a list, the entry $\text{\$type$}$ of the list specifies the type of the structure (unstructured, diagonal, factor analytic structure, or recursive model, see Table 2), other arguments include the hyper-parameters for the structure (e.g. scale and degree of freedom parameters). In the case of recursive and FA structures, an entry labeled as $M$ (logical matrix) is also needed. For FA models this matrix must have dimensions $t \times$ number of factors and each cell (TRUE/FALSE) indicates whether the trait (row) loads on the corresponding factor (column). For recursive models the matrix $M$ has dimensions $t \times t$, and the cells (also TRUE/FALSE) indicate whether the trait in column enters for the equation of the trait in row (typically, the diagonal entries of $M$ will be FALSE, except for simultaneous equation models, but this requires
fixed-effects exclusion restrictions (e.g., Goldberger 1972). Fully recursive models (those with an \( M \) matrix with \texttt{TRUE} in all cells below the diagonal and \texttt{FALSE} everywhere else) lead to an \( UN \) covariance matrix.

\textbf{Default rules for hyper-parameters:} To simplify the use of the software we have implemented default rules to set these parameters; therefore, when the user does not specify one or several of the hyper-parameters involved in the model, the rules described in supplementary File S1 are used to set default values. The hyper-parameter values selected using these rules always lead to proper priors with scale parameters chosen considering the variance of the phenotype, the number of terms in the linear predictor, and the number of covariates in each of them.

\textit{Algorithms}

The \textit{joint prior} distribution of the model defined by expressions [2]-[5] does not have a closed-form; however, with the likelihood and prior-specifications used, all the fully-conditionals have closed form; therefore, the software generates samples from the joint posterior distribution using a Gibbs sampler (e.g., Casella and George 1992).

In our implementation, most of the model unknowns are updated sampling from univariate fully conditionals; the only exception are the un-structured covariance matrices for which a sample for the matrix is obtained from an Inverse Wishart fully conditional. Describing each of the fully conditionals in detail is beyond the scope of this manuscript; for location parameters (intercepts and regression coefficients), most of the fully-conditional distributions are standard and have been previously described for Gaussian multi-trait models with conjugate priors (e.g., Sorensen and Gianola 2002). The fully conditional distribution of the parameters involved in structured covariance matrices are borrowed from de los Campos and Gianola (2007), and from the implementation of the MTM software (de los Campos and Grueneberg 2013).

Sampling regression coefficients (and the dummy variables \( d_{*jk} \) in the case of the terms with spike-slab priors) is computationally the most demanding step. Therefore, we update all these unknowns using a routine implemented in the C programming language (Kernighan and Ritchie
1988). For hyper-parameters and for covariance matrices we collect samples using code implemented in the R-language (R Core Team 2019).

Missing values: The phenotype matrix (Y in [1]) can contain missing values (NA's); when this is the case, at each iteration of the sampler the missing values are “imputed” with samples from the fully conditional distribution (see Sorensen and Gianola 2002). Missing values in the linear predictor (e.g., X or K matrices) are not allowed.

User interface and outputs

The main arguments of the Multitrait function are the phenotype matrix (y: a matrix of dimensions n x t, where n is the number of individuals and t is the number of traits), the linear predictor (ETA: a two level list, each elements of the regression function [1], either a regression term of the form X, B, or random intercepts (U), see Table 1), and error covariance structure (resCov: A list that specifies the structure and the hyper-parameters for R_0). A complete list of arguments can be found in the help included in the package.

The software returns a list with the estimated posterior means, estimated posterior deviations, together with all the elements of model specification (priors and hyper-parameters). In the case of covariance matrices only lower-triangular elements are stored; the corresponding matrices can be recreated using the function ‘xpnd’, also included originally in the in the MCMCpack package (Martin, Quinn, and Park 2011).

Datasets

The BGLR R-package includes two data sets (these data sets were also included in earlier versions of the package). The wheat data set (data(wheat)) was first made available by Crossa et al. (2010). This data set contains least-square means (average of two years) of grain yield (wheat.Y) for 599 wheat lines was evaluated in 4 environments, a kinship matrix derived from a pedigree (wheat.A) and 1,279 Diversity Array Technology (DArT) markers (wheat.X, since these are inbreed lines, genotypes are coded as 0/1). The mice data set
(data(mice)) comes from mice experiments conducted by the Wellcome Trust (Valdar, Solberg, Gauguier, Burnett, et al. 2006; Valdar, Solberg, Gauguier, Cookson, et al. 2006). The phenotypes consists of traits related to diabetes and biochemistry for 1814 mice (mice.pheno), a kinship matrix derived from pedigree (mice.A) and 10,346 SNP markers (mice.X).

Results

In this section we introduce examples of various models that can be implemented with the Multitrait function. We present in the manuscript snippets with concise scripts that illustrate how to fit models and how to retrieve results. The complete set of scripts used to simulate and analyze data are provided in an R-Markdown as a Supplementary Material and are also available at BGLR’s GitHub repository.

Gaussian (kernel) models

The example in Box 1 illustrates how to fit a single-kernel multi-trait (GBLUP) model with unstructured (co)variance matrices. Lines 13-18 show how to retrieve parameter estimates and the corresponding posterior SD. The code in line 20 shows how to retrieve the predicted genomic values. Samples from the posterior distribution of model parameters can be found in files UN_mu.dat (intercepts), UN_R.dat (residual covariance matrix) and UN_Omega_1.dat (genomic covariance matrix). Supplementary File S2 (Scripts) Box S1 provides scripts that illustrate how to create trace plots and posterior samples for genetic and residual dispersion parameters and the resulting plots are shown in supplementary File S3, Figures S1 and S2.

Box 1: Multi-trait GBLUP with unstructured and structured covariances

```
library(BGLR)
data(wheat)
K<-tcrossprod(scale(wheat.X,center=TRUE))
K<-K/mean(diag(K))
Y<-wheat.Y # 4 traits
# Fitting a GBLUP un-structured cov-matrices
LP<-list(mar=list(K=K,model="RKHS"))
set.seed(123)
```
By default, the software fits un-structured (co)variance matrices; the examples in Box 2 illustrate how to specify structured covariance matrices. The first model (fmRD) uses a fully recursive model for the genomic covariance matrix and a diagonal structure for the error terms. In the recursive structure, there are regressions of traits 3 and 4 on trait 2 and of trait 4 on trait 3. This is specified with the matrix $M$ (logical) which is used to specify recursions between traits. In the example $M$ is specified in a way that makes the genetic values of trait 1 independent from the genetic values on the other 3 traits (i.e., the resulting structure is block-diagonal). The second example uses a one-factor model for the genomic covariance matrix and a diagonal structure for the error terms (fmFD). The parameter $M$, which in this case links traits with common factors, links traits 2, 3 and 4 with a common factor; thus, also leading to a block diagonal structure. In models with structured covariance matrices the covariance matrices are non-linear functions of the model parameters. For instance, in a GBLUP model with a FA structure for the genetic covariance, $\Omega = WW' + \Psi$; likewise, for a recursive model the covariance matrix is $\Omega = (I - W)^{-1}\Psi((I - W)^{-1})'$ (see Table 2). At each iteration of the sampler the software computes these covariances and returns not only the posterior means of the parameters involved in the covariance structure ($W, \Psi$ in the preceding examples), but also the posterior means of the corresponding covariance matrix ($\Omega$) (see lines 17-21 and 34-38). Boxes S2a-S2b in the supplementary File S2 (scripts) provide code that can be used to produce posterior plots and summaries of the parameters of recursive and FA covariance structures, the resulting plots are shown in supplementary File S3, Figures S3 through S5. In genetic models, upon appropriate scaling, parameters $\Omega$ and $R_0$ can be interpreted as genetic and environmental (co)variance components. However, we will discuss later in the document approaches drawn from Lehermeier et al. (2017) which account for linkage-disequilibrium and yield estimates which often have smaller biases than those of the posterior means of $\Omega$ and $R_0$. 

```
10 fmUN<-Multitrait(y=Y,ETA=LP,nIter=10000,burnIn=5000,
11         saveAt="UN_",verbose=FALSE)
12 # Retrieving estimates and posterior SD
13 fmUN$resCov$R          # residual co-variance matrix
14 fmUN$resCov$SD.R
15 fmUN$ETA$mar$Cov$Omega # genomic covariance matrix
16 fmUN$ETA$mar$Cov$SD.Omega
17 fmUN$ETA$mar$u         # predicted random effects
```
Box 2: Multi-trait GBLUP with structured covariance matrices

```r
#(continued from Box 1)

# Omega-recursive, R-diagonal
# Matrix specifying loadings among traits 2=>3,2=>4,3=>4
M1<-matrix(nrow = 4, ncol = 4, FALSE)
CovREC<-list(type="REC", M=M1)
LP<-list(mar=list(K=K, model="RKHS", Cov=CovREC))

CovDIAG<-list(type="DIAG")

set.seed(456)
fmRD<-Multitrait(y=Y, ETA=LP, nIter=10000, burnIn=5000,
resCov=CovDIAG, saveAt="REC_DIAG_",
verbose=FALSE)

fmRD$resCov$R
fmRD$ETA$mar$Cov$Omega  # genomic covariance
fmRD$ETA$mar$Cov$W      # recursive effects
fmRD$ETA$mar$u          # predicted genetic effects
fmRD$ETA$mar$Cov$PSI    # scaling factors

# Omega-FA(2), R-diagonal
M2<-matrix(nrow=4,ncol=1,FALSE)
M2[2:4,1]<-TRUE
CovFA<-list(type="FA", M=M2)
LP<-list(mar=list(K=K, model="RKHS", Cov=CovFA))

set.seed(789)
fmFAD<-Multitrait(y=Y, ETA=LP, nIter=10000, burnIn=5000,
resCov=CovDIAG, saveAt="FA_DIAG_",
verbose=FALSE)

fmFAD$resCov$R
fmFAD$ETA$mar$Cov$Omega  # genomic covariance
fmFAD$ETA$mar$Cov$W      # recursive effects
fmFAD$ETA$mar$u          # predicted genetic effects
fmFAD$ETA$mar$Cov$PSI    # scaling factors
```

Multi-kernel models: The Gaussian models described above can be extended by adding fixed
effects and other random effects (e.g., pedigree + SNPs models). The scripts in Boxes S3a-S3b in
the supplementary File S2 (scripts) implements a model with a genomic and a pedigree
relationship matrix for the wheat data set, the script implements the un-structured and structured
(diagonal, recursive, and FA) covariance matrices.
Bayesian variable selection models

We turn now into models using priors that can perform variable selection. To illustrate the use of these models we simulated 3 traits using the genomes available in the mice data set (data(mice)). The script used to simulate data is presented in Box S4a in the supplementary File S2. Briefly, the simulation has a sample size of \( n = 1814 \), and 1000 SNPs, 12 of which had non-zero effect. Each trait had 6 QTL. Traits 1 and trait 3 had 3 ‘private’ QTL and 3 shared with trait 2. All the QTL affecting trait 2 were shared with either trait 1 or trait 2. The trait heritabilities were 0.1, 0.05, and 0.1 for traits 1, 2, and 3, respectively. For illustration purposes we present results from a single Monte Carlo replicate; we emphasize that assessing mapping power and FDR would require running multiple replicates and more complex simulation settings.

Box 3 shows how to fit a Multi-trait Bayesian regression model with a Spike-Slab prior. As before, \( Y \) is a matrix with traits in columns and subjects in rows, \( X \) is a matrix with subjects in rows and SNPs in columns. For regression models, the Multitrait function supports three priors (see Table 1). The argument saveEffects=TRUE can be used to save the samples of effects into a binary file.

**Box 3: Fitting a Multitrait Spike Slab model**

```r
# Run the code in Box S4a before running this code
fmSS<-Multitrait(y=Y, ETA=list(list(X=X,model='SpikeSlab',
            saveEffects=TRUE)),nIter=12000,burnIn=2000,
            verbose=FALSE)
```

Figure 1 shows the posterior probability of inclusion by SNP and trait (the code used to produce the plot are given in Box S4b in the supplementary File S2). The model correctly identified 8 of the 12 QTL. Trait 2 had the lowest heritability and the lowest proportion of QTL detected. Some QTL are not individually detected because of high LD in the QTL region. This happened, for example, for QTL2 which was located (position 99) in a region with high LD. In this region there were three SNPs (position 99, 100, and 101) with elevated probability of inclusion in the QTL region. The R-squared between the causal variant and SNP 100 and 101 are 0.999 and 0.937. Clearly, when SNPs are in high LD, variable selection algorithms cannot discern which
SNP is associated with the phenotype—this also happens in single-trait models. When this situation happens, there are at least two possible remedies: (i) filter SNPs based on LD, or (ii) use all the SNPs and make inferences using credible sets (i.e., groups of SNPs). For example, in the region in question, (positions 99-101) the joint posterior probability of inclusion (i.e., the proportion of times that at least one of the SNPs in the region was active) was 99.75 (see code in Box S4c in the supplementary File S2). Thus, for this region we can be confident that there is an association with the phenotype, but we are uncertain as to which SNP has an effect.

A more general approach to identify credible sets is demonstrated in Box S4d (supplementary File S2). The function ‘segments’ of the BGData R-package (Grueneberg and de los Campos 2019) is used to identify segments with local FDR smaller than 0.7 (that is SNP-inclusion probability greater than 0.3), then we used the samples from the posterior distribution to estimate the joint probability of inclusion of each segment (i.e., the probability that at least one SNP in the segment is active in the model). The result is a table with discoveries which includes both individual SNP and credible sets and their posterior probability of inclusion.

For comparison with the multi-trait model, we fitted four separate single-trait models using the BayesC implementation available in BGLR function (see supplementary File S2, Box S4e). The results are presented in Figure S6 (supplementary File S3). The single-trait analysis identified two QTL less than the corresponding multi-trait model. In particular, the single trait model did not identify (from left to right) the 3rd QTL of trait one and the first QTL of trait two; these last one was shared between trait 1 and 2. This example illustrates some potential benefits of using a multi-trait model for mapping variants with pleiotropic effects; however, as we will see below, in some cases single-trait models may be more powerful.

The previous example used a stylized simulation, to illustrate the application of multi-trait variable selection models in a more complex scenario, we used the function Multitrait to map risk loci for six blood biomarkers (glucose, total cholesterol, HDL and LDL cholesterol, creatinine, and triglycerides) using the mice data set. The results are presented Figure S7, and Table S1 (supplementary File S3). In this case, the combination of small effects, small sample size, and high LD leads to no individual variants with a probability of inclusion greater than 0.8.
However, for each of the biomarkers there are multiple regions with clearly elevated inclusion probabilities. Using the function ‘segments’ of the BGData R-package (Grueneberg and de los Campos 2019) we identified 25 regions with joint posterior probability greater than 0.8 (Figure S7, File S3). We identified five of such regions for glucose and five for HDL cholesterol, nine regions for LDL cholesterol, one for creatinine, three regions for triglycerides, and two regions for total cholesterol. Some of the regions identified for different traits overlap (e.g., chromosome 1, mbp 89-98, chromosome 4, mbp 62-67, and chromosome 12, mbp 0-3) suggesting pleiotropy. For completeness, we also analyzed the same traits using single trait models (supplementary Figure S8, File S3); most of the regions with elevated posterior probability of inclusion detected using with the multitrait model also showed elevated posterior probability of inclusion in the single-trait models.

We repeated the analysis of mice blood biomarkers using a sequence of five single-trait analyses using BGLR with a variable selection prior (BayesC) prior. Interestingly, the number of regions that cleared a joint inclusion probability of 0.8 was similar for glucose but more significant for the other traits (Table S2 and Figure S8). Overall, there was a substantial overlap in findings between the single-trait and multi-trait analyses. However, many regions identified by the single-trait models were not identified by the multi-trait analysis. Conversely, the single-trait model did not identify a few regions discovered through a multi-trait analysis (Table S3). Because these analyses are based on real-data, we cannot discern which discoveries were true and which ones are false discoveries. However, the example illustrates that multi-trait analyses, which borrow information between traits, may not always lead to a higher power and more discoveries.

**Estimation of Variance Components**

Dissecting the phenotypic (co)variance into genetic and non-genetic components is an important goal of many genetic studies. Maximum Likelihood, REML, and Bayesian estimates of (co)variance parameters are not guaranteed to be unbiased. In our experience, environmental (co)variances are often estimated with relatively small biases using the estimated posterior mean of $\mathbf{R}_0$. However, this is not always the case for genetic (co)variances. There is an extensive literature on biases that may emerge when using genomic models to estimate genetic variances (e.g., de los Campos, Sorensen, and Gianola 2015; Krishna Kumar et al. 2016; Lehermeier et al.
The amount of information at the likelihood about co-variances is often smaller than the information provided for variances; therefore, presumably, there may be also sizable biases in estimate of genetic co-variances.

The literature on genetic (co)variance estimates is largely focused on GBLUP-type models; however, at least in principle, those parameters should be estimable in variable selection models. In this section we present a simulation study in which we assess the bias of (co)variance estimates derived from multi-trait-GBLUP and Spike Slab models. For residual covariance matrix we use as estimator the posterior mean of $\mathbf{R}_0$.

For estimating genetic covariances in models not involving variable selection (e.g., “RKHS” or “BRR”) we consider two estimation methods: the posterior mean of the (co)variance parameter ($\Omega$, Method 1), and a second method (Method 2) based on an approach described in Lehermeier et al. (2017) which account for covariances between predictors (e.g., LD between SNPs). Briefly, for Method 2, at each iteration of the sampler we compute for each trait the vectors of genomic values (i.e., the columns of $\mathbf{U}$ in GBLUP models or in regression on SNPs, $\mathbf{U} = \mathbf{X}\mathbf{B}$) and compute the sample covariance between the columns of $\mathbf{U}$.

In models involving variable selection (“SpikeSlab”) only a fraction of the SNPs are active in the equation of each trait. Therefore, the (co)variance matrix of effects, $\text{Cov}(\mathbf{a}) = \Omega$, cannot be directly used to estimate the contribution of the regression. Therefore, to estimate genetic covariances in these models we consider three methods: Method 1, at each iteration of the Gibbs sampler we compute $\mathbf{B} = \mathbf{A}\#\mathbf{D}$ and then compute the sample covariance between the columns $\mathbf{B}$, the estimate is then the posterior mean of $\text{cov}(\mathbf{B})$. For Method 2 at each iteration of the sample we computed genomic values ($\mathbf{U} = \mathbf{X}\mathbf{B}$) and form there we computed the sample variance-covariance matrix of the realized genomic values ($\text{var}(\mathbf{U})$). This approach, which accounts for LD between markers, extends the method described for single-trait models by Lehermeier et al. 2017 to multitrait analyses. Finally, we consider a 3rd method which was proposed by Cheng et al. (2018). In Method 3, at each iteration of the Gibbs sampler we compute the following variance-covariance parameters:
\[
\begin{align*}
G_{0kk} &= c \times \Omega_{kk} \times p_k & k = 1, \ldots, t \\
G_{0kk'} &= c \times \Omega_{kk'} \times p_{kk'} & k, k' = 1, \ldots, t; k \neq k'
\end{align*}
\]

where \( c \) is the sum of the variances of the markers \( (c = \sum_{j=1}^{p} \text{Var}(x_j)) \), \( p_k \) is the proportion of markers with non-null effect on trait \( k \), and \( p_{kk'} \) the proportion of markers with non-null effects for traits \( k \) and \( k' \).

We evaluated the bias of each of the estimation methods above described using 200 MC replicates of the simulation described in the previous section (see Box S4a for the simulation code). The code for estimating genetic covariance matrixes is given in Box S5 in the supplementary File S2.

The error (co)variance parameters were estimated without any noticeable bias by both the GBLUP and Spike Slab model (Figure 2, panel A). The genetic variances were estimated with small biases by the GBLUP, with Method 2 having slightly smaller bias than Method 1 (Figure 2 panel B, see Table S4 (see supplementary File S3) for average estimates and SEs). The Spike Slab model gave estimates with noticeable bias with Methods 1 and 3; however, these biases were reduced when Method 2 was used. In summary, at least for this specific data set and simulation setting, it seems that regardless of the prior used, the residual (co)variance parameters can be estimated with small biases, and estimation of genetic (co)variances seem to be better with Method 2.

**Prediction**

For the entries of the phenotype matrix containing missing values, at each iteration of the sampler, the software samples the missing phenotypes from the corresponding fully conditional distributions (which in the models implemented is multivariate normal). This process leads to inferences that average over possible values of the missing phenotypes. To make the sampling of missing phenotypes more efficient we identify missing-value-patterns and impute them (with samples) jointly (missing_patterns contain the patterns identified and labeled missing_records indicate the rows of Y which map to each of the patterns).
The Multitrait function returns estimates of the posterior mean of the linear predictor \((\eta = \mathbf{1}\mu' + \mathbf{X}_1\mathbf{B}_1 + \mathbf{X}_2\mathbf{B}_2 + \cdots + \mathbf{U}_1 + \mathbf{U}_2 + \cdots + \mathbf{U}_k + \mathbf{U}_k + \cdots + \mathbf{U}_n \), named \(\text{ETAHat} \) in the returned object), these are estimates of the expected value of the phenotypes given the predictors. Box 4 shows how to retrieve these predictions (the missing value patterns in \(Y_{Na} \) were generated using the script in Box S6, File S2). The plot in Figure 3 shows the simulated phenotype versus the posterior mean of for entries that were observed and missing, using trait 3 as an example (see Box S6, in the supplementary File S2).

**Box 4: Prediction**

```r
K <- tcrossprod(X)
K <- K / mean(diag(K))
LP <- list(list(K = K, model = "RKHS"))
# Fit multivariate GBLUP with UN-structured covariance matrixes
fmG <- Multitrait(y = YNa, ETA = LP, nIter = 10000, burnIn = 5000, thin = 10,
                   verbose = FALSE)
# Missing values for trait 3
whichNa3 <- fmG$missing_records[fmG$patterns[,3]]
Y[whichNa3, 3] # Observed values
fmG$ETAHat[whichNa3, 3] # Predicted values
```

**Benchmark**

We evaluated the computational time it takes for the Multitrait function to generate 1,000 posterior samples for data sets with a sample size ranging from 10,000 to 50,000 individuals and with 5,000 to 50,000 SNPs. The data used for this benchmark was from the UK-Biobank; we fitted models to two and four traits (body mass index total cholesterol, glucose, and height, all adjusted by sex, age, center, and 10 SNP-derived PCs) using either Gaussian or Spike-Slab priors. The analyses were performed in MSU’s high performance computing cluster using nodes equipped with Intel Xeon Gold 6148 processors @ 2.40 GHz. We used R 4.2.0 compiled from source and the reference BLAS implementation was substituted with OpenBLAS version 0.3.17 (https://www.openblas.net). Jobs were run using four computing threads. To benchmark our software, we compared the computational performance of BGLR’s Multitrait function with that of Julia’s JWAS package (Cheng, Fernando, and Garrick 2018), which also uses OpenBLAS for...
matrix operations. We provide the scripts to produce the benchmarks in the case of a Gaussian prior in the Supplemental File S2, Boxes S7 and S8.

Figure 4 shows the time (in minutes) required to generate 1,000 Gibbs samples for models using variable selection priors. We found that JWAS performed slightly better using just one thread; therefore, the results in Figure 4 are for Multitrait running with four threads and JWAS running with one thread (for results with both software using four threads see Figure S9, File S3). Both software scaled approximately linearly with sample size, but the slope of the lines representing computational time versus sample size were smaller for the Multitrait function. Likewise, while in both packages the computational time increased with the number of traits, the increase in computational time that occurred when increasing the number of traits from two to four was more important for JWAS. The results for models using a Gaussian prior are presented in Figures S10 and S11 (File S3). The overall trends were very similar to those obtained with the variable selection priors (Figure 4).

The computational time required for implementing a GBLUP (or any other kernel model) depends on sample size, this without considering the time it takes to compute the genomic relationship matrix. The BGLR and Multitrait functions fit GBLUP model by first factorizing kernel matrices using an eigenvalue decomposition; then, the software runs a random regression (with Gaussian IID priors) on the scaled eigenvectors (de los Campos, Gianola, and Rosa 2009). Therefore, the computational time it takes to collect samples for kernel models is the same as the computing time it takes for a Gaussian model with a sample size of \( n \) and \( n \) SNPs (which can be inferred from the results just presented), plus the time it takes to factorize the genomic relationship matrix. We present in Figure S12 (File S3) the computational time required to perform eigenvalue decomposition. The computational time increases exponentially with the dimensions of the kernel matrix; for 30,000 subjects it took approximately 25 minutes to compute the eigenvalue decomposition. In general, the option of using a genomic relationship matrix is recommended (relative to an equivalent model implemented as a random regression on SNPs) only when \( n \) is not too large, and sample size is considerably smaller than the number of SNPs.
We presented the newly developed Multitrait function of the BGLR R-package. The software allows the user to include fixed effects (i.e., effects with a diffuse prior), shrinkage, and variable selection priors in multi-trait models. These methods can be used to fit models commonly used in genomic analysis of complex traits, including parametric genomic and pedigree models, variable selection models, and semi-parametric regressions (e.g., RKHS). The interface allows users to specify models with multiple terms, each having their own priors.

The architecture of the software was designed to take advantage of interpreted code written in R and compiled code written in C. The MCMC algorithms (Metropolis and Gibbs sampler) implemented are computationally intensive. We make an effort to optimize computations without losing much flexibility in terms of the types of models that users can implement. As a result, the software can be used with a very large sample size, large numbers of SNPs, and multiple traits. However, users must be aware that the computational time grows linearly with sample size and with the number of SNPs; thus, fitting models to very large data sets and hundreds of thousands of SNPs will have a very high memory requirement and it will be time consuming. The programming strategy we adopted makes intensive use of level-1 BLAS functions ‘ddot’ and ‘daxpy’. For large problems, we strongly recommend using an optimized version of BLAS for the hardware and running the software on multiple threads.

The possibility of specifying structured (co)variance matrices for both Gaussian and variable selection priors is a distinctive feature of Multitrait. Users should be aware that priors on (co)variance matrices can be influential; this is particularly the case of the Inverse Wishart prior which can be highly influential if sample size is small, and the number of traits is large. To remediate this problem, we described in this study an approach (referred as to Method 2 in Figure 2) for estimating the variances and co-variances that are less influenced by the prior; however, even with this approach, the Inverse Wishart prior can be influential. This method, which extend the ideas discussed in Lehermeier et al. (2017) to multi-trait models, accounts for the contribution of linkage disequilibrium to (co)variance and, importantly, can be used with any prior (including diffuse, Gaussian, and variable selection priors). Another approach that may be
less influential is to use a fully recursive model which also leads to an un-structured (co)variance matrix. In this case the prior specification involves defining priors over variances and over recursive effects. The priors on the recursive effects (which in the Multitrait are by default Gaussian) can be made effectively diffuse and this may lead to less influence of the prior on inferences.

Although the software was originally developed for genomic and pedigree model, like BGLR, the Multitrait function can also be used to implement models for other omics (including metabolomic, microbiome, and gene-expression data) and environmental information (e.g., models for environmental covariates). We look forward to continuing improving BGLR, and welcome users’ feedback. Future updates will be shared in BGLR’s GitHub repository (https://github.com/gdlc/BGLR-R) and in the CRAN (https://cran.r-project.org/web/packages/BGLR/index.html).

The reader may wonder: For what problems, or under what conditions, is a multi-trait analysis preferred relative to a sequence of single-trait analyses? Of course, the answer to that question is problem-specific; however, some guidelines can be provided for parameter estimation, mapping, and prediction problems. Obviously, multi-trait analyses are needed to estimate genetic and environmental correlations between traits (or environments).

Depending on the problem, the multi-trait model may offer higher or lower power than the single-trait analysis. Standard multi-trait models assume a homogeneous covariance of effects across the genome. In regions of the genome where the correlation of effects aligns with the average correlation of effects of the genome, borrowing information between traits may increase power. However, in regions where the correlation of effects is markedly different than the average correlation of effects in the genome, the multi-trait model may reduce power because multi-trait models tend to shrink effects towards a common covariance pattern. Variable selection priors may mitigate this problem because the model, as implemented in the Multitrait function, has separate variable selection dummy variables for each SNP-trait combination. This allows for an SNP to affect some traits but not others. However, even considering the flexibility
that variable selection priors offer for mapping, we strongly encourage considering both the multi-trait and the single-trait models.

For genomic prediction, there is evidence showing that for some prediction problems (Burgueño et al. 2012), the multi-trait model can offer higher prediction accuracy than predictions derived from single-trait models. The superiority of the multi-trait model over single-trait analyses occurs when the prediction problem enables borrowing of information within-subject between traits and environments. This happens, for example, when predicting genetic values for an unmeasured trait for individuals with records for other traits (or environments, this was labeled as CV2 in Burgueño et al. 2012). On the other hand, for the prediction of genetic values for subjects that do not have any records (labeled as CV1 in Burgueño et al. 2012), multi-trait models do not consistently outperform predictions derived from single-trait models.

Data availability

The datasets are included with the BGLR package which is freely available in the CRAN website, https://cran.r-project.org/web/packages/BGLR/index.html and also in our GitHub website, https://github.com/gdlc/BGLR-R.

Funding

The development and maintenance of BGLR has been founded by NIH grant GM R01 101219. The authors also acknowledge financial support from Michigan State University.

References


Heritability for Human Height.” *Nature Genetics* 42 (7): 565–69. https://doi.org/10.1038/ng.608.
Table 1. Specification of regression terms in the linear predictor.

<table>
<thead>
<tr>
<th>Component</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression functions ($X, B, \cdots$)</td>
<td>Diffuse prior: <code>list(X=,model=&quot;FIXED&quot;)</code></td>
</tr>
<tr>
<td></td>
<td>Gaussian: <code>list(X=,model=&quot;BRR&quot;,Cov=)</code></td>
</tr>
<tr>
<td></td>
<td>Spike-Slab: <code>list(X=,model=&quot;SpikeSlab&quot;, Cov=,inclusionProb=)</code></td>
</tr>
<tr>
<td>Random-intercept matrices ($U, \cdots$)</td>
<td><code>list(K=,model=&quot;RKHS&quot;,Cov=)</code>, or</td>
</tr>
<tr>
<td></td>
<td><code>list(EVD=,model=&quot;RKHS&quot;,Cov=)</code></td>
</tr>
<tr>
<td></td>
<td>Above, $EVD$ is a list that contains two components,</td>
</tr>
<tr>
<td></td>
<td>vectors a square matrix with eigen-vectors of $K$ and</td>
</tr>
<tr>
<td></td>
<td>values a vector containing the eigen-values of $K$</td>
</tr>
</tbody>
</table>
Table 2. Covariance-structures.

<table>
<thead>
<tr>
<th>Structure</th>
<th>Prior</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unstructured</td>
<td>$\Omega \sim IW(\Omega</td>
<td>S_0, df_0)$;</td>
</tr>
<tr>
<td></td>
<td>$S_0$: prior scale matrix $(t \times t)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$df_0$: prior degree of freedom</td>
<td></td>
</tr>
<tr>
<td>Diagonal</td>
<td>$\Omega = \text{diag}(\omega_{11}, ..., \omega_{tt})$, $p(\omega) = \prod_k^t \chi^{-2}(\omega_{kk}</td>
<td>S_{0kk}, df_{0k})$</td>
</tr>
<tr>
<td></td>
<td>$S_0$: prior scale vector $(t \times 1)$, $df_0$: prior $df$ vector $(t \times 1)$</td>
<td></td>
</tr>
<tr>
<td>Factor analytic</td>
<td>$\Omega = \mathbf{W}\mathbf{W}' + \Psi$; $\mathbf{W}$: factor loadings, $\Psi = \text{diag}(\psi_{kk})$, $p(\mathbf{W}, \Psi) = \text{MVN}(\text{vec}(\mathbf{W})</td>
<td>0, \sigma^2 \mathbf{I}) \times \prod_k^t \chi^{-2}(\psi_{kk}</td>
</tr>
<tr>
<td></td>
<td>$S_0$: prior scale vector $(t \times 1)$, $df_0$: prior $df$ vector $(t \times 1)$; $\sigma^2$: prior variance</td>
<td></td>
</tr>
<tr>
<td>Recursive</td>
<td>$\Omega = (\mathbf{I} - \mathbf{W})^{-1} \Psi ((\mathbf{I} - \mathbf{W})^{-1})'$, $\mathbf{W}<em>{txt}$; $\Psi = \text{diag}(\psi</em>{kk})$, $p(\mathbf{W}, \Psi) = \text{MVN}(\text{vec}(\mathbf{W})</td>
<td>0, \sigma^2 \mathbf{I}) \times \prod_k^t \chi^{-2}(\psi_{kk}</td>
</tr>
<tr>
<td></td>
<td>$S_0$: prior scale vector $(t \times 1)$, $df_0$: prior $df$ vector $(t \times 1)$; $\sigma^2$: prior variance.</td>
<td></td>
</tr>
</tbody>
</table>
**Figure 1:** Posterior probability of non-zero effect by SNP and trait. Each dot/circle, represents a SNP, red-circles had posterior probability of inclusion greater than 0.8; the dashed vertical lines show the position of the causal variants.

**Figure 2:** Average Monte Carlo Estimates of Error (A) and Genetic (B) (co)variance parameter, by model and estimation method. The horizontal line gives the true parameter value (for SEs see Table S4).

**Figure 3:** Predicted (posterior mean of the linear predictor) versus observed phenotypes for trait 3 (see Box S6, File S2).

**Figure 4:** Average time (in minutes, +/- SD) to collect 1000 posterior samples for variable selection models, by the number of SNPs (panels), the number of traits (horizontal axis), and the software used. Results for Gaussian prior are presented in Figures S10 and S11 (File S3).