Journal of Statistical Software

January 2016, Volume 69, Issue 2. doi: 10.18637/jss.v069.i02

R2GUESS: A Graphics Processing Unit-Based R Package for Bayesian Variable Selection Regression of Multivariate Responses

Benoît Liquet
LMAP, UMR CNRS 5142
Université de Pau et
des Pays de L'Adour

Leonardo Bottolo
Imperial College London

Gianluca Campanella
Imperial College London

Sylvia Richardson
MRC Biostatistics Unit, Cambridge

Marc Chadeau-Hyam
Imperial College London

Abstract

Technological advances in molecular biology over the past decade have given rise to high dimensional and complex datasets offering the possibility to investigate biological associations between a range of genomic features and complex phenotypes. The analysis of this novel type of data generated unprecedented computational challenges which ultimately led to the definition and implementation of computationally efficient statistical models that were able to scale to genome-wide data, including Bayesian variable selection approaches. While extensive methodological work has been carried out in this area, only few methods capable of handling hundreds of thousands of predictors were implemented and distributed. Among these we recently proposed **GUESS**, a computationally optimised algorithm making use of graphics processing unit capabilities, which can accommodate multiple outcomes. In this paper we propose **R2GUESS**, an R package wrapping the original C++ source code. In addition to providing a user-friendly interface of the original code automating its parametrisation, and data handling, **R2GUESS** also incorporates many features to explore the data, to extend statistical inferences from the native algorithm (e.g., effect size estimation, significance assessment), and to visualize outputs from the algorithm. We first detail the model and its parametrisation, and describe in details its optimised implementation. Based on two examples we finally illustrate its statistical performances and flexibility.

Keywords: Bayesian variable selection, OMICs data, C++, graphics processing unit, multivariate regression, R.

1. Introduction

The investigation of associations between genetic and genomic characteristics, such as genome-wide scan or methylation profiles, and complex phenotypes relies on the capacity to link several datasets comprising tens to hundreds of thousands of measurements available usually in hundreds (if not thousands) of individuals and an outcome of interest. In some cases (e.g., metabolic syndrome) the outcome can be complex and may consist in several correlated measures of continuous variables. In Figure 1, we illustrate the most general situation of p predictors (typically OMICs measures) being analyzed in relation to q correlated outcomes based on n observations.

With the genome-wide association study (GWAS) era, strong methodological efforts have been carried out, and reliable statistical methods for the analysis of such large and complex sets of data have emerged. Resulting methods include univariate approaches coupled with multiple testing correction strategies, dimension reduction techniques, penalized regression and Bayesian shrinkage approaches. Numerous methods are now well-established for GWAS and have been extensively reviewed (Balding 2006; Cantor, Lange, and Sinsheimer 2010; Do, Müller, and Vannucci 2006; Chadeau-Hyam *et al.* 2013).

While most of these approaches were built upon a frequentist framework, efficient Bayesian alternatives have also been proposed. Bayesian variable selection (BVS) approaches are designed to select subsets of covariates on the basis of their ability to predict the outcome of interest. BVS approaches generalize the stochastic search variable selection (SSVS) paradigm proposed by George and McCulloch (1993) by including a conjugate hierarchical setup in the model. Unlike in the original SSVS approach, where the prior structure encodes a threshold of practical importance for the effect of a variable, BVS performs variable selection on the basis of how well regression coefficients can be distinguished from 0. To date, despite methodological developments in the area, only few BVS implementations able to scale up to tens or hundreds of thousands of variables are available (Hans, Dobra, and West 2007; Guan and Stephens 2011; Bottolo and Richardson 2010). These approaches incorporate a latent binary vector of size p specifying a model by indicating which predictors it includes. BVS models therefore aim at the identification of the value(s) of that vector that corresponds to well-supported models in terms of posterior score. Their main computational challenge

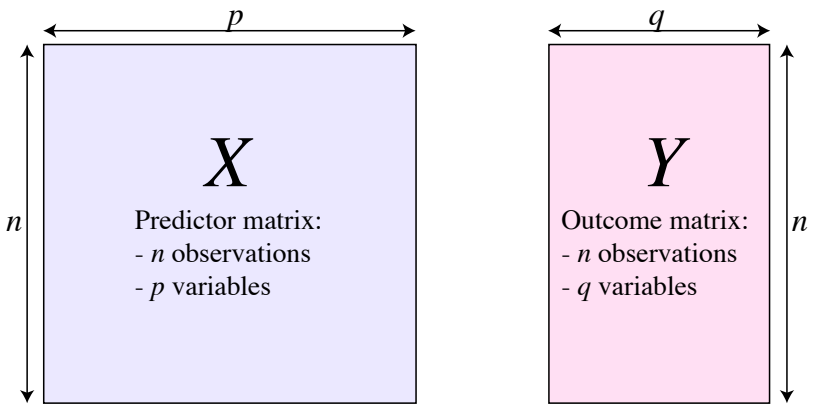

Figure 1: General representation of data for the investigation of OMICs biomarkers.

resides in the high dimensionality and potential multimodality of the model space (2^p) where the search for the latent binary vectors describing important models is performed.

Shotgun stochastic search (**SSS**; Hans *et al.* 2007) can handle tens of thousands of predictors simultaneously. The search algorithm explores at a given iteration (assuming that at the previous iteration the retained model contained m variables) all models of size $m-1$, m, and $m+1$, and proposes to move to one of these models. In **piMASS**, a full Monte Carlo Markov chain (MCMC)-based variable selection search is implemented, which includes a proposal distribution that accounts for the correlation between each predictor and the outcome in order to improve the mixing (Guan and Stephens 2011). While both **SSS** and **piMASS** can accommodate binary and continuous outcomes, their use is restricted to univariate outcomes. In order to enable the investigation of complex and multidimensional continuous outcomes, we developed **ESS++** (Bottolo *et al.* 2011a), whose search algorithm is based on an evolutionary Monte Carlo procedure (Bottolo and Richardson 2010). Further developments over **ESS++** included improvement in the computational efficiency through the exploitation of graphics processing unit (GPU) capacities for rate limiting matrix operations. The resulting GPU-enabled C++ program, **GUESS**, can jointly analyze hundreds of thousands of predictors measured in thousands of individuals, as illustrated in a recent Bayesian GWAS of lipid phenotypes (Bottolo *et al.* 2013). Applications of **GUESS** on case studies (with $n = 800$, and $p = 30,000$) showed numerical stability for $q \leq 50$ responses. However, in order to analyze multidimensional responses, **GUESS** relies on the modeling of the correlation structure among the responses. Thus, the set of responses which will be jointly associated with the same pool of predictors, should be supported by prior epidemiological/biological evidence. This limits the number of outcomes and to ensure interpretability, it is recommended to restrict the use of **R2GUESS** to a small number (no more than 10) outcomes. In case of a larger number of outcomes, filtering/clustering approaches among the (high dimensional) responses are more suitable through partitioning algorithms (Monni and Tadesse 2009) or by adding a hierarchical framework onto the **GUESS** variable selection procedure (Bottolo *et al.* 2011b).

Here we introduce **R2GUESS**, an R package which provides a user-friendly interface to run the **GUESS** algorithm (Liquet and Chadeau-Hyam 2014). In addition to wrapping the C++ source code, **R2GUESS** also automates the management of input/output files, includes functions to post-process outputs from **GUESS** and offers additional resources to assess statistical significance and to estimate effect size of the variables identified. **R2GUESS** can handle up to several thousands of observations (n), hundreds of thousands of predictors (p) and a few responses (q) simultaneously. Computational time increases mainly with n, p, and to a lesser extent with q, but unlike other approaches (e.g., penalized regression), **R2GUESS** does not require calibration through cross-validation, making it computationally competitive. **R2GUESS** is available from the Comprehensive R Archive Network (CRAN) at `http://CRAN.R-project.org/package=R2GUESS`.

We will first describe the linear model (and the corresponding prior specification) on which **R2GUESS** is based. We will then provide details on the evolutionary Monte Carlo (EMC) algorithm implemented within **R2GUESS** and describe the set of moves implemented. After detailing the installation procedure, we will illustrate the use of **R2GUESS** and its main outputs based on both a simulated and a real case example.

2. R2GUESS: Definitions, model parametrization

2.1. Formulation of the underlying linear model

R2GUESS is built upon the multivariate extension of the Gaussian linear model

$$Y - XB \sim \mathcal{N}(I_n, \Sigma), \tag{1}$$

where Y is an $n \times q$ matrix of responses, X is an $n \times p$ matrix of predictors and B is a $p \times q$ matrix of regression coefficients. In the model, both matrices X and Y are centered, and pre-processing procedures within **R2GUESS** systematically center the input X and Y matrices. $\mathcal{N}(\cdot, \cdot)$ indicates the normal matrix-variate as defined in Dawid (1981) where Σ $(q \times q)$ controls the responses' residual correlation (the variance-covariance matrix of $Y - XB$) and the observations are treated as independent (e.g., no familial structure is assumed in the data, with I_n denoting the $n \times n$ identity matrix). Setting $q = 1$, the Gaussian linear model simplifies to

$$y - X\beta \sim N_n\left(0, \sigma^2 I_n\right), \tag{2}$$

where y is an $n \times 1$ vector, β is a $p \times 1$ vector of regression coefficients, and σ^2 corresponds to the variance of the error term. We denote with $N_n(\cdot, \cdot)$ the n-variate normal distribution.

To characterize a regression model, we introduce a latent binary vector γ of size $p \times 1$ indicating which covariate contributes to the model,

$$\gamma = (\gamma_1, \ldots, \gamma_p)^\top.$$

Let B_j be the vector of q linear regression coefficients linking the jth covariate and the (q-dimensional) outcome. Then $\gamma_j = 1$ if $B_j \neq 0$, for all responses, and $\gamma_j = 0$ if $B_j = 0$, for all responses, $j = 1, \ldots, p$. Plugging the latent structure into the Gaussian linear model (1),

$$Y - X_\gamma B_\gamma \sim \mathcal{N}(I_n, \Sigma), \tag{3}$$

where B_γ is the $p_\gamma \times q$ matrix of non-zero regression coefficients extracted from B. The model size p_γ is the dimension of the non-zero elements of γ, i.e., $p_\gamma \equiv 1_p^\top \gamma$, with 1_p the p-vector of ones. X_γ is the design matrix of dimension $n \times p_\gamma$ compiling columns of X for which $\gamma_j = 1$.

In the case of univariate outcomes ($q = 1$), conditionally on the binary vector γ, the linear model (2), simplifies to

$$y - X_\gamma \beta_\gamma \sim N_n\left(0, \sigma^2 I_n\right). \tag{4}$$

2.2. Prior specification

We describe here the prior distributions for all unknowns, including the model space and the regression coefficients. The conjugate prior density for the $p_\gamma \times q$ matrix of non-zero regression coefficients B_γ is

$$B_\gamma | \gamma, \Sigma \sim \mathcal{N}(H_\gamma, \Sigma), \tag{5}$$

where H_γ is a $p_\gamma \times p_\gamma$ matrix controlling the correlation structure of the regression coefficients across the p_γ selected predictors, and Σ is the correlation between the q columns of B_γ which

for computational purposes, is set equal to Σ, the error variance. In addition, the prior density of Σ is assumed to follow an inverse Wishart distribution (denoted by $\mathcal{IW}(\cdot,\cdot)$):

$$\Sigma \sim \mathcal{IW}(d, Q), \tag{6}$$

where d and Q are respectively the degrees of freedom and a (positive definite) scale matrix such that $\mathsf{E}(\Sigma) = Q/(d-2)$. The matrix Q is defined as $Q = kI_q$ with the hyperparameter k being comparable in size with the likely error variance of Y given X and $d = 3$ representing the smallest integer value ensuring the existence of $\mathsf{E}(\Sigma)$.

When $q = 1$, (5) becomes

$$\beta_\gamma|\gamma, \sigma^2 \sim N_{p_\gamma}\left(0, \sigma^2 H_\gamma\right),$$

and the variance σ^2 follows an inverse Gamma ($\Pi\Gamma(\cdot,\cdot)$) distribution

$$\sigma^2 \sim \Pi\Gamma(a, b).$$

The mode of the distribution of σ^2 is at $b/(1+a)$ and a relatively uninformative prior is obtained by taking a and b close to 0, for example, $a = 10^{-10}$ and $b = 0.001$.

The default prior specification for H_γ relies on a g-prior setting as introduced by Zellner (1986):

$$H_\gamma = \tau\left(X_\gamma^\top X_\gamma\right)^{-1},$$

where τ, the variable selection coefficient controlling the model size (p_γ), is unknown and included in the sampling scheme. An inverse Gamma prior density is specified for this parameter

$$\tau \sim \Pi\Gamma(1/2, n/2)$$

with $\mathsf{E}(\tau) = n$.

The main assumption underlying g-priors is that the correlation structure of the regression coefficients among the p predictors replicates the covariance structure of the likelihood. The benefit is threefold: first, the calculation of the marginal likelihood is simplified; second, it enables the variance of each regression coefficient to adapt automatically to the scale of the covariates, which is of primary importance when the X matrix incorporates heterogeneous predictors; and last, it discourages positively correlated predictors to be included at the same time in the regression model.

To complete the parametrization of the model, the prior of the latent binary vector γ, $p(\gamma)$ must be specified. Assuming exchangeability across predictors, we set

$$\gamma_j \sim \text{Bernoulli}(\omega),$$

where the hyperparameter ω represents the probability that any of the predictors enter the model. This setting induces a binomial prior on p_γ:

$$p_\gamma|\omega \sim \text{Binomial}(p, \omega). \tag{7}$$

We set the prior distribution for ω as $B(a_\omega, b_\omega)$:

$$p(\omega) = \omega^{a_\omega - 1}(1-\omega)^{b_\omega - 1}/B(a_\omega, b_\omega),$$

where $B(\cdot,\cdot)$ is the Beta function. Integrating out ω from (7), the prior distribution of $p(\gamma)$

$$p(\gamma) = \int p(\gamma|\omega)p(\omega)d\omega = \frac{B(p_\gamma + a_\gamma, p - p_\gamma + b_\omega)}{B(a_\omega, b_\omega)},$$

is a Beta-Binomial distribution. The hyper-parameters a_ω and b_ω of that distribution can be determined by back calculation once $\mathsf{E}(p_\gamma)$ and $\mathsf{VAR}(p_\gamma)$ – respectively the *a priori* expected number of predictors and its variance – are specified.

Instead of generalized g-priors (Maruyama and George 2011) that can accommodate correlated predictors without restriction on the model size, we opted for a simpler g-prior setting which relies on the full rank assumption of X_γ. Numerically, this restricts the model space to models of size $\leq (n-1)$, which is consistent with the sparse nature of our approach, and does not affect its performance provided that the number of observations is sufficiently large. In addition, we have allowed the user to impose a constraint on $p_{\max}$, the total number of predictors included in the model (see Table 1).

2.3. Marginal likelihood calculation

Based on the likelihood and the prior specification of the parameters detailed in Section 2.2, the joint distribution of the variables can be written as

$$p(Y, \gamma, B_\gamma, \tau, \Sigma) = p(Y|\gamma, B_\gamma, \Sigma)p(B_\gamma|\gamma, \tau, \Sigma)p(\Sigma)p(\tau)p(\gamma).$$

For computational efficiency, the parameters B_γ and Σ are integrated out, leading to

$$\begin{aligned} p(Y|\gamma,\tau) &= \int p(Y|\gamma, B_\gamma, \Sigma)p(B_\gamma|\gamma,\tau,\Sigma)p(\Sigma)dB_\gamma d\Sigma \\ &= (1+\tau)^{-(p_\gamma/2)q}|kI_n + S(\gamma)|^{-[d+n+(q-1)-1]/2}, \end{aligned} \tag{8}$$

where

$$S(\gamma) = Y^\top Y - \tau/(1+\tau)Y^\top X_\gamma (X_\gamma^\top X_\gamma)^{-1} X_\gamma^\top Y.$$

In the case of univariate response, the linear regression model is integrated over β_γ and σ^2.

Once the regression coefficients and the error variance have been marginalized, only two parameters remain to be sampled from their joint distribution: the latent binary vector γ and the shrinkage coefficient τ. Sampling from the target distribution $p(\gamma, \tau|Y)$ is possible using their full conditional distributions:

$$p(\gamma|\cdots) \propto p(Y|\gamma,\tau)p(\gamma) \tag{9}$$

$$p(\tau|\cdots) \propto p(Y|\gamma,\tau)p(\tau). \tag{10}$$

3. MCMC implementation

The main computational challenge of the model estimation relies on the efficient exploration of the 2^p-dimensional and multi-modal model space. **R2GUESS** search algorithm is based upon an evolutionary Monte Carlo procedure combining MCMC and genetics algorithms (Liang and Wong 2000).

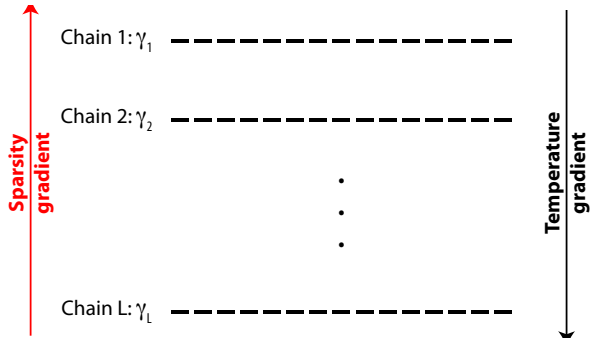

Figure 2: Illustration of the chain tempering.

In addition, **R2GUESS** features a parallel tempering strategy to improve the mixing properties of the MCMC algorithm. The principle of parallel tempering is to run several chains in parallel (Figure 2), each of which is powered by a scalar representing the inverse of its temperature.

Higher temperatures flatten the posterior density allowing (i) more variables to be included in the model, and (ii) to escape from local modes and favoring changes in the chain specific γ configuration. The first chain is not heated (temperature is set to 1) and exchanges information with heated chains through moves implemented in the genetic algorithm. Heated chains are included to improve mixing properties and to ease convergence in the product space of all chains. In the current implementation, only the state of the first (not heated) chain is recorded. Setting L the number of parallel chains, and noting t_l, $1 = t_1 < t_2 < \ldots < t_L$, the temperature corresponding to the lth chain, the "tempered" full conditionals (9) and (10) can be written as

$$[p(\gamma|\cdots)]^{1/t_l} \propto [p(Y|\gamma,\tau)]^{1/t_l}[p(\gamma)]^{1/t_l}, \tag{11}$$

and

$$p(\tau|\cdots) \propto p(\tau)\prod_{l=1}^{L}[p(Y|\gamma,\tau)]^{1/t_l}. \tag{12}$$

The **R2GUESS** search algorithm includes a specific and computationally optimized set of genetic moves: mutation, crossover, global moves. It also comprises an automated temperature calibration during the burn-in ensuring an optimal balance between mixing efficiency and computational time. Finally **R2GUESS** includes an adaptive sampling procedure for the shrinkage parameter τ.

3.1. Evolutionary Monte Carlo (EMC): Genetic moves

The EMC procedure embedded in **ESS++** and **R2GUESS** consists in the definition of a collection of specific moves ensuring the exchange of information within each chain (local or mutation moves) and between chains (crossover and exchange moves). The calibration of these moves is mostly automated to ensure good mixing, based on default values for the modeling parameters (Table 1). However, these can be user-defined and the integration of additional moves is eased by the modular structure (defining one class per type of move) of the C++ source code.

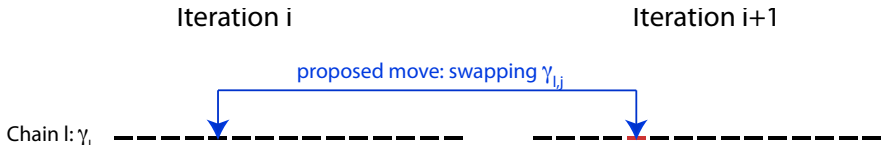

Figure 3: Principle of a mutation move.

Throughout the text, we will use the double indexing $\gamma_{l,j}$, $l = 1, \ldots, L$ and $j = 1, \ldots, p$ to denote the jth latent binary indicator in the lth chain, and $\gamma_l = (\gamma_{l,1}, \ldots, \gamma_{l,p})^\top$ will denote the vector of binary indicators that characterizes the state of the lth chain of the population $\gamma = (\gamma_1, \ldots, \gamma_L)$. In the following sections, we will briefly describe each move implemented in **R2GUESS**. More detailed information about their parametrization is detailed in Bottolo and Richardson (2010).

Mutation moves

Mutation moves are defined as changes that are restricted to a single chain (i.e., local effect). Assuming that the mutation move occurs on a specific chain l, as illustrated in Figure 3, it will simply consist in swapping the jth value of $\gamma_{l,j}$, from 1 to 0 or from 0 to 1, i.e., removing the variable j from the model if it was in the model at the previous step ($1 \to 0$), or adding it in if it was not ($0 \to 1$).

Two types of mutation moves are implemented in **R2GUESS**:

- *Gibbs move*

  The Gibbs move samples, for each j in a random order, a new value for $\gamma_{j,l}$ from its full conditional distribution. An exhaustive scan of this nature has been shown to be effective for mixing but is time-consuming (one Gibbs move corresponds to the evaluation of p models). Therefore it will only occur every `GIBBS_N_BATCH` iterations, where this number can be user-defined (see Table 1). The Gibbs move only applies to the first (not-heated) chain.

- *Fast-scan Metropolis-Hastings (FSMH)*

  An FSMH move proposes a swapping move ($1 \to 0$ or $0 \to 1$) for a randomly selected subset of elements in γ_l. The probability that a given element of γ_l is selected depends on its current value; the current model size; on ω, the hyper-parameter automatically calculated from $\mathsf{E}(p_\gamma)$ and σ_{p_γ} the user-defined *a priori* expected model size, and standard deviation of the model size respectively; and on p, the number of predictors. The FSMH move applies to all chains.

  At each iteration an FSMH move is performed with probability `Prob_mut`, which is user-defined (`P_MUTATION` see Table 1). If not sampled (with probability 1− `Prob_mut`), a global crossover move is attempted instead.

Global moves

Global moves ensure exchange of information between chains by swapping part or all of the information between two randomly selected chains. Introducing "bolder moves" arising from

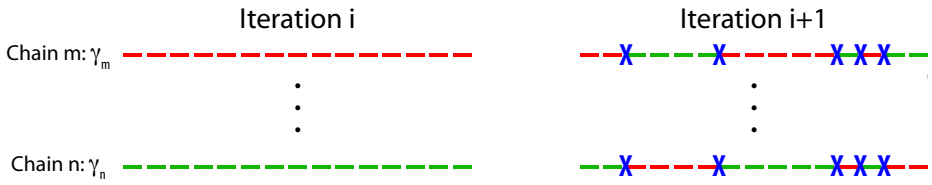

Figure 4: Illustration of a crossover move.

heated chains, enables the algorithm to escape from local modes and improves mixing. Two classes of global moves are implemented in **R2GUESS**: crossover and exchange moves.

- *Crossover moves* (Figure 4) rely on the three steps:

  1. *Selection of the chains*
     The two chains are selected at random according to normalized "Boltzmann weights" (Bottolo and Richardson 2010). These weights are derived from the (conditional) posterior density and temperature of each chain. A sub-group with high weights is selected by applying a user-defined threshold `P_SEL` (see Table 1) to the cumulative distribution of the ordered "Boltzmann weights". The sampling distribution is then created by overweighting this group of chains and re-normalizing all probabilities. Two chains are then drawn at random according to this discrete distribution. This refined sampling strategy ensures that the two selected chains will be more likely to give rise to latent binary vectors with higher posterior density.
  2. *Crossover breakpoints*
     To define the crossovers, both the number and the position of breakpoints (blue crosses in Figure 4) are randomly sampled. Two strategies are proposed.
     a) *k-points crossover*
        In this case, the number of breakpoints is uniformly sampled from 1 to $k_{\max}+1$, the user-defined maximum number of breakpoints (`K_MAX` see Table 1). The breakpoint location(s) is (are) sampled from a uniform distribution, and the chains' latent binary vectors are swapped by shuffling the regions between two breakpoints.
     b) *Block crossover*
        The block crossover swaps blocks of highly correlated variables between the two sampled chains. A "reference" predictor j is first uniformly sampled. Then, all pairwise Pearson correlation coefficients $\rho(X_j, X'_j)$ $j' = 1, \ldots, p$, $j \neq j'$, are calculated. The "block" of variables is defined as all variables correlated to the reference predictor (i.e., whose pairwise correlation coefficient $|\rho(X_j, X'_j)| > \rho_0$), where the threshold ρ_0 is user-defined (`P_CSRV_R` see Table 1). Each of the "block" elements are finally swapped from one chain to another.
  3. *Acceptance probability*
     Regardless of the way chains are swapped, the (conditional) posterior probability is calculated for each chain and the proposed move is accepted with a probability depending on the difference in the (conditional) posterior probability induced by the move from the original to the swapped chains; the difference in the Boltzmann's weight induced by the move; and the temperature of the swapped chains.

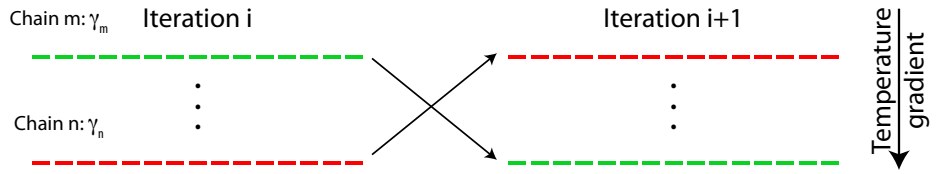

Figure 5: Illustration of an exchange move.

- *Exchange moves*

  As illustrated in Figure 5, exchange moves can be viewed as extreme cases of crossover moves, where chains are entirely swapped.

  R2GUESS includes two exchange moves: all exchange and delayed rejection.

  1. *All exchange move*

     The all exchange move is a Gibbs-type move and selects chains from a re-normalized probability depending on the (conditional) posterior probability of each chain. To guarantee reversibility of the move, the case where no chains are swapped is also considered.

  2. *Delayed rejection move (DR)*

     A first exchange move is proposed by uniformly sampling two chains. If that proposed move is rejected, a second move involving one of the two original chains and another chain which is adjacent in the temperature ladder is proposed. During the burn-in only, after every `TEMP_N_BATCH` (user-defined, see Table 1) proposed DR moves, the parameters determining the temperature scale (see below) are updated to ensure the DR acceptance rate, which is dependent on the current temperature placement, lies around its user-defined value (`TEMP_OPTIMAL`, see Table 1).

  During the burn-in, only the DR move is enabled, and once completed, the all exchange move is selected with a probability 1− `P_DR`, where `P_DR` is user-defined (see Table 1).

3.2. Temperature placement

The temperature tuning defines $t_l, l = 1, \ldots, L$ the temperature for all of the L chains assuming a geometric ladder:

$$t_l = b_t^{a_l}, \tag{13}$$

where b_t is a scalar common to all chains, and a_l is chain-dependent. For a given value of a, $a_l = (l-1)/a$. Initial values for a and b_t can be user-defined (see Table 1). During the burn-in, values for b_t are updated in order to achieve an effective exchange of information between the chains that will improve the overall convergence of the algorithm. This adaptation is stopped at the end of the burn-in.

Temperature placement is automated, and the value of b_t is updated in order to control the acceptance rate of the delayed rejection moves and keeps it close to its desired value (`TEMP_OPTIMAL` see Table 1). The initial value of b_t is user-defined (`B_T` see Table 1). Values for a_l are initialized to $(l-1)/a_{t_{\text{den}}}$, where $a_{t_{\text{den}}}$ is chosen among three possible values entered by the user: `A_T_DEN_INF_5K` if there are less than 5,000 predictors in X; `A_T_DEN_5_10K` if

there are between 5,000 and 10,000 predictors in X; `A_T_DEN_SUP_10K` if there are more than 10,000 predictors in X.

Every k (where k is user-defined, `TEMP_N_BATCH`) delayed rejection moves, the temperature placement occurs and the parameter b_t is updated as follows:

- If the acceptance rate for delayed rejection moves involving the first (non-heated) chain is 0, or if the average model size in the most heated chain is greater than 10 times the number of observations, the updated value for b_t is

$$b_t^* = \max\left\{M_0, b_t - (b_t - 1)/2\right\},$$

  where M_0 is the user-defined lower bound (`M_MIN` see Table 1) for b_t.

- If the acceptance rate for delayed rejection moves is 1, the updated value for b_t is

$$b_t^* = \min\left\{M_1, b_t - (b_t - 1)/2\right\},$$

  where M_1 is the user-defined upper bound (`M_MAX` see Table 1) for b_t.

- If the acceptance rate for delayed rejection moves is neither neither 0 nor 1, but below the optimal value (`TEMP_OPTIMAL` see Table 1), the updated value for b_t is

$$b_t^* = \max\left\{M_0, 2^{\log_2(b_t)-\delta_b}\right\},$$

  where δ_b is automatically calculated at the initialization of the temperatures using M_0, M_1, the length of the burn-in and the user-defined number of delayed rejection moves between two temperature placements (k).

- Using the same notations as above, if the acceptance rate for delayed rejection moves is neither 0 nor 1, but above the optimal value, the updated value for b_t is

$$b_t^* = \min\left\{M_1, 2^{\log_2(b_t)+\delta_b}\right\}.$$

The temperature of each chain is then updated based on the new value for b_t.

3.3. Sampling the shrinkage coefficient τ

The shrinkage coefficient τ is also included in the MCMC sampling scheme through an adaptive Metropolis-within-Gibbs algorithm. At each sweep, a candidate value τ' for τ is proposed. To ensure $\tau > 0$, $\log(\tau')$ is sampled from a Gaussian distribution

$$N\left(\log(\tau), e^{\texttt{ls}}\right),$$

where τ is the value retained at the previous sweep and `ls` is the log standard deviation of the proposal density for τ. The candidate value τ' is then accepted with a probability depending on the change in the (conditional) posterior probability over all chains induced by the proposed change. The acceptance probability also depends on the temperature of each chain.

While the initial value of `ls` is user-defined (`G_ADMH_LS` see Table 1), it is adapted along the MCMC run to control the acceptance rate of the Metropolis-within-Gibbs sampler and to keep it close to its desired value (user-defined; `G_ADMH_OPTIMAL` in Table 1). Every n_{batch} sweeps (also user-defined, `G_N_BATCH` see Table 1), the value of `ls` is updated as follows:

- If the acceptance rate for τ over the past n_{batch} sweeps is below the target value, the updated value for `ls`, `ls`*, is

$$\texttt{ls}^* = \max\{G_0, \texttt{ls} - \delta\},$$

where G_0, the lower bound for `ls`, is either specified by the user (`G_M_MIN` see Table 1), or by default is automatically set to $-\log(p)/2$ at the initialization. Similarly, G_1, the upper bound for `ls` is either user-specified (`G_M_MAX` see Table 1) or set to $\log(p)/2$. The value of δ, depends on the current sweep (to ensure ergodicity through diminishing adaptation), and on a constant δ_g which is calculated at the initialization of the adaptive Metropolis-within-Gibbs sampler using G_0, G_1, the length of the burn-in and n_{batch}, the number of sweeps between two updates of `ls`.

- If that same acceptance rate is greater than the desired value, the updated value for `ls`, `ls`*, is

$$\texttt{ls}^* = \max\{G_1, \texttt{ls} + \delta\},$$

where G_1 and δ are defined as above.

3.4. Overview of the iterative algorithm

Our algorithm can be subdivided into three main steps:

1. **Data loading and hyper-parameters initialization**

   During this first step, **R2GUESS** will read the input matrices: the predictor matrix X, the outcome matrix Y, the `xml` parameter file compiling default and amendable parameters (listed in Table 1) and modeling options specified in the command line (e.g., *a priori* expected model size, number of chains, ...). Unless specified by an additional input file listing the variables to be included in the initial model, initial values for γ are derived from a stepwise regression whose parameters are user-defined (see Table 1). When multiple outcomes are considered, stepwise regression is performed on each outcome separately and the initial model is defined as the union of the result from the q stepwise models.

2. **MCMC sampler**

   The MCMC sampler will run for S sweeps, where S is specified in the command line together with the number of sweeps to be discarded to allow for burn-in. At each sweep, the following sequences of moves will be repeated:

   a) *Local moves*
   At every `GIBBS_n_BATCH` sweep, a full Gibbs scan will be performed, to improve mixing. In addition, at each sweep an FSMH move will be performed with probability `P_MUTATION`.

   b) *Global moves*
   If a local FSMH move has not been selected, a crossover move will be performed. Next, a delayed rejection or all exchange move will be attempted. During the burn-in, only the delayed rejection move is performed, and once completed, all exchange moves are also enabled and selected with probability 1− `P_DR`.

c) *Sampling the selection coefficient* τ

τ is also updated at each sweep. Parameters of the proposal density are adapted along the MCMC run, every `G_N_BATCH` sweeps.

d) *Temperature placement (during burn-in only)*

Every `TEMP_N_BATCH` sweeps (during burn-in), the temperature ladder is updated such that the acceptance rate of the delayed rejection moves converges towards its desired value.

3. **Post-processing**

Once the MCMC run is completed, several calculations are required to describe the models visited and assess their relevance. These calculations are detailed in Section 4.

4. Post-processing calculations

Once the burn-in is completed, **R2GUESS** samples, at each of the S sweeps, a realization of both random variables γ and τ from the target distribution

$$p(\gamma, \tau|Y) \propto p(Y|\gamma, \tau)p(\gamma)p(\tau).$$

As illustrated in Table 2, each of the S samples $\left(\gamma^{(s)}, \tau^{(s)}\right)_{s=1,\ldots,S}$ is associated with a joint posterior probability $p(\gamma^{(s)}, \tau^{(s)}|Y)$.

From these results, it is possible to infer several summary statistics measuring the relevance of each model visited, the contribution of each predictor to explain the (univariate or multidimensional) outcome, as well as the strength of the association marginally linking each predictor and the outcome.

Using the history of all visited models during the MCMC run, the post-processing procedure returns the unique list of models visited together with the number of times each unique model was visited. If the null model (i.e., the model without any predictors) or any of the models with a single predictor were not visited during the MCMC run, these are added to the list.

4.1. Assessing the relevance of visited models

Model relative importance

From the entire set of models visited, the list of unique models visited is derived. The list is complemented by the null model and all univariate models, even if they have not been visited along the MCMC run, defining an ensemble of S^* unique models. Each model in S^* is associated to its estimated posterior probability

$$\widetilde{p}(\gamma^{(s)}|Y) = C^{-1}p(Y|\gamma^{(s)})p(\gamma^{(s)}) \tag{14}$$

with $s \in S^*$ and $C = \sum_{s\in S^*} p(Y|\gamma^{(s)})p(\gamma^{(s)})$. The MCMC approximation of $p(Y|\gamma^{(s)}) = \int p(Y|\gamma^{(s)}, \tau)p(\tau)d\tau$ is $p(Y|\gamma^{(s)}) = \frac{1}{S}\sum_{r\in S} p(Y|\gamma^{(s)}, \tau^{(r)})$. For computational efficiency, we use a further approximation in **R2GUESS**: $\frac{1}{S}\sum_{r\in S} p(Y|\gamma^{(s)}, \tau^{(r)}) \approx p(Y|\gamma^{(s)}, \mathsf{E}(\tau|Y))$, in

Variable name	Default	Description
General parameters		
`<MAX_P_GAM_FACTOR>`	10	The factor (F) defining the maximal model size ($p_{\max}$) given the user-defined *a priori* expected and standard deviation of the (un-truncated) model size ($\mathsf{E}(p_\gamma)$ and σ_γ) $$p_{\max} = \mathsf{E}(p_\gamma) + \sigma_\gamma \times F.$$
Parameters of the stepwise regression		
`<N_P_VALUE_ENTER>`	0.01	The maximum nominal p value for a term to be added.
`<N_P_VALUE_REMOVE>`	0.01	The minimum nominal p value for a term to be removed.
Setup parameters for the moves		
`<GIBBS_N_BATCH>`	500	Number of sweeps between two full Gibbs scans.
`<P_MUTATION>`	0.5	The probability to perform the FSMH move at each sweep.
`<P_SEL>`	0.5	The threshold on the cumulative Boltzmann weights.
`<P_CSRV_R>`	0.375	The threshold for the correlation coefficient ρ_0 to be considered in the block crossover move.
`<K_MAX>`	2	Maximum number of breakpoints in the crossover move. This number also defines the number of different crossover moves enabled.
`<P_DR>`	0.5	The probability to perform a DR move among the two possible exchange moves.
`<G_ADMH_OPTIMAL>`	0.44	The target acceptance rate for τ.
`<G_N_BATCH>`	100	The number of sweeps between two adaptations of the standard deviation of the proposal for τ.
`<G_ADMH_LS>`	0	Initial value for the log standard deviation of the τ proposal.
`<G_M_MIN>`	$-\log p/2$	Lower bound for the log standard deviation of the τ proposal.
`<G_M_MAX>`	$\log p/2$	Upper bound for the log standard deviation of the τ proposal.
`<B_T>`	2	Initial value for the argument b for the temperature ladder.
`<A_T_DEN_INF_5K>`	2	Initial value for the argument a for the temperature ladder. This value is considered if $p < 5,000$.
`<A_T_DEN_5_10K>`	4	Initial value for the argument a for the temperature ladder. This value is considered if $5,000 \leq p < 10,000$.
`<A_T_DEN_SUP_10K>`	2	Initial value for the argument a for the temperature ladder. This value is considered if $p \geq 10,000$.
`<TEMP_N_BATCH>`	50	Number of DR moves between temperature placement.
`<TEMP_OPTIMAL>`	0.5	Optimal acceptance rate for the DR move. This value is used in the temperature placement.
`<M_MIN>`	1.0	Lower bound for the value of `b_t` in the temperature placement.
`<M_MAX>`	4.0	Upper bound for the value of `b_t` in the temperature placement.

Table 1: Fields contained in the `xml` parameter file.

Sweep	γ	τ	$p(\gamma, \tau \vert Y)$
1	$\gamma^{(1)}$	$\tau^{(1)}$	$p(\gamma^{(1)}, \tau^{(1)} \vert Y)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
S	$\gamma^{(S)}$	$\tau^{(S)}$	$p(\gamma^{(S)}, \tau^{(S)} \vert Y)$

Table 2: Output from the MCMC run.

which we plug-in the posterior mean of τ, $\mathsf{E}(\tau|Y) = \frac{1}{S}\sum_{s\in S}\tau^{(s)}$. Provided that $p(\tau|Y)$ is log-concave, the bias incurred by this approximation is negligible and it does not have a major effect on the derivation of the relative importance of unique visited models and their ranking.

For simplicity, we refer to model relative importance as model posterior probability (MPP).

Jeffreys' scale

The MPP measuring the importance of each (unique) visited model among the whole set of visited models can be complemented by the Bayes factor (BF, Kass and Raftery 1995), which measures how strongly data support one particular model versus an alternative one.

We define $BF(\gamma^1;\gamma^0) = p(Y|\gamma^1)/p(Y|\gamma^0)$, where γ^1 and γ^0 are two competing models and $p(Y|\gamma^1)$ and $p(Y|\gamma^0)$ their respective marginal probabilities. Setting γ^1 as the model under investigation visited (γ^m), and γ^0 as the null model ($\gamma^\varnothing$), this Bayes factor measures the evidence provided by the data to support the model considered with respect to the null model, and the corresponding Jeffreys' scale of evidence (Kass and Raftery 1995) defined as $\log_{10}(BF(\gamma^m;\gamma^\varnothing))$ enables the identification of models that are worth reporting. Note that the MPP relating to the null model is only reported in the output file listing the best models if it is part of them, and (systematically) in the log file.

4.2. Marginal posterior probability of inclusion

The marginal contribution of each predictor to the models can be assessed by calculating the marginal posterior probability of inclusion (MPPI) for each predictor j across S^*. It is defined as the frequency of inclusion re-weighted by the relative importance of each model:

$$\widetilde{p}(\gamma_j = 1|Y) \simeq \sum_{s=1}^{S^*} \mathbb{I}_{\{\gamma_j^{(s)}=1\}} \widetilde{p}(\gamma^{(s)}|Y), \tag{15}$$

where $\gamma_j^{(s)}$ is the binary indicator of the jth covariate at the sth sweep, and the weight $\widetilde{p}(\gamma^{(s)}|Y)$ is defined as in (14). MPPI can be interpreted as the posterior strength of association between a single predictor and the (possibly multivariate) outcome.

In the case of large model space, Monte Carlo estimates of the MPPI derived from the frequency of inclusion over all the visited models have better properties than those obtained from the unique list of visited models (Clyde and Ghosh 2012; Garcia-Donato and Martinez-Beneito 2013). For that reason, **R2GUESS** also provides Monte Carlo estimates defined as

$$\widetilde{p}(\gamma_j = 1|Y) = \frac{1}{S}\sum_{s=1}^{S} \mathbb{I}_{\{\gamma_j^{(s)}=1\}}, \tag{16}$$

where S is the total number of models visited along the MCMC run. To ensure acceptable resolution of the Monte Carlo MPPI estimates, these must be calculated over a large number of iterations.

In order to identify significant predictors, we propose to derive a cut-off value for the MPPI ensuring a false discovery rate (FDR) control at a specified level. This calculation does not depend on the way MPPI are estimated, and relies on the following permutation procedure:

- For a given threshold c, define the number of associations declared significant R (i.e., the number of predictors with MPPI $< c$) based on the original data.
- For each permuted outcome $t = (1, \ldots, T)$ mimicking the null hypothesis of no association, count the number of associations declared significant A_t, which also represents the number of false positive associations found in the tth permuted dataset.
- For a given value of c, the FDR can be estimated over the T permutations by the following ratio
$$\widehat{FDR} = \frac{\widehat{\mathsf{E}(V)}}{R} = \frac{(1/T)\sum_{l=1}^{T} A_t}{R},$$
where R is the number of associations declared significant.
- Define the MPPI threshold c ensuring an FDR control at a specified level (FDR_{target}) as the lowest value satisfying $\widehat{FDR} < FDR_{\text{target}}$.

4.3. Effect size estimate

Given a value of $\gamma^{(s)}$ and $\tau^{(s)}$, the error variance matrix Σ can be simulated from the following inverse Wishart distribution

$$\Sigma^{(s)} | Y, \gamma^{(s)}, \tau^{(s)} \sim IW(d^*, Q^*), \tag{17}$$

where $d^* = d + n$, with d defined in (6), and

$$Q^* = kI_n + Y^\top Y - \frac{\tau^{(s)}}{1+\tau^{(s)}} Y^\top X_{\gamma^{(s)}} \left(X_{\gamma^{(s)}}^\top X_{\gamma^{(s)}} \right)^{-1} X_{\gamma^{(s)}}^\top Y.$$

Denison, Holmes, Mallick, and Smith (2002) also showed that the matrix of regression coefficients B could be simulated as follows

$$\left(B^{(s)} + m^* \right) | Y, \gamma^s, \Sigma^{(s)} \sim N\left(H^*_{\gamma^{(s)}}, \Sigma^{(s)} \right), \tag{18}$$

where

$$H^*_{\gamma^{(s)}} = \frac{\tau^{(s)}}{1+\tau^{(s)}} \left(X_{\gamma^{(s)}}^\top X_{\gamma^{(s)}} \right)^{-1}, \qquad \text{and} \qquad m^* = H^*_{\gamma^{(s)}} X_{\gamma^{(s)}}^\top Y = \frac{\tau^{(s)}}{1+\tau^{(s)}} B^{LS}_{\gamma^{(s)}}.$$

$B^{LS}_{\gamma^{(s)}}$ denotes the $p \times q$ matrix of least square solutions for the set of predictors selected in the sth model visited. In practice, the simulation of the matrix of regression coefficients for a given model (γ^*) relies on the following steps:

1. For a given vector γ^*, define the set of values for τ (noted s_τ) sampled when that model was visited.
2. For each value of τ in s_τ, simulate N_Σ times Σ from (17).
3. For each simulated Σ matrix, simulate N_B regression coefficient matrices B from (18).
4. The posterior distribution of B is then estimated by the empirical distribution of the $N_\Sigma \times N_B$ simulated matrices.

5. Installation

R2GUESS requires a functioning installation of the GNU Scientific Library (**GSL** version 1.12 or later, Galassi *et al.* 2009) and, to use its GPU capabilities, a **CULA**-compatible platform (Humphrey, Price, Spagnoli, Paolini, and Kelmelis 2010). To ensure an efficient use of **R2GUESS**:

1. If not already installed, install the GNU Scientific Library (**GSL**).
2. For users willing to use the GPU features of **R2GUESS**:
   - Check the **CUDA** (NVIDIA Corporation 2015) compatibility of your system's graphics card.
   - Install the **CUDA** drivers that are compatible with your graphics card (freely available from NVIDIA).
   - Install the full **CULA Dense** library (available at no cost for personal academic use).
3. Install the R software environment for statistical computing (R Core Team 2013).

During installation, **R2GUESS** tries to automatically detect the location of all required and optional libraries. GPU capabilities are automatically enabled if the **CUDA** and **CULA Dense** libraries are found on the system. If **CUDA** has not been detected, **R2GUESS** will be installed without GPU support, and linear algebra operations will be performed exclusively on the central processing unit (CPU). The GPU version of **R2GUESS** yields substantial computational improvements for datasets with large n. However, it may be slower than its CPU alternative on small datasets due to extensive data transfers becoming rate limiting. In such cases, GPU capabilities can be disabled by specifying the argument `CUDA = FALSE` to the main function `R2GUESS()`. **R2GUESS** has been developed and tested on the full version of the **CULA Dense** library, which is free for academic use. For any user, there is also a free version of the library, which performs single-precision matrix operations, and is compatible with **R2GUESS**. But the validity and numerical stability of the results in this setting cannot be guaranteed, and hence it is recommended to make use of the full **CULA Dense** library.

6. Usage

In this section we describe how to use **R2GUESS**. Based on a simulated dataset, we will first illustrate the high specificity of our model selection procedure and will investigate its

File name	Description
`$1_$2_sweeps_output_best_visited_models.txt`	Summary statistics about the best visited models: the rank of the model according to its posterior probability; the number of times the model was visited during the entire MCMC run (including the burn-in); the number of variables in the model; the log posterior probability; the posterior probability of the model; the Jeffreys' scale value for the model.
`$1_$2_iter_output_marg_prob_incl.txt`	The marginal posterior probability of inclusion for each predictor.

Table 3: Main output files produced by the C++ source code; `$1` is the file name entered by the user and `$2` is the number of sweeps entered by the user.

computational efficiency. The full capacity of **R2GUESS** will be further investigated by a step-by-step analysis of a realistic dataset embedded in the package. We will exemplify the use of the main functions that are useful to analyze the outputs the code generates (as detailed in Tables 3 and 4).

Our approach offers, as a default, automatic tuning of most hyper-parameters. This tuning has been developed with the aim of improving convergence of the MCMC procedure, and default values compiled in the `xml` parameter file, were defined such that they fit most of the datasets. However, for datasets in which convergence appears slower, these can be user-defined in a personalized parameter file.

Three parameters are required to run the model: the *a priori* expected value $\mathsf{E}(p_\gamma)$ of the (un-truncated) model size, its variance $\mathsf{VAR}(p_\gamma)$, and the number of chains L.

6.1. R2GUESS performance: Simulation study

Simulation model

Based on real genotype data (273,675 single nucleotide polymorphisms, SNPs) obtained in $n = 3,122$ individuals from the Gutenberg health study (GHS; Zeller *et al.* 2010; Bottolo *et al.* 2013), we simulated outcomes ($q = 3$) for different population sizes and numbers of predictors. A sub-matrix X_α was extracted from the original dataset by randomly sampling a subset of rows (individuals) and columns (SNPs). For a given number of true "causal" predictors r, the response matrix Y was simulated from

$$\vec{Y} = \vec{X}_\alpha \vec{B} + \vec{E}, \quad \text{vec}\left(\vec{E}^\top\right) \sim \mathcal{N}_{n\times q}\left(\vec{0}_{n\times q},\, c \times \vec{I}_n \otimes \vec{\Sigma}\right), \tag{19}$$

where $\vec{B}$ is the $r \times q$ matrix of regression coefficients, $\vec{E}$ is the $q \times q$ residual correlation matrix, and c is a scalar controlling the signal over noise ratio.

For all simulations, we assumed strong levels of correlation between the first and second outcomes, and weaker levels for the other pairwise correlations. Specifically, we defined

$$\vec{\Sigma} = \begin{pmatrix} 1 & 0.95 & 0.5 \\ 0.95 & 1 & 0.3 \\ 0.5 & 0.3 & 1 \end{pmatrix}. \tag{20}$$

File name	Description
`$1_$2_sweeps_output_time_monitor.txt`	Each line of the file represents a sweep of the sampler and shows the absolute and average computational time *per* model evaluated during that sweep.
`$1_$2_sweeps_output_gibbs_history.txt`	For every Gibbs scan that was performed the file records the sweep of the sampler, the number of variables that were added in (`0->1`) and the number of variables that were removed (`1->0`).
`$1_$2_sweeps_output_fast_scan_history.txt`	Each time an FSMH move is sampled along the MCMC run, this file records: the number of models evaluated, the number of accepted moves, the number of proposed and accepted inclusion moves and the number of proposed and accepted exclusion moves.
`$1_$2_sweeps_output_cross_over_history.txt`	For all sweeps where a crossover move was sampled, this file reports the type of crossover move sampled: numbers ranging from 1 to `k_max`, the maximum number of breakpoints, code for a k-point crossover, and `k_max + 1` codes for a block crossover). The number of breakpoints is also reported, as well as the two selected chains.
`$1_$2_sweeps_output_all_exchange_history.txt`	At each all exchange move call (`Sweep`), this file simply records the two chains involved.
`$1_$2_sweeps_output_delayed_rejection_history.txt`	As for the all exchange history, this file records the sweeps at which a delayed rejection move was selected (`Sweep`) and the two chains involved.
`$1_$2_sweeps_output_g_history.txt`	This file provides the sampled values for τ for each sweep.
`$1_$2_sweeps_output_g_adaptation_history.txt`	Each time the proposal for τ is adaptively updated, this file records the current sweep, the acceptance rate for τ since the last update acceptance rate, and the log-standard deviation of the proposal distribution of τ.
`$1_$2_sweeps_output_temperature_history.txt`	This file records the sweep at which temperature tuning took place and the resulting temperature for each chain.
`$1_$2_sweeps_output_models_history.txt`	This file describes the models visited along the MCMC run. For each sweep, it presents: the number of variables in the model, the log (conditional) marginal probability the log (conditional) posterior probability and the variables included in the model. All values are as at the end of the sweep and only refer to the first non-heated chain.
`$1_$2_sweeps_output_model_size_history.txt`	In this file the model size for each chain at the end of each sweep is reported.
`$1_$2_sweeps_output_log_cond_post_prob_history.txt`	This file reports the history of the log (conditional) posterior associated to each chain at the end of every sweep.

Table 4: Further output files produced by the C++ source code; `$1` is the file name entered by the user and `$2` is the number of sweeps entered by the user.

We also used the following matrices $\vec{B}$

$$\vec{B} = \begin{pmatrix} 0.2 & 0.1 & 0.075 \\ 0.1 & 0.075 & 0.1 \end{pmatrix} \quad \text{, and} \quad \vec{B} = \begin{pmatrix} 0.2 & 0.1 & 0.075 \\ 0.1 & 0.075 & 0.1 \\ 0.2 & 0.1 & 0.075 \\ 0.1 & 0.075 & 0.1 \\ 0.075 & 0.1 & 0.2 \\ 0.1 & 0.2 & 0.1 \\ 0.075 & 0.1 & 0.2 \\ 0.1 & 0.2 & 0.1 \end{pmatrix}, \tag{21}$$

for $r = 2$ and 8, respectively. The value of c was calibrated such that the expected proportion of variance explained for each trait did not exceed 5% to mimic small effects usually found in GWAS.

Our simulation presented in (19) was further generalized in order to account for the effect of potential confounders (noted $\vec{C}$),

$$\vec{Y} = \vec{X}_\alpha \vec{B} + \vec{C}\vec{B_C} + \vec{E}, \quad \text{vec}\left(\vec{E}^\top\right) \sim \mathcal{N}_{n\times q}\left(\vec{0}_{n\times q},\, c \times \vec{I}_n \otimes \vec{\Sigma}\right), \tag{22}$$

where $\vec{B_C}$ is an $k \times q$ matrix compiling the regression coefficients linking the k confounders in $\vec{C}$ and the q outcomes in $\vec{Y}$. As above, c is calibrated such that the expected proportion of variance explained is no greater than 5%.

We ran our simulator including confounders using the same parameters as above for the simulation containing two "causal" SNPs ($r = 2$). In that setting, k was set to 2 and we chose

$$\vec{B_C} = \begin{pmatrix} 0.5 & 0.7 & 0.6 \\ 0.7 & 0.5 & 0.6 \end{pmatrix}, \tag{23}$$

assuming a strong confounding effect of the two covariates. The two covariates are generated from two independent standard Normal distributions.

Analyzing data

Based on our simulation model, we created two datasets setting $n = 1,500$, and $p = 5,000$. These were analyzed using **R2GUESS** setting $\mathsf{E}(p_\gamma) = 5$, $\mathsf{VAR}(p_\gamma) = 5$, and $L = 3$. The algorithm ran for 30,000 iterations (including 10,000 iterations for burn-in).

Sparse results provided by **R2GUESS** are illustrated in Figure 6, where the per-SNP MPPI is plotted for $r = 2$ and 8 (Figures 6(a) and (b), respectively). Plots were obtained from the `plotMPPI()` function. In the first simulation ($r = 2$), both causal SNPs used for the simulation were associated to high MPPI values. Specifically, they were both included in all of the 100 best models visited (MPPI = 1), irrespective of their effect sizes. The corresponding MPPI cut-off value ensuring *FDR* control at 5% was estimated using the permutation procedure from `Analysis.Permutation()` to $\text{MPPI}_{FDR} = 0.46$, confirming that both causal SNPs were found significant. In the second simulation, 6 out of 8 causal SNPs were included in all best models visited (MPPI = 1). Among the two remaining signals, one had lower MPPI values but was found significant (SNP 4209, MPPI > 0.89). SNP 3682 had an MPPI lower than 0.01 but corresponded to the 8th highest value.

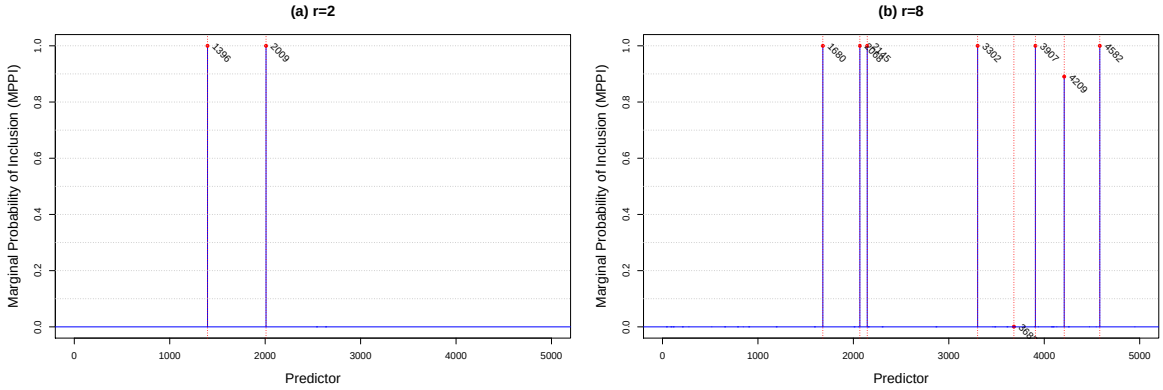

Figure 6: MPPI plots for two simulated datasets setting $n = 1,500$, $p = 5,000$, and $r = 2$ (a), and $r = 8$ (b). Results were obtained using **R2GUESS** for 30,000 iterations, setting $\mathsf{E}(p_\gamma) = 5$, $\mathsf{VAR}(p_\gamma) = 5$, and $L = 3$. Vertical dotted lines represent the "causal" SNPs used for the simulation, and red dots represent SNPs which were included in the best 100 models visited.

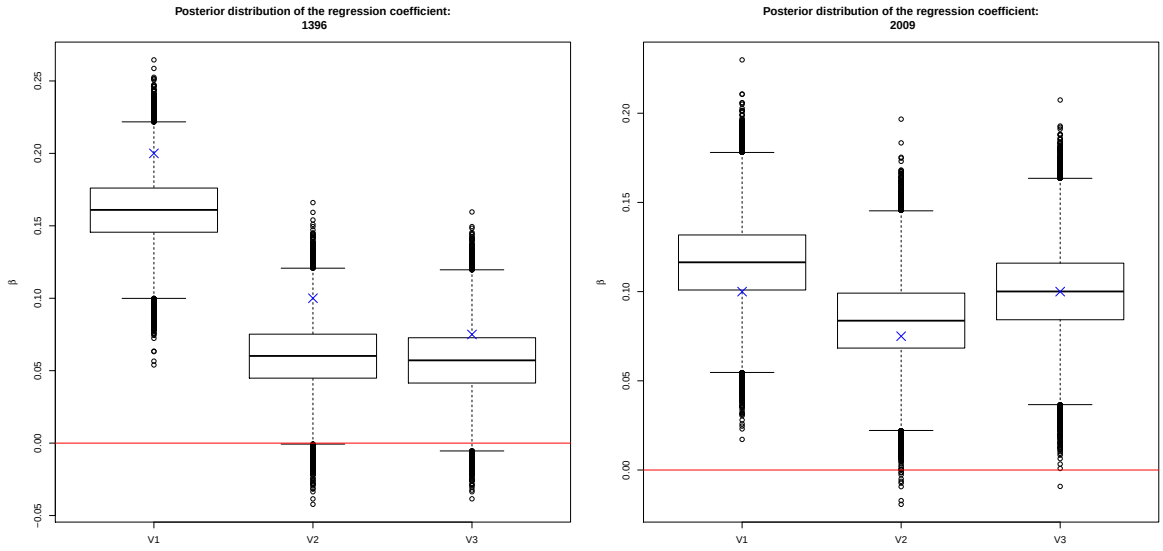

Figure 7: Boxplots summarizing the posterior distribution of the regression coefficients for SNPs 1396 (a) and 2009 (b). Regression coefficients measure the relation between each SNP and Y_1, Y_2, and Y_3 in the best model visited. Blue crosses represent the true value used in the simulation.

In Figure 7, the posterior distribution of the regression coefficients derived from the best model is summarized, using the `samplebeta()` function. Results are presented for the simulated dataset comprising two causal SNPs (1396 and 2009) which were allocated regression coefficients with respect to Y_1, Y_2, and Y_3 as described in (21).

From Figures 7(a) and (b) it is apparent that **R2GUESS** successfully inferred the effect size of the causal SNPs with relatively narrow distributions containing the true value used for the simulation, particularly for SNP 2009. Greater error is observed for SNP 1396, which can be

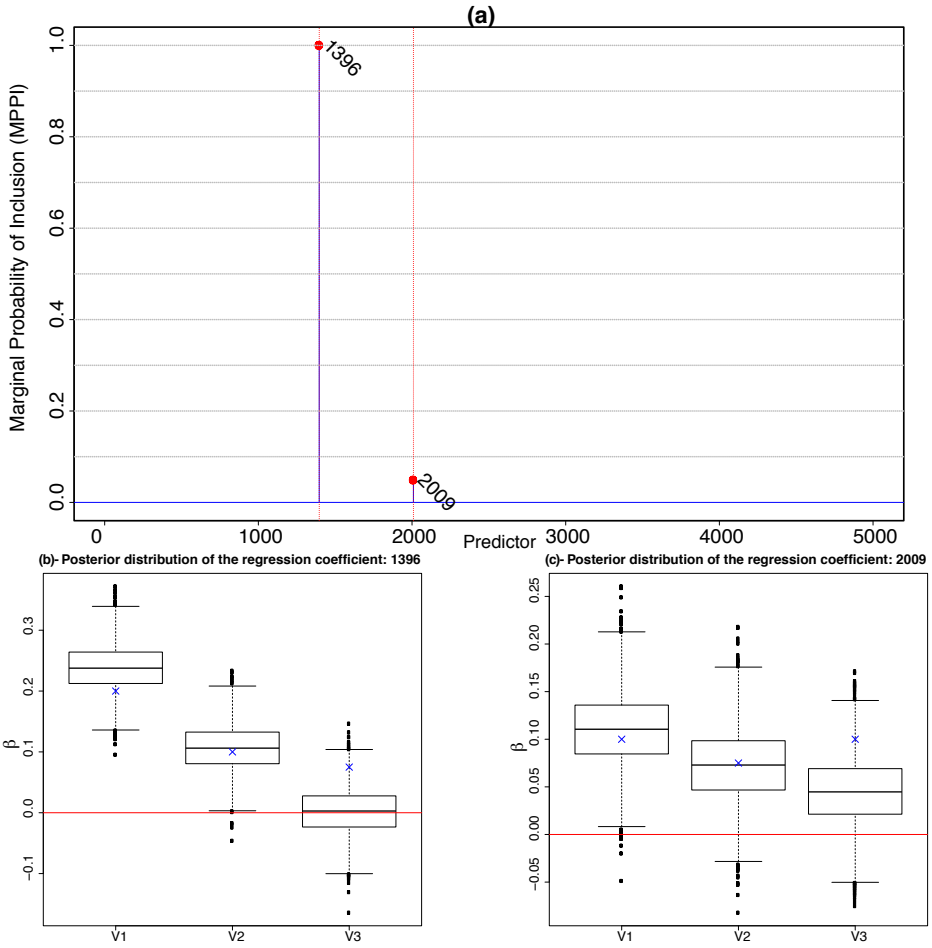

Figure 8: Summary of the **R2GUESS** outputs for the simulated dataset setting $n = 1,500$; $p = 5,000$; and $r = 2$, and $c = 2$. Results were obtained for 30,000 iterations, setting $\mathsf{E}(p_\gamma) = 2$, $\mathsf{VAR}(p_\gamma) = 2$, and $L = 3$. In (a) the per-SNP MPPI is represented and vertical dotted lines represent the causal SNPs used for the simulation. Red dots represent SNPs which were included in the best 100 models visited. Boxplots summarizing the posterior distribution of the simulated regression coefficients for SNPs 1396 (b) and 2009 (c). Blue crosses represent the true value used in the simulation. Figures 8(b) and (c) are based on the second best model (MPP = 0.19) which was the best model including both SNPs.

attributed to our simulation setup resulting in higher residual variance for that SNP.

R2GUESS was also run on the simulated dataset including two confounders as detailed in (23); results are summarized in Figure 8.

Despite strong confounding effects, **R2GUESS** identified both causal SNPs used for the simulation (Figure 8(a)). As in the original simulation, SNP 1396 was included in all 100 best models (MPPI = 1). In contrast, SNP 2009 showed lower MPPI than in the model without confounding. This is due to the fact that while adjusting for the strong effect of confounders, the best model (MPP = 0.57) did not include SNP 2009. That second SNP only appeared in the second best model (with a MPP = 0.38). In addition. Figures 8(b) and (c) show highly consistent effect size estimates for the model including confounders, with the exception of

the third outcome which corresponds to weaker effect size and small proportion of variance explained (simulated proportion of explained variance was around 5×10^{-5}, which was 40 and 10 fold lower than for the first and second outcomes respectively).

Computational performance: GPU/CPU choice

Enabling GPU capabilities in **R2GUESS** will re-route computationally expensive matrix operations to the GPU and will therefore generate extensive data transfer with the CPU. In some cases, typically for small datasets, matrix operations are not rate limiting, and data transfer between CPU and GPU represents a major bottleneck, leading to the GPU version being slower than the CPU one. In these instances it may thus be desirable to disable GPU support. This can be easily achieved by specifying the argument `CUDA = FALSE` in the call to the function `R2GUESS()`.

To guide the choice of the user, we extended our simulations to several values of n ranging from 100 to 3,000, p from 500 to 250,000 and three different values of $r = 2$, 8, and 15. For each of the 60 simulated datasets, we ran **R2GUESS** and recorded computing times. As expected, computing time increases with n, and to a lesser extent with p and r in both versions of **R2GUESS**. Relative computing performances for the CPU and GPU versions of **R2GUESS** seem to generally favor the CPU version for $n \leq 500$ irrespective of p and r. Computational benefits of the GPU version of **R2GUESS** are clear for large datasets ($n \geq 3,000$) and are increasing with p. As an indication, genome-wide runs reported in Bottolo *et al.* (2013) ($n = 3,122$, $p = 273,675$) took around 80 hours for 60,000 iterations using the GPU version, while after 200 hours, the CPU version had not reached 15,000 iterations.

6.2. R2GUESS in practice: Real data example

The example dataset included in **R2GUESS** originates from a larger study (Heinig *et al.* 2010) from which we selected the `Hopx` genes, as in Petretto *et al.* (2010). For each gene, we investigated the ability of **R2GUESS** to identify a parsimonious set of predictors that explains the joint variability of gene expression in four tissues (adrenal gland, fat, heart, and kidney). The dataset consists of 770 SNPs in 29 inbred rats as a predictor matrix ($n = 29$, $p = 770$), and the 29 measured expression levels in the 4 tissues as the outcome ($q = 4$).

In our example, we follow the parametrization of Petretto *et al.* (2010) and fix $\mathsf{E}(p_\gamma) = 5$ and $\mathsf{VAR}(p_\gamma) = 5$. This supposes an *a priori* number of predictors ranging from 0 to 12, with increasing penalization of values outside this range. We also set $L = 3$.

Correlation plot

We start by exploring the pairwise correlation structure among the four tissues (Figure 9(a)), and then resort to clustered image maps (CIM; Dejean *et al.* 2013; Figure 9(b)).

```
R> library("R2GUESS")
R> data("data.Y.Hopx", package = "R2GUESS")
R> data("data.X", package = "R2GUESS")
R> pairwise.correlation(data.Y.Hopx)
R> plotcim.explore(data.X, data.Y.Hopx)
```

This color-coded representation of the correlation levels provides an overall visualization of

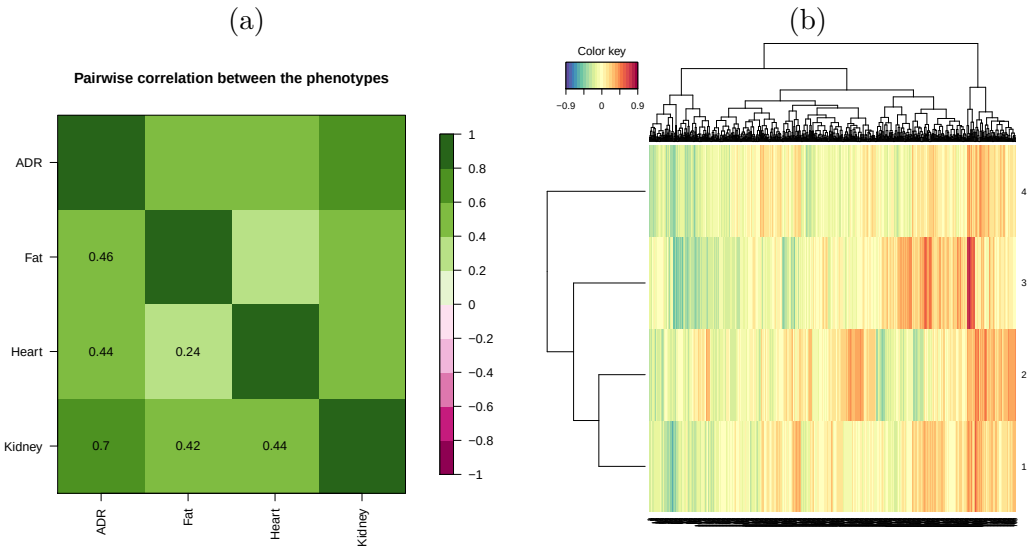

Figure 9: (a) Pairwise correlations among the four tissues; (b) Clustered image map summarizing the correlation between SNPs and the multivariate phenotype.

the correlation structures linking predictors (SNPs) and phenotypes (expression levels).

Bayesian variable selection

The function `R2GUESS()` is used to run the algorithm by calling the C++ executable and compiles inputs (including parameters detailed in Table 1), and outputs (detailed in Table 3) into an object of class '`ESS`' that can be further analyzed using, among other functions, `summary()`, `print()`, and `plot()`. The object can also contain an annotation file describing the predictors (e.g., SNP names, positions, etc.).

The following command lines give an example of how to run **R2GUESS** on the built-in example dataset. The model was run for 11,000 sweeps including 1,000 sweeps for burn-in.

```
R> path.output <- paste(getwd(), "/", sep = "")
R> path.input <- path.output
R> path.par <- paste(system.file("extdata", package = "R2GUESS"), "/",
+    sep = "")
R> file.par.Hopx <- "Par_file_example_Hopx.xml"
R> nsweep <- 30000
R> burn.in <- 10000
R> root.file.output.Hopx <- "Example-GUESS-Y-Hopx"
R> label.Y <- c("ADR", "Fat", "Heart", "Kidney")
R> data("MAP.file", package = "R2GUESS")
R> modelY_Hopx <- R2GUESS(data.Y.Hopx, data.X, path.input, path.output,
+    path.par, file.par = file.par.Hopx, nsweep = nsweep, burn.in = burn.in,
+    root.file.output = root.file.output.Hopx, time = TRUE, top = 100,
+    history = TRUE, Egam = 5, Sgam = 5, label.Y = label.Y, choice.Y = 1:4,
+    nb.chain = 3, conf = NULL, cuda = FALSE, MAP.file = MAP.file, seed = 7)
```

These commands will store results in the `modelY_Hopx` object, which can be visualized using `summary()` command to obtain a list of the best models (the number of output models is user-defined).

```
R> summary(modelY_Hopx, 5)
```

```
  Rank nVisits FirstVisit nEvalBefore1st ModeSize logCondPost postProb
1    1     858       2376         521236        6      -86.29   0.2600
2    2     150       3012         545385        7      -87.83   0.0557
3    3      78       5795         639888        3      -87.89   0.0526
4    4      18      11492         816797        1      -88.32   0.0340
5    5      41       3881         574232        2      -88.50   0.0287
  jeffrey                                                 modelName
1   21.51          D3Mit16 D6Cebrp40s27 D10Rat33 Dcp1 D11Mit4 D14Mit3
2   22.96 D3Mit16 D6Cebrp40s27 D10Rat33 Dcp1 D11Mit4 D14Mit3 D15Rat21
3   14.02                                D2Cebr204s17 D13Mit3 D14Mit3
4    8.73                                                   D14Mit3
5    11.2                                         D2Cebr204s17 D14Mit3
```

Models are described through several key features:

- their rank with respect to their posterior probability (`Rank`);
- the number of times they were visited along the run (`nVisits`);
- the sweep at which they were first visited (`FirstVisit`);
- the number of alternative models visited before their first visit (`nEvalBefore1st`);
- the number of features they include (`ModelSize`);
- their log conditional posterior probability (`logCondPost`);
- their posterior probability (`postProb`);
- the value of their Jeffreys' scale of evidence (`jeffrey`);
- the list of covariates (or their label if an annotation file is provided) they include.

Outputs from **R2GUESS** are stored as text files in the folder defined by `path.output`. 'ESS' objects can be (re)-constructed from these files using the `as.ESS.object()` function.

```
R> rm(modelY_Hopx)
R> modelY_Hopx <- as.ESS.object(dataY = "data-Y-ALL-C-CODE.txt",
+    dataX = "data-X-C-CODE.txt", path.input = path.input,
+    path.output = path.output, label.X = NULL,
+    root.file.output = root.file.output.Hopx, label.Y = label.Y,
+    path.par = path.par, file.par = file.par.Hopx, MAP.file = MAP.file)

The run is ok
You can now analyse the results
```

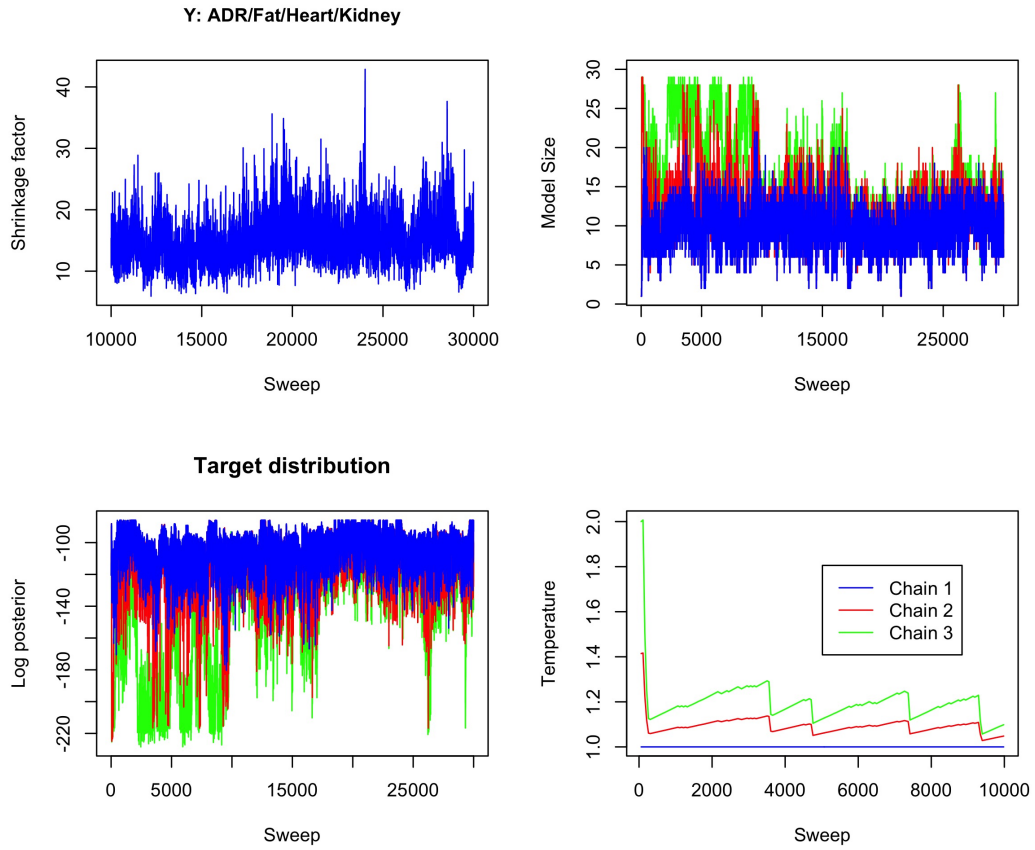

Figure 10: Diagnostic plots to assess the convergence of the algorithm produced by `plot()`.

```
R> class(modelY_Hopx)

[1] "ESS"
```

Convergence of the algorithm

The `plot()` function automatically produces a set of four diagnostic plots from an '`ESS`' object to assess the convergence and the mixing of the run.

```
R> plot(modelY_Hopx)
```

The four plots produced are:

- the trace of the shrinkage factor τ;
- the traces of the model sizes for the three different chains, showing, as expected, increased model sizes in heated chains;
- the traces of the logarithm of the posterior distributions $\log p(\gamma|Y,\tau)$ for the three different chains. A good balance between overlap and distance between these traces

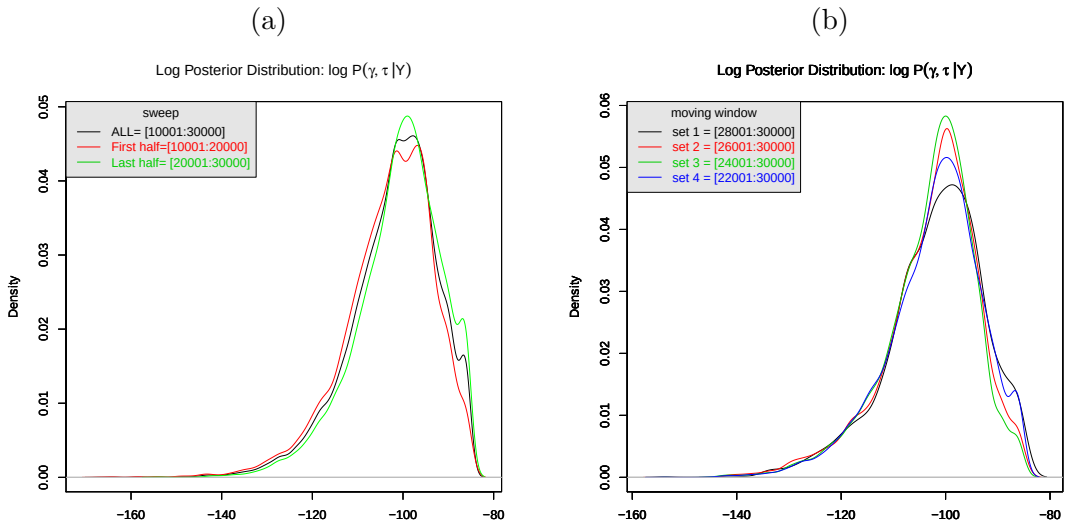

Figure 11: Diagnostic plots to assess the convergence of the algorithm produced by `check.convergence()`.

indicates good exchange of information between chains (good mixing) and minimal redundancies between chains;

- the traces of the temperatures of the three different chains (during burn-in only). Abrupt changes in the temperature history correspond to the updates of the temperature placement during burn-in.

Our example (Figure 10) suggests that convergence to the corresponding stationary distributions has been achieved. Additionally, the degree of overlap among the chains in the bottom plot on the left-hand side indicates a good level of information exchange between them.

Convergence of the algorithm can be further investigated using the `check.convergence()` function, which plots the density of the logarithm of the posterior distribution $\log p(\gamma|Y, \tau)$ at different stages of the run.

```
R> check.convergence(modelY_Hopx)
```

Our example produces two additional plots:

- Figure 11(a) shows the kernel density estimates of $\log p(\gamma|Y, \tau)$ for:
  - all sweeps;
  - the first half of the sweeps;
  - the second half of the sweeps.
- Figure 11(b) shows the sames density estimates based on user-defined sliding windows.

The strong overlap of these densities confirms a quick and efficient convergence of the algorithm.

Selection of covariates

As illustrated in Section 6.1, **R2GUESS** offers the possibility to plot the MPPIs of all variables using the `plotMPPI()` function. By default `plotMPPI()` will use re-normalized MPPI estimates. Monte Carlo estimates can alternatively be plotted by using the `useMC` option and, in our example, produces highly comparable estimates.

It is possible to assess significance of the findings by specifying an MPPI cut-off value which controls the empirical *FDR* at a specified level (`FDR.permutation()` function). This function will run `R2GUESS` for `Npermut` times on permuted datasets. Although the run can be parallelized on `nbcpu` processors, it may become time consuming when applied to large datasets. The function will provide a cut-off value for both re-weighted and Monte Carlo estimates of the MPPI.

```
R> perm.input <- sprintf("%s/perm_input/", getwd())
R> perm.output <- sprintf("%s/perm_output/", getwd())
R> FDRthresh <- FDR.permutation(x = modelY_Hopx, path.input = perm.input,
+    path.output = perm.output, Npermut = 50, nbcpu = 5)
R> plotMPPI(modelY_Hopx, threshold.model = 100,
+    threshold.variable = FDRthresh\$cutoff_int)
```

Our example first calculates the *FDR*-corrected threshold, and produces Figure 12, in which the strongest signals (with an MPPI > `threshold.variable`, here set to the *FDR* corrected threshold). are highlighted. This plot indicates that `D14Mit3` is significantly associated to the 4 levels of expression, and that the 5 other SNPs found with high MPPI are close to the *FDR*-corrected threshold.

Visualization of the best models

R2GUESS also includes some visualizing tools to describe the best models visited. The `plotmodel()` function produces a plot describing which variables are included in each of the best models visited (a), and `plotvariable()` quantifies the proximity between best models visited by assessing the proportion of overlapping variables between each pair of best models (b).

```
R> plotmodel(modelY_Hopx, 20)
R> plotvariable(modelY_Hopx, 20)
```

Figure 13(a) shows that SNP `D14Mit3` has been included in all twenty strongest models. Of these, 19 include more than one SNP. Figure 13(b) shows a proximity matrix defined as the ratio of the number of common variables within each pair of models and the size of the smallest of the two models.

Acknowledgments

The authors wish to thank Paul Newcombe and Aida Moreno Moral for their extensive testing and precious comments on the package and documentation. Marc Chadeau-Hyam acknowledges the European FP7 projects Exposomics (grant agreement 308610 to P. Vineis) and Envirogenomarkers (grant agreement 226756 to S.A. Kyrtopoulos). Leonardo Bottolo was

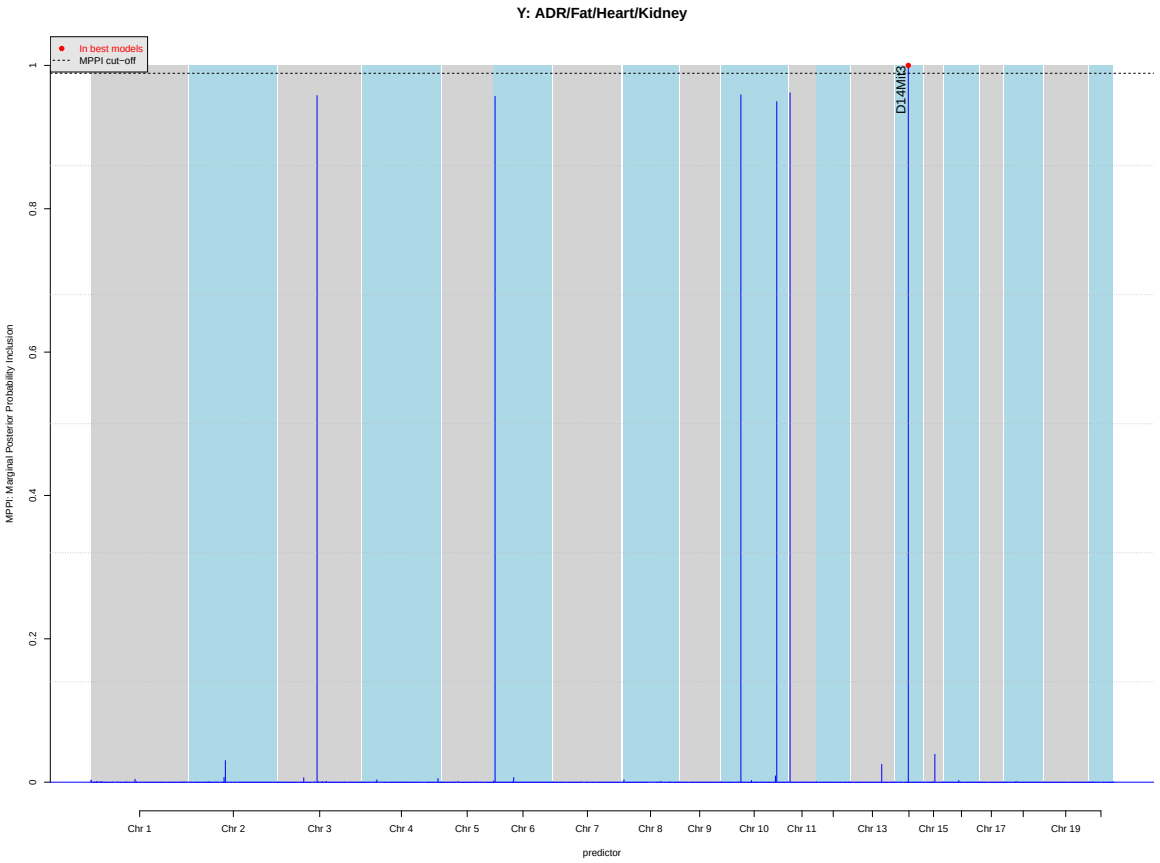

Figure 12: Plot of the MPPIs produced by `plotMPPI`. Estimates are based on our re-normalization procedure.

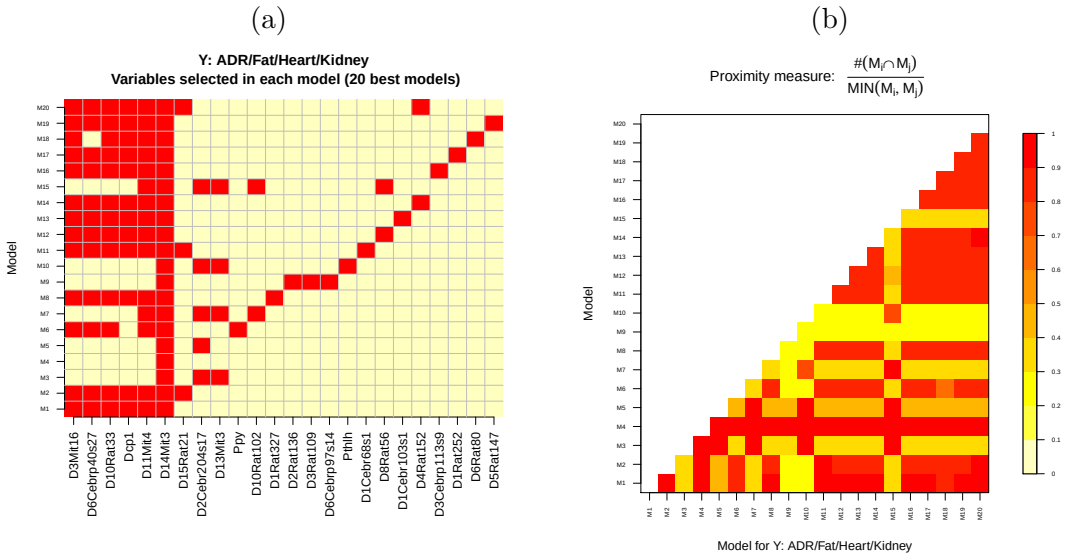

Figure 13: (a) Variables included in the best models; (b) Proximity between the best models.

supported by EPSRC Mathematics Platform grant (grant agreement EP/I019111/1) and FoNS Collaboration Kick-start funding scheme. Sylvia Richardson and Leonardo Bottolo acknowledge support from MRC grant number G 1002319.

References

Balding D (2006). "A Tutorial on Statistical Methods for Population Association Studies." *Nature Reviews Genetics*, **7**(10), 781–791. doi:10.1038/nrg1916.

Bottolo L, Chadeau-Hyam M, Hastie D, Langley S, Petretto E, Tiret L, Tregouet D, Richardson S (2011a). "**ESS++**: A C++ Objected-Oriented Algorithm for Bayesian Stochastic Search Model Exploration." *Bioinformatics*, **27**(4), 587–588.

Bottolo L, Chadeau-Hyam M, Hastie D, Zeller T, Liquet B, Newcombe P, Yengo L, Wild P, Schillert A, Ziegler A, Nielsen S, Butterworth A, Ho W, é RC, Munzel T, Tregouet D, Falchi M, Cambien F, Nordestgaard B, Fumeron F, Tybjærg-Hansen A, Froguel P, Danesh J, Petretto E, Blankenberg S, Tiret L, Richardson S (2013). "GUESS-Ing Polygenic Associations with Multiple Phenotypes Using a GPU-Based Evolutionary Stochastic Search Algorithm." *PLOS Genetics*, **9**(8), e1003657. doi:10.1371/journal.pgen.1003657.

Bottolo L, Petretto E, Blankenberg S, Cambien F, Cook S, Tiret L, Richardson S (2011b). "Bayesian Detection of Expression Quantitative Trait Loci Hot Spots." *Genetics*, **189**(4), 1449–1459. doi:10.1534/genetics.111.131425.

Bottolo L, Richardson S (2010). "Evolutionary Stochastic Search for Bayesian Model Exploration." *Bayesian Analysis*, **5**(3), 583–618. doi:10.1214/10-ba523.

Cantor R, Lange K, Sinsheimer J (2010). "Prioritizing GWAS Results: A Review of Statistical Methods and Recommendations for Their Application." *American Journal of Human Genetics*, **86**(1), 6–22. doi:10.1016/j.ajhg.2009.11.017.

Chadeau-Hyam M, Campanella G, Jombart T, Bottolo L, Portengen L, Vineis P, Liquet B, Vermeulen R (2013). "Deciphering the Complex: Methodological Overview of Statistical Models to Derive OMICS-Based Biomarkers." *Environmental and Molecular Mutagenesis*, **54**(7), 542–557. doi:10.1002/em.21797.

Clyde M, Ghosh J (2012). "Finite Population Estimators in Stochastic Search Variable Selection." *Biometrika*, **99**(4), 981–988.

Dawid A (1981). "Some Matrix-Variate Distribution Theory: Notational Considerations and a Bayesian Application." *Biometrika*, **68**(1), 265–274. doi:10.2307/2335827.

Dejean S, Gonzalez I, Le Cao KA, Monget P, Coquery J, Yao F, Liquet B (2013). ***mixOmics**: Omics Data Integration Project*. R package version 4.1-4, URL http://CRAN.R-project.org/package=mixOmics.

Denison D, Holmes C, Mallick B, Smith A (2002). *Bayesian Methods for Nonlinear Classication and Regression*. John Wiley & Sons.

Do KA, Müller P, Vannucci M (eds.) (2006). *Bayesian Inference for Gene Expression and Proteomics*. Cambridge University Press. doi:10.1017/cbo9780511584589.

Galassi M, Davies J, Theiler J, Gough B, Jungman G, Alken P, Booth M, Rossi F (2009). *GNU Scientific Library Reference Manual*. Network Theory Ltd, 3rd edition. URL http://www.gnu.org/software/gsl/.

Garcia-Donato G, Martinez-Beneito M (2013). "On Sampling Strategies in Bayesian Variable Selection Problems With Large Model Spaces." *Journal of the American Statistical Association*, **108**(501), 340–352. doi:10.1080/01621459.2012.742443.

George E, McCulloch R (1993). "Variable Selection via Gibbs Sampling." *Journal of the American Statistical Association*, **88**(423), 881–889. doi:10.2307/2290777.

Guan Y, Stephens M (2011). "Bayesian Variable Selection Regression for Genome-Wide Association Studies, and Other Large-Scale Problems." *The Annals of Applied Statistics*, **5**(3), 1780–1815. doi:10.1214/11-aoas455.

Hans C, Dobra A, West M (2007). "Shotgun Stochastic Search for "Large p" Regression." *Journal of the American Statistical Association*, **102**(478), 507–516. doi:10.1198/016214507000000121.

Heinig M, Petretto E, Wallace C, Bottolo L, Rotival M, Lu H, Li Y, Sarwar R, Langley S, Bauerfeind A, Hummel O, Lee Y, Paskas S, Rintisch C, Saar K, Cooper J, Buchan R, Gray E, Cyster J, Erdmann J, Hengstenberg C, Maouche S, Ouwehand W, Rice C, Samani N, Schunkert H, Goodall A, Schulz H, Roider H, Vingron M, Blankenberg S, Munzel T, Zeller T, Szymczak S, Ziegler A, Tiret L, Smyth D, Pravenec M, Aitman T, Cambien F, Clayton D, Todd J, Hubner N, Cook S (2010). "A Trans-Acting Locus Regulates an Anti-Viral Expression Network and Type 1 Diabetes Risk." *Nature*, **467**(7314), 460–464. doi:10.1038/nature09386.

Humphrey J, Price D, Spagnoli K, Paolini A, Kelmelis E (2010). "**CULA**: Hybrid GPU Accelerated Linear Algebra Routines." In *Proceedings of the SPIE 7705, Modeling and Simulation for Defense Systems and Applications V*. doi:10.1117/12.850538.

Kass R, Raftery A (1995). "Bayes Factors." *Journal of the American Statistical Association*, **90**(430), 773–795. doi:10.1080/01621459.1995.10476572.

Liang F, Wong W (2000). "Evolutionary Monte Carlo: Application to Cp Model Sampling and Change Point Problem." *Statistica Sinica*, **10**, 317–342.

Liquet B, Chadeau-Hyam M (2014). ***R2GUESS**: Wrapper Functions for **GUESS***. R package version 1.4, URL http://CRAN.R-project.org/package=R2GUESS.

Maruyama Y, George E (2011). "Fully Bayes Factors with a Generalized g-Prior." *The Annals of Statistics*, **39**(5), 2740–2765. doi:10.1214/11-aos917.

Monni S, Tadesse M (2009). "A Stochastic Partitioning Method to Associate High-Dimensional Responses and Covariates." *Bayesian Analysis*, **4**(3), 413–436. doi:10.1214/09-ba416.

NVIDIA Corporation (2015). ***CUDA**: Compute Unified Device Architecture*. URL http://www.nvidia.com/object/cuda_home_new.html.

Petretto E, Bottolo L, Langley SR, Heinig M, McDermott-Roe C, Sarwar R, Pravenec M, Hübner N, Aitman TJ, Cook SA, Richardson S (2010). "New Insights into the Genetic Control of Gene Expression Using a Bayesian Multi-Tissue Approach." *PLOS Computational Biology*, **6**(4), e1000737. doi:10.1371/journal.pcbi.1000737.

R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Zeller T, Wild P, Szymczak S, Rotival M, Schillert A, é RC, Maouche S, Germain M, Lackner K, Rossmann H, Eleftheriadis M, Sinning CR, Schnabel RB, Lubos E, Mennerich D, Rust W, Perret C, Proust C, Nicaud V, Loscalzo J, Hübner N, Tregouet D, Münzel T, Ziegler A, Tiret L, Blankenberg S, Cambien F (2010). "Genetics and Beyond – The Transcriptome of Human Monocytes and Disease Susceptibility." *PLOS ONE*, **5**(5), e10693. doi:10.1371/journal.pone.0010693.

Zellner A (1986). "On Assessing Prior Distributions and Bayesian Regression Analysis with g-Prior Distributions." In P Geol, A Zellner (eds.), *Bayesian Inference and Decision Techniques: Essays in Honor of Bruno De Finetti*, volume 6 of *Studies in Bayesian Econometrics and Statistics*, pp. 233–243. Elsevier.

Affiliation:

Marc Chadeau-Hyam
Department of Epidemiology and Biostatistics
Imperial College London
St Mary's Hospital
Norfolk Place
London, W21PG, United Kingdom
E-mail: m.chadeau@imperial.ac.uk

Benoît Liquet
Laboratoire de Mathématiques et de leurs Applications
Université de Pau et des Pays de l'Adour
UMR CNRS 5142, Pau, France
and
ARC Centre of Excellence for Mathematical and Statistical Frontiers
Queensland University of Technology (QUT)
Brisbane, Australia
E-mail: benoit.liquet@pau-univ.fr

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
January 2016, Volume 69, Issue 2 *Submitted:* 2014-01-21
doi:10.18637/jss.v069.i02 *Accepted:* 2015-01-23