BLRXy: A function for biobank size data analysis

Paulino Pérez-Rodríguez
Colegio de Postgraduados, México

Gustavo de los Campos
Michigan State University, USA

Abstract

The BGLR R-package implements various types of shrinkage/variable selection Bayesian procedures for both univariate and multivariate response variables via Markov Chain Monte Carlo (MCMC) sampling. In the univariate case the algorithms developed were optimized for the case when the number of covariates (p) vastly exceeds the number of phenotypical records (n). Modern biobanks contains large amounts of genetic/genomic information in which in many cases $p >> n$, in which case existing routines implemented in BGLR do not scale with the sample size. We have developed the function `BLRXy` that implements most of the univariate models included in the original version of the package using algorithms optimized for the case that $p >> n$. Samples from the posterior distributions are obtained using Gibbs sampler and Metropolis algorithms, heavy computational parts are performed using compiled routines developed using the C programming language. In this note we present an overview of the models implemented, and application example and a benchmark of the proposed routines for different sample sizes and number of covariates.

Keywords: High-dimensional regression, Gibbs sampler, Metropolis algorithm.

1. Introduction

The BGLR function of the homonymous R-package (Pérez and de los Campos 2014) implements Bayesian shrinkage and variable selection models for parametric and semi-parametric regressions (de los Campos *et al.* 2013; Meuwissen *et al.* 2001). The software was originally tailored for single-trait regressions involving many more covariates (p) than sample size (n, i.e., $p >> n$). Many modern data sets, including data from human biobanks and large-scale genomic evaluations, can have a very large sample size (hundreds of thousands of individuals with phenotype and genotype records). In some of these data sets sample size can vastly exceed the number of predictors. In such settings computational performance and speed can be improved by implementing algorithms using summary statistics ($\boldsymbol{y}'\boldsymbol{y}$, $\boldsymbol{X}'\boldsymbol{X}$, and $\boldsymbol{X}'\boldsymbol{y}$) as opposed to algorithms that use phenotypes ($\boldsymbol{y}$) and incidence matrices ($\boldsymbol{X}$) directly for computations as it is the case of the `BGLR` R-function. Thus, to offer efficient software for biobank-sized data we added to the BGLR package the `BLRXy` function which generates posterior samples using summary statistics as inputs; the function has a much faster performance than BGLR when $n >> p$.

2. Models and Methods

Consider a linear regression model of the form:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{e}, \tag{1}$$

where $\boldsymbol{y} = (y_1, \ldots, y_n)'$ is a vector of phenotypes of dimension $n \times 1$, $\boldsymbol{X} = \{x_{ij}\}$ is a matrix of genotypes of dimension $n \times p$, $\boldsymbol{\beta} = (\beta_1, ..., \beta_p)'$ is a vector of marker effects of dimension $p \times 1$ and $\boldsymbol{e} = (e_1, ..., e_n)'$ is a vector of error terms of dimension $n \times 1$. For ease of presentation we assume that the phenotype and the SNP genotypes have been centered; therefore, we do not include an intercept and covariates. Relaxing this assumption possess no conceptual or computational difficulty.

Likelihood

For a quantitative (possibly transformed) trait the errors will be assumed to be IID (identically and independently distributed) normal $e_i \sim NIID(0, \sigma_e^2)$; therefore, the conditional distribution of the data given the model parameters $\boldsymbol{\theta} = \{\boldsymbol{\beta}, \sigma_e^2\}$ becomes:

$$p(\boldsymbol{y}|\boldsymbol{\theta}) = MVN(\boldsymbol{y}|\boldsymbol{X}\boldsymbol{\beta}, \boldsymbol{I}\sigma_e^2) = (2\pi\sigma_e^2)^{-n/2} \exp\left\{-\frac{1}{2\sigma_e^2}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})'(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})\right\}, \tag{2}$$

where $MVN(\boldsymbol{y}|\boldsymbol{X}\boldsymbol{\beta}, \boldsymbol{I}\sigma_e^2)$ stands for multivariate normal density with mean $\boldsymbol{X}\boldsymbol{\beta}$ and (co)variance matrix $\boldsymbol{I}\sigma_e^2$.

Prior distributions

The prior distribution specifies probabilities over the possible values of the model unknowns, $\boldsymbol{\theta} = \{\boldsymbol{\beta}, \sigma_e^2\}$. Implicitly the prior also specifies probabilities over models; the choice of prior plays a very important role in error control. In the model above-described the most critical component is the prior of marker effects. Let $p(\boldsymbol{\theta}|H)$ the prior distribution for $\boldsymbol{\theta}$ given a set of hyper parameters H. Different prior distributions can be assigned to the elements in $\boldsymbol{\beta}$, which induces shrinkage of estimates that depend on the size of effect (Gianola 2013), so different priors that are assigned to $\boldsymbol{\beta}$ lead to different models, e.g. Bayesian Ridge Regression, Double Exponential (LASSO; Park and Casella 2008), Scaled t (BayesA; Meuwissen *et al.* 2001), Scaled t-mixture (BayesB; Meuwissen *et al.* 2001), Gaussian mixture (BayesC; Habier *et al.* 2011) among many others (see Gianola 2013, for further details).

Posterior inferences

The posterior distribution of $\boldsymbol{\theta}$ can be obtained by applying the Bayes' theorem and it is proportional to the product of the conditional distribution of the data given the unknowns (2) times the prior, that is $p(\boldsymbol{\theta}|\boldsymbol{y}, H) \propto p(\boldsymbol{y}|\boldsymbol{\theta})p(\boldsymbol{\theta}|H)$. This posterior distribution does not have closed form in general; however when we assign Gaussian (Bayesian Ridge Regression), double exponential (LASSO), Gaussian mixture (Bayes C), Scaled t-mixture (BayesB) distribution to the elements of $\boldsymbol{\beta}$ and a scaled inverse chi-squared distribution or inverted gamma distribution to σ_e^2 the fully conditional distributions $p(\boldsymbol{\beta}|else)$ and $p(\sigma_e^2|else)$ do have closed form (see for example Gianola 2013; de los Campos *et al.* 2013). Therefore, samples can be collected using

a Gibbs sampler (Geman and Geman 1984). Because p (the number of SNP effects) is often large, sampling marker effects is the most computationally involved step of each cycle of the sampler.

The likelihood function (2), and therefore the posterior distribution can be expressed either in terms of the vector of phenotypes and the matrix of genotypes $\{\boldsymbol{y}, \boldsymbol{X}\}$ or in terms of summary statistics $\{\boldsymbol{y}'\boldsymbol{y}, \boldsymbol{X}'\boldsymbol{y}, \boldsymbol{X}'\boldsymbol{X}\}$. Sampling from the posterior distribution using a posterior distribution expressed in terms of $\{\boldsymbol{y}, \boldsymbol{X}\}$ requires using operators with complexity $O(n)$; these computations must be repeated for each effect in the model, leading to an algorithm with complexity $O(np)$ per cycle of the sampler. It can be shown that the full conditional distributions for $\beta_j|else$ for well-known Bayesian models (e.g. Bayesian Ridge Regression, Bayesian LASSO, BayesA, BayesB, BayesC) is normal, with mean and variance equal to the solution(inverse of the coefficient of the left hand side) of the following equation (see de los Campos *et al.* 2009, for further details):

$$\left(\frac{1}{\sigma_e^2}\boldsymbol{x}_j'\boldsymbol{x}_j + \frac{1}{\vartheta_j}\right)\beta_j = \frac{1}{\sigma_e^2}\boldsymbol{x}_j'\boldsymbol{e}_j, \tag{3}$$

where $\boldsymbol{x}_j$ is the j-th column of $\boldsymbol{X}$, $\boldsymbol{e}_j = \boldsymbol{y} - \boldsymbol{X}_{-j}\boldsymbol{\beta}_{-j}$, with $\boldsymbol{X}_{-j}$ the matrix $\boldsymbol{X}$ after removing the j-th column, $\boldsymbol{\beta}_{-j}$ the vector $\boldsymbol{\beta}$ after removing the j-th entry, ϑ_j is a variance associated to marker j and depends on the prior assigned to marker effects. Note that estimating the posterior mode for $\boldsymbol{\beta}$, by using the Gibbs sampler has a computational complexity that is equivalent to use the Backfitting or Gauss-Seidel algorithms (Golub and Van Loan 1996). Algorithms implementations based in this strategy do not scale well for large-n problems, that is when $n >> p$. Thus, to meet the challenges emerging with big data we have developed alternative algorithms with computational complexity independent of sample size.

When $n \geq p$ substantial improvements in computational performance, can be achieved by implementing the Gibbs sampler using cross products $\{\boldsymbol{y}'\boldsymbol{y}, \boldsymbol{X}'\boldsymbol{y}, \boldsymbol{X}'\boldsymbol{X}\}$ as inputs, although the RAM memory requirements may increase. In this case equation (3) can be rewritten in terms of the summary statistics and therefore sample $\beta_j|else$ using these inputs, that is:

$$\left(\frac{1}{\sigma_e^2}\boldsymbol{x}_j'\boldsymbol{x}_j + \frac{1}{\vartheta_j}\right)\beta_j = \frac{1}{\sigma_e^2}\left[\boldsymbol{x}_j'\boldsymbol{y} - (\boldsymbol{x}_j'\boldsymbol{X}\boldsymbol{\beta} - \beta_j\boldsymbol{x}_j'\boldsymbol{x}_j)\right], \tag{4}$$

where $\boldsymbol{x}_j'\boldsymbol{x}_j$ corresponds to the j-th diagonal element from $\boldsymbol{X}'\boldsymbol{X}$, $\boldsymbol{x}_j'\boldsymbol{y}$ corresponds to the j-th row from $\boldsymbol{X}'\boldsymbol{y}$ and $\boldsymbol{x}_j'\boldsymbol{X}$ corresponds to the j-th column from $\boldsymbol{X}'\boldsymbol{X}$. This will lead to an updating algorithm per cycle of the sampler of complexity $O(p)$, because all required heavy computational inputs were already pre-computed. Computing $\boldsymbol{X}'\boldsymbol{X}$ can be both memory and computationally demanding. However, the computation of $\boldsymbol{X}'\boldsymbol{X}$ is an "embarrassingly parallel" problem (Pacheco 2011); blocks of $\boldsymbol{X}'\boldsymbol{X}$ can be computed separately at multiple nodes in a cluster and the blocks can then be (virtually) merged. In modern computing platforms where multicore CPU's are available, multithread optimized version of BLAS (Basic Linear Algebra Subprograms) are available and can be used to compute blocks of $\boldsymbol{X}'\boldsymbol{X}$ or even the full matrix.

Model (1) can be further extended to include other predictors by defining a linear predictor $\boldsymbol{\eta} = E(\boldsymbol{y}|\boldsymbol{\theta})$ that represents the conditional expectation function which is given by $\boldsymbol{\eta} = \sum_{k=1}^{K}\boldsymbol{X}_k\boldsymbol{\beta}_k$, so that the model can be rewritten as $\boldsymbol{y} = \boldsymbol{\eta} + \boldsymbol{e}$, where $\boldsymbol{X}_k$ are matrixes of

predictors and $\boldsymbol{\beta}_k$ corresponds to vectors of regression coefficients associated to $\boldsymbol{X}_k$. Let $\boldsymbol{\theta}$ represent the set of unknowns in the model, i.e., $\boldsymbol{\beta}_k$'s, residual variance, etc., and let $p(\boldsymbol{\theta}|H)$ the prior assigned to $\boldsymbol{\theta}$ given a set of hyperparameters. Then the likelihood $p(\boldsymbol{y}|\boldsymbol{\theta}) = MVN(\boldsymbol{y}|\boldsymbol{\eta}, \boldsymbol{I}\sigma_e^2)$, and therefore the posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{y}, H)$ is given by $p(\boldsymbol{\theta}|\boldsymbol{y}, H) \propto p(\boldsymbol{y}|\boldsymbol{\theta})p(\boldsymbol{\theta}|H)$. By defining $\boldsymbol{X} := [\boldsymbol{X}_1, ..., \boldsymbol{X}_K]$ and $\boldsymbol{\beta} = (\boldsymbol{\beta}_1', ..., \boldsymbol{\beta}_K')'$ and if entries in $\boldsymbol{\beta}_k, k = 1, ..., K$ are assigned either Gaussian, Laplace, scaled t, scaled t-mixtures mixtures of normal distributions, then the entries the conditional distributions of $\beta_{kj}|else$ are normally distributed as explained before.

Recently, Zhao *et al.* (2020), proposed another strategy to speed up computations in Bayesian Regression models for whole genomic prediction. The authors use an Orthogonal data augmentation strategy (e.g. Xiong *et al.* 2016) to orthogonalize the Gibbs sampler and therefore sample marker effects independently in parallel.

The strategy is interesting, but it adds some computational burden to the original problem, for example: 1) Computation of the biggest eigen-value for the matrix $\boldsymbol{X}'\boldsymbol{X}$, 2) Computation of the Cholesky decomposition of a matrix of order $p \times p$, 3) Sampling from imputed phenotypes $\boldsymbol{y}_{imp}$ at each iteration of Gibbs sampler which is a vector of order $p \times 1$, 4) Sampling marker effects based on a matrix of augmented genotypes of dimension $(n + p) \times p$. Apart from that, it is well known that augmenting the data in this way, will slow down convergency to the posterior distribution. It is also well known that parallel computing approaches do not scale linearly with the number of available processors because there exists always an overhead in times due to communication between processes and access to shared resources. Therefore we think that there is still room for improvement for fitting the models using summary statistics under this context.

3. Software

The BGLR package (Pérez and de los Campos 2014) includes a set of routines to perform genomic regression and prediction for continuous (censored and uncensored) and discrete traits. The software package includes a wide variety of parametric and semi-parametric models used in genomics (e.g. Bayesian Ridge Regression, BayesA, BayesB, BayesC, Reproducing Kernel Hilbert Spaces, etc.). The software was designed and optimized to fit efficiently the models in the case that $p >> n$. The BGLR package has been continuously updated since its initial release, including bug fixes, improving algorithms, adding new models, examples, improving documentation, etc. The updates have has been released initially in the package's github website (`https://github.com/gdlc/BGLR-R`) and once that have been tested extensively have been incorporated to the stable release available at CRAN.

In the case of the models described previously we have developed a new set of routines written in the C and R programming languages and optimized them for the case when the sample size (n) is much larger that the number of predictors (p). The routines can be accessed through the function `BLRXy` included in BGLR package. The `BLRXy` function is able to obtain posterior samples from this distribution collected using a Gibbs Sampler when the priors assigned to $\boldsymbol{\beta}_k$ corresponds to a flat prior (FIXED), Gaussian (BRR), Scaled-t (BayesA), Gaussian mixture (BayesC) and Scaled t-mixture (BayesB), see Tables 1 and S1 in the BGLR package (Pérez and de los Campos 2014). Internally, the function computes summary statistics, $\{\boldsymbol{y}'\boldsymbol{y}, \boldsymbol{X}'\boldsymbol{y}, \boldsymbol{X}'\boldsymbol{X}\}$, with $\boldsymbol{X} = [\boldsymbol{X}_1, ..., \boldsymbol{X}_K]$ and performs all the calculations based on these inputs. The user

interface in the `BLRXy` function for specifying prior distributions is exactly the same that for the case of the `BGLR` function. The software is also able to deal with response vector that contains missing values, internally, after the model is fitted the missing values are predicted using as point estimates the posterior mean of the estimated parameters. Missing values are not allowed for the predictors. The routine `BLRXy` is a wrapper for the function `BLRCross` that works directly with $\{\boldsymbol{y}, \boldsymbol{X}'\boldsymbol{y}, \boldsymbol{X}'\boldsymbol{X}\}$ and can be used for example when only this summary statistics are available or when due to time and resources efficiency, these quantities have been already pre-computed. The syntax of the `BLRCross` routine is also list based and is described in the user's manual for the BGLR package. For sake of simplicity we only illustrate the use of the `BLRXy` routine.

We benchmarked our current implementation of the proposed algorithm in the `BLRXy` function against `BGLR` function (Figure 1). For models involving 10K SNPs, the current implementation completes 1,000 cycles of the Gibbs sampler in less than 80 seconds (panel A in Figure 1). For problems involving $n \sim p$ the proposed algorithm is approximately twice as fast than `BGLR`. However, for problems involving $n > p$ the proposed algorithm is one (e.g., 32 times faster, 1/0.032) or two orders of magnitude faster (e.g. 333 times faster, 1/0.003).

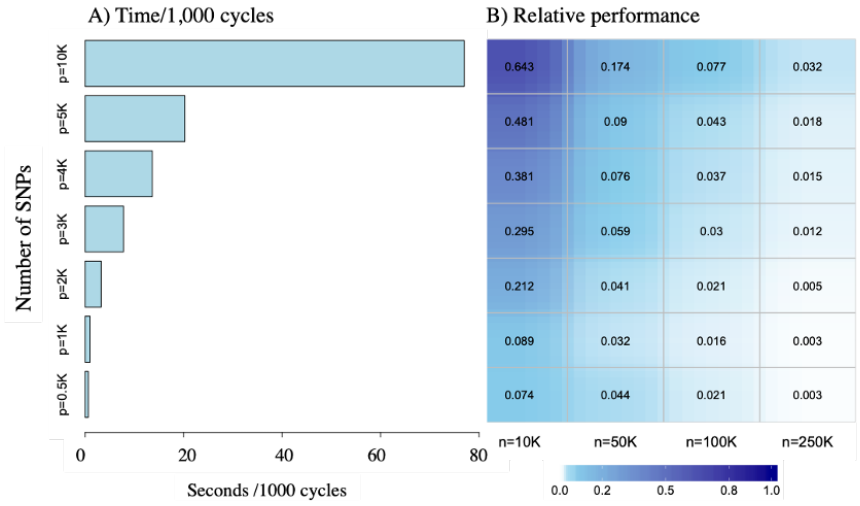

Figure 1: Computational performance of the proposed algorithm. A) Seconds needed to complete 1,000 cycles of the Gibbs sampler; B) Time required by the proposed algorithm to collect 1, 000 samples relative to the time needed for the same task by BGLR.

4. Examples

Box 1: Fitting BayesA with simulated data

```
load("mice.RData")

p=1000
n=1500

X<-scale(mice.X[1:n,1:p],center=TRUE)
A<-mice.A

A<-A[1:n,1:n]

QTL<-seq(from=50,to=p-50,by=80)
b<-rep(0,p)
b[QTL]<-1
signal<-as.vector(X%*%b)

error<-rnorm(sd=sd(signal),n=n)
y<-error+signal
y<-2+y

#BayesA, missing values not present
ETA<-list(list(X=X,model="BayesA"))
fm1<-BLRXy(y=y,ETA=ETA)
plot(fm1$yHat,y)
```

References

de los Campos G, Hickey JM, Pong-Wong R, Daetwyler HD, Calus MPL (2013). "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 (2009). "Predicting Quantitative Traits with Regression Models for Dense Molecular Markers and Pedigree." *Genetics*, **182**(1), 375–385.

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.

Golub GH, Van Loan CF (1996). *Matrix Computations*. Third edition. The Johns Hopkins University Press.

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

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

Pacheco P (2011). *An Introduction to Parallel Programming*. 1st edition. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. ISBN 9780123742605.

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

Pérez P, de los Campos G (2014). "Genome-Wide Regression and Prediction with the BGLR Statistical Package." *Genetics*, pp. 483–495.

Xiong S, Dai B, Huling J, Qian PZG (2016). "Orthogonalizing EM: A Design-Based Least Squares Algorithm." *Technometrics*, **58**(3), 285–293. doi:10.1080/00401706.2015.1054436. PMID: 27499558, https://doi.org/10.1080/00401706.2015.1054436, URL https://doi.org/10.1080/00401706.2015.1054436.

Zhao T, Fernando R, Garrick D, Cheng H (2020). "Fast parallelized sampling of Bayesian regression models for whole-genome prediction." *Genetics Selection Evolution*, **52**. doi:10.1186/s12711-020-00533-x.

Affiliation:

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

Gustavo de los Campos
Department of Epidemiology and Biostatistics
Michigan State University, USA
Telephone: +1/517/353-8623
E-mail: gustavoc@msu.edu
https://epibio.msu.edu/faculty/deloscampos